

盛群知识产权政策

专利权

盛群半导体公司在全球各地区已核准和申请中之专利权至少有 160 件以上，享有绝对之合法权益。与盛群公司 MCU 或其它产品有关的专利权并未被同意授权使用，任何经由不当手段侵害盛群公司专利权之公司、组织或个人，盛群将采取一切可能的法律行动，遏止侵权者不当的侵权行为，并追讨盛群公司因侵权行为所受之损失、或侵权者所得之不法利益。

商标权

盛群之名称和标识、Holtek 标识、HT-IDE、HT-ICE、Marvel Speech、Music Micro、Adlib Micro、Magic Voice、Green Dialer、PagerPro、Q-Voice、Turbo Voice、EasyVoice 和 HandyWriter 都是盛群半导体公司在台湾地区和其它国家的注册商标。

著作权

Copyright © 2008 by HOLTEK SEMICONDUCTOR INC.

规格书中所出现的信息在出版当时相信是正确的，然而盛群对于规格内容的使用不负责任。文中提到的应用其目的仅仅是用来做说明，盛群不保证或不表示这些应用没有更深入的修改就能适用，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。盛群产品不授权使用于救生、维生器件或系统中做为关键器件。盛群拥有不事先通知而修改产品的权利，对于最新的信息，请参考我们的网址 <http://www.holtek.com.tw>; <http://www.holtek.com.cn>

技术相关信息

- [工具信息](#)
- [FAQs](#)
- [应用范例](#)
 - [HA0003S HT48 & HT46 MCU 与 HT93LC46 的通信](#)
 - [HA0016S HT48读写 HT24 EEPROM](#)
 - [HA0018S HT48控制 HT1621 LCD](#)
 - [HA0049S HT48读写控制 HT1380](#)

特性

- 工作电压：
 - fsys = 4MHz : 2.2V – 5.5V
 - fsys = 8MHz : 3.3V – 5.5V
 - fsys = 12MHz : 4.5V – 5.5V
- 7 个双向输入/输出口和一个输入口
- 与输入/输出引脚复用的外部中断输入
- 4 种振荡方式
 - 外部晶体振荡器
 - 外部 RC 振荡器
 - 内部 RC+输入/输出口 (PA5,PA6)
 - 内部 RC+RTC 振荡 (32768Hz)
- 内部 RC 振荡器
 - 三种频率选择: 4MHz/8MHz/12MHz
 - 4MHz 有±5%的误差(2.7V~5.5V,25°C)
 - 8MHz 有±5%的误差(3.3V~5.5V,25°C)
 - 12MHz 有±5%的误差(4.5V~5.5V,25°C)
- 看门狗定时器
- 程序空间: 最大 4096×15
- 数据空间: 最大 160×8
- 支持蜂鸣器驱动和 PFD
- HALT 和唤醒功能可降低功耗
- 在 VDD=5V, 系统时钟为 8MHz 时, 指令周期为 0.5 μs
- 所有指令在 1 或 2 个指令周期内完成
- 查表指令, 表格内容字长 14 位或 15 位
- 8 层硬件堆栈
- 位操作指令
- 低电压复位功能
- 10-pin MSOP 封装

概述

HT48R01/HT48R02/HT48R03 是一款八位高性能精简指令集单片机, 专为经济型多输入输出控制的产品设计。

拥有低功耗、I/O 口灵活、定时功能、振荡选择、省电和唤醒功能、看门狗定时器、蜂鸣器驱动、以及低价位等优势, 使此款多功能芯片可以广泛地适用于各种应用, 例如工业控制、消费类产品、子系统控制器等。

选型表

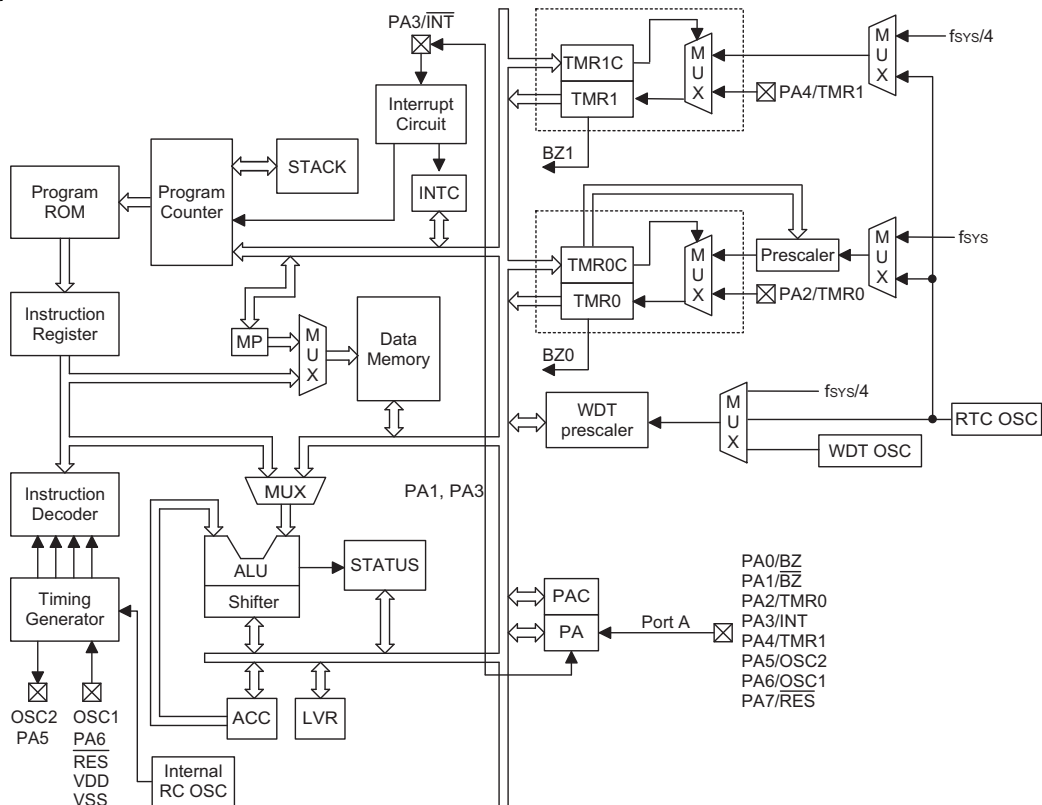
| 型号 | VDD | 程序存储器 | 数据存储器 | 输入/输出 | 定时器 | 外部中断 | 蜂鸣器 | 堆栈 | 封装种类 |
|---------|-----------|-------|-------|-------------------|---------|------|-----|----|--------|
| HT48R01 | 2.2V~5.5V | 1K×14 | 64×8 | 7 个输入输出口 一个输入口 | 8-bit×1 | 1 | √ | 4 | 10MSOP |
| HT48R02 | 2.2V~5.5V | 2K×14 | 96×8 | 7 个输入输出口 一个输入口 | 8-bit×2 | 1 | √ | 6 | 10MSOP |
| HT48R03 | 2.2V~5.5V | 4K×15 | 160×8 | 7 个输入输出口 一个输入口 | 8-bit×2 | 1 | √ | 8 | 10MSOP |

设备型号

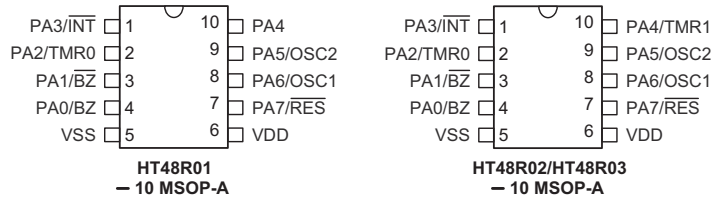
HT48R01/02/03 分别具有三种不同的内部 RC 振荡频率 4MHz、8MHz 和 12MHz 分别用后缀 1、2 和三表示，如下表所示：

| 型号 | 设备型号 | 内部 RC 振荡频率 |
|---------|-----------|------------|
| HT48R01 | HT48R01-1 | 4MHz |
| | HT48R01-2 | 8MHz |
| | HT48R01-3 | 12MHz |
| HT48R02 | HT48R02-1 | 4MHz |
| | HT48R02-2 | 8MHz |
| | HT48R02-3 | 12MHz |
| HT48R03 | HT48R03-1 | 4MHz |
| | HT48R03-2 | 8MHz |
| | HT48R03-3 | 12MHz |

方框图



引脚图



引脚说明

| 引脚名称 | 输入输出 | 掩膜选项 | 说明 |
|----------------------|----------|------------------------------------|--|
| PA0/BZ PA1/BZ | 输入/输出 | — | 2 位双向输入/输出口。每位脚可通过软件设置为唤醒输入。可由软件指令设置为 CMOS 输出或斯密特触发输入。可通过软件指令设置上拉电阻。PA0/PA1 分别与 BZ 和 BZ 复用 |
| PA2/TMR0 | 输入/输出 | — | 1 位双向输入/输出口。PA2 可通过软件指令设置为唤醒输入。可由软件指令设置为 CMOS 输出或斯密特触发输入。可通过软件设置上拉电阻。此脚与 TMR0 复用。 |
| PA3/INT | 输入/输出 | — | 1 位双向输入/输出口。PA2 可通过软件指令设置为唤醒输入。可由软件指令设置为 CMOS 输出或斯密特触发输入。可通过软件设置上拉电阻。此脚与 INT 复用。 |
| PA4/TMR1 | 输入/输出 | — | 1 位双向输入/输出口。PA4 可通过软件指令设置为唤醒输入。可由软件指令设置为 CMOS 输出或斯密特触发输入。可通过软件设置上拉电阻。此脚与 TMR1 复用。 |
| OSC1/PA6 OSC2/PA5 | 输入 输出 | RC 振荡, 晶体振荡 RTC 振荡或 输入输出口 | 2 位双向输入/输出口或振荡器引脚。如设定为输入/输出口, 可通过软件指令设置为 CMOS 输出或斯密特触发输入。可通过软件设置上拉电阻。可通过掩膜选项决定是振荡器模式或输入/输出口。四种振荡模式为: 1. 内部 RC 晶振: 这两个脚同时被设置为输入/输出口 2. 外部晶体振荡: 这两个脚同时被设置为 OSC1/OSC2 3. 内部 RC + RTC 晶振: 这两个脚同时被设置为 OSC1/OSC2 4. 外部 RC 晶振 + PA5: PA6 被设置为晶振输入脚, PA5 被设置为输入/输出口 注: 当系统频率是内部 RC 晶振时, 频率可有三种选择: (12MHz,8MHz,4MHz), 由设备型号决定。 |
| PA7/RES | 输入 | PA7 或 RES | PA7 输入或斯密特触发复位输入 (低电平有效)。 |
| VDD | — | — | 正电源 |
| VSS | — | — | 负电源, 接地 |

