

**HT49R30A-1, HT49R50A-1,  
HT49R70A-1, HT49RU80**

**LCD 型单片机  
使用手册**

第二版

二〇〇六年二月

Copyright © 2006 by HOLTEK SEMICONDUCTOR INC.本使用手册版权为盛群半导体股份有限公司所有，非经盛群半导体股份有限公司书面授权同意，不得透过任何形式复制、储存或传输。



# 目录

第一部份 单片机概论 .....	1
第一章 硬件结构 .....	3
简介 .....	3
特性 .....	4
技术特性 .....	4
内核特性 .....	4
周边特性 .....	5
选择表 .....	6
系统框线图 .....	7
引脚分配 .....	8
引脚说明 .....	10
极限参数 .....	15
直流电气特性 .....	16
交流电气特性 .....	20
系统结构 .....	22
时序和流水线结构(Pipelining) .....	22
程序计数器 .....	23
堆栈 .....	26
算术及逻辑单位 - ALU .....	27
程序存储器 .....	27
结构 .....	27
特殊向量 .....	29
多 bank 管理 .....	30
查表 .....	33
查表程序范例 .....	34

数据存储器.....	37
结构.....	37
通用数据存储器.....	38
专用数据存储器.....	39
LCD 存储器.....	40
特殊功能寄存器.....	42
间接寻址寄存器 – IAR0, IAR1.....	42
间接寻址指针 – MP0, MP1.....	42
存储区指针 – BP.....	43
累加器 – ACC.....	44
程序计数器低字节寄存器 – PCL.....	44
查表寄存器 – TBLP, TBHP, TBLH.....	45
实时时钟控制寄存器 – RTCC.....	45
状态寄存器 – STATUS.....	46
中断控制寄存器 – INTC0, INTC1, MFIC.....	47
定时/计数寄存器.....	47
输入/输出寄存器.....	48
UART 寄存器 – USR, UCR1, UCR2, TXR/RXR, BRG.....	48
输入/输出.....	48
上拉电阻.....	49
PA 口的唤醒.....	49
引脚共用功能.....	50
编程注意事项.....	53
液晶显示(LCD)驱动器.....	54
LCD 存储器.....	54
LCD 时钟.....	56
LCD 驱动器输出.....	56
LCD 电压源与偏压.....	62
编程注意事项.....	65
定时/计数器.....	66
配置定时/计数器输入时钟源.....	66
定时/计数寄存器 – TMR, TMR0, TMR1, TMR1L/TMR1H, TMR2L/TMR2H.....	69
定时/计数控制寄存器 – TMRC, TMR0C, TMR1C, TMR2C.....	70
定时器模式.....	73
事件计数模式.....	73
脉冲宽度测量模式.....	74
可编程分频器 – PFD.....	75
输入/输出接口.....	76
编程注意事项.....	76
中断.....	77
中断寄存器.....	77
中断优先权.....	83
外部中断.....	84
定时/计数器中断.....	84

时基中断 .....	85
实时时钟中断 .....	87
UART 中断 .....	88
多功能中断 .....	88
编程注意事项 .....	89
复位和初始化 .....	89
复位 .....	90
异步串行口——UART .....	95
UART 特性 .....	95
UART 外部引脚 .....	96
数据发送 .....	96
UART 状态和控制寄存器 .....	96
波特率发送器 .....	103
UART 设置和控制 .....	105
UART 发送器 .....	106
UART 接收器 .....	108
接收错误处理 .....	109
接收中断图解 .....	111
地址检测模式 .....	111
暂停模式下的 UART 功能 .....	112
UART 应用范例 .....	112
振荡器 .....	113
系统时钟配置 .....	113
系统晶体/陶瓷振荡器 .....	114
系统电阻电容振荡器 .....	114
RTC 振荡器 .....	115
看门狗定时振荡器 .....	116
内部时钟源 .....	116
暂停和唤醒 .....	117
暂停 .....	117
进入暂停 .....	117
静态电流 .....	118
唤醒 .....	118
低电压检测器 – LVD .....	119
看门狗定时器 .....	119
蜂鸣器 .....	121
掩膜选项 .....	122
应用电路 .....	124

<b>第二部份 程序语言 .....</b>	<b>129</b>
<b>第二章 指令集介绍.....</b>	<b>131</b>
指令集.....	131
指令周期.....	131
数据的传送.....	132
算术运算.....	132
逻辑和移位运算.....	132
分支和控制的转换.....	132
位运算.....	132
查表运算.....	133
其它运算.....	133
指令设定一览表.....	134
惯例.....	134
<b>第三章 指令定义.....</b>	<b>137</b>
<b>第四章 汇编语言和编译器.....</b>	<b>151</b>
常用符号.....	151
语句语法.....	152
名称.....	152
操作项.....	152
操作数项.....	152
注解.....	152
编译伪指令.....	153
条件编译伪指令.....	153
文件控制伪指令.....	154
程序伪指令.....	156
数据定义伪指令.....	160
宏指令.....	162
汇编语言指令.....	166
名称.....	166
助记符.....	166
操作数、运算符和表达式.....	166
其它.....	169
前置引用.....	169
局部标号.....	169
汇编语言保留字.....	170
编译器选项.....	171
编译列表文件格式.....	171
源程序列表.....	171
编译总结.....	172

其它 .....	172
<b>第三部份 开发工具 .....</b>	<b>175</b>
<b>第五章 单片机开发工具.....</b>	<b>177</b>
HT-IDE 集成开发环境.....	177
盛群单片机仿真器(HT-ICE) .....	179
HT-ICE 接口卡 .....	179
OTP 烧录器.....	179
OTP 转接座.....	179
系统配置.....	180
HT-ICE 接口卡设定 .....	181
安装.....	183
系统需求 .....	183
硬件安装 .....	183
软件安装 .....	184
<b>第六章 快速开始.....</b>	<b>189</b>
步骤一：建立一个新项目 .....	189
步骤二：将源程序文件加到项目中 .....	189
步骤三：建立项目 .....	189
步骤四：烧录 OTP 单片机.....	190
步骤五：传送程序与掩膜选项单至 Holtek .....	190
<b>第七章 LCD 仿真器 .....</b>	<b>191</b>
简介.....	191
LCD 面板配置文件.....	191
面板文件的文件名和项目的关系 .....	192
选择 HT-LCDS.....	192
LCD 面板图形文件.....	194
建立 LCD 面板配置文件.....	195
建立面板的配置结构 .....	195
选择图形并设定位置 .....	196
加入新的图形 .....	196
删除图形 .....	197
改变图形 .....	197
改变图形位置 .....	197
如何加入用户定义的矩阵 .....	198
使用 Panel Editor 定义图形 .....	199
使用批处理文件将图形加入面板 .....	200
选择 LCD 面板的颜色 .....	200
LCD 仿真.....	201
停止仿真.....	201

附录 .....	202
附录 A 特性曲线图 .....	203
附录 B 封装信息 .....	218



# 前言

自从盛群半导体公司成立以来，即致力于单片机产品的设计与开发。虽然盛群半导体提供给客户各式各样的半导体器件，但其中单片机仍是盛群的主要关键产品，未来盛群半导体仍将继续扩展单片机产品系列完整性与功能性。通过长期累积的单片机研发经验与技术，盛群半导体能为各式各样的应用范围开发出高性能且低价位的单片机芯片。有些单片机内部集成了全双工的 UART 的功能，使得它们很方便的与外部串行通信。盛群的 LCD 型单片机为客户提供了绝佳的产品方案，大大地为顾客增强了他们以 LCD 为主的产品功能，当设计者使用盛群所开发出的各式开发工具时，更可减少产品开发周期并大大的增加他们的产品附加价值。

为了使用者阅读方便，本手册分成三部份。关于一般的单片机的规格信息可在第一部份中找到。与单片机程序相关的信息，如指令集、指令定义和汇编语言编译伪指令，可在第二部份找到。第三部份则是关于盛群半导体的开发工具有关如何安装和使用的相关信息。

希望客户通过这本 LCD 型单片机使用手册，能以一种简单、有效、且完整的方法，实现他们在单片机上的各种应用。由于盛群半导体将单片机规格、程序规划和开发工具等信息结合在一本使用手册上，相信客户可充分利用盛群半导体各种单片机的特性，获得最大的产品优势。建议客户随时访问盛群网站获得最新手册，同时也欢迎提供宝贵的意见和建议，以作为我们未来的改进参考。



## 第一部份

# 单片机概论



## 第一章

# 硬件结构

# 1

本章主要为 LCD 型单片机的规格信息，并且包含了所有参数和相关的硬件信息，这些信息提供设计者此类单片机的主要硬件特性细节，结合程序部份的信息将能够让使用者快速且成功地实现各种单片机的应用。参考本章中的相关部份，也保证使用者可以充分利用 LCD 型单片机。

## 简介

HT49R30A-1/HT49C30-1/HT49C30L、HT49R50A-1/HT49C50-1/HT49C50L 和 HT49R70A-1/HT49C70-1/HT49C70L,HT49RU80/HT49CU80 是 8 位高性能、高效益的 RISC 结构单片机，适用于 LCD 控制产品的应用。内部的特殊特性，如暂停、唤醒功能、振荡器选择、蜂鸣器驱动和 UART 等，增加了单片机的灵活度，而这些特性也同时保证实际应用时只需要最少的外部组件，进而降低了整个产品成本。有了低功耗、高性能、灵活控制的输入/输出和低成本等优势，这些芯片具有许多功能，并适合被广泛应用在以 LCD 为主的场合，如天平、电子万用表、气体计量器、定时器、计算器、遥控器以及许多其它以 LCD 为主的工业用品和家用电器上。所有的单片机都具有相同的特性，主要的不同在于 LCD 段数、I/O 引脚数目、RAM 和 ROM 的容量，定时器数目和封装种类等方面。另外 HT49RU80/HT49CU80 还集成了一个全双工的串行口。

HT49R30A-1、HT49R50A-1、HT49R70A-1 和 HT49RU80 都是属于一次可编程（One-Time Programmable, OTP）单片机，当配合使用盛群半导体的程序开发工具时，可简单有效的更新程序。这提供了设计者快速有效的开发途径。而对于那些已经设计成熟的应用，掩膜版的 HT49C30-1、HT49C50-1、HT49C70-1、HT49CU80，HT49C30L、HT49C50L 和 HT49C70L 则可满足大量生产和低成本的需求。由于和 OTP 版的功能完全兼容，掩膜版对于已经设

计完成而想要降低成本的产品，提供了一个理想的解决方案。

HT49C30L, HT49C50L 和 HT49C70L 是 LCD 单片机系列的低电压版，它们有在只有提供 1.2V 最小低电压功率下工作的能力，这些设备非常适合单一电池的应用。虽然它们只有掩膜版可用，但和 OTP 相关的设备 HT49R30A-1, HT49R50-1 和 HT49R70-1 在产品开发期间是完全适合且可用的。

## 特性

### 技术特性

- 高性能 RISC 结构
- 低功率完全静态 CMOS 设计
- 工作电压：  
对于 HT49R30A-1/HT49C30-1、HT49R50A-1/HT49C50-1、  
HT49R70A-1/ HT 49C70-1, HT49RU80/HT49CU80  
在 4MHz 下，由 2.2V 到 5.5V  
在 8MHz 下，由 3.3V 到 5.5V  
对于 HT49C30L、HT49C50L、HT49C70L  
在 500kHz 下，由 1.2V 到 2.2V
- 功率损耗：  
在 5V/8MHz 下，典型值为 4mA
- 周期时间：  
在 8MHz 系统时钟下指令周期达到 0.5 $\mu$ s (HT49R30A-1/HT49C30-1、  
HT49R50A-1/HT49C50-1、HT49R70A-1/ HT 49C70-1，  
HT49RU80/HT49CU80)  
在 500KHz 系统时钟下指令周期达到 8 $\mu$ s (HT49C30L、HT49C50L、  
HT 49C70L)
- 温度范围  
工作温度-40°C 到 85°C (工业级规格)  
储存温度-50°C 到 125°C

### 内核特性

- 程序存储器  
2K $\times$ 14 OTP/Mask ROM (HT49R30A-1/HT49C30-1/HT49C30L)  
4K $\times$ 15 OTP/Mask ROM (HT49R50A-1/HT49C50-1/HT49C50L)  
8K $\times$ 16 OTP/Mask ROM (HT49R70A-1/HT49C70-1/HT49C70L)  
16K $\times$ 16 OTP/Mask ROM (HT49RU80/HT49CU80)

- 数据存储器
  - 96×8 RAM (HT49R30A-1/HT49C30-1/HT49C30L)
  - 160×8 RAM (HT49R50A-1/HT49C50-1/HT49C50L)
  - 224×8 RAM (HT49R70A-1/HT49C70-1/HT49C70L)
  - 576×8 RAM (HT49RU80/HT49CU80)
- LCD 驱动
  - 18×4, 19×3 或 19×2 (HT49R30A-1/HT49C30-1/HT49C30L)
  - 32×4, 33×3 或 33×2 (HT49R50A-1/HT49C50-1/HT49C50L)
  - 40×4, 41×3 或 41×2 (HT49R70A-1/HT49C70-1/HT49C70L)
  - 47×4, 48×3 或 48×2 (HT49RU80/HT49CU80)
- 表格读取功能
- 多层硬件堆栈
  - 4-level (HT49R30A-1/HT49C30-1/HT49C30L)
  - 6-level (HT49R50A-1/HT49C50-1/HT49C50L)
  - 16-level (HT49R70A-1/HT49C70-1/HT49C70L, HT49RU80/HT49CU80)
- 位操作指令
- 63 条强大的指令
- 大多数指令执行时间只需要一个指令周期

#### 周边特性

- 从 6 个到 8 个具有上拉电阻的输入引脚
- 从 8 个到 16 个具有上拉功能的双向输入输出口
- 7 个输出引脚(HT49RU80/HT49CU80)
- 内部 LCD 驱动
- 内部专用的 LCD 存储器
- PA 端口具有唤醒功能
- 一个、二个或三个具内部中断的事件计数输入
- UART( 只有 HT49RU80/HT49CU80 具有)
- 二个外部中断输入
- 具有预分频器(Prescaler)及中断功能的定时器
- 看门狗定时器(WDT)
- 暂停与唤醒特性可以节省功耗
- PFD 输出
- 蜂鸣器驱动输出
- 芯片内置晶体、电阻电容振荡器和 32768Hz 晶体振荡器
- 32768Hz 实时时钟(RTC)功能
- 具有低电压复位/检测(LVR/LVD)特性 (HT49C30L/HT49C50L/HT49C70L 除外)

- 具有烧录电路接口及程序代码保护功能
- 掩膜版单片机适用于大量生产
- 低电压版单片机适用于单个电池工作(只 HT49C30L/HT49C50L/HT49C70L 具有)
- 提供高效的软硬支持工具

## 选择表

这系列的 LCD 型单片机具有广泛的功能特性，其中有些是普通的，有些则是独有的。大部份的特性对该系列所有的单片机来说是共通的，主要的区别在于程序存储器、数据存储器的容量、LCD 段数和定时器功能。为了帮助使用者在应用时能选择适当的单片机，以下的表格提供了各个单片机主要的特性概述。

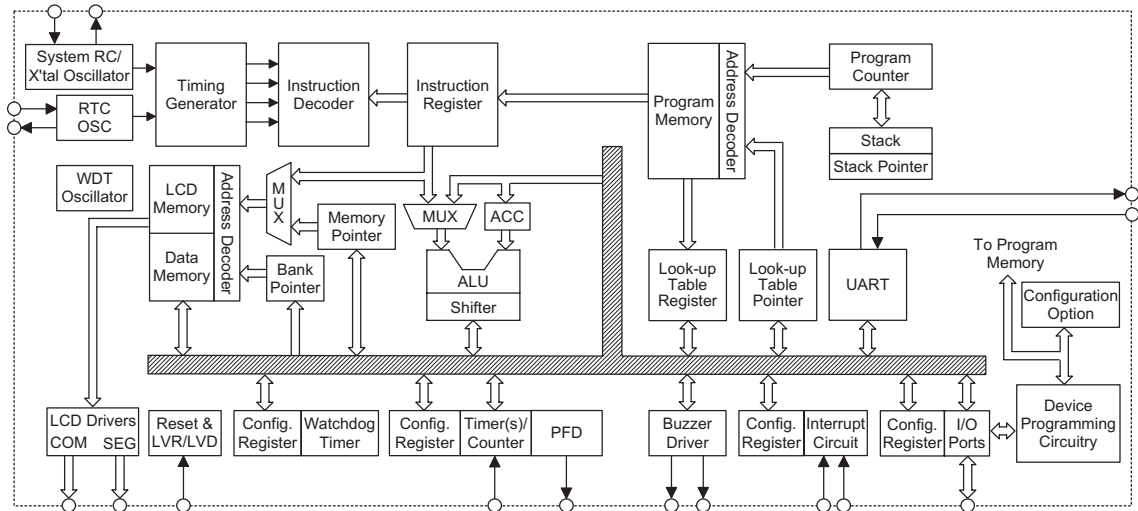
型号	VDD	程序存储器	数据存储器	I/O	LCD	定时器	中断	UART	堆栈	封装种类
HT49R30A-1	2.2V~5.5V	2K×14	96×8	14	19×2	8-bit×1	5	-	4	48SSOP
HT49C30-1					19×3					
HT49C30L	1.2V~2.2V				18×4					
HT49R50A-1	2.2V~5.5V	4K×15	160×8	20	33×2	8-bit×2	6	-	6	48SSOP 100QFP
HT49C50-1					33×3					
HT49C50L	1.2V~2.2V				32×4					
HT49R70A-1	2.2V~5.5V	8K×16	224×8	24	41×2	8-bit×1 16-bit×1	6	-	16	100QFP
HT49C70-1					41×3					
HT49C70L	1.2V~2.2V				40×4					
HT49RU80	2.2V~5.5V	16K×16	576×8	31	48×2	8-bit×1 16-bit×2	8	√	16	100QFP
HT49CU80					48×3					
					47×4					

- 注意：**
1. 型号部份包含“C”的为Mask版本，而“R”则是OTP版本。
  2. 型号部份包含“L”的为低电压Mask版本。
  3. 对于 HT49R50A-1/HT49C50-1/HT49C50L，表格中说明的是 100-pin QFP 封装的数值。对于 48-pin SSOP 的封装，它具有 8 个 I/O 和 6 个输入引脚，19×3 或 18×4 的 LCD 驱动引脚。



## 系统框线图

以下的系统框线图为 LCD 型单片机系列的主要功能模块。



- 注意：**
- 1、本系统框线图为 OTP 单片机，至于 Mask 型单片机则没有 Device Programming Circuitry。
  - 2、只有在 HT49RU80/HT49CU80 中才具有 UART 功能。

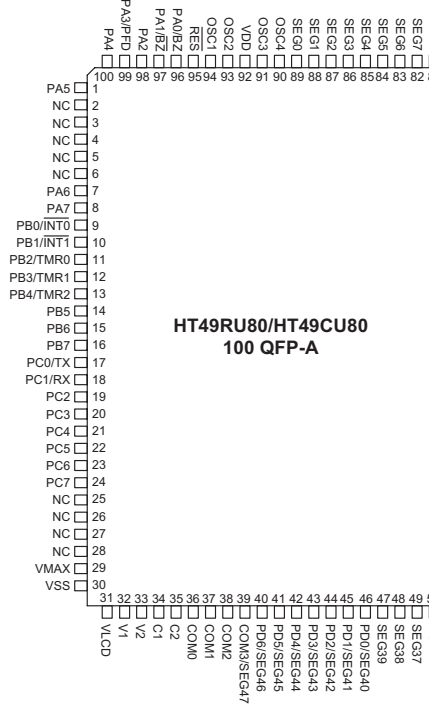
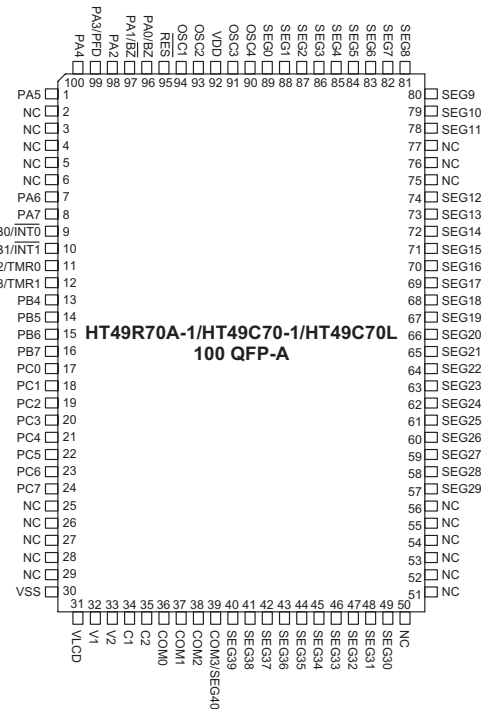
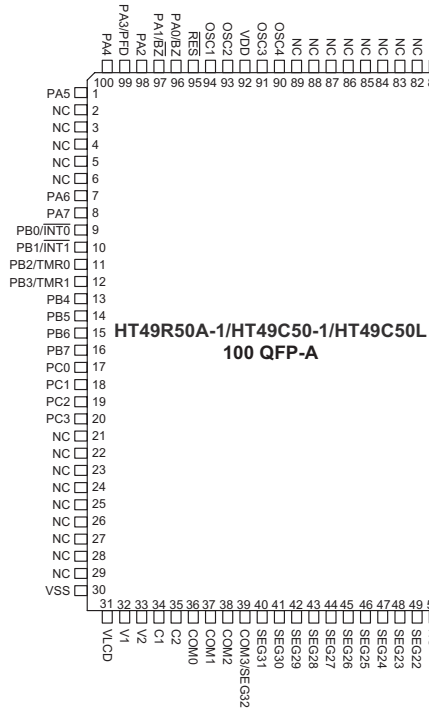
## 引脚分配

PA0/BZ	1	48	RES
PA1/BZ	2	47	OSC1
PA2	3	46	OSC2
PA3/PFD	4	45	VDD
PA4	5	44	OSC3
PA5	6	43	OSC4
PA6	7	42	SEG0
PA7	8	41	SEG1
PB0/INT0	9	40	SEG2
PB1/INT1	10	39	SEG3
PB2/TMR	11	38	SEG4
PB3	12	37	SEG5
PB4	13	36	SEG6
PB5	14	35	SEG7
VSS	15	34	SEG8
VLCD	16	33	SEG9
V1	17	32	SEG10
V2	18	31	SEG11
C1	19	30	SEG12
C2	20	29	SEG13
COM0	21	28	SEG14
COM1	22	27	SEG15
COM2	23	26	SEG16
COM3/SEG18	24	25	SEG17

HT49R30A-1/HT49C30-1/HT49C30L  
48 SSOP-A

PA0/BZ	1	48	RES
PA1/BZ	2	47	OSC1
PA2	3	46	OSC2
PA3/PFD	4	45	VDD
PA4	5	44	OSC3
PA5	6	43	OSC4
PA6	7	42	SEG10
PA7	8	41	SEG11
PB0/INT0	9	40	SEG12
PB1/INT1	10	39	SEG13
PB2/TMR0	11	38	SEG14
PB3/TMR1	12	37	SEG15
PB4	13	36	SEG16
PB5	14	35	SEG17
VSS	15	34	SEG18
VLCD	16	33	SEG19
V1	17	32	SEG20
V2	18	31	SEG21
C1	19	30	SEG22
C2	20	29	SEG23
COM0	21	28	SEG24
COM1	22	27	SEG25
COM2	23	26	SEG26
COM3/SEG32	24	25	SEG27

HT49R50A-1/HT49C50-1/HT49C50L  
48 SSOP-A



**注意:** 单片机的封装引脚具有兼容特性,使其在硬件应用时以最小的改变去提供设备直接升级到更高的功能。

## 引脚说明

### HT49R30A-1/HT49C30-1/HT49C30L

引脚名称	I/O	掩膜选项	说明
PA0/BZ PA1/ $\overline{\text{BZ}}$ PA2 PA3/PFD PA4~PA7	I/O	Wake-up CMOS or NMOS Pull-high PA0/PA1 or BZ/ $\overline{\text{BZ}}$ PA3 or PFD	8 位双向输入/输出口, 每个位可由掩膜选项配置成唤醒输入。掩膜选项决定引脚 PA0~PA3 是配置成 CMOS 输出或 NMOS 输入/输出引脚。如果 PA0~PA3 配置成 NMOS 输入/输出引脚, 则可以使用上拉电阻选项, 而且是同时加到这 4 个引脚上, 不是加到单独的引脚。引脚 PA4~PA7 是一直配置成具有上拉电阻的 NMOS 输入/输出引脚。所有的输入都是斯密特触发器输入。引脚 PA0、PA1 和 PA3 分别与 BZ、 $\overline{\text{BZ}}$ 和 PFD 引脚共用, 该功能由掩膜选项选择。
PB0/ $\overline{\text{INT0}}$ PB1/ $\overline{\text{INT1}}$ PB2/TMR PB3~PB5	I	—	6 位斯密特触发器输入口。每个输入引脚连接一个内部上拉电阻。PB0、PB1 和 PB2 分别与 $\overline{\text{INT0}}$ 、 $\overline{\text{INT1}}$ 和 TMR 引脚共用。
VLCD	I	—	提供 HT49R30A-1/HT49C30-1 的 LCD 电源。 提供 HT49C30L 的 LCD 电压泵。
V2	I	—	提供 HT49R30A-1/HT49C30-1 的 LCD 电压泵。 提供 HT49C30L 的 LCD 电源。
V1, C1, C2	I	—	提供 LCD 的电压泵。
COM0~COM2 COM3/SEG18	O	1/2, 1/3 or 1/4 Duty	1/4 LCD 占空比周期掩膜选项决定在 LCD 面板上的引脚 COM3/SEG18 是配置成 SEG18 驱动还是 COM3 输出驱动。COM0~COM2 是 LCD 的 common 输出。
SEG0~SEG17	O	—	LCD 面板中 Segment 部分的 LCD 驱动输出。
OSC1 OSC2	I O	Crystal or RC	OSC1、OSC2 连接外部 RC 电路或晶体振荡器(由掩膜选项决定)作为内部系统时钟。对于外部 RC 系统时钟的操作, OSC2 的输出端信号是系统时钟四分频。如果使用 OSC3 和 OSC4 引脚上的 RTC 振荡器作为系统时钟, 则 OSC1 和 OSC2 引脚应该悬空。
OSC3 OSC4	I O	RTC or System Clock	OSC3 和 OSC4 连接一个 32768Hz 的晶体振荡器形成一个定时用的实时时钟, 或是形成一个系统时钟。
$\overline{\text{RES}}$	I	—	触发复位输入。低电平有效。
VDD	—	—	正电源供应。
VSS	—	—	负电源供应, 接地。

- 注意:**
1. PA 上的每个引脚可通过掩膜选项被设定成具有唤醒功能。
  2. 只有引脚 PA0~PA3 具有上拉电阻掩膜选项，且只有当这四个引脚被配置成 NMOS 时，才能选择此选项。如果选择了上拉选项，则所有的四个引脚都将被连接到上拉电阻，引脚不能单独的选择上拉电阻。如果设定成 CMOS 输出，则上拉电阻掩膜选项无效。

**HT49R50A-1/HT49C50-1/HT49C50L**

引脚名称	I/O	掩膜选项	说明
PA0/BZ PA1/ $\overline{\text{BZ}}$ PA2 PA3/PFD PA4~PA7	I/O	Wake-up CMOS or NMOS Pull-high PA0/PA1 or BZ/ $\overline{\text{BZ}}$ PA3 or PFD	8 位双向输入/输出口，每个位可由掩膜选项配置成唤醒输入。掩膜选项决定引脚 PA0~PA3 是配置成 CMOS 输出或 NMOS 输入/输出引脚。如果 PA0~PA3 配置成 NMOS 输入/输出引脚，则可以使用上拉电阻选项，而且是同时加到这 4 个引脚上，不是加到单独的引脚。引脚 PA4~PA7 是一直配置成具有上拉电阻的 NMOS 输入/输出引脚。所有的输入都是斯密特触发器输入引脚 PA0、PA1 和 PA3 分别与 BZ、 $\overline{\text{BZ}}$ 和 PFD 引脚共用，该功能由掩膜选项选择。
PB0/ $\overline{\text{INT0}}$ PB1/ $\overline{\text{INT1}}$ PB2/TMR0 PB3/TMR1 PB4~PB7	I	—	8 位斯密特触发器输入口。每个输入引脚连接一个内部上拉电阻。引脚 PB0 和 PB1 分别与 $\overline{\text{INT0}}$ 和 $\overline{\text{INT1}}$ 引脚共用。引脚 PB2 和 PB3 分别与 TMR0 和 TMR1 引脚共用。
PC0~PC3	I/O	CMOS or NMOS Pull-high	4 位双向输入/输出口。掩膜选项决定所有四个引脚 PC0~PC3 是配置成 CMOS 输出或 NMOS 输入/输出引脚。引脚不能单独的被选择配置成 CMOS 输出或 NMOS 输入/输出引脚。如果 PC0~PC3 配置成 NMOS 输入/输出引脚，则上拉电阻选项是有效的，但同样的，只能同时提供给所有的四个引脚，不能单独的配置。所有的输入都是斯密特触发器输入
VLCD	I	—	提供 HT49R50A-1/HT49C50-1 的 LCD 电源。 提供 HT49C50L 的 LCD 电压泵。
V2	I	—	提供 HT49R50A-1/HT49C50-1 的 LCD 电压泵。 提供 HT49C50L 的 LCD 电源。
V1, C1, C2	I	—	LCD 电压泵。
COM0~COM2 COM3/SEG32	O	1/2, 1/3 or 1/4 Duty	1/4 LCD 占空比周期掩膜选项决定在 LCD 面板上的引脚 COM3/SEG32 是配置成 SEG32 驱动还是 COM3 输出驱动。COM0~COM2 是 LCD 的 common 输出。
SEG0~SEG31	O	—	LCD 面板中 segment 部分的 LCD 驱动输出。
OSC1 OSC2	I O	Crystal or RC	OSC1、OSC2 连接外部 RC 电路或晶体振荡器(由掩膜选项决定)作为内部系统时钟。对于外部 RC 系统时钟的操作，OSC2 的输出端信号是系统时钟四分频。如果使用 OSC3 和 OSC4 引脚上的 RTC 振荡器作为系统时钟，则 OSC1 和 OSC2 引脚应该被悬空。
$\overline{\text{RES}}$	I	—	触发复位输入。低电平有效。

引脚名称	I/O	掩膜选项	说明
OSC3 OSC4	I O	RTC or System Clock	OSC3 和 OSC4 连接一个 32768Hz 的晶体振荡器形成一个定时用的实时时钟，或是形成一个系统时钟。
VDD	—	—	正电源供应。
VSS	—	—	负电源供应，接地。

- 注意：**
1. PA 上的每个引脚可通过掩膜选项被设定成具有唤醒功能。
  2. 只有引脚 PA0~PA3 和 PC0~PC3 具有上拉电阻掩膜选项。且只有当四个引脚被配置成 NMOS 时，才能选择此选项。如果选择了上拉选项，则所有的四个引脚都将被连接到上拉电阻，引脚不能单独的选择上拉电阻。如果设定成 CMOS 输出，则上拉掩膜选项无效。
  3. 由于 48-pin SSOP 封装的限制，I/O 引脚的 PB6 和 PB7、PC 端口的引脚和 LCD 部分的引脚 SEG0~SEG9 与 SEG28~SEG31 不存在。

**HT49R70A-1/HT49C70-1/HT49C70L**

引脚名称	I/O	掩膜选项	说明
PA0/BZ PA1/ $\overline{\text{BZ}}$ PA2 PA3/PFD PA4~PA7	I/O	Wake-up CMOS or NMOS Pull-high PA0/PA1 or BZ/ $\overline{\text{BZ}}$ PA3 or PFD	8 位双向输入/输出口，每个位可由掩膜选项配置成唤醒输入。掩膜选项决定引脚 PA0~PA3 是配置成 CMOS 输出或 NMOS 输入/输出引脚。如果 PA0~PA3 配置成 NMOS 输入/输出引脚，则可以使用上拉电阻选项，而且是同时加到这 4 个引脚上，不是加到单独的引脚。引脚 PA4~PA7 是一直配置成具有上拉电阻的 NMOS 输入/输出引脚。所有的输入都是斯密特触发器输入引脚 PA0、PA1 和 PA3 分别与 BZ， $\overline{\text{BZ}}$ 和 PFD 引脚共用，该功能由掩膜选项选择。
PB0/ $\overline{\text{INT0}}$ PB1/ $\overline{\text{INT1}}$ PB2/TMR0 PB3/TMR1 PB4~PB7	I	—	8 位斯密特触发器输入口。每个输入引脚连接一个内部上拉电阻。引脚 PB0 和 PB1 分别与 $\overline{\text{INT0}}$ 和 $\overline{\text{INT1}}$ 引脚共用。引脚 PB2 和 PB3 分别与 TMR0 和 TMR1 引脚共用。
PC0~PC7	I/O	CMOS or NMOS Pull-high	8 位双向输入/输出口。两个掩膜选项决定 PC0~PC3 四个引脚和 PC4~PC7 四个引脚是配置成 CMOS 输出或 NMOS 输入/输出引脚。四个引脚必须作为一个整体被配置成 CMOS 输出或 NMOS 输入/输出引脚，不能单独的选择引脚。如果 PC0~PC3 或 PC4~PC7 配置成 NMOS 输入/输出引脚，则上拉电阻选项是有效的，但同样的，只能同时提供给这四个引脚，不能单独的配置。所有的输入都是斯密特触发器输入
VLCD	I	—	提供 HT49R70A-1/HT49C70-1 的 LCD 电源。 提供 HT49C70L 的 LCD 电压泵。
V2	I	—	提供 HT49R70A-1/HT49C70-1 的 LCD 电压泵。 提供 HT49C70L 的 LCD 电源。

引脚名称	I/O	掩膜选项	说明
V1, C1, C2	I	—	LCD 电压泵。
COM0~COM2 COM3/SEG40	O	1/2, 1/3 or 1/4 Duty	1/4 LCD 占空比周期掩膜选项决定在 LCD 面板上的引脚 COM3/SEG40 是配置成 SEG40 驱动还是 COM3 输出驱动。COM0~COM2 是 LCD 的 common 输出。
SEG0~SEG39	O	—	LCD 面板中 segment 部分的 LCD 驱动输出。
OSC1 OSC2	I O	Crystal or RC	OSC1、OSC2 连接外部 RC 电路或晶体振荡器(由掩膜选项决定)作为内部系统时钟。对于外部 RC 系统时钟的操作, OSC2 的输出端信号是系统时钟四分频。如果使用 OSC3 和 OSC4 引脚上的 RTC 振荡器作为系统时钟, 则 OSC1 和 OSC2 引脚应该被悬空。
OSC3 OSC4	I O	RTC or System Clock	OSC3 和 OSC4 连接一个 32768Hz 的晶体振荡器形成一个定时用的实时时钟, 或是形成一个系统时钟。
RES	I	—	触发复位输入。低电平有效。
VDD	—	—	正电源供应。
VSS	—	—	负电源供应, 接地。

- 注意:**
1. PA 上的每个引脚可通过掩膜选项被设定成具有唤醒功能。
  2. 只有引脚 PA0~PA3、PC0~PC3 和 PC4~PC7 具有上拉电阻掩膜选项。且只有当四个引脚被配置成 NMOS 时, 才能选择此项。如果选择了上拉选项, 则所有的四个引脚都将被连接到上拉电阻, 引脚不能单独的选择上拉电阻。如果设定成 CMOS 输出, 则上拉掩膜选项无效。

#### HT49RU80/HT49U80

引脚名称	I/O	掩膜选项	说明
PA0/BZ PA1/ $\overline{\text{BZ}}$ PA2 PA3/PFD PA4~PA7	I/O	Wake-up CMOS or NMOS Pull-high PA0/PA1 or BZ/ $\overline{\text{BZ}}$ PA3 or PFD	8 位双向输入/输出口, 每个位可由掩膜选项配置成唤醒输入。掩膜选项决定引脚 PA0~PA3 是配置成 CMOS 输出或 NMOS 输入/输出引脚。如果 PA0~PA3 配置成 NMOS 输入/输出引脚, 则可以使用上拉电阻选项, 而且是同时加到这 4 个引脚上, 不是加到单独的引脚。引脚 PA4~PA7 是一直配置成具有上拉电阻的 NMOS 输入/输出引脚。所有的输入都是斯密特触发器输入引脚 PA0、PA1 和 PA3 分别与 BZ, $\overline{\text{BZ}}$ 和 PFD 引脚共用, 该功能由掩膜选项选择。
PB0/ $\overline{\text{INT0}}$ PB1/ $\overline{\text{INT1}}$ PB2/TMR0 PB3/TMR1 PB4/TMR2 PB5~PB7	I	—	8 位斯密特触发器输入口。每个输入引脚连接一个内部上拉电阻。引脚 PB0 和 PB1 分别与 $\overline{\text{INT0}}$ 和 $\overline{\text{INT1}}$ 引脚共用。引脚 PB2, PB3 和 PB4 分别与 TMR0, TMR1 和 TMR2 引脚共用。

引脚名称	I/O	掩膜选项	说明
PC0/TX PC1/RX PC2~PC7	I/O	CMOS or NMOS Pull-high	8 位双向输入/输出口。两个掩膜选项决定 PC0~PC3 四个引脚和 PC4~PC7 四个引脚是配置成 CMOS 输出或 NMOS 输入/输出引脚。四个引脚必须作为一个整体被配置成 CMOS 输出或 NMOS 输入/输出引脚，不能单独的选择引脚。如果 PC0~PC3 或 PC4~PC7 配置成 NMOS 输入/输出引脚，则上拉电阻选项是有效的，但同样的，只能同时提供给这四个引脚，不能单独的配置。所有的输入都是斯密特触发器输入。引脚 PC0 和 PC1 分别与 UART 的 TX 和 RX 引脚共用。
PD0/SEG40~ PD6/SEG46	O	CMOS Output Or SEG Output	7 位输出口。每个引脚可由掩膜选项配置为 CMOS 输出或 SEG
VLCD	I	—	提供 LCD 电源。这个脚位只是用于 LCD 电源。VLCD 的电平可以高于或低于 VDD。
VMAX	—	—	IC 最大电压。连接到 VDD,VLCD 或 V1。
V1,V2,C1,C2	I	—	LCD 电压泵。
COM0~COM2 COM3/SEG47	O	1/2, 1/3 or 1/4 Duty	1/4 LCD 占空比周期掩膜选项决定在 LCD 面板上的引脚 COM3/SEG47 是配置成 SEG47 驱动还是 COM3 输出驱动。COM0~COM2 是 LCD 的 common 输出。
SEG0~SEG39	O	—	LCD 面板中 segment 部分的 LCD 驱动输出。
OSC1 OSC2	I O	Crystal or RC	OSC1、OSC2 连接外部 RC 电路或晶体振荡器(由掩膜选项决定)作为内部系统时钟。对于外部 RC 系统时钟的操作，OSC2 的输出端信号是系统时钟四分频。如果使用 OSC3 和 OSC4 引脚上的 RTC 振荡器作为系统时钟，则 OSC1 和 OSC2 引脚应该被悬空。
OSC3 OSC4	I O	RTC or System Clock	OSC3 和 OSC4 连接一个 32768Hz 的晶体振荡器形成一个定时用的实时时钟，或是形成一个系统时钟。
RES	I	—	触发复位输入。低电平有效。
VDD	—	—	正电源供应。
VSS	—	—	负电源供应，接地。

- 注意:**
1. PA 上的每个引脚可通过掩膜选项被设定成具有唤醒功能。
  2. 只有引脚 PA0~PA3、PC0~PC3 和 PC4~PC7 具有上拉电阻掩膜选项。且只有当四个引脚被配置成 NMOS 时，才能选择此项。如果选择了上拉选项，则所有的四个引脚都将被连接到上拉电阻，引脚不能单独的选择上拉电阻。如果设定成 CMOS 输出，则上拉掩膜选项无效。



## 极限参数

供应电压 (HT49C30L、HT49C50L、HT49C70L 除外) .....	$V_{SS}-0.3V$ to $V_{SS}+6.0V$
供应电压 (对于 HT49C30L、HT49C50L、HT49C70L) .....	$V_{SS}-0.3V$ to $V_{SS}+2.5V$
输入电压 .....	$V_{SS}-0.3V$ to $V_{DD}+0.3V$
储存温度 .....	$-50^{\circ}C$ ~ $125^{\circ}C$
工作温度 .....	$-40^{\circ}C$ ~ $85^{\circ}C$

这里只强调额定功率，超过极限参数功率的范围将对芯片造成损害，芯片在所标示范围外的表现并不能预期，而长期工作在标示范围外条件下也可能影响芯片的可靠性。

## 直流电气特性

HT49RU80/HT49CU80 除外

 $V_{DD} = 3V$  &  $V_{DD} = 5V$  (HT49C30L, HT49C50L, HT49C70L 除外)

 $V_{DD} = 1.5V$  (对于 HT49C30L, HT49C50L, HT49C70L)

 $T_a = 25^\circ C$ 

符号	参数	测试条件		最小	典型	最大	单位
		$V_{DD}$	条件				
$V_{DD}$	Operating Voltage	—	$f_{SYS} = 500kHz$ (For HT49C30L, HT49C50L, HT49C70L)	1.2	—	2.2	V
			$f_{SYS} = 4MHz$ LVR disabled, (Except HT49C30L, HT49C50L, HT49C70L)	2.2	—	5.5	V
			$f_{SYS} = 8MHz$ (Except HT49C30L, HT49C50L, HT49C70L)	3.3	—	5.5	V
VLCD	LCD Power Supply	—	$V_A \leq 5.5V$ (Except HT49C30L, HT49C50L, HT49C70L)	2.2	-	5.5	V
$I_{DD1}$	Operating Current (Crystal OSC)	1.5V	No load, $f_{SYS} = 455kHz$	—	60	100	$\mu A$
		3V	No load, $f_{SYS} = 4MHz$	—	1	2	mA
		5V	No load, $f_{SYS} = 4MHz$	—	3	5	mA
$I_{DD2}$	Operating Current (RC OSC)	1.5V	No load, $f_{SYS} = 400kHz$	—	50	100	$\mu A$
		3V	No load, $f_{SYS} = 4MHz$	—	1	2	mA
		5V	No load, $f_{SYS} = 4MHz$	—	3	5	mA
$I_{DD3}$	Operating Current (Crystal OSC, RC OSC)	5V	No load, $f_{SYS} = 8MHz$	—	4	8	mA
$I_{DD4}$	Operating Current ( $f_{SYS} = RTC$ OSC)	1.5V	No load	—	2.5	5	$\mu A$
		3V		—	0.3	0.6	mA
		5V		—	0.6	1	mA
$I_{STB1}$	Standby Current ( $f_S = T1$ )	1.5V	No load, system HALT	—	0.1	0.5	$\mu A$
		3V	LCD Off at HALT	—	—	1	$\mu A$
		5V	LCD Off at HALT	—	—	2	$\mu A$
$I_{STB2}$	Standby Current ( $f_S = RTC$ OSC)	1.5V	No load, system HALT	—	1	2	$\mu A$
		3V	LCD On at HALT, C type	—	2.5	5	$\mu A$
		5V	LCD On at HALT, C type	—	10	20	$\mu A$
$I_{STB3}$	Standby Current ( $f_S = WDT$ RC OSC)	1.5V	No load, system HALT	—	0.5	1	$\mu A$
		3V	LCD On at HALT, C type	—	2	5	$\mu A$
		5V	LCD On at HALT, C type	—	6	10	$\mu A$

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
I <sub>STB4</sub>	Standby Current (f <sub>s</sub> = RTC OSC)	3V	No load, system HALT LCD On at HALT, R type,	—	17	30	μA
		5V	1/2 bias	—	34	60	μA
I <sub>STB5</sub>	Standby Current (f <sub>s</sub> = RTC OSC)	3V	No load, system HALT LCD On at HALT, R type,	—	13	25	μA
		5V	1/3 bias	—	26	50	μA
I <sub>STB6</sub>	Standby Current (f <sub>s</sub> = WDT RC OSC)	3V	No load, system HALT LCD On at HALT, R type,	—	14	25	μA
		5V	1/2 bias	—	28	50	μA
I <sub>STB7</sub>	Standby Current (f <sub>s</sub> = WDT RC OSC)	3V	No load, system HALT LCD On at HALT, R type,	—	10	20	μA
		5V	1/3 bias	—	20	40	μA
V <sub>IL1</sub>	Input Low Voltage for I/O Ports, TMR and $\overline{\text{INT}}$	—	—	0	—	0.3V <sub>DD</sub>	V
V <sub>IH1</sub>	Input High Voltage for I/O Ports, TMR and $\overline{\text{INT}}$	—	(For HT49C30L, HT49C50L, HT49C70L)	0.8V <sub>DD</sub>	—	V <sub>DD</sub>	V
		—	(Except HT49C30L HT49C50L, HT49C70L)	0.7V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>IL2</sub>	Input Low Voltage (RES)	—	—	0	—	0.4V <sub>DD</sub>	V
V <sub>IH2</sub>	Input High Voltage (RES)	—	—	0.9V <sub>DD</sub>	—	V <sub>DD</sub>	V
I <sub>OL1</sub>	I/O Port Sink Current	1.5V	V <sub>OL</sub> =0.1V <sub>DD</sub>	0.4	0.8	—	mA
		3V		6	12	—	mA
		5V		10	25	—	mA
I <sub>OH1</sub>	I/O Port Source Current	1.5V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-0.3	-0.6	—	mA
		3V		-2	-4	—	mA
		5V		-5	-8	—	mA
I <sub>OL2</sub>	LCD Common and Segment Current	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	210	420	—	μA
		5V		350	700	—	μA
I <sub>OH2</sub>	LCD Common and Segment Current	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-80	-160	—	μA
		5V		-180	-360	—	μA
R <sub>PH</sub>	Pull-high Resistance	1.5V	—	75	150	300	kΩ
		3V		20	60	100	kΩ
		5V		10	30	50	kΩ
V <sub>LVR</sub>	Low Voltage Reset Voltage	—	—	2.7	3.2	3.6	V
V <sub>LVD</sub>	Low Voltage Detector Voltage	—	—	3.0	3.3	3.6	V

HT49RU80/HT49CU80

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
V <sub>DD</sub>	Operating Voltage	—	LVR disabled , f <sub>sys</sub> =4MHz	2.2	—	5.5	V
			LVR disabled , f <sub>sys</sub> =8MHz	3.3	—	5.5	V
V <sub>LCD</sub>	LCD Power Supply	—	VA ≤ 5.5V	2.2	—	5.5	V
I <sub>DD1</sub>	Operating Current (Crystal OSC, RC OSC)	3V	No load, f <sub>sys</sub> =4MHz	—	1	2	mA
		5V	UART Off	—	3	5	mA
I <sub>DD2</sub>	Operating Current (Crystal OSC, RC OSC)	3V	No load, f <sub>sys</sub> =4MHz	—	2	4	mA
		5V	UART On	—	5	10	μA
I <sub>DD3</sub>	Operating Current (Crystal OSC, RC OSC)	5V	No load, f <sub>sys</sub> =8MHz UART Off	—	4	8	mA
I <sub>DD4</sub>	Operating Current (Crystal OSC, RC OSC)	.5V	No load, f <sub>sys</sub> =8MHz UART On	—	6	12	mA
I <sub>DD5</sub>	Operating Current (f <sub>s</sub> = RTC OSC)	3V	No load, UART Off	—	0.3	0.6	mA
		5V		—	0.6	1	mA
I <sub>STB1</sub>	Standby Current (f <sub>s</sub> = T1)	3V	No load, system HALT	—	—	1	μA
		5V	LCD Off at HALT, UART Off	—	—	2	μA
I <sub>STB2</sub>	Standby Current (f <sub>s</sub> = RTC OSC)	3V	No load, system HALT	—	2.5	5	μA
		5V	LCD On at HALT, C type, UART Off	—	10	20	μA
I <sub>STB3</sub>	Standby Current (f <sub>s</sub> = WDT OSC)	3V	No load, system HALT	—	2	5	μA
		5V	LCD On at HALT, C type, UART On	—	6	10	μA
I <sub>STB4</sub>	Standby Current (f <sub>s</sub> = RTC OSC)	3V	No load, system HALT	—	17	30	μA
		5V	LCD On at HALT,R type, 1/2bias, UART Off	—	34	60	μA
I <sub>STB5</sub>	Standby Current (f <sub>s</sub> = RTC OSC)	3V	No load, system HALT	—	13	25	μA
		5V	LCD On at HALT, R type, 1/3 bias,UART Off	—	26	50	μA
I <sub>STB6</sub>	Standby Current (f <sub>s</sub> = WDT OSC)	3V	No load, system HALT	—	14	25	μA
		5V	LCD On at HALT, R type, 1/2 bias, UART Off	—	28	50	μA
I <sub>STB7</sub>	Standby Current (f <sub>s</sub> = WDT OSC)	3V	No load, system HALT	—	10	20	μA
		5V	LCD On at HALT, R type, 1/3 bias, UART Off	—	20	40	μA
V <sub>IL1</sub>	Input Low Voltage for I/O Ports, TMR and $\overline{\text{INT}}$	—	—	0	—	0.3V <sub>DD</sub>	V
V <sub>IH1</sub>	Input High Voltage for I/O Ports, TMR and $\overline{\text{INT}}$	—	—	0.7V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>IL2</sub>	Input Low Voltage (RES)	—	—	0	—	0.4V <sub>DD</sub>	V

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
V <sub>IH2</sub>	Input High Voltage ( $\overline{\text{RES}}$ )	—	—	0.9V <sub>DD</sub>	—	V <sub>DD</sub>	V
I <sub>OL1</sub>	I/O Port Sink Current	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	6	12	—	mA
		5V		10	25	—	
I <sub>OH1</sub>	I/O Port Source Current	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-2	-4	—	mA
		5V		-5	-8	—	mA
I <sub>OL2</sub>	LCD Common and Segment Current	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	210	420	—	μA
		5V		350	700	—	μA
I <sub>OH2</sub>	LCD Common and Segment Current	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-80	-160	—	μA
		5V		-180	-360	—	μA
R <sub>PH</sub>	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V		10	30	50	kΩ
V <sub>LVR</sub>	Low Voltage Reset Voltage	—	—	2.7	3.0	3.3	V
V <sub>LVD</sub>	Low Voltage Detector Voltage	—	—	3.0	3.3	3.6	V

## 交流电气特性

HT49RU80/HT49CU80 除外

$V_{DD} = 3V$  &  $V_{DD} = 5V$  (HT49C30L, HT49C50L, HT49C70L 除外)

$V_{DD} = 1.5V$  (对于 HT49C30L, HT49C50L, HT49C70L)

$T_a = 25^\circ C$

符号	参数	测试条件		最小	典型	最大	单位
		$V_{DD}$	条件				
$f_{SYS1}$	System Clock (Crystal OSC)	—	1.2V~2.2V (For HT49C30L, HT49C50L, HT49C70L)	400	—	500	kHz
		—	2.2V~5.5V (Except HT49C30L, HT49C50L, HT49C70L)	400	—	4000	kHz
		—	3.3V~5.5V (Except HT49C30L, HT49C50L, HT49C70L)	400	—	8000	kHz
$f_{SYS2}$	System Clock(RC OSC)	—	1.2V~2.2V (For HT49C30L, HT49C50L, HT49C70L)	400	—	500	kHz
		—	2.2V~5.5V (Except HT49C30L, HT49C50L, HT49C70L)	400	—	4000	kHz
		—	3.3V~5.5V (Except HT49C30L, HT49C50L, HT49C70L)	400	—	8000	kHz
$f_{SYS3}$	System Clock (32768Hz Crystal OSC)	—	—	—	32768	—	Hz
$f_{RTCOSC}$	RTC Frequency	—	—	—	32768	—	Hz
$f_{TIMER}$	Timer I/P Frequency	—	1.2V~2.2V (For HT49C30L, HT49C50L, HT49C70L)	0	—	500	kHz
		—	2.2V~5.5V (Except HT49C30L, HT49C50L, HT49C70L)	0	—	4000	kHz
		—	3.3V~5.5V (Except HT49C30L, HT49C50L, HT49C70L)	0	—	8000	kHz
$t_{WDTOSC}$	Watchdog Oscillator Period	1.5V	—	35	70	140	$\mu s$
		3V		45	90	180	$\mu s$
		5V		32	65	130	$\mu s$

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
t <sub>RES</sub>	External Reset Low Pulse Width	—	(For HT49C30L, HT49C50L, HT49C70L)	10	—	—	μs
			(Except HT49C30L, HT49C50L, HT49C70L)	1	—	—	μs
t <sub>SST</sub>	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	*t <sub>SYS</sub>
t <sub>LVR</sub>	Low Voltage Width to Reset	—	—	0.25	1	2	ms
t <sub>INT</sub>	Interrupt Pulse Width	—	(For HT49C30L, HT49C50L, HT49C70L)	10	—	—	μs
			(Except HT49C30L, HT49C50L, HT49C70L)	1	—	—	μs

\*t<sub>SYS</sub>=1/f<sub>SYS1</sub>, 1/f<sub>SYS2</sub> 或 1/f<sub>SYS3</sub>

**HT49RU80/HT49CU80**

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
f <sub>SYS1</sub>	System Clock (Crystal OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f <sub>SYS2</sub>	System Clock(RC OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f <sub>SYS3</sub>	System Clock (32768Hz Crystal OSC)	—	—	—	32768	—	Hz
f <sub>RTCOSC</sub>	RTC Frequency	—	—	—	32768	—	Hz
f <sub>TIMER</sub>	Timer I/P Frequency (50%Duty)	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
t <sub>WDTOSC</sub>	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V		32	65	130	μs
t <sub>RES</sub>	External Reset Low Pulse Width	—	—	1	—	—	μs
t <sub>SST</sub>	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	*t <sub>SYS</sub>
t <sub>LVR</sub>	Low Voltage Width to Reset	—	—	0.25	1	2	ms
t <sub>INT</sub>	Interrupt Pulse Width	—	—	1	—	—	μs

