特性

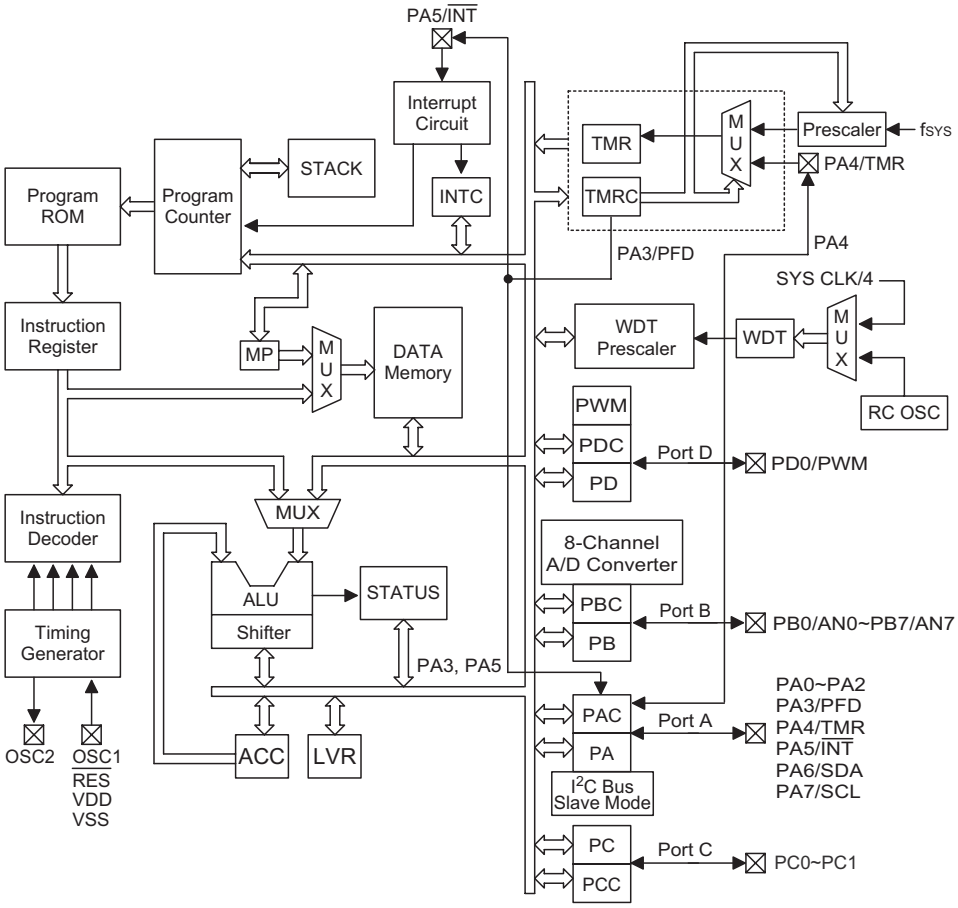
- 工作电压：
  f_{SYS}=4MHz：2.2V~5.5V
  f_{SYS}=8MHz：3.3V~5.5V
- 最多可有 19 个双向输入/输出口
- 1 个与输入/输出口共用引脚的外部中断输入
- 8 位可编程定时/计数器，具有溢出中断和 7 级预分频器
- 内置晶体和 RC 振荡电路
- 看门狗定时器
- 2048×14 程序存储器 ROM
- 64×8 数据存储器 RAM
- 具有 PFD 功能，可用于发声
- HALT 和唤醒功能可降低功耗
- 在 V_{DD}=5V，系统频率为 8MHz 时，指令周期为 0.5μs
- 6 层硬件堆栈
- 8 通道 9 位解析度（8 位精度）的 A/D 转换器
- 1 通道（6+2）/（7+1）位的 PWM 输出，与输入/输出口共用引脚
- 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
- I^2C 总线（slave 模式）
- 24-pin SKDIP/SOP 封装

概述

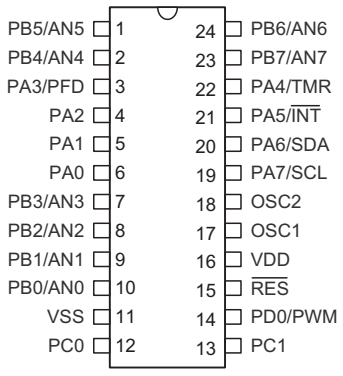
HT46R22/HT46C22 是 8 位高性能精简指令集单片机，专门为需要 A/D 转换的产品而设计，例如传感器信号输入。掩膜版本 HT46C22 与 OTP 版本 HT46R22 引脚和功能完全相同。低功耗、I/O 使用灵活、可编程分频器、计数器、振荡类型选择、多通道 A/D 转换、脉冲测量功能、I^2C 通信、暂停和唤醒功能，使这款单片机可以广泛应用于带传感器的 A/D 转换、马达控制、工业控制、消费类产品等系统中。

I^2C 为 Philips Semiconductors 的商标

方框图



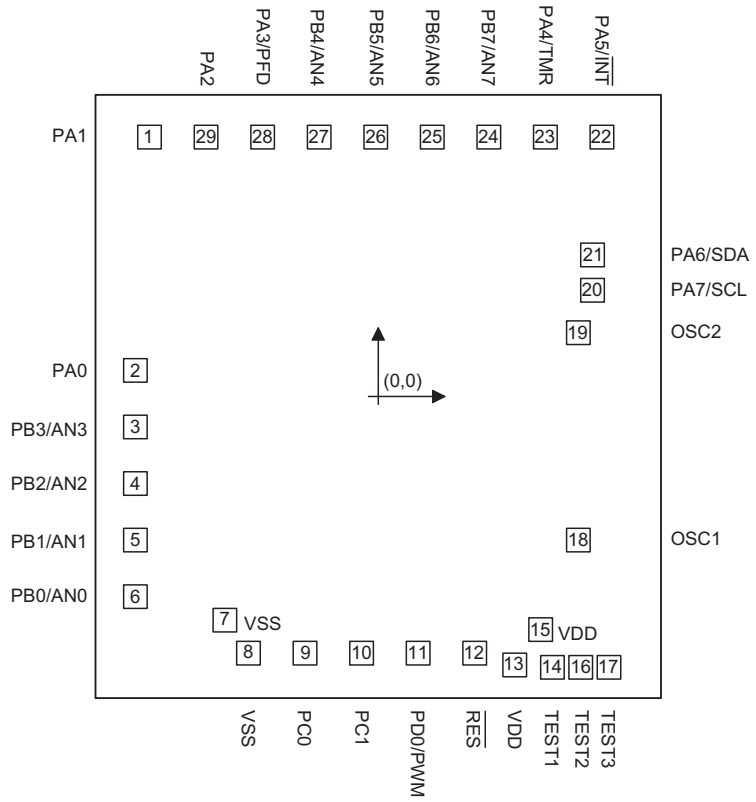
引脚图



HT46R22/HT46C22
－ 24 SKDIP-A/SOP-A

Pad 图

HT46C22



*IC 的衬底要连接到 PCB 板上的 VSS

引脚说明

引脚名称	输入/输出	掩膜选项	说明
PB0/AN0 PB1/AN1 PB2/AN2 PB3/AN3 PB4/AN4 PB5/AN5 PB6/AN6 PB7/AN7	输入/输出	上拉电阻	8 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、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。
PA0~PA2 PA3/PFD PA4/TMR PA5/$\overline{\text{INT}}$ PA6/SDA PA7/SCL	输入/输出	上拉电阻 唤醒功能 PA3 或 PFD I/O 或 I^2C	8 位双向输入/输出口。每一位可由掩模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。PFD、TMR、$\overline{\text{INT}}$ 分别与 PA3、PA4、PA5 共用引脚。一旦使用 I^2C 总线功能，则与 PA6、PA7 相关的寄存器将被禁用。
VSS	—	—	负电源，接地。
PC0~PC1	输入/输出	上拉电阻	2 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。
PD0/PWM	输入/输出	上拉电阻 I/O 或 PWM	1 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。PWM 输出与 PD0 共用引脚（由 PWM 选项决定）。

$\overline{RES}$	输入	—	斯密特触发复位输入，低电平有效。
VDD	—	—	正电源。
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。
TEST1 TEST2 TEST3	输入	—	测试模式下输入引脚，正常使用时不必连接。

极限参数

电源供应电压……V_{SS}-0.3V~V_{SS}+6.0V　　储存温度……-50℃~125℃

端口输入电压……V_{SS}-0.3V~V_{DD}+0.3V　　工作温度……..….-40℃~85℃

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

直流电气特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	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 ADC 关闭	—	0.6	1.5	mA
		5V		—	2	4	mA
I_{DD2}	工作电流（RC 振荡）	3V	无负载，f_{SYS}=4MHz ADC 关闭	—	0.8	1.5	mA
		5V		—	2.5	4	mA
I_{DD3}	工作电流	5V	无负载，f_{SYS}=8MHz ADC 关闭	—	3	5	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/输出口、TMR、$\overline{INT}$ 的低电平输入电压	3V	—	0	—	0.3 V_{DD}	V
		5V	—	0	—	0.3V_{DD}	V
V_{IH1}	输入/输出口、TMR、$\overline{INT}$ 的高电平输入电压	3V	—	0.7V_{DD}	—	V_{DD}	V
		5V	—	0.7V_{DD}	—	V_{DD}	V
V_{IL2}	低电平输入电压($\overline{RES}$)	3V	—	0	—	0.4 V_{DD}	V
		5V	—	0	—	0.4 V_{DD}	V
V_{IH2}	高电平输入电压($\overline{RES}$)	3V	—	0.9 V_{DD}	—	V_{DD}	V
		5V	—	0.9 V_{DD}	—	V_{DD}	V
V_{LVR}	低电压复位	—	—	2.7	3	3.3	V
I_{OL}	输入/输出口灌电流	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V	V_{OL}=0.1V_{DD}	10	20	—	mA
I_{OH}	输入/输出口源电流	3V	V_{OH}=0.9V_{DD}	-2	-4	—	mA
		5V	V_{OH}=0.9V_{DD}	-5	-10	—	mA
R_{PH}	上拉电阻	3V	—	40	60	80	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}	ADC 打开，其它关闭	3V	无负载	—	0.5	1	mA
		5V		—	1.5	3	mA