* 所有的上拉电阻通过寄存器选项控制。

极限参数

电源供应电压 Vss -0.3V 至 Vss +6.0V
 端口输入电压 Vss -0.3V 至 VDD +0.3V
 IOL 总电流.....150mA
 总消耗电流.....500mW

储存温度 -50℃ 至 125℃
 工作温度 -40℃ 至 85℃
 IOH 总电流..... -100mA

注意: 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。

D.C.特性

Ta=25°C

| 符号 | 参数 | 测试条件 | | 最小 | 典型 | 最大 | 单位 |
|-------------------|--------------------------------|-----------------|--------------------------------|--------------------|------|--------------------|-----|
| | | V _{DD} | 条件 | | | | |
| V _{DD} | 工作电压 | — | f _{sys} =4MHz | 2.2 | — | 5.5 | V |
| | | — | f _{sys} =8MHz | 3.3 | — | 5.5 | V |
| | | — | f _{sys} =12MHz | 4.5 | — | 5.5 | V |
| I _{DD1} | 工作电流 (晶体振荡, RC 振荡) | 3V | 无负载 | — | 1 | 2 | mA |
| | | 5V | f _{sys} =4MHz | — | 2.5 | 5 | |
| I _{DD2} | 工作电流 (晶体振荡, RC 振荡) | 3V | 无负载 | — | 2 | 4 | mA |
| | | 5V | f _{sys} =8MHz | — | 4 | 8 | |
| I _{DD3} | 工作电流 (晶体振荡, RC 振荡) | 5V | 无负载 f _{sys} =12MHz | — | 6 | 12 | mA |
| I _{DD4} | 工作电流 (内部 RC + RTC 振荡, 正常模式) | 3V | 无负载 | — | 1 | 2 | mA |
| | | 5V | f _{sys} =4MHz | — | 2.5 | 5 | mA |
| I _{DD5} | 工作电流 (内部 RC + RTC 振荡, 正常模式) | 3V | 无负载 | — | 2 | 4 | mA |
| | | 5V | f _{sys} =8MHz | — | 4 | 8 | mA |
| I _{DD6} | 工作电流 (内部 RC + RTC 振荡, 正常模式) | 5V | 无负载 f _{sys} =12MHz | — | 6 | 12 | mA |
| I _{DD7} | 工作电流 (内部 RC + RTC 振荡, 低速模式) | 3V | 无负载 | — | 10 | 20 | μ A |
| | | 5V | f _{sys} =32768Hz | — | 20 | 40 | μ A |
| I _{STB1} | 静态电流 (看门狗打开, RTC 关闭) | 3V | 无负载 暂停模式 | — | — | 5 | μ A |
| | | 5V | | — | — | 10 | |
| I _{STB2} | 静态电流 (看门狗关闭, RTC 关闭) | 3V | 无负载 暂停模式 | — | — | 1 | μ A |
| | | 5V | | — | — | 2 | |
| I _{STB3} | 静态电流 (看门狗关闭, RTC 打开) | 3V | 无负载 暂停模式 | — | — | 5 | μ A |
| | | 5V | | — | — | 10 | |
| V _{IL1} | PA0~PA6,TMR0,TMR1, INT 低电平输入电压 | — | — | 0 | — | 0.3V _{DD} | V |
| V _{IH1} | PA0~PA6,TMR0,TMR1, INT 高电平输入电压 | — | — | 0.7V _{DD} | — | V _{DD} | V |
| V _{IL2} | 低电平输入电压 (PA7/RES) | — | — | 0 | — | 0.4V _{DD} | V |
| V _{IH2} | 高电平输入电压 (PA7/RES) | — | — | 0.9V _{DD} | — | V _{DD} | V |
| V _{LVR1} | 低电压复位 1 | — | 掩模选项: 4.2V | 3.98 | 4.2 | 4.42 | V |
| V _{LVR2} | 低电压复位 2 | — | 掩模选项: 3.15V | 2.98 | 3.15 | 3.32 | V |

| | | | | | | | |
|-------------------|-----------|----|-------------------------------------|------|-----|------|----|
| V _{LVR3} | 低电压复位 3 | — | 掩模选项: 2.1V | 1.98 | 2.1 | 2.22 | V |
| I _{OL} | 输入/输出口灌电流 | 3V | V _{OL} =0.1V _{DD} | 4 | 8 | — | mA |
| | | 5V | | 10 | 20 | — | |
| I _{OH} | 输入/输出口源电流 | 3V | V _{OH} =0.9V _{DD} | -2 | -4 | — | mA |
| | | 5V | | -5 | -10 | — | |
| R _{PH} | 上拉电阻 | 3V | — | 20 | 60 | 100 | kΩ |
| | | 5V | | 10 | 30 | 50 | |

A.C.特性

Ta=25°C

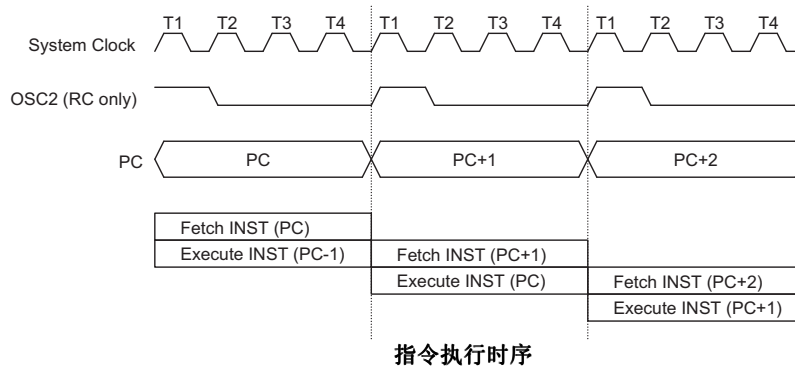
| 符号 | 参数 | 测试条件 | | 最小 | 典型 | 最大 | 单位 |
|---------------------|---------------------|-----------------|---------------|-------|-------|-------|------------------|
| | | V _{DD} | 条件 | | | | |
| f _{SYS1} | 系统时钟（晶体振荡,RC 振荡） | — | 2.2V~5.5V | 400 | — | 4000 | kHz |
| | | — | 3.3V~5.5V | 400 | — | 8000 | |
| | | — | 4.5V~5.5V | 400 | — | 12000 | |
| f _{SYS2} | 系统时钟（内部 RC 振荡）(±5%) | 4.5V~5.5V | 12MHz,Ta=25°C | 11400 | 12000 | 12600 | kHz |
| | | 3.3V~5.5V | 8MHz,Ta=25°C | 7600 | 8000 | 8400 | |
| | | 2.7V~5.5V | 4MHz,Ta=25°C | 3800 | 4000 | 4200 | |
| f _{SYS3} | 系统时钟（32768Hz） | — | — | — | 32768 | — | Hz |
| f _{TIMER} | 定时器输入频率 (TMR) | — | 2.2V~5.5V | 0 | — | 4000 | KHz |
| | | — | 3.3V~5.5V | 0 | — | 8000 | |
| | | — | 4.5V~5.5V | 0 | — | 12000 | |
| t _{WDTOSC} | 看门狗振荡周期 | 3V | — | 45 | 90 | 180 | μs |
| | | 5V | | 32 | 65 | 130 | |
| t _{RES} | 外部复位低电平脉宽 | — | — | 1 | — | — | μs |
| t _{SST} | 系统启动延时周期 | — | HALT 模式唤醒 | — | 1024 | — | t _{sys} |
| t _{LVR} | 低电压复位周期 | — | — | 0.25 | 1 | 2 | ms |
| V _{POR} | 上电复位电压 | — | — | — | — | 100 | mV |
| R _{POR} | 上电复位电压速率 | — | — | 0.035 | — | — | V/ms |

 备注: t_{sys}=1/f_{SYS1} , 1/f_{SYS2}或1/f_{SYS3}

系统功能说明

指令系统

系统时钟由晶体振荡器或 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。



指令执行时序

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。

程序计数器 (PC)

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 | | | | | | | | | | | |
|------------|---------|-----|----|----|----|----|----|----|----|----|----|----|
| | *11 | *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 | 0 |
| 外部中断 | 0 | 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 | 0 | 1 | 0 | 0 | 0 |
| 条件跳越 | PC+2 | | | | | | | | | | | |
| 装入 PCL | *11 | *10 | *9 | *8 | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |
| 跳转、子程序调用 | #11 | #10 | #9 | #8 | #7 | #6 | #5 | #4 | #3 | #2 | #1 | #0 |
| 从子程序返回 | S11 | S10 | S9 | S8 | S7 | S6 | S5 | S4 | S3 | S2 | S1 | S0 |

程序计数器

注意：*11 ~ *0：程序计数器位
 S11 ~ S0：堆栈寄存器位
 #11 ~ #0：指令代码位
 @7 ~ @0：PCL 位
 对 HT48R01 来说，PC 指针是 10 位，从 *9~*0
 对 HT48R02 来说，PC 指针是 11 位，从 *10~*0
 对 HT48R03 来说，PC 指针是 12 位，从 *11~*0

程序存储器 (ROM)

程序存储器被用来存放要执行的指令代码还包括数据、表格和中断入口。HT48R01 为 1024×14 位, HT48R02 为 2048×14 位或 HT48R03 为 4096×15 位, 可以由程序计数器或表格指针来寻址。

