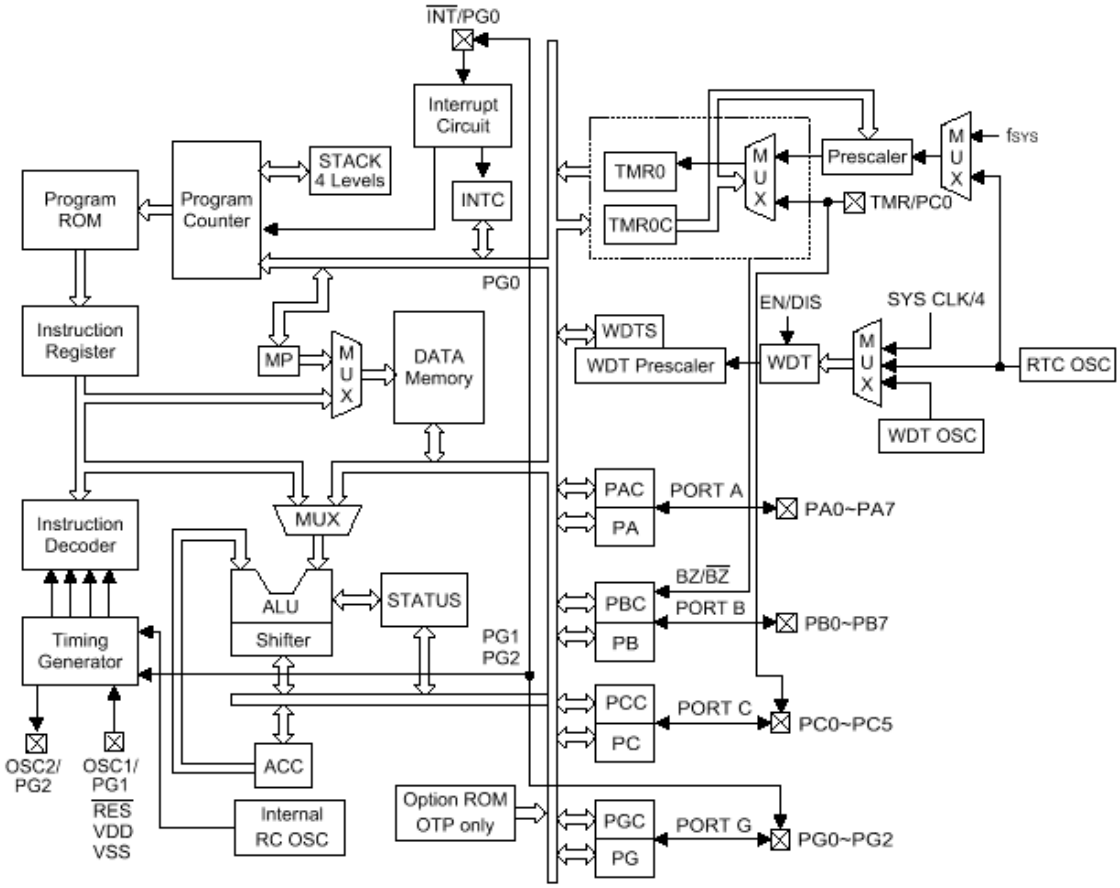
特点

- 工作电压:
  fsys = 4MHz：3.3V～5.5V
  fsys = 8MHz：4.5V～5.5V
- 低电压复位功能
- 25 个双向输入/输出口（最大情况）
- 1 个与外部中断复用的 I/O 口
- 带溢出中断及 8 阶预分频器的 8 位可编程定时/计数器
- 内部 RC 振荡器，外部石英和 RC 振荡器
- 为定时设计的 32768Hz 的 RC 振荡器
- 看门狗定时器
- 2048×14 位的程序存储器 ROM
- 96×8 位的数据存储器 RAM
- 一对蜂鸣器输出和 PFD 输出
- 具有暂停和唤醒功能来降低功耗
- 4 级硬件堆栈
- 在 V_{DD} = 5V 时系统时钟为 8MHz 时，指令周期为 0.5μs
- 位操作指令
- 查表指令可读取 14 位的表格内容
- 63 条功能强大的指令
- 所有指令执行时间皆为 1 或 2 个指令周期
- 24/28 引脚 SKDIP/SOP 的封装

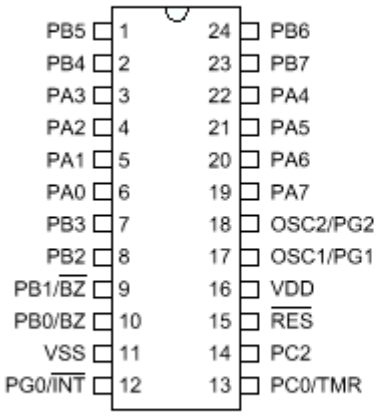
概述

HT48R30A_1 为八位高性能精简指令集单片机。专门为多输入/输出口的产品而设计的。这些器件适用于遥控器、电扇 / 电灯控制器、洗衣机控制器、电子秤、玩具以及各种电子系统的控制器。可进入暂停模式来降低功耗。

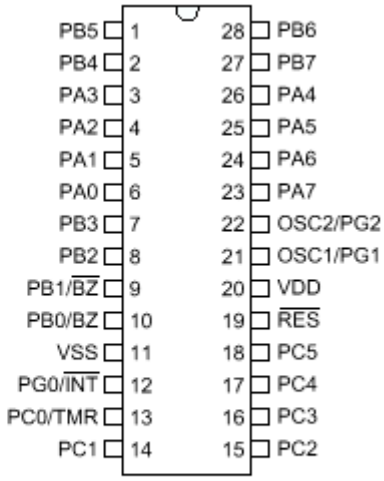
HT48R30A-1 系统框图



芯片引脚图



HT48R30A-1-A
– 24 SKDIP/SOP



HT48R30A-1-A
– 28 SKDIP/SOP

HT48R30A-1 引脚说明

引脚名称	输入/输出	掩模选项	说　　明
PA0~PA7	输入/输出	上拉电阻* 唤醒 CMOS/斯密特触发输入	8 位双向输入/输出端 每一位能由掩模选项设置为唤醒输入 可由软件指令设置为 CMOS 输出或斯密特触发输入，作为 CMOS 输入时可由掩模选项设置是否带上拉电阻
PB0/BZ PB1/$\overline{BZ}$ PB2~PB7	输入/输出	上拉电阻* PB0 或$\overline{BZ}$ PB1 或 BZ	8 位双向输入/输出口 可由软件指令设置为 CMOS 输出或斯密特触发输入，作为输入时可由掩模选项设置是否带上拉电阻 PB0 和 PB1 是与 BZ 和$\overline{BZ}$ 复用的引脚。一旦 PB0 和 PB1 选为蜂鸣器输出，它的输出信号则由内部的 PFD 发生器（由定时/计数器编程决定）提供
V_{SS}	—	—	负电源(接地)
PG0/$\overline{INT}$	输入/输出	上拉电阻*	双向输入/输出口 可由软件指令设置为 CMOS 输出或斯密特触发输入，作为输入时可由掩模选项设置是否带或上拉电阻（可一位选择） 外部中断输入与 PG0 共用；在下降沿触发有效
PC0/TMR PC1~PC5	输入/输出	上拉电阻*	双向输入/输出口 可由软件指令设置为 CMOS 输出或斯密特触发输入，作为输入时可由掩模选项设置是否带或上拉电阻（可对每一位进行选择） 定时/计数器外部输入与 PC0 共用
$\overline{RES}$	输入	—	斯密特触发复位输入端，低电平有效
V_{DD}	—	—	正电源输入
OSC1/PG1 OSC2/PG2	输入/输出	上拉电阻* 外部 Crystal 或 RC 内置 RC 振荡+输入/输出 内置 RC 振荡+RTC	OSC1 和 OSC2 连接至 RC 或 Crystal 振荡器（由掩模选项确定）来产生内部系统时钟；在 RC 振荡方式下，OSC2 是系统时钟四分频的输出端；这两个引脚也可以通过掩模来选择作为 RTC 振荡器（32768 Hz）或是输入/输出口，在这两种情况下，系统的时钟由内部的 RC 振荡器提供，频率可有四种选择（3.2 MHz，1.6 MHz，800 kHz，400 kHz）；若选择作为输入/输出时，可掩模选择是否具有上拉电阻。否则，PG1 和 PG2 作为内部的寄存器且不带上拉电阻