交流电气特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{SYS}	系统时钟	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f_{TIMER}	定时器输入频率（TMR）	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
t_{WDTOSC}	看门狗振荡器	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	—	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	5V	—	1	—	—	μs
t_{ADC}	A/D 转换时间	—	—	—	76	—	t_{AD}
t_{ADCS}	A/D 采样时间	—	—	—	32	—	t_{AD}
t_{IIC}	I^2C 总线时钟周期	—	外接 2kΩ上拉电阻	64	—	—	$*t_{SYS}$

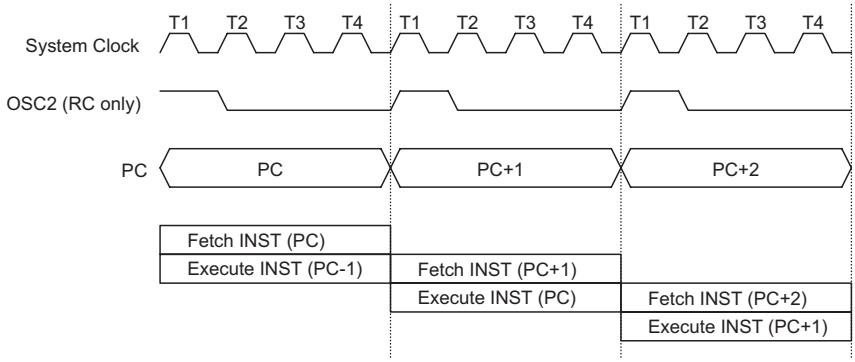
注：$*t_{SYS}=1/f_{SYS}$

系统功能说明

指令执行时序

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

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



指令执行时序

程序计数器 — PC

程序计数器(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
外部中断	0	0	0	0	0	0	0	0	1	0	0
定时/计数器溢出	0	0	0	0	0	0	0	1	0	0	0
A/D 转换中断	0	0	0	0	0	0	0	1	1	0	0
I^2C 总线中断	0	0	0	0	0	0	1	0	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　　S10～S0 ： 堆栈寄存器位

#10～#0 ： 指令代码位　　@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

该地址为外部中断服务程序保留。当$\overline{INT}$引脚有触发信号输入，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

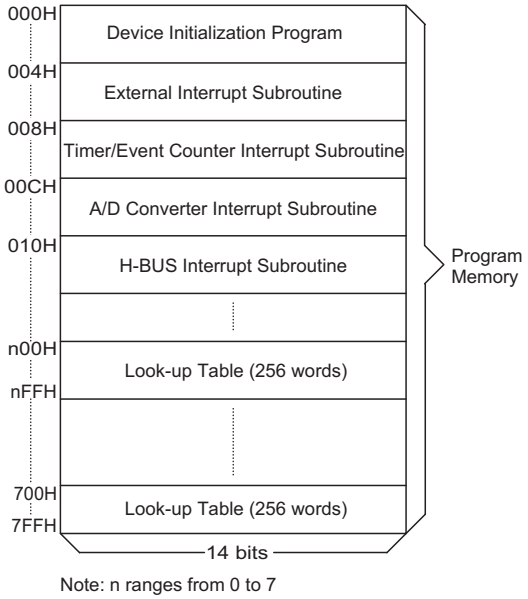
该地址为定时/计数器中断服务程序保留。当定时/计数器溢出，如果中断允许且堆栈未满，则程序会跳转到 008H 地址开始执行。

- 地址 00CH

该地址为 A/D 转换中断服务程序保留。当 A/D 转换完成，如果中断允许且堆栈未满，则程序会跳转到 00CH 地址开始执行。

- 地址 010H

该地址为 I^2C 总线中断服务程序保留。当 I^2C 总线从器件地址匹配或收到一个完整的数据，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。



Note: n ranges from 0 to 7

- 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。

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：表格地址字节　　P10~P8：当前程序指针字节　　@7~@0：表格指针字节

堆栈寄存器 — STACK

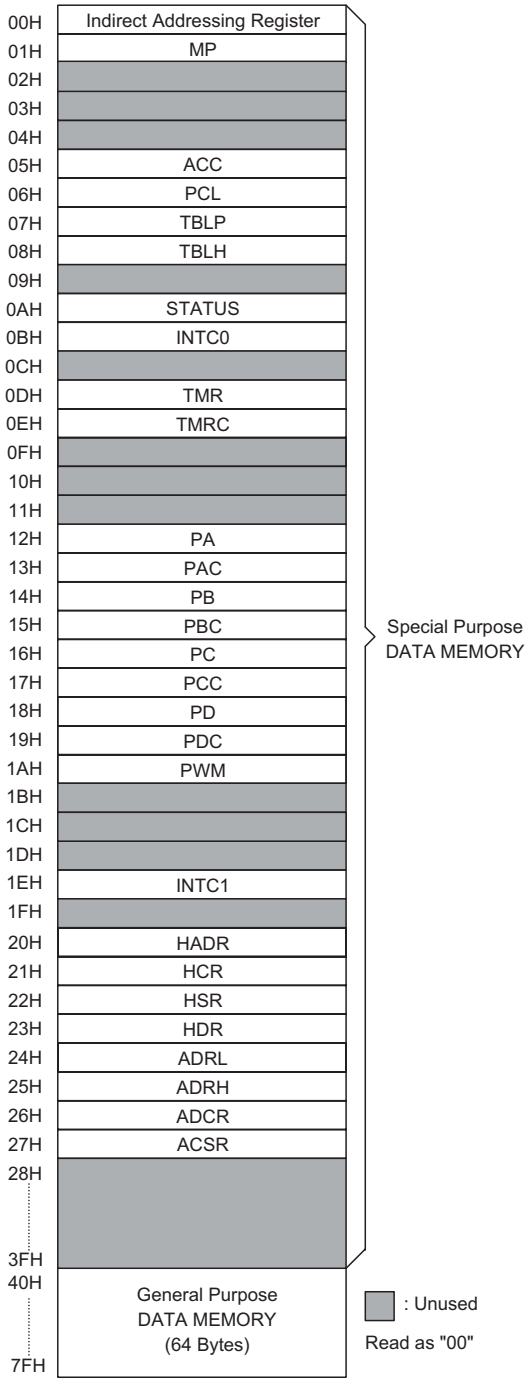
堆栈寄存器是特殊的存储器空间，用来保存 PC 的值。HT46x22 有 6 层堆栈，堆栈寄存器既不是数据存储器的一部分，也不是程序存储器的一部分，而且它既不能读出，也不能写入。堆栈的使用是通过堆栈指针（SP）来实现的，堆栈指针也不能读出或写入。当发生子程序调用或中断响应时，程序计数器（PC）的值会被压入堆栈；在子程序调用结束或中断响应结束时（执行指令 RET 或 RETI），堆栈将原先压入堆栈的内容弹出，重新装入程序计数器中。在系统复位后，堆栈指针会指向堆栈顶部。

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 6 个返回地址会被保留。

数据存储器 — RAM

数据存储器由 92×8 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（64×8），数据存储器单元大多数是可读/写的，但有些只读的。

特殊功能寄存器包括间接寻址寄存器（00H），间接寻址指针寄存器（MP；01H），累加器（ACC；05H），程序计数器低字节寄存器（PCL；06H），表格指针寄存器（TBLP；07H），表格内容高字节寄存器（TBLH；08H），状态寄存器（STATUS；0AH），中断控制寄存器 0（INTC0；0BH），定时/计数器(TMR;0DH)，定时/计数器控制寄存器(TMRC；0EH），输入/输出寄存器（PA；12H，PB；14H，PC；16H，PD；18H），输入/输出控制寄存器（PAC；13H，PBC；15H，PCC；17H，PDC；19H），PWM 数据寄存器（PWM；1AH），中断控制寄存器 1（INTC1；1EH），I^2C 总线从器件地址寄存器（HADR；20H），I^2C 总线控制寄存器（HCR；21H），I^2C 总线状态寄存器（HSR；22H），I^2C 总线数据寄存器（HDR；23H），A/D 转换结果低字节寄存器(ADRL；24H），A/D 转换结果高字节寄存器（ADRH；25H），A/D 转换控制寄存器(ADCR；26H），A/D 时钟设置寄存器(ACSR；27H）。其余在 40H 之前的空间保留给系统以后扩展使用，读取这些地址的返回值为“00H”。通用数据寄存器地址从 40H 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由“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 MP 进行间接寻址。

间接寻址寄存器

地址 00H 是一个间接寻址寄存器，并无实际的物理区存在。任何对[00H]的读/写操作，都是访问由 MP（01H）所指向的 RAM 单元。间接读取此地址得到的值为 00H，间接写入此地址，不会产生任何操作。

间接寻址指针 MP（01H）是一个 7 位寄存器。MP 的最高位未定义，读取该位得到的结果是“1”。任何对 MP 的写操作只能将低 7 位数据写入 MP 中。

累加器

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

算术逻辑单元 — ALU

算术逻辑单元（ALU）是执行 8 位算术、逻辑运算的电路，它提供有以下功能：

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

ALU 不仅可以储存数据运算的结果，还会改变状态寄存器的值。

状态寄存器 — STATUS

8 位的状态寄存器（0AH），由零标志位（Z）、进位标志位（C）、辅助进位标志位（AC）、溢出标志位（OV）、暂停标志位（PDF）和看门狗定时器溢出标志位（TO）组成。该寄存器不仅记录状态信息，而且还控制操作顺序。

符号	位	功能
C	0	如果在加法运算中结果产生了进位或在减法运算中结果不产生借位，则 C 被置位；反之，C 被清除。它也可被循环移位指令影响。
AC	1	如果在加法运算中低 4 位产生了进位或减法运算中低 4 位不产生借位，则 AC 被置位；反之，AC 被清除。
Z	2	如果算术或逻辑运算的结果为零，则 Z 被置位；反之，Z 被清除。
OV	3	如果运算结果向最高位进位，但最高位并不产生进位输出，则 OV 被置位；反之，OV 被清除
PDF	4	系统上电或执行“CLR WDT”指令，PDF 被清除；执行“HALT”指令，PDF 被置位。
TO	5	系统上电、执行“CLR WDT”或“HALT”指令，TO 被清除；WDT 定时溢出，TO 被置位。
—	6	未用，读出为“0”
—	7	未用，读出为“0”

状态寄存器

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

标志位 Z、OV、AC 和 C 反映的是最近一次操作的状态。

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

中断

HT46x22 提供一个外部中断、一个内部定时/计数器中断、一个 A/D 转换中断和一个 I^2C 总线中断。中断控制寄存器 0（INTC0；0BH）和中断控制寄存器 1（INTC1；1EH）包含了中断控制位和中断请求标志，中断控制位用来设置中断允许/禁止。