在程序存储器中某几个地址被保留作为特殊用途。

- 地址 000H

此地址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

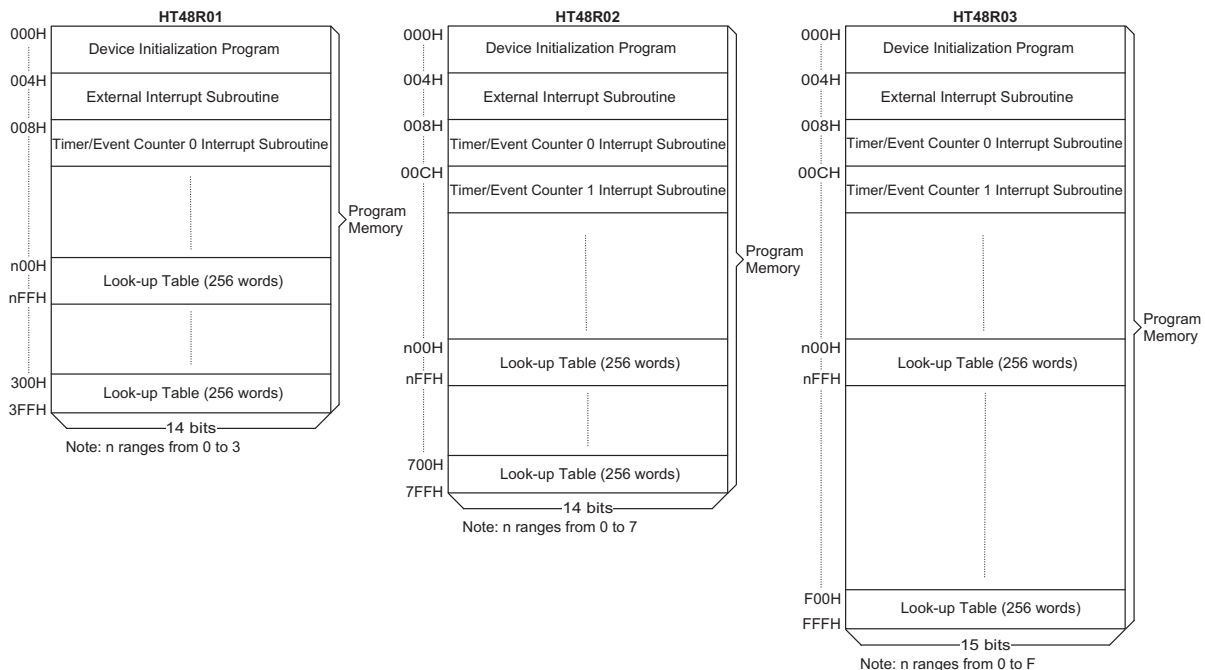
该地址为外部中断服务程序保留。当 $\overline{\text{INT}}$ 引脚有触发信号输入, 如果中断允许且堆栈未滿, 则程序会跳转到 004H 地址开始执行。服务程序。

- 地址 008H

此地址保留给定时/计数器 0 中断服务使用。当定时/计数器 0 溢出, 如果中断允许且堆栈未滿, 则程序会从 008H 地址开始执行中断服务程序。

- 地址 00CH

此地址保留给定时/计数器 1 中断服务使用。当定时/计数器 1 溢出, 如果中断允许且堆栈未滿, 则程序会从 00CH 地址开始执行中断服务程序。


程序寄存器

- 表格区 (Table Location)

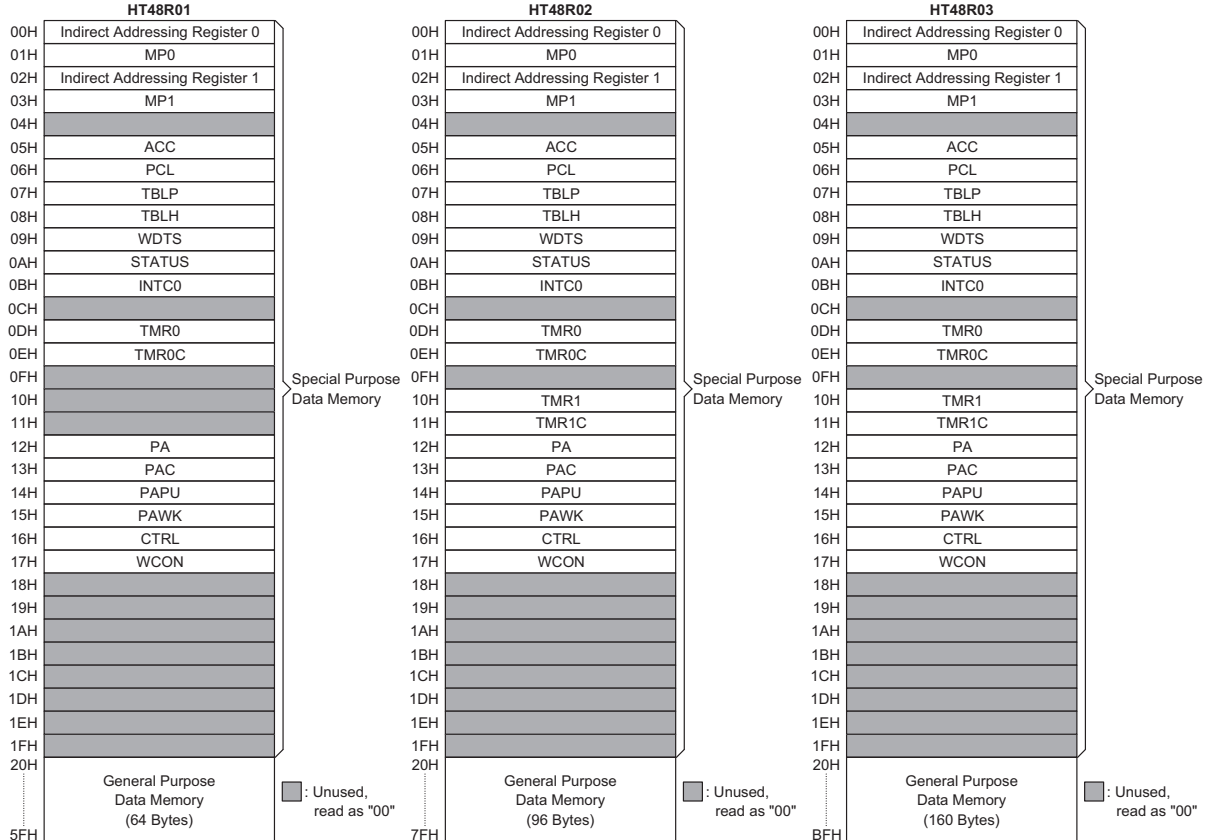
ROM 内的任何地址都可被用来作为查表地址使用。查表指令为 TABRDC [m] 与 TABRDL [m]。TABRDC [m]是查表当前页的数据 [1 页=256 个字 (word)]。TABRDL [m]是查表最后一页的数据。[m] 为数据被存入的地址。在执行 TABRDC [m]指令 (或 TABRDL [m] 指令) 后, 将会传送当前页 (或最后一页) 上的一个字的低位字节到[m], 而这个字的高位字节传送到 TBLH (08H)。只有表格中的低位字节被定义到目标地址中, 而高位字节传送到表格的高位字节寄存器 (TBLH), TBLH 高 2 位读数为“0”。TBLH 为只读寄存器。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低位地址。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

算术逻辑单元 (ALU)

算术逻辑单元是执行 8 位算术逻辑运算的电路。它提供如下的功能：

- 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)
- 递增和递减运算 (INC, DEC)
- 分支跳转 (SZ, SNZ, SIZ, SDZ 等)

算术逻辑单元 ALU 不仅会保存运算的结果而且会改变状态寄存器。



数据存储器

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）包含了中断控制位和中断请求标志，中断控制位用来设置中断允许/禁止。

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 脚上的下降沿触发的，相关的中断请求位（EIF，INTC.4）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 位也将被清除，以禁止其他中断发生。

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；INTC 的第 5 位）会被置位。当中断允许，堆栈又未滿，当发生定时/计数器中断时，就会产生地址 08H 的子程序调用。该中断请求标志位（TF）和 EMI 位将被清除，以便禁止其他中断。

在执行中断子程序期间，其他的中断响应会被屏蔽，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（当堆栈是未滿时）。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 则不会。

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。这种优先等级也可以通过 EMI 位的复位来屏蔽。

| NO | 中断源 | 优先级 | 中断 |
|----|----------|-----|-----|
| 1 | 外部中断 | 1 | 04H |
| 2 | 定时/计数器中断 | 2 | 08H |

HT48R01 中断向量

| NO | 中断源 | 优先级 | 中断 |
|----|------------|-----|-----|
| 1 | 外部中断 | 1 | 04H |
| 2 | 定时/计数器中断 0 | 2 | 08H |
| 3 | 定时/计数器中断 1 | 3 | 0CH |

HT48R02/03 中断向量

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，因为中断随时都可能发生，而且在许多应用场合需要立刻给予响应。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。

| INTC0 (0BH) | 位 | 符号 | 功 能 |
|----------------|---|------------|-------------------------|
| | 0 | EMI | 总中断控制位(1=允许, 0=禁止) |
| | 1 | E EI | 外部中断控制位(1=允许, 0=禁止) |
| | 2 | ETOI | 定时/计数器中断控制位(1=允许, 0=禁止) |
| | 3 | — | 未用, 读出为零 |
| | 4 | EIF | 外部中断请求标志位(1=有, 0=无) |
| | 5 | TF | 定时/计数器中断请求位(1=有, 0=无) |
| | 6 | — | 未使用位, 读出为零 |
| 7 | — | 未使用位, 读出为零 | |