注意： “*”每一个 I/O（PA，PB，PC，PG）端口的上拉电阻在掩模选择时，可以以位为单位选择；
PA 口的 COMS 或斯密特触发也可在掩模选择时一位选择

极限参数

电源电压 …… $V_{SS}-0.3 \sim V_{SS}+5.5V$　　储存温度 …… −50℃~ 125℃

输入电压 …… $V_{SS}-0.3 \sim V_{DD}+0.3V$　　工作温度 …… −40℃~ 85℃

注意：这是器件的极限参数。超越上述的极限参数范围可能会对器件造成严重的损害。器件功能并非必然可于未列于规格中的其他情况而且长期暴露于极限状况下可能会影响器件的可靠性。

直流特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD1}	工作电压	—	F_{sys}=4MHz	3.3	—	5.5	V
V_{DD2}	工作电压	—	F_{sys}=8MHz	4.5	—	5.5	V
I_{DD1}	工作电流 (石英振荡器)	3.3V	无负载 f_{sys}=4MHz	—	1	2	mA
		5V		—	3	5	
I_{DD2}	工作电流 (RC 振荡)	3.3V	无负载 f_{sys}=4MHz	—	1	2	mA
		5V		—	3	5	
I_{DD3}	工作电流 (石英振荡器)	5V	无负载 f_{sys}=8MHz	—	4	8	mA
I_{STB1}	静态电流 (WDT 使能，RTC 关闭)	3.3V	无负载 系统暂停模式	—	—	5	μA
		5V		—	—	10	
I_{STB2}	静态电流 (WDT 除能，RTC 关闭)	3.3V	无负载 暂停模式	—	—	1	μA
		5V		—	—	2	
I_{STB3}	静态电流 (WDT 除能，RTC 打开)	3.3V	无负载 暂停模式	—	—	5	μA
		5V		—	—	10	
V_{IL1}	输入/输出口输入低电压	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	输入/输出口输入高电压	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	低电平输入电压（$\overline{RES}$）	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	高电平输入电压（$\overline{RES}$）	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	输入/输出口灌电流	3.3V	V_{OL}=$0.1V_{DD}$	4	8	—	mA
		5V	V_{OL}=$0.1V_{DD}$	10	20	—	
I_{OH}	输入/输出口源电流	3.3V	V_{OH}=$0.9V_{DD}$	−2	−4	—	mA
		5V	V_{OH}=$0.9V_{DD}$	−5	−10	—	
R_{PH}	上拉电阻	3.3V	—	40	60	80	KΩ
		5V	—	10	30	50	
V_{LVR}	低电压复位	—	选择 3.3V	2.7	3.0	3.3	V

交流特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{SYS1}	系统时钟 （石英晶体振荡器）	3.3V	—	400	—	4000	kHz
		5V		400	—	8000	
f_{SYS2}	系统时钟 （RC 振荡器）	3.3V	—	400	—	4000	kHz
		5V		400	—	8000	
f_{SYS3}	系统时钟 （内置 RC 振荡器）	3.3V	3.2MHz	1600	2500	3500	kHz
		5V		2000	3200	4500	
F_{TIMER}	定时/计数器的输入频率 （TMR0/TMR1）	3.3V	—	0	—	4000	kHz
		5V	—	0	—	8000	
t_{WDTOSC}	看门狗定时器振荡器	3.3V	—	43	86	168	μs
		5V	—	36	72	144	
t_{WDT1}	看门狗定时溢出周期 (WDT 时钟)	3.3V	没有看门狗前置分频器	11	22	43	ms
		5V		9	18	37	

t_{WDT2}	看门狗定时溢出周期(系统时钟)	—	没有看门狗前置分频器	—	1024	—	t_{SYS}
t_{WDT3}	看门狗定时溢出周期(RTC 时钟)	—	没有看门狗前置分频器	—	7.812	—	m s
t_{RES}	外部复位低电平脉冲宽度	—	—	1	—	—	μ s
t_{SST}	系统启动计时器周期	—	上电或从暂停状态唤醒	—	1024	—	t_{SYS}
t_{INT}	中断脉冲宽度	—	—	1	—	—	μ s

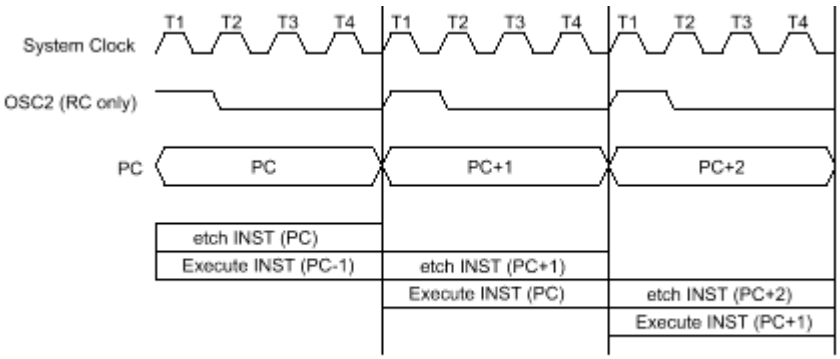
附注：$t_{SYS} = 1/f_{SYS}$

功能说明

指令执行时序

系统时钟由石英晶体振荡器或 RC 振荡器产生，系统内部将此频率分为四个不重叠的时钟（一般称为 T 状态，分别为 T1、T2、T3、T4），一个指令周期包含了四个 T 状态，即一个指令周期为四个系统时钟周期。

指令的读取与执行是以流水线方式来进行的。这种方式允许在一个指令周期进行读取指令操作，而在下一个指令周期里进行解码与执行该指令。这种流水线方式能在一个指令周期里有效地执行一个指令。但是，如果涉及到的指令要改变程序计数器（如 JMP，CALL 等），就需要花两个指令周期来完成这一条指令。



Execution flow

程序计数器（Program Counter）

程序计数器是作为程序存储器寻址之用，控制了程序的流程单片机通过 PC 指向的程序存储器的地址取得一条指令码后， PC 会自动地指向下一条指令的地址（PC 值自动加 1）。

但是若执行的是如下指令：跳转、条件跳跃、读取 PCL 的值、子程序调用、初始复位、内部或外部中断响应、子程序返回等，则 PC 要根据每一条指令获得其相应的地址来控制程序的转向。

模式	程序计数器内容										
	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
初始复位	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/计数器溢出中断	0	0	0	0	0	0	0	1	0	0	0
跳过下一指令	PC+2										
装入 PCL	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
转子（Call Branch）	#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

附注： *10~*0：程序指针
#10~#0：指令指针
s10~10：堆栈指针
@7~@0：PCL 字节

比如执行条件跳转指令，一旦条件符合，则在当前执行指令时所获取的指令码不会被执行，并且同时会插入一个假的指令周期（dummy cycle），换句话说，相当与执行了一个 NOP 指令（空操作），这样 PC 才会指向正确的指令码的地址；反之，条件不符合时，PC 将指向下一条指令的地址。