寄存器	位	符号	功　　能
INTC0（0BH）	0	EMI	总中断控制位（1=允许；0=禁止）
	1	EEI	外部中断控制位（1=允许；0=禁止）
	2	ETI	定时/计数器中断控制位（1=允许；0=禁止）
	3	EADI	A/D 转换中断控制位（1=允许；0=禁止）
	4	EIF	外部中断请求标志（1=有；0=无）
	5	TF	定时/计数器中断请求标志（1=有；0=无）
	6	ADF	A/D 转换中断请求标志（1=有；0=无）
	7	—	未用，读出为“0”

INTC0 寄存器

寄存器	位	符号	功　　能
INTC1（1EH）	0	EHI	I^2C 总线中断控制位（1=允许；0=禁止）
	1	—	未用，读出为“0”
	2	—	未用，读出为“0”
	3	—	未用，读出为“0”
	4	HIF	I^2C 总线中断请求标志（1=有；0=无）
	5	—	未用，读出为“0”
	6	—	未用，读出为“0”
	7	—	未用，读出为“0”

INTC1 寄存器

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

所有的中断都具有唤醒能力。当有中断被服务，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。

外部中断是由 $\overline{\text{INT}}$ 引脚下降沿信号触发的，其中断请求标志位（EIF；INTC0 的第 4 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 04H 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/计数器中断是由定时/计数器溢出触发的，其中断请求标志（TF；INTC0 的第 5 位）会被置位。如果中断允许，且堆栈未满，当发生定时/计数器中断时，会产生地址 08H 的子程序调用；而中断请求标志 TF 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

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；INTC0 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 0CH 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。

I^2C 总线中断是由从器件地址匹配（HAAS="1"）或接收到完整的数据触发的，其中断请求标志位（HIF；INTC1 的第 4 位）会被置位。如果中断允许，且堆栈未满，当发生 I^2C 总线中断时，会产生地址 10H 的子程序调用；而中断请求标志位 HIF 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。

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

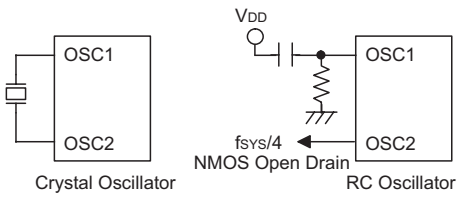
No.	中断源	优先级	中断向量
a	外部中断	1	04H
b	定时/计数器中断	2	08H
c	A/D 转换中断	3	0CH
d	I^2C 总线中断	4	10H

中断控制寄存器 0（INTC0），由定时/计数器中断请求标志（TF）、外部中断请求标志（EIF）、A/D 转换中断请求标志（ADF）、定时/计数器中断允许（ETI）、外部中断允许（EEI）、A/D 转换中断允许（EADI）和总中断允许（EMI）组成，其对应于数据存储器地址 0BH。中断控制寄存器 1（INTC1），由 I^2C 总线中断请求标志（HIF）和 I^2C 总线中断允许（EHI）组成，其对应于数据存储器地址 1EH。EMI、EEI、ETI、EADI 和 EHI 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（TF、EIF、ADF、HIF）被置位，会一直保留在 INTC0 和 INTC1 寄存器中，直到中断被响应或用软件指令清除为止。

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

振荡电路

HT46x22 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。



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 30kΩ到 750kΩ；而 OSC2 上会输出系统频率的 4 分频信号，可用于同步外部逻辑。RC 振荡方式是一

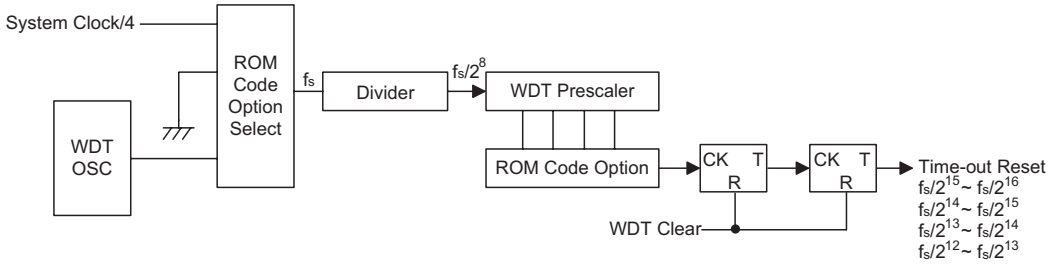
种低成本的方案，但是，RC 振荡频率会随着 VDD、温度和芯片自身参数的漂移而产生误差。因此，在需要精确振荡频率做为计时操作的场合，并不适合使用 RC 振荡方式。

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）。

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 — WDT

看门狗定时器的时钟来源有两种：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

如果 WDT 时钟源为内部 WDT 振荡器输出(RC 振荡周期一般为 65μs)，该频率可再经过 2^{12}~2^{15} 的分频（由掩膜选项决定）。最短的 WDT 溢出周期大约是 300mS~600mS。溢出周期会因为温度、VDD 以及芯片参数的漂移而变化。通过掩膜选项设置，可以得到更长的溢出周期，如果选择 2^{15}，则分频系数为 2^{15}~2^{16}，最长的溢出周期可达到 2.1s～4.3s。

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），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$\overline{\text{RES}}$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。

如果 WDT 的分频系数选择 $f_S/2^{12}$（由掩膜选项决定），则 WDT 的溢出周期为 $f_S/2^{12}$～$f_S/2^{13}$，因为“CLR WDT”和“CLR WDT1”、“CLR WDT2”指令只能清除最后两级 WDT 分频器。

暂停模式 — HALT

暂停模式是由 HALT 指令来实现的，暂停模式时系统状态如下：

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

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

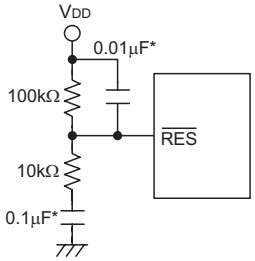
为减小功耗，在进入暂停模式之前，应小心处理所有的输入/输出口状态。

复位

总共有三种方法会产生初始复位：

- 正常运行时由$\overline{RES}$引脚发生复位。
- 在暂停模式由$\overline{RES}$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：“*”连线应该尽量靠近$\overline{RES}$引脚，以减小干扰影响

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

注：“u”表示不变

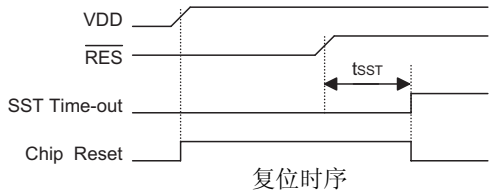
为了保证系统振荡器起振并稳定运行，系统复位（包括上电复位、WDT 溢出或由$\overline{RES}$端复位）或由暂停状态唤醒时，系统启动定时器（SST）提供了一个额外的延迟时间，共 1024 个系统时钟周期。

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

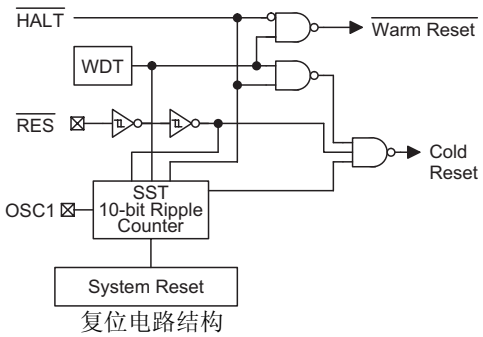
系统复位（包括上电复位、正常运行时 WDT 溢出或由$\overline{RES}$端复位）需要额外增加一个加载掩膜选项（Option）的时间。

系统复位时各功能单元的状态如下所示：

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



复位时序



复位电路结构

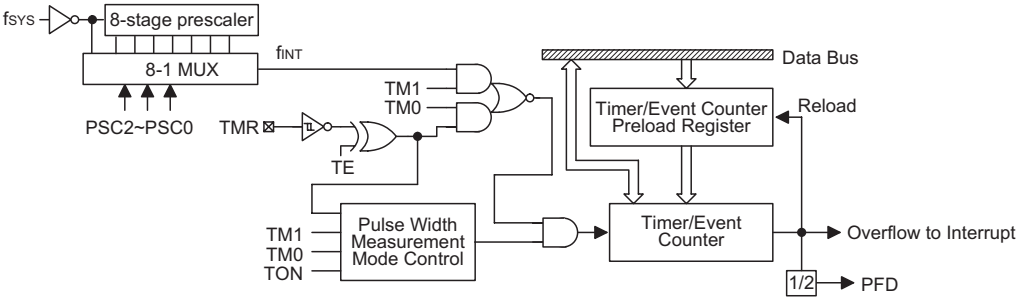
有关寄存器的状态如下

寄存器	复位 (上电复位)	WDT 溢出 (正常运行)	$\overline{RES}$复位 (正常运行)	$\overline{RES}$复位 (暂停模式)	WDT 溢出 (暂停模式)*
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
PC	000H	000H	000H	000H	000H
MP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--lu uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	---0---0	---0---0	---0---0	---0---0	---u---u
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	---- --11	---- --11	---- --11	---- --11	---- --uu
PCC	---- --11	---- --11	---- --11	---- --11	---- --uu
PD	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u
PDC	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u
PWM	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
HADR	xxxx xxx-	xxxx xxx-	xxxx xxx-	xxxx xxx-	uuuu uuu-
HCR	0--0 0---	0--0 0---	0--0 0---	0--0 0---	u--u u---
HSR	100- -0-1	100- -0-1	100- -0-1	100- -0-1	uuu- -u-u
HDR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu 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	1--- --00	1--- --00	1--- --00	1--- --00	u--- --uu

注： “*”表示“热复位” “u”表示“不变” “x”表示“未知”

定时/计数器

HT46x22 有一个 8 位可编程向上计数的定时/计数器。定时/计数器的时钟来源可以是外部信号输入或系统时钟。



定时/计数器

如果用做内部定时器方式，则只有一个时钟来源，即系统时钟 f_{SYS}。外部信号输入可以用来计数外部事件、测量时间间隔、测量脉冲宽度或产生一个精确的时基信号。

有两个与定时/计数器有关的寄存器，TMR（[0DH]）和 TMRC（[0EH]）。TMR 寄存器有两个物理空间；写入 TMR 会将初始值装入到定时/计数器的预置寄存器中，而读 TMR 则会取得定时/计数器的内容。TMRC 是定时/计数器控制寄存器，用来定义定时/计数器一些选项。