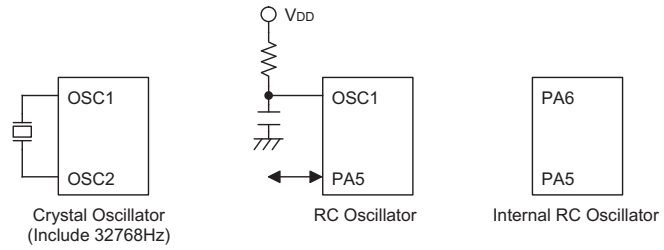
HT48R01 中断控制寄存器—INTC0 (0BH)

| INTC1 (0BH) | 位 | 符号 | 功 能 |
|----------------|---|------------|----------------------------|
| | 0 | EMI | 总中断控制位(1=允许, 0=禁止) |
| | 1 | E EI | 外部中断控制位(1=允许, 0=禁止) |
| | 2 | ETOI | 定时/计数器 0 中断控制位(1=允许, 0=禁止) |
| | 3 | ETI1 | 定时/计数器 1 中断控制位(1=允许, 0=禁止) |
| | 4 | EIF | 外部中断请求标志位(1=有, 0=无) |
| | 5 | T0F | 定时/计数器 0 中断请求位(1=有, 0=无) |
| | 6 | T1F | 定时/计数器 1 中断请求位(1=有, 0=无) |
| 7 | — | 未使用位, 读出为零 | |

HT48R02/HT48R03 中断控制寄存器—INTC1 (0BH)

振荡器

通过掩膜选项，可以选择外部 RC 振荡 (ERC)，外部晶体振荡 (ECRY)，内部 RC 振荡加输入输出和内部 RC 振荡加 RTC 晶振。四种都可作为系统时钟。不管用哪种振荡器，其信号都支持系统时钟。HALT 模式会停止系统振荡器，除 RTC 振荡并忽略任何外部信号，由此来节省功耗。



如果采用内部 RC+RTC 振荡方式，系统支持两种不同的系统时钟。当系统进入 HALT 模式，可选择三种不同工作模式。两种系统频率可通过 CTRL 寄存器的 CLKMOD 位来选择为内部 RC 振荡或 RTC 晶振 (32768Hz)。三种工作模式是正常模式，低速模式或 HALT 模式。下面的表格说明了关系：

| HALT 指令 | CLKMOD | RC 振荡 | 32768Hz | 系统时钟 | 模式 |
|------------------|--------|-------|---------|---------|------|
| 在运行状态 (不运行 HALT) | 0 | On | On | RC 振荡 | 正常 |
| | 1 | Off | On | 32768Hz | 慢速 |
| 在运行状态 (运行 HALT) | × | Off | On | HALT | HALT |

如果采用 RC 振荡方式，那么在 VDD 与 OSC1 之间要接一个外部电阻。其阻值范围为 24K Ω ~1.5M Ω 。在 OSC2 端可获得系统时钟四分频信号，它可以用于同步系统外部逻辑。RC 振荡器是一种低成本方案，但是振荡频率由于 VDD，温度及芯片自身参量的漂移而产生误差。因此，对计时敏感场合，要求精确度高的振荡器频率，RC 振荡器是不适用的。

如果选用的是晶体振荡器，那么在 OSC1 和 OSC2 之间需要连接一个晶体，用来提供晶体振荡器所需要的反馈和相移。另外，在 OSC1 和 OSC2 之间还可以用谐振器代替晶体振荡器，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输入/输出引脚或者是作为 32768Hz 晶体连接口 (RTC OSC)。内部的 RC 振荡器的频率可以选择 4MHz、8MHz 和 12MHz (由设备型号选择)。

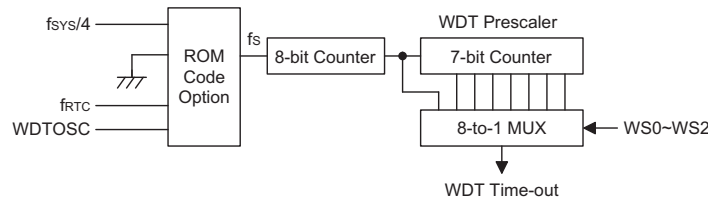
WDT 振荡器是 IC 内部 RC 型振荡器，不需要任何外部元件。即使在系统进入暂停模式，系统时钟被停止，但这个 RC 振荡器仍会运作。在 5V 其振荡周期大约为 65 μ s。在掩膜时，如欲节省电源，可在掩膜选项中关闭 WDT 振荡器。

看门狗定时器 (WDT)

WDT 的时钟源是有三种：看门狗振荡器 (WDT 振荡器)、RTC 振荡器或是指令时钟 (系统时钟 4 分频)，由掩膜选项设置。看门狗主要用来避免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。看门狗可用掩膜选项设置为打开或关闭，如果在关闭状态，所有的 WDT 指令都是没有作用的。RTC 时钟只有在内部 RC+RTC 模式时才能工作。

WDT 时钟 (f_s) 通过一个内部计数器提供更长的溢出时间。分频系数为 215。

如果选择了内部 WDT 振荡器 (RC 振荡周期一般为 $65\mu s$) 的话，这个频率会先除以 256 (8 级) 产生 $17ms/5V$ 的溢出时间，这个溢出时间会因为温度， V_{DD} ，以及芯片参数的漂移而变化。如果使用 WDT 的预分频器，则可实现延长 WDT 溢出时间。设置 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，其分频级数最大为 1: 128，得到最长的 WDT 溢出周期 $2.1s/5V$ 。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，WDT 会停止计数而失去保护功能。在这种情况下，只能由外部逻辑复位来重新启动系统。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。


看门狗定时器

如果单片机工作在干扰很大的环境中，那么强烈建议使用片内 RC 振荡器 (WDT OSC)，因为 HALT 模式会使系统时钟终止运作。

| 位 | 符号 | 功能 |
|-----|-------------|------------|
| 0~2 | WS0~ WS2 | WDT 频率分频选择 |
| 3~7 | — | 未定义，读出为“0” |

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”，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 预分频器) 可以有三种方法：外部复位 (低电平输入到 RES 端)，用软件指令和 HALT 指令。软件指令由 CLR WDT 和另一组指令 CLR WDT1 及 CLR WDT2 组成。这两组指令中，只能选取其中一种。选择的方式由掩膜选项的 CLR WDT 次数选项决定。如果“CLR WDT”被选择 (即 CLR WDT 次数为 1)，那么只要执行 CLR WDT 指令就会清除 WDT。在 CLR WDT1 和 CLR WDT2 被选择的情况下 (即 CLR WDT 次数为 2)，那么要执行二条指令才会清除 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 |

看门狗定时器预置寄存器(09H)

WDT 控制寄存器其中 4 位控制 WDT 的打开和关闭。WDT 的打开和关闭可通过掩模选项或 WDT 控制寄存器 (WDTEN[3: 0]=0101B) 来选择。

| 位 | 符号 | 功能 |
|-----|-------------------|---|
| 0~3 | WDTEN0~ WDTEN3 | 3~0 位, WDTEN3~WDTEN0=1010B: WDT 关闭 其他: 打开 (在干扰很大的环境中, 强烈建议使用 0101B 打开 WDT) |
| 4~5 | — | 未定义, 读出为 “0” |
| 6~7 | INTES0~ INTES1 | 外部中断边沿触发选择 (默认=10) 00: 关闭 01: 上升沿触发 10: 下降沿触发 11: 上升沿下降沿同时触发 |

WCON (17H) 寄存器

暂停模式 (HALT)

暂停模式是由 HALT 指令来实现的, 产生如下结果:

- 关闭系统振荡器, 但 WDT 振荡器继续工作 (如果 WDT 时钟来源是 WDT 振荡器)。
- RAM 及寄存器的内容保持不变。
- WDT 和 WDT 预分频器被清除并再次重新计数 (如果 WDT 时钟来源是 WDT 振荡器)。
- 所有的输入/输出口都保持其原先状态。
- PDF 标志位被置位, TO 标志位被清零。

外部复位、中断或 PA 口下降沿信号或 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”。通过检测 TO 和 PDF 标志, 即可了解系统复位的原因。PDF 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, 同时产生唤醒, 使得程序计数的 PC 和堆栈指针复位。其他都保持原状态。

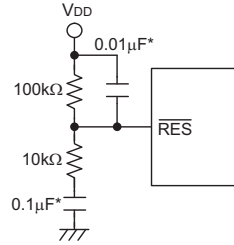
PA 口的唤醒和中断唤醒看作为正常运行的继续, PA 口的每一位都可以通过掩膜选项来设定为唤醒功能。如果唤醒是来自于输入/输出口的的信号变化, 程序会继续执行下一条命令。如果唤醒是来自中断的话, 则会产生二种情况: 如果相关的中断被禁止或中断是允许的, 但堆栈已满, 那么程序将继续执行下条指令, 如果中断允许并且堆栈未满, 那么这个中断响应就发生了。当唤醒事件发生时, 要花 1024tsys (系统时钟周期) 后, 系统重新正常运行。这就是说, 在唤醒后被插入了一个等待时间。如果唤醒是来自于中断响应, 那么实际的中断程序执行就被延迟了一个以上的周期。但是如果唤醒导致下一条指令执行, 那么在一个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/输出口的状态。但在 HALT 模式时, RTC 振荡器仍然运作 (如果选用 RTC 振荡器)。

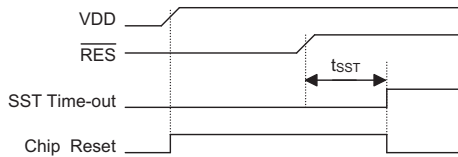
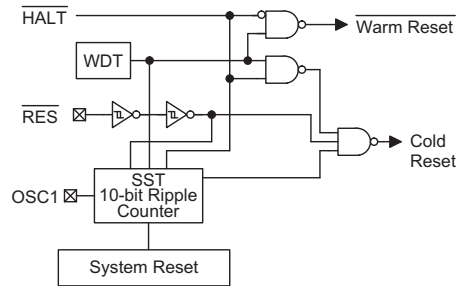
复位 (RESET)

有三种方法可以产生复位

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


REST 电路

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，热复位只复位程序计数器 PC 和堆栈指针 SP，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。


复位时序

复位电路结构

| TO | PDF | 复位条件 |
|----|-----|-------------------------------------|
| 0 | 0 | 电源上电复位 |
| u | u | 正常运作时由 $\overline{\text{RES}}$ 发生复位 |
| 0 | 1 | 由 $\overline{\text{RES}}$ 唤醒暂停模式 |
| 1 | u | 正常运作时看门狗定时器溢出 |
| 1 | 1 | 暂停模式看门狗定时器唤醒 |