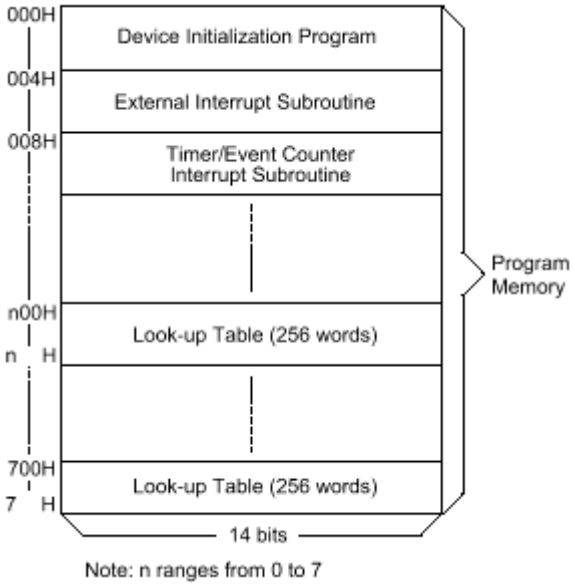
PC 的低位（PCL）是可读写的暂存器（06H）。若向 PCL 写入一个值将会产生一个短程的跳跃动作，这个短程跳跃的地址范围是 256 个地址，即在 ROM 的当前页。

当发生控制转移时，就会加入一个假的指令周期。

程序存储器（Program Memory － ROM）

程序存储器用来存储要执行的程序指令，也包含数据、表格、中断入口地址，由 2048×14 个位组成，由 PC 和表格指针来确定其地址。

ROM 里面的某些地址是为一些特殊使用而保留的，使用时应加以注意，避免误用，导致程序运行的不正常，以下是说明：



Note: n ranges from 0 to 7

- 地址 000H
  此区域保留给程序初始化之用。当系统复位时，程序会从 000H 地址开始执行。
- 地址 004H
  此区域保留给外部中断服务使用。当中断是开放的，且堆栈又未满，则一旦 INT 端被正确电平触发，就能产生中断，程序会从 004H 地址开始执行外部中断
  服务程序。
- 地址 008H
  此区域保留给定时/计数器中断服务使用。当中断是开放的，且堆栈又未满，则一旦定时/计数器发生溢出时，就能产生中断，程序会从 008H 地址开始执行中断服务程序。
- 表格地址（Table Location）

  ROM 内的任何地址都可被用来作为查表地址使用。查表指令为 TABRDC [m] 与 TABRDL [m]。TABRDC [m]是查表当前页的数据 [1 页=256 个字 (word)]。TABRDL [m]是查表最后一页的数据。[m] 为数据被存入的地址。在执行 TABRDC [m]指令（或 TABRDL [m] 指令）后，将会传送当前页（或最后一页）上的一个字的低位字节到[m]，而这个字的高位字节传送到 TBLH（08H）。只有表格中的低位字节被定义到目标地址中，而高位字节传送到表格的高位字节寄存器（TBLH）。TBLH 为只读寄存器。而表格指针（TBLP；07H）是可以读写的寄存器，用来指明表格地址。在访问表格以前，通过对 TBLP 寄存器赋值来指明表格地址。高位字节寄存器 TBLH 只能读出，不能写入。如果主程序和中断服务程序（ISR）同时使用查表指令，那么主程序读取的高位字节（即存放于高位字节寄存器 TBLH 之中）可能会被中断服务程序的查表指令改写而产生错误。因此，应该避免主程序和中断服务程序（ISR）同时使用查表指令。但是，如果主程序和中断服务程序必须同时使用查表指令的话，那么，主程序在使用查表指令之前，必须先关闭所有使用查表指令的相关中断，直到高位字节寄存器 TBLH 的内容被备份好再开放这些中断。查表指令要花两个指令周期来完成这一条指令的操作。按照用户的需要，表格地址这些位置可以作为正常的程序存储器来使用。

指令	表格地址										
	*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

表格地址

附注： *10~*0：表格地址字节
@7~@0：表格指针字节
P10~P8：当前程序指针字节

堆栈寄存器（STACK）

堆栈寄存器是用来存放程序计数器 PC 内容的一个特殊的寄存器。HT48R30A-1 有四层堆栈，该寄存器既不是数据存储器的一部分又不是程序的一部分，而且也不可读不可写。所进入的级数是由堆栈指针 SP 来指示的，而堆栈指针 SP 也是不可读不可写的。一旦发生了子程序的调用或是中断响应，则程序计数器 PC 的内容会被压入堆栈中。在子程序调用或中断响应结束时（可从返回指令（RET 或 RETI）看出），程序计数器 PC 的值会从堆栈中还原。在系统复位后，堆栈指针 SP 会指向堆栈的顶端。

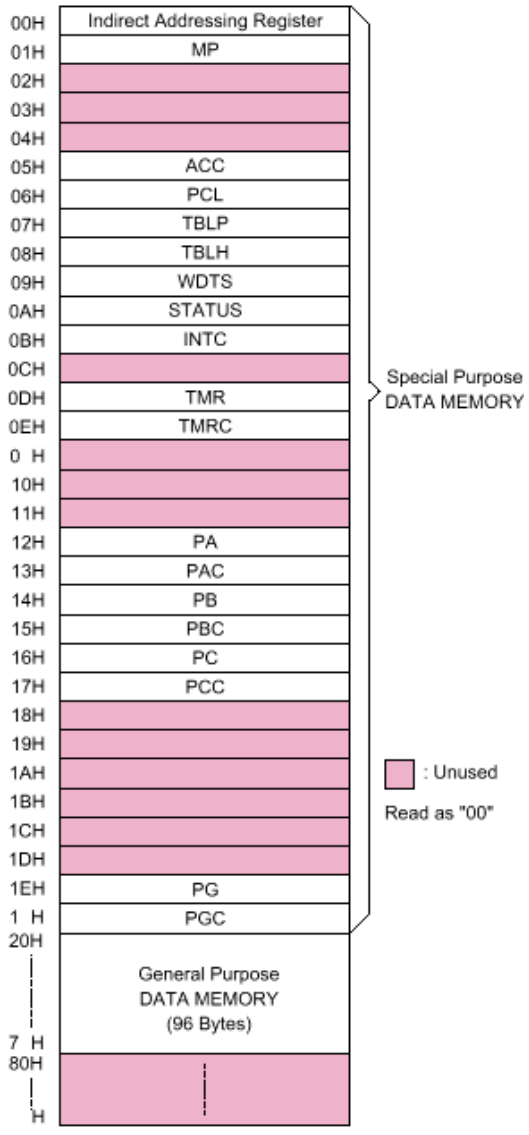
当堆栈已满，而此时又发生的中断请求，则这个中断的请求标志会被记录下来，该中断服务仍被禁止。一旦堆栈指针 SP 发生了递减（由于 RET 或 RETI）则会响应此未被服务的中断。这个功能就可确保堆栈不会溢出，使得编程者更加方便地使用该结构。同样地，如果堆栈已满，而随后又执行了 CALL 指令，此时会发生堆栈溢出，并且第一个返回地址将会丢失（只有最近的四个返回地址会被保存）。