符号(TMRC)	位	功能
PSC0~PSC2	0~2	定义预分频器级数，PSC2，PSC1，PSC0= 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：$f_{INT}=f_{SYS}/128$
TE	3	定义定时/计数器 TMR 的触发方式 （0=上升沿作用，1=下降沿作用）
TON	4	打开/关闭定时/计数器（1=打开，0=关闭）
—	5	未用，读出为“0”
TM0 TM1	6 7	定义工作模式： 01=事件计数模式（外部时钟） 10=定时模式（内部时钟） 11=脉冲宽度测量模式 00 =未用

TMRC 寄存器

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

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

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

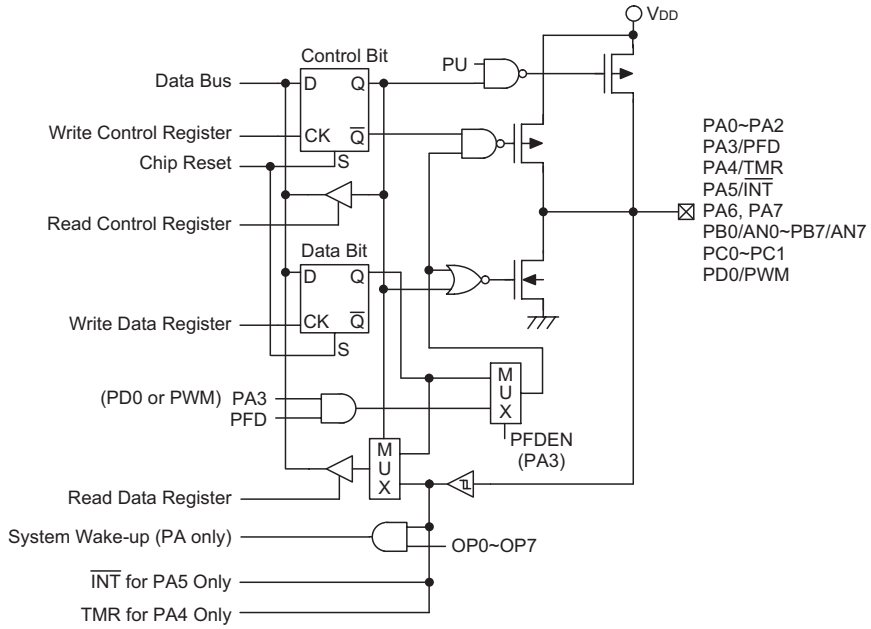
只能由指令来清除。定时/计数器溢出可以做为唤醒信号。不管是什么模式，只要写 0 到 ETI 即可禁止定时/计数器中断服务。

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

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

输入/输出口

HT46x22 有 19 个双向输入/输出口，记为 PA、PB、PC 和 PD，其分别对应 RAM 地址[12H]、[14H]、[16H]和[18H]。所有端口都可以进行输入/输出操作。输入时，端口没有锁存功能，输入信号必须在 MOV A，[m]（m=12H、14H、16H 或 18H）指令的 T2 上升沿到来前准备好；输出时，端口有锁存功能，端口上的数据会保持不变直到执行下一个写入操作。



输入/输出口

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

做为输出时，只能采用 CMOS 输出。控制寄存器对应 RAM 地址 13H、15H、17H、19H。

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

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

PA 的每一个口都具有唤醒系统的能力。PC 的高 6 位和 PD 的高 7 位没有实际的物理结构；读取这些位会返回“0”，而写入则是一个空操作。

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，必须注意在输入模式下，输入/输出口会产生浮空状态。

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

PA5、PA4 口分别与 $\overline{INT}$、TMR 共用引脚。

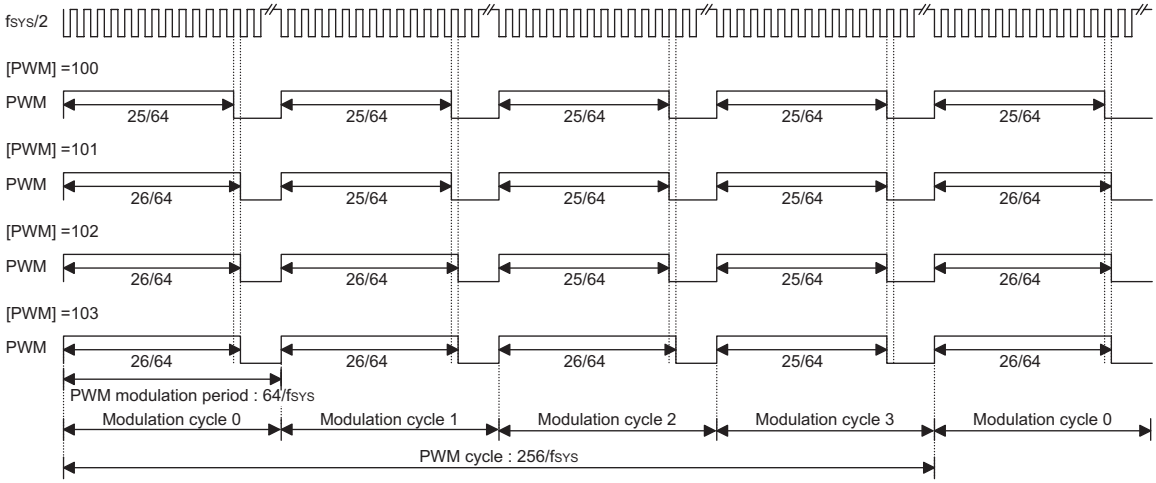
PB 口可以用做 A/D 转换输入，A/D 转换功能将在下面说明。PD0 与 PWM 共用引脚。如果选择 PWM 功能，则 PD0 口会有 PWM 信号输出（PD0 为输出模式）。向 PD0 数据寄存器写入“1”，则输出 PWM 信号；向 PD0 数据寄存器写入“0”，则 PD0 输出为“0”。PD0 的 I/O 输入/输出如下所示：

I/O 模式	I/P（正常）	O/P（正常）	I/P（PWM）	O/P（PWM）
PD0	逻辑输入	逻辑输出	逻辑输入	PWM

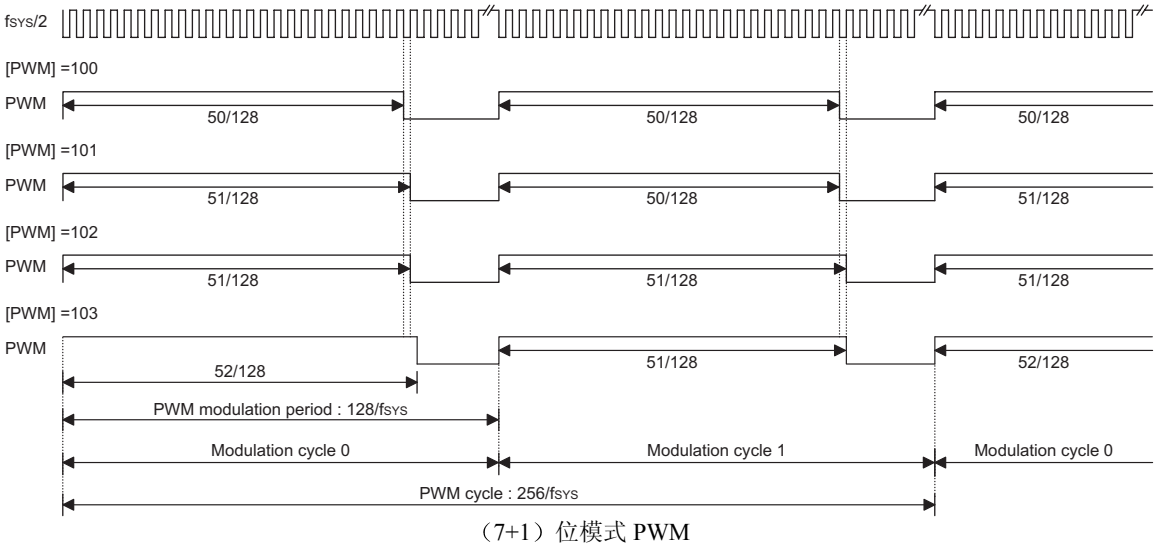
建议用软件将未使用和没有外接的输入/输出口设置为输出模式，以防止这些端口在输入浮空时增加系统的功耗。

PWM

HT46x22 有 1 个通道（6+2）/（7+1）位的 PWM 输出（由掩膜选项决定），与 PD0 共用引脚。PWM 通道由一个数据寄存器 PWM（1AH）来控制输出。PWM 计数器的时钟来源为系统时钟（f_{SYS}）。PWM 寄存器是一个 8 位的寄存器。PWM 的输出波形如图所示。一旦 PD0 选择为 PWM 输出，并且 PD0 为输出模式（PDC.0=“0”），则向 PD0 寄存器写“1”能够产生 PWM 输出，向 PD0 寄存器写“0”会使 PD0 输出保持为“0”。



（6+2）位模式 PWM



（7+1）位模式 PWM

在（6+2）位 PWM 模式中，一个 PWM 周期被分为四个调制周期（调制周期 0~调制周期 3），每个调制周期有 64 个 PWM 输入时钟。在（6+2）位 PWM 模式中，PWM 寄存器被分为 2 个部分。第一部分是直流分量，由 PWM.7~PWM.2 控制；第二部分是交流分量，由 PWM.1~PWM.0 控制。

在（6+2）位 PWM 模式中，每个调制周期的占空比见下表：

参数	AC（0~3）	占空比
调制周期 i （i=0~3）	i<AC	$\frac{DC+1}{64}$
	i≥AC	$\frac{DC}{64}$

在（7+1）位 PWM 模式中，一个 PWM 周期被分为两个调制周期（调制周期 0~调制周期 1），每个调制周期有 128 个 PWM 输入时钟。在（7+1）位 PWM 模式中，PWM 寄存器被分为 2 个部分。第一部分是直流分量，由 PWM.7~PWM.1 控制；第二部分是交流分量，由 PWM.0 控制。

在（7+1）位 PWM 模式中，每个调制周期的占空比见下表：

参数	AC（0~1）	占空比
调制周期 i （i=0~1）	i<AC	$\frac{DC+1}{128}$
	i≥AC	$\frac{DC}{128}$

PWM 的调制频率、周期频率和占空比的关系总结如下：

PWM 调制频率	PWM 周期频率	PWM 占空比
$f_{SYS}/64$（6+2 模式） $f_{SYS}/128$（7+1 模式）	$f_{SYS}/256$	[PWM]/256