注意：u 表示不变

为了保证系统振荡器起振并稳定运行，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，SST（系统启动定时器）会提供额外的 1024 个系统时钟周期的延迟。

系统复位时，SST 被加到复位延时中。任何来自 HALT 的唤醒都将产生 SST 延迟。

当系统上电、正常运行时 WDT 溢出或 $\overline{\text{RES}}$ 脚复位，系统需要额外增加一个加载掩模选项的时间。系统复位时各功能单元的状态如下所示：

| | |
|-----------|------------------|
| 程序计数器(PC) | 000H |
| 中断 | 禁止 |
| 预分频器 | 清除 |
| 看门狗定时器 | 清除，复位后看门狗定时器开始计数 |
| 定时/计数器 | 关闭 |
| 输入/输出口 | 输入模式 |
| 堆栈指针 | 指向堆栈的顶端 |

有关寄存器的状态如下：

| 寄存器 | 上电复位 | 正常运行期间 | | 暂停模式 | |
|-----------------------|-----------|-----------|-----------|-----------|-----------|
| | | WDT 溢出 | RES端复位 | RES端复位 | WDT 溢出* |
| PC | 000H | 000H | 000H | 000H | 000H |
| MP0 (HT48R01/02) | 1xxx xxxx | 1uuu uuuu | -uuu uuuu | -uuu uuuu | 1uuu uuuu |
| MP1 (HT48R01/02) | 1xxx xxxx | 1uuu uuuu | -uuu uuuu | -uuu uuuu | 1uuu uuuu |
| MP0 (HT48R03) | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| MP1 (HT48R03) | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu 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 |
| WDTS | ---- -111 | ---- -111 | ---- -111 | ---- -111 | ---- -uuu |
| STATUS | --00 xxxx | --1u uuuu | --uu uuuu | --01 uuuu | --11 uuuu |
| INTC0 (HT48R01) | --00 -000 | --00 -000 | --00 -000 | --00 -000 | --uu -uuu |
| INTC0 (HT48R02/03) | -000 0000 | -000 0000 | -000 0000 | -000 0000 | -uuu uuuu |
| TMR0 | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| TMR0C | 0000 1000 | 0000 1000 | 0000 1000 | 0000 1000 | uuuu uuuu |
| TMR1 | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| TMR1C | 0000 1--- | 0000 1--- | 0000 1--- | 0000 1--- | uuuu u--- |
| PA | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAC | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAPU | -000 0000 | -000 0000 | -000 0000 | -000 0000 | -uuu uuuu |
| PAWK | -000 0000 | -000 0000 | -000 0000 | -000 0000 | -uuu uuuu |
| CTRL | -0-- 0000 | -0-- 0000 | -0-- 0000 | -0-- 0000 | -u--uuuu |
| WCON | 10-- 1010 | 10-- 1010 | 10-- 1010 | 10-- 1010 | uu-- uuuu |

注意：“*”表示“热复位”。

“-”表示未定义

“U”表示不变化。

“×”表示不确定。

定时/计数器

HT48R01/02/03 提供一个或两个定时/计数器。该定时/计数器是一个 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信号输入、系统时钟或是 RTC 时钟。

如果外部时钟输入，可以用来计数外部事件、测量时间间隔、脉冲宽度或产生一个精确的时基。当采用内部时钟输入，产生一个精确的时基。

无论是外部还是内部时钟来源，定时/计数器可以产生蜂鸣器信号。

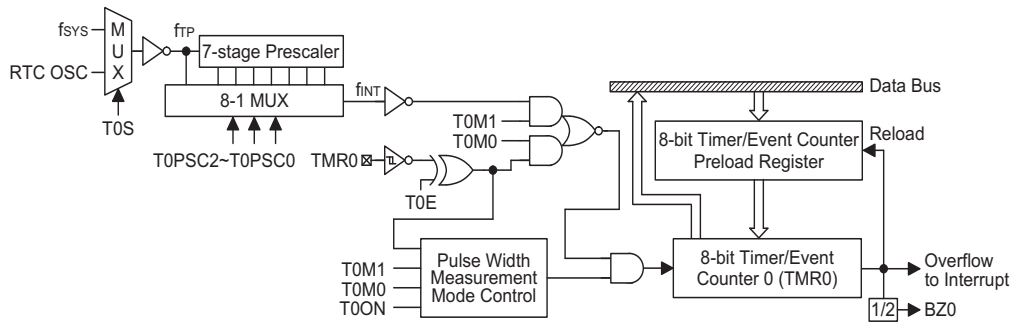
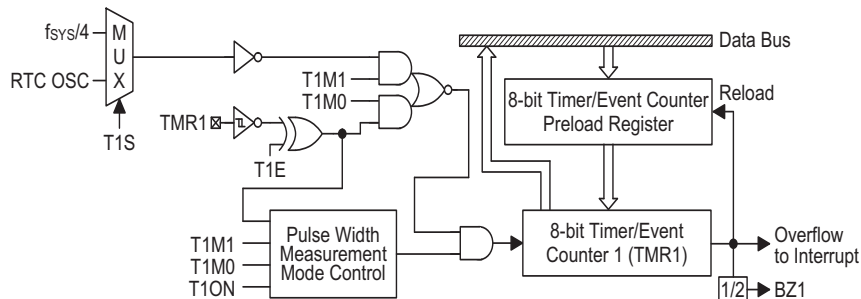
| 位 | 符号 | 功能 |
|--------|-------------------|---|
| 0~2 | T0PSC0~ T0PSC2 | 定义预分频器级数，T0PSC2,T0PSC1,T0PSC0= 000: $f_{INT} = f_{TP}$ 001: $f_{INT} = f_{TP}/2$ 010: $f_{INT} = f_{TP}/4$ 011: $f_{INT} = f_{TP}/8$ 100: $f_{INT} = f_{TP}/16$ 101: $f_{INT} = f_{TP}/32$ 110: $f_{INT} = f_{TP}/64$ 111: $f_{INT} = f_{TP}/128$ |
| 3 | TOE | 定义定时/计数器 TMR 触发方式 计数模式 (TOM1,TOM0) = (0, 1) : 1: 下降沿计数 0: 上升沿计数 测量脉宽模式 (TOM1,TOM0) = (1, 1) 1: 上升沿计数, 下降沿停止 0: 下降沿计数, 上升沿停止 |
| 4 | T0ON | 打开或关闭定时/计数器 (0=关闭; 1=打开) |
| 5 | T0S | 时钟来源选择 0: f_{sys} 1: RTC |
| 6 7 | TOM0 TOM1 | 定义工作模式 (TOM1, TOM0) 01=外部事件计数模式 (外部时钟来源) 10=定时器模式 (内部时钟来源) 11=脉宽测量模式 00=未定义 |

TMR0C(0EH)寄存器

| 位 | 符号 | 功能 |
|--------|--------------|---|
| 0~2 | — | 未定义，读出为“0” |
| 3 | T1E | 定义定时/计数器 TMR 触发方式 计数模式 (T1M1,T1M0) = (0, 1) : 1: 下降沿计数 0: 上升沿计数 测量脉宽模式 (T1M1,T1M0) = (1, 1) 1: 上升沿计数, 下降沿停止 0: 下降沿计数, 上升沿停止 |
| 4 | T1ON | 打开或关闭定时/计数器 (0=关闭; 1=打开) |
| 5 | T1S | 时钟来源选择 0: $f_{sys}/4$ 1: RTC |
| 6 7 | T1M0 T1M1 | 定义工作模式 (T0M1, T0M0) 01=外部事件计数模式 (外部时钟来源) 10=定时器模式 (内部时钟来源) 11=脉宽测量模式 00=未定义 |

TMR1C (11H) 寄存器

有两个寄存器与定时/计数器相关联，即 TMR0 [0DH]和 TMR0C [0EH] (TMR1 [10H],TMR1C [11H])。TMR 有两个物理空间。写入 TMR0 (TMR1) 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。TMR0C (TMR1C) 是定时/计数器控制寄存器。


定时/计数器 0

定时计数器 1 (HT48R02/HT48R03)

T0M0 和 T0M1 (T1M0, T1M1) 用来定义工作模式。外部事件计数模式用来记录外部事件, 它的时钟来自外部 TMR0 (TMR1) 引脚输入。定时器模式是一个常用模式, 它的时钟源来自 f_{INT} 时钟。脉冲宽度测量模式用来计算引脚 TMR0 (TMR1) 上的高/低电平的宽度。计数是基于 f_{INT} 时钟。

在外部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/计数器中的数值向上计数到 OFFH。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初值, 并且同时产生相应的中断请求状态位 (T0F ; INTC0 的第 5 位或 T1F; INTC0 的第 6 位)。