\*t<sub>SYS</sub>=1/f<sub>SYS1</sub>, 1/f<sub>SYS2</sub> 或 1/f<sub>SYS3</sub>

## 系统结构

内部系统结构是盛群半导体公司 LCD 型单片机具有良好运行性能的主要因素。由于采用 RISC 结构，此系列单片机具有高运算速度和高性能的特性。通过流水线的方式，即指令的取得和执行同时进行，此举使得除了分支、调用和查表指令外，其它指令都能在一个指令周期内完成。8 位的 ALU 参与指令集中所有的运算，它可完成算术运算、逻辑运算、移位、加、减和分支等功能，而内部的数据路径则以通过累加器或 ALU 的方式加以简化。有些寄存器在数据存储器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供最大可靠度和灵活性的 LCD 控制系统时，仅需要少数的外部器件。这使得这些单片机适合用在低成本高产量的控制应用上，可以提供 2K 至 16K 字的程序存储器和 96 至 576 字节数据储存。

### 时序和流水线结构(Pipelining)

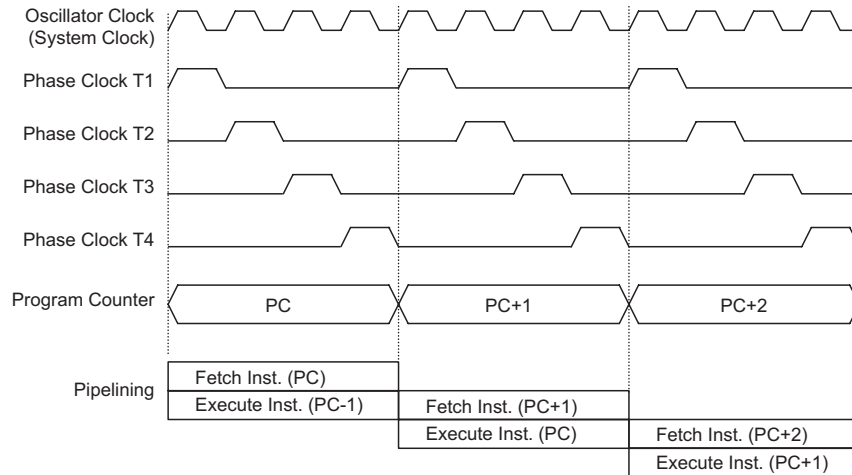
系统时钟由晶体/陶瓷振荡器，或是由 RC 振荡器提供，细分为 T1~T4 四个内部产生的非重叠时序。程序计数器在 T1 时自动加一并抓取一条新的指令。剩下的 T2~T4 时钟完成解码和执行功能，因此一个 T1~T4 时钟组成一个指令周期。虽然指令的取得和执行发生在连续的指令周期，但单片机流水线的结构会保证指令在一个指令周期内被有效的执行。特殊的情况发生在程序计数器的内容被改变的时候，如子程序的调用或跳转，在这情况下指令将需要多一个指令周期的时间去执行。

---

**注意：** 当使用 RC 振荡器时，OSC2 可以如同一个 T1 相时钟同步引脚一样地被使用，这个 T1 相时钟有  $f_{SYS}/4$  的频率，具有 1:3 高/低的占空比。

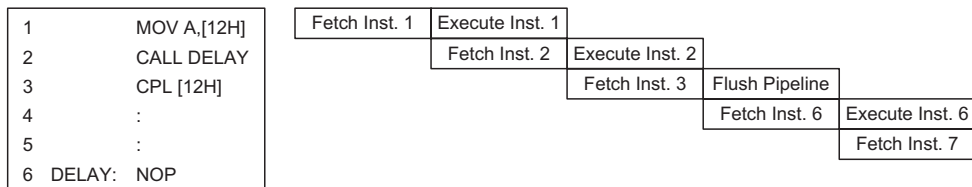
---





系统时序和流水线

如果指令牵涉到分支，例如跳转或调用等指令，则需要两个指令周期才能完成指令执行。需要一个额外周期的原因是程序先用一个周期取出当前指令地址的下一条指令，再用另一个周期去实际执行分支动作，因此程序设计师必须特别考虑额外周期的问题，尤其是在执行时间要求比较严格的时候。



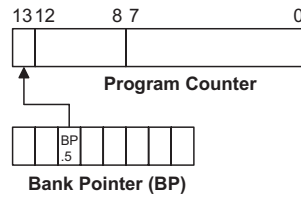
### 程序计数器

程序执行期间，程序计数器用来指向下一条要执行的指令地址。除了 JMP 或 CALL 这些要求跳转到一个非连续的存储器地址之外，它会在每条指令执行完后自动增加一。对于 LCD 系列的单片机，根据所选择的单片机型号不同，程序计数器宽度会因程序存储器容量的不同而不同。然而必须要注意只有较低的 8 位，即所谓的程序计数器低字节寄存器，是可以让使用者直接读写的。

当执行的指令要求跳转到非连续的地址时，如跳转指令、子程序调用、中断或复位等，单片机通过载入所需的地址到程序计数器来控制程序。对于条件跳转指令，一旦条件符合，下一条在现在指令执行时所取得的指令即会被摒弃，而由一个空指令周期来加以取代。

程序计数器较低字节，即程序计数器低字节寄存器或 PCL，可以通过程序控制取得，且它是可以读取和写入的寄存器。通过直接传送数据到这寄存器，一个程序短跳转可以直接被执行，然而因为只有低字节的运用是有效的，因此跳转被限制在同页存储器，即 256 个存储器地址的范围内，当这样一个程序跳转要执行时，需注意会插入一个空指令周期。

对于程序存储器分为两个 bank 的 HT49RU80/HT49CU80，由 BP 的第 5 位选择 Bank。Bank 选择位与程序计数器关系如图所示：



**注意：** 程序计数器较低字节在程序控制下是完全可用的。PCL 的使用可能导致程序分支，所以额外的周期需要预先取得。有关 PCL 寄存器更多的信息可在特殊功能寄存器部份中找到。

### HT49RU80/HT49CU80 除外

模式	程序计数器												
	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
复位	0	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	0	0	1	0	0
外部中断 1	0	0	0	0	0	0	0	0	0	1	0	0	0
定时/计数器 0 溢出	0	0	0	0	0	0	0	0	0	1	1	0	0
定时/计数器 1 溢出 (HT49R30A-1/HT49C30-1/ HT49C30L 除外)	0	0	0	0	0	0	0	0	1	0	0	0	0
时基中断 (对于 HT49R30A-1/ HT49C30-1/HT49C30L)	0	0	0	0	0	0	0	0	1	0	0	0	0
时基中断 (HT49R30A-1/HT49C30-1/ HT49C30L 除外)	0	0	0	0	0	0	0	0	1	0	1	0	0
RTC 中断 (对于 HT49R30A-1/ HT49C30-1/HT49C30L)	0	0	0	0	0	0	0	0	1	0	1	0	0

RTC 中断 (HT49R30A-1/HT49C30-1/ HT49C30L 除外)	0	0	0	0	0	0	0	0	1	1	0	0	0
条件跳转	Program Counter + 2												
写入 PCL 寄存器	PC12	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
跳转或调用子程序	#12	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0

**HT49RU80/HT49CU80**

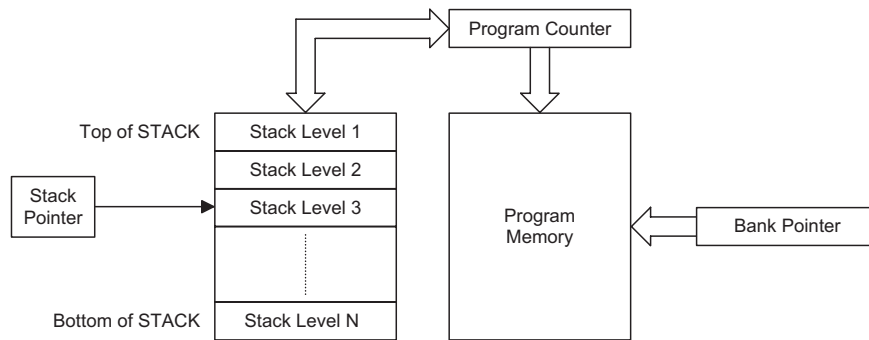
模式	程序计数器													
	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0
外部中断 0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
外部中断 1	0	0	0	0	0	0	0	0	0	0	1	0	0	0
定时/计数器 0 溢出	0	0	0	0	0	0	0	0	0	0	1	1	0	0
定时/计数器 1 溢出	0	0	0	0	0	0	0	0	0	1	0	0	0	0
UART Bus interrupt	0	0	0	0	0	0	0	0	0	1	0	1	0	0
Multi-function interrupt	0	0	0	0	0	0	0	0	0	1	1	0	0	0
条件跳转	Program Counter + 2 (在同 bank 内)													
写入 PCL 寄存器	PC13	PC12	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
跳转或调用子程序	BP.5	#12	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
由子程序返回	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

- 注意:**
1. PC13~PC8: 目前程序计数器位
  2. @7~@0: PCL 位
  3. BP.5: Bank 选择位
  4. #12~#0: 指令码位
  5. S13~S0: 堆栈寄存器位
  6. 对于 HT49RU80/HT49CU80, 程序计数器有 14 位, 即从 b13~b0。
  7. 对于 HT49R70A-1/HT49C70-1/HT49C70L, 程序计数器有 13 位, 即从 b12~b0。
  8. 对于 HT49R50A-1/HT49C50-1/HT49C50L, 由于程序计数器只有 12 位, 表格中的列 b12 是无效的。
  9. 对于 HT49R30A-1/HT49C30-1/HT49C30L, 由于程序计数器只有 11 位, 表格中的列 b11 和 b12 是无效的。
  10. 定时/计数器 1 溢出行只有 HT49R50A-1/HT49C50-1/HT49C50L, HT49R70A-1/HT49C70-1/HT49C70L 和 HT49RU80/HT49CU80 可利用。
  11. 对于 HT49R30A-1/HT49C30-1/HT49C30L, 定时/计数器 0 即 TMR。
  12. UART 和多功能中断只有在 HT49RU80/HT49CU80 中可用。

### 堆栈

堆栈是存储器中一个特殊的部分，它只用来储存程序计数器中的内容。根据选择的单片机，堆栈可介于 4、6 或 16 层之间，它们既不是数据部份也不是程序空间部份，且既不是可读取也不是可写入的。当前层由堆栈指针(Stack Pointer, SP)加以指示，同样也是不可读写的。在子程序调用或中断响应服务时，程序计数器的内容被压入到堆栈中。当子程序或中断服务程序结束时，返回指令 (RET 或 RETI)使程序计数器从堆栈中重新得到它以前的值。当一个芯片复位之后，SP 将指向堆栈的顶部。

如果堆栈已满，且有非屏蔽的中断发生，中断请求标志位会被置位，但是中断响应将被禁止。当堆栈指针减少(执行 RET 或 RETI)，中断将被响应。这个特性提供程序设计者简单的方法来预防堆栈溢出。然而即使堆栈已满，CALL 指令仍然可以被执行，而造成堆栈溢出。使用时应避免堆栈溢出的情况发生，因为这可能会造成不可预期的程序分支指令执行错误。



- 
- 注意:**
1. 对 HT49R30A-1/HT49C30-1/HT49C30L 而言，N=4，即 4 层堆栈可用。
  2. 对 HT49R50A-1/HT49C50-1/HT49C50L 而言，N=6，即 6 层堆栈可用。
  3. 对 HT49R70A-1/HT49C70-1/HT49C70L 和 HT49RU80/HT49CU80 而言，N=16，即 16 层堆栈可用。
-

### 算术及逻辑单位 – ALU

算术逻辑单元是单片机中很重要的部份，执行指令集中的算术和逻辑运算。ALU 连结到单片机的数据总线，在接收相关的指令码后执行需要的算术与逻辑操作，并将结果储存在指定的寄存器，当 ALU 计算或操作时，可能导致进位、借位或其它状态的变化，而相关的状态寄存器会因此更新内容以显示这些改变，ALU 所提供的功能如下：

- 算术运算：ADD、ADDM、ADC、ADCM、SUB、SUBM、SBC、SBCM、DAA
- 逻辑运算：AND、OR、XOR、ANDM、ORM、XORM、CPL、CPLA
- 移位：RRA、RR、RRCA、RRC、RLA、RL、RLCA、RLC
- 增加和减少：INCA、INC、DECA、DEC
- 分支判断：JMP、SZ、SZA、SNZ、SIZ、SDZ、SIZA、SDZA、CALL、RET、RETI

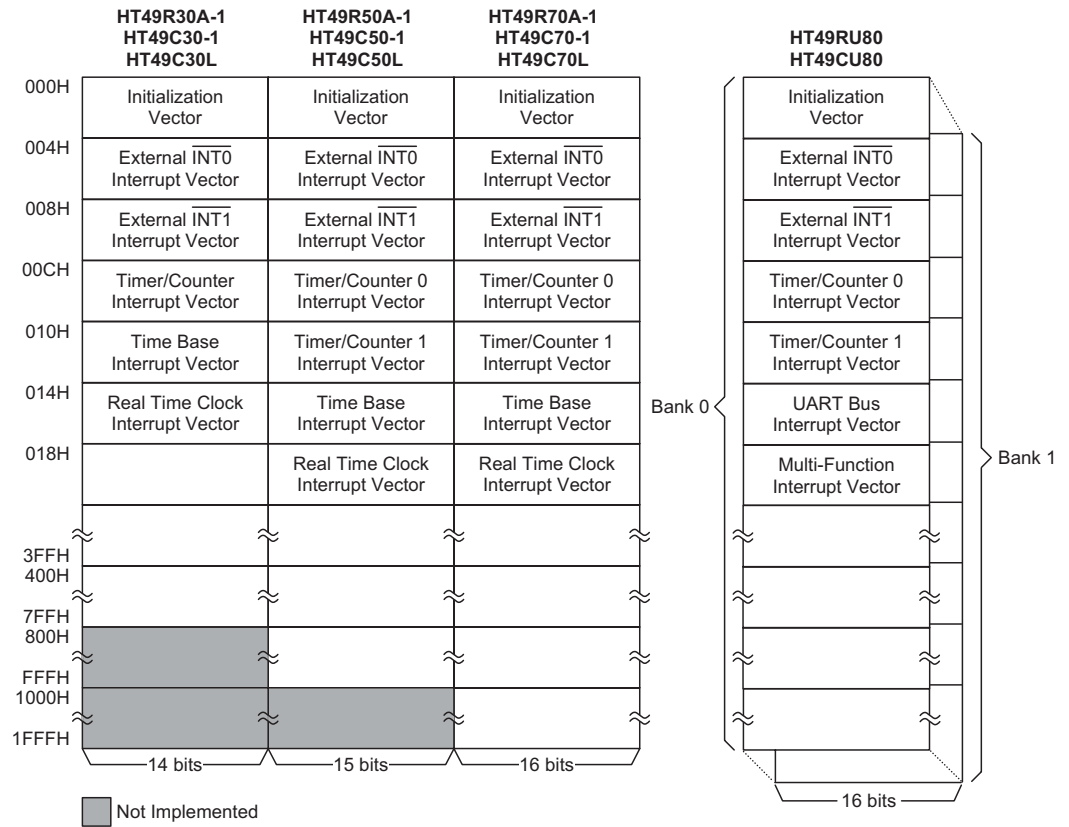
## 程序存储器

程序存储器用来存放用户代码即储存程序。对于 LCD 型的单片机而言，有两种程序存储器可供使用。第一种是一次可编程存储器(OTP)，使用者可编程他们的应用码到芯片中，具有 OTP 存储器的单片机在芯片名称上有“R”做标示。使用适当的烧录工具，OTP 单片机可以提供使用者以灵活的方式来自由开发他们的应用，这对于除错或需要经常升级与改变程序的产品是很有帮助的。对于中小型量产，OTP 亦为极佳的选择。另一种存储器为掩膜存储器，单片机名称上有“C”做标示，这些芯片对于大量生产提供最佳的成本效益。

### 结构

14 位的程序存储器的容量是 2K，16 位的程序存储器的容量则是 16K，这取决于选用哪种单片机。程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口。数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。对于 HT49RU80/HT49CU80 程序存储器分为两个 Bank，每个 Bank 的容量为 8K，它由 BP 的第 5 位来控制，清零选择 Bank0，置位选择 Bank1。由于数据存储器也由 BP 来控制，所以使用时必须注意。

以下是 LCD 型单片机程序存储器结构图。



### 特殊向量

程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。

- 地址 000H  
这个向量是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。
- 地址 004H  
这个向量用做  $\overline{\text{INT0}}$  外部中断入口，假如单片机外部中断引脚  $\overline{\text{INT0}}$  转成低电平，而外部  $\overline{\text{INT0}}$  中断使能且堆栈没有满的情况下，程序将跳到这个地址开始执行。
- 地址 008H  
这个向量用做  $\overline{\text{INT1}}$  外部中断入口，假如单片机外部中断引脚  $\overline{\text{INT1}}$  转成低电平，且外部  $\overline{\text{INT1}}$  中断使能且堆栈没有满的情况下，程序将跳到这个地址开始执行。
- 地址 00CH  
此内部中断向量被定时/计数器所使用，当定时器发生溢出，而内部中断使能且堆栈没有满的情况下，程序将跳到这个地址并开始执行。对于 HT49R50A-1/HT49C50-1/HT49C50L、HT49R70A-1/HT49C70-1/HT49C70L 和 HT49RU80/HT49CU80，这个定时/计数器称为定时/计数器 0。
- 地址 010H  
除了 HT49R30A-1/HT49C30-1/HT49C30L 只有一个内部定时/计数器外，这个内部中断向量被定时/计数器 1 所使用。当定时器发生溢出，而内部中断使能且堆栈没有满的情况下，程序将跳到这个地址并开始执行。而对于 HT49R30A-1/HT49C30-1/HT49C30L，这个向量被时基中断所使用。当产生时基中断信号，而内部中断使能且堆栈没有满的情况下，程序将跳到这个地址并开始执行。
- 地址 014H  
除了 HT49R30A-1/HT49C30-1/HT49C30L 和 HT49RU80/HT49CU80 外，这个内部中断向量被时基中断所使用。当产生时基中断信号，而内部中断使能且堆栈没有满的情况下，程序将跳到这个地址并开始执行。而对于 HT49R30A-1/HT49C30-1/HT49C30L，这个向量被实时时钟中断所使用。当

产生实时时钟中断信号，而内部中断使能且堆栈没有满的情况下，程序将跳到这个地址并开始执行。对于 HT49RU80/HT49CU80，此内部中断向量被 UART 使用。当发送或接收结束产生中断信号，而内部中断使能且堆栈没有满的情况下，程序将跳到这个地址并开始执行。

- 地址 018H

除了 HT49R30A-1/HT49C30-1/HT49C30L 和 HT49RU80/HT49CU80 外，这个内部中断向量被实时时钟中断所使用。而对于 HT49R30A-1/HT49C30-1/HT49C30L，它的实时时钟中断向量位于 014H，这个内部中断向量是无效的。当产生实时时钟中断信号，而内部中断使能且堆栈没有满的情况下，程序将跳到这个地址并开始执行。而对于 HT49RU80/HT49CU80 而言此内部中断被多功能中断所使用，如果定时器 2、实时时钟或时基产生中断，而内部中断使能且堆栈没有满的情况下，程序将跳到这个地址并开始执行。

### 多 Bank 管理

HT49RU80/HT49CU80 有多个程序存储 Bank，一些特殊的事项需要注意。首先，伪指令 ROMBANK 将程序段放置于不同的 Bank 中。当执行”CALL”指令来调用位于不同 Bank 的子程序，或者执行”JMP”指令来跳转到不同 Bank 的地址时，必须先正确设置 Bank 指针来确定目标 Bank。伪指令 ROMBANK 最佳用法如下所列。当执行”CALL”或”JMP”指令，BP 特殊寄存器中的 Bank 指针会自动装载到程序计数器中。若子程序被位于另一个 Bank 的主程序调用，当执这个子程序的”RET”指令时，程序会自动返回到原先主程序所在的 Bank，然而，BP 的值不会改变，它仍保持着子程序所在 Bank 的值。因此在不同的 Bank 之间的转移过程中，BP 必须小心处理。以下的范例就是母体 HT49RU80/HT49CU80 的程序，说明了如何在不同的 Bank 之间使用”CALL”和”JMP”指令。

```
include HT49RU80.inc
:
:
rombank 0 codesec0                ; define rombank 0
rombank 1 codesec1                ; define rombank 1
:
codesec0 .section at 000h 'code'   ; locates the following program section
; into Bank 0
```



```

clr bp                                ; re-initializing the BP
jmp start
:
:
start:
:
:
lab0:
:
:
mov a, BANK routb1                    ; routine routb1 is located in Bank 1
mov bp,a                              ; load bank number for routb1 into BP
call routb1                            ; call subroutine located in Bank 1
clr bp                                  ; program will return to this location
:                                       ; after RET in Bank 1
:                                       ; but BP will retain Bank 1 value
:                                       ; so clear the BP
codesec1 .section at 000h 'code'       ; locates following program section
:                                       ; into Bank 1
:
:
routb1 proc
:
:
ret                                    ; return program to Bank 0 but BP will
:                                       ; retain Bank 1 value

routb1 endp
:
:

```

中断处理时，Bank指针的处理需十分小心。无论程序运行至哪个Bank，一旦中断发生，包括外部中断和内部中断，程序都会立刻跳转至相应的位于Bank 0的中断子程序入口。然而要注意的是，虽然无论什么情况下程序都会跳转到Bank0，但是Bank指针仍然保持原来的值，而不是指向Bank 0。因为这个原因，进入中断处理子程序后，除了保存累加器和状态寄存器之外，另一个重要的动作就是，保存Bank指针以及清Bank指针让它指向Bank 0，尤其是在Bank 0中执行调用子程序或跳转指令。在”RETI”指令执行以前，Bank指针和累加器、状态特殊寄存器一样必须被恢复，确保程序返回至正确的Bank地址，并指向此地址。以下范例说明了如何处理中断子程序：

```

include HT49RU80.inc
:

```

```

:
rombank 0 codesec0          ; define rombank 0
rombank 1 codesec1          ; define rombank 1
:
:
codesec0 .section at 000h 'code' ; locates the following program section
; into Bank 0
clr bp                       ; clear Bank Pointer after power-on reset
:
:
org 004h                     ; jump here from any bank when ext0_int.
; occurs - BP retains original value
mov accbuf0,a                ; backup accumulator
mov a,bp                     ; backup Bank Pointer
clr bp                       ; clear BP to indicate Bank 0 otherwise
; original BP value will remain and give
; rise to false jmp or call addresses
jmp ext0_int                 ; jump to external 0 interrupt subroutine
:
:
org 008h                     ; jump here from any bank when ext1_int.
; occurs - BP retains original value
mov accbuf1,a                ; backup accumulator
mov a,bp                     ; backup Bank Pointer
clr bp                       ; clear BP to indicate Bank 0 otherwise
; original BP value will remain and give
; rise to false jmp or call addresses
jmp ext1_int                 ; jump to external 1 interrupt subroutine
:
:
org 00Ch                     ; jump here from any bank when Timer 0 int.
; occurs - BP retains original value
:
:
ext0_int:                    ; external 0 interrupt subroutine
mov bp_ext0,a                ; backup Bank Pointer
mov a,status                 ; backup status register
mov statusbuf0,a            ; backup status register
:
:
mov a,statusbuf0             ; restore status register
mov status,a
mov a,bp_ext0                ; restore Bank Pointer
mov bp,a

```

```

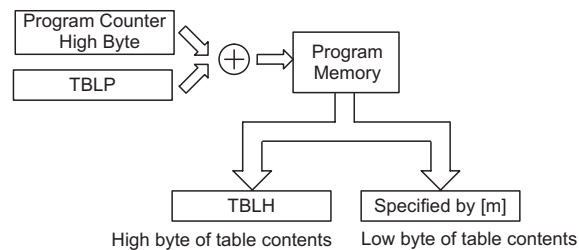
mov a,acbuf0          ; restore accumulator
reti                 ; return to main program and original
                    ; calling bank
:
:
:
ext1_int:            ; external 1 interrupt subroutine
mov bp_ext1,a       ; backup Bank Pointer
mov a,status        ; backup status register
mov statusbuf1,a
:
:
:
mov a,statusbuf1    ; restore status register
mov status,a
mov a,bp_ext1       ; restore Bank Pointer
mov bp,a
mov a,acbuf1        ; restore accumulator
reti                 ; return to main program and original
                    ; calling bank

```

### 查表

程序存储器中的任何地址都可以定义成一个表格，以便储存固定的数据。使用表格时，表格指针必须先行设定，其方式是将表格的低字节地址放在表格指针寄存器 TBLP 中，这个寄存器定义表格较低的 8 位地址。在设定完表格指针后，表格数据可以使用“TABRDC [m]”或“TABRDL [m]”指令从当前程序所在的存储器页或存储器最后一页中来查表读取。当这些指令执行时，程序存储器中表格数据的低字节，将被传送到使用者所指定的数据存储器，程序存储器中表格数据的高字节，则被传送到 TBLH 特殊寄存器，而高字节中未使用的位将被读取为“0”。

下图是除 HT49RU80/HT49CU80 以外的查表寻址/数据流程图：



### 查表程序范例

以下范例说明 HT49R30A-1/ HT49C30-1/ HT49C30L LCD 单片机中，表格指针和表格数据如何被定义和执行。这个例子使用的表格数据用 ORG 伪指令储存在存储器的最后一页，在此 ORG 伪指令中的值为 700H，即 2K 程序存储器 HT49R30A-1/ HT49C30-1/ HT49C30L 单片机中最后一页存储器的起始地址，而表格指针的初始值则为 06H，这可保证从数据表格读取的第一笔数据位于程序存储器地址 706H 即最后一页起始地址后的第六个地址。值得注意的是，假如“TABRDC [m]”指令被使用，则表格指针指向当前页。在这个例子中，表格数据的高字节等于零，而当“TABRDL [m]”指令被执行时，此值将会自动的被传送到 TBLH 寄存器。

```

tempreg1 db ? ; temporary register #1
tempreg2 db ? ; temporary register #2
:
:

mov a,06h ; initialize table pointer – note that this address
; is referenced

mov tblp,a ; to the last page or present page
:
:

tabrdl tempreg1 ; transfers value in table referenced by table pointer
; to tempreg1
; data at prog. memory address 706H transferred to
; tempreg1 and TBLH

dec tblp ; reduce value of table pointer by one

tabrdl tempreg2 ; transfers value in table referenced by table pointer
; to tempreg2
; data at prog. memory address 705H transferred to
; tempreg2 and TBLH
; in this example the data "1A" is transferred to
; tempreg1 and data "0F" to register tempreg2
; the value "0" will be transferred to the high byte
; register TBLH

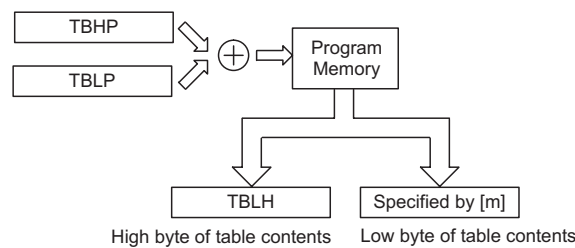
org 700h ; sets initial address of last page (for HT49R30A-1)

dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:

```

对于芯片 HT49RU80/HT49CU80，它们有两个表格指针寄存器 TBLP 和 TBHP，可以从中获取表格地址低字节和表格地址高字节，必须事先设定。不同于仅使用表格地址低字节 TBLP 的芯片，增加的 TBHP 寄存器允许在任何地址定义表格，并且允许连续直接访问任何地址任何页的表格数据。这类芯片中，当设定了低字节表格指针和高字节表格指针后，就可以使用指令“TABRDC [m]”获取任何程序存储器地址的表格数据，或者使用指令“TABRDL [m]”获取最后一页的表格数据。无论使用何种指令，程序存储器中的低字节表格数据都会被送往用户定义的数据寄存器[m]中。程序存储器中高字节表格数据会被送往 TBLH 特殊寄存器。高字节中未使用的位将被读取为零。

下图是 HT49RU80/HT49CU80 的查表寻址/数据流程图：



以下范例说明 HT49RU80/HT49CU80 如何定义表格指针、如何查表。这个例子使用的表格数据用 ORG 伪指令储存在存储器中。在此 ORG 伪指令中的值为 000H，这是相对于 Bank 1 起始地址而言，绝对地址是“2000H”。这里高字节表格指针是 20H，低字节表格指针的初值则为 05H。这可以保证从数据表格读取的第一笔数据位于程序存储器地址“2005H”，即 ORG 伪指令定义地址后五个地址。执行“TABRDC [m]”指令，表格数据低字节“FFH”被送往指定的“temp”寄存器，而表格数据高字节“55H”将会被传送到 TBLH 寄存器。

```

include HT49RU80.inc
:
:
data .section 'data'
temp db ?
:
:
rombank 0 codesec0 ; Bank 0 definition
  
```

```

rombank 1 codesec1 ; Bank 1 definition
:
:
codesec0 .section at 0 'code'
jmp start
:
org 010h
start:
:
:
mov a,020h ; setup table high byte address
mov tbhp,a
mov a,005h ; setup table low byte address
mov tblp,a ; table pointer address is now 2005H
tabrdc temp ; read table data from PC address 2005H
nop ; FFH will be placed in temp register
; and 55H will be placed in the TBLH
; register
codesec1 .section at 000h 'code' ; Bank 1 code located here
org 0000h ; this defines the offset from the
; start address of Bank 1 which is
; 2000H
dc 000AAh, 011BBh, 022CCh, 033DDh, 044EEh, 055FFh
:
:

```

因为 TBLH 寄存器是只读寄存器，不能重新储存，若主程序和中断服务程序都使用表格读取指令，应该注意它的保护。使用表格读取指令，中断服务程序可能会改变 TBLH 的值，若随后在主程序中再次使用这个值，则会发生错误。因此建议避免同时使用表格读取指令。然而在某些情况下，如果同时使用表格读取指令是不可避免的，则在执行任何主程序的表格读取指令前，中断应该先除能，另外要注意的是所有与表格相关的指令，都需要两个指令周期去完成操作。

**除 HT49RU80/HT49CU80**

指令	表格地址												
	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC [m]	PC12	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

## HT49RU80/HT49CU80

指令	表格地址													
	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC [m]	TBHP	TBHP	TBHP	TBHP	TBHP	TBHP	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

- 注意：**
1. PC12 ~ PC8：当前程序计数器位
  2. @7~@0：是表格指针 TBLP 位
  3. 对于 HT49RU80/HT49CU80，表格地址是 14 位，从 b13~b0。
  4. 对于 HT49R70A-1/HT49C70-1/HT49C70L，表格地址是 13 位，从 b12~b0。
  - 5 对于 HT49R50A-1/HT49C50-1/HT49C50L，表格地址是 12 位，从 b11~b0。
  6. 对于 HT49R30A-1/HT49C30-1/HT49C30L，表格地址是 11 位，从 b10~b0。

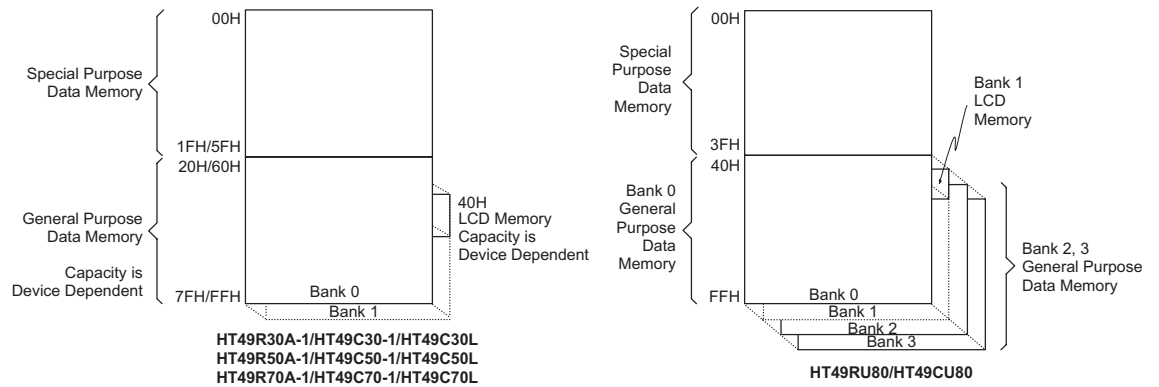
## 数据存储器的

数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据，且分为三部份。第一部份是特殊功能寄存器，这些寄存器有固定的地址且与单片机的正确操作密切相关。大多数特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二部份数据存储器是做一般用途使用，都可在程序控制下进行读取和写入。第三部分为 LCD 存储器。数据存储器的这个特殊区域直接映射到 LCD 显示器，写入这部分存储器的数据将直接影响显示的数据。LCD 存储器区和通用数据存储器区的地址重叠，可以通过设定正确的存储区指针值，对这两个区域进行切换。

### 结构

专用和通用数据存储器位于连续的地址。全部 RAM 为 8 位宽度，但存储器长度因所选择的单片机而不同。所有芯片的数据存储器的起始地址都是 00H。常见的寄存器，如 ACC 和 PCL 等，全都有相同的数据存储器地址。LCD 数据存储器对应到数据存储器的存储区 1，然而只有较低四位可使用。如果通过程序读取高四位的值，将会返回“0”。所有芯片的 LCD 数据存储器的起始地址为 40H。由于 LCD 数据存储器位于存储区 1，所以为了进入这个区域，存储区指针必须先设定为 01H。要注意的是，在上电后，数据存储器的内容，包括

LCD 数据存储器，是未知的，因此程序设计者必须对数据存储器进行适当的初始化。



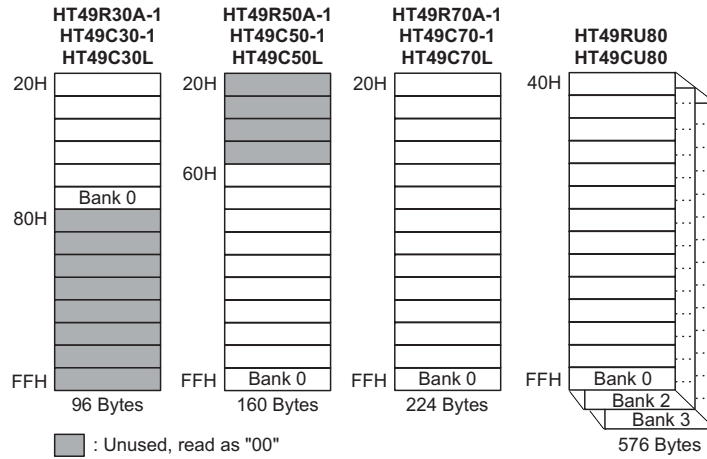
**注意：**除了少数专用的位，大部份数据存储器的位都可以直接使用“SET [m].i”和“CLR [m].i”加以操作。也可以通过间接寻址指针寄存器 MP0 和 MP1 对数据存储器进行存取。

### 通用数据存储器

所有的单片机程序需要一个读/写的存储区，让临时数据可以被储存和再使用。该 RAM 区域就是通用数据存储器。这个数据存储器区可让使用者进行读取和写入的操作。使用“SET [m].i”和“CLR [m].i”指令可对个别的位做置位或复位的操作，方便用户在数据存储器内进行位操作。而对于 HT49RU80/HT49CU80 而言，通用数据存储器位于存储区 0、存储区 2 和存储区 3。在对通用数据存储器进行存取操作之前，必须先正确的设定存储区指针的值。当存储区指针设定为 01H 时，将对 LCD 存储器进行存取。存储区 1、存储区 2 或存储区 3 必须使用间接寻址指针 MP1 和间接寻址寄存器 IAR1 进行间接存取。任何直接寻址或使用间接寻址指针 MP0 和间接寻址寄存器 IAR0 进行的间接寻址，只会在存储区 0 存取数据。

以下是 LCD 型单片机通用数据存储器的详细结构图：





**注意：**HT49RU80/HT49CU80 中，通用数据存储器的 576 个字节储存在三个独立的存储区，即存储区 0、存储区 2 和存储区 3。在读取或写入数据到通用数据存储器之前，必须先确认是否正确的设定了存储区指针，选择了正确的数据存储器区。存储区 1、存储区 2 和存储区 3 必须使用间接寻址指针 MP1 和间接寻址寄存器 IAR1 进行间接存取。

### 专用数据存储器

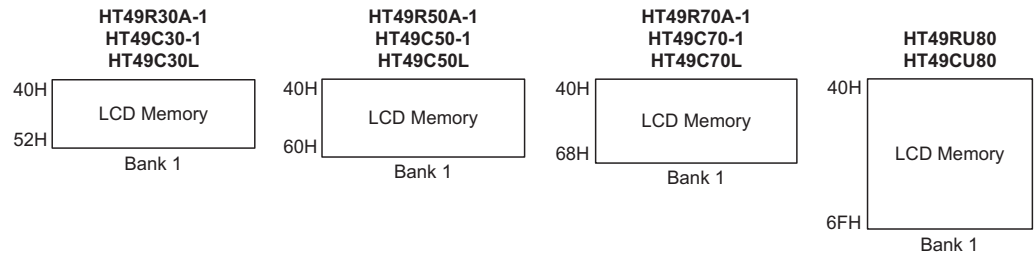
这个区域的数据存储器是存放特殊寄存器的，这些寄存器和单片机的正确操作密切相关。大多数的寄存器可进行读取和写入，只有一些是被保护而只能读取的，相关细节的介绍请参看有关特殊功能寄存器的部份。要注意的是，任何读取指令对存储器中未使用的地址进行读取将得到“00H”的值。

以下是 LCD 型单片机中，专用数据存储器的详细结构图：



须先确定存储区指针是否被设为 01H。LCD 存储器必须使用间接寻址指针 MP1 和间接寻址寄存器 IAR1 进行间接存取。当存储区指针设为存储区 1 时，是对 LCD 数据存储区进行存取，如果地址值小于 40H，则是对存储区 0 的通用数据存储区进行操作。同样的，如果存储区指针设为存储区 1，而读取到的地址高于最大地址，则返回 00H。

以下是 LCD 型单片机 LCD 存储器的详细结构图：



## 特殊功能寄存器

为了确保单片机能成功的操作，数据存储器中设置了一些内部寄存器。这些寄存器确保内部功能(如定时器、中断等)和外部功能(如输入/输出数据控制)的正确操作。在数据存储器中，这些寄存器以 00H 作为起始地址。在特殊功能寄存器存储空间和通用数据存储器的起始地址之间，有一些未定义的数据存储器，被保留用来做未来的扩充，若从这些地址读取数据将返回 00H 值。

### 间接寻址寄存器 – IAR0, IAR1

间接寻址的方法准许使用间接寻址指针做数据操作，以取代定义实际存储器地址的直接存储器寻址方式。在间接寻址寄存器上的任何动作，将对间接寻址指针(MP)所指定的存储器地址产生对应的读/写操作。所有的 LCD 系列单片机均含有两个间接寻址寄存器(IAR0 和 IAR1)和两个间接寻址指针(MP0 和 MP1)。要注意的是，这些间接寻址寄存器并不是实际存在的，直接读取 IAR 寄存器将返回 00H 的结果，而直接写入此寄存器则不做任何操作。

### 间接寻址指针 – MP0, MP1

所有的 LCD 系列单片机均含有两个间接寻址指针，即 MP0 和 MP1。由于这些指针在数据存储器中能象普通的寄存器一样被写入和操作，因此提供了一个寻址和数据跟踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由间接寻址指针所指定的地址。当存储区指针是设为从存储区 0 存取数据时，间接寻址指针 MP0 和 MP1 都可以用来从通用数据存储器存储区 0 存取数据。不过请注意，当存储区指针设为从存储区 1 或在 HT49RU80/HT49CU80 设为从存储区 2 或存储区 3，即 LCD 数据存储器，存取数据时，间接寻址指针 MP0 只能从存储区 0 存取数据，而只有间接寻址指针 MP1 可以从存储区 1，存储区 2 或存储区 3 存取数据。

---

**注意:** 对 HT49R30A-1/HT49C30-1/HT49C30L 而言，间接寻址指针的第 7 位没有作用。可是，必须注意当间接寻址指针被读取时，其值为 1。

---

以下的例子说明如何清除一个具 4 个 RAM 地址的区块，它们已事先被定义成地址 adres1 到 adres4

```

data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block db ?
code .section at 0 'code'
org 00h
start:
    mov a,04h                ; setup size of block
    mov block,a
    mov a,offset adres1     ; Accumulator loaded with first RAM address
    mov mp0,a               ; setup Memory Pointer with first RAM address
loop:
    clr IAR0                 ; clear the data at address defined by MP0
    inc mp0                  ; increment Memory Pointer
    sdz block                ; check if last memory location has been cleared
    jmp loop
continue:
    
```

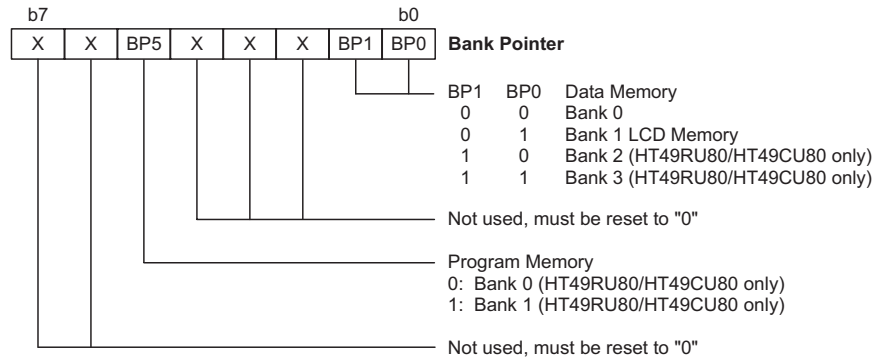
在上面的例子中有一值得注意，即并没有确定 RAM 地址。

### 存储区指针 – BP

请注意，在数据存储器区域中，通用数据存储器 and LCD 存储器具有相同的数据存储器地址。因此，当使用指令去存取 LCD 存储器或通用数据存储器的数据时，必须先确定所选的区域是正确的。除了 HT49RU80/HT49CU80，其它单片机的通用数据存储器位于存储区 0，而 HT49RU80/HT49CU80 的通用数据存储器分为三个存储区块（存储区 0、存储区 2 和存储区 3）。对于所有单片机来说，LCD 存储器均位于存储区 1，可以使用存储区指针 BP 来选择正确的数据存储器区。如果要对存储区 1，存储区 2 或存储区 3 进行数据存取，存储区指针 BP 的值必须先设置为“01H”，“02H”或“03H”。请注意，这三个存储区必须使用间接寻址指针 MP1 和间接寻址寄存器 IAR1 进行间接存取。任何的直接寻址或使用 MP0 和 IAR0 的间接寻址，只会在存储区 0 存取数据。除了在暂停模式下的 WDT 溢出复位，不会改变通用数据存储器的存储区号以外，

其他情况下复位后，通用数据存储器都会初始化到存储区 0。请注意，特殊功能数据存储器不受存储区的影响，也就是说，不论是在存储区 0、存储区 1、存储区 2 或存储区 3，都能对特殊功能寄存器进行读写操作。

对于 HT49RU80/HT49CU80，它的程序存储器分为两个 8K 的 Bank，即 Bank0 和 Bank1。存储区指针的第 5 位用来选择 Bank。尽管存储区指针寄存器某些位被用来指示存储区号，但该寄存器的 8 个位都可以使用。没有用到的那些位复位为“0”。



### 累加器 – ACC

对任何单片机来说，累加器是相当重要的且与 ALU 所完成的运算有密切关系，所有 ALU 得到的运算结果都会暂时储存在 ACC 累加器里。若没有累加器，ALU 必须在每次进行如加法、减法和移位的运算时，将结果写入到数据存储器，这样会造成程序编写和时间的负担。另外数据传送也常常牵涉到累加器的临时储存功能，例如在一使用者定义的寄存器和另一个寄存器之间传送数据时，由于两寄存器之间不能直接传送数据，因此必须通过累加器来传送数据。

### 程序计数器低字节寄存器 – PCL

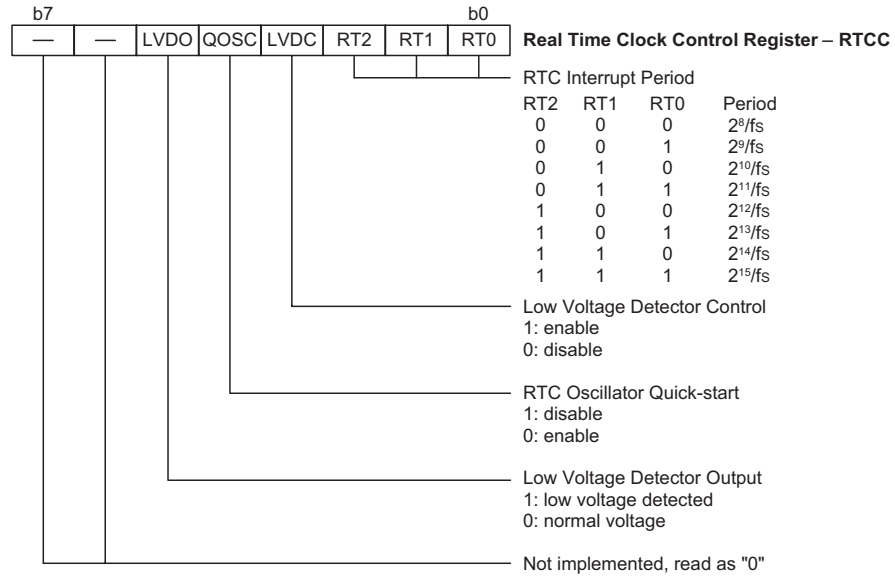
为了提供额外的程序控制功能，程序计数器较低字节设置在数据存储器的特殊功能区域内，程序员可对此寄存器进行操作，很容易的直接跳转到其它程序地址。直接给 PCL 寄存器赋值将导致程序直接跳转到程序存储器的某一地址，然而由于寄存器只有 8 位的长度，因此只允许在本页的程序存储器范围内进行跳转，而当使用这种运算时，要注意会插入一个空指令周期。

### 查表寄存器 – TBLP, TBHP, TBLH

这些特殊功能寄存器对储存在程序存储器中的表格进行操作。TBLP 为表格指针的低字节，指向表格数据的地址。它的值必须在任何表格读取指令执行前加以设定，由于它的值可以被如 INC 或 DEC 的指令所改变，这就提供了一种简单的方法对表格数据进行读取。对于 HT49RU80/HT49CU80，还有一个表格指针高字节 TBHP。表格读取数据指令执行之后，表格数据高字节存储在 TBLH 中。其中要注意的是，表格数据低字节会被传送到使用者指定地址。

### 实时时钟控制寄存器 – RTCC

RTCC 寄存器控制数个内部功能，实时时钟（RTC）中断就是其中一个，它的功能是提供一有周期固定的内部中断信号。RTC 中断的驱动时钟来自内部时钟源，即  $f_s$ 。它被进一步分频以获得更长的时间值，并由此产生中断信号。RTCC 寄存器的第 0~2 位，即 RT2~RT0，用以确定分频系数。直接将值写入此 RTCC 寄存器位，可产生  $2^8/f_s$  到  $2^{15}/f_s$  的溢出时间值。RTCC 寄存器也可控制 RTC 振荡器的快速启动功能。此振荡器具有固定的频率 32768Hz，通过设置第 4 位(即 QOSC 位)为“0”，可以以更快的速度启动。上电后，此位的值将被设置为“0”，然而这样会加大功耗，约 2 秒后 QOSC 位应该被设置为“1”，以降低功耗。低电压检测器是 RTCC 寄存器可以控制的另一个内部功能。通过把第 3 位(即 LVDC 位)置为“1”可以使能该功能。当电源供应的电压低于 DC 特性所标示的 VLVD 值时，第 5 位(即 LVDO 位，此位为只读)将被设置为“1”。如果电源供应的电压高于这个值，则这个位会保持“0”。请注意，RTCC 寄存器的第 6 位和第 7 位未定义。



### 状态寄存器 – STATUS

这 8 位寄存器(0AH)包含零标志位(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)和看门狗溢出标志位(TO)。它同时记录状态数据和控制运算顺序。

除了 TO 和 PDF 标志位外，状态寄存器中的位像其它大部份寄存器一样可以被改变，但任何数据写入到状态寄存器将不会改变 TO 或 PDF 标志位。另外，执行不同的指令后，与状态寄存器有关的运算可能会得到不同的结果。TO 标志位只会受系统上电、看门狗溢出、或执行“CLR WDT”或“HALT”指令影响。PDF 标志位只会受执行“HALT”或“CLR WDT”指令或系统上电影响。

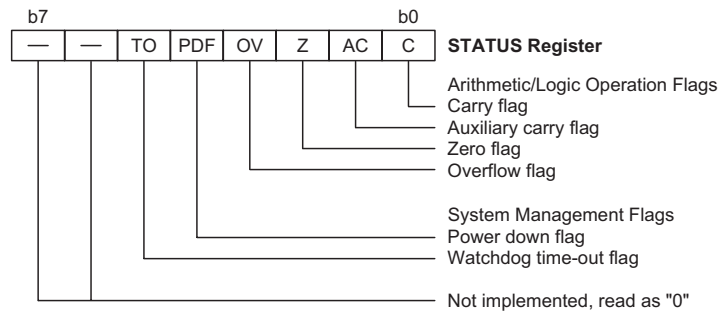
Z、OV、AC 和 C 标志位通常反映最近运算的状态

- 当加法运算的结果产生进位，或减法运算的结果没有产生借位时，则 C 被置位，否则 C 被清零，同时 C 也会被带进位/借位的移位指令所影响。
- 当低半字节加法运算的结果产生进位，或高半字节减法运算的结果没有产生借位时，AC 被置位，否则 AC 被清零。
- 当算术或逻辑运算结果是零时，Z 被置位，否则 Z 被清零。
- 当运算结果高两位的进位状态异或结果为 1 时，OV 被置位，否则 OV 被清零。



- 系统上电或执行“CLR WDT”指令会清零 PDF，而执行“HALT”指令则会置位 PDF。
- 系统上电或执行“CLR WDT”或“HALT”指令会清零 TO，而当 WDT 溢出则会置位 TO。

另外当进入一个中断程序或执行子程序调用时，状态寄存器不会自动压入到堆栈保存。假如状态寄存器的内容是重要的且子程序可能改变状态寄存器的话，则需谨慎的去做正确的储存。



### 中断控制寄存器 – INTC0, INTC1, MFIC

8 位的 INTC0、INTC1 和 MFIC 寄存器用来控制外部和内部中断的动作。通过使用标准的位操作指令来设定这寄存器的位的值，外部中断和内部中断的使能和除能功能可分别被控制。寄存器中主中断位(EMI)控制所有中断的使能/除能，用来设定所有中断使能位的开或关。当一个中断程序被响应时，就会自动屏蔽其它中断，EMI 位将被清零，而执行“RETI”指令则会置位 EMI 位。

---

**注意：** 若遇到在当前中断服务程序中要再响应其它的中断程序时，可以在进入该中断服务程序后，在程序中用手动的方式将 EMI 置为“1”。

---

### 定时/计数寄存器

该系列的单片机集成了一个、二个或三个8位或16位的定时/计数器，这取决于您选择的型号。对于具有一个8位定时器的HT49R30A-1/HT49C30-1/HT49C30L而言，寄存器TMR是8位定时数值存放的位置。对应的控制寄存器TMRC，含有此定时/计数器的设定信息。对于具有两个8位定时器的HT49R50A-1/HT49C50-1/HT49C50L而言，分别是TMR0和TMR1存放8位定时

数据，而控制寄存器TMR0C和TMR1C分别含有定时/计数器的设定信息。对于HT49R70A-1/HT49C70-1/HT49C70L来说，具有一个8位和一个16位的定时/计数器，定时数据分别存放在TMR0和TMR1L/TMR1H,相应的控制寄存器分别为TMR0C和TMR1C，含有定时/计数器的设定信息。HT49RU80/HT49CU80具有一个8位和两个16位的定时/计数器，定时数据分别存放在TMR0，TMR1L/TMR1H和TMR2L/TMR2H,它们相应的控制寄存器分别为TMR0C，TMR1C和TMR2C。请注意，定时器寄存器可以预先写入固定的数据，以允许设定不同的时间间隔。

### 输入/输出寄存器

在特殊功能寄存器中，输入/输出寄存器很重要。所有的输入/输出端口都有相对应的寄存器，且被标示为 PA、PB 和 PC 等等。如数据存储器结构图中所示，这些寄存器映射到数据存储器的特定地址，用以传送端口上的输入/输出数据。在 LCD 型 MCU 系列中，端口 A 和端口 C 是输入/输出端口，其对应的寄存器 PA 和 PC 可以传送输入和输出的数据，而端口 B 只是输入端口，其对应的寄存器 PB 只能读取数据。它们的一大灵活特性是可以使用“SET [m].i”和“CLR [m].i”指令直接设置寄存器的某一位。

### UART 寄存器 – USR,UCR1,UCR2,TXR/RXR,BRG

HT49RU80/HT49CU80 具有一个内部 UART 功能，有 5 个相关的寄存器。USR 是 UART 的状态寄存器，UCR1 和 UCR2 是控制寄存器。通过串发送或接收的数据存放在 TXR/RXR 寄存器中，而波特率的配置是通过寄存器 BRG 来实现的。

## 输入/输出

盛群单片机的输入/输出端口具有很大的灵活性。尽管端口 B 固定为输入和在 HT49RU80/HT49CU80 系列端口 D 固定为输出，但端口 A 和端口 C 的所有引脚都可以设置为输入或输出。再加上其它选项，如 COMS 或 NOMS 输出型态、上拉电阻和唤醒功能等，使得此类单片机在广泛应用上都能符合开发的要求。依据所选单片机及封装类别的不同，LCD 型单片机提供从 14 到 31 个不等双向输入/输出。这些输入/输出为 PA 和 PC，而 PB 则为纯输入口，PD 为纯输出口。它们在数据存储器的对应指定地址如表所示。PA 和 PC 输入/输出口

可以作为输入及输出之用，然而，必须要注意，与其它系列单片机不同的是，在 LCD 型单片机中没有端口控制寄存器。要设定某个输入/输出引脚为输入，必须先输出高电平以取代 NMOS 晶体管的高阻抗状态，这样才能使该引脚作为输入。请注意，只有先将引脚配置为 NMOS 输出引脚，这样的动作才会有效。如果掩膜选项已经设置 CMOS 输出，则不能再作为输入使用。

作为输入操作时，输入/输出引脚不是锁存的，也就是输入数据必须在指令“MOV A, [m]”的 T2 上升沿准备好，m 表示端口地址。对于输出操作，所有数据是锁存的，而且持续到输出锁存被重写。请注意，CMOS 或 NMOS 输出类型选项只有 PA 的引脚 PA0~PA3 以及 PC 的引脚 PC0~PC3 和 PC4~PC7 才有。CMOS 或 NMOS 输出类型选项同时作用于四个引脚，引脚不能单独的选择。PB 只是输入口，没有输出功能。PB 的所有引脚都连接内部上拉电阻。没有与 PB 相关的掩膜选项。在 HT49RU80/HT49CU80 中附加的端口 PD 只能作为输出，可以通过掩膜选项将它设为 CMOS 输出或 LCD SEGMENT 输出。