A/D 转换

HT46x22 有 8 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，$\overline{EOC}$ 位会被清除，并且产生 A/D 转换中断（如果 A/D 转换允许）。ACSR 是 A/D 时钟控制寄存器，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 寄存器

ACS2	ACS1	ACS0	模拟通道
0	0	0	A0
0	0	1	A1
0	1	0	A2
0	1	1	A3
1	0	0	A4
1	0	1	A5
1	1	0	A6
1	1	1	A7

模拟输入通道选择

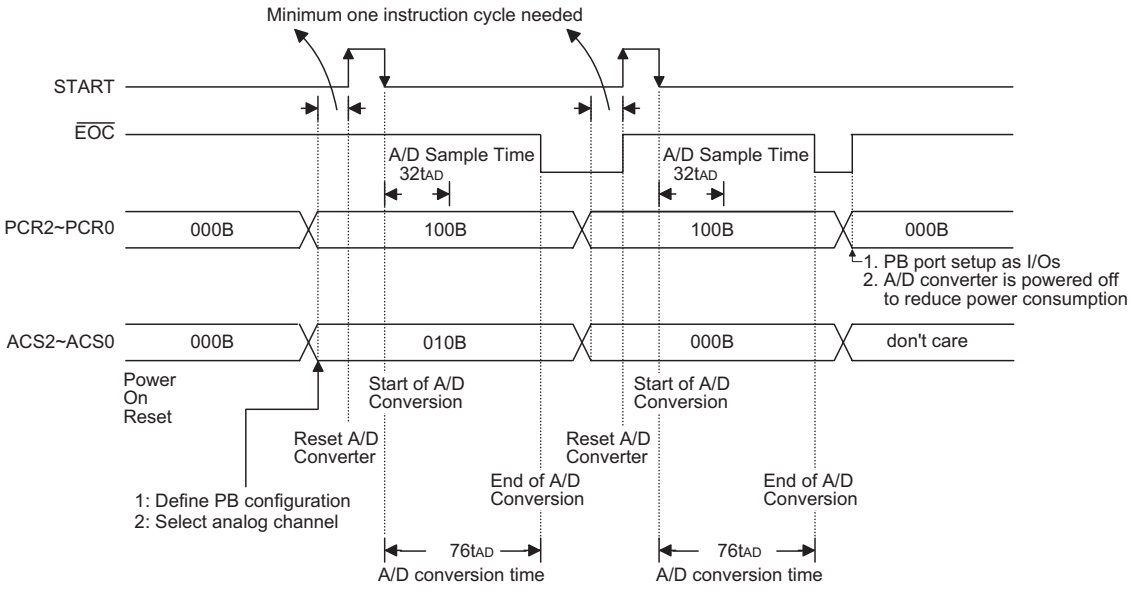
PCR2	PCR1	PCR0	7	6	5	4	3	2	1	0
0	0	0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0	0	1	PB7	PB6	PB5	PB4	PB3	PB2	PB1	A0
0	1	0	PB7	PB6	PB5	PB4	PB3	PB2	A1	A0
0	1	1	PB7	PB6	PB5	PB4	PB3	A2	A1	A0
1	0	0	PB7	PB6	PB5	PB4	A3	A2	A1	A0
1	0	1	PB7	PB6	PB5	A4	A3	A2	A1	A0
1	1	0	PB7	PB6	A5	A4	A3	A2	A1	A0
1	1	1	A7	A6	A5	A4	A3	A2	A1	A0

PB 口的设置

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 8 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。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。$\overline{EOC}$ 位（ADCR 的第 6 位）是 A/D 转换结束标志位。通过检测这个标志位可以知道 A/D 转换是否结束。ADCR 的 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 $\overline{EOC}$ 位变为“0”（A/D 转换完成信号）。

符号(ADCR)	位	功能
ACS0 ACS1 ACS2	0 1 2	选择模拟输入通道
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
$\overline{EOC}$	6	A/D 转换结束标志（0：A/D 转换结束）
START	7	A/D 转换启始控制位 0→1→0：开始； 0→1：A/D 转换复位

ADCR 寄存器



Note: ADC clock must be fSYS/2, fSYS/8 or fSYS/32

A/D 转换时序

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 中断标志被置位。当 START 标志由“0”置为“1”时，$\overline{EOC}$ 也置为“1”。

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

注：D0~D8 是 A/D 转换结果的低位~高位

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

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

```
clr INTC0.3                ; 在中断控制寄存器中禁止A/D中断
mov a,00100000B
mov ADCR,a                 ; 在ADCR寄存器中设置Port PB0~PB3做为A/D输入
                           ; 设置AN0进行A/D转换
mov a,00000001B
mov ACSR,a                 ; 设置ACSR寄存器，选择fSYS/8做为A/D转换时钟
Start_conversion:
clr ADCR.7
set ADCR.7                 ; A/D转换复位
clr ADCR.7                 ; 开始A/D转换
Polling_EOC:
sz ADCR.6                  ; 扫描ADCR寄存器的EOC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。