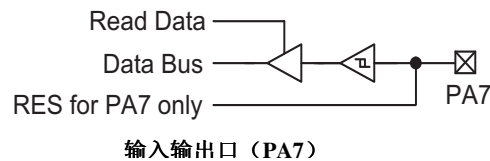
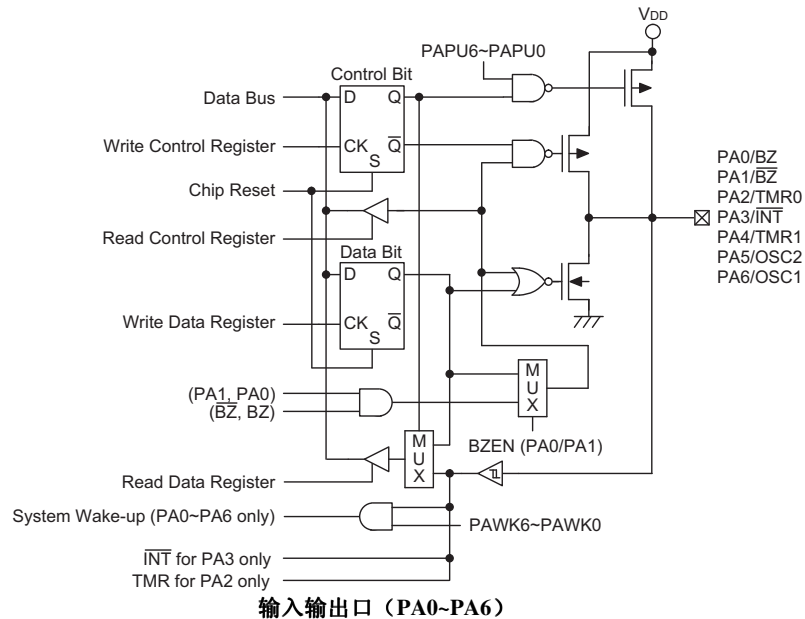
在脉冲宽度测量中, 将 T0ON 和 T0E (T1ON 和 T1E) 置为“1”, 如果 TMR0 (TMR1) 接收到上升沿 (如果 T0E (T1E) 位是零, 下降沿), 就开始计数, 直到 TMR0 (TMR1) 返回到原来的电平, 同时复位 T0ON (T1ON) 位。测量的结果被保留在定时/计数器中, 甚至电平跳变再一次发生也不会改变。换句话说, 一次只能测量一个脉冲宽度。当 T0ON (T1ON) 重新被置位, 只要再接到跳变信号, 那么测量过程会再次执行。要注意在这个操作模式中, 定时/计数器的启动计数不是根据逻辑电平, 而是依据信号的边沿跳变触发。一旦发生计数器溢出, 计数器会从定时/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 T0ON (T1ON) 置 1。在脉宽测量模式中, T0ON (T1ON) 在测量周期结束后自动被清零。但在另外两个模式中, T0ON (T1ON) 只能由指令来复位。定时/计数器的溢出是唤醒的信号之一。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/计数器为关闭的状态下, 写数据到定时/计数器的预置寄存器之中, 同时也会将数据装入定时/计数器中。但若是定时/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, 计数会被停止, 以防出错。停止计数会导致计数错误, 所以程序员必须仔细加以考虑。

TMR0C 的 0~2 位被用于定义定时/计数器的内部时钟源的预分频级数。定义如表所示。定时/计数器的溢出信号可用于产生驱动蜂鸣器的信号。

输入/输出口

单片机具有 7 个双向输入输出口和一个输入口，标号为 PA，其分别对应的 RAM 的[12H]。所有的输入/输出口都能被作为输入或输出使用。就输入而言这些口不具有锁存功能，即，输入数据必须在“MOV A, [m]” (m=12H) 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。



每个输入输出口都有其自己的控制寄存器 (PAC)，用来控制输入/输出模式 (PA7 只能作为输入口)。使用控制寄存器，可对 CMOS 输出或斯密特触发输入 (带或不带上拉电阻) 在软件下动态地进行改变 (PA7 口只能作为输入口使用)。作为输入时，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 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)。

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。

- 唤醒和上拉电阻功能:

PA (除 PA.7) 的每个端口都具有唤醒和上拉电阻功能分别由 PAWK, PAPU 寄存器控制。PA7 不带唤醒功能和上拉电阻。

| 位 | 符号 | 功能 |
|-----|-----------------|--|
| 0~6 | PAWK0~ PAWK6 | PAWK _n =0, PAn 口关闭唤醒功能 PAWK _n =1, PAn 口打开唤醒功能 |
| 7 | — | 未定义, 读数为“0” |

PAWK (15H) 寄存器

| 位 | 符号 | 功能 |
|-----|-----------------|---|
| 0~6 | PAPU0~ PAPU6 | PAPU _n =0, PAn 口不带上拉电阻 PAPU _n =1, PAn 口带上拉电阻 |
| 7 | — | 未定义, 读数为“0” |

PAPU (14H) 寄存器

- 蜂鸣器功能:

PA0/PA1 分别与 BZ/ \overline{BZ} 管脚复用。如蜂鸣器被选择, 那么 PA0 (PA1) 被设为输出脚, PA0 (PA1) 上的信号将成为蜂鸣器信号。如果设为输入口, 那么 PA0 (PA1) 保持原来的功能。

蜂鸣器的输出信号 (在输出模式) 只受 PA0 数据寄存器控制。PA0/BZ 和 PA1/ \overline{BZ} 输出功能如下表格所示。PA 口同样具有 COMS 输出或斯密特触发输入功能选择。PA0/BZ 和 PA1/ \overline{BZ} 输入输出功能如下。

| | | | | | | | | | | | | |
|------------|---|---|---|---|---|---|---|----------------|---|---|---|---|
| PA0 输入/输出 | I | I | I | I | O | O | O | O | O | O | O | O |
| PA1 输入/输出 | I | O | O | O | I | I | I | O | O | O | O | O |
| PA0 模式 | x | x | x | x | C | B | B | C | B | B | B | B |
| PA1 模式 | x | C | B | B | x | x | x | C | C | C | B | B |
| PA0 数据 | x | x | 0 | 1 | D | 0 | 1 | D ₀ | 0 | 1 | 0 | 1 |
| PA1 数据 | x | D | x | x | x | x | x | D ₁ | D | D | x | x |
| PA0 Pad 状态 | I | I | I | I | D | 0 | B | D ₀ | 0 | B | 0 | B |
| PA1 Pad 状态 | I | D | 0 | B | I | I | I | D ₁ | D | D | 0 | B |

注意: I: 输入; O: 输出; D, D₀, D₁: 数据

B: 蜂鸣器选项, BZ 或 \overline{BZ} ; x: 任意值

C: COMS 输出

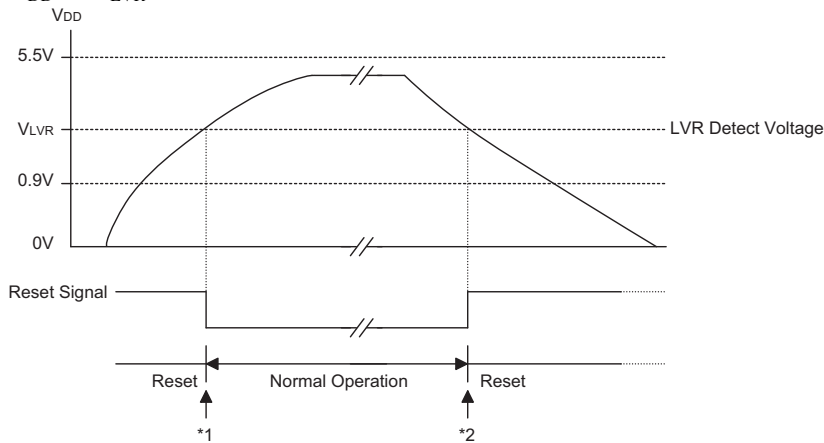
低电压复位 — LVR

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

LVR 功能说明如下：

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

V_{DD} 与 V_{LVR} 之间的关系如下所示：



低电压复位

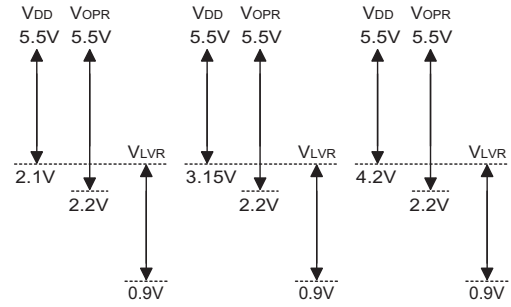
注：*1：要保证系统振荡器起振并稳定运行，在系统进入正常运行以前，SST 提供额外的 1024 个系统时钟周期的延迟。

*2：低电压状态必须保持 t_{LVR} 以上，进入复位模式就要有 t_{LVR} 的延迟。

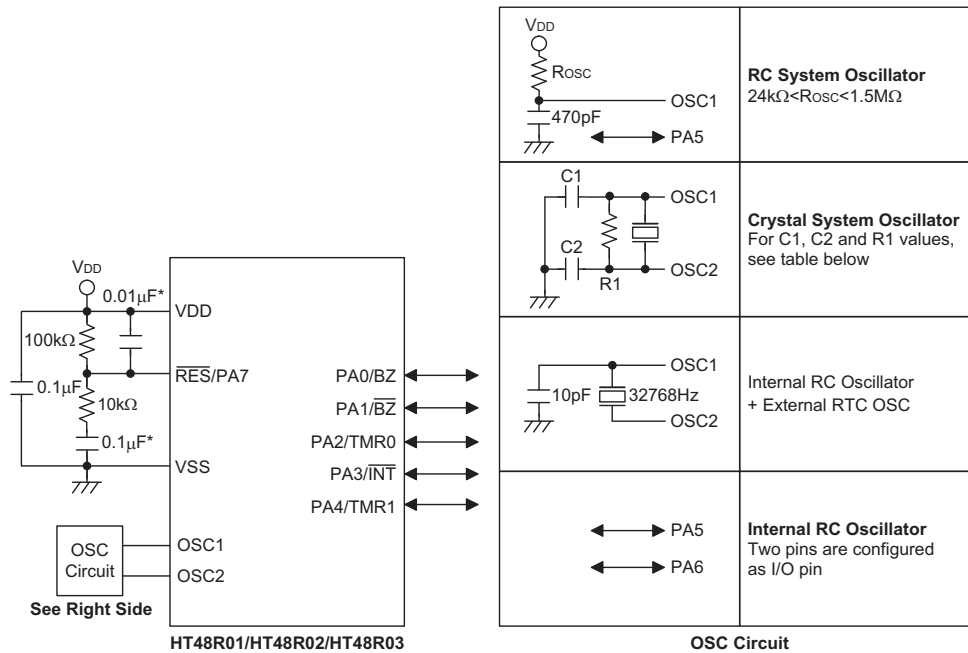
掩膜选项

以下的表格，列出了这种单片机的各种类型的掩膜选项。所有的掩膜选项必须正确定义。

| 编号 | 选项 |
|----|---|
| 1 | 系统振荡器选择 <ul style="list-style-type: none"> • 内部 RC+PA5/PA6 • 内部 RC+RTC • 外部晶振 • 外部 RC+PA5 |
| 2 | 内部 RC 频率选择：4MHz, 8MHz, 12MHz |
| 3 | WDT 功能：打开或关闭 |
| 4 | WDT 时钟来源：WDTOSC, $f_{SYS}/4$ 或者 RTC 晶振 |
| 5 | 清除看门狗指令条数：1 或 2 条清除 WDT |
| 6 | LVR 功能：打开或关闭 |
| 7 | LVR 选择：2.1V/3.15V/4.2V |
| 8 | \overline{RES} 或 PA7 选择 |