数据存储器－RAM

数据存储器（RAM）由 115×8 个位组成，包含特别功能寄存器、通用数据寄存器两个不同功能的寄存器。这两个寄存器大多是可读可写的，只有少部分是只读的。

特殊功能寄存器包括：间接寻址寄存器 R0（00H），定时/计数器（TMR；0DH），定时/计数器控制寄存器（TMRC；0EH），程序计数器低字节寄存器（PCL；06H），存储器指针寄存器 MP（01H），累加器（ACC；05H），表格指针（TBLP；07H），表格高字节寄存器（TBLH；08H），状态寄存器（STATUS；0AH），中断控制寄存器（INTC；0BH），看门狗定时器寄存器（WDTS；09H），I/O 寄存器（PA；12H，PB；14H，PC；16H，PG；1EH），输入/输出控制寄存器（PAC；13H，PBC；15H，PCC；17H，PGC；1FH）。在 20H 以前的剩余单元都保留为将来进一步扩展使用。读取这些被保留单元的值，都将返回 00H 的值。

通用数据存储器的地址从 20H~7FH，作为数据和控制信息使用。



RAM mapping

所有的 RAM 都可以直接执行算术、逻辑、递增、递减和移位等运算。除了一些少数指定的位之外，RAM 的每个位都可以由 SET[m].i 和 CLR[m].i 指令来置位和复位。这些 RAM 地址可以通过存储器指针寄存器 MP（01H）来存取。

间接寻址寄存器（Indirect Addressing Register）

地址 00H 是间接寻址寄存器，但并无实际的地址存在。直接读取地址 00H 会得到结果 00H，而直接向该地址写值也是无效的，不会产生任何结果。地址 00H 的任何读/写操作是依据 MP（01H）所指向的 RAM 的地址动作的，即对间接寻址寄存器做读写的目的地址为存储器指针寄存器所指的地址。

存储器指针寄存器 MP（01H）是一个 8 位的寄存器。

累加器（ACC）

累加器（ACC）与算术逻辑单元（ALU）有关，同样也是对应至 RAM 的地址 05H，作为运算的立即数据，存储器之间的数据传送必须经过 ACC。

算术逻辑单元（ALU）

算术逻辑单元（ALU）为执行八位算术及逻辑运算的电路，提供有下列的功能：

- 算术运算（ADD，ADC，SUB，SBC，DAA）
- 逻辑运算（AND，OR，XOR，CPL）
- 移位（PL，RR，RLC，RRC）
- 递增及递减（INC，DEC）
- 进位及无进位减法
- 分支判断（SZ，SNZ，SIZ，SDZ 等）

ALU 不仅可以储存数据运算的结果，还可以改变状态寄存器

状态寄存器—STATUS

状态寄存器（0AH）的宽度为 8 位，由零标志位（Z），进位标志位（C），辅助进位标志位（AC），溢出标志位（OV），掉电标志位（PD），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，而且还控制运算顺序。

除了 TO 和 PD 以外，状态寄存器中的位都可用指令来改变，这种情况与其它寄存器一样。任何写到状态寄存器的数据不会改变 TO 或 PD 标志位。但是与状态寄存器有关的操作会导致状态寄存器的改变。系统上电，看门狗定时器溢出，执行 HALT 指令，或清除看门狗定时器都能改变 TO 和 PD。

Z，OV，AC 和 C 标志位都反映了最近运算的状态。

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

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发生借位,那么 C 被置位；反之，C 被清除。它也可被一个带进位循环移位指令影响。
AC	1	在加法运算中低四位产生了向高四位进位，或减法运算中低四位不发生从高四位借位，AC 被置位；反之，AC 被清除。
Z	2	算术运算或逻辑运算的结果为零则 Z 被置位；反之，Z 被清除。
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，或那么 OV 被置位；反之，OV 被清除。
PD	4	系统上电或执行了 CLR WDT 指令，PD 被清除。执行 HALT 指令 PD 被置位。
TO	5	系统上电或执行了 CLR WDT 指令，或执行 HALT 指令，TO 被清除。WTD 定时溢出，TO 被置位。
—	6	未定义，读出为零
—	7	未定义，读出为零

状态寄存器

中 断—INT

本单片机提供一个外部中断和内部定时/计数器中断。中断控制寄存器（INTC；0BH）包含了中断控制位，用来设置中断允许/禁止及中断请求标志。

一旦有中断子程序被服务，所有其它的中断将被禁止（通过清除 EMI 位）。这种机制能防止中断嵌套。这时如有其它中断请求发生，这个中断请求的标志会被记录下来。如果在一个中断服务程序中有另一个中断需要服务的话，程序员可以设置 EMI 位及 INTC 所对应的位来允许中断嵌套服务。如果堆栈已满，该中断请求将不会被响应。即使相关的中断被允许，也要到堆栈指针发生递减时才会响应。如果需要立即得到中断服务，则必须避免让堆栈饱和。

所有的中断都具有唤醒功能。当一个中断被服务时，会产生一个控制传送：通过将程序计数器（PC）压入堆栈，然后转移到中断服务程序的入口。只有程序计数器的内容能压入堆栈。如果寄存器和状态寄存器的内容会被中断服务程序改变，从而破坏主程序的预定控制，那么程序员必须事先将这些数据保存起来。

外部中断是由 INT 脚上的电平由高到低的变化触发的，相关的中断请求位（EIF，INTC 的第 4 位）被置位。当中断允许，堆栈也没有满，一个外部中断触发时，那么将会产生地址 04H 的子程序调用。中断请求标志（EIF）和 EMI 位将也会被清除来禁止另外的中断发生。

内部定时/计数器中断是通过置位定时/计数器中断请求标志位（TF，INTC 的第五位）来初始化的，中断的请求是由定时器溢出产生的。当中断允许，堆栈又未满，并且 TF 已被置位，就会产生地址 08H 的子程序调用。该中断请求标志位（TF）被复位并且 EMI 位也将被清除，以便将其他中断禁止。

单片机在执行中断子程序期间，其他的中断响应会被暂停，直到执行 RETI 指令或是 EMI 位和相关的中断控制位都被置为 1（当堆栈是未满时）。若要从中断子程序返回时，只要执行 RET 或 RETI 指令即可。RETI 指令将会自动置位 EMI 来再次允许中断服务，而 RET 则不能自动置位 EMI。

如果中断在内部二个连续的 T2 脉冲上升沿间发生，而且中断响应被允许的话，那么在第二个 T2 脉冲，该中断会被服务。如果同时发生中断服务请求，那么下列表中列出了中断服务优先等级。这种优先级也可以通过 EMI 位的复位来屏蔽。

NO	中断源	优先级	中断
A	外部中断	1	04H
B	定时/计数器溢出	2	08H

中断控制寄存器（INTC）由定时/计数器中断请求标志位（TF），外部中断请求标志位（EIF），定时/计数器允许位（ETI），外部中断允许位（EEI），和主中断控制允许位（EMI）组成。EMI、EEI 和 ETI 都是用来控制中断的允许/禁止状态的。这些位防止正在进行的中断服务中的中断请求。一旦中断请求标志位被置位（TF，EIF)， 它们将在 INTC 寄存器中被保留下来，直到相关的中断被服务或由软件指令来清除。

建议不要在中断子程序中使用“CALL”指令来调用子程序，因为它可能会破坏原来的控制序列，而中断经常在不可预见的情况下发生或某一个确定的应用程序可能要求立即服务。基于上述情况，如果只剩下一个堆栈，若此时中断不能很好地被控制，而且在这个中断服务程中又执行了 CALL 子程序调用，则会造成堆栈溢出，而破坏原先的控制序列。

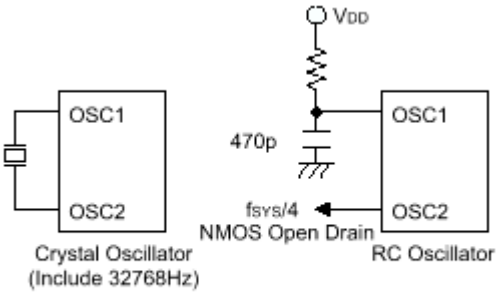
INTC (0BH)	位	符号	功　　能
	0	EMI	总中断控制位 1=允许， 0=禁止
	1	EEI	外部中断控制位 1=允许， 0=禁止
	2	ETI	定时/计数器中断控制位，1=允许， 0=禁止
	3	—	未使用位，读出为零
	4	EIF	外部中断请求标志位 1=有效，0=无效
	5	TF	定时/计数器中断请求位 1=有效，0=无效
	6	—	未使用位，读出为零