```
    set INTC0.3            ; 在中断控制寄存器中设置A/D中断允许
    mov a,00100000B
    mov ADCR,a             ; 在ADCR寄存器中设置Port PB0~PB3做为A/D输入
                           ; 设置AN0进行A/D转换
    mov a,00000001B
    mov ACSR,a             ; 设置ACSR寄存器，选择fSYS/8做为A/D转换时钟
start_conversion:
    clr .7
    set ADCR.7             ; A/D转换复位
    clr ADCR.7             ; 开始A/D转换
        :
        :
; 中断服务子程序
```

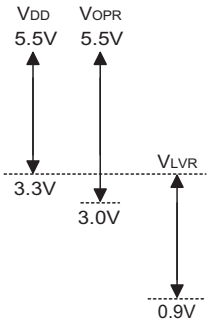
```
EOC_service routine:
    mov a_buffer,a          ; 将ACC保存到用户定义的寄存器中
    mov a,ADRH                  ; 从ADRH寄存器读取A/D转换结果的高位字节
    mov adrh_buffer,a       ; 将结果放入用户定义的寄存器中
    mov a,ADRL              ; 从ADRL寄存器读取A/D转换结果的低位字节
    mov adrl_buffer,a       ; 将结果放入用户定义的寄存器中
    clr ADCR.7
    set ADCR.7              ; A/D转换复位
    clr ADCR.7              ; 开始A/D转换
    mov a,a_buffer          ; 将ACC从暂存器中读出
    reti
```

低电压复位—LVR

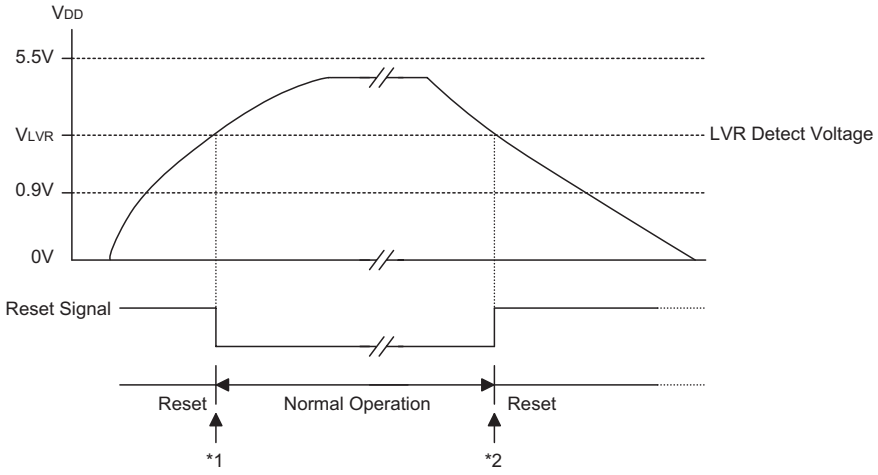
为了监控器件的工作电压，HT46x22 提供低电压复位功能。如果器件的工作电压在 0.9V～3.3V 之间，例如电池电压的变化，那么 LVR 会自动使器件产生内部复位。

LVR 功能说明如下：

- 低电压（0.9V～3.3V）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。

I2C 总线接口

HT46x22 提供 I^2C 总线接口，I^2C 总线是双向两线结构，数据线和时钟线分别为 SDA 和 SCL。SDA 和 SCL 是 NMOS 开漏输出，使用时必须外接上拉电阻。

I^2C 总线有两种数据传输方式，一种为从器件发送模式（slave transmit mode），另一种为从器件接收模式（slave receive mode）。有四个与 I^2C 总线相关寄存器：HADR（[20H]）、HCR（[21H]）、HSR（[22H]）、HDR（[23H]）。HADR 寄存器用来存放从器件地址，如果主控制器发送的调用地址与该从器件地址匹配，则表明该器件被选中。HCR 是 I^2C 总线控制寄存器，用来控制 I^2C 总线功能的开/关状态和 I^2C 总线是工作于发送模式还是接收模式。HSR 是 I^2C 总线状态寄存器，该寄存器反映了 I^2C 总线的工作状态。HDR 是数据输入/输出寄存器，发送和接收的数据都必须经过 HDR 寄存器。

Bit7～Bit1	Bit0
从地址	—

HADR 寄存器

注：“—”为未定义

标号(HCR)	位	功能
HEN	7	打开/关闭I^2C总线（0=关闭；1=打开）
—	6	未用，读出为“0”
—	5	未用，读出为“0”
HTX	4	定义发送/接收模式（0=接收模式；1=发送模式）
TXAK	3	打开/关闭应答信号（0=应答；1=不应答）
—	0~2	未用，读出为“0”

HCR 寄存器

标号(HCR)	位	功能
HCF	7	HCF在数据开始传送时清“0”；HCF被置“1”表示8位数据已传送完毕。
HAAS	6	HAAS在调用地址匹配时被置“1”，同时发生I^2C 总线中断且HIF被置位。
HBB	5	HBB在I^2C总线忙时被置“1”；而当I^2C总线空闲时，该位将被清“0”。
—	4	未用，读出为“0”
—	3	未用，读出为“0”
SRW	2	当SRW被置“1”，表示主控制器要从I^2C总线读数据，从器件须将数据发送到主控制器；当SRW被清“0”，表示主控制器要写数据到I^2C总线，从器件必须从主控制器接收数据。
—	1	未用，读出为“0”
RXAK	0	当主控制器收到8位数据，并发出应答信号时，RXAK被清“0”；如果没有应答信号，则RXAK被置“1”。

HSR 寄存器

I^2C 总线控制寄存器包括 3 位：HEN、HTX 和 TXAK。HEN 控制 I^2C 总线打开/关闭，如果数据想通过 I^2C 总线传送，该位就必须置“1”。HTX 定义 I^2C 总线是用于发送模式还是接收模式，如果要用于发送模式，该位就必须置“1”。TXAK 定义应答信号，当器件接收到 8 位数据后，在第 9 个时钟时，器件将该位送到 I^2C 总线上，如果要继续接收下一个数据，在接收数据前该位必须清“0”。

I^2C 总线状态寄存器包括 5 位：HCF、HAAS、HBB、SRW 和 RXAK。HCF 在开始传送数据时被清“0”；在数据传送结束后被置“1”。HAAS 当从器件地址匹配时被置“1”，同时 I^2C 总线中断请求标志被置“1”；如果中断允许且堆栈未满，则程序会跳到地址 10H 开始执行。写数据到 I^2C 总线控制寄存器可以清除 HAAS；如果地址不匹配，HAAS 被清“0”。HBB 置“1”表示 I^2C 总线忙，即系统检测到“START”信号；HBB 清“0” 表示 I^2C 总线空闲，即系统检测到“STOP”信号，此时 I^2C 总线空闲。SRW 表示调用地址

匹配时，器件的读/写状态。当 HAAS 被置“1”，可以通过检测 SRW 来确定器件是工作在发送模式还是接收模式。当 SRW 被置“1”，表示主控制器要从 I^2C 总线读数据，从器件必须将数据写到 I^2C 总线，即从器件为发送模式；当 SRW 被清“0”，表示主控制器要写数据到 I^2C 总线，从器件要从总线读取数据，即从器件为接收模式。RXAK 清“0”，表示接收到一个应答信号。在发送模式，发送器件通过检测 RXAK 以确定接收器件是否要接收下一个数据，发送器件会一直写数据到 I^2C 总线直到 RXAK 置“1”，同时发送器件释放 SDA 线，这样主控制器可以发送 STOP 信号来释放总线。

HADR 寄存器的第 7~1 位定义从器件地址，开始数据传送时，主控制器通过发送从器件地址来选择器件；第 0 位没有用，不需要定义。一旦 I^2C 总线接上出现起始信号，所有的从器件都会接收连续的 8 位数据，该数据的前 7 位是从器件地址，高位在前，低位在后。如果从器件地址匹配，系统会置位 HAAS，同时产生 I^2C 总线中断。进入中断服务程序后，系统要检测 HAAS 位，以确定 I^2C 总线中断是来自从器件地址匹配，还是来自 8 位数据传送完毕。8 位数据的最后一位是读/写控制位，该位会反映到 SRW。从器件通过检测 SRW 以确定主控制器是要发送数据还是接收数据，并确定自己是作发送器还是接收器。

HDR 寄存器是 I^2C 总线的数据输入/输出寄存器。在发送数据前，要先将发送的数据写到 HDR 寄存器；在接收数据前，要先从 HDR 虚拟读取数据。从 I^2C 总线发送或接收数据都必须经过 HDR 寄存器。在 I^2C 总线开始传送数据前，需要先初始化 I^2C 总线，在初始化 I^2C 总线时，下面几点注意

1：向 I^2C 总线地址寄存器（HADR）写入从器件地址。

2：置位 I^2C 总线控制寄存器（HCR）的 HEN 位，以打开 I^2C 总线。

3：置位 I^2C 总线中断控制寄存器 1（INTC1）的 EHI 位，以允许 I^2C 总线中断。

起始信号

起始信号只能由主控制器产生。总线上的其它器件必须侦测起始信号以设定 I^2C 总线的忙标志位（HBB）。起始信号是指在 SCL 为高时，SDA 发生电平从高到低的变化。

从器件地址

发送起始信号后，主控制器必须发送从器件地址以选择要进行数据传输的从器件。所有在 I^2C 总线上的从器件都会接收到这个从器件地址（7 位），并与各自内部的从器件地址进行比较。如果从器件地址匹配，该从器件会产生一个中断，并将接下来的一位数据（即第 8 位）保存到 SRW 位，并发出一个应答信号。当从器件地址匹配时，还会置位状态标志（HAAS）。

在中断服务子程序中，通过检测 HAAS 位可以确定 I^2C 总线中断是来自从器件地址匹配，还是来自 8 位数据传送完毕。当是从器件地址匹配发生中断时，则器件必定是用于发送模式或接收模式，所以必须写数据到 HDR 或从 HDR 虚拟读取数据以释放 SCL 口线。

SRW 位

SRW 表示主控制器是要从 I^2C 总线上读取数据还是要将数据写到 I^2C 总线上。从器件则通过检测该位以确定自己是做为发送器还是接收器。SRW 置“1”，表示主控制器要从 I^2C 总线读数据，从器件必须将数据写到 I^2C 总线，即从器件做为发送器；SRW 清“0”，表示主控制器要写数据到 I^2C 总线，从器件要从总线读取数据，即从器件做为接收器。

应答位

当从器件地址匹配时，会发送一个应答信号。主控制器通过检测应答信号以判断从器件是否接接受了调用。如果没有应答信号，主控制器必须发送停止（STOP）信号以结束通讯。当 I^2C 总线状态寄存器的第 6 位（HAAS）是高时，表示地址匹配，则从器件需检查 SRW，以确定自己是做为发送器还是做为接收器。

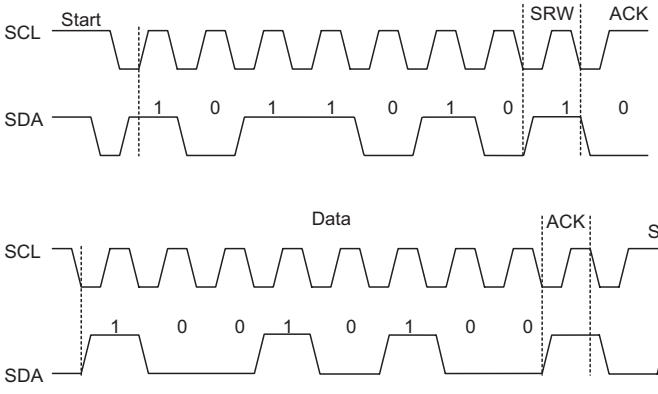
数据字节

在从器件发出应答信号后，就会进行数据传输，一个数据长度为 8 位，高位在前，低位在后。接收器

在接收到数据后会发出一个应答信号（“0”）以继续接收下一个数据。如果发送器没检测到应答信号，发送器将释放 SDA 线，同时，主控制器将发出 STOP 信号以释放 I^2C 总线。所传送的数据存储在 HDR 寄存器中，发送器在发送数据之前必须将数据写到 HDR；接收器在接收数据之后必须从 HDR 读取数据。

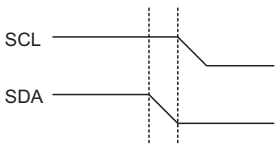
接收应答位

当接收器想要继续接收下一个数据时，必须在第 9 个时钟发出应答信号（TXAK）。发送器检测应答信号（RXAK）以是继续写数据到 I^2C 总线，还是改变为接收模式并虚读 HDR 寄存器以释放 SDA 线，同时主控制器发出停止信号。

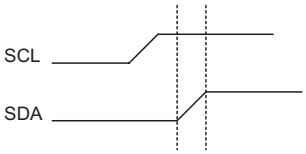


Note: S= Start (1 bit)
SA= Slave Address (7 bits)
SR= SRW bit (1 bit)
M= Slave device send acknowledge bit (1 bit)
D= Data (8 bits)