注： V_{OPR} 为系统时钟 4MHz 时一般芯片正常工作电压的范围。

应用电路


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 $\overline{\text{RES}}$ 引脚的线请尽可能的短

下表所示为根据不同的晶振值选择 R1、C1、C2（仅供参考）

| 晶体振荡或谐振器 | C1, C2 | R1 |
|-------------|--------|--------|
| 8MHz 晶振和谐振器 | 35pF | 3.9k Ω |
| 4MHz 晶振 | 10pF | 10k Ω |
| 4MHz 谐振器 | 10pF | 12k Ω |
| 3.58MHz 晶振 | 10pF | 12k Ω |
| 3.58MHz 谐振器 | 10pF | 12k Ω |
| 2MHz 晶振和谐振器 | 35pF | 12k Ω |
| 1MHz 晶振 | 68pF | 18k Ω |
| 480KHz 谐振器 | 300pF | 10k Ω |
| 455KHz 谐振器 | 300pF | 10k Ω |
| 429KHz 谐振器 | 300pF | 10k Ω |
| 400KHz 谐振器 | 300pF | 10k Ω |

电阻 R1 保证了在低电压状态下，晶振被关闭。这里的低电压，是指低于 MCU 正常工作电压范围。请注意，当启动了 LVR 功能，R1 可以不接。

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 and TBLH | 2 ⁽¹⁾ | 无 |
| TABRDL | [m] 读取最后页的 ROM 内容, 并送至数据存储器 and TBLH | 2 ⁽¹⁾ | 无 |
| 其它指令 | | | |
| NOP | 空指令 | 1 | 无 |
| CLR | [m] 清除数据存储器 | 1 ⁽¹⁾ | 无 |
| SET | [m] 置位数据存储器 | 1 ⁽¹⁾ | 无 |
| CLR | WDT 清除看门狗定时器 | 1 | TO,PDF |
| CLR | WDT1 预清除看门狗定时器 | 1 | TO ⁽⁴⁾ ,PDF ⁽⁴⁾ |
| CLR | WDT2 预清除看门狗定时器 | 1 | TO ⁽⁴⁾ ,PDF ⁽⁴⁾ |
| SWAP | [m] 交换数据存储器的高低字节, 结果放入数据存储器 | 1 ⁽¹⁾ | 无 |
| SWAPA | [m] 交换数据存储器的高低字节, 结果放入 ACC | 1 | 无 |
| HALT | 进入暂停模式 | 1 | TO,PDF |

注: x: 立即数

m: 数据存储器地址

A: 累加器

i: 第 0~7 位

addr: 程序存储器地址

√: 影响标志位

—: 不影响标志位

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

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

(3): (1)和(2)

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

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | √ | √ | √ | √ |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | √ | √ | √ | √ |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | √ | √ | √ | √ |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | √ | √ | √ | √ |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | √ | √ | √ | √ |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| 0 | 0 | — | — | — | — |

CLRWDT1 预清除看门狗定时器

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

运算过程： WDT \leftarrow 00H*

PDF&TO \leftarrow 0*

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| 0* | 0* | — | — | — | — |

CLRWDT2 预清除看门狗定时器

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

运算过程： WDT \leftarrow 00H*

PDF&TO \leftarrow 0*

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| 0* | 0* | — | — | — | — |

CPL [m] 对数据存储器取反，结果放入数据存储器

说明： 本指令是将数据存储器内保存的数值取反。

运算过程： [m] \leftarrow [m]

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

CPLA [m] 对数据存储器取反，结果放入累加器

说明： 本指令是将数据存储器内保存的值取反后，结果存放在累加器中。

运算过程： ACC \leftarrow [m]

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

DAA [m] 将加法运算后放入累加器的值调整为十进制数，并将结果放入数据存储器
 说明 本指令将累加器高低四位分别调整为 BCD 码。如果低四位的值大于“9”或 AC=1，那么 BCD 调整就执行对原值加“6”，并且内部进位标志 $AC1 = \overline{AC}$ ，即 AC 求反；否则原值保持不变。如果高四位的值大于“9”或 C=1，那么 BCD 调整就执行对原值加“6”再加 AC1，并把 C 置位；否则 BCD 调整就执行对原值加 AC1，C 的值保持不变。结果存放到数据存储器中，只有进位标志位(C)受影响。

操作 如果 $ACC.3 \sim ACC.0 > 9$ 或 $AC=1$
 那么 $[m].3 \sim [m].0 \leftarrow (ACC.3 \sim ACC.0) + 6$ ， $AC1 = \overline{AC}$
 否则 $[m].3 \sim [m].0 \leftarrow (ACC.3 \sim ACC.0)$ ， $AC1 = 0$
 并且
 如果 $ACC.7 \sim ACC.4 + AC1 > 9$ 或 $C=1$
 那么 $[m].7 \sim [m].4 \leftarrow (ACC.7 \sim ACC.4) + 6 + AC1$ ， $C=1$
 否则 $[m].7 \sim [m].4 \leftarrow (ACC.7 \sim ACC.4) + AC1$ ， $C=C$

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | √ |

DEC [m] 数据存储器内容减 1，结果放入数据存储器
 说明： 本指令将数据存储器内的数值减一再放回数据存储器。
 运算过程： $[m] \leftarrow [m] - 1$
 影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

DECA [m] 数据存储器内容减 1，结果放入累加器
 说明： 本指令将存储器内的数值减一，再放到累加器。
 运算过程： $ACC \leftarrow [m] - 1$
 影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

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

运算过程： $PC \leftarrow PC + 1$
 $PDF \leftarrow 1$
 $TO \leftarrow 0$

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| 0 | 1 | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

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

| 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： $PC \leftarrow addr$
 影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

NOP 空指令

说明: 本指令不作任何运算, 而只将程序计数器加一。

 运算过程: $PC \leftarrow PC+1$

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: $ACC \leftarrow ACC \text{ “OR” } [m]$

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

OR A, x 累加器与立即数做“或”运算, 结果放入累加器

说明: 本指令是把累加器值、立即数做逻辑或, 结果放到累加器。

 运算过程: $ACC \leftarrow ACC \text{ “OR” } x$

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

ORM A, [m] 累加器与数据存储器做“或”运算, 结果放入数据存储器

说明: 本指令是把累加器值、存储器值做逻辑或, 结果放到数据存储器。

 运算过程: $[m] \leftarrow ACC \text{ “OR” } [m]$

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

RET 从子程序返回

说明: 本指令是将堆栈寄存器中的程序计数器值送回程序计数器。

 运算过程: $PC \leftarrow Stack$

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: $PC \leftarrow Stack$
 $ACC \leftarrow x$

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

运算过程： $PC \leftarrow Stack$
 $EMI \leftarrow 1$

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

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

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

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

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

RLC[m] 带进位将数据存储器左移一位，结果放入数据存储器
 说明：本指令是将存储器内的数值与进位标志左移一位，第 7 位取代进位标志，进位标志移到第 0 位，结果送回数据存储器。

运算过程： $[m].(i+1) \leftarrow [m].i; (i=0\sim6)$

$[m].0 \leftarrow C$
 $C \leftarrow [m].7$

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

$ACC.0 \leftarrow C$
 $C \leftarrow [m].7$

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | √ |

RR [m] 数据存储器右移一位，结果放入数据存储器
 说明：本指令是将存储器内的数值循环右移，第 0 位移到第 7 位，结果送回数据存储器。
 运算过程： $[m].7 \leftarrow [m].0, [m].i \leftarrow [m].(i+1); \quad (i=0\sim6)$
 影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

RRA [m] 数据存储器右移一位，结果放入累加器
 说明：本指令是将数据存储器内的数值循环右移，第 0 位移到第 7 位，结果送回累加器，而数据存储器内的数值不变。
 运算过程： $ACC.7 \leftarrow [m].0, ACC.i \leftarrow [m].(i+1); \quad (i=0\sim6)$
 影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

RRC [m] 带进位将数据存储器右移一位，结果放入数据存储器
 说明：本指令是将存储器内的数值加进位标志循环右移，第 0 位取代进位标志，进位标志移到第 7 位，结果送回存储器。
 运算过程： $[m].i \leftarrow [m].(i+1); \quad (i=0\sim6)$
 $[m].7 \leftarrow C$
 $C \leftarrow [m].0$
 影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | √ |

RRCA [m] 带进位将数据存储器右移一位，结果放入累加器
 说明：本指令是将数据存储器内的数值加进位标志循环右移，第 0 位取代进位标志，进位标志移到第 7 位，结果送回累加器，数据存储器内的数值不变。
 运算过程： $ACC.i \leftarrow [m].(i+1); \quad (i=0\sim6)$
 $ACC.7 \leftarrow C$
 $C \leftarrow [m].0$
 影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | √ |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | √ | √ | √ | √ |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | √ | √ | √ | √ |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

运算过程: 如果 $([m]+1=0)$, 跳过下一行指令; $[m] \leftarrow [m]+1$

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

运算过程: 如果 $[m]+1=0$, 跳过下一行指令; $ACC \leftarrow ([m]+1)$

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

运算过程: 如果 $[m].i \neq 0$, 跳过下一行指令。

影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

SUB A, [m] 累加器与数据存储器相减, 结果放入累加器
 说明: 本指令是把累加器值、数据存储器值相减, 结果放到累加器。

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

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

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： $[m] \leftarrow ACC + [\overline{m}] + 1$
 影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | √ | √ | √ | √ |

SWAP [m] 交换数据存储器的高低字节，结果放入数据存储器
 说明： 本指令是将数据存储器的低四位和高四位互换，再将结果送回数据存储器。
 运算过程： $[m].7 \sim [m].4 \leftrightarrow [m].3 \sim [m].0$
 影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

SWAPA [m] 交换数据存储器的高低字节，结果放入累加器
 说明： 本指令是将数据存储器的低四位和高四位互换，再将结果送回累加器。
 运算过程： $ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$
 $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
 影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