	7	—	未使用位，读出为零

中断控制寄存器—INTC （0BH）

振荡器

HT48R30A-1 有三种振荡电路。这三种振荡器都是针对系统时钟而设计的，分别是外部 RC 振荡器、外部石英振荡器以及内部的 RC 振荡器，可在掩模时选择使用哪一种振荡器。不管所选的是哪一种振荡器，其信号都可以支持系统的时钟。进入暂停模式会使系统时钟工作停止。进入暂停模式还可以阻止外部信号的进入，可以降低功耗。



System oscillator

如果使用 RC 型振荡器，在 OSC1 和 V_{DD} 之间需要一个外部电阻，其阻值范围为 51kΩ至 1MΩ。在 OSC2 端可获得系统时钟四分频信号，它可用于同步外部逻辑电路。一般 RC 型振荡器提供了最经济的解决方式，可是，振荡频率会随着 V_{DD}、温度和制造漂移而不同。因此，在用于需要非常精确振荡频率的计时操作场合，我们并不建议使用 RC 型振荡器。

如果选用的是石英振荡器，那么在 OSC1 和 OSC2 之间需要连接一个石英，用来提供石英振荡器所需要的反馈和相移。另外，在 OSC1 和 OSC2 之间还可以用谐振器代替石英振荡器，来产生系统时钟，但是在 OSC1 和 OSC2 需要多连接两个电容器至地。如果使用的是内部的 RC 振荡器，那么 OSC1 和 OSC2 可以选择作为通用的 I/O 引脚或者是作为 32768Hz 石英振荡器（RTC OSC）。内部的 RC 振荡器的频率可以选择 3.2MHz、1.6MHz、800kHz 和 400kHz（在掩模时选择）。

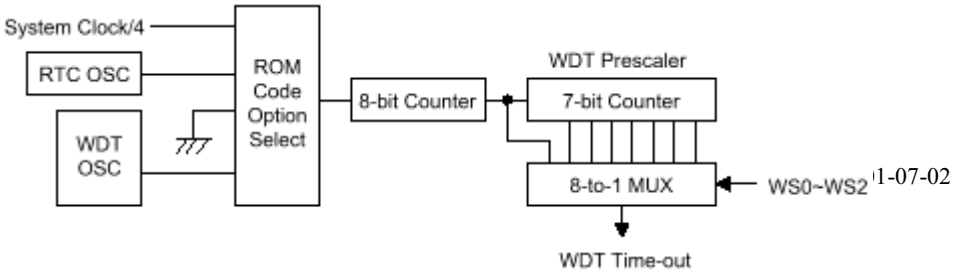
WDT 振荡器是 IC 内部自由振荡的 RC 型振荡器，不需要任何外部元件。即使在系统进入暂停模式，系统时钟被停止，但这个 RC 振荡器仍会运作。(其振荡周期大约为 72μs)。在掩模制造时，如欲节省电源，可将 WDT 振荡器除能。

看门狗计时器(Watchdog Timer)

WDT 的时钟源是可选专用的 RC 振荡器（WDT 振荡器）、RTC 时钟或是指令时钟（系统时钟 4 分频），由掩模选项来决定看门狗主要用来避免程序运行失常和程序跳入一死循环而导致不可预测的结果。看门狗可用掩模来使能或除能（disable），如果在除能状态，所有的 WDT 指令都是没有作用的。只有选择“内部的 RC 振荡器＋RTC”模式时，RTC 时钟才是有效的。

如果选择了内部 WDT 振荡器（以 72μS/5V 为周期的 RC 振荡器）的话，这个频率会先除以 256（8 级）产生 18.6mS/5V 的溢出时间，这个溢出时间会因为温度，V_{DD}，以及芯片参数的变化而变化。如果启动 WDT 的前置分频器，则可实现延长溢出时间。写数据到 WS2,WS1,WS0 (WDTS 的第 2、1、0 位）会产生不同的溢出时间。举例来说，如果 WS2，WS1，WS0 的位都为 1，其分频因子最大为 1：128，同时溢出时间最长为 2.4S/5V。如果 WDT 振荡器被禁止，那么 WDT 的时钟来源可为指令时钟，其运作与 WDT 振荡器一样。但当在 HALT 状态时，来源于指令时钟的 WDT 会在暂停模式时停止计数并失去保护功能。在这种情况下，只能由外部逻辑来重新启动系统。WDTS 的高四位及其第 3 位保留给用户定义标志来使用，程序员可以利用这些标志来指示某些特殊的状态。

如果单片机工作在干扰很大的环境中，那么建议使用片内的 RC 振荡器（WDT OSC）或是 32kHz 的石英振荡器（RTC OSC），因为 HALT 模式会使系统时钟停止运作。



在正常运作下，WDT 溢出会使系统复位并设置 TO 状态位。但在 HALT 模式下，溢出只产生一个热复位，并只能使 PC 程序计数器和堆栈指针 SP 复位到零。要清除 WDT 的值（包括 WDT 前置分频器）可以有三种方法：外部复位（低电平输入到$\overline{\text{RES}}$端），用软件指令和 HALT 指令三种。软件指令由 CLR WDT 和 CLR WDT1 及 CLR WDT2 二组指令组成。这两组指令中，只能选取其中一种。选择的方式由掩模选项的 CLR WDT 次数选项决定。如果选择“CLR WDT”（即 CLR WDT 次数为 1），那么只要执行 CLR WDT 指令就会清除 WDT。在选择 CLR WDT1 和 CLR WDT2 的情况下（即 CLR WDT 次数为 2），那么要执行二条件指令才会清除 WDT，否则，WDT 会由于溢出而使系统复位。

WS2	WS1	WS0	分频率
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

看门狗定时器前置分频寄存器

暂停模式（HALT）

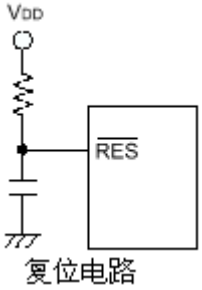
暂停模式是由 HALT 指令来实现的，有如下结果：

- 关闭系统振荡器，但 WDT 振荡器依然工作（如果 WDT 振荡器被选择）。
- RAM 及寄存器的内容保持不变。
- WDT 和 WDT 前置分频器被清除，并重新计数（如果 WDT 的时钟是来自 WDT 振荡器）。
- 所有的输入/输出端口都保持其原先状态。
- PD 标志位被置位，TO 标志位被清除。

外部复位、中断或外部输入一个下降沿的信号到 PA 口或 WDT 溢出均可使系统脱离暂停状态。外部复位能使系统初始化，而 WDT 溢出能执行“热复位”。测试 TO 和 PD 状态后，系统复位的原因就可以确定了。PD 标志位是由系统上电复位和执行 CLR WDT 指令被清除，而它的置位是由于执行了 HALT 指令。如 WDT 产生溢出，使 TO 标志位置位，还能产生唤醒，使得程序计数的 PC 和堆栈指针复位。其他都保持原状态。