### 上拉电阻

很多产品应用在端口处于输入状态时需要外加一个上拉电阻来实现上拉的功能。为了免去这个外加的电阻，当引脚规划为 NMOS 型态时，可由内部连接到一上拉电阻，这些上拉电阻可通过掩膜选项来加以选择，它用一个 PMOS 晶体管来实现。上拉选项只对某些输入/输出引脚有效。对于 LCD 系列的单片机，不能对单独的引脚进行上拉选项的选择，对于引脚 PA0~PA3、PC0~PC3 或 PC4~PC7，每次只能对四个引脚同时作用，而对于引脚 PA4~PA7 和 PB 输入口的所有引脚，则一直连接内部上拉电阻。

### PA 口的唤醒

本系列的单片机都具有暂停功能，使得单片机进入暂停模式以节省功耗，此功能对于电池及低功率应用是很重要的。唤醒单片机有很多种方法，其中之一就是使 PA 口其中的一只引脚从高电平转为低电平。当使用暂停指令“HALT”迫使单片机进入暂停状态以后，单片机将保持闲置即低功率状态，直到 PA 口上被选为唤醒输入的引脚电平发生下降沿跳变。这个功能特别适合于通过外部开关来唤醒的应用。值得注意的是 PA 口的每个引脚都可单独的选择具有唤醒的功能。

### 引脚共用功能

引脚的共用功能可以增加单片机的灵活程度。有限的引脚个数会严重的限制设计者，但是引脚的多功能特性，可以解决很多此类问题。多功能输入/输出引脚的功能选择，有些是由掩膜选项进行设定，有些则是在应用程序中进行控制。

#### → 蜂鸣器

蜂鸣器引脚 BZ 及  $\overline{\text{BZ}}$  与输入/输出引脚 PA0 及 PA1 共用。假如定义为蜂鸣器引脚，则必须选择正确的掩膜选项。

#### → PFD 输出

PFD 引脚与输入/输出引脚 PA3 共用。如果定义为 PFD 输出，则必须选择正确的掩膜选项。

#### → 外部中断输入

外部中断引脚  $\overline{\text{INT0}}$  和  $\overline{\text{INT1}}$  与输入引脚 PB0 和 PB1 共用。如果不需要外部中断输入，这些引脚可当作一般输入引脚使用，此时 INTC0 寄存器中的外部中断使能位必须除能。

#### → 外部定时器时钟输入

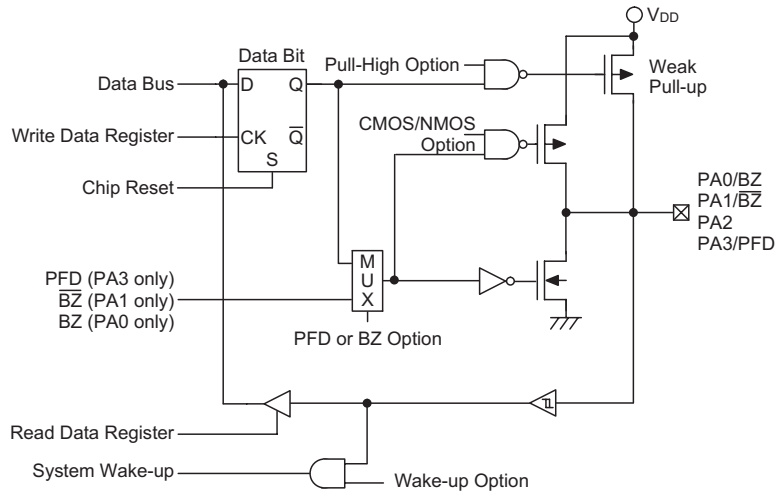
每款 LCD 系列的单片机都包含一个，两个或三个定时器，定时器的个数取决于选用的型号。每个定时器都有一个外部输入引脚，对于具有一个定时器的单片机，称为 TMR。对于具有两个定时器的单片机，它们的外部定时器输入引脚称为 TMR0 和 TMR1。HT49RU80/HT49CU80 有三个定时器，它们的外部定时器输入引脚分别为 TMR0, TMR1 和 TMR2。对所有只具有一个定时器的单片机来说，外部输入引脚 TMR 与输入引脚 PB2 共用。而对于具有两个定时器的单片机，外部输入引脚 TMR0 和 TMR1 分别与输入引脚 PB2 和 PB3 共用。HT49RU80/HT49CU80 的三个外部输入引脚分别与 PB2, PB3 和 PB4 共用。要使这些共用引脚作为定时器输入，必须正确地设置定时器控制寄存器中对应的控制位。在不需要外部定时器输入的场所，这些外部定时器引脚可以当作一般输入引脚使用。对于此种应用，TMRC 寄存器中的定时器模式位必须选为定时器模式(内部时钟源)，以避免输入引脚与定时器操作的冲突。

#### → UART 引脚

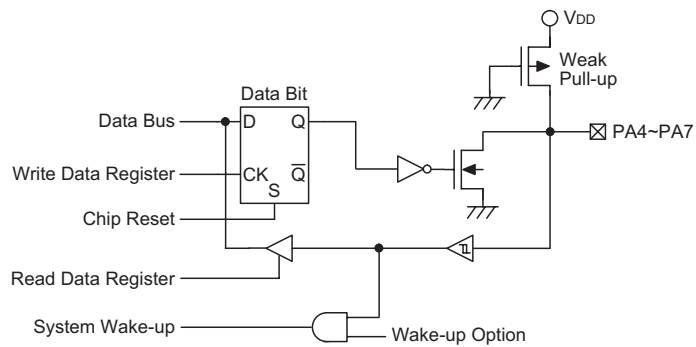
HT49RU80/HT49CU80 具有一个内部的异步串行通信端口，它通过两个引脚与外部芯片相连，这两个外部引脚 TX 和 RX 分别与引脚 PC0 和 PC1 共享。

→ **COM/SEG 输出**

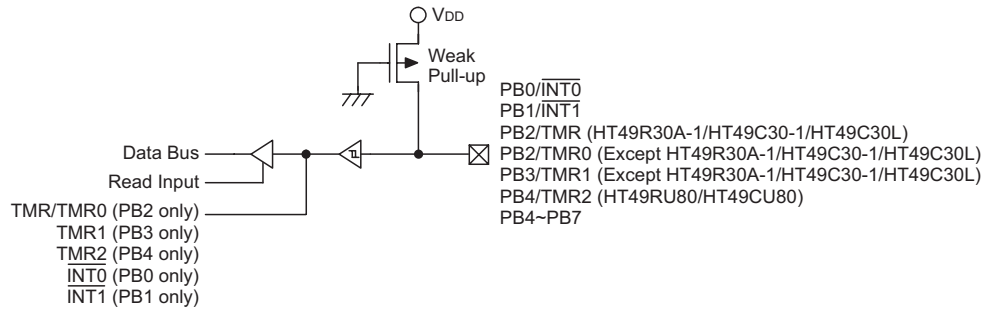
COM 和 SEG 引脚用来直接驱动 LCD 上 segment 和 common 引脚。每款单片机均有一个引脚即可用作 segment 驱动也可用作 common 驱动，根据选用的型号，这些引脚称为 COM3/SEG18、COM3/SEG32、COM3/SEG40 和 COM3/SEG47。由占空比掩膜选项决定这些引脚是作为 common 还是 segment 功能使用。如果选用 1/4 占空比掩膜选项，则该引脚作为 COM3 驱动，如果选用 1/2 或 1/3 占空比掩膜选项，则选择作为 SEG 功能使用。在 HT49RU80/HT49CU80 中，有一个掩膜选项可以把 PD 口设为 SEG 输出或标准的 CMOS 输出。



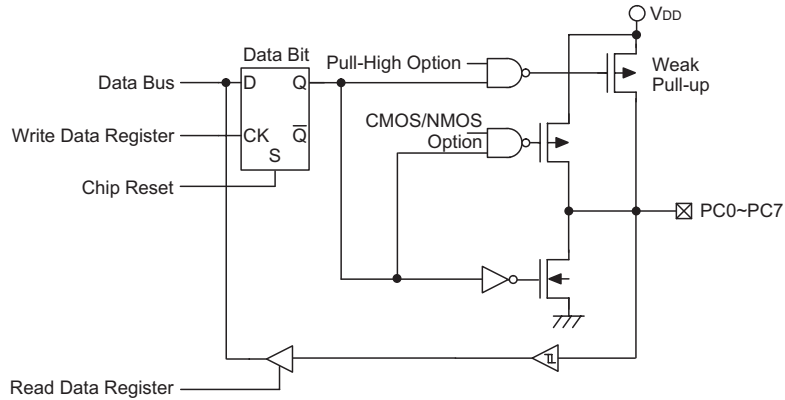
**PA0-PA3 输入/输出口**



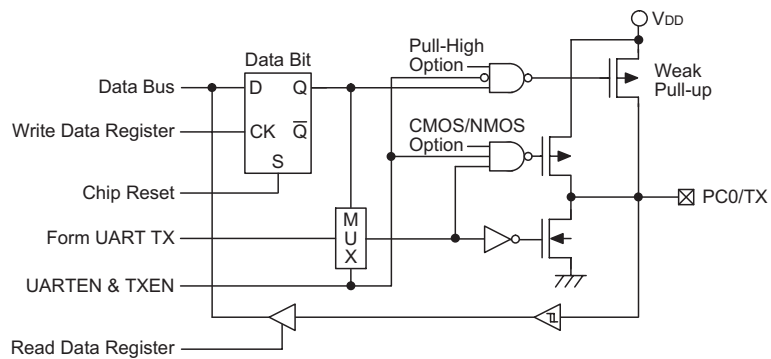
**PA4-PA7 输入/输出口**



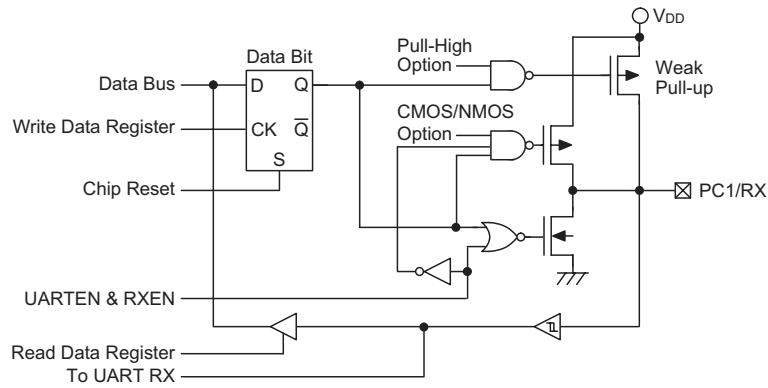
**PB 输入口**



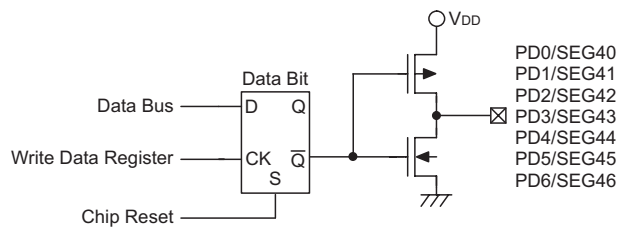
**PC 输入/输出口**



**PC0/TX 输入/输出口—HT49RU80/HT49CU80**



PC1/RX 输入/输出口—HT49RU80/HT49CU80



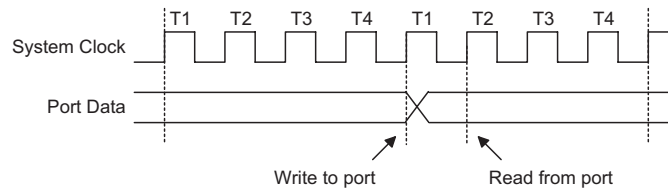
PD 输出口—HT49RU80/HT49CU80

### 编程注意事项

在应用程序中，最先要考虑的是端口的初始化。复位之后，输入/输出口寄存器 PA 和 PC 将被设定为逻辑高。请注意，如果掩膜选项选择为 NMOS 型态，则 NMOS 晶体管将呈现高阻抗状况，而不是输出逻辑高，以允许引脚作为输入使用。NMOS 的输出是否能形成逻辑高，要取决于它的外部连接电路以及上拉选项是否被选择。如果掩膜选项是选择 CMOS 输出，则不会形成高阻抗状况，因此不能作为输入使用。

当引脚作为输出使用时，可以通过载入一个字节宽度的数据到相应的端口寄存器 PA 或 PC，或者是使用指令“SET [m].i”和“CLR [m].i”来设定端口寄存器中个别的位。要注意的是当使用这些位控制指令时，一个读-修改-写的操作将会发生。单片机必须先读入整个口上的数据，修改个别位的值，然后重新把这些数据写入到输出端口。在 NMOS 引脚中，要注意一些特殊的状况。如果单片机将 NMOS 引脚设定为逻辑高，也就是高阻抗状况，但又被外部连接的电路拉低，则在“SET [m].i”和“CLR [m].i”指令的读取阶段会被读为逻辑

低。在确定写入阶段，这个在读取阶段读作逻辑低的引脚，会因此而错误的设为逻辑低。因此在使用 NMOS 输出型态的位控制指令时，要特别的谨慎。



PA 口有唤醒的额外功能，当芯片在 HALT 状态时有很多方法去唤醒此单片机，其中之一就是 PA 口任一引脚电平由高到低的转换，PA 口的一个或多个引脚都可被设定有这项功能。

## 液晶显示(LCD)驱动器

对大多数包含 LCD 的设计应用而言，与较昂贵的以字符为基础的显示方式相比，用户自定义的显示方式能有效地降低成本。但因为需要有可变增益和时间的对应信号来驱动此类自定义显示器，因此为了适当的操作此类 LCD 需要有特殊的考虑。盛群公司的 LCD 系列单片机有内部 LCD 信号产生电路及多种掩膜选项，可以自动地产生时间与增益可变的信号直接驱动 LCD，与用户 LCD 的接口连接也相当容易。

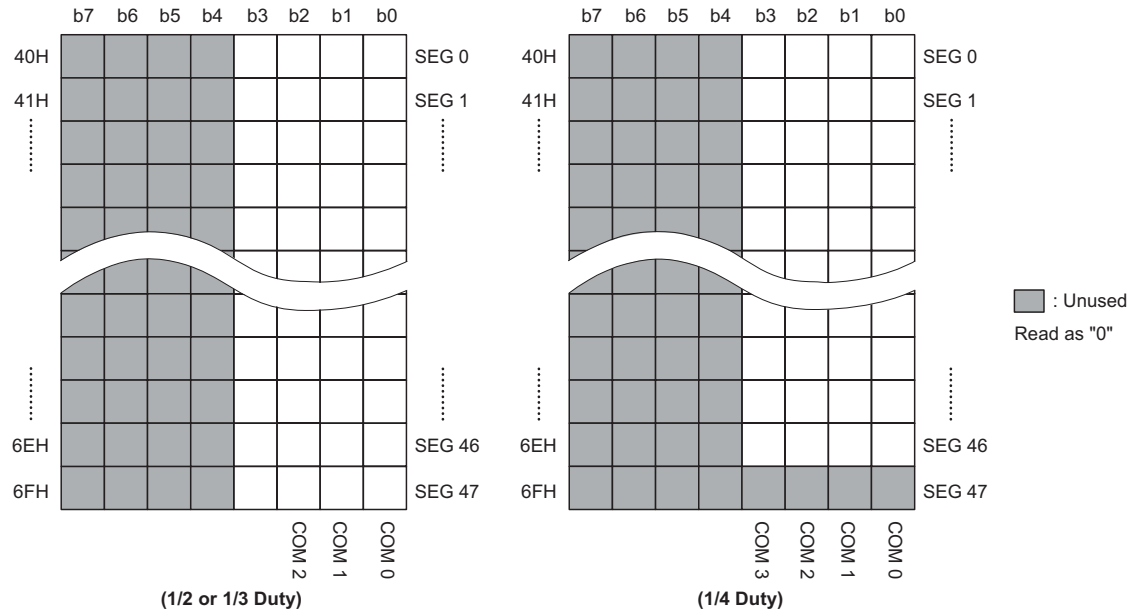
### LCD 存储器

每款 LCD 系列单片机，都为 LCD 数据提供特殊的数据存储器。这样的数据区域就是 LCD 存储器。任何写入此处的数据，会自动地被内部 LCD 驱动电路读取，进而自动地产生所需的 LCD 驱动信号。因此任何写入 LCD 存储器的数据，会立即映射到连接单片机的 LCD 显示器上。每款 LCD 系列单片机的 LCD 存储器的起始地址为 40H。根据所选单片机 LCD 存储器容量的不同，LCD 存储器的结束地址在 52H 至 6FH 之间。

由于 LCD 数据存储器地址与通用数据存储器地址重迭，因此 LCD 数据存储器储存在自己的数据存储区块。除了 HT49RU80/HT49CU80，其余单片机的通用数据存储器均储存在存储区 0。而 HT49RU80/HT49CU80 有三个通用数据存储器 Bank0、Bank2 和 Bank3。对于所有的单片机，LCD 数据存储器位于存储区 1。存储区的选择是通过使用存储区指针来完成，它是数据存储器中的一个特



殊功能寄存器，名为 BP。对于 HT49RU80/HT49CU80，当 BP 的最低两位为二进制的“00”、“10”或“11”时，通用数据存储区才能被读写，LCD 存储区不会有读写动作发生。对 LCD 存储区操作时，首先要将存储区指针 BP 的低两位设为二进制的“01H”，以选中存储区 1。此后，可以通过使用间接寻址指针 MP1 间接寻址来存取 LCD 存储器。存储区 1 被选中后，可以使用 MP1 读取或写入数据到范围为 40H~52H、40H~60H、40H~68H 或 40H~6FH (根据选取的单片机)的存储器。直接寻址在 LCD 存储器中是不适用的，结果会是存取存储区 0 的数据。



LCD 存储器映射图 - HT49RU80/HT49CU80

以上为 HT49RU80/HT49CU80 LCD 存储器的结构，它具有 48×2、48×3 或 47×4 格式的像素驱动能力，LCD 存储器结束地址为 6EH 或 6FH。HT49R70A-1/HT49C70-1/HT49C70L 具有 41×2，41×3 或 40×4 格式的像素驱动能力，LCD 存储器结束地址为 68H 或 67H。HT49R50A-1/HT49C50-1/HT49C50L 具有 33×2，33×3 或 32×4 格式的像素驱动能力，LCD 存储器结束地址为 60H 或 5FH。HT49R30A-1/ HT49C30-1/HT49C30L 具有 19×2，19×3 或 18×4 格式的像素驱动能力，LCD 存储器结束地址为 52H 或 51H。对所有 LCD 型单片机来说，当选择 1/4 占空比掩膜选项时，会自动地设定成 4-COM 格式，当选择

1/2 或 1/3 占空比掩膜选项时，会自动地设定成 2-COM 或 3-COM 格式。

**注意：** 对所有 LCD 型单片机来说，若选择 1/2 或 1/3 占空比掩膜选项，则只提供 3-COM 连接，且每个 LCD 存储器地址的第 3 位可以做为通用数据存储器使用。而选择 1/4 占空比掩膜选项时，则提供 4-COM 连接，但是会少一个 segment 连接提供，LCD 存储器的最后地址不使用且使用者无法存取，如果读取的话，将会返回“00”。

## LCD 时钟

LCD 时钟由内部时钟源  $f_S$  驱动，它由 WDT 振荡器、RTC 振荡器或  $f_{SYS}/4$  提供，由掩膜选项决定。为了适于 LCD 操作，这个  $f_S$  内部时钟源通过一个分频器，提供接近 4kHz 的 LCD 时钟源频率。

$f_S$ 时钟源	LCD 时钟选择
WDT 振荡器	$WDT/2^2$
RTC 振荡器	$RTC/2^3$
$f_{SYS}/4$	$\frac{f_{SYS}/4}{2^2} \sim \frac{f_{SYS}/4}{2^8}$

### LCD 时钟频率选择

分频比例的选择根据内部时钟源  $f_S$  而定。若时钟源  $f_S$  来自 WDT 振荡器，则分频比例是固定的  $f_S/2^2$ ，若时钟源  $f_S$  来自 RTC 振荡器，分频比例则是固定的  $f_S/2^3$ ，若时钟源  $f_S$  来自  $f_S/4$ ，则 LCD 时钟频率范围为  $f_S/2^2$  到  $f_S/2^8$ ，由掩膜选项进一步确定。这些分频比例可以保证 LCD 工作在接近 4kHz 的频率。对于具有 4kHz 的 LCD 时钟频率，单片机的 LCD 驱动电路将会产生 55Hz 到 62Hz 的 LCD 帧频率。这与 25Hz 到 250Hz 之间的 LCD 操作频率范围是一致的。要注意的是，若选择的 LCD 时钟频率太高，会产生高于所需的帧频率，引起较高的功耗，而选择过低的频率又会导致不稳定。因此如果选择  $f_{SYS}/4$  作为时钟源，正确的选择掩膜选项，获得尽可能接近 4kHz 的 LCD 时频率是非常重要的。

## LCD 驱动器输出

LCD 驱动器提供的 COM 和 SEG 输出数目，以及偏压和占空比选项，取决于选用的单片机和配置选项的选择。下表列出每一款 LCD 型单片机的不同选项。

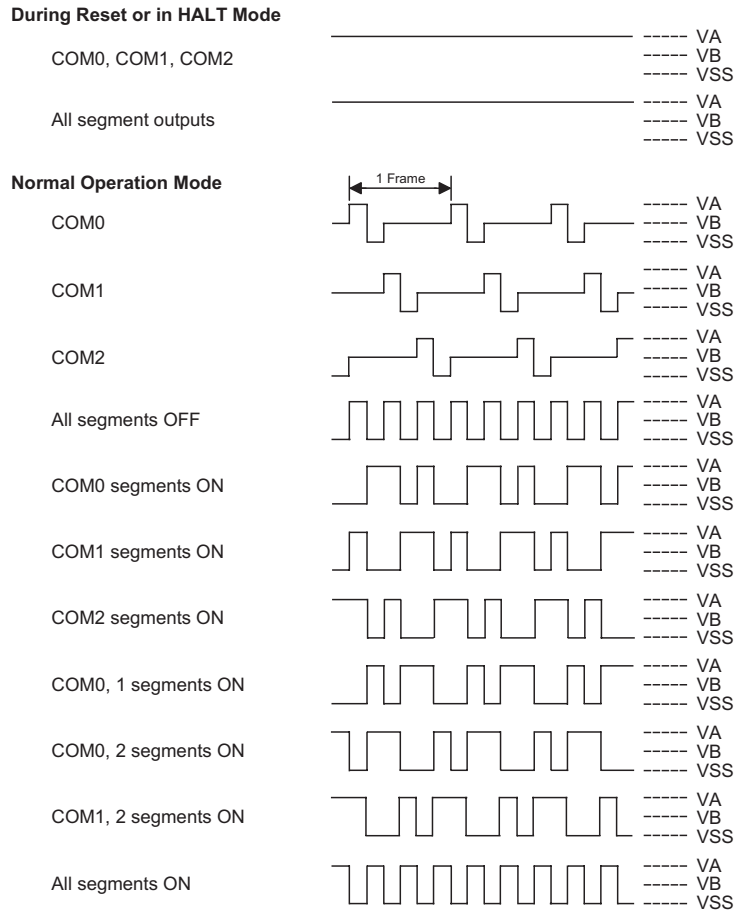
单片机型号	占空比	驱动数目	偏压	偏压类型
HT49R30A-1	1/2	19×2	1/2 或 1/3	C 或 R 类型
HT49C30-1	1/3	19×3		
	1/4	18×4		

HT49R50A-1 HT49C50-1	1/2	33×2	1/2 或 1/3	C 或 R 类型
	1/3	33×3		
	1/4	32×4		
HT49R70A-1 HT49C70-1	1/2	41×2	1/2 或 1/3	C 或 R 类型
	1/3	41×3		
	1/4	40×4		
HT49RU80 HT49CU80	1/2	48×2	1/2 或 1/3	C 或 R 类型
	1/3	48×3		
	1/4	47×4		

LCD 驱动输出，占空比与偏压选项

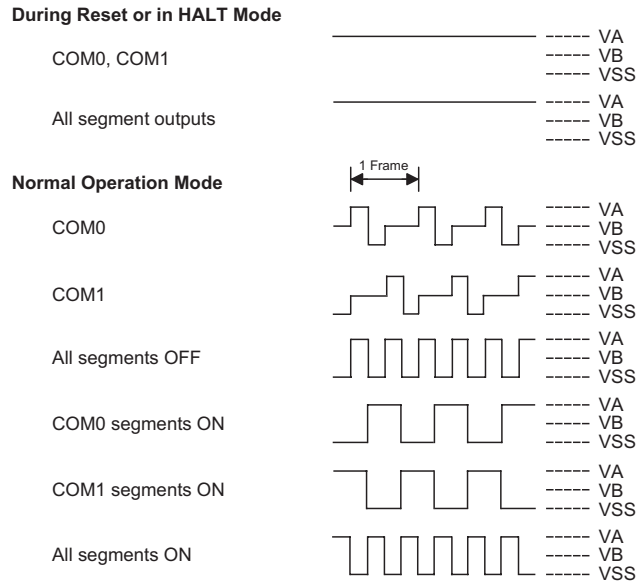
**注意：**低电压版的 HT49C30L、HT49C50L、HT49C70L 不同于其它单片机，具有固定的 1/2 偏压和 C 型偏压，但与其它 OTP 版具有相同的占空比和驱动数目选项。

由于 LCD 基本性质的缘故，它们的像素点只能加上 AC 电压，如果加上 DC 电压，将会引起永久性的损害。因此 LCD 显示器的对比度由提供到每个像素的实际 RMS 电压控制，这个值相当于 COM 引脚上的 RMS 电压值减去 SEG 引脚上的电压值。RMS 电压必须大于 LCD 的饱和电压，以便能打开像素点，但同时也要小于阈值电压，以便能关闭像素点。此要求限制 DC 电压为零并且以最少数目的连接去控制尽可能多的像素，为此需要产生时间与增益都可变的信号，以提供 LCD 应用。这些时间与增益都可变的信号由单片机内的 LCD 驱动电路自动产生。占空比决定使用 common 口的个数，也称为底板或 COMs。由掩膜选项选择的占空比有 1/2、1/3 或 1/4 的值，它代表 COM 的数目为 2、3 或 4，定义了每个 LCD 信号帧内的时间分割数。下面的时序图描述单片机所产生的对于各种占空比和偏压值的 LCD 信号。



**LCD 驱动输出(1/3 占空比, 1/2 偏压)**

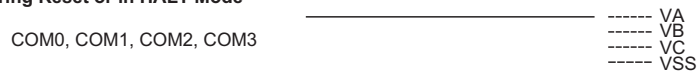
- 
- 注意:** 1. 对 HT49R30A-1/HT49C30-1、HT49R50A-1/HT49C50-1、HT49R70A-1/HT49C70-1 和 HT49RU80/ HT49CU80 而言, R 与 C 型偏压的  $VA=VLCD$ 、 $VB=VLCD \times 1/2$ 。
2. 对 HT49C30L、HT49C50L 和 HT49C70L 而言,  $VA=2V2$ 、 $VB=V2$ 、只有 C 型偏压可用。
-



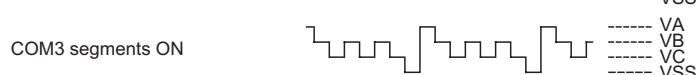
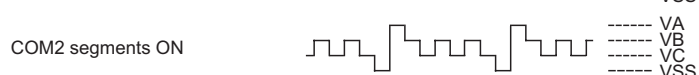
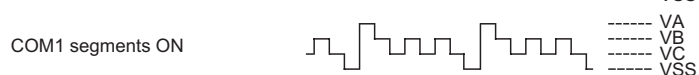
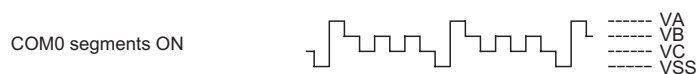
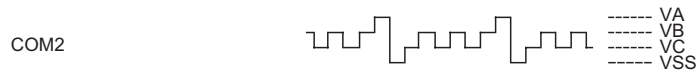
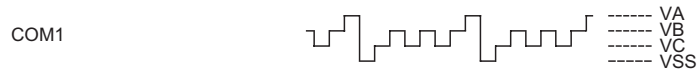
LCD 驱动输出(1/2 占空比, 1/2 偏压)

- 
- 注意: 1. 对 HT49R30A-1/HT49C30-1、HT49R50A-1/HT49C50-1、HT49R70A-1/HT49C70-1 和 HT49RU80/ HT49CU80 而言, R 与 C 型偏压的  $VA=VLCD$ 、 $VB=VLCD \times 1/2$ 。
2. 对 HT49C30L、HT49C50L 和 HT49C70L 而言,  $VA=2V2$ 、 $VB=V2$ 、只有 C 型偏压可用。
-

**During Reset or in HALT Mode**

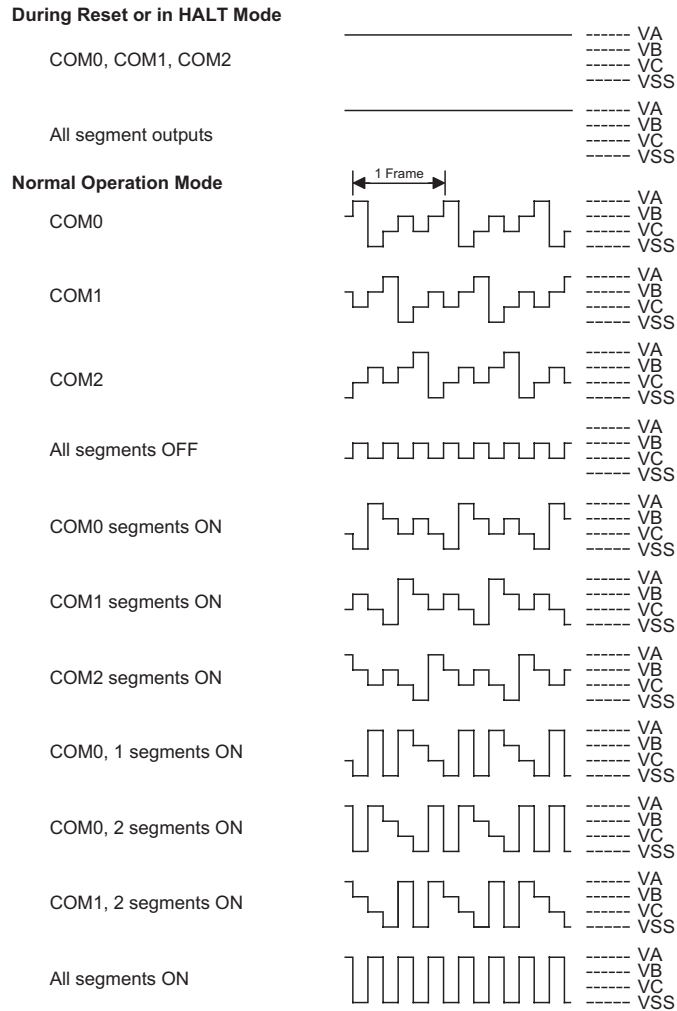


**Normal Operation Mode**



LCD 驱动输出(1/4 占空比, 1/3 偏压)

- 注意： 1. 对 HT49R30A-1/HT49C30-1、HT49R50A-1/HT49C50-1、HT49R70A-1/HT49C70-1、HT49RU80/HT49CU80 而言，R 型偏压的  $VA=VLCD$ 、 $VB=VLCD \times 2/3$ 、 $VC=VLCD \times 1/3$ 。C 型偏压的  $VA=VLCD \times 1.5$ 、 $VB=VLCD$ 、 $VC=VLCD \times 1/2$ 。
2. 对 HT49C30L、HT49C50L 和 HT49C70L 而言，不提供 1/3 偏压。



LCD 驱动输出(1/3 占空比, 1/3 偏压)

- 
- 注意: 1. 对 HT49R30A-1/HT49C30-1, HT49R50A-1/HT49C50-1、HT49R70A-1/HT49C70-1 和 HT49RU80/ HT49CU80 而言,R 型偏压的  $VA=VLCD$ 、 $VB=VLCD \times 2/3$ 、 $VC=VLCD \times 1/3$ 。C 型偏压的  $VA=VLCD \times 1.5$ 、 $VB=VLCD$ 、 $VC=VLCD \times 1/2$ 。
2. 对 HT49C30L、HT49C50L 和 HT49C70L 而言, 不提供 1/3 偏压。
- 

### LCD 电压源与偏压

盛群 LCD 类型的单片机能产生时间与增益都可变的信号, 此信号在操作时需要几种电压值。信号使用的电压值数目取决于选用的单片机型号和偏压掩膜选项。

### LCD 偏压

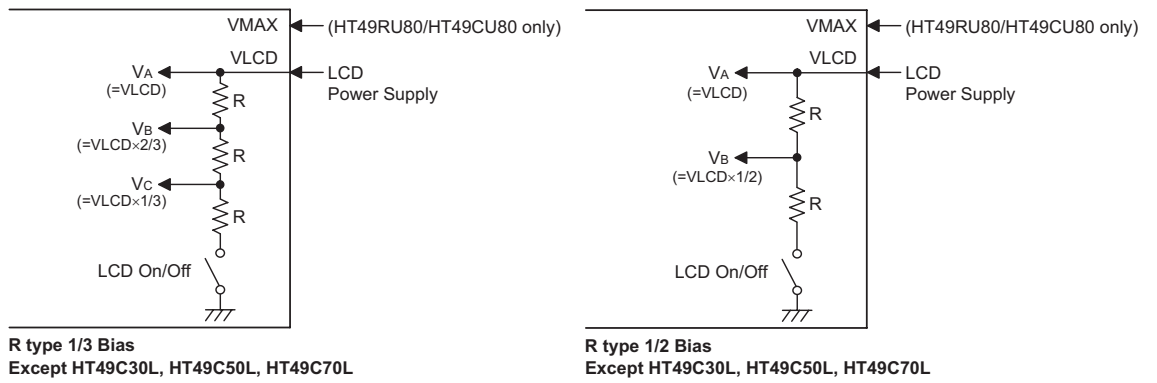
HT49C30L、HT49C50L 和 HT49C70L 有固定的偏压值 1/2, 而其它单片机则是通过掩膜选项选择 1/2 偏压或 1/3 偏压。对于 1/2 偏压的掩膜选项, 要用到三个电压值, 分别是 VSS、VA 和 VB。除了 HT49C30L、HT49C50L 和 HT49C70L, 在 R 型偏压中 VA 等于 VLCD, VLCD 是外部提供的电压源。这可以是单片机的电压源或是其它的电压源。VB 由单片机内部产生且电压值等于 VLCD/2。对于 1/3 偏压的掩膜选项, 要用到四个电压值, 分别是 VSS、VA、VB 和 VC。引脚 VLCD 上的外部 LCD 电压源用来产生这些电压。VA、VB 和 VC 的实际电压值由 R 或 C 偏压类型选项决定。要注意的是, C 型偏压会使用充电泵电路, 所以会产生比外部提供的 VLCD 还要高的电压。在单片机提供的电压小于 LCD 所需电压时, 这项特性相当有用。

对于 R 型 1/3 偏压选项, 电压 VA 和外部提供的 VLCD 电压相同, 电压 VB 和 VC 由内部电阻分压决定。对于 R 型 1/2 偏压选项, VC 是不需要的, 也不需要外部电阻, 然而仍建议要连接外部电容到引脚 V1 和 V2, 以稳定这些内部产生的电压。在 C 型偏压中, 为了要产生所需的电压值, 必须要在引脚 C1 与 C2 之间连接充电泵电容, 在引脚 V1 与 V2 之间连接的滤波电容。在 HT49C30L、HT49C50L 和 HT49C70L 中, 引脚 V2 作为 LCD 电压源的输入, 可以省略这个引脚上的电容, 而在 VLCD 引脚上要额外接一个电容。

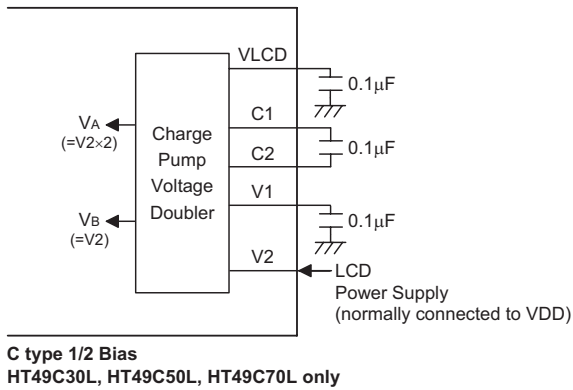
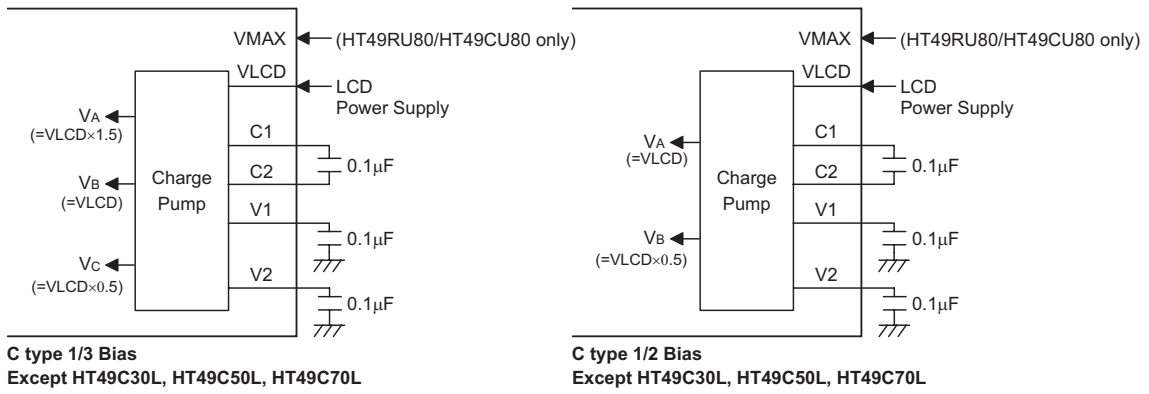
对于 HT49C30L、HT49C50L 和 HT49C70L, 由于经常使用低电压电源或单一电池电源, 因此 LCD 所需的较高的供应电压是由内部充电泵倍压电路产生。对这些单片机, 外部电压供应连接到引脚 V2, 通常情况下这就是单片机的供应电源, 但也可由其它电压源提供。供应到引脚 V2 的电压会通过内部充电泵



倍压电路，使电压值加倍以建立  $V_A$  值。供应引脚 V2 的外部电压的实际值则是根据 LCD 应用的需求而定。



**R 型偏压电压值**



### C 型偏压电压值

- 注意:**
1. 对于 HT49C30L、HT49C50L 和 HT49C70L 而言，R 型和 1/3 偏压掩膜选项都是无效的。
  2. 只有 HT49RU80/HT49CU80 具有 VMAX 脚

### VMAX 脚连接—只有 HT49RU80/HT49CU80 具有

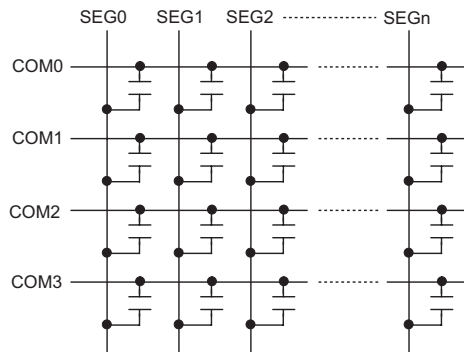
在 HT49RU80/HT49CU80 中有另一个引脚—VMAX，它的连接方式取决于加在 VLCD 上的电压。下面表格提供了 VMAX 引脚的正确连接方式：

偏压类型		VLCD 电压	VMAX 脚连接
C Type	1/3Bias	$VDD > VLCD \times 1.5$	Connect VMAX to VDD
		Otherwise	Connect VMAX to V1
	1/2Bias	$VDD > VLCD$	Connect VMAX to VDD
		Otherwise	Connect VMAX to VLCD
R Type	1/2or1/3Bias	$VDD > VLCD$	Connect VMAX to VDD
		Otherwise	Connect VMAX to VLCD

**编程注意事项**

规划 LCD 时要采取一些预防措施，其中之一就是在单片机上电后，要保证 LCD 存储器正确地初始化。与通用数据存储器一样，在上电后，LCD 存储器的内容是未知的。由于 LCD 存储器的内容会映射到实际的 LCD，所以在上电后，为获得正确的显示图形，初始化此存储器内容是非常重要的。

在实际应用中，必须要考虑 LCD 的实际电容性负载。通过 LCD 像素出现在单片机中的负载，通常可以模式化成为电容性，一定要确保它不能过多，对可以连接多个 LCD 像素点的 COM 的口来说，这一点尤其如此。接下来的流程图描述 LCD 的等效电路。



**LCD 面板等效电路**

在实际应用中，设置正确的 LCD 时钟频率是另外一个需要特别注意的地方。为了使 LCD 工作在最佳帧频率，此频率通常在 25Hz 到 250Hz 之间，选择一个合适的 LCD 时钟频率的掩膜选项是非常重要的。正确的选择必须使得 LCD 时钟频率尽可能地接近 4kHz。选择这样的频率，单片机内部的 LCD 驱动电路将会产生适当的 LCD 驱动信号，以获得合适的 LCD 帧频率。

另一个必须考虑的是单片机进入 HALT 状态时，究竟会发生什么？在 HALT 模式时，为了减少功耗，掩膜选项会允许 LCD 关闭电源。若选择此选项，则在执行 HALT 指令后，LCD 驱动信号会停止，从而产生一个空白显示的图形，减少任何与 LCD 有关的功耗。由于 HALT 指令的执行并不影响 LCD 存储器，当单片机被唤醒、LCD 驱动信号重新开始后，将会恢复原来显示的图形。在 HALT 模式时，如果掩膜选项选择 LCD 显示器保持打开状态，则 LCD 驱动信号会持续产生，LCD 图形保持不受干扰，然而应该注意的是，这样的话会导致功耗。

## 定时/计数器

定时/计数器在任何单片机中都是一个很重要的部分，提供程序设计者一种实现和时间有关功能的方法。在 LCD 型单片机中，通常包含一个、两个或三个 8 或 16 位的向上计数器，这取决于选用哪款单片机。第个定时/计数器有三种不同的工作模式，可以被当作一个普通定时器、外部的事件计数器、或者脉冲宽度测量器使用。

有两个和定时/计数器相关的寄存器。其中，一个寄存器是存储实际的计数值，赋值给此寄存器可以设定初始值，读取此寄存器可获得定时/计数器的内容。另一个寄存器是定时/计数器的控制寄存器，此寄存器设置定时/计数器的选项，控制定时/计数器的使用。定时/计数器的时钟源可来自内部时钟源或在外部定时器引脚，下列附表列举了对应定时/计数器寄存器的名称。

	HT49R30A-1 HT49C30-1 HT49C30L	HT49R50A-1 HT49C50-1 HT49C50L	HT49R70A-1 HT49C70-1 HT49C70L	HT49RU80 HT49CU80
<b>8 位定时/计数器编号</b>	1	2	1	1
定时/计数器寄存器名称	TMR	TMR0/TMR1	TMR0	TMR0
定时控制寄存器名称	TMRC	TMR0C/TMR1C	TMROC	TMROC
<b>16 位定时/计数器编号</b>	—	—	1	2
定时/计数器寄存器名称	—	—	TMR1L/TMR1H	TMR1L/TMR1H TMR2L/TMR2H
定时控制寄存器名称	—	—	TMR1C	TMR1C TMR2C

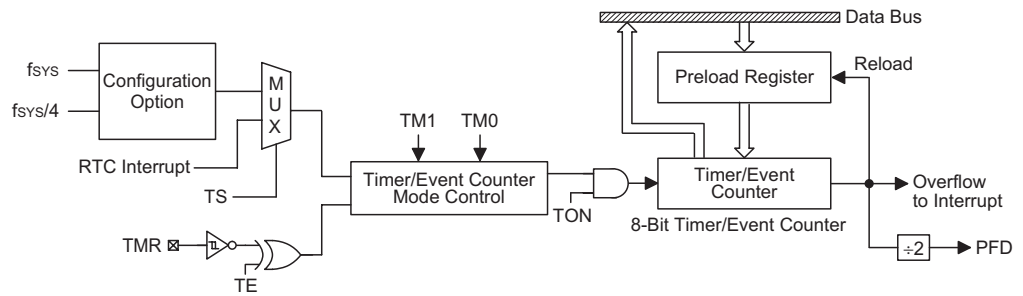
定时/计数器在事件计数模式下使用外部时钟源，而时钟源从外部计数器的引脚输入，即 TMR、TMR0、TMR1 或 TMR2，这取决于选用哪种型号的单片机。这些外部引脚可能与 PB 输入引脚共用。每当外部定时/计数器输入引脚由高电平到低电平或者由低电平到高电平（由 TE，T0E，T1E 或 T2E 位决定）进行转换时，将使得计数器值增加一。

### 配置定时/计数器输入时钟源

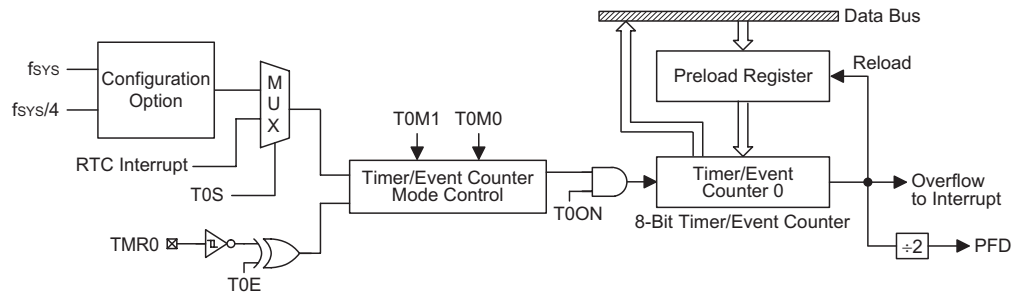
内部定时/计数器的时钟源可以来自不同的时钟源，它取决于所用的芯片和使用的定时/计数器。当定时/计数器在定时器模式或者在脉冲宽度测量模式时，使用内部时钟作为时钟源。内部时钟源可以来自系统时钟、系统时钟/4、RTC 时钟、时基或 TMR0 溢出中的任何一种，这取决于定时器的选取(由掩膜选项

选择)以及应用程序对计数器的配置。

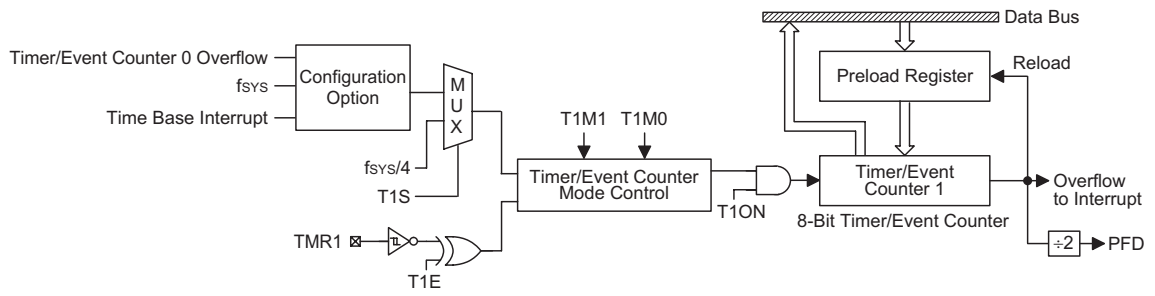
定时/计数器在事件计数模式时使用外部计时源，而时钟源是由外部定时/计数器引脚 TMR、TMR0、TMR1 或 TMR2 来提供，这取决于那种型号的单片及哪一个定时/计数器被使用。每当外部引脚由高电平到低电平或者由低电平到高电平（由 TE，T0E，T1E 或 T2E 位决定）进行转换时，将使得计数器值增加一。



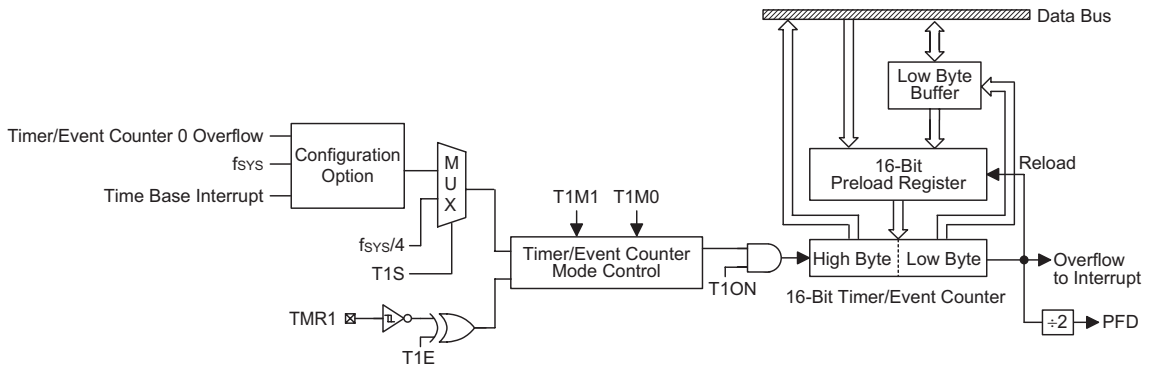
8 位定时/计数器结构- HT49R30A-1/HT49C30-1/HT49C30L



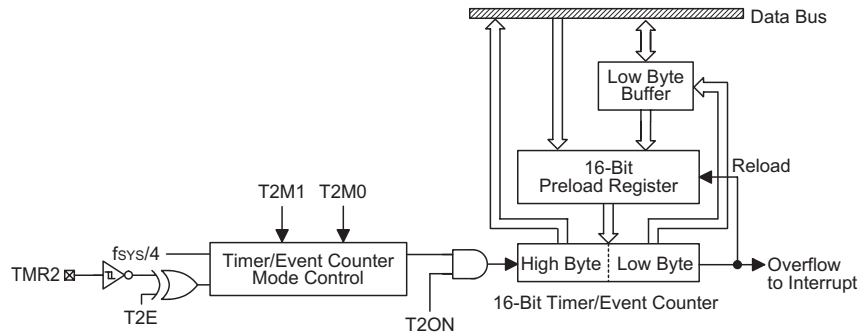
8 位定时/计数器 0 结构- HT49R50A-1/HT49C50-1/HT49C50L,  
HT49R70A-1/HT49C70-1/HT49C70L  
HT49RU80/HT49CU80



8 位定时/计数器 1 结构- HT49R50A-1/HT49C50-1/HT49C50L



16 位定时/计数器 1 结构- HT49R70A-1/HT49C70-1/HT49C70L  
HT49RU80/HT49CU80



16 位定时/计数器 2 结构- HT49RU80/HT49CU80

### 定时/计数寄存器 – TMR, TMR0, TMR1, TMR1L/TMR1H, TMR2L/TMR2H

定时/计数器寄存器是位于专用数据存储器的特殊功能寄存器，储存实际的定时器值。对 8 位定时/计数器来说，这个寄存器是 HT49R30A-1/HT49C30-1/HT49C30L 中的 TMR，HT49R70A-1/HT49C70-1/HT49C70L 和 HT49RU80/HT49CU80 中的 TMR0，HT49R50A-1/HT49C50-1/HT49C50L 中的 TMR1。对于 16 位定时/计数器来说，需要用一对 8 位寄存器来储存 16 位定时/计数器的值。这些成对的寄存器即为 HT49R70A-1/HT49C70-1/HT49C70L 中 TMR1L/TMR1H 和 HT49RU80/HT49CU80 中的 TMR1L/TMR1H、TMR2L/TMR2H。当用作内部定时器模式时收到一个内部计时脉冲时，或用作外部计数模式时外部定时/计数器引脚发生电平转换时，定时/计数寄存器的值将会加一。定时器从预置寄存器所载入的值开始向上计数，直到 8 位定时/计数器计到 FFH 或 16 位定时/计数器计到 FFFFH，此时定时器发生溢出且会产生一个内部中断信号。定时器的值随后被预置寄存器的值重设，定时/计数器继续计数。

注意当预置寄存器被清为零，就可以得到 8 位定时/计数器 FFH 或 16 位定时/计数器 FFFFH 的最大计算范围。此时要注意的是，上电后预置寄存器中的数值处于未知状态。定时/计数器在 OFF 条件下，如果把数据写入预置寄存器，这数据将被立即写入实际的定时器。然而如果定时/计数器已经被打开且正在计数，在这个周期内写入到预置寄存器的任何新数据将被保留在预置寄存器中，等到下一个溢出发生时才会被写入实际的定时器。当定时/计数寄存器被读取时，定时器计时时钟会停止以避免错误，然而这可能造成某些时序的错误，因此程序设计者必须考虑到这点。

对于 16 位定时/计数器，它有低字节与高字节两个定时/计数寄存器，访问这些寄存器需要以指定方式进行。必须要注意的是当使用指令载入数据到低字节寄存器，即 TMR1L 或 TMR2L 时，数据只被载入到低字节缓冲器而不是直接送到低字节寄存器。当数据写入相应高字节寄存器，即 TMR1H 或 TMR2H 时，低字节缓冲器中的数据才真正被写入低字节寄存器。换句话说，写入数据到高字节定时/计数寄存器时，数据会被直接写入到高字节寄存器。同时在低字节缓冲器里的数据将被写入相应低字节寄存器。所以当载入数据到 16 位定时/计数寄存器时，低字节数据应该先写入。另外要注意的是读取低字节寄存器的内容时，必须先读取高字节寄存器的内容，相应低字节寄存器中的内容就会载

入低字节缓冲器中并被锁存。在此动作执行之后，低字节寄存器中的内容可使用一般的方式读取。请注意，读取定时/计数器低字节寄存器实际是读取先前锁存在低字节缓冲器中的内容，而非定时/计数器低字节寄存器的实际内容。