SZA [m] 数据存储器送至累加器，如果内容为“0”，则跳过下一条指令
 说明： 本指令是判断存储器内的数值是否为0，若为0则跳过下一行指令，即放弃在目前指令执行期间所取得的下一条指令，并插入一个空周期用以得正确的指令(二个指令周期)。并把存储器内值送到累加器，而存储器的值保持不变。否则执行下一条指令(一个指令周期)。
 运算过程： 如果 $[m] = 0$ ，跳过下一行指令，并 $ACC \leftarrow [m]$ 。
 影响标志位

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | — | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

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

| TO | PDF | OV | Z | AC | C |
|----|-----|----|---|----|---|
| — | — | — | √ | — | — |

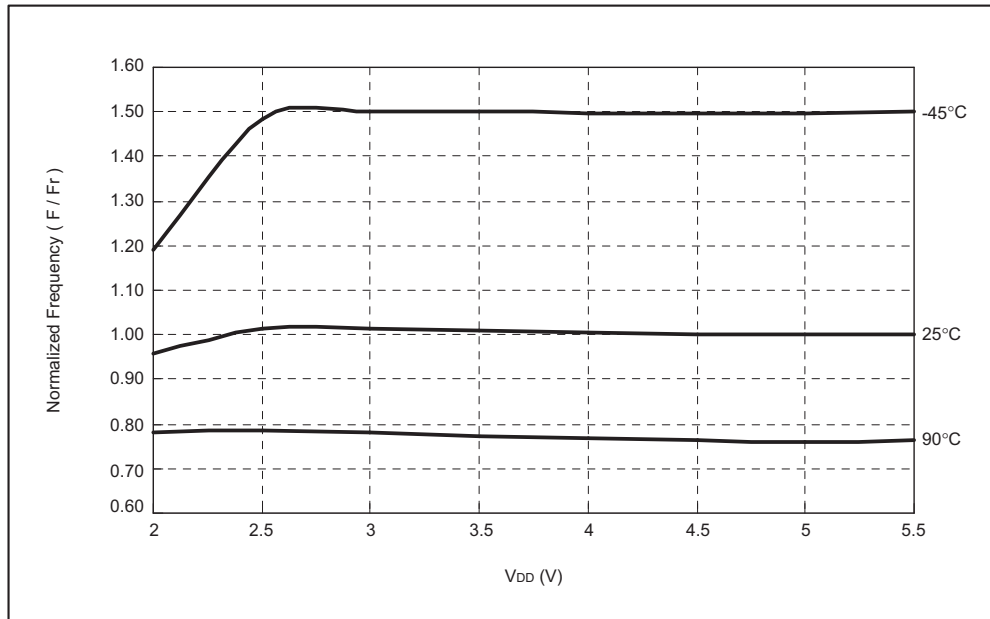
设备特性图表

以下特性图表描绘了设备典型的动作。这里的数据是收集来自不同单元数据的统计总结。此特性仅供参考，并没有在制造过程中测试过。

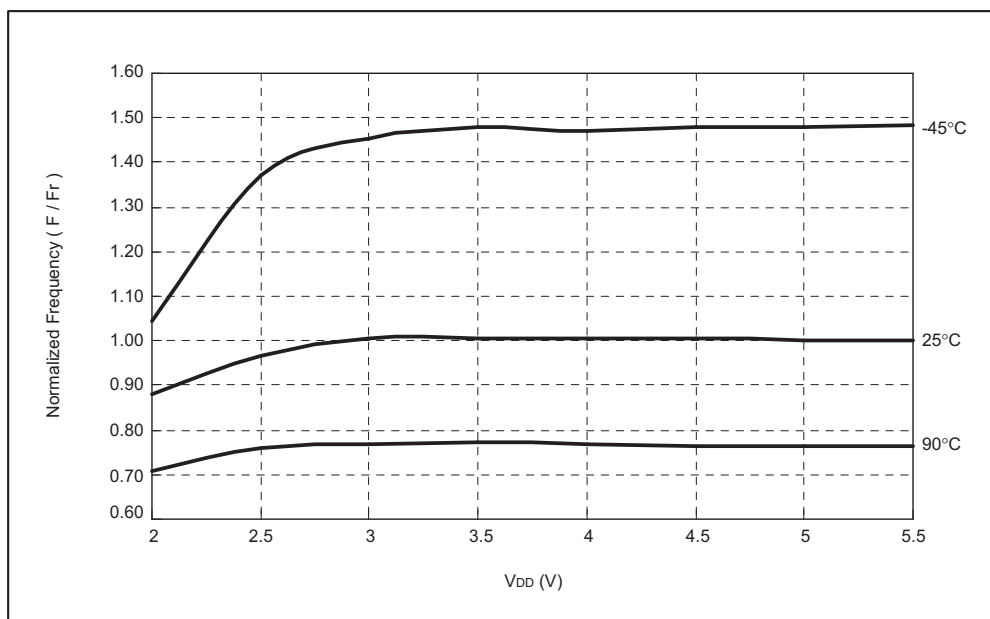
在一些图表中显示的，超出指定的运行范围的数据仅作为参考。设备只有在指定的范围内才能正常的运行。

标准频率是实际频率(F)和参考频率 (Fr)的比值(在 VDD=5V, Ta=25°C时)。

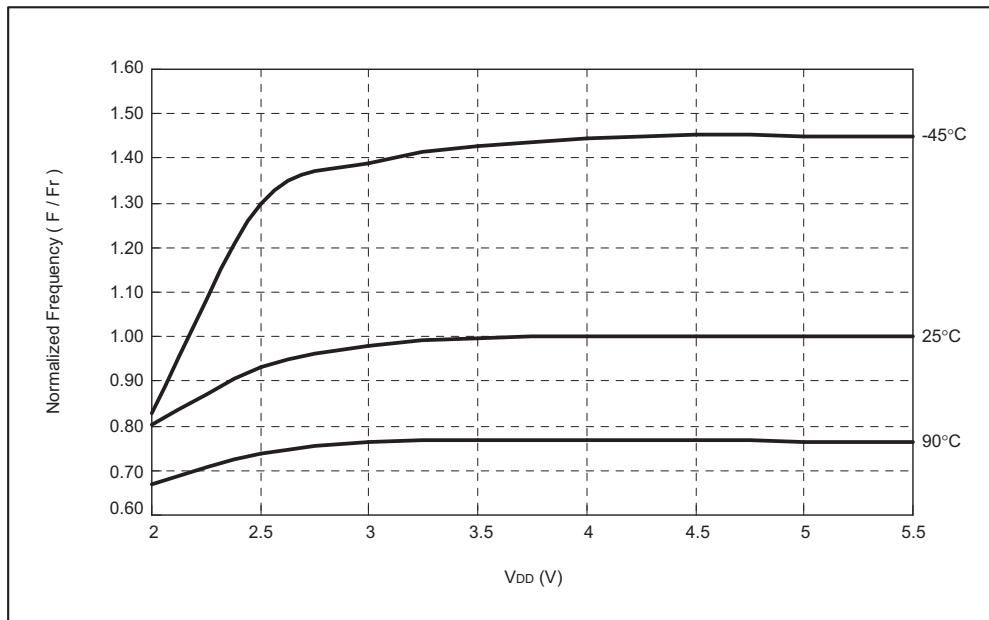
4MHZ IRC 频率 vs.VDD



8MHZ IRC 频率 vs.VDD

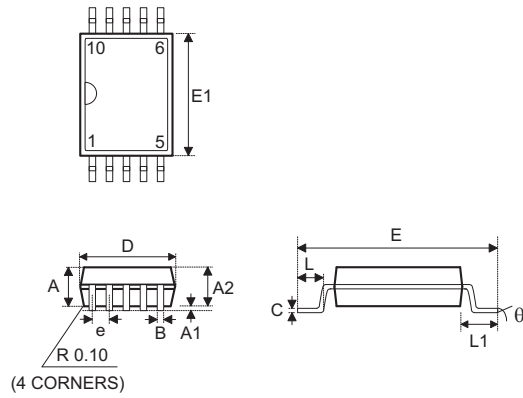


12MHZ IRC 频率 vs.VDD



封装信息:

10-pin MSOP 外形尺寸



| 标号 | 尺寸 (mm) | | |
|----------|---------|------|------|
| | Min | Nom | Max |
| A | — | — | 1.1 |
| A1 | 0 | — | 0.15 |
| A2 | 0.75 | — | 0.95 |
| B | 0.17 | — | 0.27 |
| C | — | — | 0.25 |
| D | — | 3 | — |
| E | — | 4.9 | — |
| E1 | — | 3 | — |
| e | — | 0.5 | — |
| L | 0.4 | — | 0.8 |
| L1 | — | 0.95 | — |
| θ | 0° | — | 8° |

盛群半导体股份有限公司（总公司）

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

盛群半导体股份有限公司（台北业务处）

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

盛扬半导体有限公司（上海业务处）

上海市宜山路2016号合川大厦1号楼3楼G室 200103
电话: 86-21-5422-4590
传真: 86-21-5422-4596
网站: www.holtek.com.cn

盛扬半导体有限公司（深圳业务处）

深圳市南山区科技园科技中三路与高新中二道交汇处生产力大楼A单元五楼 518057
电话: 86-755-8616-9908, 86-755-8616-9308
传真: 86-755-8616-9722

盛扬半导体有限公司（北京业务处）

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

盛扬半导体有限公司（成都业务处）

成都市东大街97号香槟广场C座709室 610016
电话: 028-6653-6590
传真: 028-6653-6591

Holmate Semiconductor, Inc.（北美业务处）

46712 Fremont Blvd., Fremont, CA 94538
电话: 510-252-9880
传真: 510-252-9885
网站: www.holtek.com

Copyright © 2008 by HOLTEK SEMICONDUCTOR INC.

使用指南中所出现的信息在出版当时相信是正确的，然而盛群对于说明书的使用不负任何责任。文中提到的应用目的仅仅是用来做说明，盛群不保证或表示这些没有进一步修改的应用将是适当的，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。盛群产品不授权用于救生、维生器件或系统中做为关键器件。盛群拥有不事先通知而修改产品的权利，对于最新的信息，请参考我们的网址 <http://www.holtek.com.tw>