PA 端口的唤醒和中断的方式看作为正常运行，PA 口的每一位都可以通过掩模选项来单独设定为对唤醒系统。如果唤醒是来自于输入/输出口的信号变化，程序会继续执行下一条命令。如果唤醒是来自中断的话，则会产生二种情况：如果相关的中断被禁止或中断是允许的，但堆栈已满，那么程序将继续执行下条指令，如果中断允许并且堆栈未满，那么这个中断响应就发生了。当唤醒事件发生时，要花 1024tsys（系统时钟周期）后，系统重新正常运行。这就是说，在唤醒后被插入了一个等待时间。如果唤醒是来自于中断响应，那么实际的中断程序执行就被延迟了一个以上的周期。但是如果唤醒导致下一条指令执行，那么在一个等待周期结束后指令就立即被执行。进入 HALT 模式前，如果中断请求标志位被置“1”，那么相关的中断唤醒功能被禁止。

为了减小功耗，在进入 HALT 模式之前必须要小心处理输入/输出端口的状态。但在 HALT 模式时，RTC 振荡器仍然运作（如果 RTC 振荡器被使能）



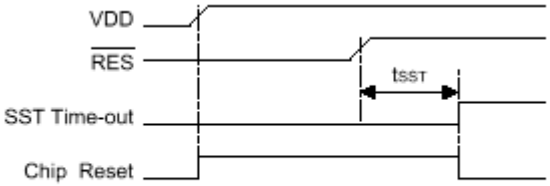
复位电路

复 位 （RESET）

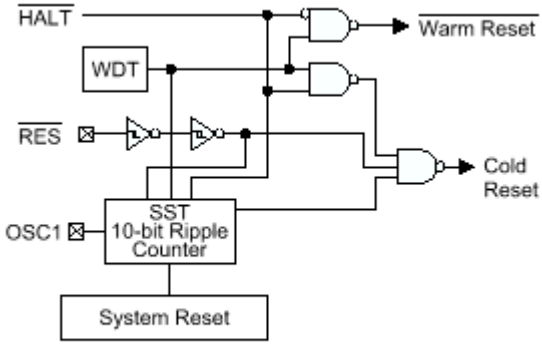
有三种方法可以产生复位

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

在 HALT 期间 WDT 溢出是不同于其它的复位操作条件，因为它可执行 “热复位”，结果只能使程序计数器 PC 和堆栈指针 SP 复位，而别的电路均保持原来的状态。在其他复位条件下，某些寄存器保持不变。当复位条件被满足时，极大多数的寄存器被复位到“初始状态”，通过测试 **PD** 标志位和 **TO** 标志位，程序能分辨不同的系统复位。



Reset timing chart



Reset configuration

TO	PD	复 位 条 件
0	0	上电时复位$\overline{\text{RES}}$
u	u	正常动作时$\overline{\text{RES}}$复位
0	1	$\overline{\text{RES}}$时唤醒 HALT
1	u	正常动作时 WDT 定时溢出
1	1	WDT 时唤醒 HALT

u 表示不变

为了保证系统振荡器起振并稳定运行，SST（系统启动定时器）当系统复位时（上电、WDT 定时器溢出或是$\overline{\text{RES}}$引脚复位）或是 HALT 模式唤醒时，它会提供额外的 1024 个系统时钟周期的延迟。

当系统上电时，SST 延迟被加到复位周期中。任何来自 HALT 的唤醒都将产生 SST 延迟。系统复位时各功能单元的状态如下所示：

PC	000H
中断	禁止
前置分频器	清除
WDT	清除，在主系统复位后，WDT 开始计数
定时/计数器 0/1	关闭
输入/输出端口	输入模式
堆栈指针 SP	指向堆栈顶部

特殊功能寄存状态概述表

寄存器	上电复位	正常运行期间		HALT 状态	
		WDT 溢出	RES 端复位	RES 端复位	WDT 溢出*
TMR	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	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
INTC	--00 -000	--00 -000	--00 -000	--00 -000	--uu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu 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
PC	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PCC	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PG	---- -111	---- -111	---- -111	---- -111	---- -uuu
PGC	---- -111	---- -111	---- -111	---- -111	---- -uuu

注意： 1.“*”表示“热复位。
2.“U”表示不变化。
3.“X”表示不确定。

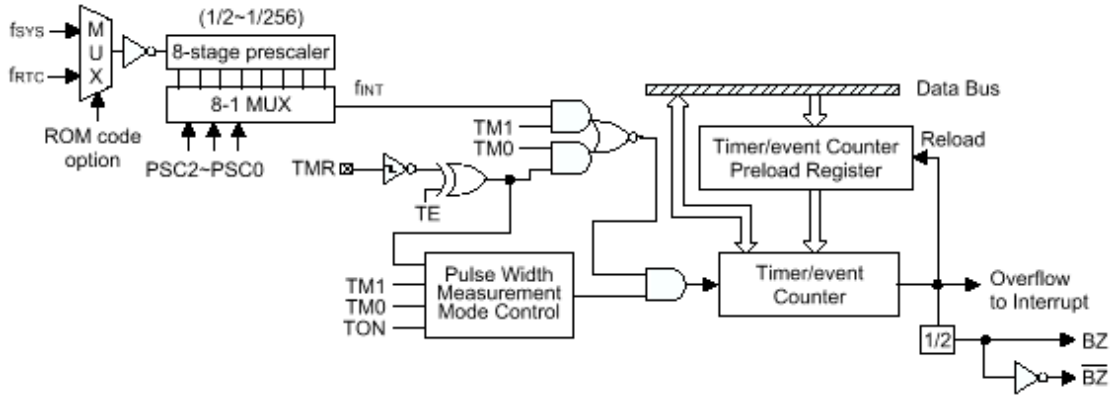
定时/计数器

HT48R30A-1 提供一个一个定时/计数器。定时/计数器包含一个 8 位可编程的向上计数的计数器，该定时/计数器的时钟来源可以是外部时钟、系统时钟或是 RTC 时钟。

如果采用内部系统时钟，那么该单片机有两种参考时基提供给定时/计数器。这个内部时钟源来自 f_{sys} 或 f_{RTC}（当系统时钟选内部的 RC＋RTC 模式时），可在掩模时选择。外部时钟输入，允许用户去计量外部事件，测量时间长度或脉冲宽度或产生一个精确的时基和 PFD 信号。

符号	位	功能
PSC0~PSC2	0-2	定义前置分频器因子，PSC2，PSC1，PSC0= 000：f_{INT}= fsys/2 001：f_{INT}= fsys/4 010：f_{INT}= fsys/8 011：f_{INT}= fsys/16 100：f_{INT}= fsys/32 101：f_{INT}= fsys/64 110：f_{INT}= fsys/128 111：f_{INT}= fsys/256
TE	3	定义定时/计数器 TMR 的边缘触发方式: 0 = 电平由低到高有效 1 = 电平由高到低有效
TON	4	允许/禁止计数(1=允许 0=禁止)
—	5	未用，读出为 0
TM0 TM1	6 7	定义操作模式 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

TMR 寄存器



有两个寄存器与定时/计数器相关联，即 TMR（[0DH]）和 TMRC（[0EH]）。有两个物理寄存器对应 TMR 的位置。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 则会获得定时/计数器的内容。TMRC 是定时/计数器控制寄存器。

TM0 和 TM1 位定义操作模式。事件计数模式用来记录外部事件，这意味着时钟来自外部 TMR 引脚。定时器模式是作为一个普通的定时器功能，时钟源来自 f_{INT} 时钟或是 RTC 时钟。脉冲宽度测量模式能用来计算外部引脚 TMR 上的高电平或低电平的宽度。计数是基于 f_{INT} 时钟或者 RTC 时钟。

在事件计数或定时器模式中，一旦定时/计数器开始计数，它将会从当前定时/计数器中的数值开始计数到 FFH。一旦产生溢出，计数器会从定时/计数器预置寄存器重新装载并且同时产生相应的中断请求状态位（TF ；INTC 的第 5 位）。