A= ACK (RXAK bit for transmitter; TXAK bit for receiver 1 bit)
P= Stop (1 bit)

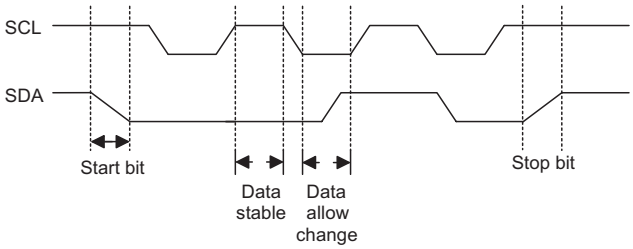
S	SA	SR	M	D	A	D	A	----------	S	SA	SR	M	D	A	D	A	----------	P



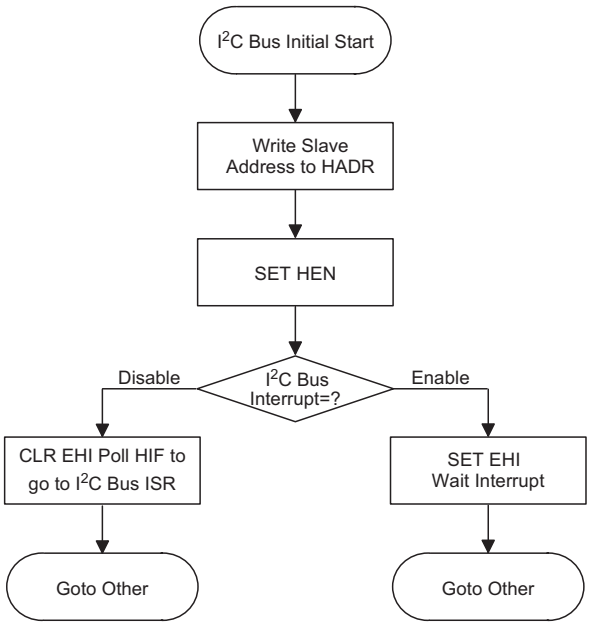
起始位



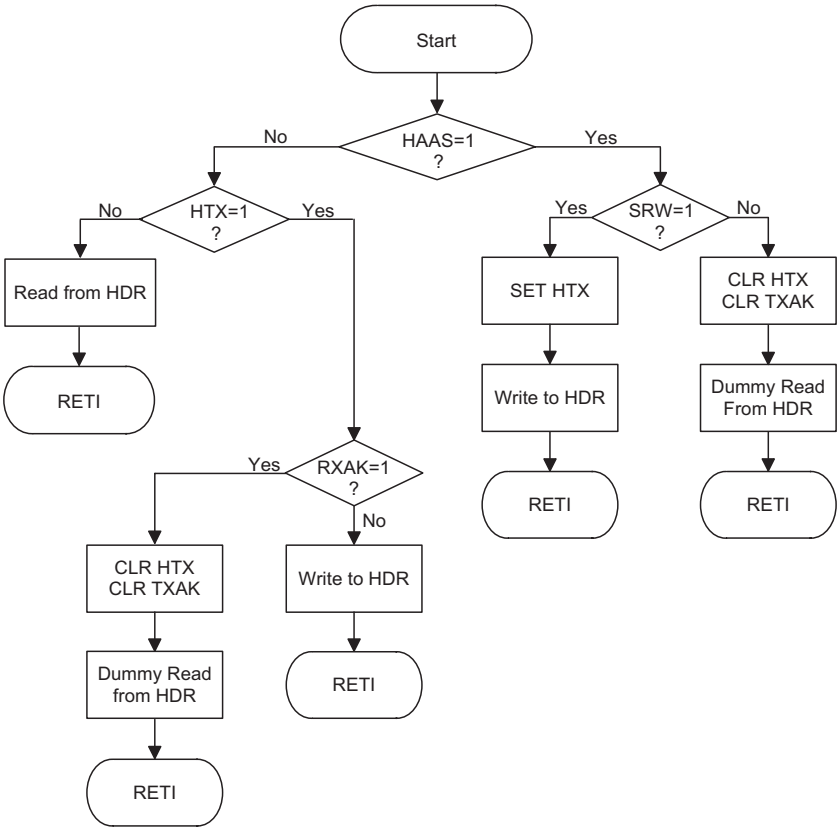
停止位



I²C 总线初始化程序流程图



I²C 总线中断服务程序流程图

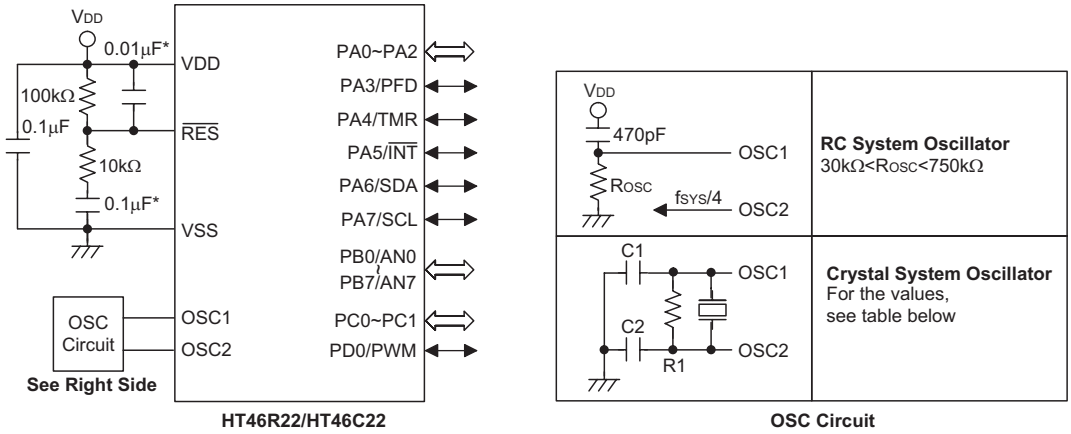


掩模选项

下表列出了所有掩膜选项。所有选项必须正确定义，以保证统正常运行。

编号	选项
1	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。
2	WDT 时钟源选择。 有三种选择：内部 WDT 振荡、指令时钟或关闭。
3	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。
4	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。
5	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。
6	PFD 选择：PA3 是电平输出或 PFD 输出
7	PWM 选择：（7+1）或（6+2）模式 PD0：电平输出或 PWM 输出
8	WDT 预分频选择。 有四种分频选择：2^{12}、2^{13}、2^{14}、2^{15}
9	低电压复位功能：打开/关闭
10	I^2C 总线：打开/关闭

应用电路



下表是不同晶体频率时，C1、C2 和 R1 的不同取值。

晶体或谐振器	C1,C2	R1
4MHz 晶体	0pF	10kΩ
4MHz 谐振器 (3 个引脚)	0pF	12kΩ
4MHz 谐振器 (2 个引脚)	10pF	12kΩ
3.58 MHz 晶体	0pF	10kΩ
3.58MHz 谐振器 (2 个引脚)	25pF	10kΩ
2 MHz 晶体或谐振器 (2 个引脚)	25pF	10kΩ
1 MHz 晶体	35pF	27kΩ
480kHz 谐振器	300pF	9.1kΩ
455 kHz 谐振器	300pF	10kΩ
429 kHz 谐振器	300pF	10kΩ

注： 电阻和电容值选取的原则是使 VDD 保持稳定并在$\overline{\text{RES}}$置为高以前把工作电压保持在允许的范围内。
“*”为了避免噪声干扰，连接$\overline{\text{RES}}$引脚的线请尽可能地短

指令集摘要

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

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

注： x：立即数

m：数据存储器地址

A：累加器

i：第 0~7 位

addr：程序存储器地址

√：影响标志位

—：不影响标志位

(1)：如果数据是加载到 PCL 寄存器，则指令执行周期会被延长一个指令周期(四个系统时钟)。

(2)：如果满足跳跃条件，则指令执行周期会被延长一个指令周期(四个系统时钟)；否则指令执行周期不会被延长。

(3)：(1)和(2)

(4)：如果执行 CLR WDT1 或 CLR WDT2 指令后，看门狗定时器被清除，则会影响 TO 和 PDF 标志位；否则不会影响 TO 和 PDF 标志位。

ADC A, [m] 累加器与数据存储器、进位标志相加，结果放入累加器
说明： 本指令把累加器、数据存储器值以及进位标志相加，结果存放到累加器。
运算过程： ACC←ACC+[m]+C
影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADCM A, [m] 累加器与数据存储器、进位标志相加，结果放入数据存储器
说明： 本指令把累加器、数据存储器值以及进位标志相加，结果存放到存储器。
运算过程： [m]←ACC+[m]+C
影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADD A, [m] 累加器与数据存储器相加，结果放入累加器
说明： 本指令把累加器、数据存储器值相加，结果存放到累加器。
运算过程： ACC←ACC+[m]
影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADD A, x 累加器与立即数相加，结果放入累加器
说明： 本指令把累加器值和立即数相加，结果存放到累加器。
运算过程： ACC←ACC+x
影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADDM A, [m] 累加器与数据存储器相加，结果放入数据存储器
说明： 本指令把累加器、数据存储器值相加，结果存放到数据存储器。
运算过程： [m]←ACC+[m]
影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

AND A, [m] 累加器与数据存储器做“与”运算，结果放入累加器
说明： 本指令把累加器值、数据存储器值做逻辑与，结果存放到累加器。
运算过程： ACC←ACC “AND” [m]
影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR WDT 清除看门狗定时器
说明： 本指令清除 WDT 计数器(从 0 开始重新计数)，暂停标志位(PDF)和看门狗溢出标志位(TO)也被清零。
运算过程： WDT ← 00H
PDF&TO ← 0
影响标志位

TO	PDF	OV	Z	AC	C
0	0	—	—	—	—

CLR WDT1 预清除看门狗定时器

说明： 必须搭配 CLR WDT2 一起使用，才可清除 WDT 计时器(从 0 开始重新计数)。当程序只执行过该指令，没有执行 CLR WDT2 时，系统只会不会将暂停标志位(PDF)和计数溢出位(TO)清零，PDF 与 TO 保留原状态不变。

运算过程： WDT ← 00H*

PDF&TO ← 0*

影响标志位

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CLR WDT2 预清除看门狗定时器

说明： 必须搭配 CLR WDT1 一起使用，才可清除 WDT 计时器(从 0 开始重新计数)。当程序只执行过该指令，没有执行 CLR WDT1 时，系统只会不会将暂停标志位(PDF)和计数溢出位(TO)清零，PDF 与 TO 保留原状态不变。

运算过程： WDT ← 00H*

PDF&TO ← 0*

影响标志位

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CPL [m] 对数据存储器取反，结果放入数据存储器

说明： 本指令是将数据存储器内保存的数值取反。

运算过程： [m] ← $[\overline{m}]$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

CPLA [m] 对数据存储器取反，结果放入累加器

说明： 本指令是将数据存储器内保存的值取反后，结果存放在累加器中。

运算过程： ACC ← $[\overline{m}]$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

DAA [m] 将加法运算后放入累加器的值调整为十进制数，并将结果放入数据存储器

说明 本指令将累加器高低四位分别调整为 BCD 码。如果低四位的值大于“9”或 AC=1，那么 BCD 调整就执行对原值加“6”，并且内部进位标志 AC1= $\overline{AC}$ ，即 AC 求反；否则原值保持不变。如果高四位的值大于“9”或 C=1，那么 BCD 调整就执行对原值加“6”再加 AC1，并把 C 置位；否则 BCD 调整就执行对原值加 AC1，C 的值保持不变。结果存放到数据存储器中，只有进位标志位(C)受影响。

操作 如果 ACC.3~ACC.0 > 9 或 AC=1

那么 [m].3~[m].0 ←(ACC.3~ACC.0)+6，AC1= $\overline{AC}$

否则 [m].3~[m].0 ←(ACC.3~ACC.0)，AC1=0

并且

如果 ACC.7~ACC.4+AC1 > 9 或 C=1

那么 [m].7~[m].4 ←(ACC.7~ACC.4)+6+ AC1，C=1

否则 [m].7~[m].4 ←(ACC.7~ACC.4)+ AC1，C=C

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

DEC [m] 数据存储器的内容减 1，结果放入数据存储器

说明： 本指令将数据存储器内的数值减一再放回数据存储器。

运算过程： [m] ← [m]-1

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

DECA [m] 数据存储器的内容减 1，结果放入累加器

说明： 本指令将存储器内的数值减一，再放到累加器。

运算过程： ACC ← [m]-1

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

HALT 进入暂停模式

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

运算过程： PC ← PC+1

PDF ← 1

TO ← 0

影响标志位

TO	PDF	OV	Z	AC	C
0	1	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RETI 从中断返回

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

运算过程： PC ← Stack

EMI ← 1

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RL [m] 数据存储器左移一位，结果放入数据存储器

说明： 本指令是将数据存储器内的数值左移一位，第 7 位移到第 0 位，结果送回数据存储器。

运算过程： [m].0 ←[m].7, [m].(i+1) ←[m].i； (i=0~6)

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RLA [m] 数据存储器左移一位，结果放入累加器

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

运算过程： ACC.0 ←[m].7, ACC.(i+1) ←[m].i； (i=0~6)