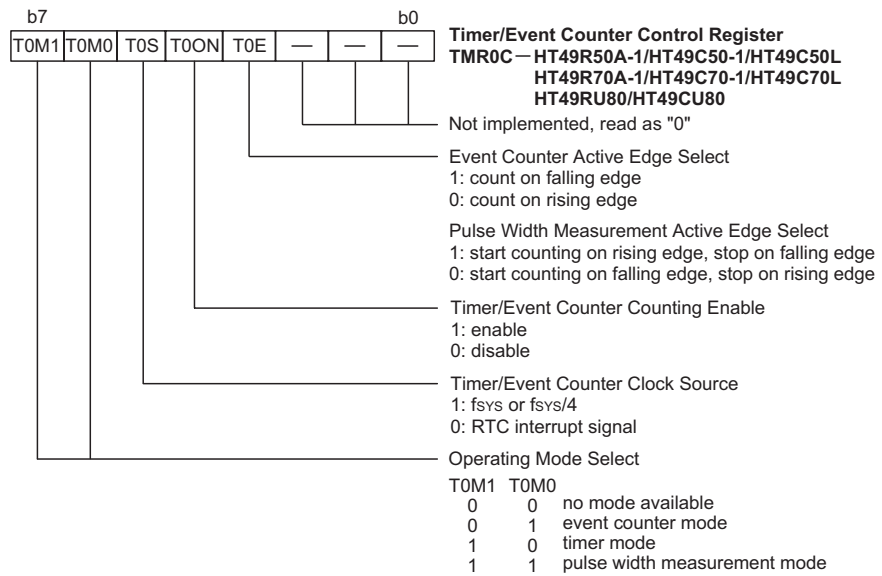
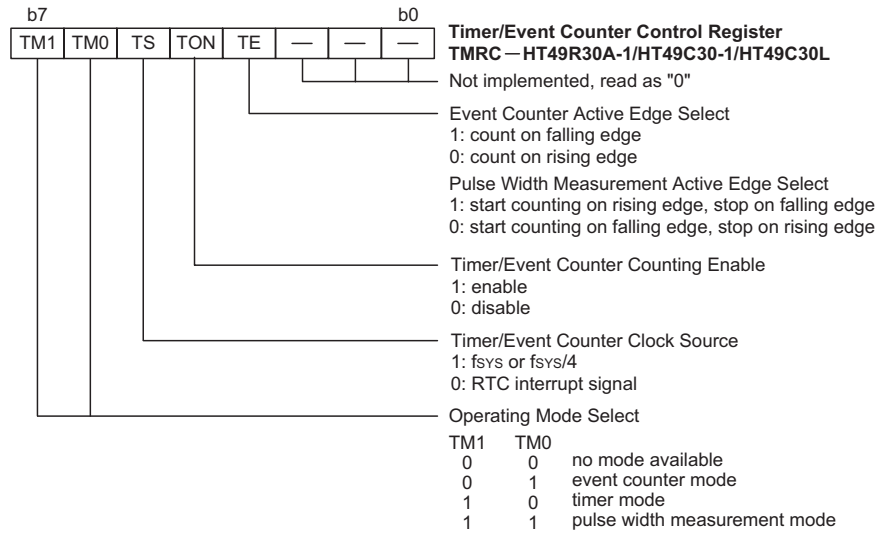
### 定时/计数控制寄存器 - TMRC, TMR0C, TMR1C, TMR2C

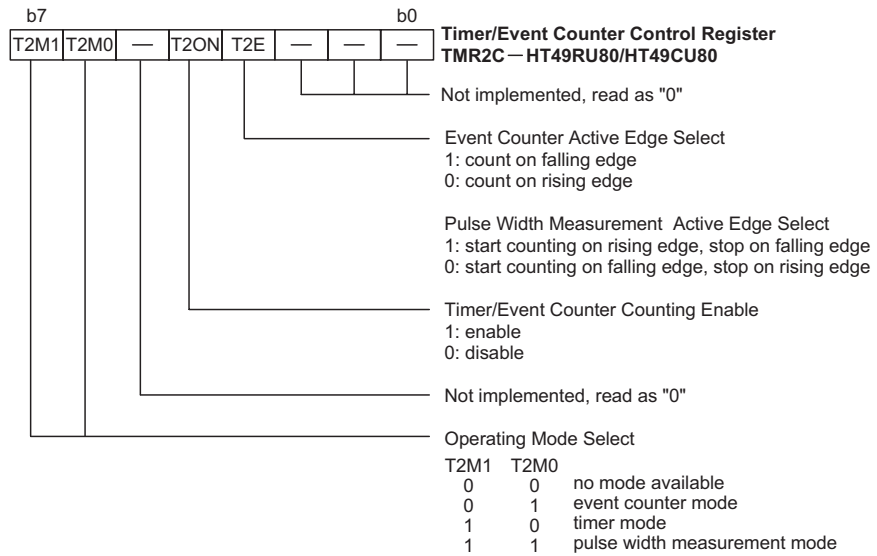
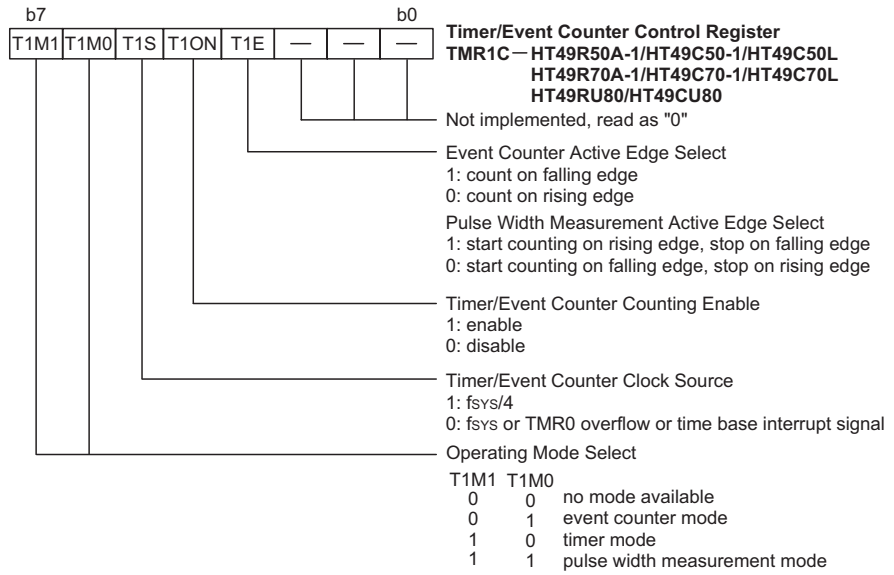
定时/计数器能工作在三种不同的模式，至于选择工作在哪一种模式则是由各自的控制寄存器的内容决定。对于只有一个定时/计数器的单片机而言，定时/计数控制寄存器即 TMRC，而对于有两个定时/计数器的单片机而言，定时/计数控制寄存器为 TMR0C 和 TMR1C。对于有三个定时/计数器的单片机而言，具有三个定时/计数控制寄存器，它们分别为 TMR0C, TMR1C 和 TMR2C。定时/计数器寄存器和定时/计数控制寄存器控制计时/事件计数器的全部操作。在定时器使用之前必须先正确地设定定时/计数控制寄存器，以便保证定时器能正确操作，而这个过程通常在程序初始化期间完成。

为了确定定时器工作在哪一种模式，位 TM1/TM0、T0M1/T0M0, T1M1/T1M0 或 T2M1/T2M0 必须设定到要求的逻辑电平。定时器打开位 TON、T0ON, T1ON 或 T2ON，即定时/计数控制寄存器的第 4 位，是定时器控制的开关，设定逻辑高时定时器开始计数，而清零时则定时器停止计数。如果定时器工作在事件计数或脉冲宽度测量模式，TE、T0E, T1E 或 T2E 的逻辑电平即 TMRC 寄存器的第 3 位将可用来选择上升或下降沿触发。

除了 HT49RU80/ HT49CU80 的定时/计数器 2，另一个时钟源位即 TS、T0S 或 T1S 依据所使用的定时/计数器来决定定时/计数器使用何种内部时钟源。

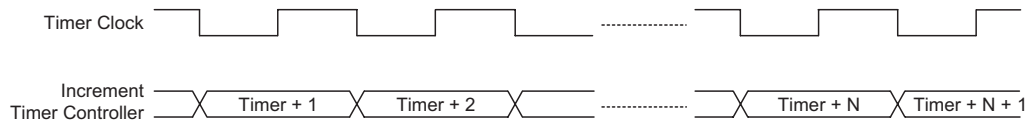






### 定时器模式

在这个模式，定时/计数器可以用来测量固定时间间距，当定时器发生溢出时，就会提供一个内部中断信号。要工作在这个模式，TMRC 寄存器中位 TM1/TM0、T0M1/T0M0，T1M1/T1M0 或 T2M1/T2M0 必须分别设为“1”和“0”。在这个模式，内部时钟源被用来当定时器的计时源。定时/计数器的输入计时频率可以是  $f_{SYS}$ 、 $f_{SYS}/4$ ， $f_{RTC}$ ，时基中断或定时/计数器 0 溢出中断，这取决于所使用的定时器，时钟源掩膜选项和 TS、T0S 或者 T1S 位的逻辑状态。定时器打开位 TON、T0ON，T1ON 或 T2ON 必须被设为逻辑高才能让定时器开始工作。每次内部时钟由高到低的电平转换都会使定时器值增加一；当定时器已满即溢出时，会产生中断信号且定时器会重新载入已经载入到预置寄存器的值，然后继续向上计数。定时器溢出是中断的一种，也是唤醒暂停模式的一种方法，但是如果相应特殊寄存器中的 ETI、ETOI，ETI1 或 ET2I 位被置 0，那么内部中断将被关闭。

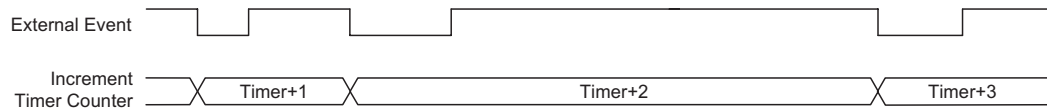


定时器模式时序图

### 事件计数模式

在这个模式，发生在外部引脚的外部逻辑事件改变的数目，可以通过内部定时/计数器来记录。为使定时/计数器工作于事件计数模式，TMRC 寄存器中位 TM1/TM0、T0M1/T0M0，T1M1/T1M0 或 T2M1/T2M0 必须分别设为“0”和“1”。定时器打开位 TON、T0ON，T1ON 或 T2ON 必须设为逻辑高，令定时器开始计数。当 TE、T0E，T1E 或 T2E 为逻辑低时，每次外部定时/计数器引脚接收到由低到高电平的转换将使计数器加一。而当 TE、T0E，T1E 或 T2E 为逻辑高时，每次外部定时/计数器引脚接收到由高到低电平的转换将使计数器加一。与另外两个模式一样，当计数器计满时，将会发生溢出且产生一个内部中断信号，同时定时/计数器重新载入一个已经载入到预置寄存器的值。由于外部定时/计数器引脚与其它输入/输出引脚共用，为确保事件计数模式正常工作，需要确定 TM1/TM0、T0M1/T0M0，T1M1/T1M0 或 T2M1/T2M0 位将定

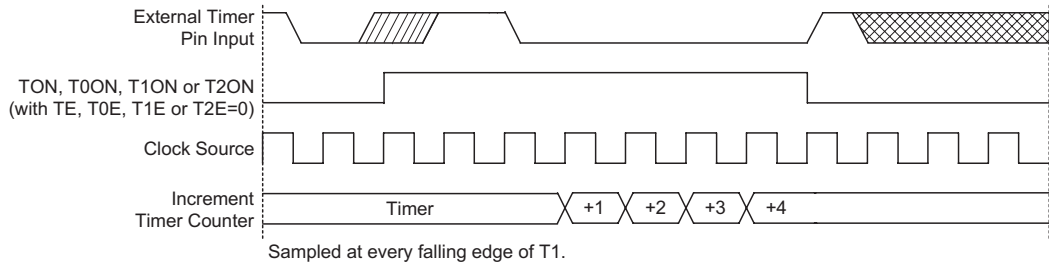
时/计数器设定在事件计数模式。定时/计数器的溢出是中断的一种，也是唤醒暂停模式的一种方法。



事件计数模式时序图

### 脉冲宽度测量模式

在这个模式，可以测量外部定时/计数器引脚的外部脉冲宽度。在脉冲宽度测量模式中，定时/计数器时钟源由内部时钟提供，而位 TM1/TM0、T0M1/T0M0，T1M1/T1M0 或 T2M1/T2M0 则必须都设为逻辑高。如果 TE、T0E、T1E 或 T2E 位是逻辑低，当外部定时/计数器引脚接收到一个由高到低电平的转换时，定时/计数器将开始计数直到外部定时/计数器引脚回到它原来的高电平，此时 TON、T0ON、T1ON 或 T2ON 位将自动清除为零且定时/计数器将停止计数。而如果 TE、T0E、T1E 或 T2E 位是逻辑高，则当外部定时/计数器引脚接收到一个由低到高电平的转换时，定时/计数器开始计数直到外部定时/计数器引脚回到原来的低电平。请注意，在脉冲宽度测量模式，当外部定时/计数器引脚的外部控制信号回到原来的低电平，TON、T0ON、T1ON 或 T2ON 位将自动地清除为零，而在其它两种模式下，TON、T0ON、T1ON 或 T2ON 位要在程序控制下才会被清除为零。这时定时/计数器中剩下的值被程序读取，可以由此得知外部定时/计数器引脚接收到的脉冲的长度。当 TON、T0ON、T1ON 或 T2ON 位被清零时，任何在外部定时/计数器引脚的进一步转换将被忽略，而直到 TON、T0ON 或者 T1ON 位再次被程序设定为逻辑高，定时/计数器才又开始脉冲宽度测量。利用这种方法可以轻松完成单一脉冲测量，要注意的是在这种模式下，定时/计数器是通过外部定时/计数器引脚上的逻辑转换来控制，而不是通过逻辑电平。与另外两个模式一样，当定时/计数器已满，将溢出且产生一个内部中断信号，定时/计数器也将清零并载入预置寄存器的值。由于外部定时/计数器引脚与其它输入/输出引脚共用，为确保脉冲宽度测量模式正常工作，需要确定 TM1/TM0、T0M1/T0M0、T1M1/T1M0 或 T2M1/T2M0 位将定时/计数器设定在脉冲宽度测量模式。定时/计数器的溢出是中断的一种，也是唤醒暂停模式的一种方法。

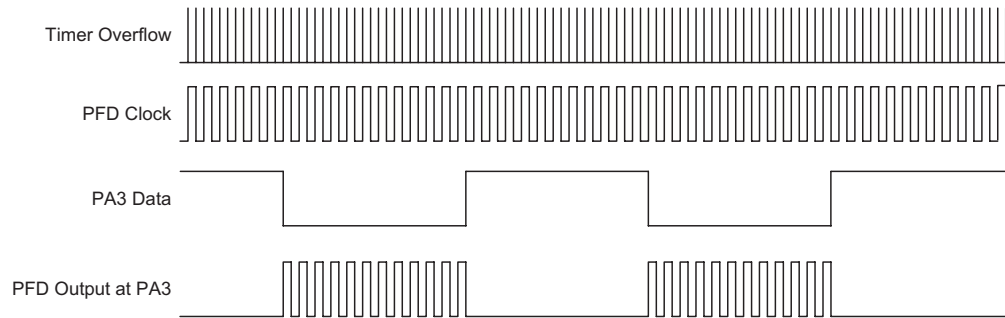


脉冲宽度测量模式时序图

### 可编程分频器 - PFD

PFD 输出与输入/输出引脚 PA3 共用。PFD 功能通过掩膜选项来选择，如果没有选择这项功能，则 PA3 可以作为一般的输入/输出引脚使用。定时/计数器的溢出信号是 PFD 电路的时钟源。请注意，HT49R50A-1/HT49C50-1/HT49C50L，HT49R70A-1/HT49C70-1/HT49C70L 和 HT49RU80/HT49CU80 具有一个以上内部定时/计数器，PFD 的时钟源可以通过掩膜选项确定是来自定时/计数器 0 还是定时/计数器 1。

定时器由内部系统时钟驱动，向预置寄存器内写值可以控制它的初始值。定时器从预置寄存器内的值开始向上计数，直到溢出，并由此产生溢出信号而导致 PFD 改变输出状态。接下来定时器会自动重新载入预置寄存器内的值并继续计数。PFD 的频率是定时/计数器溢出信号频率的一半。关于定时/计数器设置与操作的详细内容，请参考相关章节。只有 PA3 被清“0”，PFD 的输出才有效。这个输出数据位相当于 PFD 输出的开/关控制位。请注意，如果 PA3 输出数据位被设为“1”，则 PFD 输出逻辑低。如果掩膜选项选择引脚 PA3 的功能为 PFD 输出，则引脚 PA0~PA3 将被配置为 CMOS 形式。



假如系统时钟使用晶体振荡器，则使用这种频率产生的方法可以产生非常精确的频率值。

### 输入/输出接口

当运行在事件计数器或脉冲宽度测量模式时，定时/计数器需要使用外部定时/计数器引脚以确保正确的动作。外部定时/计数器引脚与其它 PB 口输入引脚共用，因此始终连接上拉电阻。定时器可以设置为驱动引脚共用的 PFD。当通过掩膜选项选择 PFD 引脚时，定时器可以根据定时控制寄存器的内容和时钟源频率，以不同的频率来驱动 PFD。

### 编程注意事项

当定时/计数器运行在定时器模式时，定时器的时钟源是使用内部系统时钟，与单片机所有运算都能同步。在这个模式下，当定时器寄存器溢出时，单片机将产生一个内部中断信号，使程序进入相应的内部中断向量。对于脉冲宽度测量模式，计数器的时钟源也是使用内部系统时钟，但定时器只有在正确的逻辑条件出现在外部定时/计数器输入引脚时才执行动作。当这个外部事件没有和内部定时器时钟同步时，只有当下一个定时器时钟到达时，单片机才会看到这个外部事件，因此在测量值上可能有小的差异，需要程序设计师在程序应用时加以注意。同样的情况发生在定时器配置为外部事件计数模式时，它的时钟来源是外部事件，和内部系统时钟或者定时器时钟不同步。

对于 HT49R50A-1/HT49C50-1/HT49C50L、HT49R70A-1/HT49C70-1/HT49C70L 和 HT49RU80/HT49CU80 具有两个或三个内部定时/计数器的单片机，有一项掩膜选项使能定时/计数器 0 的溢出成为定时/计数器 1 的时钟源。通过这种串联方式，HT49R50A-1/HT49C50-1/HT49C50L 可以创造出一个 16 位的定时/计数器，而 HT49R70A-1/HT49C70-1/HT49C70L 和 HT49RU80/HT49CU80 可以创造出一个 24 位的定时/计数器。如果以这种串联方式使用定时/计数器，则在初始化程序时，定时/计数器 1 必须先使能，然后在使用前立刻除能。

下面的程序范例，是以 HT49R70A-1/HT49C70-1/HT49C70L 为例，表明如何初始化中断和定时控制寄存器，以及定时器使能位如何控制定时/计数器的开/关。在这个范例中，定时/计数器 0 作为定时/计数器 1 的时钟源，这是通过设定掩膜选项将两个定时/计数器串联在一起，进而得到一个 24 位定时/计数器。请注意在此配置中，定时/计数器 1 必须先使能，然后马上除能以确保正确的初始化。

范例：使用定时/计数器 0 作为定时/计数器 1 的时钟源，配置一个 24 位计数器

```

START:
    mov a,09h                ; Set ET0I & EMI bits to enable Timer 0 and global
                            ; interrupt
    mov intc0,a              ;
    mov a,01h                ; Set the ET1I bit to enable the Timer 1 interrupt
    mov intc1,a              ;
    mov a,80h                ; Configure Timer 1 to operate in timer mode
    mov tmr1c,a              ; Timer 1 clock source depends on configuration option
    mov a,0a0h               ; Configure Timer 0 to operate in timer mode
    mov tmr0c,a              ; and select system clock/4 as Timer 0 clock source
    set tmr1c.4              ; Enable then disable Timer 1
    clr tmr1c.4              ; Necessary step for cascaded timers
    mov a,00h                ; Load a desired value into both the TMR0 and TMR1
                            ; registers

    mov tmr0,a               ;
    mov a,00h                ;
    mov tmr1l,a              ;
    mov tmr1h,a              ;
    set tmr0c.4              ;Turn Timer 0 on
    set tmr1c.4              ;Turn Timer 1 on
END
    
```

## 中断

LCD 型单片机均具有一些外部和内部中断功能。外部中断由外部引脚  $\overline{\text{INT0}}$  和  $\overline{\text{INT1}}$  的动作控制。内部中断由不同的时钟源控制，如定时/计数器、时基和实时时钟。另外，HT49RU80/HT49CU80 还具有 UART 中断和多功能中断。

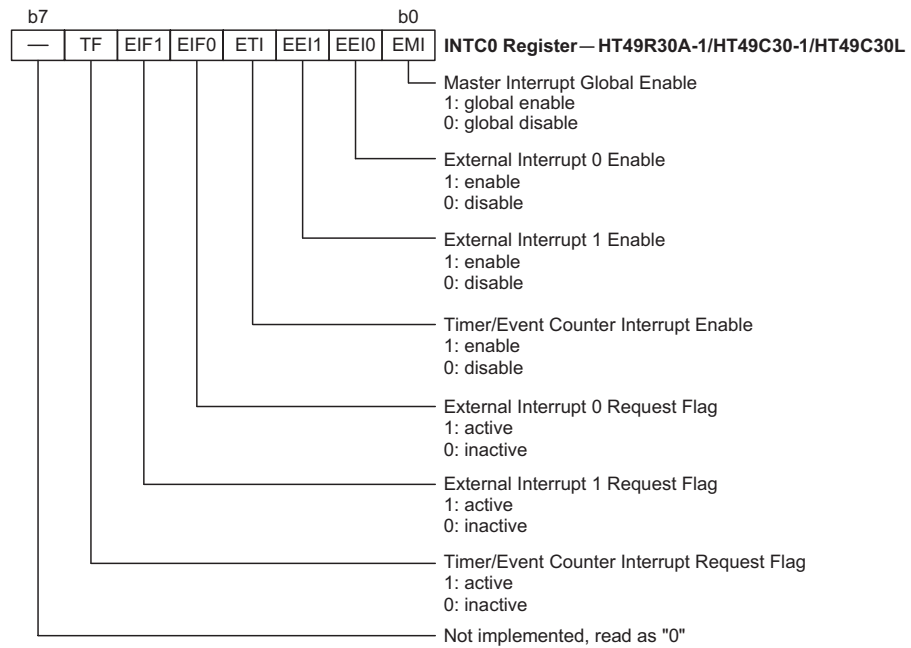
### 中断寄存器

对于 LCD 型单片机而言，提供了三个中断控制寄存器—INTC0,INTC1 和 MFIC 来控制所有的中断控制特性。注意，只有 HT49RU80/HT49CU80 才具有 MFIC 寄存器。

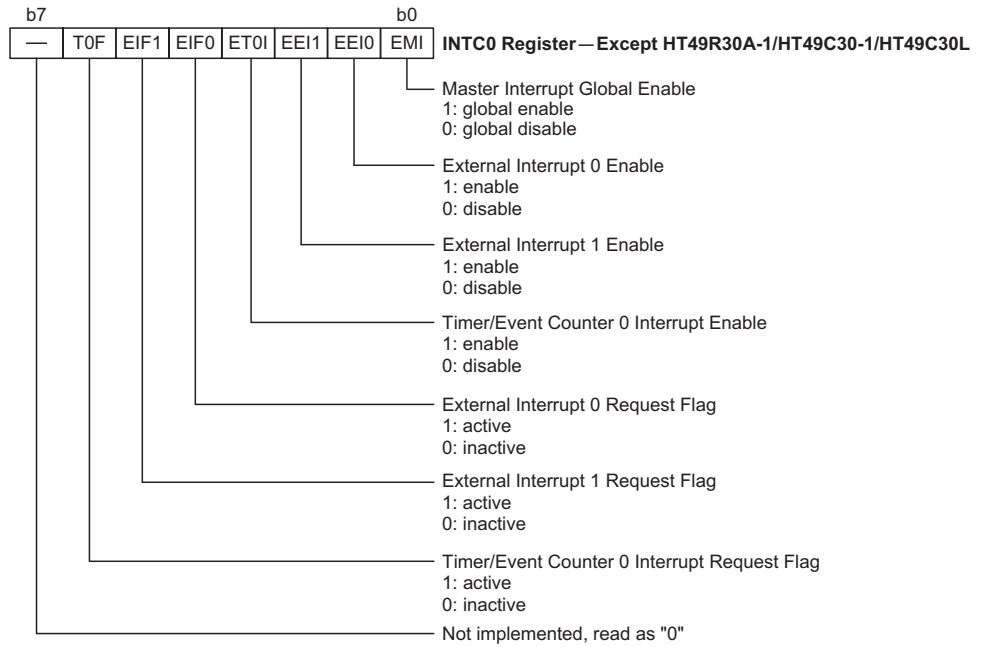
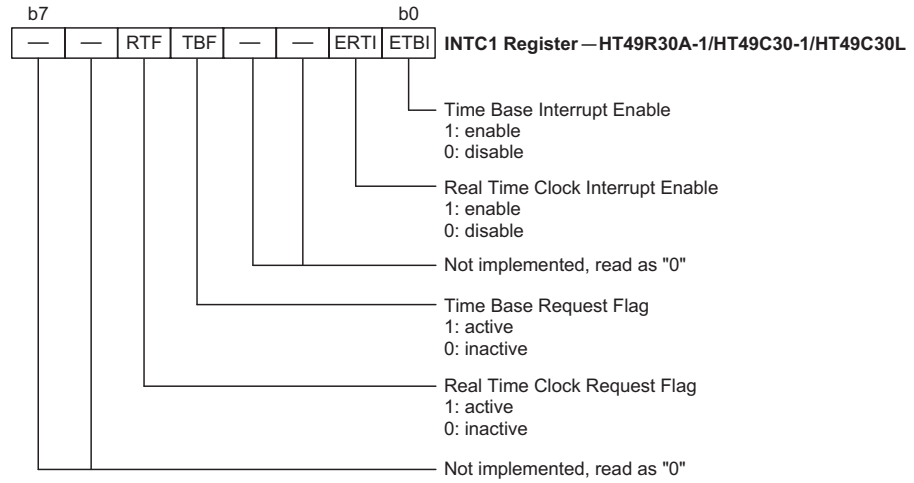
一旦中断子程序被响应，由于 EMI 被自动清零，所有其它的中断将被屏蔽。这个方式可以预防任何进一步的中断相互嵌套。然而，如果其它的中断请求可

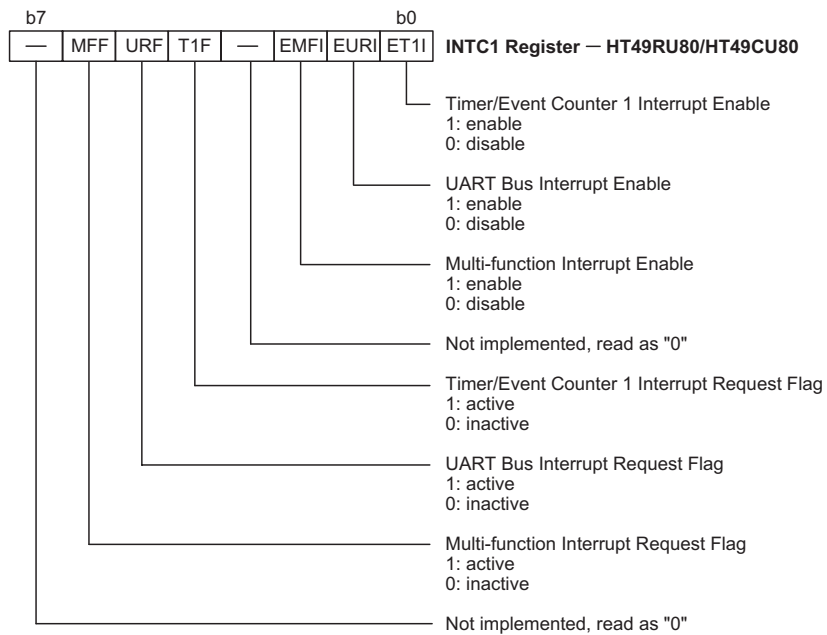
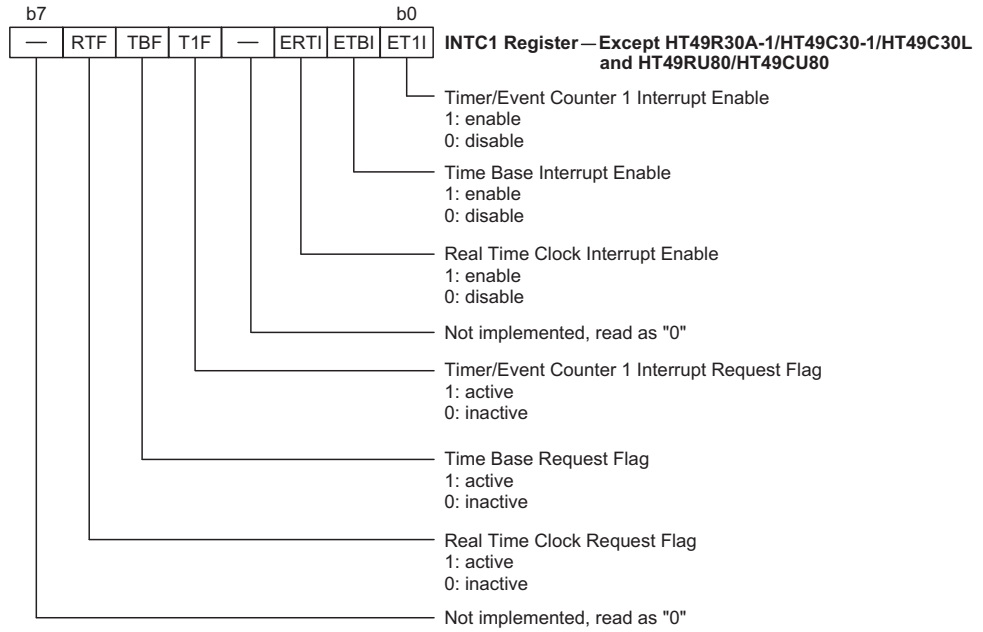
能发生在这个期间，但中断不会立即被响应，只有中断请求标志位会被记录。如果某个中断服务子程序正在执行，此时有另一个中断要求响应，则在进入子程序后应该置位 EMI 位，以允许中断嵌套。如果堆栈已满，即使此中断已经被使能，中断请求也不会响应，直到状态指针减少为止。如果要求立刻动作，堆栈必须避免成为储满状态。

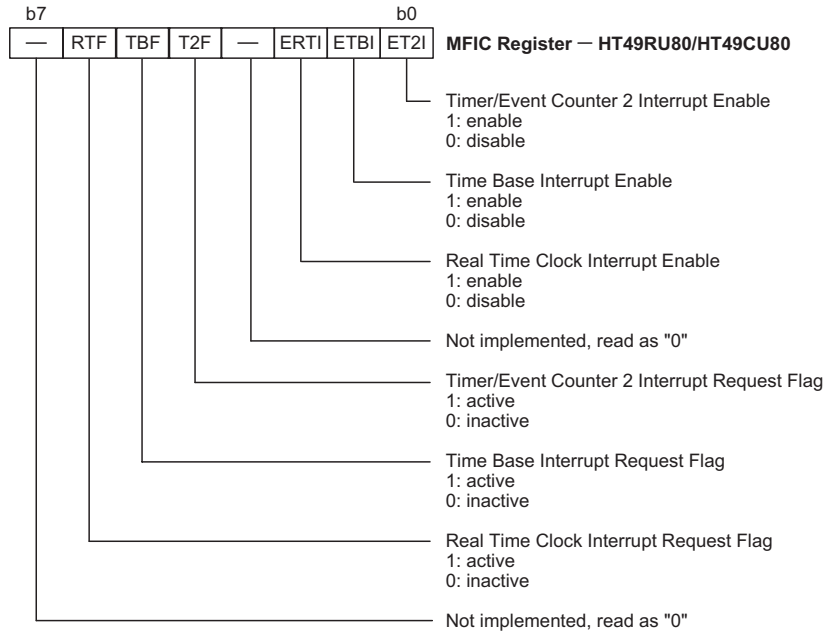
在暂停模式中，所有的中断都具有唤醒单片机的能力。当中断被响应时，首先将程序计数器的值压入堆栈，并跳转到程序存储器中特殊地址的子程序。只有程序计数器的值会被压入堆栈。如果累加器、状态寄存器或是其它寄存器的内容会被中断服务程序改变，而这些变化可能会破坏原先想要的控制顺序，则这些内容应该预先加以储存。



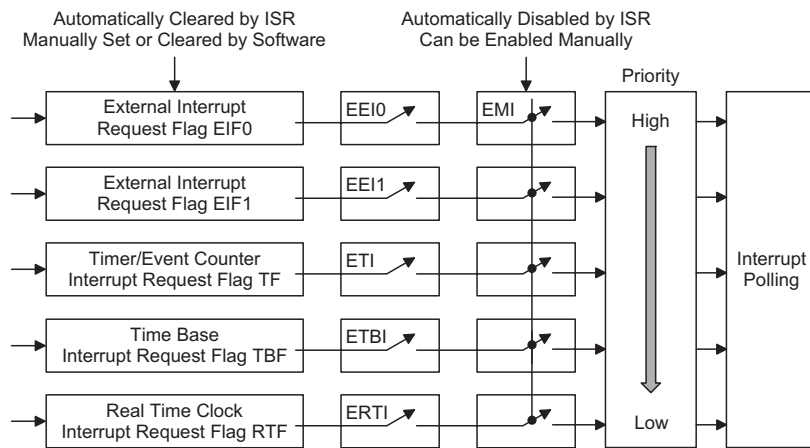




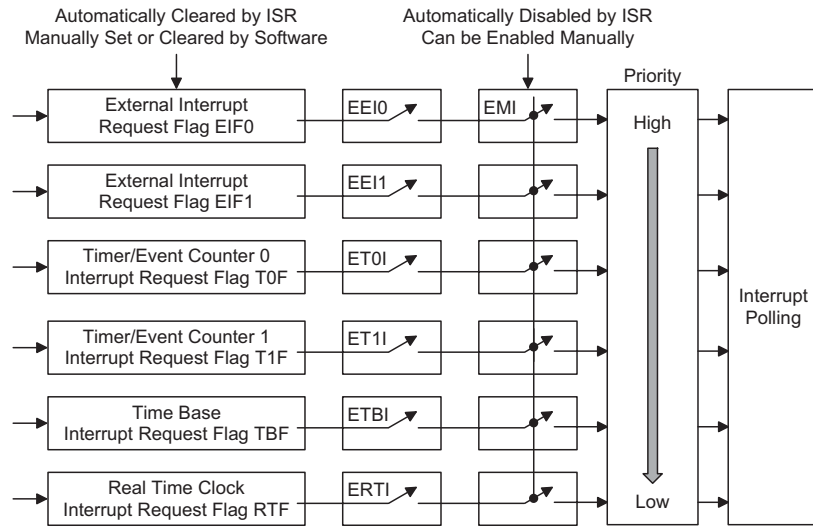




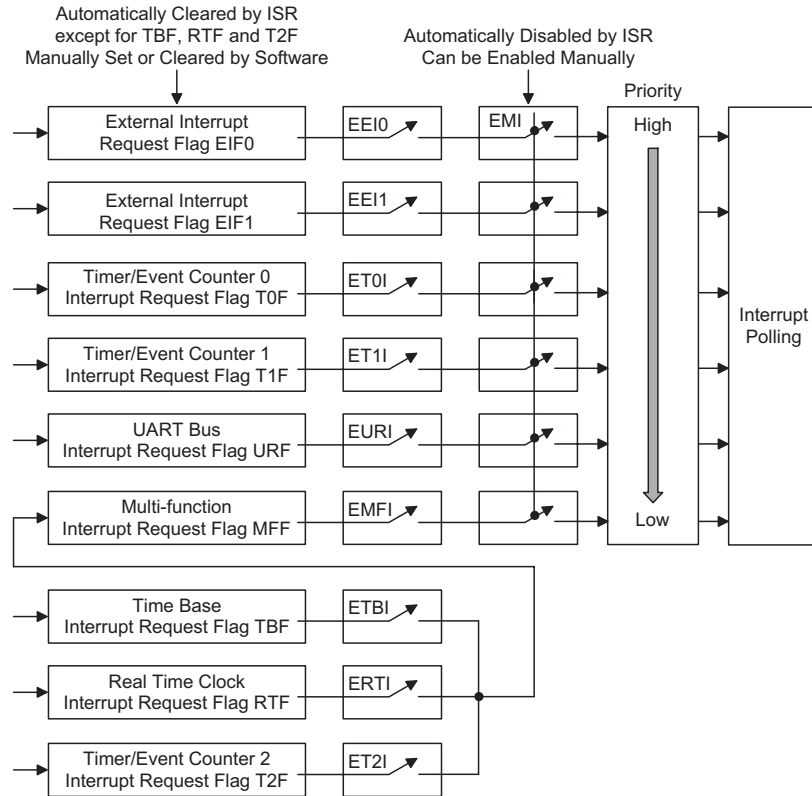
下图标明了各个中断的允许位，相关的请求标志位，以及中断优先顺序。



中断示意图—HT49R30A-1/ HT49C30-1/HT49C30L



中断示意图一 HT49R50A-1/HT49C50-1/HT49C50L  
HT49R70A-1/HT49C70-1/HT49C70L



中断示意图一 HT49RU80/HT49CU80

### 中断优先权

当中断是发生在两个连续的 T2 脉冲上升沿之间时，如果相应的中断请求被允许，中断将在后一 T2 脉冲响应。下表指出在同时提出请求的情况下所提供的优先权。

中断源	HT49R30A-1 HT49C30-1 HT49C30L 优先权	HT49R50A-1 HT49C50-1 HT49C50L 优先权	HT49R70A-1 HT49C70-1 HT49C70L 优先权	HT49RU80 HT49CU80 优先权
外部中断 0	1	1	1	1
外部中断 1	2	2	2	2
定时/计数器或定时计数器 0 溢出	3	3	3	3
定时/计数器 1 溢出	N/A	4	4	4

定时/计数器 2 溢出	N/A	N/A	N/A	6
UART 中断	N/A	N/A	N/A	5
时基中断	4	5	5	6
RTC 中断	5	6	6	6
多功能中断	N/A	N/A	N/A	6

**注意:**

1. 对于 HT49R30A-1/HT49C30-1/HT49C30L 只具有一个定时/计数器。  
HT49R50A-1/HT49C50-1/HT49C50L 和 HT49R70A-1/HT49C70-1/HT49C70L 具有两个内部定时器 HT49RU80/HT49CU80 具有三个内部定时器。
2. 只有 HT49RU80/HT49CU80 具有一个 UART 中断。
3. 在 HT49RU80/HT49CU80 中，定时/计数器 2、时基和实时时钟中断都包含在一个多功能中断向量中。

假如外部和内部中断均使能且外部和内部中断同时发生，则外部中断优先处理。使用 INTC0, INTC1 和 MFIC 寄存器适当屏蔽个别中断，可以防止同时发生的情总。外部中断引脚  $\overline{INT0}$  和  $\overline{INT1}$  分别与输入引脚 PB0 和 PB1 共用，如果寄存器的位正确的写值，可以配置为外部中断引脚。请注意，这些输入引脚一直连接上拉电阻。

### 外部中断

每款 LCD 型单片机都具有两组外部中断，即外部中断 0 和外部中断 1，相应的有两个外部输入共用引脚  $\overline{INT0}$  和  $\overline{INT1}$ 。要使外部中断发生，相应的外部中断使能位必须被置位。对于外部中断 0，为 INTC0 寄存器的第 1 位，即 EEI0。对于外部中断 1，为 INTC0 寄存器的第 2 位，即 EEI1。外部中断 0 是通过  $\overline{INT0}$  端口上由高到低的电平转换来触发，之后相应的中断请求标志位 (EIF0; INTC0 的第 4 位) 被置位。外部中断 1 是通过  $\overline{INT1}$  端口由高到低的电平转换来触发，之后相应的中断请求标志位 (EIF1; INTC0 的第 5 位) 被置位。当中断被使能且堆栈没有满，而  $\overline{INT0}$  发生由高到低的电平转换，将调用地址 04H 的子程序。如果是  $\overline{INT1}$  发生由高到低的电平转换，将调用地址 08H 的子程序。中断请求标志位 EIF0 或 EIF1(取决于发生哪个外部中断)会被复位且 EMI 位会被清零以除能其它中断。

### 定时/计数器中断

要使定时/计数器内部中断发生，相应的内部中断使能位必须被置位。对于具有一组内部定时/计数器的单片机，中断使能位是 INTC0 寄存器的第 3 位，

即 ETI。对于具有两组内部定时/计数器的单片机，定时/计数器 0 的中断使能位是 INTC0 寄存器的第 3 位，即 ET0I。定时/计数器 1 的中断使能位是 INTC1 寄存器的第 0 位。对于具有三组定时器的 HT49RU80/HT49CU80，定时器 0 的中断使能位 ET0I 为 INTC0 寄存器的第三位，定时器 1 的中断使能位 ET1I 为 INTC1 寄存器的第 0 位，而定时器 2 的中断使能位 ET2I 为 MFIC 寄存器的第 0 位。当定时/计数器溢出，会置位定时/计数器中断请求标志位，定时/计数器中断发生。在只有一组定时/计数器的 HT49R30A-1/HT49C30-1/HT49C30L 中，此位是 INTC0 寄存器的第 6 位，即 TF。而在具有两组定时/计数器的 HT49R50A-1/HT49C50-1/HT49C50L，HT49R70A-1/HT49C70-1/HT49C70L 和 HT49RU80/HT49CU80 中，定时/计数器 0 的请求标志位是 INTC0 寄存器的第 6 位，即 T0F。定时/计数器 1 的请求标志位是 INTC1 寄存器的第 4 位，即 T1F。对于有三组定时器的 HT49RU80/HT49CU80，定时器 2 的请求标志 T2F 为 MFIC 寄存器的第 4 位。由于定时器 2 的中断向量包含在多功能中断向量中，所以 INTC1 寄存器中多功能中断允许位 EMFI 必须置位才能响应定时器 2 产生的中断。当定时器 2 溢出时，INTC1 寄存器中的第六位多功能请求标志位 MFF 也会被置位。

当主中断使能位被置位、堆栈未满且相应内部中断使能位置位，定时/计数器溢出时会发生内部中断。对于具有一组定时器的单片机，该中断将产生地址 00CH 处的子程序调用。而对于具有两组定时器的单片机，由定时器 0 产生的中断，调用地址 00CH 处的子程序，而由定时器 1 产生的中断，调用地址 010H 处的子程序。对于具有三组定时器的 HT49RU80/HT49CU80，由定时器 0 产生的中断，调用地址 00CH 处子程序，由定时器 1 产生的中断，调用地址 010H 处子程序，而由定时器 2 产生的中断，调用地址 018H 处的子程序。需要注意的是，定时器 2 和其它一些中断共享一个中断向量。当内部中断被响应时，中断请求标志位 TF、T0F 或 T1F 会被复位且 EMI 位会被清零以除能其它中断。当 HT49RU80/HT49CU80 中的定时器 2 发生中断时，请求标志 MFF 会被复位且 EMI 会被清零以除能其它中断，但是请求标志 T2F 不会被自动复位，必须软件清零。

### 时基中断

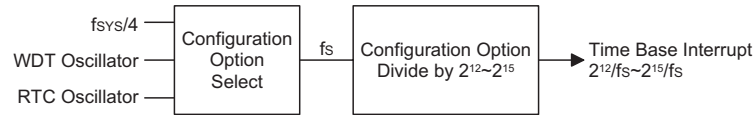
要时时基中断发生，相应的内部中断使能位 ETBI 必须被置位。对于 HT49R30A-1/HT49C30-1/HT49C30L，此位为 INTC1 寄存器的第 0 位，而对于 HT49R50A-1/HT49C50-1/HT49C50L、HT49R70A-1/HT49C70-1/HT49C70L，则

为 INTC1 寄存器的第 1 位，而对于 HT49RU80/HT49CU80，则为 MFIC 寄存器的第一位。当时基产生暂停信号时，就会置位中断请求标志位 TBF，产生时基中断。在 HT49R30A-1/HT49C30-1/HT49C30L 中，此位为 INTC1 寄存器的第 4 位，而在 HT49R50A-1/HT49C50-1/HT49C50L 和 HT49R70A-1/HT49C70-1/HT49C70L 中，则为 INTC1 寄存器的第 5 位。而在 HT49RU80/HT49CU80 中，则为 MFIC 寄存器的第 5 位。对于 HT49RU80/HT49CU80，由于时基的中断向量包含在多功能中断向量中，所以 INTC1 寄存器多功能中断向量允许位 EMFI 必须置位才能响应时基产生的中断。当时基溢出时，INTC1 寄存器中的第 6 位多功能请求标志位 MFF 也会被置位。当主中断使能位被置位、堆栈未满且对应的时基中断使能位被置位时，一旦时基产生暂停信号，就会产生内部时基中断。对于 HT49R30A-1/HT49C30-1/HT49C30L，将调用地址 010H 的子程序，而对于 HT49R50A-1/HT49C50-1/HT49C50L 和 HT49R70A-1/HT49C70-1/HT49C70L，将调用地址 14H 的子程序。而对于 HT49RU80/HT49CU80，将调用地址 018H 处的子程序，需要注意的是，时基和其它一些中断共享一个多功能中断向量。除了 HT49RU80/HT49CU80，当时基中断被响应，中断请求标志位 TBF 会被复位且 EMI 位将被清零以除能其它中断，而对于 HT49RU80/HT49CU80 时基发生中断，请求标志 MFF 会被复位且 EMI 位会被清零以除能其它中断，但是请求标志 TBF 不会自动复位，必须采用软件清零。

时基中断的目的是要提供一个具有固定时间周期的中断信号。时基中断的时钟源是内部时钟 fS，这个 fS 输入时钟先通过一个分频器，分频比例可通过掩膜选项进行选择，以提供更长的时基中断周期。时基中断的暂停周期范围由  $212/fS$  至  $215/fS$ 。产生 fS 的时钟源同时也决定时基中断的周期，它有三种不同的来源，分别为 RTC 振荡器、看门狗定时振荡器或系统振荡器/4，至于选择何种时钟源作为 fS 时钟源则是由掩膜选项来选择。请注意，如果选择 RTC 振荡器作为系统时钟，则 fS 和相应的时基中断也将使用 RTC 振荡器作为时钟源。

时基中断的周期由掩膜选项控制，它同时决定内部时钟 fS 的来源和内部的分频比例。



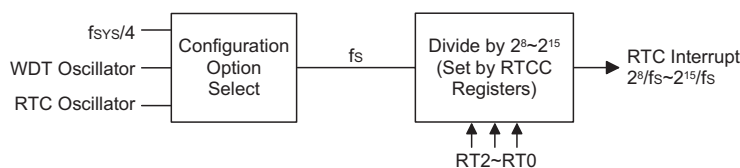


时基中断

### 实时时钟中断

要使实时时钟中断发生，相应的内部中断使能位 ERTI 必须被置位。对于 HT49RU80/HT49CU80 此位为 MFIC 寄存器的第 2 位，对于 HT49R30A-1/HT49C30-1/HT49C30L，此位为 INTC1 寄存器的第 1 位，而对于其他单片机，则为 INTC1 寄存器的第 2 位。当实时时钟产生暂停信号时，就会置位中断请求标志位 RTF，产生实时时钟中断。当主中断使能位被置位、堆栈未满足且相应的实时时钟中断使能位被置位时，一旦实时时钟产生暂停信号，就是产生内部实时时钟中断，对于 HT49R30A-1/HT49C30-1/HT49C30L，将调用地址 014H 的子程序，而对于 HT49R50A-1/HT49C50-1/HT49C50L 和 HT49R70A-1/HT49C70-1/HT49C70L，将调用地址 18H 的子程序。对于 HT49RU80/HT49CU80，由于实时时钟的中断向量包含在多功能中断向量中，所以 INTC1 寄存器多功能中断允许位 EMFI 必须置位才能响应实时时钟产生的中断。当实时时钟溢出时，INTC1 寄存器的第 6 位多功能请求标志位 MFF 也会被置位。除了 HT49RU80/HT49CU80，当实时时钟中断被响应，中断请求标志位 RTF 会被复位且 EMI 位将被清零以除能其它中断，而 HT49RU80/HT49CU80 发生实时时钟中断，请求标志 MFF 会被复位且 EMI 位会被清零以除能其它中断，但是请求标志 RTF 不会自动复位，必须软件清零。不要混淆 RTC 中断与 RTC 振荡器。

相似于时基中断，RTC 中断的目的也是要提供一个具有固定时间周期的中断信号。RTC 中断的时钟源来自于内部时钟源  $f_S$ ，这个  $f_S$  输入时钟先通过一个分频器，分频比例可通过写入数据到 RTCC 寄存器中适当的位，以获得更长的 RTC 中断周期，其范围由  $2^8/f_S$  至  $2^{15}/f_S$ 。产生  $f_S$  的时钟源同时也决定 RTC 中断的周期，它有三种不同的来源，分别为 RTC 振荡器、看门狗定时振荡器或系统振荡器/4，至于选择何种时钟源作为  $f_S$  时钟源则是由掩膜选项来选择。请注意，如果选择 RTC 振荡器作为系统时钟，则  $f_S$  和相应的 RTC 中断也将使用 RTC 振荡器作为时钟源。



### RTC 中断

请注意，RTC 中断周期是由掩膜选项和内部寄存器 RTCC 控制。掩膜选项用于选择内部时钟  $f_S$  的时钟源，而 RTCC 寄存器的 RT2、RT1 和 RT0 位，则是选择  $2^8/f_S$  至  $2^{15}/f_S$  的分频比例。对于 RTC 的中断周期，请参考 RTCC 寄存器的章节。

**注意：**在单片机被唤醒后，系统需要 1024 个时钟周期来恢复正常运行。如果 32768Hz 的 RTC 振荡器被选择作为系统时钟，对于唤醒后对定时灵敏的 RTC 中断的应用程序而言，在选择  $2^8$ 、 $2^9$  和  $2^{10}$  的 RTC 中断分频时必须特别小心。对于这些分频比例，在唤醒后的 1024 个时钟周期内，接下来的 RTC 中断事件会被遗漏。

### UART 中断

对于具有一个串行口的 HT49RU80/HT49CU80，要使 UART 中断发生，相应的内部中断使能位 EURI 必须被置位，此位为 INTC1 的第一位。当 UART 产生暂停信号时，就会置位中断请求标志 URF，产生 UART 中断。当主中断使能位被置位，堆栈未满足且相应的 UART 中断使能位 EURI 被置位时，一旦 UART 产生请求，就会产生 UART 中断，将会调用地址 014H 处的子程序。当 UART 中断响应时，中断请求标志位 URF 会被复位且 EMI 位将被清零以除能其它中断。

产生 UART 中断有多种情况，例如，发送接收数据、帧错误检测和自动地址检测。这些状态反映在 UART 的状态寄存器 USR 中。UART 控制寄存器 UCR2 决定是否产生中断信号。更多关于这两个寄存器的介绍可查阅 UART 相关章节。

### 多功能中断

对于 HT49RU80/HT49CU80，具有一个多功能中断。不像其它的中断，它没有独立的中断源，而是由时基、实时时钟和定时器 2 这三个中断组成。要使多功能中断发生，相应的中断允许位 EMFI 必须置位，此位为 INTC1 的第 2 位。

当多功能中断产生暂停信号，就会置位中断请求标志 MFF，产生多功能中断。当主设备的中断使能位被置位、堆栈未滿且对应的中断使能位被置位时，一旦时基、实时时钟或定时器 2 溢出，就会产生中断，该中断将调用地址 018H 处的子程序。当多功能中断被响应，多功能中断请求标志位 MFF 会被复位且 EMI 位被清零以除能其它中断。需要注意的是，时基、实时时钟和定时器 2 的请求标志不会自动复位需软件清零。

在 HT49RU80/HT49CU80 中，时基、实时时钟和定时器 2 没有独立的中断向量，它们共享一个多功能中断向量。

### 编程注意事项

中断请求标志位 TF、T0F、T1F、T2E、URF、MFF、EIF0、EIF1、TBF 和 RTF 与中断使能位 ETI、ET0I、ET1I、ET2I、EURI、EMFI、EEI0、EEI1、ETBI 和 ERTI 共同形成数据存储器中的中断控制寄存器 INTC0，NTC1 和 MFIC。通过除能中断使能位，可以屏蔽中断请求。然而，一旦中断请求标志位被设定，它们会被保留在 INTC0，INTC1 和 MFIC 寄存器内，直到相应的中断服务子程序执行或被软件指令清除。

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

## 复位和初始化

复位功能在任何单片机中基本的部分，使得单片机可以设定一些与外部参数无关的先置条件。最重要的是，复位条件在初次提供电源给单片机后，经短暂延迟，内部电路将使得单片机被定义在良好的状态且准备执行第一条程序语句。上电复位之后，在程序未开始执行前，部分重要的内部寄存器将会被预先设定状态。程序计数器就是其中之一，它会被清除为零，使得单片机从最低的程序存储器地址开始执行程序。

除了上电复位外，即使单片机正在执行状态，有些情况的发生也迫使单片机必须加以复位。其中一个例子是当提供电源给单片机以执行程序后， $\overline{\text{RES}}$  引脚被强制拉下至低电平。这个例子为正常操作复位，单片机中只有一些寄存器受影

响，而大部分寄存器则是不受影响，以便复位引脚回复至高电平后，单片机仍可以正常操作。复位的另一种形式是看门狗定时器溢出而复位单片机，所有复位操作类型导致不同的寄存器条件被加以设定。

另外一种复位以低电压复位，即 LVR 的型态存在，在电源提供电压低于某一临界值的情况下，一种和  $\overline{\text{RES}}$  引脚复位类似的完全复位将会被执行。

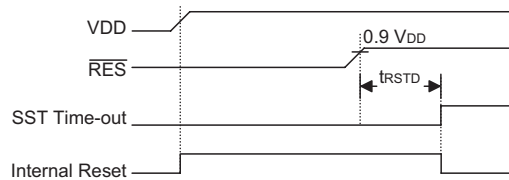
### 复位

通过内部与外部事件触发复位，单片机共有五种复位方式：

#### → 上电复位

这是最基本而不可避免的复位，发生在单片机上电后。除了保证程序存储器会从起始地址开始执行，上电复位也使得其它寄存器被设定在预设条件，所有的输入/输出寄存器在上电复位时会保持高电平，以确保上电后所有引脚被设为输入状态。