在脉冲宽度测量中，将 TON 和 TE 置为“1”，如果 TMR 接收到从低到高的电平跳变（或从高到低的变化，如果 TE 位被清零），就开始计数直到 TMR 返回到原来的电平并且复位 TON 位。测量的结果被保留在定时/计数器中，甚至电平跳变再一次发生也不会改变。换句话说，一次只能测量一个周期。直到 TON 再次被置位，只要再接到跳变信号，那么测量过程会再次执行。要注意在这个操作模式中，定时/计数器的启动计数不是根据逻辑电平，而是依据信号的边缘跳变触发。一旦发生计数器溢出，计数器会从定时/计数器的预置寄存器重新装入，并引发出中断请求，这种情况与其另外两个模式一样。要使得计数运行，只要将定时器启动位（TON；TMRC 的第 4 位）置 1。在脉宽测量模式中，TON 在测量周期结束后自动被清零。但在另外两个模式中，TON 只能由指令来复位。定时/计数器的溢出是唤醒的信号之一。不管任何模式，若写 0 到 ETI 位即可禁止相应的中断服务。

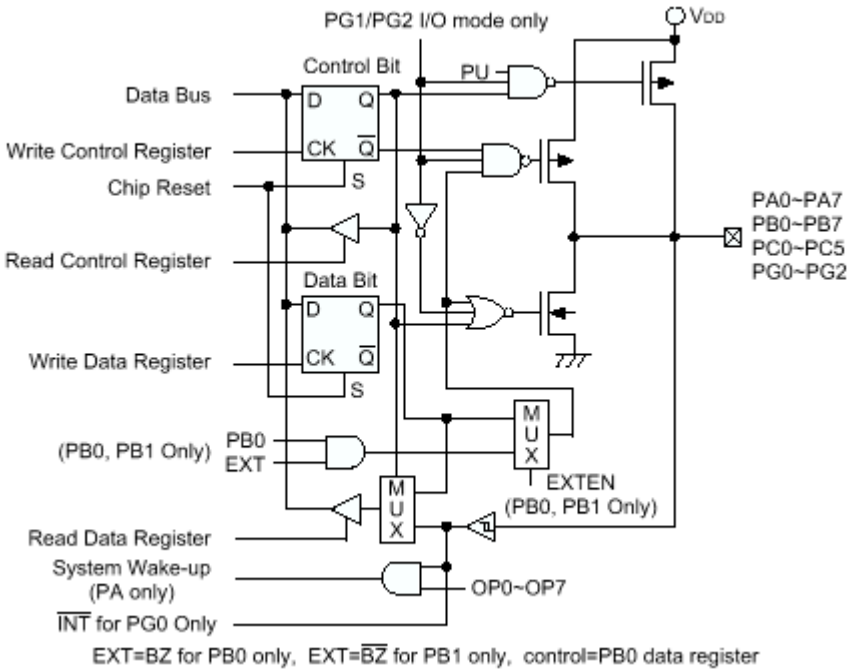
在定时/计数器为 0FF 的状态下，写数据到定时/计数器的预置寄存器之中，同时也会将数据装入定时/计数器中。但若是定时/计数器已经开启，写到定时/计数器的数据只会被保留在定时/计数器的预置寄存器中，直到定时/计数器发生计数溢出为止，再由预置寄存器加载新的值。当定时/计数器的数据被读取时，会禁止时钟输入以防出错。因为禁止时钟输入可能导致计数错误，所以程序员必须仔细加以考虑。

TMRC 的 0~2 位被用于定义定时/计数器的内部时钟源的前置分频因子。定义如表所示。定时/计数器的溢出信号被用于产生驱动蜂鸣器的 PFD 信号。

输入/输出端口

单片机具有 25 个双向输入输出口，标号从 PA 到 PC 以及 PG，其分别对应的 RAM 的[12H]，[14H]，[16H]和[1EH]。所有的 I/O 端口都能被作为输入或输出使用。就输入而言这些端口不具有锁存功能，即，输入数据必须在“MOV A，[m]”(m=12H,14H，16H 或 1EH) 指令的 T2 上升沿被准备好。对输出而言，所有的数据被锁存并保持不变，直到输出锁存器重新被改写。

每个 I/O 口都有其自己的控制寄存器（PAC，PBC，PCC，PGC），用来控制输入/输出的设置。使用控制寄存器，可对 CMOS 输出或带或不带上拉电阻斯密特触发输入在软件下动态地进行改变。要设置为输入功能，相应的控制寄存器必须写“1”。信号源的输入也取决于控制寄存器。如果控制寄存器的某位值为“1”那么输入信号是读取自这个引脚（PAD）的状态，但是如果控制寄存器的某位值为“0”，那么锁存器的内容将会被送到内部总线。后者，可以在“读改写”指令中发生。



对于输出功能，只能设置为 CMOS 输出。这些控制寄存器是对应于内存的 13H，15H，17H 和 1FH 地址。

芯片复位后，这些输入/输出口都会是高电平或浮空状态（取决于上拉电阻的选项）。每一个输入/输出锁存位都能被 SET [m].i 或 CLR [m].i 指令置位或清零，（m=12H，14H，16H 或 1EH。）

某些指令会首先输入数据然后进行输出操作。例如，SET [m].i ，CLR [m].i，CPL [m]和 CPLA [m]指令，读取输入口的状态到 CPU，执行这个操作（位操作），然后将数据写回锁存器或累加器。

PA 的每一个口都具有唤醒系统的能力。PC 端口的高 6 位和 PB 端口的高 5 位在物理上是不存在的；读这些位将返回“0”，而写这些位结果为无操作。请看应用注释。

所有的 I/O 口都可以有上拉电阻的选择。一旦选择上拉电阻，所有的 I/O 口都具有上拉电阻。必须要注意的是：没有上拉电阻的 I/O 口工作在输入模式会产生浮空状态。

PB0 和 PB1 分别与 BZ 和$\overline{BZ}$管脚复用。如果 BZ/$\overline{BZ}$的选项被选择，PB0/PB1 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。在输入模式始终保持它的原来的功能。一旦 BZ/$\overline{BZ}$的选项被选择，蜂鸣器的输出信号只受 PB0 数据寄存器控制。PB0/PB1 的 I/O 功能如下所示：

PB0 I/O	I	I	I	I	O	O	O	O	O	O
PB1 I/O	I	O	O	O	I	I	I	O	O	O
PB0 模式	X	X	C	B	B	C	B	B	B	B
PB1 模式	X	C	X	X	X	C	C	C	B	B
PB0 数据	×	×	0	1	D	0	1	D_0	0	1
PB1 数据	×	D	×	×	×	×	×	D_1	×	×
PB0 Pad 状态	I	I	I	I	D	0	B	D_0	0	B
PB1 Pad 状态	I	D	0	B	I	I	I	D_1	0	B

注释： I：输入；O：输出；D，D0，D1 ：数据；
B：蜂鸣器的选项，BZ 或$\overline{BZ}$ ；×：任意值；
C：CMOS 输出；

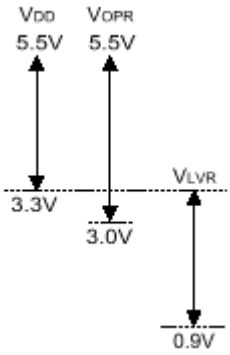
PG0 与$\overline{INT}$管脚复用。

如果在“内部 RC 振荡＋I/O”方式时，PG1、PG2 分别与 OSC1、OSC2 共用引脚。一旦选择了“内部 RC 振荡＋I/O”模式，PG1、PG2 就可以作为通用 I/O 口使用；否则 PG1 和 PG2 上的上拉电阻及 I/O 功能就被除能。