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RLC [m] 带进位将数据存储器左移一位，结果放入数据存储器

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

运算过程： [m].(i+1) ←[m].i； (i=0~6)

[m].0 ←C

C ← [m].7

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RLCA [m] 带进位将数据存储器左移一位，结果放入累加器

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

运算过程： ACC.(i+1) ←[m].i； (i=0~6)

ACC.0 ←C

C ← [m].7

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RR [m] 数据存储器右移一位，结果放入数据存储器

说明： 本指令是将存储器内的数值循环右移，第 0 位移到第 7 位，结果送回数据存储器。

运算过程： [m].7 ←[m].0, [m].i ←[m].(i+1); (i=0~6)

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RRA [m] 数据存储器右移一位，结果放入累加器

说明： 本指令是将数据存储器内的数值循环右移，第 0 位移到第 7 位，结果送回累加器，而数据存储器内的数值不变。

运算过程： ACC.7 ←[m].0, ACC.i ←[m].(i+1); (i=0~6)

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RRC [m] 带进位将数据存储器右移一位，结果放入数据存储器

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

运算过程： [m].i ←[m].(i+1); (i=0~6)

[m]. 7 ←C

C ← [m].0

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RRCA [m] 带进位将数据存储器右移一位，结果放入累加器

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

运算过程： ACC.i ←[m].(i+1); (i=0~6)

ACC.7 ←C

C ← [m].0

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

SBC A,[m] 累加器与数据存储器、进位标志相减，结果放入累加器

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

运算过程： ACC←ACC+[$\overline{m}$]+ C

影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZ [m] 数据存储器加 1，如果结果为“0”，则跳过下一条指令

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

运算过程： 如果 ([m]+1=0)，跳过下一行指令；[m] ←[m]+1

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZA 数据存储器加 1，将结果放入累加器，如果结果为“0”，则跳过下一条指令

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

运算过程： 如果[m]+1=0，跳过下一行指令；ACC ←([m]+1)

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SNZ [m]. i 如果数据存储器的第 i 位不为“0”，则跳过下一条指令

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

运算过程： 如果 [m].i≠0，跳过下一行指令。

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SUB A, [m] 累加器与数据存储器相减，结果放入累加器

说明： 本指令是把累加器值、数据存储器值相减，结果放到累加器。

运算过程： ACC←ACC+[$\overline{m}$]+1

影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SUB A, x 累加器与立即数相减，结果放入累加器

说明： 本指令是把累加器值、立即数相减，结果放到累加器。

运算过程： ACC←ACC+$\overline{x}$+1

影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SUBM A, [m] 累加器与数据存储器相减，结果放入数据存储器

说明： 本指令是把累加器值、存储器值相减，结果放到存储器。

运算过程： [m]←ACC+[$\overline{m}$]+1

影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SWAP [m] 交换数据存储器的高低字节，结果放入数据存储器

说明： 本指令是将数据存储器的低四位和高四位互换，再将结果送回数据存储器。

运算过程： [m].7~[m].4↔[m].3~[m].0

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SWAPA [m] 交换数据存储器的高低字节，结果放入累加器

说明： 本指令是将数据存储器的低四位和高四位互换，再将结果送回累加器。

运算过程： ACC.3~ACC.0← [m].7~[m].4

ACC.7~ACC.4← [m].3~[m].0

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZ [m] 如果数据存储器为“0”，则跳过下一条指令

说明： 本指令是判断数据存储器内的数值是否为 0，为 0 则跳过下一行指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以得正确的指令(二个指令周期)。否则执行下一条指令(一个指令周期)。

运算过程： 如果 [m] = 0, 跳过下一行指令。

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZA [m] 数据存储器送至累加器，如果内容为“0”，则跳过下一条指令

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

运算过程： 如果[m] = 0，跳过下一行指令，并 ACC← [m]。

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

TABRDC [m] 读取 ROM 当前页的内容，并送至数据存储器和 TBLH
说明： 本指令是将表格指针指向程序寄存器当前页，将低位送到存储器，高位直接送到 TBLH 寄存器内。
运算过程： [m] ←程序存储器低四位
TBLH←程序存储器高四位
影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

TABRDL [m] 读取 ROM 最后一页的内容，并送至数据存储器和 TBLH
说明： 本指令是将 TABLE 指针指向程序寄存器最后页，将低位送到存储器，高位直接送到 TBLH 寄存器内。
运算过程： [m] ←程序存储器低四位
TBLH←程序存储器高四位
影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

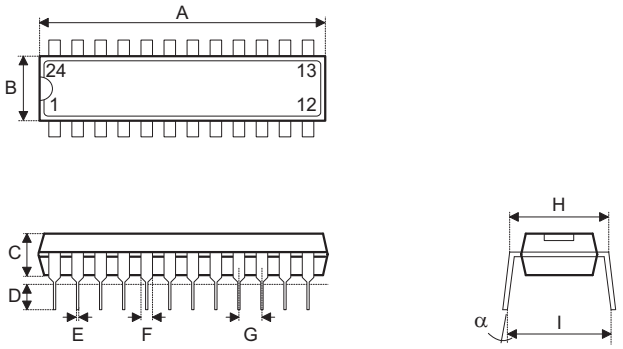
TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

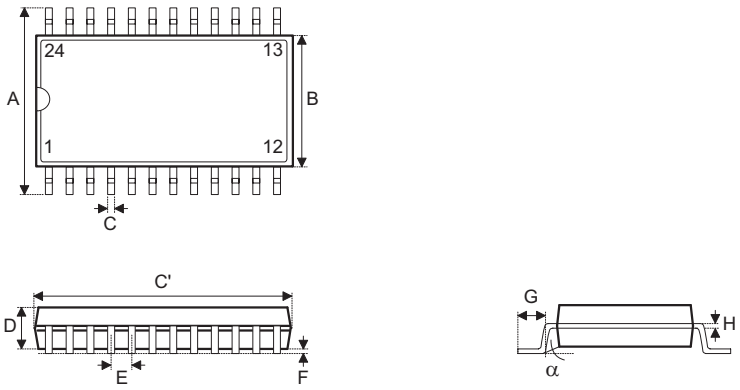
封装信息

24-pin SKDIP (300mil)外形尺寸



符号	尺寸 (单位：mil)		
	最小	典型	最大
A	1235	—	1265
B	255	—	265
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	345	—	360
α	0°	—	15°

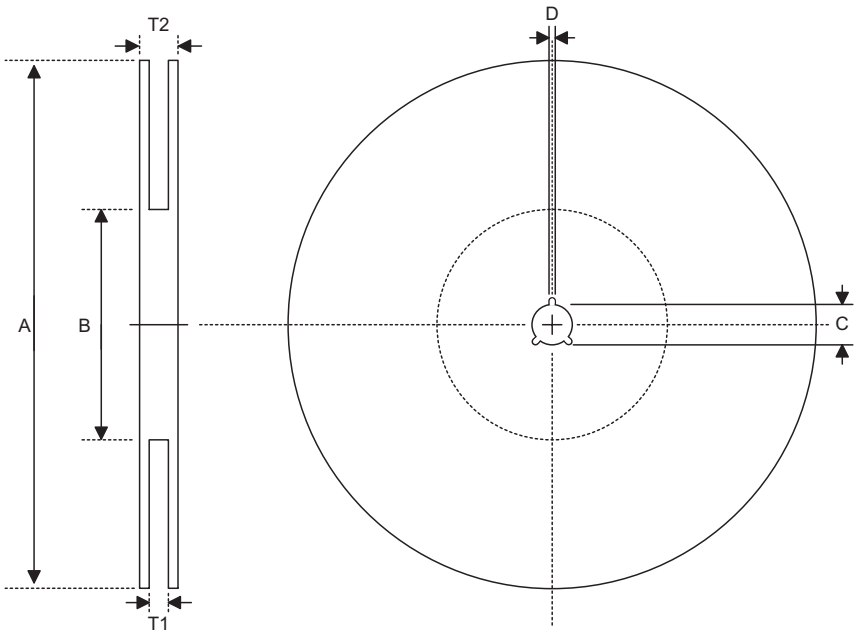
24-pin SOP (300mil)外形尺寸



符号	尺寸 (单位：mil)		
	最小	典型	最大
A	394	—	419
B	290	—	300
C	14	—	20
D	590	—	614
E	92	—	104
F	—	50	—
G	4	—	—
H	32	—	38
I	4	—	12
α	0°	—	10°

包装带和卷轴规格

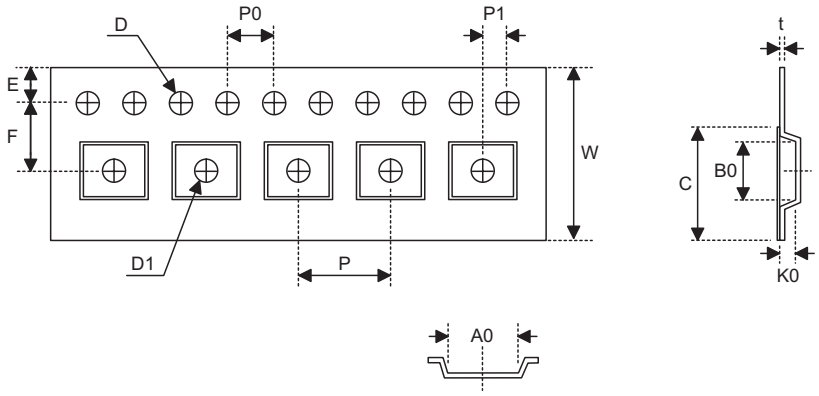
卷轴尺寸



SOP 24W

符号	说明	尺寸 (单位：mm)
A	卷轴外圈直径	330±1.0
B	卷轴内圈直径	62±1.5
C	轴心直径	12.75+0.15
D	缝宽	2.0+0.6
T1	轮缘宽	24.4±0.2
T2	卷轴宽	28.4+0.4

运输带尺寸



SOP 24W

符号	说明	尺寸 (单位：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+0.1
D1	空穴中之小孔直径	1.5+0.25
P0	穿孔间距	4.0±0.1
P1	空穴至穿孔距离(长度)	2.0±0.1
A0	空穴长	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

盛群半导体股份有限公司（总公司）
台湾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（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（上海）有限公司
上海宜山路 889 号 2 号楼 7 楼　200233
电话: 021-6485-5560
传真: 021-6485-0313
网站: www.holtek.com.cn

盛群半导体（香港）有限公司
香港九龙长沙湾道 777-779 号天安工业大厦 3 楼 A 座
电话: 852-2-745-8288
传真: 852-2-742-8657

Holmate Semiconductor, Inc.
46712 Fremont Blvd., Fremont, CA 94538
电话: 510-252-9880
传真: 510-252-9885
网站: www.holmate.com