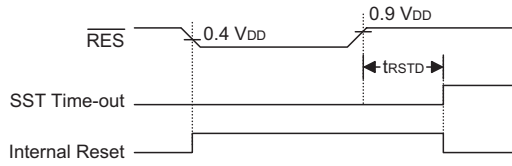
虽然单片机有一个内部 RC 复位功能，由于接通电源不稳定，还是推荐使用和  $\overline{\text{RES}}$  引脚连接的外部 RC 电路。由 RC 电路所造成的时间延迟使得  $\overline{\text{RES}}$  引脚在电源供应稳定前的一段延长周期内保持在低电平。在这段时间内，单片机的正常操作是被禁止的。 $\overline{\text{RES}}$  引脚达到一定电压值后，再经过延迟时间  $t_{\text{RST D}}$ ，单片机可开始进行正常操作。下图中 SST 是系统延迟周期 System Start-up Timer 的缩写。



上电复位时序图

#### → $\overline{\text{RES}}$ 引脚复位

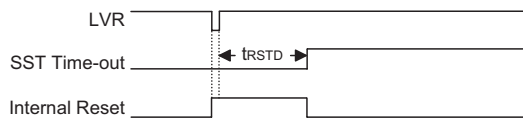
当单片机正常工作，而  $\overline{\text{RES}}$  引脚通过外部硬件(如外部开关)被强迫拉至低电平时，此种复位型式即会发生。这种复位形式与其它复位的例子一样，程序计数器会被清除为零且程序从头开始执行。



RES引脚复位时序图

→ 低电压复位 – LVR

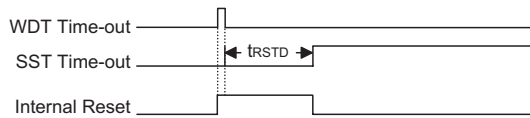
单片机具有低电压复位电路，可以监测它的电源电压。例如在更换电池的情况下，单片机供应的电压可能会落在  $0.9V \sim V_{LVR}$  的范围内，这是 LVR 将会自动从内部复位单片机。LVR 包含以下的规格：有效的 LVR 信号，即在  $0.9V \sim V_{LVR}$  的低电压，必须存在超过 1ms。如果低电压存在不超过 1ms，则 LVR 将会忽略它且不会执行复位功能。



低电压复位时序图

→ 正常操作时看门狗溢出复位

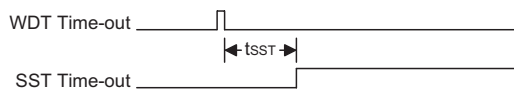
除了看门狗溢出标志位 TO 将被设为 1 之外，正常操作时看门狗溢出复位和 RES 复位相同。



正常操作时看门狗溢出复位时序图

→ 暂停时看门狗溢出复位

暂停时看门狗溢出复位有些不同于其它种类的复位，除了程序计数器与堆栈指针将被清除为 0 及 TO 标志位被设为 1 外，绝大部份的条件保持不变。图中  $t_{SST}$  的细节请见 A.C.特性。



暂停时看门狗溢出复位时序图

不同的复位方法以不同的方式影响复位标志位。这些标志位，即 PDF 和 TO，被放在状态寄存器中，由如暂停功能或看门狗计数器等多种控制器操作控制。复位标志位如下所示：

TO	PDF	复位条件
0	0	上电时的 RES 复位
u	u	一般运行时的 RES 复位或 LVR 低压复位
1	u	一般运行时的 WDT 溢出复位
1	1	HALT 暂停时的 WDT 溢出复位

“u”表示不变化

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

项 目	复位后情况
程序计数器	清除为零
中断	所有中断被除能
看门狗定时器/实时时钟中断/时基	复位后清除
定时/计数器	所有定时/计数器停止
输入/输出口	所有输入/输出口设为逻辑高
堆栈指针	堆栈指针指向堆栈顶端
UART	UART 除能， TX 和 RX 引脚设为 PC0/PC1

不同的复位以不同的方式影响单片机中的内部寄存器。为保证复位发生后程序的正常执行，在特定的复位发生后，了解单片机内的情况是非常重要的。下表描述了不同的复位如何影响单片机的内部寄存器。

**HT49R30A-1/HT49C30-1/HT49C30L**

寄存器	RES复位 (上电时)	RES或 LVR 复位 (一般运行时)	WDT 溢出复位 (一般运行时)	WDT 溢出复位 (HALT 暂停时)
MP0	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu
MP1	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu
BP	---- --0	---- --0	---- --0	---- --u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu
RTCC	--00 0111	--00 0111	--00 0111	--uu uuuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	0000 1---	0000 1---	0000 1---	uuuu u---

PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	--xx xxxx	--xx xxxx	--xx xxxx	--uu uuuu
INTC1	--00 --00	--00 --00	--00 --00	--uu --uu

“u”表示不变化

“x”表示不确定

“-”表示不存在

**HT49R50A-1/HT49C50-1/HT49C50L**

寄存器	RES复位 (上电时)	RES或LVR复位 (一般运行时)	WDT溢出复位 (一般运行时)	WDT溢出复位 (HALT暂停时)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	---- --0	---- --0	---- --0	---- --u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu
RTCC	--00 0111	--00 0111	--00 0111	--uu uuuu
STATUS	--00 xxxx	--uu uuuu	--lu uuuu	--ll uuuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1---	0000 1---	0000 1---	uuuu u---
TMR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	0000 1---	0000 1---	0000 1---	uuuu u---
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PC	---- 1111	---- 1111	---- 1111	---- uuuu
INTC1	-000 -000	-000 -000	-000 -000	-uuu -uuu

“u”表示不变化

“x”表示不确定

“-”表示不存在

**HT49R70A-1/HT49C70-1/HT49C70L**

寄存器	RES复位 (上电时)	RES或LVR复位 (一般运行时)	WDT溢出复位 (一般运行时)	WDT溢出复位 (HALT暂停时)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	---- --0	---- --0	---- --0	---- --u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu

TBLH	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
RTCC	--00 0111	--00 0111	--00 0111	--uu uuuu
STATUS	--00 x x x x	--uu u u u u	--1u u u u u	--11 u u u u
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
TMR0C	0000 1---	0000 1---	0000 1---	u u u u u---
TMR1H	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
TMR1L	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
TMR1C	0000 1---	0000 1---	0000 1---	u u u u u---
PA	1111 1111	1111 1111	1111 1111	u u u u u u u u
PB	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
PC	1111 1111	1111 1111	1111 1111	u u u u u u u u
INTC1	-000 -000	-000 -000	-000 -000	-uuu -uuu

“u”表示不变化

“x”表示不确定

“-”表示不存在

#### HT49RU80/HT49CU80

寄存器	RES复位 (上电时)	RES或LVR复位 (一般运行时)	WDT溢出复位 (一般运行时)	WDT溢出复位 (HALT暂停时)
MP0	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
MP1	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
BP	--0- --00	--0- --00	--0- --00	---- --u
ACC	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
TBLH	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
RTCC	--00 0111	--00 0111	--00 0111	--uu uuuu
STATUS	--00 x x x x	--uu u u u u	--1u u u u u	--11 u u u u
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
TMR0C	0000 1---	0000 1---	0000 1---	u u u u u---
TMR1H	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
TMR1L	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
TMR1C	0000 1---	0000 1---	0000 1---	u u u u u---
PA	1111 1111	1111 1111	1111 1111	u u u u u u u u
PB	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
PC	1111 1111	1111 1111	1111 1111	u u u u u u u u
PD	-111 1111	-111 1111	-111 1111	-uuu uuuu
INTC1	-000 -000	-000 -000	-000 -000	-uuu -uuu
TBHP	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
TMR2H	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
TMR2L	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
TMR2C	00-0 1---	00-0 1---	00-0 1---	uu-u u---



MFIC	-000 -000	-000 -000	-000 -000	-uuu -uuu
USR	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00×0	0000 00×0	0000 00×0	uuuu uuuu
UCR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
TXR/RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

“u”表示不变化  
 “x”表示不确定  
 “-”表示不存在

### 异步串行口——UART

此章节仅适用于 HT49RU80/HT49CU80。HT49RU80/HT49CU80 具有一个全双工的串行口。这样可以很方便的与其它具有串行口的芯片通讯。UART 功能占用一个内部中断向量，当接收到数据或数据发送结束，触发 UART 中断。

#### UART 特性

UART 具有的功能：

- 全双工异步传输
- 可选择 8 位或 9 位传输格式
- 可选择奇校验、偶校验或无校验
- 可选择 1 位或 2 位停止位
- 8 位预分频的波特率发生器
- 奇偶、帧、噪声和超越检测
- 支持地址检测
- 独立的发送和接收允许
- 两层 FIFO 接收缓冲器
- 发送和接收中断

下列条件可触发中断

- 发送完成
- 发送寄存器空闲
- a) 接收完成
- b) 过速错误
- c) 地址匹配

### UART 外部引脚

UART 是通过 TX 和 RX 引脚与外部芯片通讯。当 UCR2 寄存器的 TXEN 位为零，UART 发送功能被禁止，则 TX 引脚可作为普通 IO 口使用。同样的，当 UCR2 寄存器的 RXEN 位为零，UART 接收功能被禁止，则 RX 引脚可作为普通 IO 口使用。若 UARTEN、TXEN 和 RXEN 置位，这些 IO 口将自动转换成相应 TX 输出和 RX 输入并且对应的上拉电阻将无效。

### 数据发送

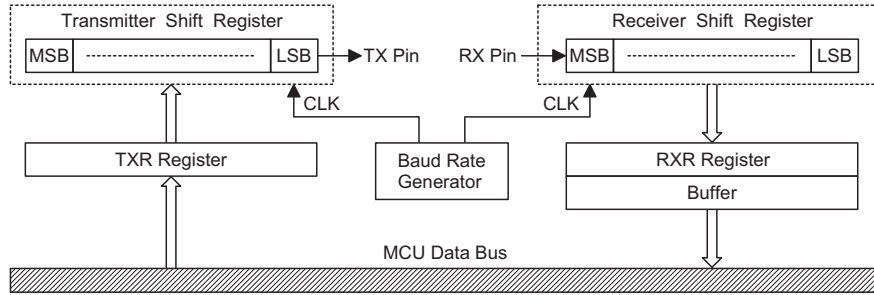
下图显示了 UART 的整体结构。需要发送的数据首先写入 TXR 寄存器，然后在波特率发生器的控制下将寄存器中数据一位位地移到 TX 引脚上，低位在前。只有 TXR 寄存器映像到单片机的数据存储器中，而发送移位寄存器没有实际地址，所以对用户程序不可见。

数据在波特率发生器的控制下，低位在前高位在后，从外部引脚 RX 进入接收移位寄存器。当数据接收完成，数据从接收移位寄存器移入可被用户程序操作的 RXR 寄存器中。只有 RXR 寄存器映像到单片机数据存储器中，而接收移位寄存器没有实际地址，所以对用户程序不可见。

需要注意的是，上文中提到的发送寄存器 TXR 和接收寄存器 RXR，其实是共享一个地址的数据寄存器 TXR/RXR 寄存器。

### UART 状态和控制寄存器

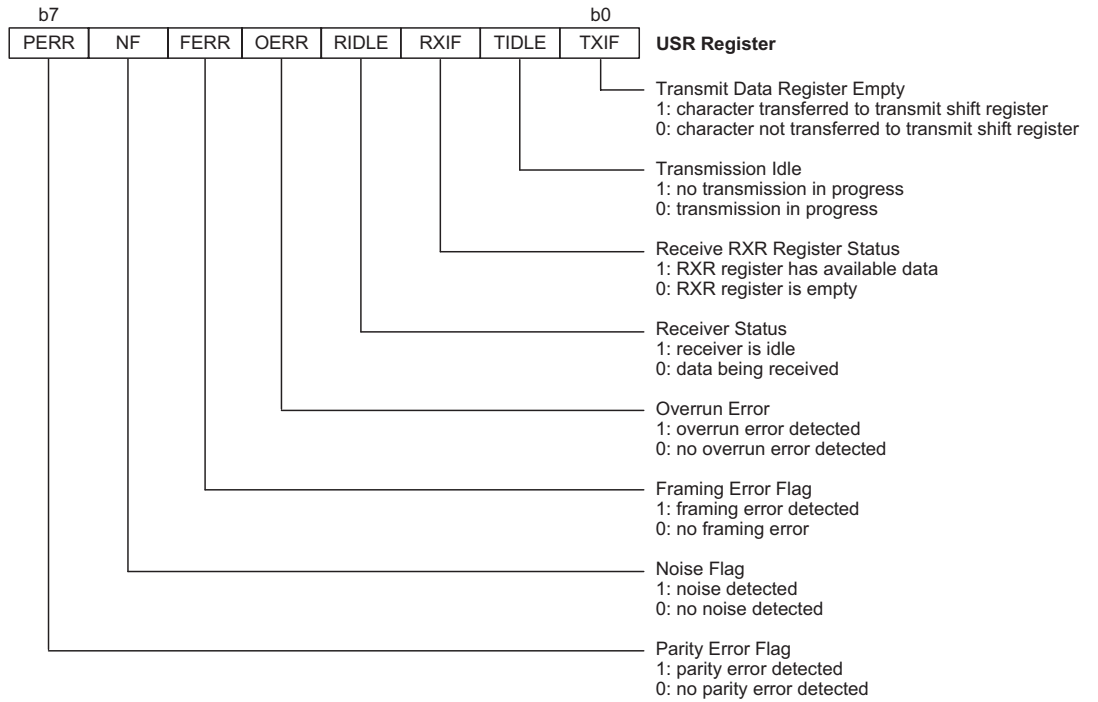
有 5 个寄存器与 UART 功能相关。寄存器 USR、UCR1 和 UCR2 全面控制 UART，而寄存器 BRG 控制波特率。发送和接收数据是通过寄存器 TXR/RXR 的。



UART 发送器框图

**USR 寄存器**

寄存器 USR 是 UART 的状态寄存器，可以通过程序读取。所有 USR 位是只读的。



详细解释如下：

- 振荡器 TXIF

TXIF 是发送数据寄存器为空标志。若 TXIF=0，数据还没有从缓冲器加载到移位寄存器中；若 TXIF=1，数据已从缓冲器中加载到移位寄存器中。读取 USR 寄存器再写 TXR 寄存器将清除 TXIF。当 TXEN 被置位，由于发送缓冲器未满，TXIF 也会被置位。

- TIDLE

TIDLE 是数据发送完标志位。若 TIDLE=0，数据传输中。当 TXIF=1 且数据发送完毕或者暂停字被发送时，TIDLE 置位。TIDLE=1，TX 引脚空闲。读取 USR 寄存器再写 TXR 寄存器将清除 TIDLE 位。

- RXIF

RXIF 是接收寄存器状态标志。当 RXIF=0，RXR 寄存器为空；当 RXIF=1，RXR 寄存器接收到新数据。当数据从移位寄存器中加载到 RXR 寄存器中，如果 UCR2 寄存器中的 RIE=1，则会触发中断。当接收数据时检测到一个或多个错误时，相应的标志位 NF、FERR 或 PERR 会在同一周期内置位。读取 USR 寄存器再读 RXR 寄存器，如果 RXR 寄存器中没有新的数据，那么将清除 RXIF 标志。

- RIDLE

RIDLE 是接收状态标志。若 RIDLE=0，正在接收数据；若 RIDLE=1，接收器空闲。在接收到停止位和下一个数据的起始位之间，RIDLE 被置位，表明 UART 空闲。

- OERR

OERR 是过速错误标志，表示接收缓冲器是否溢出。若 OERR=0，没有数据溢出；若 OERR=1，发生了过速错误，它将影响下一组数据的接收。先读取 USR 寄存器再读 RXR 寄存器将清除此标志位。

- FERR

FREE 是帧错误标志位。若 FREE=0，没有帧错误发生；若 FREE=1，当前的数据发生了帧错误。任何复位都会清除该标志位，也可以先读取 USR 寄存器再读 RXR 寄存器来清除此位。

● NF

NF 是噪声干扰标志。若 NF=0，没有受到噪声干扰；若 NF=1，UART 接收数据时受到噪声干扰。它与 RXIF 在同周期内置位，但不会与过速标志位同时置位。先读取 USR 寄存器再读 RXR 寄存器将清除此标志位。

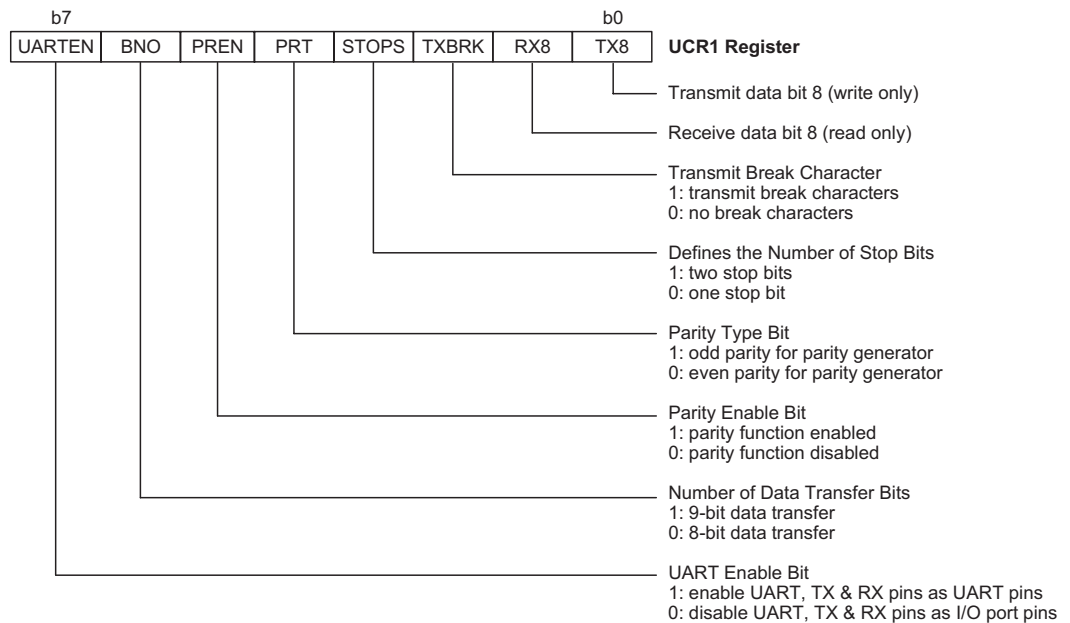
● PERR

PERR 是奇偶校验出错标志。若 PERR=0，奇偶校验正确；若 PERR=1，接收到的数据奇偶校验出错。只有使能了奇偶校验此位才有效。任何复位都会清除该标志位，也可以先读取 USR 寄存器再读 RXR 寄存器来清除此位。

不同的振荡器选择可以让使用者在不同的应用需求中获得更多范围的功能。LCD 系列单片机除了系统振荡器外，还含有看门狗振荡器。有三种系统时钟可供选择，看门狗定时器也有多种时钟源选择，提供给用户更大的灵活性。此外，连接到这些振荡器的内部电路也提供了多种定时和中断功能，其频率和操作可通过掩膜选项和内部寄存器决定。

→ UCR1 寄存器

UCR1 和 UCR2 是 UART 的两个控制寄存器，用来定义各种 UART 功能，例如 UART 的使能与除能、奇偶校验控制和传输数据的长度等等。



详细解释如下：

- TX8

此位只有在传输数据为 9 位的格式中有效，用来存储发送数据的第 9 位。BNO 是用来控制传输位数是 8 位还是 9 位。

- RX8

此位只有在传输数据为 9 位的格式中有效，用来存储接收数据的第 9 位。BNO 是用来控制传输位数是 8 位还是 9 位。

- TXBRK

TXBRK 是暂停字发送控制位。TXBRK=0，没有暂停字要发送，TX 引脚正常操作；TXBRK=1，将会发送暂停字，发送器将发送逻辑 0。若 TXBRK 为高，缓冲器中数据发送完毕后，发送器将至少保持 13 位宽的低电平直至 TXBRK 复位。

- STOP

此位用来设置停止位的长度。STOP=1，有两位停止位；STOP=0，只有一位停止位。

- PRT

奇偶校验选择位。PRT=1，奇校验；PRT=0，偶校验。

- PREN

此位为奇偶校验使能位。PREN=1，使能奇偶校验；PREN=0，除能奇偶校验。

- BNO

BNO 是发送位数控制位。BNO=1，传输数据为 9 位；BNO=0，传输数据为 8 位。若选择了 9 位数据传输格式，RX8 和 TX8 将分别存储接收和发送数据的第 9 位。

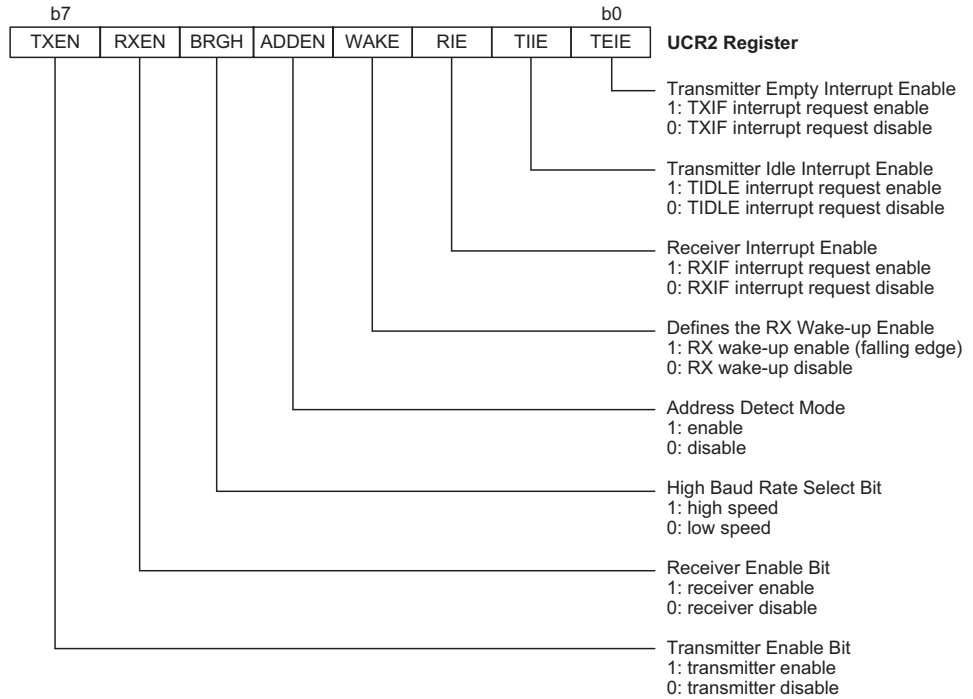
- UARTEN

此位为 UART 的使能位。UARTEN=0，UART 除能，RX 和 TX 可用作普通的输入输出；UARTEN=1，UART 使能，TX 和 RX 将分别由 TXEN 和 RXEN 控制。当 UART 被除能将清除缓冲器，所有缓冲器中的数据将被忽略，另外波特率计数器、错误和状态标志位被复位，TXEN、RXEN、TXBRK、RXIF、OERR、FERR 和 NF 清零而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2 和 BRG 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零，所有发送

和接收将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

→ **UCR2 寄存器**

UCR2 是 UART 的另一个控制寄存器，它的主要功能是使能或除能发送和接收允许以及 UART 的各种中断源。它也可用来控制波特率，使能接收唤醒和地址侦测。



详细解释如下：

- TEIE

此位为发送寄存器为空中断的使能或除能位。若 TEIE=1，当 TXIF 置位时，UART 的中断请求标志置位；若 TEIE=0，UART 中断请求标志不受 TXIF 的影响。

- TIIE

此位为发送器空闲中断的使能或除能位。若 TIIE=1，当 TIDLE 置位时，UART 的中断请求标志置位；若 TIIE=0，UART 中断请求标志不受 TIDLE 的影响。

- RIE

此位为接收中断使能或除能位。若 RIE=1，当 OERR 或 RXIF 置位时，UART 的中断请求标志置位；若 RIE=0，UART 中断请求标志不受 OERR 和 RXIF 影响。

- WAKE

此位为接收唤醒功能的使能和除能位。若 WAKE=1 且在暂停模式下，RX 引脚的下降沿将唤醒单片机；若 WAKE=0 且在暂停模式下，RX 引脚的任何边沿都不能唤醒单片机。

- ADDEN

此位为地址检测使能和除能位。ADDEN=1，地址检测使能，此时数据的第 8 位 (BON=0) 或第 9 位 (BON=1) 为高，那么接到的是地址而非数据。若相应的中断使能且接收到的值最高位为 1，那么中断请求标志将会被置位，若最高位为 0，那么将不会产生中断且收到的数据也会被忽略。

- BRGH

此位为波特率发生器高低速选择位，它和 BRG 寄存器一起控制 UART 的波特率。BRGH=1，为高速模式；BRGH=0，为低速模式。

- RXEN

此位为接收使能位。RXEN=0，接收将被除能，接收器立刻停止工作。另外缓冲器将被复位，此时 RX 引脚可作普通的输入输出端口使用。若 RXEN=1 且 UARTEN=1，则接收将被使能，RX 引脚将由 UART 来控制。在数据传输时清除 RXEN 将中止数据接收且复位接收器，此时 RX 引脚可作为普通输入输出端口使用。



- TXEN

此位为发送使能位。TXEN=0，发送将被除能，发送器立刻停止工作。另外缓冲器将被复位，此时 TX 引脚可作为普通的输入输出端口使用。若 TXEN=1 且 UARTEN=1，则发送将被使能，TX 引脚将由 UART 来控制。在数据传输时清除 TXEN 将中止数据发送且复位发送器，此时 TX 引脚可作为普通的输入输出端口使用。

### 波特率发生器

UART 自身具有一个波特率发生器，通过它可以设定数据传输速率。波特率是由一个独立的内部 8 位计数器产生，它由 BRG 寄存器和 UCR2 寄存器的第 2 位 BRGH 来控制。BRGH 是决定波特率发生器处于高速模式还是低速模式，从而决定计算公式的选用。BRG 寄存器的值 N 可根据下表中的公式计算，N 的范围是 0 到 255。

BRGH	0	1
波特率	$\frac{f_{sys}}{64(N+1)}$	$\frac{f_{sys}}{16(N+1)}$

为的得到相应的波特率，首先需要设置 BRGH 来选择相应的计算公式从而算出 BRG 的值。由于 BRG 的值不连续，所以实际波特率和理论值之间有一个偏差。下面举例怎样计算 BRG 寄存器中的值 N 和误差。

### 系统时钟配置波特率和误差的计算

系统选用 8M 晶振且 BRGH=0，若期望的波特率为 9600，计算它的 BRG 寄存器的值 N，实际波特率和误差。

根据上表，波特率  $BR = \frac{f_{sys}}{64(N+1)}$

转换后的公式  $N = \frac{f_{sys}}{BR \times 64} - 1$

代入参数  $N = \frac{8000000}{9600 \times 64} - 1 = 12.0208$

取最接近的值，十进制 12 写入 BRG 寄存器，实际波特率如下

$$BR = \frac{8000000}{64(12+1)} = 9615$$

$$\text{误差} = \frac{9615 - 9600}{9600} = 0.16\%$$

下面两表给出 BRGH 取不同值时的实际波特率和误差。

波特率 Kbps	BRGH=0											
	f <sub>SYS</sub> =8M			f <sub>SYS</sub> =7.159M			f <sub>SYS</sub> =4M			f <sub>SYS</sub> =3.579545M		
	BRG	Kbps	Err	BRG	Kbps	Err	BRG	Kbps	Err	BRG	Kbps	Err
0.3	-	-	-	-	-	-	207	0.3	0	185	0.3	0
1.2	103	1.202	0.16	92	1.203	0.23	51	1.202	0.16	46	1.19	-0.83
2.4	51	2.404	0.16	46	2.38	-0.83	25	2.404	0.16	22	2.432	1.32
4.8	25	4.807	0.16	22	4.863	1.32	12	4.808	0.16	11	4.661	-2.9
9.6	12	9.615	0.16	11	9.322	-2.9	6	8.929	-6.99	5	9.321	-2.9
19.2	6	17.857	-6.99	5	16.64	-2.9	2	20.83	8.51	2	18.643	-2.9
38.4	2	41.667	8.51	2	37.29	-2.9	1	-	-	1	-	-
57.6	1	62.5	8.51	1	55.93	-2.9	0	62.5	8.51	0	55.93	-2.9
115.2	0	125	8.51	0	111.86	-2.9	-	-	-	-	-	-

BRGH=0 时的波特率和误差

波特率 Kbps	BRGH=1											
	f <sub>SYS</sub> =8M			f <sub>SYS</sub> =7.159M			f <sub>SYS</sub> =4M			f <sub>SYS</sub> =3.579545M		
	BRG	Kbps	Err	BRG	Kbps	Err	BRG	Kbps	Err	BRG	Kbps	Err
0.3	-	-	-	-	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	207	1.202	0.16	185	1.203	0.23
2.4	207	2.404	0.16	185	2.405	0.23	103	2.404	0.16	92	2.406	0.23
4.8	103	4.808	0.16	92	4.811	0.23	51	4.808	0.16	46	4.76	-0.83
9.6	51	9.615	0.16	46	9.520	-0.832	25	9.615	0.16	22	9.727	1.32
19.2	25	19.231	0.16	22	19.454	1.32	12	19.231	0.16	11	18.643	-2.9
38.4	12	38.462	0.16	11	37.287	-2.9	6	35.714	-6.99	5	37.286	-2.9
57.6	8	55.556	-3.55	7	55.93	-2.9	3	62.5	8.51	3	55.930	-2.9
115.2	3	125	8.51	3	111.86	-2.9	1	125	8.51	1	111.86	-2.9
250	1	250	0	-	-	-	0	250	0	-	-	-

BRGH=1 时的波特率和误差

## UART 设置与控制

### → 简介

UART 采用标准的不归零码传输数据。它由 1 位起初位，8 位或 9 位数据位和 1 位或者两位停止位组成。奇偶校验是由硬件自动完成的，可设置成奇校验、偶校验和无校验三种格式。常用的数据传输格式由 8 位数据位，1 位停止位，无校验组成，用 8, N, 1 表示，它是系统上电的默认格式。数据位数、停止位数和奇偶校验由 UCR1 寄存器的 BNO、PRT、PREN 和 STOPS 设定。用于数据发送和接收的波特率由一个内部的 8 位定时器产生，数据传输时低位在前高位在后。尽管 UART 发送器和接收器在功能上相互独立，但它们使用相同的数据传输格式和波特率，在任何情况下，停止位是必须的。

### → UART 的使能和除能

UART 是由 UCR1 寄存器的 UARTEN 位来使能和除能的。它的发送引脚 TX 和接收引脚 RX 分别与普通 IO 口复用，UARTEN 位的一个基本功能就是控制这两个引脚。若 UARTEN、TXEN 和 RXEN 都置 1，则这两个引脚分别为 UART 的发送端口和接收端口，而不能作为普通的输入输出端口使用。若没有数据发送，TX 引脚默认状态为高电平。

UARTEN 清零将除能 TX 和 RX，使 PC0 和 PC1 可作为普通的输入输出端口使用。当 UART 被除能将清除缓冲器，所有缓冲器中的数据将被忽略，另外错误和状态标志位被复位，TXEN、RXEN、TXBRK、RXIF、OERR、FERR 和 NF 清零而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2 和 BRG 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零，所有发送和接收将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

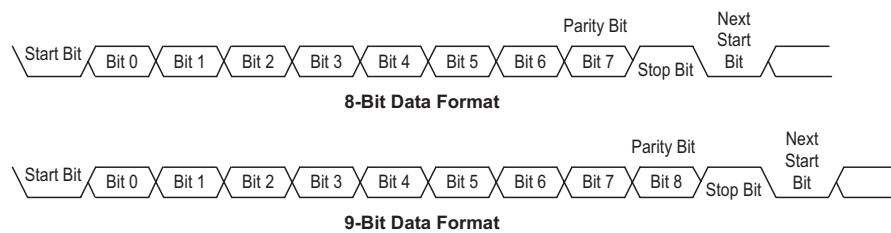
### → 数据位、停止位位数以及奇偶校验的选择

数据传输格式由数据长度、是否校验、校验类型地址表明位以及停止位长度组成。它们都是由 UCR1 寄存器的各个位控制的。BNO 决定数传输是 8 位还是 9 位；PRT 决定校验类型；PRTEN 决定时是否选择奇偶校验；而 STOPS 决定选用 1 位还是 2 位停止位。下表列出了各种数据传输格式。地址表明位用来确定此帧是否为地址。

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bits
8 位数据位				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
9 位数据位				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

### 发送和接收数据格式

下图是传输 8 位和 9 位数据的波形。



### UART 发送器

UCR1 寄存器的 BNO 位是控制数据传输的长度。BNO=1 其长度为 9 位，第 9 位 MSB 存储在 UCR1 寄存器的 TX8 中。发送器的核心是发送移位寄存器 TSR，它的数据由发送寄存器 TXR 提供，应用程序只须将发送数据写入 TXR 寄存器。上组数据的停止位发出前，TSR 寄存器禁止写入。如果还有新的数据要发送，一旦停止位发出，待发数据将会从 TXR 寄存器加载到 TSR 寄存器。TSR 不像其它寄存器一样映射在数据存储器，所以应用程序不能对其进行读写操作。TXEN=1，发送使能，但若 TXR 寄存器没有数据或者波特率没有设置，发送器将不会工作。先写 TXR 寄存器再 TXEN 也会触发发送。当发送器使能，若 TSR 寄存器为空，数据写入 TXR 寄存器将会直接加载到 TSR 寄存器中。发送器工作时，TXEN 清零，发送器将立刻停止工作并且复位，此时 TX 引脚可作为普通的输入输出使用。

### → 发送数据

当 UART 发送数据时，数据从移位寄存器中移到 TX 引脚上，其低位在前高位在后。在发送模式中，TXR 寄存器在内部总线和发送移位寄存器间形成一个缓冲。如果选择 9 位数据传输格式，最高位 MSB 存储在 UCR1 寄存器的 TX8 中。发送器初始化可由如下步骤完成：

- 正确地设置 BNO、PRT、PREN 和 STOPS 位以确定数据长度、校验类型和停止位长度。
- 设置 BRG 寄存器，选择期的波特率。
- 置高 TXEN，使引脚作为 UART 的发送端而非普通的输入输出端口。
- 读取 USR 寄存器，然后将待发数据写入 TXR 寄存器，此步骤会清除 TXIF 标志位。
- 如果要发送多个数据只需重复上一步骤。

当 TXIF=0 时，数据将禁止写入 TXR 寄存器。可以通过以下步骤来清除 TXIF：

1. 读取 USR 寄存器
2. 写 TXR 寄存器

只读标志位 TXIF 由 UART 硬件置位。若 TXIF=1，TXR 寄存器为空，其它数据可以写入而不会覆盖以前的数据。若 TEIE=1，TXIF 标志位会影响中断。在数据传输时，写 TXR 指令会将待发数据暂存在 TXR 寄存器中，当前数据发送完毕后，待发数据被加载到发送移位寄存器中。当发送器空闲时，写 TXR 指令会将数据直接加载到 TSR 寄存器中，数据传输立刻开始且 TXIF 置位。当一帧数据发送完毕，TIDLE 将被置位。可以通过以下步骤来清除 TIDLE：

1. 读取 USR 寄存器
2. 写 TXR 寄存器

清除 TXIF 和 TIDLE 软件执行次序相同。

### → 发送暂停字

若 TXBRK=1，下一帧将会发送暂停字。它是同一个起始位、13\*N(N=1, 2, ……) 位逻辑 0 以及停止位组成。置位 TXBRK 将会发送暂停字，而清除 TXBRK 产生停止位，传输暂停字不会产生中断。需要注意的是，暂停字至少 13 位宽。

若 TXBRK 持续为高，那么发送器会一直发送暂停字；当应用程序清除了 TXBRK，发送器将传输最后一帧暂停字再加上一位或者两位停止位。暂停字后的高电平保证下一帧数据起始位的检测。

## UART 接收器

### → 简介

UART 接收器支持 8 位或者 9 位数据接收。若 BNO=1，数据长度为 9 位，而最高位 MSB 存放在 UCR1 寄存器的 RX8 中。接收器的核心是串行移位寄存器 RSR。RX 引脚上的数据送入数据恢复器中，它在 16 倍波特率的频率下工作，而串行移位器工作在正常波特率下。当在 RX 引脚上检测到停止位，数据从 RSR 寄存器中加载到 RXR 寄存器。RX 引脚上的每一位数据会被采样三次以判断其逻辑状态。RSR 不像其它寄存器一样映射在数据存储器，所以应用程序不能对其进行读写操作。

### → 接收数据

当 UART 接收数据时，数据低位在前高位在后，连续地从 RX 引脚进入。RXR 寄存器在内部总线和接收移位寄存器间形成一个缓冲。RXR 寄存器是一个两层的 FIFO 缓冲器，它能保存两帧数据的同时接收第三帧数据，应用程序必须保证在接收完第三帧前读取 RXR 寄存器，否则忽略第三帧数据并且发生过速错误。接收器的初始化可由如下步骤完成：

- 正确地设置 BNO、PRT、PREN 和 STOPS 位以确定数据长度、校验类型和停止位长度。
- 设置 BRG 寄存器，选择期的波特率。
- 置高 TXEN，使引脚作为 UART 的发送端而非普通的输入输出端口。

此时接收器被使能并检测起始位。

接收数据将会发生如下事件：

- 当 RXR 寄存器中有一帧以上的数据时，USR 寄存器中的 RXIF 位将会置位。
- 若 RIE=1，数据从 RSR 寄存器加载到 RXR 寄存器中将产生中断。
- 若接收器检测到帧错误、噪声干扰、奇偶出错或过速错误，那么相应的错误标志位置位。

可以通过如下步骤来清除 RXIF:

1. 读取 USR 寄存器
2. 读取 RXR 寄存器

#### → 接收暂停字

UART 接收任何暂停字都会当作帧错误处理。接收器只根据 BNO 和 STOPS 位确定一帧数据的长度。若暂停字数大于 BNO 和 STOPS 位指定的长度，接收器认为接收已完结，RXIF 和 FERR 置位，RXR 寄存器清 0，若相应的中断允许且 RIDLE 为高将会产生中断。若暂停字较长，接收器收到起始位、数据位将会置位 FERR 标志，且在下一起始位前必须检测到有效的停止位。暂停字只会被认为包含信息 0 且会置位 FERR 标志。暂停字将会加载到缓冲器中，在接收到停止位前不会再接收数据，没有检测到停止位也会置位只读标志位 RIDLE。UART 接收到暂停字会产生以下事件:

- 帧错误标志位 FERR 置位。
- RXR 寄存器清零。
- OERR、NF、PERR、RIDLE 或 RXIF 可能会置位。

#### → 空闲状态

当 UART 接收数据时，即在起初位和停止位之间，USR 寄存器的接收标志位 RIDLE 清零。在停止位和下一帧数据的起始位之间，RIDLE 被置位，表示接收器空闲。

#### → 接收中断

USR 寄存器的只读标志位 RXIF 由接收器的边缘触发置位。若 RIE=1，数据从移位寄存器 RSR 加载到 RXR 寄存器时产生中断，同样地，过速也会产生中断。

#### 接收错误处理

UART 会产生几种接收错误，下面部分将描述各错误以及怎样处理。

#### → 过速——OERR 标志

RXR 寄存器是一个两层的 FIFO 缓冲器，它能保存两帧数据的同时接收第三帧数据，应用程序必须保证在接收完第三帧前读取 RXR 寄存器，否则发生过速错误。

产生过速错误时将会发生以下事件：

- USR 寄存器中 OERR 被置位。
- RXR 寄存器中数据不会丢失。
- RSR 寄存器数据将会被覆盖。
- 若 RIE=1，将会产生中断。

→ **噪声干扰——NF 标志**

数据恢复时多次采样可以有效的鉴别出噪声干扰。当检测到数据受到噪声干扰时将会发生以下事件：

- 在 RXIF 上升沿，USR 寄存器中只读标志位 NF 置位。
- 数据从 RSR 寄存器加载到 RXR 寄存器中。
- 不产生中断，此位置位的同时由 RXIF 请求中断。
- 先读取 USR 寄存器再读取 RXR 寄存器将复位 NF。

→ **帧错误——FERR 标志**

若在停止位上检测到 0，USR 寄存器中只读标志 FERR 置位。若选择两位停止位，两停止都必须为高，否则将置位 FERR。它同数据一起存储在缓冲器中，可被任何复位清除。

→ **奇偶校验错误——PERR 标志**

若接收到数据出现奇偶校验错误，USR 寄存器中只读标志 PERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。它同数据一起存储在缓冲器中，可被任何复位清除。注意，FERR 和 PERR 与相应的数据一起存储在缓冲器中，在读取数据之前必须先访问错误标志位。

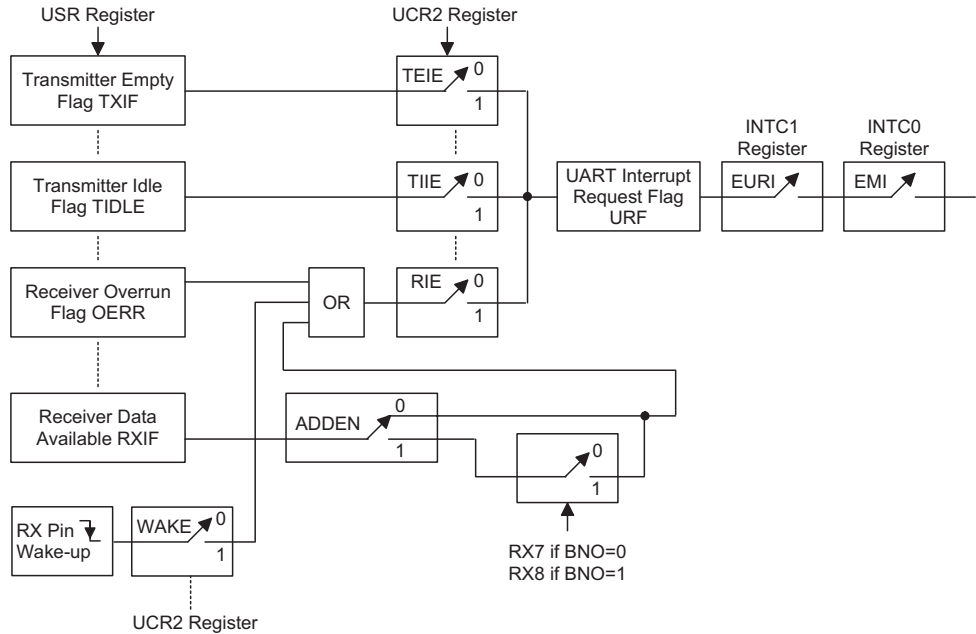
### 接收中断图解

UART 占用一个独立的中断向量。发送寄存器为空、发送器空闲、接收到有效数据、过速和地址检测等情况都会产生中断。若 UART 中断允许且堆栈未满，程序将会跳转到相应的中断向量执行中断服务程序。其中四种，若 UCR2 寄存器中相应中断允许位被置位，USR 寄存器中标志位将会影响中断。发送器有两个相应的中断允许位而接收器共用一个中断允许位。这些允许位可用于禁止 UART 的各个中断源。

地址检测也是 UART 的中断源，它没有相应的标志位，若 UCR2 寄存器中



ADDEN=1，当检测到地址将会产生 UART 中断。RX 引脚唤醒也可以产生 UART 中断，它没有相应的标志位。当 UXR2 中 WAKE 的 RIE 位置位，RX 引脚上有下降沿可以唤醒单片机。由 RX 唤醒单片机，系统必须延时 1024 个系统时钟才能正常工作。



UART 中断框图

注意，USR 是只读寄存器，不能由应用程序直接清除或置位，当程序跳转到相应的中断服务程序也不能清除此位。UART 寄存器章节中介绍的某些动作执行时将会自动清除这些标志位。所有 UART 中断都可由 INTC1 寄存器中 EURI 使能或除能。

### 地址检测模式

置位 UCR2 寄存器中的 ADDEN 将启动地址检测模式。若 ADDEN 有效，只有在接收到数据最高位为 1 才会产生中断，中断允许位 EURI 和 EMI 也要使能才会产生中断。地址的最高位为第 9 位 (BNO=1) 或第 8 位 (BNO=0)，若此位为高，则接收到的是地址而非数据。只有接收的数据的最后位为高才会产生中断。若 ADDEN 除能，每接收到一个有效数据便会置位 RXIF，而不用考虑数据的最后一位。地址检测和奇偶校验在功能上相互排斥，若地址检测模式使能，必须保证操作的正确，同时必须除能奇偶校验。

ADDEN	位 9 (BN0=1), 位 8 (BN0=0)	产生 UART 中断
0	0	√
	1	√
1	0	×
	1	√

ADDEN 功能

### 暂停模式下的 UART 功能

当 MCU 进入暂停模式，UART 将停止工作。当 MCU 进入暂停模式，模块的所有时钟关闭。当 UART 传送数据时，MCU 进入暂停模式，发送将停止并且 TX 引脚保持高电平。同样地，当 MCU 接收数据时进入暂停模式，数据接收也会停止。当 MCU 进入暂停模式，USR、UCR1 和 UCR2、接收发送寄存器、BRG 寄存器都不会受到影响。

RX 引脚具有唤醒功能，由 UCR2 寄存器中 WAKE 位控制。进入暂停模式前，若 WAKE、UARTEN、RXEN 和 RIE 都置位，则 RX 引脚的下降沿可唤醒单片机。唤醒后系统需延时 1024 个系统时钟才能正常工作，此时 RX 引脚上的任何数据将被忽略。

若要唤醒并产生 UART 中断，全局中断允许 EMI 位和 UART 中断允许 EURI 位也必须置位；若 EURI 或 EMI 为 0，单片机可以被唤醒但不会产生中断。同样唤醒后系统需延时 1024 个系统时钟才能正常工作，然后才会产生 UART 中断。

### UART 应用范例

下面的程序将显示如何通过 UART 发送和接收数据：

```

t_uart_TX:
    clr intc0                ; disable INTC0
    clr intc1                ; disable INTC1
    mov a,80h
    mov ucr1,a               ; enable UARTEN
    mov brg,a                ; set BRG=80H
    mov ucr2,a               ; enable TXEN
    mov a,055h
    mov txr,a                ; set TXR=55H
    :
    :

```

```

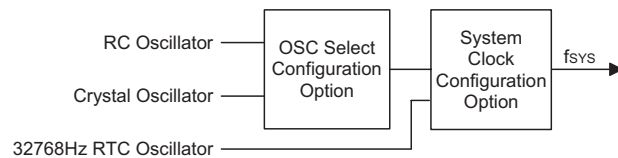
        jmp t_uart_TX
t_uart_RX:
        clr intc0           ; disable INTC0
        clr intc1           ; disable INTC1
        mov a,80h
        mov ucr1,a         ; enable UARTEN
        mov brg,a         ; set BRG=80H
        mov a,40h
        mov ucr2,a         ; enable RXEN
        mov a,rxr
        mov pa,a           ; pa=RXR
        :
        :
        jmp t_uart_RX
    
```

## 振荡器

不同的振荡器选择可以让使用者在不同的应用需求中获得更多范围的功能。LCD 系列单片机除了系统振荡器外，还含有看门狗振荡器。有三种系统时钟可供选择，看门狗定时器也有多种时钟源选择，提供用户更大的灵活性。此外，连接到这些振荡器的内部电路也提供了多种定时和中断功能，其频率和操作可通过掩膜选项和内部寄存器决定。

### 系统时钟配置

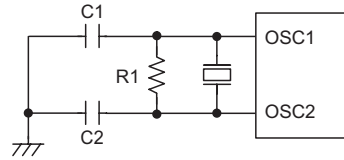
有三种方法产生系统时钟：使用外部晶体/陶瓷振荡器、外部 RC 电路或外部 RTC 32768Hz 的晶体产生实时时钟。可以通过掩膜选项来选择。



系统时钟配置

### 系统晶体/陶瓷振荡器

对于晶体振荡器的结构配置，晶体只要简单的连接至 OSC1 和 OSC2，则会产生所需的相移及反馈。为了确保某些低频率的晶体振荡和所有陶瓷共振器的应用，建议使用两个小电容和电阻，具体数值如表所示，请如下图方式连接。



晶体/陶瓷振荡器

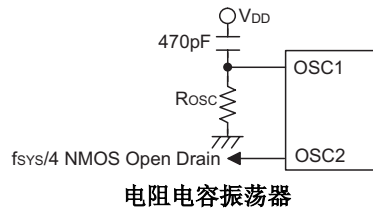
下表表示对于几种晶体/陶瓷振荡器频率下的典型的 C1、C2 和 R1 的值。

晶体或共振器	C1, C2	R1
4MHz 晶体	0pF	10kΩ
4MHz 共振器	10pF	12kΩ
3.58MHz 晶体	0pF	10kΩ
3.58MHz 共振器	25pF	10kΩ
2MHz 晶体与共振器	25pF	10kΩ
1MHz 晶体	35pF	27kΩ
480kHz 共振器	300pF	9.1kΩ
455kHz 共振器	300pF	10kΩ
429kHz 共振器	300pF	10kΩ

R1 的作用是在低电压的时候确保关闭振荡，此低电压值低于单片机的最低工作电压。需要注意的是如果 LVR 使能，可以不加 R1。

### 系统电阻电容振荡器

使用外部 RC 电路作为系统振荡器，需要在 OSC1 和 GND 之间连接一个电阻。对于 HT49C30L, HT49C50L 和 HT49C70L，其阻值约在 560kΩ到 1MΩ，而连接在其它单片机上的阻值约在 30kΩ到 750kΩ。产生的系统时钟 4 分频后提供给 OSC2 作输出，以达到与外部同步化的目的。虽然此振荡器配置成本较低，但振荡频率会因 VDD、温度和芯片本身的制成而改变，因此不适合用来做计时严格或需要精确振荡器频率的场合。对于外部电阻 R<sub>OSC</sub> 的阻值，请参考附录章节中典型 RC 振荡器对温度以及对 V<sub>DD</sub> 特性曲线分析。



**注意：** 内部电容和外部电阻  $R_{osc}$  共同作用决定频率值。图中显示的外部电容并不会影响振荡器的频率值。如果应用电路中用到 OSC2 的 open-drain 输出，则应该加上这个电容以改善振荡器的稳定性。

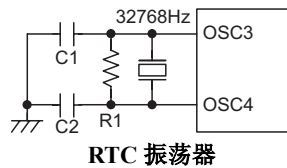
### RTC 振荡器

当单片机进入暂停即 HALT 状态，内部时钟通常被关闭以停止单片机的运作以及节省能源。然而在一些单片机的应用中，在 HALT 状态下也必须要维持某些内部功能，如定时器的运行。为了提供这特性，所有盛群 LCD 系列单片机都加上了一个 RTC 振荡器，它可以在任何时刻保持振荡，甚至是单片机进入 HALT 模式。这个时钟源具有固定的频率值 32768Hz，并要求在 OSC3 和 OSC4 引脚间连接一个 32768Hz 的晶体。

RTC 振荡器可以和连接在 OSC1 和 OSC2 引脚上的 RC 或晶体系统振荡器联合使用。同时，它也可以单独使用以实现单片机的所有功能，包括系统振荡器（这时 OSC1 和 OSC2 引脚不使用）。由于这个振荡器的频率很低，即使在暂停模式下仍然可以振荡，整体的功耗也相对很低。RTC 振荡器的功能通过掩膜选项和 RTCC 寄存器中的 QOSC 位控制。

当 RTC 振荡器不做系统振荡器时，必须要在 OSC1 和 OSC2 引脚上连接 RC 或晶体系统振荡器。当单片机进入 HALT 状态，较高频率的系统时钟将停止振荡以减少功耗，但 RTC 时钟仍保持振荡。此时 RTC 振荡器的主要功能是保持一些内部时钟功能的运行，如看门狗定时器、内部 RTC 定时器以及时基功能的运行。然而如果 RTC 振荡器通过掩膜选项的选择作为系统振荡器使用，就不再需要其它振荡器，引脚 OSC1 和 OSC2 也接空。当单片机进入暂停模式，RTC 振荡器将保持振荡以维持内部定时器功能，但不执行指令以节省功耗。请注意，此时的系统时钟只有 32768Hz，比一般的系统时钟速度要低很多。

如果掩膜选项选择 RTC 振荡器作为系统时钟，则内部时钟源  $f_s$ ，以及看门狗定时器，内部 RTC 中断和时基功能也必须使用 RTC 振荡器作为它们的时钟源。



**注意：** 在提供振荡的 32768Hz 晶体上并不需要连接外部电阻和电容。而在需要精确 RTC 频率的应用中，由于晶体有着不同的制造公差，可以加上这些元件作为频率补偿。有些应用场合只需要电容 C1。

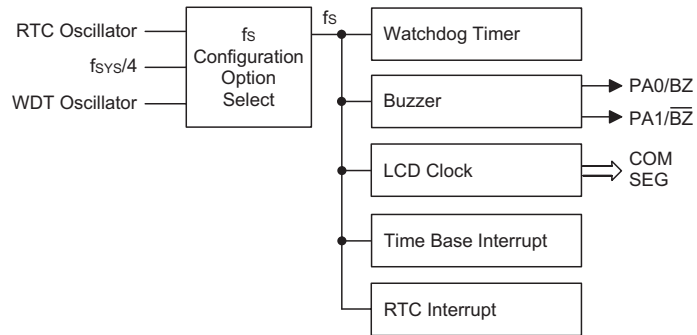
在上电时，RTC 振荡器有一个等待起振的时间延迟，为了减少这个延迟的时间，RTCC 寄存器的第 4 位(QOSC)提供快速起振功能。在上电时，这个位被清为“0”以实现 RTC 振荡器快速起振功能。由于快速起振功能有额外的功耗，为了减少这种功耗，建议在上电约 2 秒后，在应用程序中将 QOSC 位置为“1”。请注意，无论 QOSC 位如何设置，RTC 振荡器仍保持正常功能，只是快速起振会有较大的功耗。

### 看门狗定时振荡器

WDT 振荡器是一个完全独立在芯片上且自由动作的振荡器，它在 5V 时的周期时间典型值为 65 $\mu$ s 且不需外部的器件搭配。它可以通过掩膜选项成为内部时钟源  $f_S$ 。当单片机进入暂停模式时，系统时钟将停止动作，但 WDT 振荡器继续自由动作且保持  $f_S$  内部时钟源有效。通过这种方式，所有使用  $f_S$  作为时钟源的内部功能仍然有效，如看门狗定时器、RTC 中断、时基、LCD 时钟和蜂鸣器等。请注意，当单片机进入暂停模式时，如果看门狗定时振荡器仍然有效，则功耗将有微量的增加。如果  $f_S$  没有选为时钟源，则它的振荡功能将无效以节省功耗。