为了避免在浮空状态下功耗太大，建议将未用的或没有连结到外部的 I/O 口由软件指令设置成输出引脚。

低电压复位 （LVR）

为了监控器件的工作电压，单片机提供低电压复位电路。如果器件的工作电压在 0.9V～V_{LVR}V 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

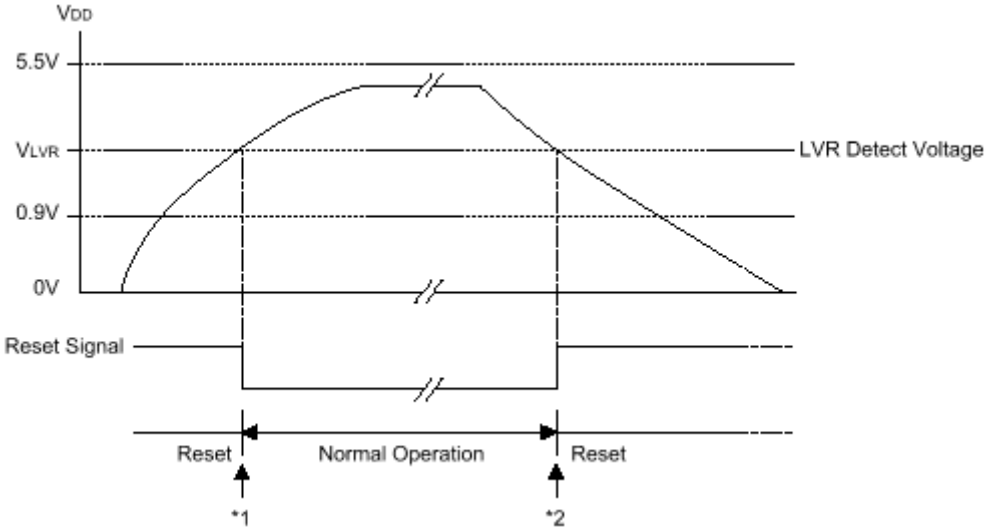


附注：VOPR 是在系统时钟为 4MHz 时，使得芯片正常运行的电压域值

LVR 具有下列功能说明：

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

VDD 与 VLVR 之间的关系如下所示：



低电压复位

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

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

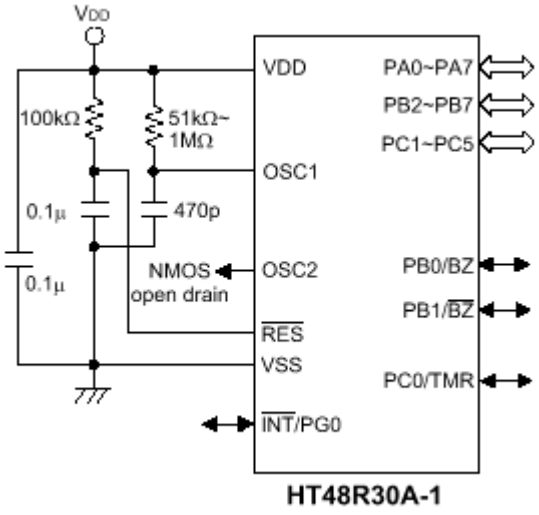
掩模选项

下表示出了这种单片机的各种类型的掩模选项。为了满足系统功能，所有的掩模选项必须正确定义。

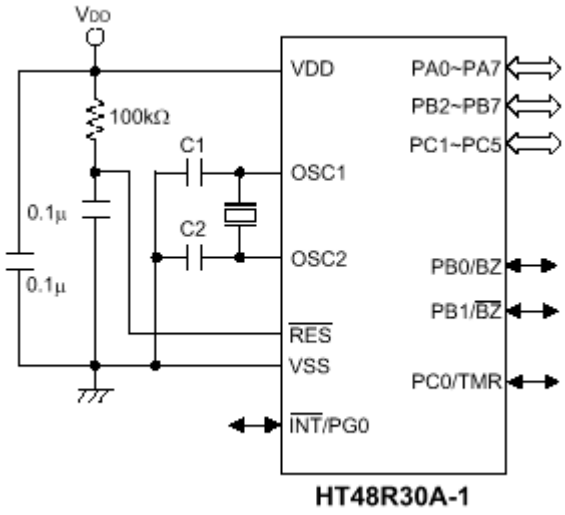
编号	选　　项
1	WDT 时钟源：WDTOSC/F_{tiD}/RTC OSC/除能
2	CLR WDT 指令：1 或 2 条指令
3	定时/计数器时钟来源：f_{SYS} 或 RTC OSC
4	PA 口唤醒（位）
5	PA 口 CMOS/斯密特输入
6	上拉电阻（PA~PC，PG）：无上拉电阻或有上拉电阻
7	BZ 选项：允许/禁止
8	LVR：允许/禁止
9	系统振荡器：外部 RC 振荡/外部石英振荡/内部 RC+RTC 或内部 RC+PG1/PG2
10	内部 RC 振荡频率选择：3.2MHz，1.6MHz，800kHz 或 600kHz
11	锁定：不锁定/锁定

应用电路

RC 振荡的多 I/O 口应用

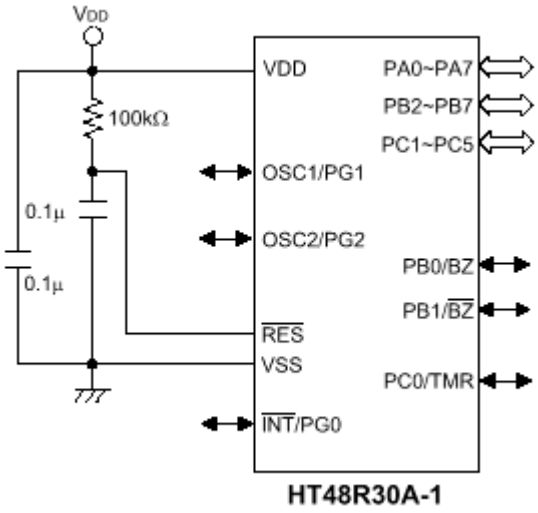


石英或陶瓷振荡器的多 I/O 口应用

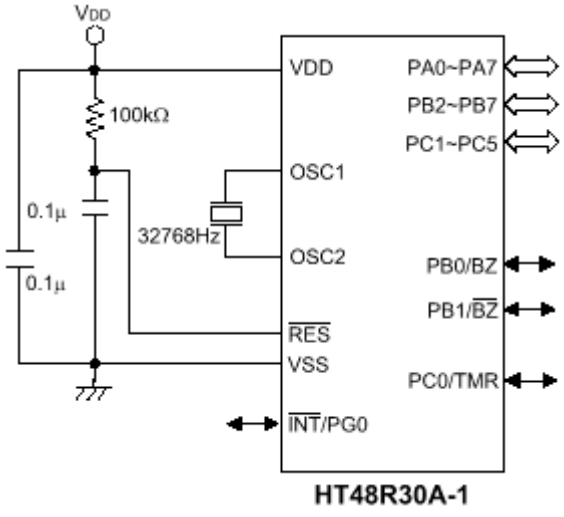


注意：如果 Fsys<1MHz　则 C1 = C2 = 300pF
否则 C1 = C2 = 0

内部 RC 振荡的多 I/O 口应用



内部带 RTC 的 RC 振荡的多 I/O 口应用



注意：为了确保 VDD 的稳定以及在将$\overline{\text{RES}}$引脚拉高之前保证操作电压在一定的范围内，复位电路的电阻和电容要按上设计

注：本文译至 July 02，2001 的 48r30a_1.pdf 文档