### 内部时钟源

内部时钟源  $f_S$  由内部产生并可作为其它内部功能的时钟源，如看门狗定时器、内部 RTC 中断、时基中断、LCD 时钟和蜂鸣器功能。 $f_S$  有三种不同的时钟源，可以来自 RTC 振荡器、看门狗定时振荡器或是系统振荡器的四分频，由  $f_S$  时钟源掩膜选项决定。请注意，如果选择 RTC 振荡器作为系统时钟  $f_{SYS}$ ，则  $f_S$  也将由 RTC 振荡器得到。



内部时钟源结构

系统时钟源	$f_s$ 时钟源选项
RTC 振荡器	RTC 振荡器
晶体振荡器	RTC 振荡器, WDT 振荡器或 $f_{SYS}/4$
电阻电容振荡器	RTC 振荡器, WDT 振荡器或 $f_{SYS}/4$

内部时钟源选项

## 暂停和唤醒

### 暂停

所有盛群单片机都具有暂停模式，也称为 HALT 或者睡眠模式。当芯片进入此模式，正常的工作电流降到很低，这是因为系统的晶振已停止振荡而降低了功耗。芯片维持现在状态直到被唤醒后继续运行。这一特性对电池供电系统特别重要。

### 进入暂停

暂停模式是通过“HALT”指令实现且造成如下结果：

- 系统振荡器将被关闭，应用程序停止在“HALT”指令处
- 在 RAM 芯片和寄存器上的内容保持不变
- 若 WDT 时钟源来自 WDT 振荡器，WDT 将被清零然后再重新计数。若 WDT 时钟来自系统振荡器，WDT 将停止计数
- 所有输入/输出端口保持不变

- PDF 标志位被置位而 TO 标志位被清零

### 静态电流

进入暂停模式可以极大的降低功耗,只有几个微安大小。若想将功耗降到最低,还需要考虑其它问题,特别是输入/输出端口。所有带上拉电阻的输入口必须接高或者接低,因为浮空的输入口会增加功耗。注意,芯片也会消耗电流以维持 Watchdog、RTC 和 LCD 驱动工作。

### 唤醒

当系统进入暂停可由以下情况唤醒:

- 外部复位
- PA 口下降沿
- 系统中断
- WDT 溢出
- RX 引脚的下降沿

如果系统由外部复位唤醒,芯片将完全复位。若由 WDT 溢出复位将初始化 WDT 计数器。尽管都会产生复位,但可以通过 TO 和 PDF 标志区分。系统上电或执行清除看门狗指令,PDF 被清除;执行 HALT 指令,PDF 被置位。TO 标志由 WDT 溢出置位,同时唤醒单片机,但只有程序计数器 Program Counter 和堆栈指针 SP 被复位,其它都保持原有状态。

若由 PA 口或在 HT49RU80/HT49CU80 中 UART 的 RX 引脚下降沿唤醒,单片机会执行 HALT 指令的下一条指令。所有 RAM 和寄存器都保持未执行 HALT 指令前的状态,程序将继续执行。掩膜选项可设置 PA 口每个引脚为单独的唤醒源。UART 接收和相应的唤醒位被使能, RX 引脚才可能作为唤醒源。

如果由中断唤醒,可能会发生两种情况:如果中断禁止或中断允许但堆栈已满,程序将会从下一条指令开始运行;如果中断允许且堆栈未滿,则会产生一般的中断响应。如果在进入暂停模式前,中断请求标志位已被置位,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,则实际中断子程序的执行会延时一个以上的周期。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。



## 低电压检测器 – LVD

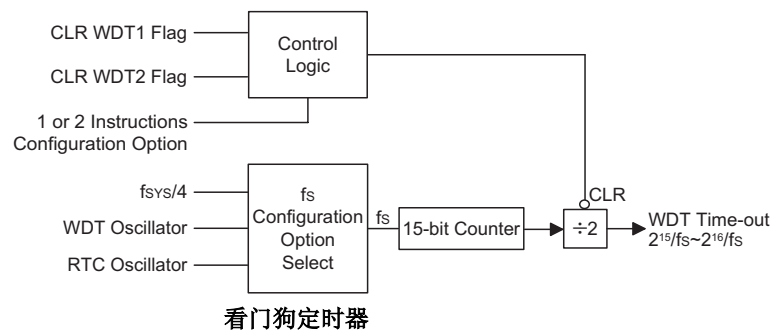
除 HT49C30L、HT49C50L 和 HT49C70L 外，其余 LCD 单片机都包含低电压检测功能。这个内部功能提供使用者一个监测的工具，警告使用者系统电源电压低于 DC 特性中所描述的固定值。

RTCC 寄存器的第 3 位和第 5 位用来控制 LVD 的整体功能。第 3 位 LVDC 是使能/除能控制位，当设定为逻辑低时，LVD 为除能状态。第 5 位 LVDO 是 LVD 检测器的输出位。在正常操作下，当电源电压高于 DC 特性中的 VLVD 值时，LVDO 位将保持“0”。一旦电源电压低于这个 VLVD 值，则 LVDO 位将变为“1”，表示低电压状态。请注意，LVDO 位是只读位，通过检测 RTCC 寄存器中的 LVDO 位，应用程序可以判断是否出现低电压状态。

在上电或复位后，通过清除 RTCC 寄存器中的 LVDC 位为“0”可将 LVD 关闭。请注意，如果 LVD 使能，会由于它的内部电路而增加一些功耗，可以通过清除 LVDC 位为“0”减少功耗。请不要混淆 LVR 和 LVD 的功能。在 LVR 功能中，单片机会产生自动复位，而在 LVD 功能中，只有 LVDO 位会受影响，不会影响单片机的其它功能。

## 看门狗定时器

看门狗定时器的功能在防止如电的干扰等外部不可控制事件，所造成的程序不正常动作或跳转到未知的地址。当 WDT 计数器溢出时，它产生一个“芯片复位”的动作。要注意的是假如 WDT 掩膜选项设为除能，则任何相关指令将无效。



看门狗定时器时钟源来自内部时钟源  $f_S$ 。此内部时钟  $f_S$  通过一个内部分频器，其分频比例被内部 15 位计数器固定且不能改变。因此实际看门狗定时器溢出时间由  $f_S$  使用的时钟源及内部固定的分频比例而决定。产生  $f_S$  的时钟源，同时也控制溢出时间，有三种不同的来源，RTC 振荡器、看门狗定时振荡器或系统振荡器的四分频，由掩膜选项决定选择哪种  $f_S$  的时钟源。请注意，如果选择 RTC 振荡器作为系统时钟，则  $f_S$  和看门狗定时器也将把 RTC 振荡器作为它们的时钟源。

系统时钟源	WDT 时钟源选项
RTC 振荡器	RTC 振荡器
晶体/陶瓷振荡器	RTC 振荡器, WDT 振荡器或 $f_{SYS}/4$
电阻电容振荡器	RTC 振荡器, WDT 振荡器或 $f_{SYS}/4$

#### 看门狗定时器时钟源选项

在 LCD 系列的单片机中没有与看门狗定时器相关的内部寄存器。内部看门狗定时振荡器是看门狗定时器的一个时钟源，它在 5V 电压下有接近 65 $\mu$ s 的周期。要注意的是这个内部时钟周期会因 VDD、温度和制成而变化。看门狗定时器时钟源的选项还有指令时钟(系统时钟的四分频  $f_{SYS}/4$ )和 RTC 振荡器。无论看门狗定时器的时钟源是来自它本身的内部 WDT 振荡器、指令时钟或 RTC 振荡器，它们都将通过一个内部的 15 位计数器来分频，以得到更长的看门狗溢出时间。对于 LCD 系列的单片机，此分频比例是固定的，看门狗定时器溢出时间的范围为  $2^{15}/f_S$  到  $2^{16}/f_S$ 。由于清除指令只复位计数链的最后一级，因此实际的分频系数和相应的 WDT 溢出时间会有因数为 2 的变化。实际的分频系数依据清除指令被执行前 WDT 计数器剩余的值。请注意，没有独立的内部寄存器或掩膜选项设置看门狗定时器的溢出时间，它完全由内部时钟源频率  $f_S$  决定。

如前所述，内部时钟源  $f_S$  和由此得到的看门狗定时器时钟源，来自于内部看门狗定时振荡器、系统时钟的四分频或 RTC 振荡器。如果使用  $f_{SYS}/4$  作为 WDT 时钟源，要注意的是，当系统进入暂停模式后，指令时钟会停止且 WDT 会失去其保护目的。在这种情况下，系统只能通过外部逻辑重新复位。当系统操作在干扰严重的环境时，建议使用内部 WDT 振荡器或 32kHz RTC 振荡器。

系统在正常运行状态下，WDT 溢出将导致“芯片复位”，且置位状态标志位“TO”。然而如果系统处于暂停模式，则只有一个从暂停模式来的 WDT 溢出

复位发生，它只复位程序计数器和 SP。有三种方法可以用来清除 WDT 的内容。第一种是外部硬件复位( $\overline{\text{RES}}$  引脚低电平)，第二种是通过软件指令，而第三种是通过“HALT”指令。使用软件指令有两种方法去清除看门狗寄存器，必须由掩膜选项选择。第一种选择是使用单一“CLR WDT”指令，而第二种是使用“CLR WDT1”和“CLR WDT2”两个指令。对于第一种选择，只要执行“CLR WDT”便清除 WDT。而第二种选择，必须交替执行“CLR WDT1”和“CLR WDT2”两者才能成功的清除 WDT。关于第二种选择要注意的是，如果“CLR WDT1”正被用来清除 WDT，接着再执行这条指令将是无效的，只有执行“CLR WDT2”指令才能清除 WDT。同样的“CLR WDT2”指令已经执行后，只有接着执行“CLR WDT1”指令才可以清除看门狗定时器。

## 蜂鸣器

与可编程分频器的作用相似，单片机中蜂鸣器的功能在于提供可变频率的输出，以适用于像压电蜂鸣器驱动或其它需要精确频率输出的场合。引脚 BZ 和  $\overline{\text{BZ}}$  形成互补对且和输入/输出引脚 PA0 和 PA1 共用。通过掩膜选项来选择 PA0 和 PA1 两个引脚是作为标准输入/输出引脚使用还是作为蜂鸣器引脚使用。要注意的是  $\overline{\text{BZ}}$  引脚是 BZ 引脚的反向输出，可以提供更多的功率差动输出到接口，如蜂鸣器。

蜂鸣器由内部时钟源  $f_s$  驱动，然后通过一个分频器，分频比例由掩膜选项选择，可以产生范围由  $f_s/2^2$  到  $f_s/2^9$  的蜂鸣器频率。产生  $f_s$  的时钟源有三种不同的来源，RTC 振荡器、看门狗定时振荡器或系统振荡器的四分频，这也同时控制蜂鸣器的频率。选择哪一种则是由  $f_s$  时钟源掩膜选项决定。请注意，如果选择

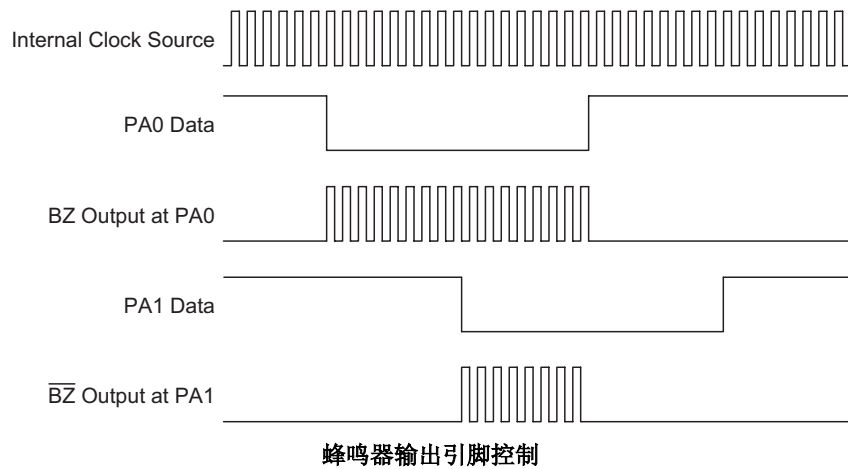
PA0/PA1 引脚功能控制

PA 寄存器 PA1 位	PA 寄存器 PA0 位	输出引脚状态 PA0/BZ, PA1/ $\overline{\text{BZ}}$
0	0	PA0=BZ PA1= $\overline{\text{BZ}}$
1	0	PA0=BZ PA1=0
Don't care	1	PA0=0 PA1=0

蜂鸣器输出引脚控制

RTC 振荡器作为系统时钟，则  $f_s$  和相应的蜂鸣器也将把 RTC 振荡器作为它们的时钟源。蜂鸣器的频率由掩膜选项确定，它可以选择内部时钟  $f_s$  的来源和分频比例，但不存在与蜂鸣器频率相关的内部寄存器。

为了实现蜂鸣器的输出，除了选择适当的掩膜选项外，要正确设置 PA 数据寄存器中的 PA0 和 PA1 数据位。为了让 BZ 和  $\overline{BZ}$  两个输出做为蜂鸣器的输出互补对，必须将 PA 数据寄存器中的 PA0 和 PA1 两个位清除为 0。如果只有 PA0 位被清除为 0 而 PA1 位被设为逻辑高，则只有 BZ 输出被操作为蜂鸣器输出，而  $\overline{BZ}$  输出则固定在逻辑 0。如果 PA0 位被设为逻辑高，则 BZ 和  $\overline{BZ}$  两个输出将固定在逻辑 0，与 PA1 位的设定无关。因此 PA 数据寄存器中的 PA0 位可做为 BZ 和  $\overline{BZ}$  输出的开启/关闭控制位。当掩膜选项选择引脚 PA0 和 PA1 以便 BZ 和  $\overline{BZ}$  能做为蜂鸣器输出时，引脚 PA0~PA3 将自动配置成 CMOS 形式。



## 掩膜选项

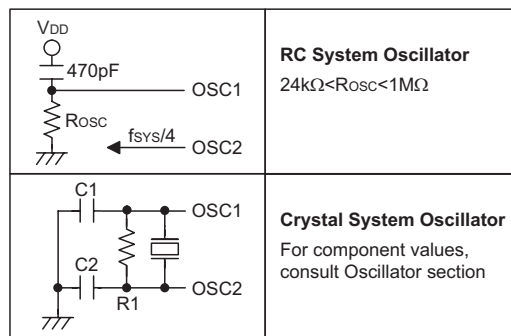
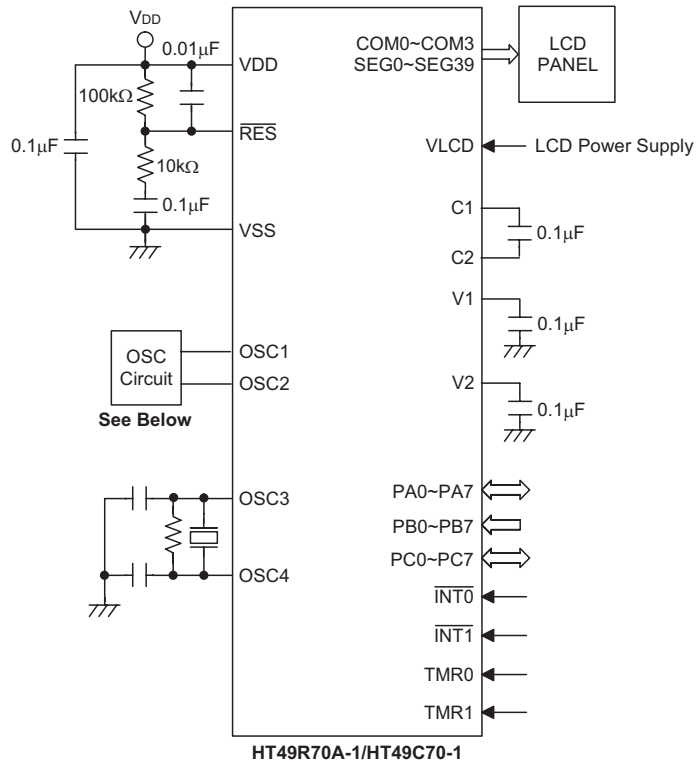
通过 HT-IDE 的软件介面，使用者可以选择掩膜选项。对于 OTP 版单片机，单片机掩膜选项会被储存在选项存储器。所有的位必须按照适合的系统功能去设定，具体内容可由下表得到。请注意，当使用者把掩膜选项烧入单片机之后，就不可以再被应用程序修改。对于掩膜版单片机，掩膜选项一经定义则会在工厂生产时制作完成，使用者不能再重新配置。

选项	
<b>I/O 选项</b>	
1	PA0~PA3: CMOS 或 NMOS 类型(半字节选项)
2	PA0~PA3: 上拉电阻的使能或除能(半字节选项)
3	PA0~PA7: 唤醒功能的使能或除能(位选项)
4	PC0~PC3: CMOS 或 NMOS 类型(半字节选项)
5	PC4~PC7: CMOS 或 NMOS 类型(半字节选项)
6	PC0~PC3: 上拉电阻的使能或除能(半字节选项)
7	PC4~PC7: 上拉电阻的使能或除能(半字节选项)
8	PD0~PD6 或 SEG40~SEG46(位选项) (只有 HT49RU80/HT49CU80 具有)
<b>LCD 选项</b>	
9	LCD 时钟: $f_S/2^2, f_S/2^3, f_S/2^4, f_S/2^5, f_S/2^6, f_S/2^7, f_S/2^8$
10	LCD 占空比: 1/2, 1/3, 1/4
11	LCD 偏压: 1/2, 1/3 (HT49C30L, HT49C50L, HT49C70L 只有 1/2 偏压)
12	LCD 偏压类型: R 型或 C 型(HT49C30L, HT49C50L, HT49C70L 只有 C 型)
13	HALT 时 LCD 的开/关: 使能或除能
<b>振荡器选项</b>	
14	OSC 类型选择: 电阻电容或晶体
15	$f_{SYS}$ 时钟源: OSC 或 RTC 振荡器
16	$f_S$ 内部时钟源: RTC 振荡器, WDT 振荡器或 $f_{SYS}/4$
<b>定时器选项</b>	
17	定时/计数器 0 时钟源: $f_{SYS}$ 或者 $f_{SYS}/4$
18	定时/计数器 1 时钟源: 系统时钟、时基中断信号或者 TMR0 溢出信号 (HT49R30A-1/HT49C30-1/HT49C30L 除外)
<b>PFD 选项</b>	
19	PA3: 标准 I/O 或 PFD 输出
20	PFD 时钟选择: TMR0 或 TMR1 (HT49R30A-1/HT49C30-1/HT49C30L 除外)
<b>蜂鸣器选项</b>	
21	PA0/PA1: 标准 I/O 或蜂鸣器功能
22	蜂鸣器频率: $f_S/2^2, f_S/2^3, f_S/2^4, f_S/2^5, f_S/2^6, f_S/2^7, f_S/2^8, f_S/2^9$
<b>时基选项</b>	
23	时基分频比例: $f_S/2^{12}, f_S/2^{13}, f_S/2^{14}, f_S/2^{15}$
<b>看门狗定时器选项</b>	
24	WDT 使能或除能
25	CLRWDT 指令条数: 1 或 2 条指令
<b>LVD/LVR 选项</b>	
26	低电压侦测: 使能或除能(HT49C30L, HT49C50L, HT49C70L 除外)
27	低电压复位功能: 使能或除能(HT49C30L, HT49C50L, HT49C70L 除外)

**注意:** 并不是每一款单片机都具有列出的所有 I/O 端口, 因此相应的, 有些列出的 I/O 选项在单片机中并不提供。

## 应用电路

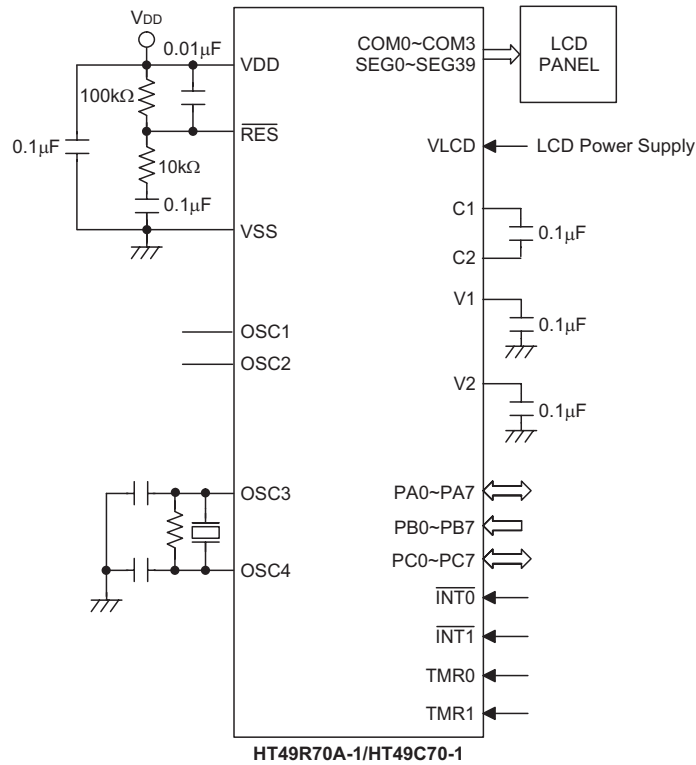
电阻电容或晶体系统振荡器和 RTC 振荡器  
(HT49C30L、HT49C50L、HT49C70L 除外)



OSC Circuit

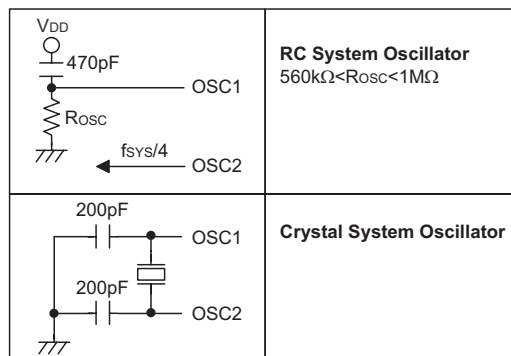
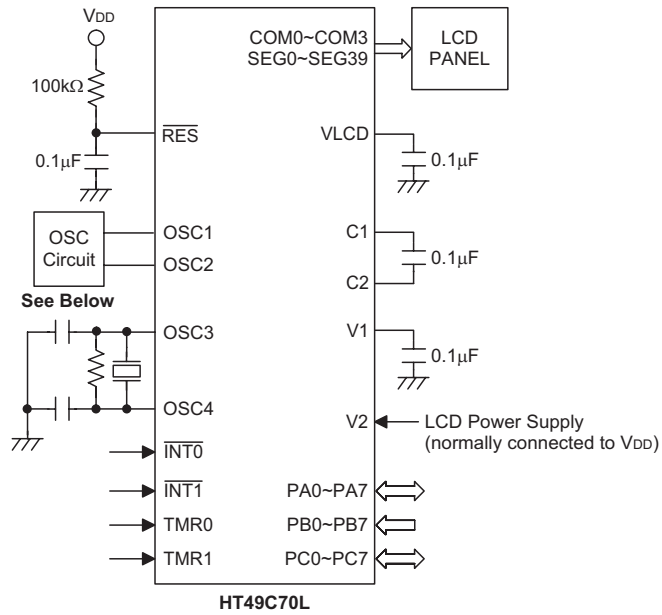
注意: 这个应用电路是以 HT49R70A-1/HT49C70-1 为例，它也可以应用在 HT49R30A-1/HT49C30-1, HT49R50A-1/HT49C50-1 和 HT49RU80/HT49CU80 上。

32768Hz 晶体振荡器用于单片机的所有功能  
(HT49C30L、HT49C50L、HT49C70L 除外)



注意: 这个应用电路是以 HT49R70A-1/HT49C70-1 为例，它也可以应用在 HT49R30A-1/HT49C30-1, HT49R50A-1/HT49C50-1 和 HT49RU80/HT49CU80 上。

电阻电容或晶体系统振荡器和 RTC 振荡器  
(对于 HT49C30L、HT49C50L、HT49C70L)

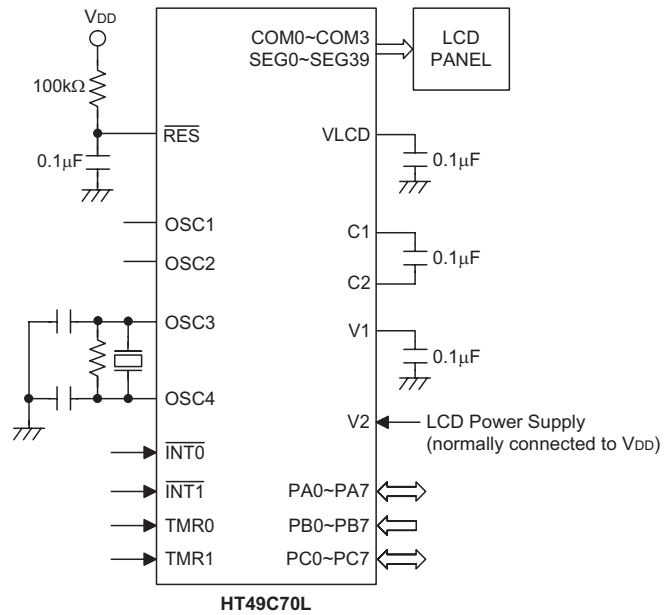


OSC Circuit

**注意:** 这个应用电路是以 HT49C70L 为例，它也可以应用在 HT49C30L 和 HT49C50L 上。



32768Hz 晶体振荡器用于单片机的所有功能  
(对于 HT49C30L、HT49C50L、HT49C70L)



**注意:** 这个应用电路是以 HT49C70L 为例，它也可以应用在 HT49C30L 和 HT49C50L 上。



## 第二部份

# 程序语言



## 第二章

# 指令集介绍

# 2

## 指令集

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

为了更加容易了解各式各样的指令码，接下来按功能分组介绍它们。

### 指令周期

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

### 数据的传送

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

### 算术运算

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

### 逻辑和移位运算

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

### 分支和控制的转换

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

### 位运算

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

### 查表运算

数据的储存通常由寄存器完成，然而当处理大量固定的数据时，它的存储量常常造成对个别存储器的不便。为了改善此问题，盛群单片机允许在程序存储器中设定一块数据可直接存储的区域，只需要一组简易的指令即可对数据进行查表。

### 其它运算

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

## 指令设定一览表

### 惯例

x: 立即数

m: 数据存储器地址

A: 累加器

i: 0~7 号位

addr: 程序存储器地址

助记符	指令简易描述	周期	影响标志位
<b>算术运算</b>			
ADD A, [m]	ACC 与数据存储器相加, 结果放入 ACC	1	Z,C,AC,OV
ADDMA, [m]	ACC 与数据存储器相加, 结果放入数据存储器	1 <sup>Note</sup>	Z,C,AC,OV
ADD A, x	ACC 与立即数相加, 结果放入 ACC	1	Z,C,AC,OV
ADC A, [m]	ACC 与数据存储器、进位标志相加, 结果放入 ACC	1	Z,C,AC,OV
ADCM A, [m]	ACC 与数据存储器、进位标志相加, 结果放入数据存储器	1 <sup>Note</sup>	Z,C,AC,OV
SUB A, x	ACC 与立即数相减, 结果放入 ACC	1	Z,C,AC,OV
SUB A, [m]	ACC 与数据存储器相减, 结果放入 ACC	1	Z,C,AC,OV
SUBMA, [m]	ACC 与数据存储器相减, 结果放入数据存储器	1 <sup>Note</sup>	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
SBCMA, [m]	ACC 与数据存储器、进位标志相减, 结果放入数据存储器	1 <sup>Note</sup>	Z,C,AC,OV
DAA [m]	将加法运算中放入 ACC 的值调整为十进制数, 并将结果放入数据存储器	1 <sup>Note</sup>	C



助记符	指令简易描述	周期	影响标志位
<b>逻辑运算</b>			
AND A, [m]	ACC 与数据存储器做“与”运算，结果放入 ACC	1	Z
OR A, [m]	ACC 与数据存储器做“或”运算，结果放入 ACC	1	Z
XOR A, [m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	1	Z
ANDM A, [m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	1 <sup>Note</sup>	Z
ORM A, [m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	1 <sup>Note</sup>	Z
XORM A, [m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	1 <sup>Note</sup>	Z
AND A, x	ACC 与立即数做“与”运算，结果放入 ACC	1	Z
OR A, x	ACC 与立即数做“或”运算，结果放入 ACC	1	Z
XOR A, x	ACC 与立即数做“异或”运算，结果放入 ACC	1	Z
CPL [m]	对数据存储器取反，结果放入数据存储器	1 <sup>Note</sup>	Z
CPLA [m]	对数据存储器取反，结果放入 ACC	1	Z
<b>递增和递减</b>			
INCA [m]	递增数据存储器，结果放入 ACC	1	Z
INC [m]	递增数据存储器，结果放入数据存储器	1 <sup>Note</sup>	Z
DECA [m]	递减数据存储器，结果放入 ACC	1	Z
DEC [m]	递减数据存储器，结果放入数据存储器	1 <sup>Note</sup>	Z
<b>移位</b>			
RRA [m]	数据存储器右移一位，结果放入 ACC	1	无
RR [m]	数据存储器右移一位，结果放入数据存储器	1 <sup>Note</sup>	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 <sup>Note</sup>	C
RLA [m]	数据存储器左移一位，结果放入 ACC	1	无
RL [m]	数据存储器左移一位，结果放入数据存储器	1 <sup>Note</sup>	无
RLCA [m]	带进位将数据存储器左移一位，结果放入 ACC	1	C
RLC [m]	带进位将数据存储器左移一位，结果放入数据存储器	1 <sup>Note</sup>	C
<b>数据传送</b>			
MOV A, [m]	将数据存储器送至 ACC	1	无
MOV [m], A	将 ACC 送至数据存储器	1 <sup>Note</sup>	无
MOV A, x	将立即数送至 ACC	1	无
<b>位运算</b>			
CLR [m].i	清除数据存储器的位	1 <sup>Note</sup>	无
SET [m].i	设置数据存储器的位	1 <sup>Note</sup>	无

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

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

## 第三章

## 指令定义

## 3

- ADC A, [m]** Add Data Memory to ACC with Carry  
指令说明 将指定数据存储器、累加器和进位标志位的内容相加后，把结果储存回累加器。  
功能表示  $ACC \leftarrow ACC + [m] + C$   
影响标志位 OV, Z, AC, C
- ADCM A, [m]** Add ACC to Data Memory with Carry  
指令说明 将指定数据存储器、累加器和进位标志位的内容相加后，把结果储存回指定数据存储器。  
功能表示  $[m] \leftarrow ACC + [m] + C$   
影响标志位 OV, Z, AC, C
- ADD A, [m]** Add Data Memory to ACC  
指令说明 将指定数据存储器内容和累加器的内容相加后，把结果储存回累加器。  
功能表示  $ACC \leftarrow ACC + [m]$   
影响标志位 OV, Z, AC, C
- ADD A, x** Add immediate data to ACC  
指令说明 将累加器和立即数的内容相加后，把结果储存回累加器。  
功能表示  $ACC \leftarrow ACC + x$   
影响标志位 OV, Z, AC, C

<b>ADDM A, [m]</b>	Add ACC to Data Memory
指令说明	将指定数据存储器 and 累加器的内容相加后，把结果储存回指定数据存储器。
功能表示	$[m] \leftarrow ACC + [m]$
影响标志位	OV, Z, AC, C
<b>AND A, [m]</b>	Logical AND Data Memory to ACC
指令说明	将存在累加器和指定数据存储器中的数据作 AND 的运算，然后把结果储存回累加器。
功能表示	$ACC \leftarrow ACC \text{ "AND" } [m]$
影响标志位	Z
<b>AND A, x</b>	Logical AND immediate data to ACC
指令说明	将存在累加器中的数据和立即数作 AND 的运算，然后把结果储存回累加器。
功能表示	$ACC \leftarrow ACC \text{ "AND" } x$
影响标志位	Z
<b>ANDM A, [m]</b>	Logical AND ACC to Data Memory
指令说明	将存在指定数据存储器 and 累加器中的数据作 AND 的运算，然后把结果储存回数据存储器。
功能表示	$[m] \leftarrow ACC \text{ "AND" } [m]$
影响标志位	Z
<b>CALL addr</b>	Subroutine call
指令说明	无条件地调用指定地址的子程序，此时程序计数器先加 1 获得下一个要执行的指令地址并压入堆栈，接着载入指定地址并从新地址继续执行程序，由于此指令需要额外的运算，所以为一个 2 周期的指令。
功能表示	$Stack \leftarrow Program Counter + 1$ $Program Counter \leftarrow addr$
影响标志位	None
<b>CLR [m]</b>	Clear Data Memory
指令说明	指定数据存储器中的每一位均清除为 0。
功能表示	$[m] \leftarrow 00H$
影响标志位	None

<b>CLR [m].i</b>	Clear bit of Data Memory
指令说明	指定数据存储器中的 i 位清除为 0。
功能表示	$[m].i \leftarrow 0$
影响标志位	None
<b>CLR WDT</b>	Clear Watchdog Timer
指令说明	将 TO、PDF 标志位和 WDT 全都清除。
功能表示	WDT cleared $TO \leftarrow 0$ $PDF \leftarrow 0$
影响标志位	TO, PDF
<b>CLR WDT1</b>	Pre-clear Watchdog Timer
指令说明	将 TO、PDF 标志位和 WDT 全都清除，请注意此指令要结合 CLR WDT2 一起动作且必须交替执行才有作用，重复执行此项指令而没有与 CLR WDT2 交替执行将无任何作用。
功能表示	WDT cleared $TO \leftarrow 0$ $PDF \leftarrow 0$
影响标志位	TO, PDF
<b>CLR WDT2</b>	Pre-clear Watchdog Timer
指令说明	将 TO、PDF 标志位和 WDT 全都清除，请注意此指令要结合 CLR WDT1 一起动作且必须交替执行才有作用，重复执行此项指令而没有与 CLR WDT1 交替执行将无任何作用。
功能表示	WDT cleared $TO \leftarrow 0$ $PDF \leftarrow 0$
影响标志位	TO, PDF
<b>CPL [m]</b>	Complement Data Memory
指令说明	将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1。
功能表示	$[m] \leftarrow \overline{[m]}$
影响标志位	Z

<b>CPLA [m]</b>	Complement Data Memory with result in ACC
指令说明	将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1，而结果被储存回累加器且数据存储器中的内容不变。
功能表示	$ACC \leftarrow \overline{[m]}$
影响标志位	Z
<b>DAA [m]</b>	Decimal-Adjust ACC for addition with result in Data Memory
指令说明	将存在累加器中的内容数值转换为 BCD（二进制转成十进制）数值，如果低 4 位大于 9 或 AC 标志位被设置，则在低 4 位加上一个 6，不然低 4 位的内容不变，如果高 4 位大于 9 或 C 标志位被设置，则在高 4 位加上一个 6，十进制的转换主要是依照累加器和标志位状况，分别加上 00H、06H、60H 或 66H，只有 C 标志位也许会被此指令影响，它会指出原始 BCD 数是否大于 100，并可以进行双精度十进制数相加。
功能表示	[m] $\leftarrow$ ACC + 00H 或 [m] $\leftarrow$ ACC + 06H 或 [m] $\leftarrow$ ACC + 60H 或 [m] $\leftarrow$ ACC + 66H
影响标志位	C
<b>DEC [m]</b>	Decrement Data Memory
指令说明	将在指定数据存储器内的数据减 1。
功能表示	$[m] \leftarrow [m] - 1$
影响标志位	Z
<b>DECA [m]</b>	Decrement Data Memory with result in ACC
指令说明	将在指定数据存储器内的数据减 1，把结果储存回累加器且数据存储器中的内容不变。
功能表示	$ACC \leftarrow [m] - 1$
影响标志位	Z

<b>HALT</b>	Enter power down mode
指令说明	此指令停止程序的执行并且关闭系统时钟，但数据存储器 and 寄存器的内容仍被保留，WDT 和预分频器(Prescaler)被清除，暂停标志位 PDF 被设置且 WDT 溢出标志位 TO 被清除。
功能表示	$TO \leftarrow 0$ $PDF \leftarrow 1$
影响标志位	TO, PDF
<b>INC [m]</b>	Increment Data Memory
指令说明	将指定数据存储器内的数据加 1。
功能表示	$[m] \leftarrow [m] + 1$
影响标志位	Z
<b>INCA [m]</b>	Increment Data Memory with result in ACC
指令说明	将指定数据存储器内的数据加 1，把结果储存回累加器且数据存储器中的内容不变。
功能表示	$ACC \leftarrow [m] + 1$
影响标志位	Z
<b>JMP addr</b>	Jump unconditionally
指令说明	程序计数器的内容被指定地址所取代，程序由新地址继续执行，当新地址被载入时，必须插入一个空指令周期，所以此指令为 2 个周期的指令
功能表示	$Program\ Counter \leftarrow addr$
影响标志位	None
<b>MOV A, [m]</b>	Move Data Memory to ACC
指令说明	将指定数据存储器的内容复制到累加器中。
功能表示	$ACC \leftarrow [m]$
影响标志位	None
<b>MOV A, x</b>	Move immediate data to ACC
指令说明	将立即数载入至累加器中。
功能表示	$ACC \leftarrow x$
影响标志位	None

<b>MOV [m], A</b>	Move ACC to Data Memory
指令说明	将累加器的内容复制到指定数据存储器。
功能表示	$[m] \leftarrow ACC$
影响标志位	None
<b>NOP</b>	No operation
指令说明	空操作，接下来顺序执行下一条指令。
功能表示	No operation
影响标志位	None
<b>OR A, [m]</b>	Logical OR Data Memory to ACC
指令说明	将存在累加器和指定数据存储器中的数据作 OR 的运算，然后把结果储存回累加器。
功能表示	$ACC \leftarrow ACC \text{ "OR" } [m]$
影响标志位	Z
<b>OR A, x</b>	Logical OR immediate data to ACC
指令说明	将存在累加器中的数据和立即数作 OR 的运算，然后把结果储存回累加器。
功能表示	$ACC \leftarrow ACC \text{ "OR" } x$
影响标志位	Z
<b>ORM A, [m]</b>	Logical OR ACC to Data Memory
指令说明	将存在指定数据存储器 and 累加器中的数据作 OR 的运算，然后把结果储存回数据存储器。
功能表示	$[m] \leftarrow ACC \text{ "OR" } [m]$
影响标志位	Z
<b>RET</b>	Return from subroutine
指令说明	将堆栈区的数据取回至程序计数器，程序由取回的地址继续执行。
功能表示	$Program\ Counter \leftarrow Stack$
影响标志位	None



<b>RET A, x</b>	Return from subroutine and load immediate data to ACC
指令说明	将堆栈区的数据取回至程序计数器且累加器载入立即数，程序由取回的地址继续执行。
功能表示	Program Counter $\leftarrow$ Stack ACC $\leftarrow$ x
影响标志位	None
<b>RETI</b>	Return from interrupt
指令说明	将堆栈区的数据取回至程序计数器且中断功能通过 EMI 位重新被使能，EMI 是控制中断使能的主中断位(寄存器 INTC 的第 0 位),如果在执行 RETI 指令之前还有中断未被响应，则这个中断将在返回主程序之前被响应。
功能表示	Program Counter $\leftarrow$ Stack EMI $\leftarrow$ 1
影响标志位	None
<b>RL [m]</b>	Rotate Data Memory left
指令说明	将指定数据存储器的内容向左移 1 个位，且第 7 位移回第 0 位。
功能表示	[m].(i+1) $\leftarrow$ [m].i; (i = 0~6) [m].0 $\leftarrow$ [m].7
影响标志位	None
<b>RLA [m]</b>	Rotate Data Memory left with result in ACC
指令说明	将指定数据存储器的内容向左移 1 个位，且第 7 位移回第 0 位，而移位的结果储存回累加器且数据存储器中的内容不变。
功能表示	ACC.(i+1) $\leftarrow$ [m].i; (i = 0~6) ACC.0 $\leftarrow$ [m].7
影响标志位	None
<b>RLC [m]</b>	Rotate Data Memory Left through Carry
指令说明	将指定数据存储器的内容连同进位标志位向左移 1 个位，第 7 位取代进位位且原本的进位标志位移至第 0 位。
功能表示	[m].(i+1) $\leftarrow$ [m].i; (i = 0~6) [m].0 $\leftarrow$ C C $\leftarrow$ [m].7
影响标志位	C

<b>RLCA [m]</b>	Rotate Data Memory left through Carry with result in ACC
指令说明	将指定数据存储器的内容连同进位标志位向左移 1 个位, 第 7 位取代进位位且原本的进位标志位移至第 0 位, 而移位的结果储存回累加器且数据存储器中的内容不变。
功能表示	$ACC.(i+1) \leftarrow [m].i; (i = 0\sim 6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位	C
<b>RR [m]</b>	Rotate Data Memory right
指令说明	将指定数据存储器的内容向右移 1 个位, 且第 0 位移回第 7 位。
功能表示	$[m].i \leftarrow [m].(i+1); (i = 0\sim 6)$ $[m].7 \leftarrow [m].0$
影响标志位	None
<b>RRA [m]</b>	Rotate Data Memory right with result in ACC
指令说明	将指定数据存储器的内容向右移 1 个位, 且第 0 位移回第 7 位, 而移位的结果储存回累加器且数据存储器中的内容不变。
功能表示	$ACC.i \leftarrow [m].(i+1); (i = 0\sim 6)$ $ACC.7 \leftarrow [m].0$
影响标志位	None
<b>RRC [m]</b>	Rotate Data Memory right through Carry
指令说明	将指定数据存储器的内容连同进位标志位向右移 1 个位, 第 0 位取代进位位且原本的进位标志位移至第 7 位。
功能表示	$[m].i \leftarrow [m].(i+1); (i = 0\sim 6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位	C

<b>RRCA [m]</b>	Rotate Data Memory right through Carry with result in ACC
指令说明	将指定数据存储器的内容连同进位标志位向右移 1 个位, 第 0 位取代进位位且原本的进位标志位移至第 7 位, 而移位的结果储存回累加器且数据存储器中的内容不变。
功能表示	$ACC.i \leftarrow [m].(i+1); (i = 0\sim 6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位	C
<b>SBC A, [m]</b>	Subtract Data Memory from ACC with Carry
指令说明	将累加器中的数据与指定数据存储器内容和进位标志位的反相减, 把结果储存回累加器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
功能表示	$ACC \leftarrow ACC - [m] - \bar{C}$
影响标志位	OV, Z, AC, C
<b>SBCM A, [m]</b>	Subtract Data Memory from ACC with Carry and result in Data Memory
指令说明	将累加器中的数据与指定数据存储器内容和进位标志位的反相减, 把结果储存回数据存储器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
功能表示	$[m] \leftarrow ACC - [m] - \bar{C}$
影响标志位	OV, Z, AC, C
<b>SDZ [m]</b>	Skip if Decrement Data Memory is 0
指令说明	将指定数据存储器的内容先减去 1 后, 如果结果为 0, 则程序计数器再加 1 跳过下一条指令, 由于取得下一指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行底下的指令。
功能表示	$[m] \leftarrow [m] - 1$ Skip if $[m] = 0$
影响标志位	None

<b>SDZA [m]</b>	Skip if decrement Data Memory is zero with result in ACC
指令说明	将指定数据存储器的内容先减去 1 后，如果结果为 0，则程序计数器再加 1 跳过下一条指令，此结果会被储存回累加器且指定存储器中的内容不变，由于取得下一指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行底下的指令。
功能表示	ACC ← [m] - 1 Skip if ACC = 0
影响标志位	None
<b>SET [m]</b>	Set Data Memory
指令说明	将指定数据存储器的每一个位设置为 1。
功能表示	[m] ← FFH
影响标志位	None
<b>SET [m].i</b>	Set bit of Data Memory
指令说明	将指定数据存储器的第 i 位设置为 1。
功能表示	[m].i ← 1
影响标志位	None
<b>SIZ [m]</b>	Skip if increment Data Memory is 0
指令说明	将指定数据存储器的内容先加上 1 后，如果结果为 0，则程序计数器再加 1 跳过下一条指令，由于取得下一指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行底下的指令。
功能表示	[m] ← [m] + 1 Skip if [m] = 0
影响标志位	None
<b>SIZA [m]</b>	Skip if increment Data Memory is zero with result in ACC
指令说明	将指定数据存储器的内容先加上 1 后，如果结果为 0，则程序计数器再加 1 跳过下一条指令，此结果会被储存回累加器且指定存储器中的内容不变，由于取得下一指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行底下的指令。
功能表示	ACC ← [m] + 1 Skip if ACC = 0
影响标志位	None

<b>SNZ [m].i</b>	Skip if bit i of Data Memory is not 0
指令说明	如果指定数据存储器的第 i 位不为 0，则程序计数器再加 1 跳过下一条指令，由于取得下一指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，程序继续执行底下的指令。
功能表示	Skip if [m].i $\neq$ 0
影响标志位	None
<b>SUB A, [m]</b>	Subtract Data Memory from ACC
指令说明	将累加器中内容减去指定数据存储器的数据，把结果储存回累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	ACC $\leftarrow$ ACC - [m]
影响标志位	OV, Z, AC, C
<b>SUBM A, [m]</b>	Subtract Data Memory from ACC with result in Data Memory
指令说明	将累加器中内容减去指定数据存储器的数据，把结果储存回数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	[m] $\leftarrow$ ACC - [m]
影响标志位	OV, Z, AC, C
<b>SUB A, x</b>	Subtract immediate Data from ACC
指令说明	将累加器中内容减去立即数，把结果储存回累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	ACC $\leftarrow$ ACC - x
影响标志位	OV, Z, AC, C
<b>SWAP [m]</b>	Swap nibbles of Data Memory
指令说明	将指定数据存储器的低 4 位与高 4 位互相交换。
功能表示	[m].3~[m].0 $\leftrightarrow$ [m].7~[m].4
影响标志位	None

<b>SWAPA [m]</b>	Swap nibbles of Data Memory with result in ACC
指令说明	将指定数据存储器的低 4 位与高 4 位互相交换, 然后把结果储存回累加器且数据存储器的内容不变。
功能表示	ACC.3~ACC.0 $\leftarrow$ [m].7~[m].4 ACC.7~ACC.4 $\leftarrow$ [m].3~[m].0
影响标志位	None
<b>SZ [m]</b>	Skip if Data Memory is 0
指令说明	如果指定数据存储器的内容为 0, 则程序计数器再加 1 跳过下一条指令, 由于取得下一指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 程序继续执行底下的指令。
功能表示	Skip if [m] = 0
影响标志位	None
<b>SZA [m]</b>	Skip if Data Memory is 0 with data movement to ACC
指令说明	将指定数据存储器的内容复制到累加器, 如果值为 0, 则程序计数器再加 1 跳过下一条指令, 由于取得下一指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 程序继续执行底下的指令。
功能表示	ACC $\leftarrow$ [m] Skip if [m] = 0
影响标志位	None
<b>SZ [m].i</b>	Skip if bit i of Data Memory is 0
指令说明	如果指定存储器第 i 位为 0, 则程序计数器再加 1 跳过下一条指令, 由于取得下一指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 程序继续执行底下的指令。
功能表示	Skip if [m].i = 0
影响标志位	None
<b>TABRDC [m]</b>	Read table (current page) to TBLH and Data Memory
指令说明	将表格指针 TBLP 所指的程序代码低字节(当前页)移至指定存储器且将高字节移至 TBLH。
功能表示	[m] $\leftarrow$ 程序代码(低字节) TBLH $\leftarrow$ 程序代码(高字节)
影响标志位	None

<b>TABRDL [m]</b>	Read table (last page) to TBLH and Data Memory
指令说明	将表格指针 TBLP 所指的程序代码低字节(最后一页)移至指定数据存储器且将高字节移至 TBLH。
指令说明	[m] ← 程序代码(低字节) TBLH ← 程序代码(高字节)
影响标志位	None
<b>XOR A, [m]</b>	Logical XOR Data Memory to ACC
指令说明	将存在累加器和指定数据存储器中的数据作 XOR 的运算，然后把结果储存回累加器。
功能表示	ACC ← ACC “XOR” [m]
影响标志位	Z
<b>XORM A, [m]</b>	Logical XOR ACC to Data Memory
指令说明	将存在指定数据存储器 and 累加器中的数据作 XOR 的运算，然后把结果储存回数据存储器。
功能表示	[m] ← ACC “XOR” [m]
影响标志位	Z
<b>XOR A, x</b>	Logical XOR immediate data to ACC
指令说明	将存在累加器中的数据和立即数作 XOR 的运算，然后把结果储存回累加器。
功能表示	ACC ← ACC “XOR” x
影响标志位	Z





## 第四章

# 汇编语言和编译器

# 4

源程序由汇编语言程序构成，由盛群编译器(Holtek Assembler)编译成目标文件(Object File)，再由连接器(Linker)连接并产生任务文件(Task File)。

源程序(source program)由语句(statement)和表格(look up table)组成，在编译器进行编译或程序执行时会给予指示，而语句是由助记符(mnemonic)、操作数(operand)和注解(comment)组成。

## 常用符号

下表描述了文章中所用到的常用符号

范例	描述
	中括号内的项目是可选择的，下列的命令行语法中：
[ <i>optional items</i> ]	<b>HASM</b> [ <i>options</i> ] <i>filename</i> [:]  <i>options</i> 和分号[:]都是可选的，而 <i>filename</i> 则必须设定。但是在指令操作数中的中括号则是指定存储器地址之用，必须要有。
{ <i>choice1</i>   <i>choice2</i> }	大括号和垂直线代表两个或更多的选项，大括号圈出这些选项而垂直线则用来分隔选项，只能有一个选项被选出。
	三个连续的点表示允许输入更多同样形式的数据，例如以下的指令格式：
Repeating elements...	<b>PUBLIC</b> <i>name1</i> [, <i>name2</i> [...]]  <i>name2</i> 之后的三个连续点表示允许输入更多的名称，只要每两个名称之间用逗号隔开即可。

## 语句语法

语句的语法格式如下：

[ 名称 ] [ 操作项 ] [ 操作数项 ] [ ;注解 ]

- 上述四个成员不一定都要指定。
- 每两个成员之间（除了注解）最少要以一个空格或一个 tab 符号分隔开。
- 成员的字型无大小写之分，换言之，编译器在编译之前会将小写字母改为大写字母。

### 名称

语句前可有标号以便于其它语句使用，如果名称当做标号使用，则必须在名称后紧接一个冒号(:)。名称由下列字符组成：

**A~Z a~z 0~9 ? \_ @**

在使用上有以下的限制：

- 不可使用数字 0~9 作为名称的第一个字符
- ?不能单独作为名称
- 只有最前面的 31 个字符被认可

### 操作项

操作项定义两种形态的语句，伪指令与指令。伪指令用来指导编译器如何在编译时产生目标码。指令则是引导单片机执行各种运算。两者都会在编译时产生目标码，目标码会在执行时指导单片机的运作。

### 操作数项

操作数项定义伪指令与指令所使用的数据，由符号、常数、表达式和寄存器所组成。

### 注解

注解是对程序代码的一种叙述与说明。编译器不会编译它。任何在分号之后的文字均被视为注解。

## 编译伪指令

编译伪指令用来指导编译器如何在编译时产生目标码。编译伪指令可以依其行为细分如下。

### 条件编译伪指令

条件区段的格式如下：

```
IF
statements
[ELSE
statements]
ENDIF
```

#### → 语法

```
IF expression
IFE expression
```

- 说明

伪指令 **IF** 和 **IFE** 对其后的 *expression* 进行检测。

如果 *expression* 的数值为真，换言之不为零，则在 **IF** 与 **ELSE** 或 **IF** 与 **ENDIF**（没有 **ELSE**）之间所有的语句会被编译。

如果 *expression* 的数值为假，换言之为零，则在 **IFE** 与 **ELSE** 或 **IFE** 与 **ENDIF**（没有 **ELSE**）之间所有的语句会被编译。

- 范例

```
IF debugcase
    ACC1 equ 5
    extern username: byte
ENDIF
```

在此范例中，如果符号 `debugcase` 的数值为真，也就是不为零，则变量 `ACC1` 的数值将被设定为 5 同时 `username` 被声明为外部变量。

→ 语法

**IFDEF** *name*

**IFNDEF** *name*

• 说明

**IFDEF** 和 **IFNDEF** 的差异在检测 *name* 是否被定义, 只要 *name* 已在前面定义为标号、变量或符号, 则在 **IFDEF** 与 **ENDIF** 之间的语句都会被编译, 相反如果 *name* 还未被定义, 则在 **IFNDEF** 与 **ENDIF** 之间的语句会被编译, 条件编译伪指令提供最多 7 层的嵌套。

• 范例

```
IFDEF          buf_flag
                buffer DB 20 dup (?)
ENDIF
```

在此范例中, 只要 *buf\_flag* 被事先定义, 即配置存储器空间给 *buffer*。

### 文件控制伪指令

→ 语法

**INCLUDE** *file-name*

或

**INCLUDE** "*file-name*"

• 说明

此伪指令会在编译时, 把包含入文件 *file-name* 的内容, 嵌入至当前的源程序文件, 并被视为源程序。编译器可提供最多 7 层的嵌套。

• 范例

```
INCLUDE      macro.def
```

在此范例中, 编译器把包含入文件 *macro.def* 内的源程序, 嵌入至当前的源程序文件。

→ 语法

**PAGE** *size*

• 说明

此伪指令指定程序列表文件(program listing file)中每一页的行数, 其范围介于 10 行至 255 行之间, 编译器的默认值为 60 行。

• 范例

```
PAGE 57
```

在此范例中, 程序列表文件的每一页最多为 57 行。

## → 语法

**.LIST**  
**.NOLIST**

## • 说明

伪指令 **.LIST** 和 **.NOLIST** 用来决定是否要将源程序行存储到程序列表文件 (program listing file)。**.NOLIST** 会禁止将其后的源程序存写到程序列表文件, 而 **.LIST** 则会将其后的源程序存写到程序列表文件。编译器的默认值为 **.LIST**。

## • 范例

```
.NOLIST
mov a, 1
mov b1, a
.LIST
```

上面的范例中, 被 **.NOLIST** 和 **.LIST** 所包围的两条指令将不会被存写到程序列表文件。

## → 语法

**.LISTMACRO**  
**.NOLISTMACRO**

## • 说明

伪指令 **.LISTMACRO** 会引导编译器将宏指令中包括注解的所有语句都存写到程序列表文件。伪指令 **.NOLISTMACRO** 则中止写入所有宏指令的语句。编译器的默认值为 **.NOLISTMACRO**。

## → 语法

**.LISTINCLUDE**  
**.NOLISTINCLUDE**

## • 说明

**.LISTINCLUDE** 会引导编译器将所有包含文件 (included files) 的内容写入程序列表文件中, **.NOLISTINCLUDE** 则会禁止编译器将其后的包含文件的内容写进程序列表文件, 编译器的默认值为 **.NOLISTINCLUDE**。

## → 语法

**MESSAGE** 'text-string'

## • 说明

伪指令 **MESSAGE** 引导编译器将 *text-string* 显示于屏幕上, 'text-string' 的字符必须使用一对单引号括起来。

→ 语法

**ERRMESSAGE** 'error-string'

• 说明

伪指令 **ERRMESSAGE** 引导编译器显示错误信息，'error-string' 的字符必须使用一对单引号括起来。

### 程序伪指令

→ 语法（注解）

; text

• 说明

注解是以分号(semicolon)开始的字符所组成，而由回车/换行符结束。

→ 语法

name **.SECTION** [align] [combine] 'class'

• 说明

伪指令 **.SECTION** 标明程序段(program section)或数据段(data section)的起始地址。程序段是由指令和/或数据所组成，这些指令与数据的地址是以该程序段的段名 name 为起始标准而定出的。段名 name 可以是唯一的或者是与其它段的段名相同，若两个段设定有完全相同的名称(complete name)，则被视为是同一个段。完全相同的名称是表示任何两个程序段的段名 name 及类别名 class 都相同。

选项 align 定义程序段起始地址的形态，可以用下列中的一种：

**BYTE** 以任意字节地址(byte)当做起始地址（编译器的默认形式）

**WORD** 以字地址（word，两个字节，即偶数地址）当做起始地址

**PARA** 以节段地址（paragraph，16 的倍数）当做起始地址

**PAGE** 以分页地址（page，256 的倍数）当做起始地址

针对 CODE 类别(class)的程序段，是以一条指令当做一个字节地址。**BYTE** 会将程序段的起始地址安排在任何指令的地址，**WORD** 则将程序段的起始地址安排在偶数的指令地址，**PARA** 将程序段的起始地址安排在 16 倍数的指令地址，而 **PAGE** 则将程序段的起始地址安排在 256 倍数的指令地址。

对于 DATA 类别的数据段而言，是以一个字节（8 位/字节）当做地址的计算单位。**BYTE** 会将数据段的起始地址安排在任何字节地址，**WORD** 则将数据段的起始地址安排在偶数地址，**PARA** 将数据段的起始地址安排在 16 倍数的地址，而 **PAGE** 会将数据段的起始地址安排在 256 倍数的地址。

选项 *combine* 定义如何结合有完全相同名称的程序段的方法，可以选用下列中的一种：

- **COMMON**  
将具有完全相同名称的所有程序段的起始地址安排在同一个地址，所使用的存储器长度则是以最长的程序段的长度为准。
- **AT *address***  
此选项是指定程序段的起始地址为 *address*，一个固定地址。编译器及连接器不能把它安排到其它的地址，而其内的标号(*label*)和变量(*variable*)的地址可以直接从 *address* 计算出。除了不可有前置引用(*forward reference*)的变量或符号外，可以使用任何合乎规定的表达式来表示 *address*，而运算结果的数值必须是合法的 ROM/RAM 存储器地址，且不能超出 ROM/RAM 的大小范围。

如果没有设定 *combine* 的形式，则该程序段是可结合的，换句话说，此程序段和其它具有完全相同名称的程序段可以连接成一个单一的程序段。

*Class* 是定义段存放的存储器类别。相同类别的段被安排在存储器中的连续区域。以其输入的先后顺序一个个紧接地安排在存储器中。类别名称为 **CODE** 的程序段将会放置在程序存储器(program memory - ROM)，而类别名称为 **DATA** 的数据段则是存储在数据存储器(data memory - RAM)。完整的段伪指令包括一个 *section* 名称和一个 *class* 名称。在此伪指令之后，直到下一段伪指令之前的所有指令及数据，都属于此段。

#### → 语法

**ROMBANK** *banknum section-name [,section-name,...]*

##### • 说明

此伪指令是用来声明程序存储器(program memory)的某一区块(*bank*)所包含的程序段。*banknum* 指定程序存储器的区块编号，范围从 0 到单片机的最大程序存储器区块数。*section-name* 则是先前已定义的程序段的名称。可以在同一个存储器区块内声明多个程序段，只要这些被声明的程序段的总和不超过 8K 字。如果程序中没有声明此伪指令，则所有类别为 **CODE** 的程序段都被视为属于区块 0 (*bank 0*)。如果某个类别为 **CODE** 的程序段没有被声明为属于任何程序存储器的区块内，此程序段将被视为属于区块 0。

## → 语法

```
RAMBANK banknum section-name [,section-name,...]
```

## • 说明

此伪指令与 **ROMBANK** 相似，不同的地方是声明数据存储器(data memory)的区块所包含的数据段(data section)。数据存储器区块的大小则为 256 字节。

## → 语法

```
END
```

## • 说明

此伪指令声明程序的结束，因此应该避免在任何包含文件(included file)中加入此伪指令。

## → 语法

```
ORG expression
```

## • 说明

此伪指令会将 *expression* 的计算数值设定给编译器的地址计数器(location counter)，其后的程序代码和数据偏移地址将根据 *expression* 所计算的偏移量做相对的调整。程序代码和数据偏移量与伪指令 **ORG** 所在的程序段的起始地址有关，程序段的属性会决定偏移量的实际值(是绝对地址或相对地址)。

## • 范例

```
ORG 8  
mov A, 1
```

在此范例中，语句 `mov A, 1` 的地址是在程序段的第 8 个地址。



## → 语法

**PUBLIC** *name1* [, *name2* [, ...]]

**EXTERN** *name1*:*type* [, *name2*:*type* [, ...]]

## • 说明

伪指令 **PUBLIC** 用来声明可被其它程序文件中的程序模块所使用的变量或标号，也就是公用变量或标号。另一方面，伪指令 **EXTERN** 则用来声明程序将使用的外部变量、标号或符号的名称和类型。这里提到的类型可使用下列四种形式中的一种：**BYTE**、**WORD**、**BIT**（这三种形式适用于数据变量）和 **NEAR**（用于调用或跳转的标号形式）。

## • 范例

```

PUBLIC start, setflag
EXTERN tmpbuf:byte
CODE    .SECTION 'CODE'
start:
    mov     a, 55h
    call   setflag
    ...
setflag proc
    mov     tmpbuf, a
    ret
setflag endp
end
    
```

在此范例中，标号 `start` 和程序 `setflag` 都被声明为公用变量，而其它源程序文件中的程序可以使用这些变量。变量 `tmpbuf` 则被声明为外部变量。在其它的源程序文件中，一定有一个名为 `tmpbuf` 的 `byte` 型变量定义，而且被声明为公用变量。

## → 语法

*name* **PROC**

*name* **ENDP**

## • 说明

伪指令 **PROC** 和 **ENDP** 用来定义一段可被其它程序调用或跳转到的程序代码。必须要指定一个名称 *name* 给 **PROC** 代表此程序(procedure)第一条指令的地址，而编译器会将标号的值设定至地址计数器中。

## • 范例

```

toggle    PROC
mov        tmpbuf, a
mov        a, 1
xorm      a, flag
mov        a, tmpbuf
ret
toggle    ENDP
    
```

→ 语法

[label:] DC expression1 [,expression2 [...]]

• 说明

伪指令 **DC** 会将 *expression1* 及 *expression2* 等的值存储在存储器的连续地址里，此伪指令只能使用于 **CODE** 类别的程序段之内，*expression1* 及 *expression2* 计算的数值将视单片机的程序存储器的宽度大小而定，编译器会将任何多余的位清除掉，*expression1* 必须为数值或标号，此伪指令通常被用在程序段之内建立表格以便查询。

• 范例

```
table: DC    0128H, 025CH
```

在此范例中，编译器会预留两个地址的 ROM 空间，并将 0128H 和 025CH 储存至这两个地址中。

### 数据定义伪指令

汇编程序由语句和注解组成。语句或注解则是由字符、数字和名称构成。汇编语言支持整数数字。整数可由二进制、八进制、十进制或十六进制加以表示（配合字尾的基数），如果未选基数，则编译器会使用默认值(十进制)，下表为可用的基数。

基数	形态	数字
B	二进制	01
O	八进制	01234567
D	十进制	0123456789
H	十六进制	0123456789ABCDEF

## → 语法

```
[name] DB value1 [,value2 [,...]]
[name] DW value1 [,value2 [,...]]
[name] DBIT
[name] DB repeated-count DUP(?)
[name] DW repeated-count DUP(?)
```

## • 说明

上述伪指令会引导编译器在数据存储器(data memory)内保留空间给变量 *name* (如果有指定 *name*)。存储器保留的空间大小则由其后的个数及数据类型,或由重复次数及数据类型来决定。由于单片机的数据存储器无法事先记录数据内容,编译器不会对数据存储器做初始值的设定,因此 *value1* 和 *value2* 必须为“?”,表示只是保留存储器空间给程序执行时使用,并没有设定其初始值。**DBIT** 只保留一个位,编译器会将每 8 个 **DBIT** 整合在一起并且保留一字节给这 8 个 **DBIT** 变量。

## • 范例

```
DATA                .SECTION      `DATA'
tbuf                 DB ?
chksum               DW ?
flag1                DBIT
sbuf DB ?
cflag                DBIT
```

在此范例中,编译器保留字节地址 0 给变量 *tbuf*、字节地址 1 和 2 给变量 *chksum*、字节地址 3 的第 0 位给变量 *flag1*、字节地址 4 给变量 *sbuf* 以及字节地址 3 的第 1 位给变量 *cflag*。

## → 语法

```
name LABEL {BIT | BYTE | WORD}
```

## • 说明

此伪指令会将 *name* 的地址设定为与其后的变量相同的存储器地址。

## • 范例

```
lab1                 LABEL      WORD
d1                   DB ?
d2                   DB ?
```

在这个范例中, *d1* 是 *lab1* 的低字节,而 *d2* 则是 *lab1* 的高字节。

## → 语法

```
name EQU expression
```

## • 说明

**EQU** 伪指令将 *expression* 传送给 *name*，从而产生一个绝对符号、别名或文字符号。绝对符号是一个代表 16 位值的名称；别名则替代另一个符号；而文字符号则是代表一串字符组合的名称。*name* 必须是唯一的，即先前未被定义过。*expression* 可以是一个整数、字符串常数、指令助记符、数字表达式或地址表达式。

## • 范例

```
    accreg      EQU    5
    bmove      EQU    mov
```

在这个范例中，变量 *accreg* 等于 5，而 *bmove* 相当于指令 *mov*。

### 宏指令

宏指令定义一个名称来代表一段源程序语句，而在源程序文件中可以重复使用这个名称以取代这段语句。在编译时，编译器会自动将每一个宏指令的名称用宏指令所定义的程序语句来取代。

宏指令可以在源文件的任何地方被定义，只要调用此宏指令是在宏指令定义之后即可。宏指令的定义中，可以调用先前已经被定义的其他宏指令，如此将形成一种嵌套的结构，编译器提供最多 7 层的嵌套。

## → 语法

```
nameMACRO [dummy-parameter [,...]]
    statements
ENDM
```

在宏指令中，可以使用伪指令 **LOCAL** 来定义只能在宏指令本体内使用的变量。

## → 语法

```
name LOCAL dummy-name [, ...]
```

## • 说明

宏指令 **LOCAL** 用来定义只能在宏指令本体内使用的符号，使用时必须定义在 **MACRO** 伪指令之后的第一行。*dummy-name* 是一个暂时使用的名称，当宏指令被调用展开时，它将被一个唯一的名称所取代。编译器会对 *dummy-name* 产生对应的实际名称。这个实际名称的格式为??digit，其中 digit 数字为十六进制且范围由 0000 至 FFFF。当 **MACRO/ENDM** 的定义区段中使用到一些标号(label)时，要将这些标号加入 **LOCAL** 伪指令中，否则当 **MACRO** 被源文件多次引用时，相同的标号名称会重复出现在程序中，编译器会发布程序错误的信息。

下面的范例中，tmp1 和 tmp2 都是形式参数，当调用此宏指令时，都会被实际参数所取代，label1 和 label2 都被声明为 **LOCAL**，如果没有其它的 **MACRO** 被引用，在第一次引用时将分别被??0000 和??0001 所取代，如果没有声明 **LOCAL**，label1 和 label2 则会类似于源程序中的标号声明，而在第二次调用此宏指令时，就会出现重复定义的错误信息。

```
Delay MACRO    tmp1, tmp2
LOCAL         label1, label2
    mov       a, 70h
    mov       tmp1, a
label1:
    mov       tmp2, a
label2:
    clr       wdt1
    clr       wdt2
    sdz       tmp2
    jmp       label2
    sdz       tmp1
    jmp       label1
ENDM
```

下面的源程序将会调用名为 Delay 宏指令

```

; T.ASM
; Sample program for MACRO.
.ListMacro
Delay MACRO tmp1, tmp2
    LOCAL label1, label2
    mov a, 70h
    mov tmp1, a
label1:
    mov tmp2, a
label2:
    clr wdt1
    clr wdt2
    sdz tmp2
    jmp label2
    sdz tmp1
    jmp label1
ENDM

data .section 'data'
BCnt db ?
SCnt db ?

code .section at 0 'code'
Delay BCnt, SCnt
end
    
```

编译器会将宏指令 Delay 展开如下列的程序。请注意在宏指令本体内的第 4 行到第 17 行，它们的偏移地址(offset)都是 0000，也就是宏指令在定义时，本体内的指令并不占用存储器空间。在程序第 24 行调用 Delay 宏指令时，它就被展开成 11 行，也就是展开为宏指令程序。形式参数 tmp1 和 tmp2 分别被实际参数 BCnt 和 SCnt 所取代。

File: T.asm

Holtek Cross-Assembler Version 2.80

Page 1

```

1 0000 ; T.ASM
2 0000 ; Sample program for MACRO.
3 0000 .ListMacro
4 0000 Delay MACRO tmp1, tmp2
5 0000 LOCAL label1, label2
6 0000 mov a, 70h
7 0000 mov tmp1, a
8 0000 label1:
9 0000 mov tmp2, a
10 0000 label2:
11 0000 clr wdt1
12 0000 clr wdt2
13 0000 sdz tmp2
14 0000 jmp label2
15 0000 sdz tmp1
16 0000 jmp label1
17 0000 ENDM
18 0000
19 0000 data .section 'data'
20 0000 00 BCnt db ?
21 0001 00 SCnt db ?
22 0002
23 0000 code .section at 0 'code'
24 0000 Delay BCnt, SCnt
24 0000 0F70 1 mov a, 70h
24 0001 0080 R1 mov BCnt, a
24 0002 1 ??0000:
24 0002 0080 R1 mov SCnt, a
24 0003 1 ??0001:
24 0003 0001 1 clr wdt1
24 0004 0005 1 clr wdt2
24 0005 1780 R1 sdz SCnt
24 0006 2803 1 jmp ??0001
24 0007 1780 R1 sdz BCnt
24 0008 2802 1 jmp ??0000
25 0009 end

```

0 Errors

## 汇编语言指令

指令的语法形式如下：

```
[name:] mnemonic [operand1 [,operand2]] [; comment]
```

其中

<i>name</i> :	→ 符号名称
<i>mnemonic</i>	→ 指令名称（关键字）
<i>operand1</i>	→ 寄存器 存储器地址
<i>operand2</i>	→ 寄存器 存储器地址 立即数

### 名称

名称由字母、数字或特殊字符所组成，可以当标号(label)使用。当标号使用时，必须在名称后面紧接一个冒号(colon)。

### 助记符

助记符是源程序中使用的指令名称，它取决于源程序所使用的单片机型号。

### 操作数、运算子和表达式

操作数（源操作数或目的操作数）定义被指令所使用的数值。它们可以是常数、变量、寄存器、表达式或关键字。当使用指令时，必须谨慎选择正确的操作数，即源操作数和目的操作数。符号\$是一个特殊的操作数，它代表当前的地址。

表达式是由操作数所组成，在程序编译时用来计算出数值或存储器地址。表达式是常数、符号以及任何被算术运算子分隔的常数和符号组合。

运算子定义表达式中各操作数之间的运算动作。编译器提供了许多运算子去处理操作数。有些运算子只处理常数，有些则处理存储器数值，也有两者兼具的。如果运算子处理的是常数，则在程序编译时就会直接计算出数值。以下是编译器所提供的运算子。



- 算术运算符  $+ - * / \%$  (MOD)

- SHL 和 SHR 运算符

- 语法

```
expression SHR count
expression SHL count
```

这些位平移运算符的值全都为常数，*expression* 依照 *count* 所指定的数目向右移 (**SHR**) 或向左移 (**SHL**)，如果被平移的位超过有效位数时，则对应的位会以 0 填满，如：

```
mov A, 01110111b SHR 3 ; result ACC=00001110b
mov A, 01110111b SHL 4 ; result ACC=01110000b
```

- 逻辑运算符 NOT、AND、OR、XOR

- 语法

```
NOT expression
expression1 AND expression2
expression1 OR expression2
expression1 XOR expression2
```

**NOT** 各位的 1 阶补码

**AND** 各位 AND 运算

**OR** 各位 OR 运算

**XOR** 各位 XOR 运算

- 偏移运算符 OFFSET

- 语法

```
OFFSET expression
```

**OFFSET** 运算符返回 *expression* 的偏移地址。*expression* 可以是标号、变量或其它直接存储器的操作数。被 **OFFSET** 运算符所返回的数值必须是立即数。

- LOW、MID 和 HIGH 运算符

- 语法

```
LOW expression
MID expression
HIGH expression
```

如果 *expression* 的结果为一个立即数的话，则 **LOW/MID/HIGH** 运算符返回值就是 *expression* 的值，而且是分别取此数值的低/中/高字节。但是如果 *expression* 是标号，则 **LOW/MID/HIGH** 运算符将取得此标号所在的程序存储器地址的低/中/高字节的数值。

- BANK 储存区块运算符

- 语法

**BANK** *name*

**BANK** 运算符会返回程序段所在的存储器区块的编号，此程序段的名称是 *name*。如果 *name* 是标号，则返回 ROM 程序存储器区块。如果 *name* 是数据变量则返回 RAM 数据存储器区块。存储器区块的数值格式与寄存器 BP 的格式相同，请参考各单片机的规格。（注意：不同的单片机可能有不同的 BP 格式。）

范例 1:

```
mov A, BANK start
mov BP, A
jmp start
```

范例 2:

```
mov A, BANK var
mov BP, A
mov A, OFFSET var
mov MP1, A
mov A, R1
```

- 运算符的优先权

优先权	运算符
1(Highest)	(), []
2	+, - (unary), LOW, MID, HIGH, OFFSET, BANK
3	*, /, %, SHL, SHR
4	+, - (binary)
5	> (greater than), >= (greater than or equal to), < (less than), <= (less than or equal to)
6	== (equal to), != (not equal to)
7	! (bitwise NOT)
8	& (bitwise AND)
9(Lowest)	(bitwise OR), ^ (bitwise XOR)

## 其它

### 前置引用

当标号、变量名称和其它符号在源程序中被声明之前，编译器允许它们被使用(前置命名引用)，但是在伪指令 **EQU** 右边的符号是不允许前置引用的。

### 局部标号

局部标号有固定的形式，即 `$number`。其中 `number` 可以为 0 至 29，局部标号除了可以重复使用外，其它功用与一般标号相同。局部标号必须使用在任意两个连续的标号之间而且同样的局部标号名称也可以用在其它的两个连续标号之间。在编译源程序文件之前，编译器会将每一个局部标号转换成唯一的标号。任何两个连续标号之间，最多可以定义 30 个局部标号。

### 范例

```
Label1:                                ; label1
    $1:                                ;; local label
        mov a, 1
        jmp $3
    $2:                                ;; local label
        mov a, 2
        jmp $1
    $3:                                ;; local label
        jmp $2
Label2:                                ; label
    jmp $1
    $0:                                ;; local label
        jmp Label1
    $1:                                jmp $0
Label3:
```

### 汇编语言保留字

下表是汇编语言上使用的保留字。

- 保留字（伪指令、运算符）

\$	DUP	INCLUDE	NOT
*	DW	LABEL	OFFSET
+	ELSE	.LIST	OR
-	END	.LISTINCLUDE	ORG
.	ENDIF	.LISTMACRO	PAGE
/	ENDM	LOCAL	PARA
=	ENDP	LOW	PROC
?	EQU	MACRO	PUBLIC
[ ]	ERRMESSAGE	MESSAGE	RAMBANK
AND	EXTERN	MID	ROMBANK
BANK	HIGH	MOD	.SECTION
BYTE	IF	NEAR	SHL
DB	IFDEF	.NOLIST	SHR
DBIT	IFE	.NOLISTINCLUDE	WORD
DC	IFNDEF	.NOLISTMACRO	XOR

- 保留字（指令助记符）

ADC	HALT	RLCA	SUB
ADCM	INC	RR	SUBM
ADD	INCA	RRA	SWAP
ADDM	JMP	RRC	SWAPA
AND	MOV	RRCA	SZ
ANDM	NOP	SBC	SZA
CALL	OR	SBCM	TABRDC
CLR	ORM	SDZ	TABRDL
CPL	RET	SDZA	XOR
CPLA	RETI	SET	XORM
DAA	RL	SIZ	
DEC	RLA	SIZA	
DECA	RLC	SNZ	

- 保留字（寄存器名称）

A	WDT	WDT1	WDT2
---	-----	------	------

## 编译器选项

编译器选项可以通过 HT-IDE3000 中的 Options 菜单的 Project 命令来设定，编译器的选项位于 Project Option 对话框的中心部分。

可在符号定义(Define Symbol)编辑框中定义符号。

### → 语法

```
symbol1 [=value1] [,symbol2 [=value2] [,...]]
```

### • 范例

```
debugflag=1, newver=3
```

产生列表文件的检查框用来指定是否要生成列表文件(Listing file)，如果检查框被选中，则要生成列表文件，否则将不会产生列表文件。

## 编译列表文件格式

编译列表文件包含源程序的列表和概要信息，每页的第一行是标题，内容则包括公司名称、编译器版本、源文件名称、编译时的日期、时间以及页码。

### 源程序列表

在源程序中的每行语句都以下列的格式输出到编译列表文件：

```
[line-number] offset [code] statement
```

- *Line-number* 是指语句在源程序文件的第几行，从第一个语句开始计算起(4 位十进制数)。
- *offset*-是从语句所在的程序段开始到这个语句的存储器地址的偏移量(4 位十六进制数)。
- *code*-只有会产生机器码(machine code)或数据的语句才会出现此项(两个 4 位十六进制数)。

如果数值在编译时已确定的话，会用十六进制数字表示 *code* 的数值，否则的话，将使用适当的标志位表明应该使用何种方式去计算此数值。下列两个标志位可能会出现于 *code* 项之后。

**R** → 需要重新安置地址(连接器解决此状况)

**E** → 需要参考外部符号(连接器解决此状况)

下列标志位可能会出现于 *code* 项之前。

**=** → **EQU** 或等号

code 项中可能出现下列的符号或数字。

- → 代表程序段的起始地址(连接器会解决此符号)
- nn[xx] → **DUP** 符号: nn **DUP**(?)重复次数
- *statement* -源文件对应的源程序语句或是宏指令所展开的语句, 在语句之前可能会出现下列的符号。
  - n → 宏指令展开时的嵌套层次
  - C → 此语句是从包含文件(**INCLUDE** 文件)引进的
- 总结

```

0           1           2           3           4           5           6
123456789012345678901234567890123456789012345678901234567890...
IIII  oooo hhhh hhhh EC source-program-statement
                        Rn
    
```

- IIII → 行号 (4 位数, 向右靠齐)
- oooo → 机器码的地址偏移量 (4 位数)
- hhhh → 两个 4 位数的机器码
- E → 外部引用
- C → 从包含文件加入的语句
- R → 需要重新安置地址
- n → 宏指令展开后的嵌套层次

### 编译总结

在编译列表文件的结尾处会统计此次编译所发生的警告及错误的总数。

### 其它

在编译期间如果发生错误, 则错误信息和编号会直接出现在发生错误的语句下方。

## → 编译列表文件的范例

File: SAMPLE.ASM      Holtek Cross-Assembler    Version 2.86      Page 1

```

1 0000          page 60
2 0000          message   'Sample Program 1'
3 0000
4 0000          .listinclude
5 0000          .listmacro
6 0000
7 0000          #include "sample.inc"

1 0000          C pa      equ    [12h]
2 0000          C pac     equ    [13h]
3 0000          C pb      equ    [14h]
4 0000          C pbc     equ    [15h]
5 0000          C pc      equ    [16h]
6 0000          C pcc     equ    [17h]
7 0000          C

8 0000
9 0000          extern extlab : near
10 0000         extern extb1 : byte
11 0000
12 0000         clrpb macro
13 0000         clr pb
14 0000         endm
15 0000
16 0000         clrpa macro
17 0000         mov a, 00h
18 0000         mov pa, a
19 0000         clrpb
20 0000         endm
21 0000
22 0000         data .section 'data'
23 0000 00      b1      db ?
24 0001 00      b2      db ?
25 0002 00      bit1    dbit
26 0003
27 0000         code .section 'code'
28 0000 0F55     mov a, 055h
29 0001 0080     R mov b1, a
30 0002 0080     E mov extb1, a
31 0003 0FAA     mov a, 0aah
32 0004 0093     mov pac, a
33 0005         clrpa
33 0005 0F00     1 mov a, 00h
33 0006 0092     1 mov [12h], a
33 0007         1 clrpb
33 0007 1F14     2 clr [14h]
34 0008 0700     R mov a, b1
35 0009 0F00     E mov a, bank extlab
36 000A 0F00     E mov a, offset extb1
37 000B 2800     E jmp  extlab
38 000C
39 000C 1234 5678 dw  1234h, 5678h, 0abcdh, 0ef12h
   ABCD EF12
40 0010         end
    
```

0 Errors





## 第三部分

# 开发工具



## 第五章

## 单片机开发工具

## 5

为简化应用程序的开发过程，支持工具的重要性和有效性对于单片机来说是不可低估的。为了支持所有系列的 MCU，盛群用心的提供了具有完整功能的工具，让用户在开发与使用上更加便利，例如众所周知的 HT-IDE 集成开发环境，软件方面有 HT-IDE3000 软件，提供友好的视窗接口以便进行程序的编辑及除错，同时硬件方面为 HT-ICE 仿真器，提供多种实时仿真功能，包含多功能跟踪、单步执行和设定断点功能。HT-IDE 开发系统提供完整的接口卡与定期软件服务包的更新，因此保证设计者可以有最佳的工具，且能以最高效率进行单片机应用程序的设计与开发。

### HT-IDE 集成开发环境

HT-IDE(Holtek Integrated Development Environment)是一个具有高效能，使用于设计盛群 8 位 MCU 应用程序的集成开发环境。系统中的硬件及软件工具能帮助客户使用盛群 8 位 MCU 芯片，快速且容易的开发应用程序。在 HT-IDE 中最主要的组件为 HT-ICE (In-Circuit Emulator)，它提供了盛群 8 位单片机的实时仿真功能，和强有力的除错和跟踪功能。最新版本的 HT-ICE 仿真器进一步集成 OTP 烧写器在仿真器上，提供使用者从程序设计、除错到烧写所有功能。

在软件方面，HT-IDE3000 开发系统提供友好的工作平台。此平台将所有的软件工具，例如编辑器、编译器、连接器、函数库管理器和符号除错器，并入到视窗环境，使程序开发过程更为容易。HT-IDE3000 还提供软件仿真功能，无需接上 HT-ICE 仿真器，就可以进行程序开发。此软件仿真器可以仿真盛群 8 位 MCU，以及 HT-ICE 硬件的所有基本功能。

HT-IDE3000 使用手册中包含 HT-IDE 开发系统的相关细节。为了确保开发系统包含有最新的单片机和软件更新信息，盛群也定期提供 HT-IDE3000 服务软件 (Service Pack) 以配合 HT-IDE3000 的安装。这些服务软件不是用来取代 HT-IDE3000，它必须要在 HT-IDE3000 系统软件安装后才被安装。

HT-IDE3000 开发系统具有下列的特性：

- **仿真**
  - 程序指令的实时仿真
- **硬件**
  - 使用及安装容易
  - 可使用内部或外部振荡器
  - 断点功能
  - 支持跟踪功能与触发能力的跟踪仿真器
  - HT-ICE 通过打印口与计算机连接
  - 使用者的应用电路板通过 I/O 接口卡连接至 HT-ICE
  - HT-ICE 中集成 OTP 烧写器
- **软件**
  - 通用的视窗软件
  - 源程序层次的除错方式(符号除错器)
  - 支持多个源程序文件的工作平台(一个应用项目可以包含一个以上的源程序文件)
  - 所有的工具都用于开发、除错、评估和产生最后的应用程序代码(Mask ROM file 和 OTP file)
  - 可以建立公用程序的函数库，之后被连接到另一个项目去使用
  - 软件仿真器不需要连接 HT-ICE 硬件即可进行程序的仿真和除错
  - 虚拟外围器件管理(VPM)可仿真外围器件的行为
  - LCD 仿真器可仿真 LCD 面板的动作

## 盛群单片机仿真器(HT-ICE)

对于盛群的 8 位单片机而言，盛群的 ICE 是全功能的仿真器，系统中的硬件及软件工具能帮助客户快速方便的开发应用程序。系统中最主要的是硬件仿真器，除了能够有效地提供除错和跟踪功能之外，还能以实时的方式进行盛群 8 位 MCU 的仿真工作。在软件方面，HT-IDE3000 开发系统提供友好的工作平台，将所有软件工具，例如编辑器、编译器、连接器、函数库管理器和符号除错器，合并到视窗环境。此外系统在软件仿真模式下不需连接 HT-ICE 硬件即能执行程序仿真。

### HT-ICE 接口卡

HT-ICE 的接口卡可以被许多的应用电路使用，但是使用者也可自行设计接口卡，将必要的接口电路放在他们自己的接口卡上，使用者可以直接把他们的应用电路板连接到 HT-ICE 的 CN1 和 CN2 连接器。

### OTP 烧写器

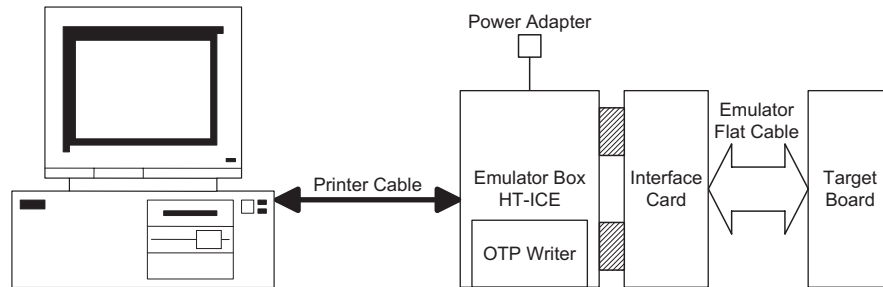
所有盛群的 OTP 芯片都有烧写器支持。对于工业级的 OTP 芯片烧写而言，盛群 OTP 烧写工具提供一个快速且有效的方法，来进行 OTP 的小规模量产烧写。最新版本的 HT-ICE 仿真器更进一步的将 OTP 烧写器集成在 HT-ICE 仿真器上，提供使用者从程序设计、除错到烧写验证的所有必需功能。另外有更多的烧写器供应商可提供有效及更大容量的烧写服务。请参阅网站以获得进一步供应商情况。

### OTP 转接座

OTP 烧写器本身提供一个标准的芯片插座，而 OTP 转接座则是使用在烧写其它封装形式的 OTP 芯片，这些封装形式的芯片无法在标准芯片插座上烧写，需要在 OTP 烧写器插上此转接座才能烧写。

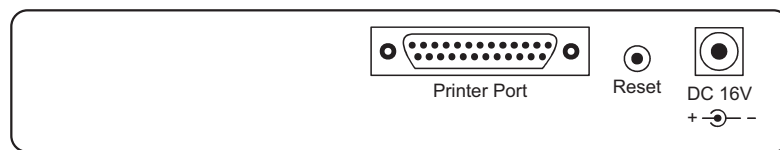
## 系统配置

HT-IDE 集成开发环境硬件配置如下，主机需为 Pentium 兼容机器(Windows 95/98/NT/2000/XP 或更新)。注意在 Windows NT/2000/XP 系统下安装 HT-IDE3000 时，需在 Supervisor Privilege 模式下执行 HT-IDE3000 软件安装。

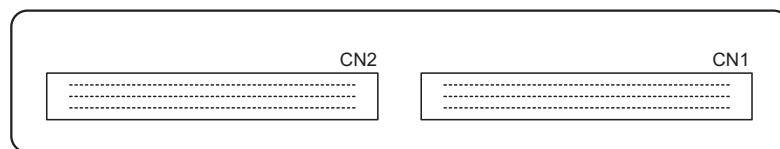


HT-IDE 集成系统环境包含下列硬件组成：

- HT-ICE 机台，包含印刷电路板 PCB，其中有一条打印口用于连接硬件仿真器与主机、I/O 接口接头以及电源指示灯 LED，如下图所示。
- 连接目标电路板与 HT-ICE 机台的 I/O 接口卡
- 变压器(输出 16V)
- 25 脚的 D 型打印并口线
- OTP 烧写器



HT-ICE Rear View

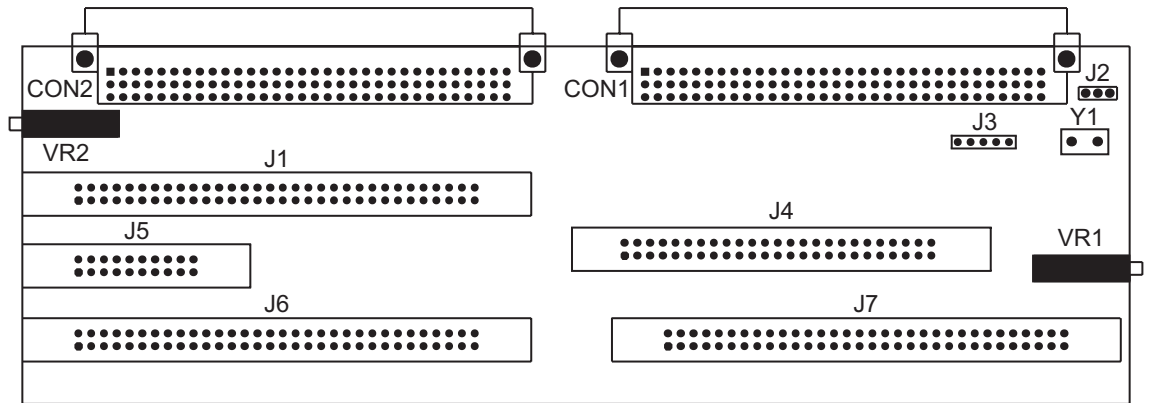


HT-ICE Front View

### HT-ICE 接口卡设定

HT-ICE 接口卡(CPCB49C000001B)连接使用者的应用电路板与 HT-ICE，它提供下列功能：

- 外部时钟
- 外部输入的跟踪信号
- MCU 的 I/O 与 LCD 引脚配置



外部的时钟源有两种模式：RC 和 Crystal。使用 Crystal 模式时，必须将 J2 位置的第二和第三脚短路，然后在 Y1 位置插入合适的晶振。而 RC 模式则是将第一和第二脚短路，并且可以通过 VR1 来调整系统频率。请参考 HT-IDE3000 使用者手册中 Tools 菜单的 Mask Option 命令选择时钟源和系统频率。VR2 用于调整 LCD 电压(VLCD)。

四个外部信号的跟踪输入端口位于 J3，标示为 ET0~ET3，它们提供了用外部信号作为内部断点触发的方式以及跟踪功能。要获得更多的信息，请参考 HT-IDE3000 使用者手册中断点与跟踪应用程序的说明。

对于不同的单片机，连接器 J1 和 J4~J7 的使用情况如下表所示：

<b>Part No</b>	<b>Socket</b>
<b>HT49R30A-1/HT49C30-1/HT49C30L</b>	<b>J7 or J1 + J4</b>
<b>HT49R50A-1/HT49C50-1/HT49C50L</b>	<b>J6 or J1 + J4</b>
<b>HT49R70A-1/HT49C70-1/HT49C70L</b>	<b>J5 + J6 or J1 + J4</b>
<b>HT49RU80/HT49CU80</b>	<b>J1 + J4</b>



## 安装

### 系统需求

安装 HT-IDE3000 系统的硬件及软件需求如下：

- Pentium 等级以上 CPU 之 PC/AT 兼容机器
- SVGA 彩色显示器
- 至少 32M 以上的 RAM
- CD ROM 装置(从 CD 安装时需要)
- 至少 20M 以上的硬盘空间
- 具有并行口，可连接 PC 和 HT-ICE
- 操作系统 Windows 95/98/NT/2000/XP

\* Window 95/98/NT/2000/XP 是 Microsoft 公司注册商标

### 硬件安装

- 步骤 1  
将电源变压器插入 HT-ICE 的电源插孔
- 步骤 2  
通过 I/O 接口卡或排线连接目标电路板与 HT-ICE
- 步骤 3  
使用打印并口线连接 HT-ICE 至主机

此时 HT-ICE 上的 LED 应该是亮的，如果不是，则重新操作连接的步骤或与代理商联系。

---

**警告：** 请小心使用电源变压器，勿使用输出不是 16V 的变压器，否则可能导致 HT-ICE 损坏，因此强烈建议使用由盛群所提供的变压器。首先将电源变压器插入 HT-ICE 的电源插座。

---

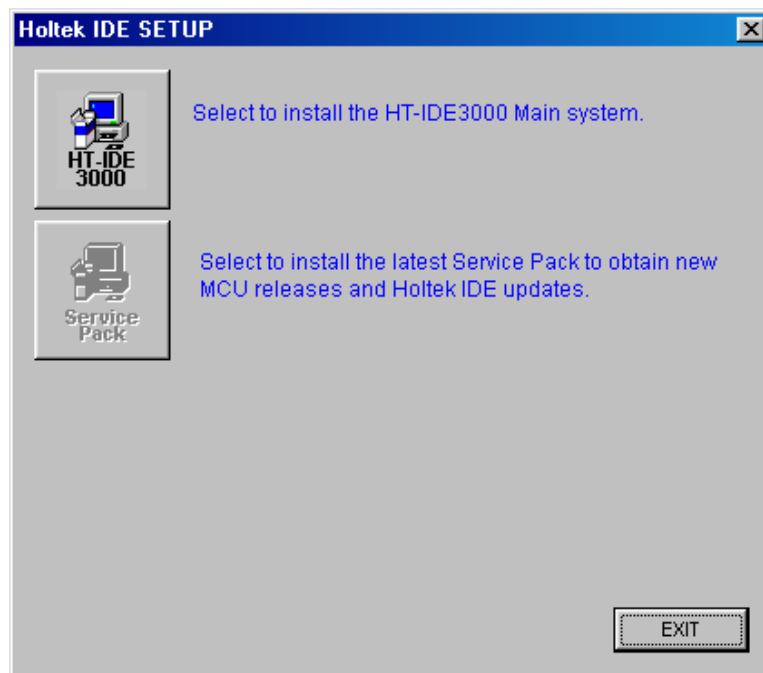
软件安装

- 步骤 1

将 HT-IDE3000 CD 放入 CD ROM 装置中，将出现下列的对话框。



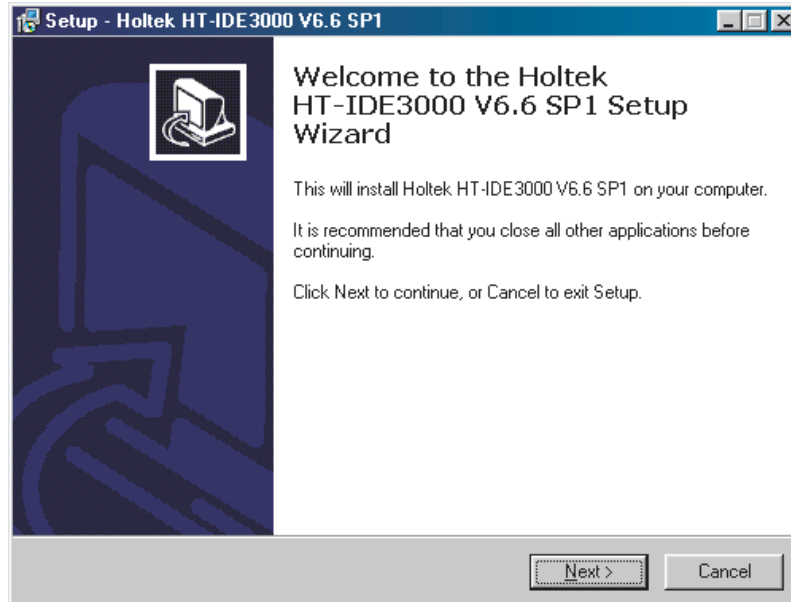
按下<HT-IDE3000>按钮，下列的对话框会出现。



依照你想要安装的功能，请按下<HT-IDE3000>或者<Service Pack>。以下为选择安装<HT-IDE3000>的范例说明，按下<HT-IDE3000>。

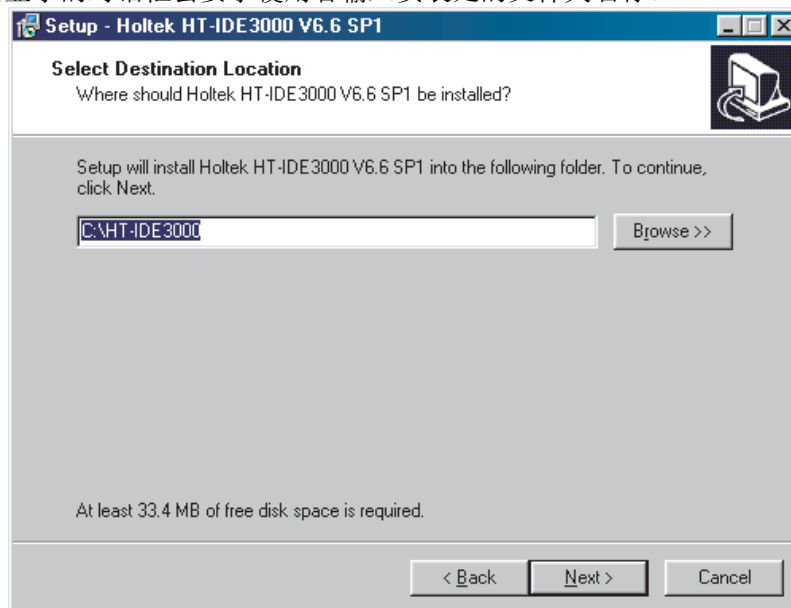
- 步骤 2

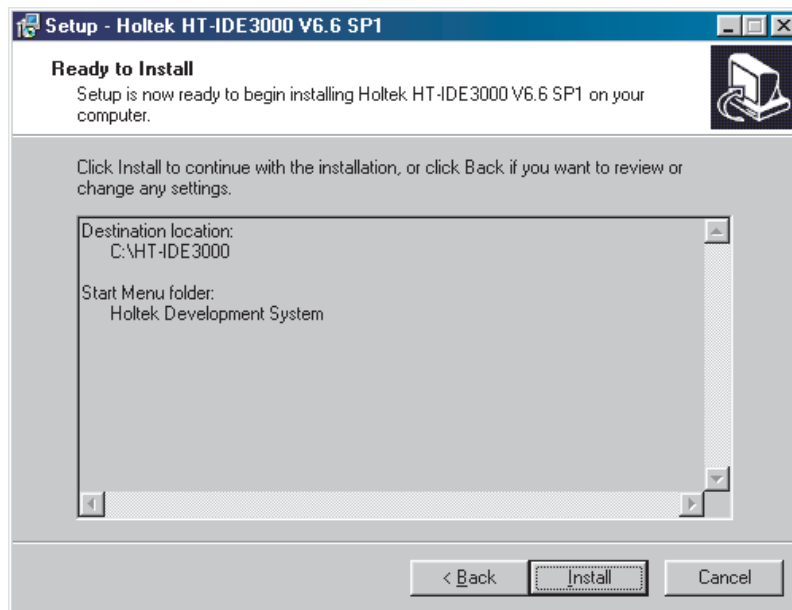
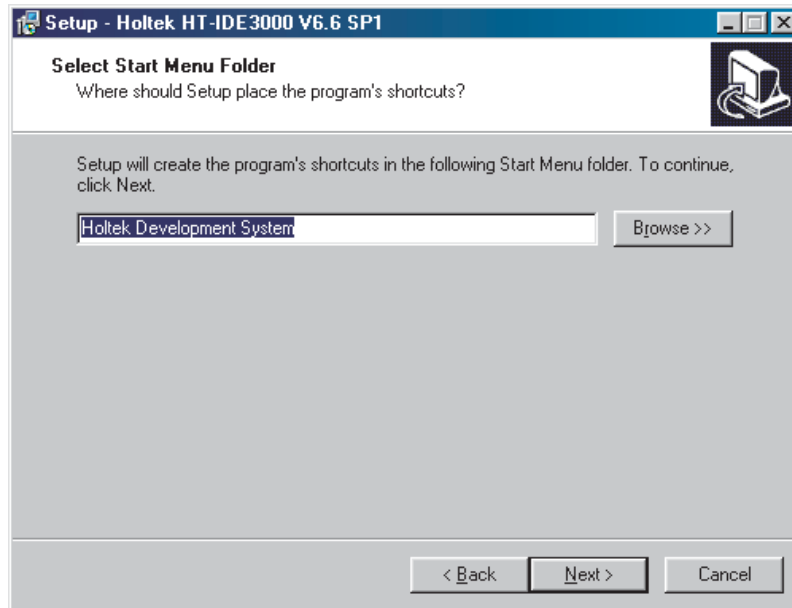
按下<Next>按钮(继续安装)或按下<Cancel>按钮(中止安装)。



- 步骤 3

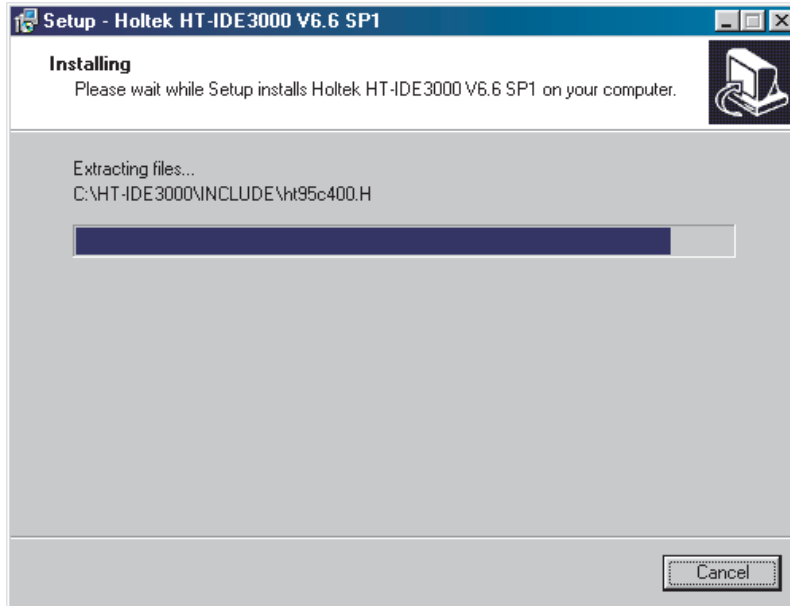
下面显示的对话框会要求使用者输入安装处的文件夹名称。



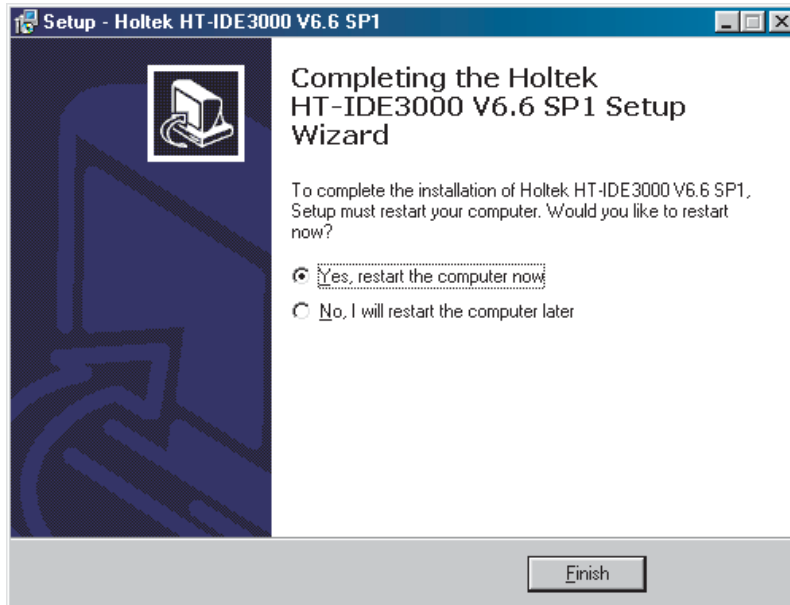


- 步骤 4  
指定你希望安装 HT-IDE3000 的文件夹路径，然后按下<Next>按钮。

- 步骤 5  
SETUP 会将文件复制到你指定的文件夹。



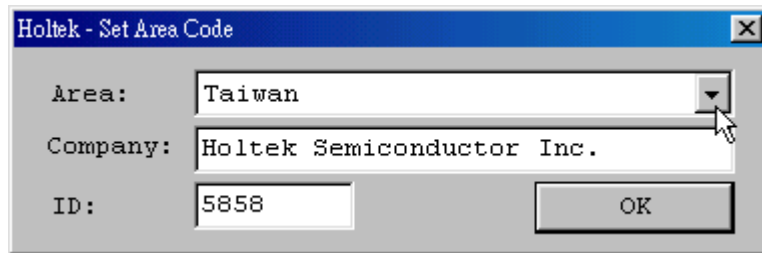
- 步骤 6  
安装成功的话，则会出现下面的对话框。



- 步骤 7

请按下<Finish>按钮完成安装并重新启动计算机系统之后，你就可以执行 HT-IDE3000 程序了。SETUP 会在你指定的文件夹下，建立四个子文件夹 (BIN、INCLUDE、LIB、SAMPLE)。BIN 子文件夹包含所有的系统可执行文件(EXE)、动态链接库(DLL)和配置文件(CFG、FMT)，INCLUDE 子文件夹包含所有由盛群所提供的包含文件(.H、.INC)，LIB 子文件夹包含由盛群所提供的库文件(.LIB)，SAMPLE 子文件夹包含范例程序。

注意，在第一次执行 HT-IDE3000 之前，系统会要求输入如下图所示的公司资料，请选择适当的区域并填入公司名称及识别码，其中识别码可由 HT-IDE3000 的供应商提供。



## 第六章

## 快速开始

## 6

本章简述如何快速使用 HT-IDE3000 去开发一个应用程序项目。

**步骤一: 建立一个新项目**

- 按下 Project 菜单并选择 New 命令
- 输入项目名称并从组合框选择此项目使用的单片机型号
- 按下 OK 按键则系统将会要求设定单片机的掩膜选项
- 设定所有掩膜选项并按下 SAVE 键

**步骤二: 将源程序文件加到项目中**

- 使用 File/New 命令建立源程序文件
- 撰写完程序后存盘, 如 TEST.ASM 档名
- 按下 Project 菜单并选择 Edit 命令
- 进入 Edit Project 对话框以便将源程序文件加入项目, 或从项目中删除
- 选择一个源程序文件名称, 如 TEST.ASM, 按下 Add 按钮
- 当所有源程序文件都被加入项目后, 按下 OK 按钮

**步骤三: 编译项目**

- 按下 Project 菜单并选择 Build 命令
- 系统将会对项目中的所有源程序文件执行编译动作
  - 如果程序中有错误, 只要在错误信息行连按两次, 则系统将会提示错误发生的位置并且打开此错误所在的源程序文件, 可直接修改程序及存储文件
  - 如果所有程序文件都没有错误, 则系统会产生一个执行文件并且载入到 HT-ICE 中, 准备仿真及除错
- 你可以重复此步骤直到没有错误

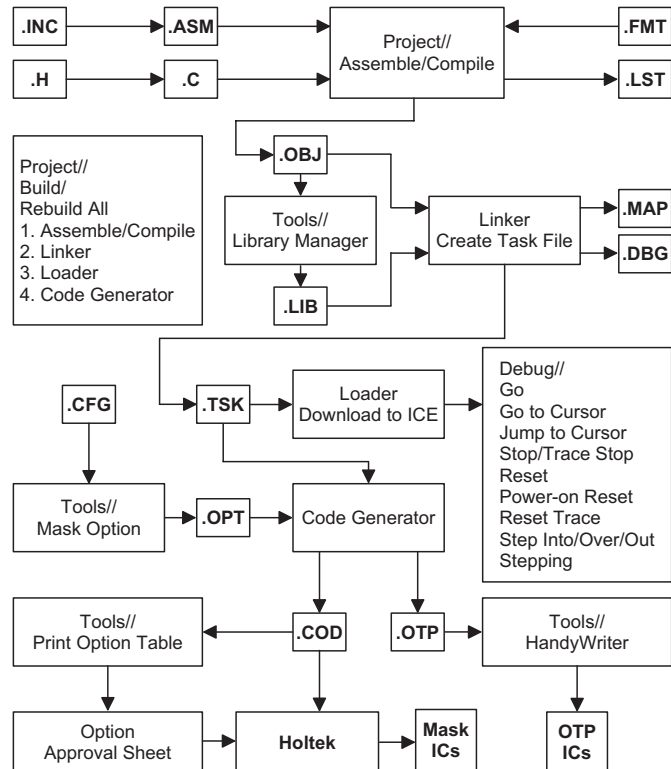
**步骤四:烧写 OTP 单片机**

- 建立项目, 产生.OTP 文件
- 按下 Tools 菜单并选择 OTP Writer 命令去烧写 OTP 芯片

**步骤五:传送程序与掩膜选项单至 Holtek**

- 按下 Project 菜单并选择 Print Option Table 命令去打印掩膜选项确认单
- 传送.COD 文件和掩膜选项确认单到盛群半导体公司, 进行生产

程序及数据流程可由下图表示:





## 第七章

# LCD 仿真器

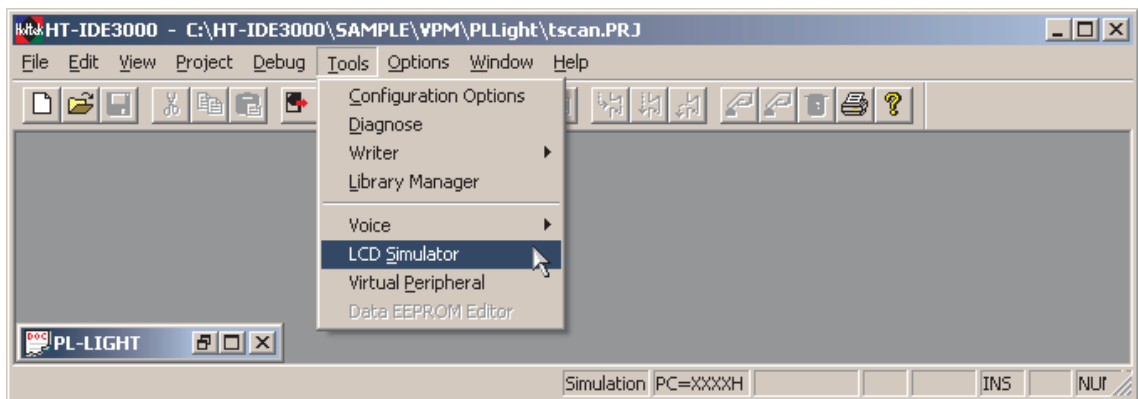
# 7

### 简介

盛群的 LCD 仿真器，即是 HT-LCDS，提供 LCD 驱动器的软件仿真。根据使用者设计的图形和控制程序，HT-LCDS 在显示器上实时显示这些图形，在尚未取得真实的 LCD 硬件面板之前，LCD 仿真器将使开发的过程更为便利。但是如果项目所用的单片机没有支持 LCD 功能，则这些命令是无效的。

### LCD 面板配置文件

在开始 LCD 的仿真之前，必须先建立一个 LCD 面板配置文件。HT-LCDS 将从 LCD 面板配置文件取得 LCD 的设定数据并根据仿真时取得的数据将图形显示在显示器上，如果这个文件不存在，HT-LCDS 将无法仿真 LCD 的动作。对于具有处理 LCD 功能的单片机，必须要建立 LCD 面板配置文件以便执行 LCD 的仿真。工具菜单(Tools menu)中的 LCD Simulator 命令可用于建立面板配置文件及执行 LCD 的仿真。LCD 面板配置文件包含面板的配置数据及图形信息，用户可以使用 HT-LCDS 进行设定。



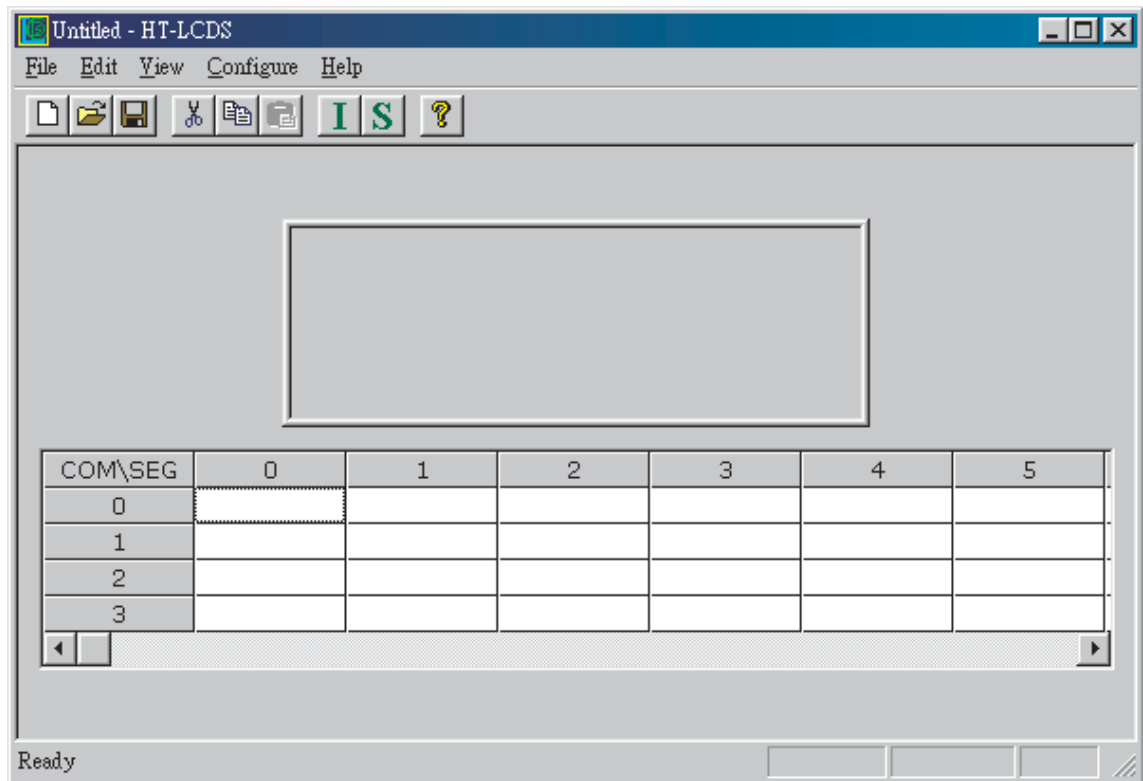
### 面板文件的文件名与项目的关系

原则上，除了后缀名为.lcd 外，面板配置文件的主文件名和当前项目(current project)的名称是相同的。HT-LCDS 会认为这个文件是对应于当前项目的面板配置文件。使用 HT-LCDS 的 File 菜单中的 New 命令或工具栏的 New 按钮可以建立新的面板配置文件。如果想要将面板配置文件改为与当前项目不同的文件名时，可以使用 File 菜单的 Save 命令或工具栏中的 Save 按钮。

当 HT-LCDS 开始仿真时，它要参考当前的面板配置文件以获取仿真所需要的信息，此时可由选择 HT-LCDS 的 File 菜单的 New 或 Open 命令将面板配置文件启动为当前文件。LCD 面板配置文件的文件名可能和当前项目的名称相同，也可以选用不同的名称。

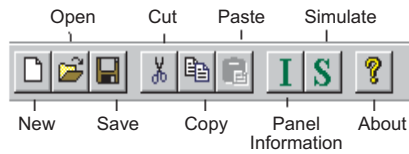
### 选择 HT-LCDS

从 Tools 菜单中选用 LCD simulator 命令时，如果当前项目对应的面板配置文件已存在，会显示 LCD 仿真的对话框。在此窗口下方的表格中，某些指定的 COM/SEG 位置中会显示所对应的位图图形文件(bitmap)的文件名，同时这些图形会显示在窗口上方的面板显示框。如果在项目的路径中并未存有相关的面板配置文件，则 LCD 仿真器在启动时并不会显示面板显示框和 COM/SEG 表格。



LCD 仿真器对话框

下图为 HT-LCDS 菜单栏



- New: 建立一个新的面板配置文件。
- Open: 打开一个已存在的面板配置文件。
- Save: 储存面板配置文件。
- Cut: 删除一个图形。
- Copy: 复制一个图形到剪贴板。
- Paste: 加入已复制的图形到控制板上。
- I: 面板信息对话框。
- S: 进入 LCD 仿真模式。

## LCD 面板图形文件

LCD 面板中的图形是一种位图文件 (.bmp)，用来表示面板上实际的图形和它们在面板上的位置，任何位图编辑器都可以用来建立此种文件。HT-LCDS 中 Edit 菜单的 Panel Editor 命令也可以用来设定 LCD 面板图形的信息。位图文件可有可无，即使 LCD 面板图形文件不存在，使用者也可以设定 LCD 面板信息。

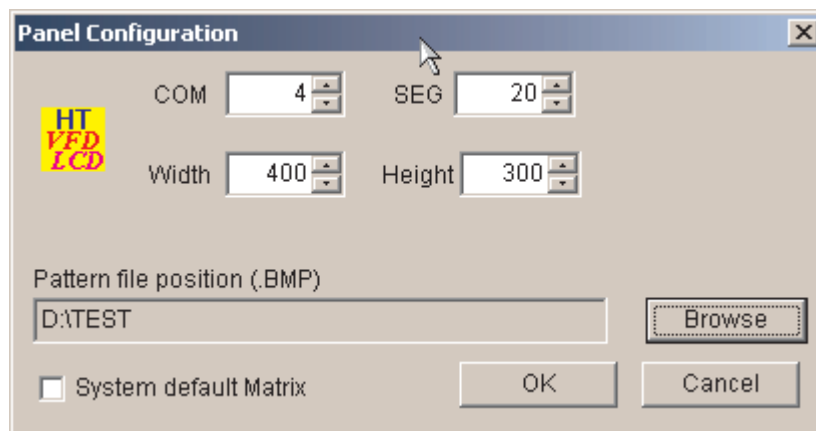
## 建立 LCD 面板配置文件

以下两个步骤用来建立一个面板配置文件。

- 设定面板的配置，包括 LCD 驱动器的 segment 和 common 的个数、面板的宽度和高度(以像素为单位)，也可以设定面板配置文件的路径以及点矩阵形式的面板。
- 选择图形和它们的位置，这将建立图形与 COM/SEG 位置间的关系。

### 建立面板的配置结构

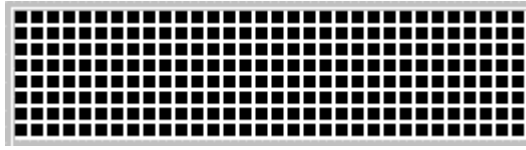
使用 HT-LCDS 中 File 菜单的 New 命令建立面板的配置信息。接下来会显示面板配置对话框。设定正确的 LCD 规格数据，如 COM/SEG 的个数、面板的宽度、高度和图形文件所在的路径，之后按下[OK]钮。完成面板配置的设置后，系统会回到 Tools 菜单的 LCD 仿真的对话框以便选取图形。



面板配置对话框

面板的配置包括：

- COM 和 SEG。设定 LCD 驱动器的 COMMON 个数和 SEGMENT 个数。会显示出单片机中 LCD 驱动器的默认个数，会在面板配置对话框显示时告知。必须确认这些个数和单片机中 LCD 驱动器的实际个数是一致的。
- 宽度和高度。这是面板显示在显示器上的尺寸。可以改变此项以调整面板画面的大小。
- 面板配置文件的路径。使用 Browse 钮选择面板配置文件所存放的路径或设定在项目的路径。
- 点矩阵形式。仿真点矩阵形式的 LCD 面板。下图显示点矩阵显示屏。



**注意：** 不要设定与实际 LCD 驱动器不相同的 COM 和 SEG 个数，否则会发生不可预测的结果。

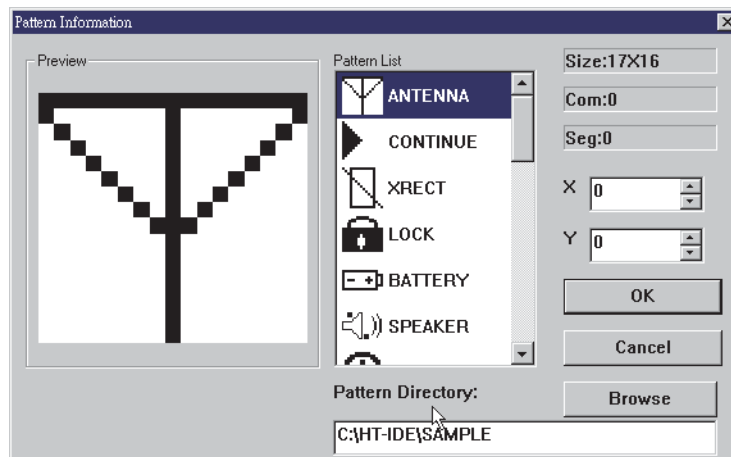
### 选择图形并设定位置

底下的步骤说明如何选择图形和位置。

- 使用 HT-LCDS 中 File 菜单的 New 命令建立一个新的面板配置文件。接着需要从 Pattern Information 对话框中选择图形以及设定 COM/SEG 位置。在加入新的图形一节中会有更详细的介绍。
- 使用 HT-LCDS 中 File 菜单 Open 命令打开一个存在的面板配置文件。所有的图形都会显示在面板显示框内，而图形文件的文件名会出现在 COM/SEG 的表格中相应的位置上。使用者可以加入，删除或改变图形信息、包括图形的文件名和位置。
- 使用 HT-LCDS 中 Edit 菜单的 Panel Editor 命令打开一个面板图形文件，如果这个面板图形文件已经被设定，便不需要选择图形，只需设定图形的位置。使用 Panel Editor 定义图形一节中将对此部分有详细的描述。

### 加入新的图形

- 将光标移动到 LCD 仿真器对话框中 COM/SEG 位置的格子中，双击鼠标，随后弹出图形信息对话框。Pattern List 列表框会将此项目路径中所有的图形文件 (.bmp) 排列显示。Size 区域显示所选图形的位图大小，Com 和 Seg 区域是这个图形被设定的 COM/SEG 位置，这三个区域的数据不能修改。
- 从 Pattern List 列表框中选择一个图形文件（位图文件）或单击 Browse 钮选择其它路径下的图形文件。HT-LCDS 使用双色的位图文件作为图形的图像源。预览窗口会将所选取的图形放大显示。
- 设定图形在面板显示框内的 X/Y 坐标位置。
- 按下[OK]钮回到 LCD 仿真对话框，然后点击 File 菜单的 Save 命令或再点击工具栏上的 Save 钮，面板文件的建立或修改就完成了。



图案信息对话框

### 删除图形

- 选取欲删除图形的 COM/SEG 并按下[Delete]键或点击工具栏上的 Cut 键。

### 改变图形

- 首先删除要改变的图形，然后加入一个新的图形以更换之。
- 另外一种方式，选定图形的 COM/SEG 位置并双击鼠标，图形信息对话框会出现，从这个对话框中的 Pattern List 列表框内选取一个图形并且按下[OK]键。

### 改变图形位置

- 可使用点击并拖拉(Select-drag-drop)的方法直接将图形移到面板显示框的适当位置。
- 或者，选择需要改变位置图形的 COM/SEG 位置并双击鼠标，随后弹出图形信息对话框，在这个对话框中的 X 及 Y 方格内设定图形的新位置并且按下[OK]键。

当以上的操作都已完成且系统回到 LCD 仿真对话框时，点击 HT-LCDS 中 File 菜单的 Save 命令或点击工具栏上的 Save 键，完成面板文件的建立或修改。

### 如何加入用户定义的矩阵

在 COM/SEG 个数与面板的 ROW/COL 个数不一致时，HT-LCDS 提供一个映射策略（File 菜单的 Import user matrix 命令）以帮助定义一个新的矩阵，例如：假设一个 2 COMs 和 6 SEGs 的 LCD 面板要使用 3 ROWs × 4 COLs 的矩阵，规划成下列的对应表：

COM0-SEG0	COM0-SEG1	COM0-SEG2	COM0-SEG3
COM1-SEG0	COM1-SEG1	COM1-SEG2	COM1-SEG3
COM0-SEG4	COM0-SEG5	COM1-SEG4	COM1-SEG5

上面的矩阵可用底下的定义文件定义：

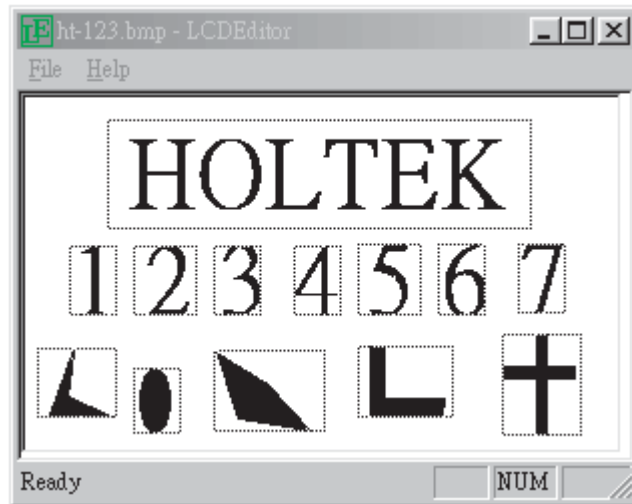
```

; MATRIX.DEF
; Comment line
ROW=3
COLUMN=4
; mapping syntax: ROW,COL=>COM,SEG
0,0 => 0,0 ; Map Row0 col0 to COM0 SEG0
0,1 => 0,1 ; Map Row0 col1 to COM0 SEG1
0,2 => 0,2 ; Map Row0 col2 to COM0 SEG2
0,3 => 0,3 ; Map Row0 col3 to COM0 SEG3
1,0 => 1,0 ; Map Row0 col0 to COM1 SEG0
1,1 => 1,1 ; Map Row0 col1 to COM1 SEG1
1,2 => 1,2 ; Map Row0 col2 to COM1 SEG2
1,3 => 1,3 ; Map Row0 col3 to COM1 SEG3
2,0 => 0,4 ; Map Row0 col0 to COM0 SEG4
2,1 => 0,5 ; Map Row0 col1 to COM0 SEG5
2,2 => 1,4 ; Map Row0 col2 to COM1 SEG4
2,3 => 1,5 ; Map Row0 col3 to COM1 SEG5
    
```



### 使用 Panel Editor 定义图形

HT-LCDS 提供一个完整的面板编辑接口来定义 LCD 面板的图形，如果已经存在整个面板的图形文件，就可以不用在面板上去设定所有图形的文件名，只需要去设定这些图形在面板上的位置。



LCD 编辑器

底下的步骤说明如何设定图形在 LCD 面板上的位置

- 在完成面板配置的设定之后，使用 Edit 菜单的 Panel Editor 命令进入 Panel Editor 窗口。
- 在 Panel Editor 窗口中使用 File 菜单的 Open 命令打开面板图形文件(.bmp)。

**注意：** Panel Editor 只支持双色的.BMP 文件(黑白)

- 面板图形文件中所有的图形会显示在 Panel Editor 窗口中。
- 双击鼠标或使用按下拖放的方法，在 COM/SEG 字段中设定显示的图形文件名。在 Save Pattern 对话框出现后就可以输入图形的数据。
- 针对面板上的每个图形重复执行上述的步骤。
- 在完成所有图形的数据设定后，回到 Panel Editor 窗口。再使用 File 菜单的 Save 命令将这些设定储存。
- 离开 Panel Editor 窗口回到 HT-LCDS 窗口，此时面板上会显示新的设定。

### 使用批处理文件将图形加入面板

通过 Edit 菜单的 Add Item Batch 命令，HT-LCDS 可从批处理文件中将图形加入面板。批处理文件是后缀名为.BTH 的文字文件，在批处理文件中每个图形项目需要定义图形文件的文件名和图形在面板中的位置。当使用 Menu 菜单的 Add Item Batch 命令选取一个批处理文件后，HT-LCDS 会将此批处理文件中描述的所有图形加入到面板中的指定位置。底下是一个.BTH 文件的例子。

```

; this is a comment line.
; item syntax: BMPfile.bmp, COM, SEG, X, Y
CRYSTAL.BMP,      0, 2, 120, 30
FION.BMP,         2, 3, 200, 50
CLIN.BMP,         3, 2, 130, 90
STEVE.BMP,        4, 4, 20, 040
    
```

### 选择 LCD 面板的颜色

HT-LCDS 提供调色对话框，可使用 HT-LCDS 中 Configure 菜单的 Set Panel Color 命令来选择面板的颜色。

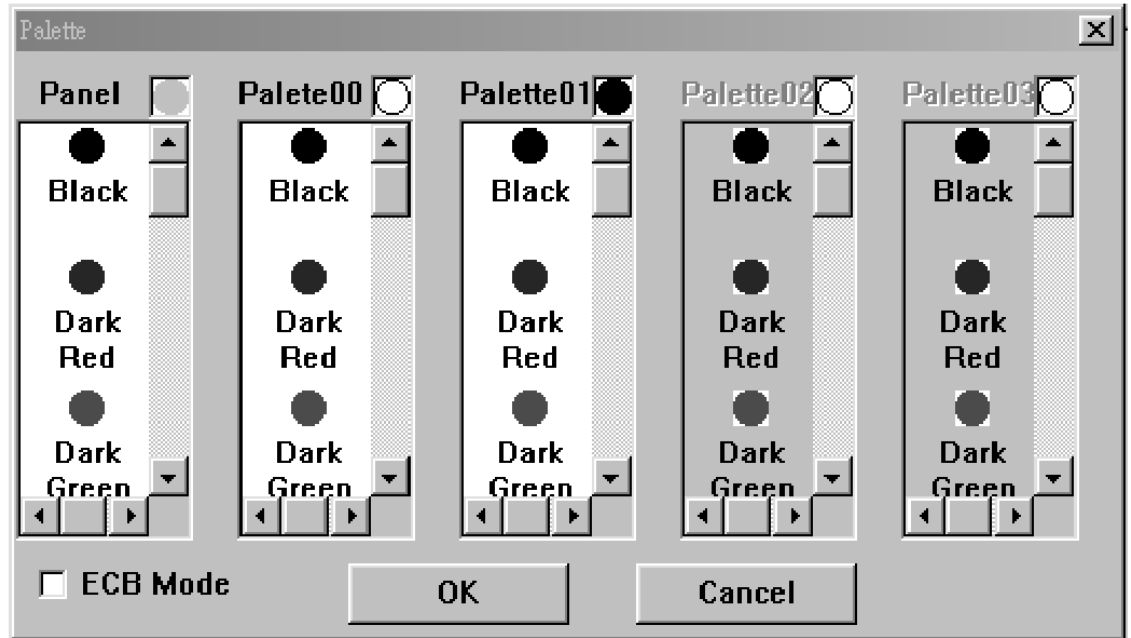


图 14-8

---

注意: ECB 模式只用于 HTG21X0

---

## LCD 仿真

在开始 LCD 仿真之前，确定 HT-LCDS 参照到正确的面板配置文件。通过选择 Tools 菜单中的 LCD Simulator 命令，即可进入 HT-LCDS 环境。

- 点击工具栏的 S 钮，HT-LCDS 会参阅对应的面板配置文件开始进行 LCD 的仿真。
- 如果打开了一个并非当前项目对应的面板配置文件，并点击工具栏的 S 钮，则 HT-LCDS 会参阅这个打开的面板配置文件并进行 LCD 的仿真。

当 HT-LCDS 开始仿真时将会显示 LCD 仿真器窗口，此时最新状态的 LCD 图形会出现在面板显示框之中。

### 停止仿真

用鼠标双击 LCD 仿真窗口的名称栏将使 HT-LCDS 回到编辑模式。





# 附录



## 附录 A

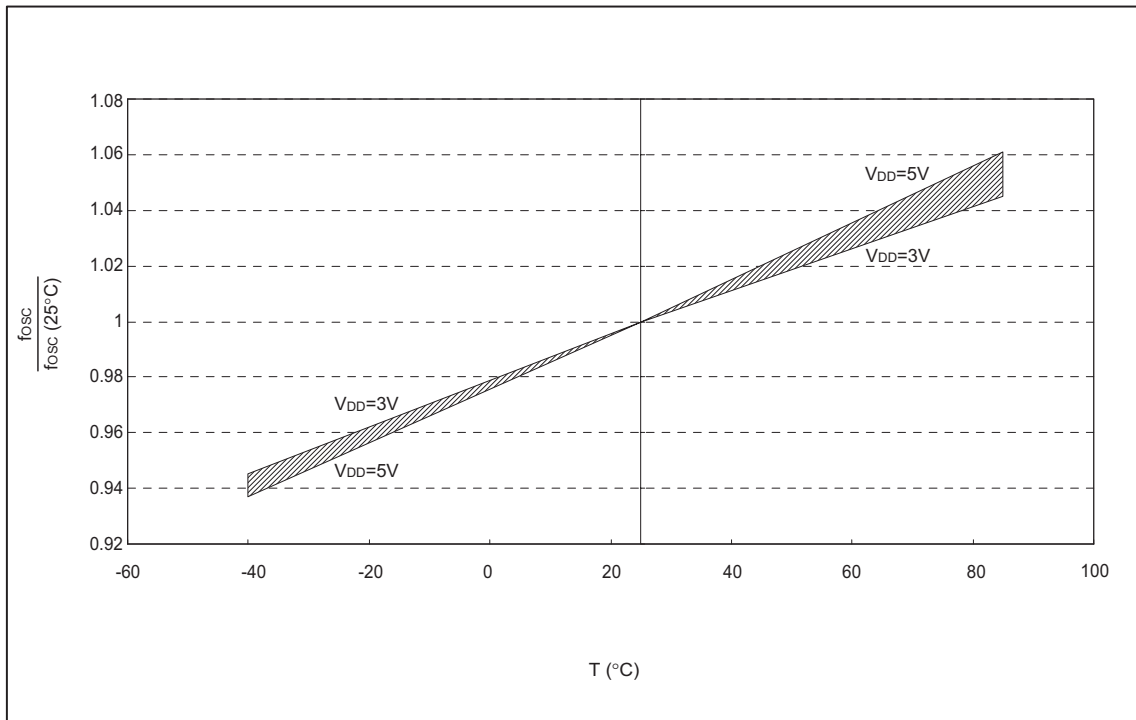
## 特性曲线图



下面的特性曲线图描述典型的单片机行为特性，此处所显示的是在某一时期，测试不同批号的产品所收集的统计数据，这个信息只提供参考，且其特性图不是在生产过程中被测试。

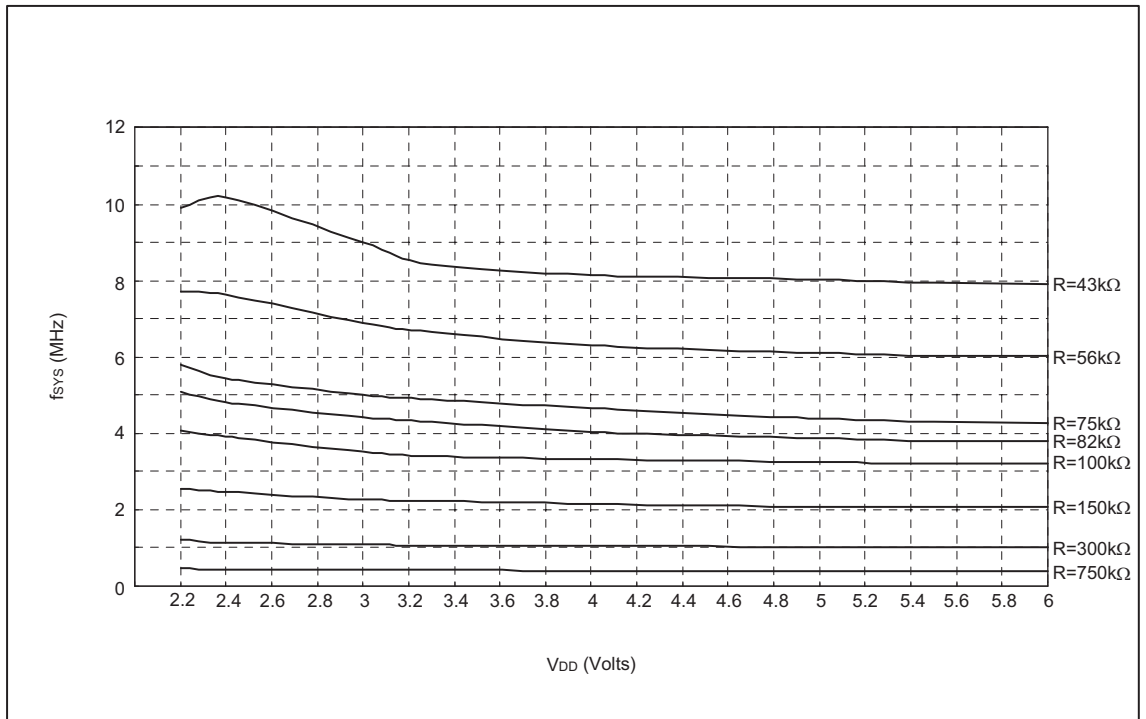
在一些特性图中，超过操作范围的数据只是为显示之用，单片机只在规格范围内会正常操作。

Typical RC OSC vs. Temperature

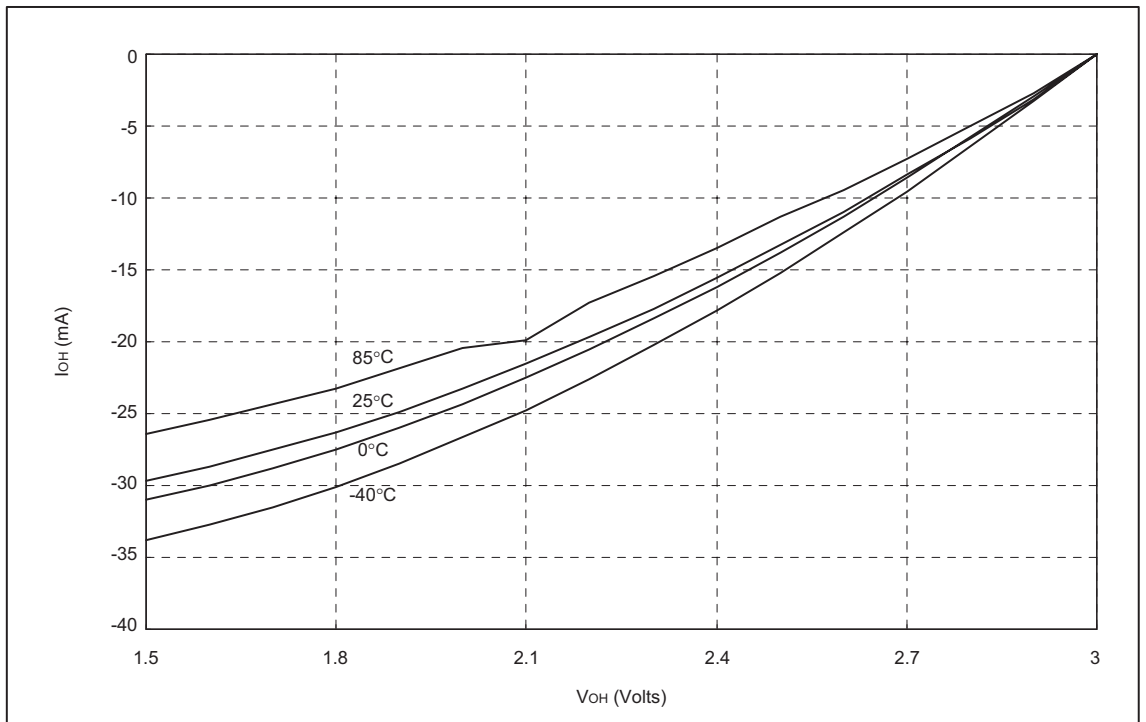




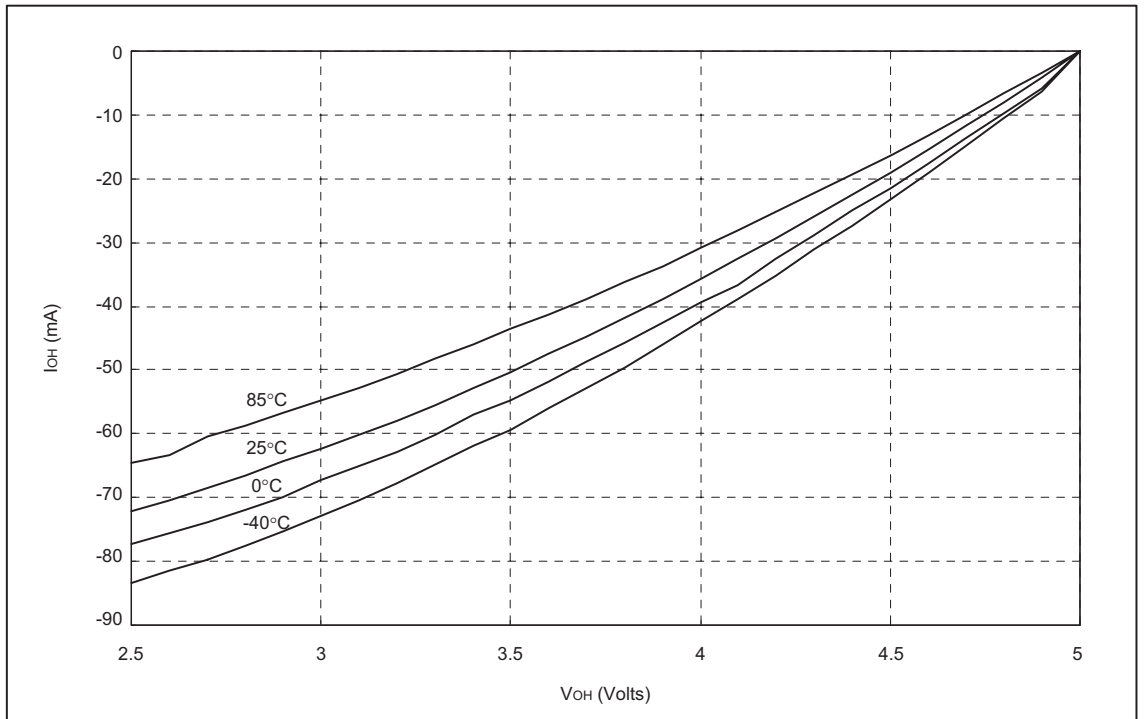
Typical RC Oscillator Frequency vs.  $V_{DD}$



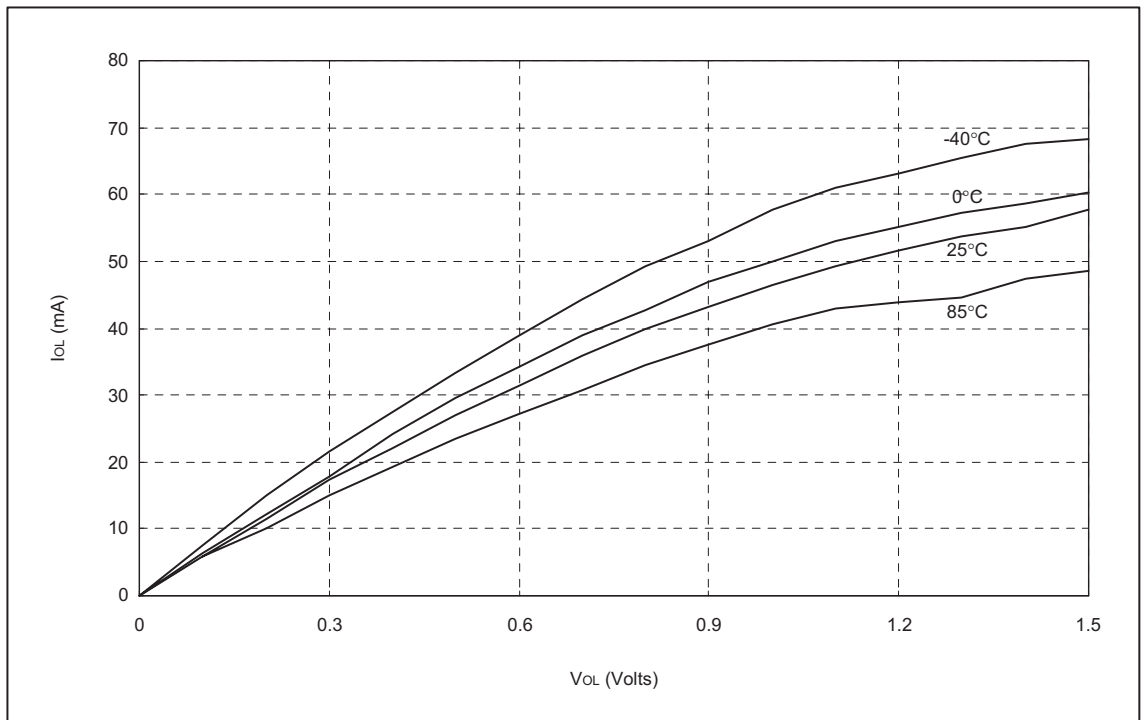
$I_{OH}$  vs.  $V_{OH}$ ,  $V_{DD}=3V$



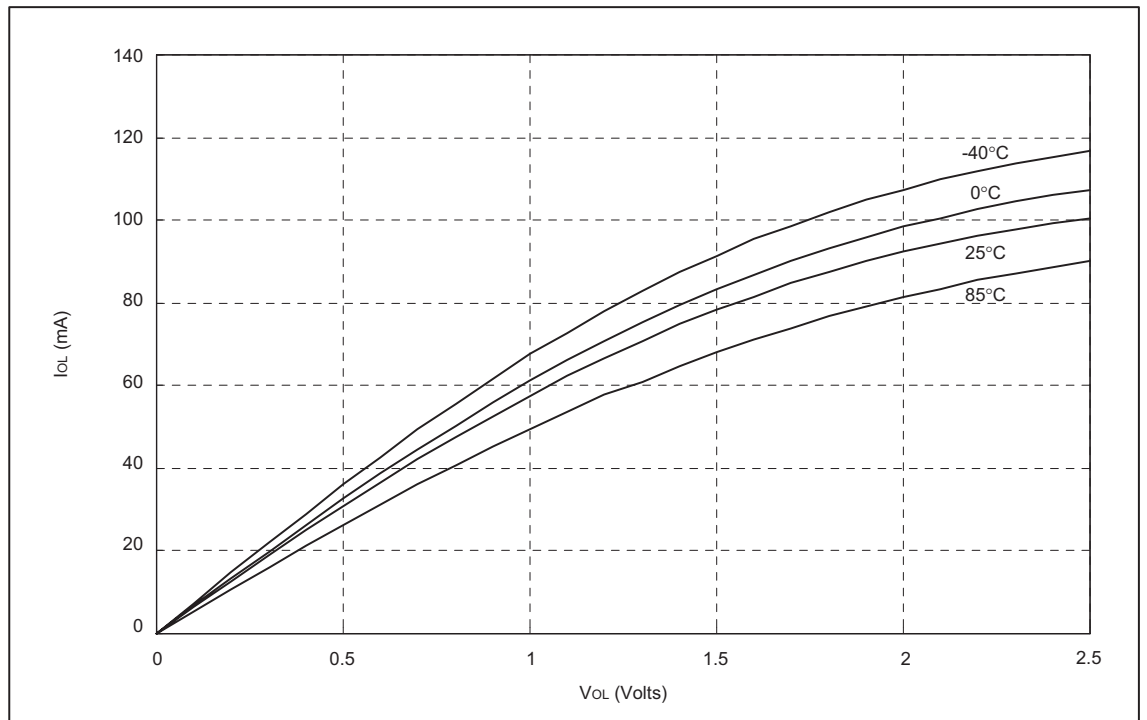
$I_{OH}$  vs.  $V_{OH}$ ,  $V_{DD}=5V$



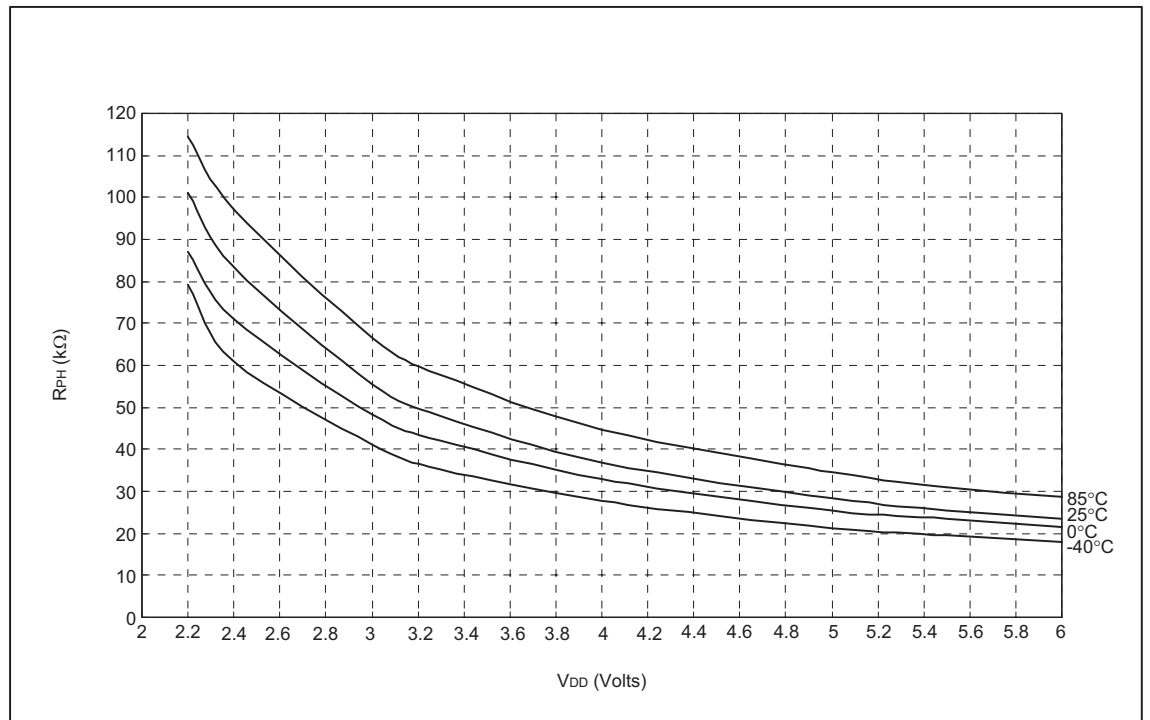
$I_{OL}$  vs.  $V_{OL}$ ,  $V_{DD}=3V$



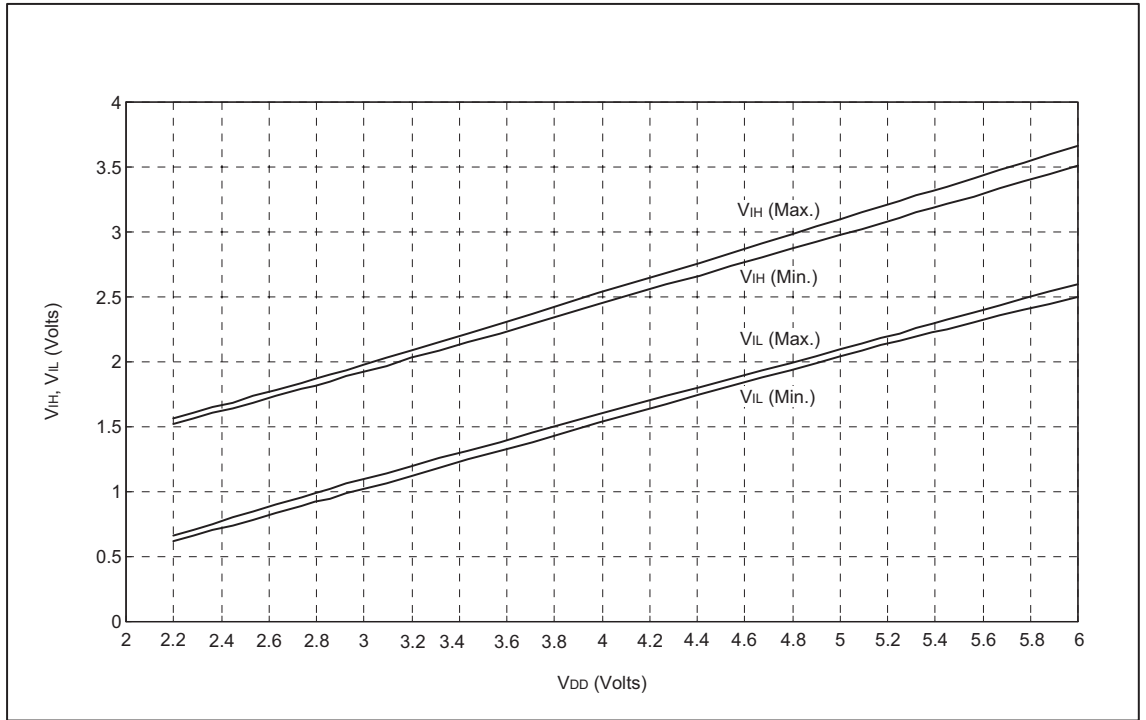
$I_{OL}$  vs.  $V_{OL}$ ,  $V_{DD}=5V$



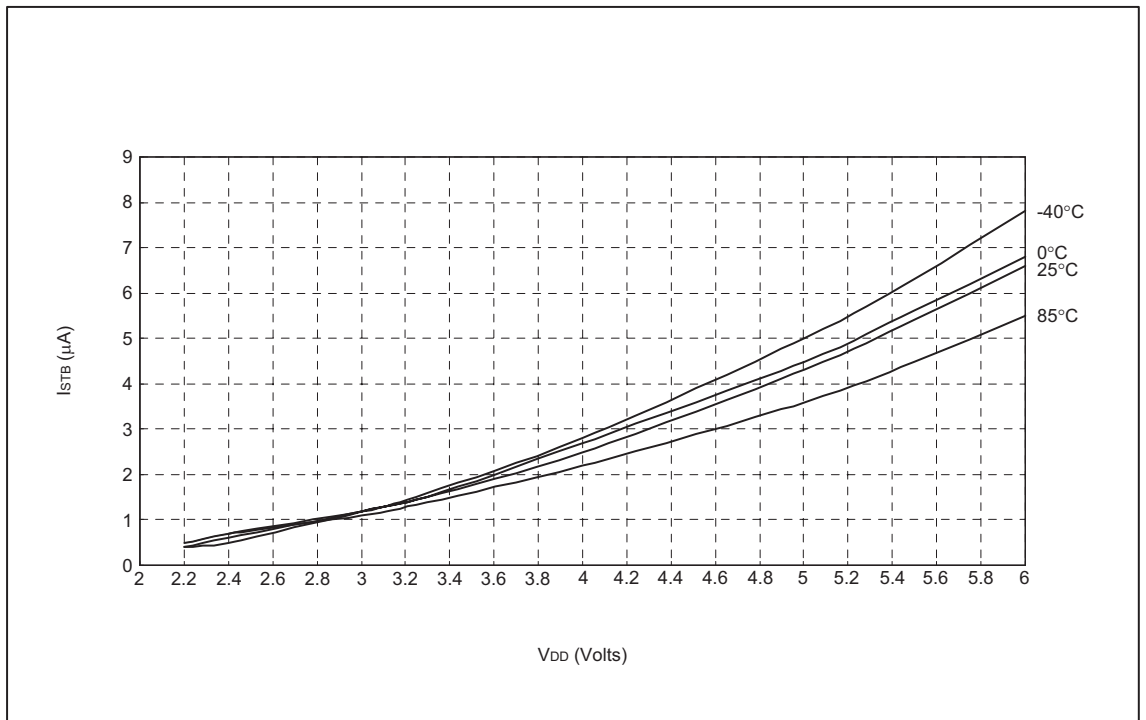
Typical  $R_{PH}$  vs.  $V_{DD}$



Typical  $V_{IH}$ ,  $V_{IL}$  vs.  $V_{DD}$  in  $-40^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$

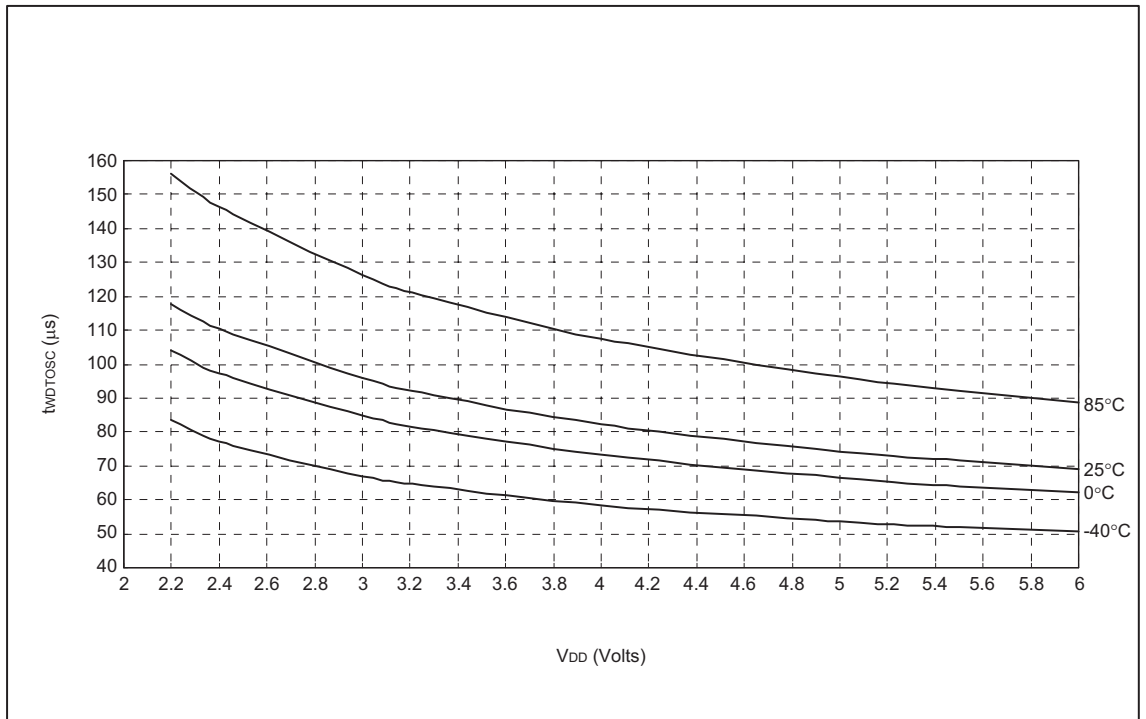


Typical  $I_{STB}$  vs.  $V_{DD}$  Watchdog Enable

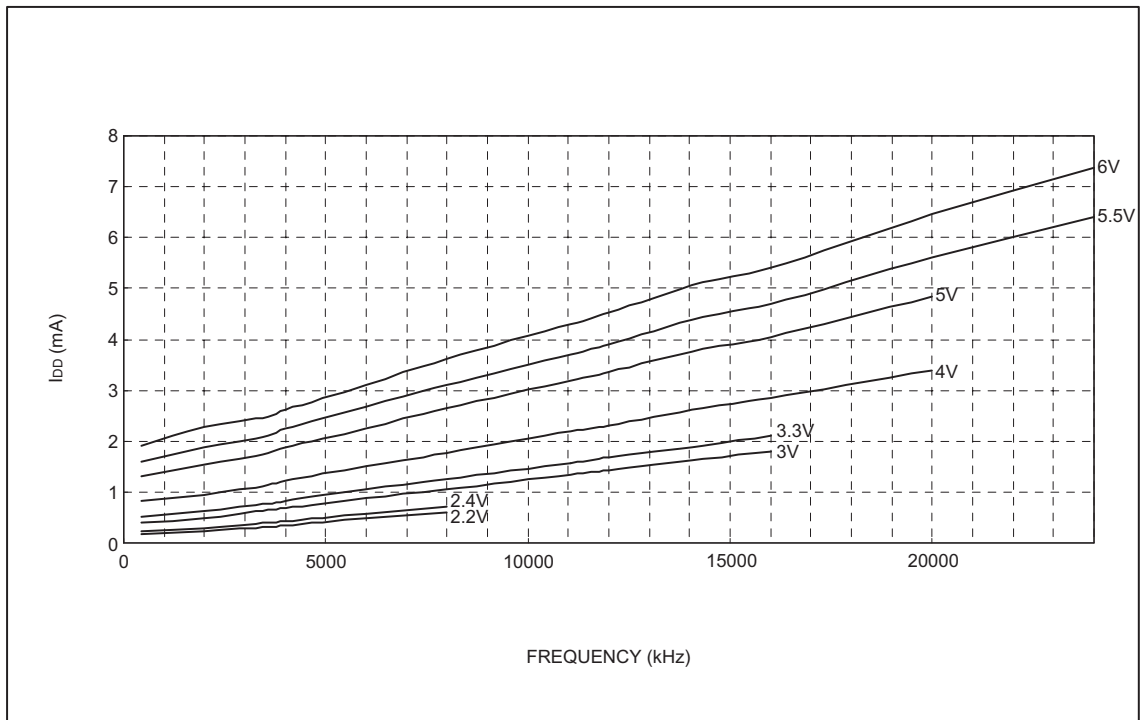




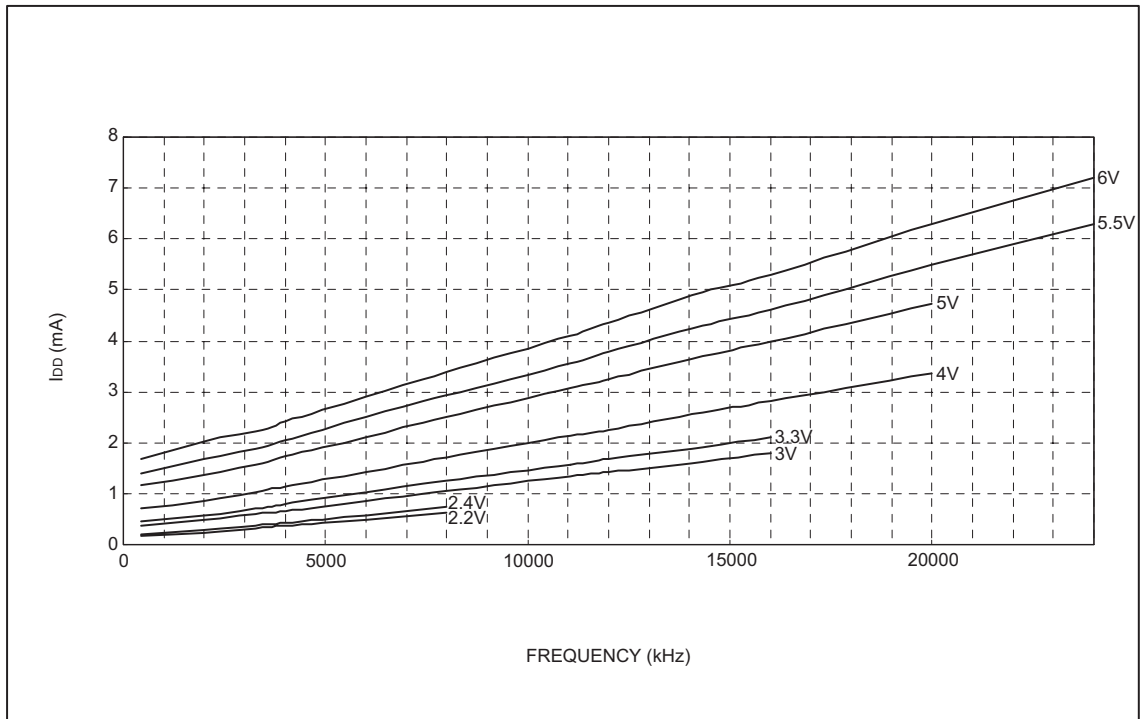
Typical  $t_{WDTO5C}$  vs.  $V_{DD}$



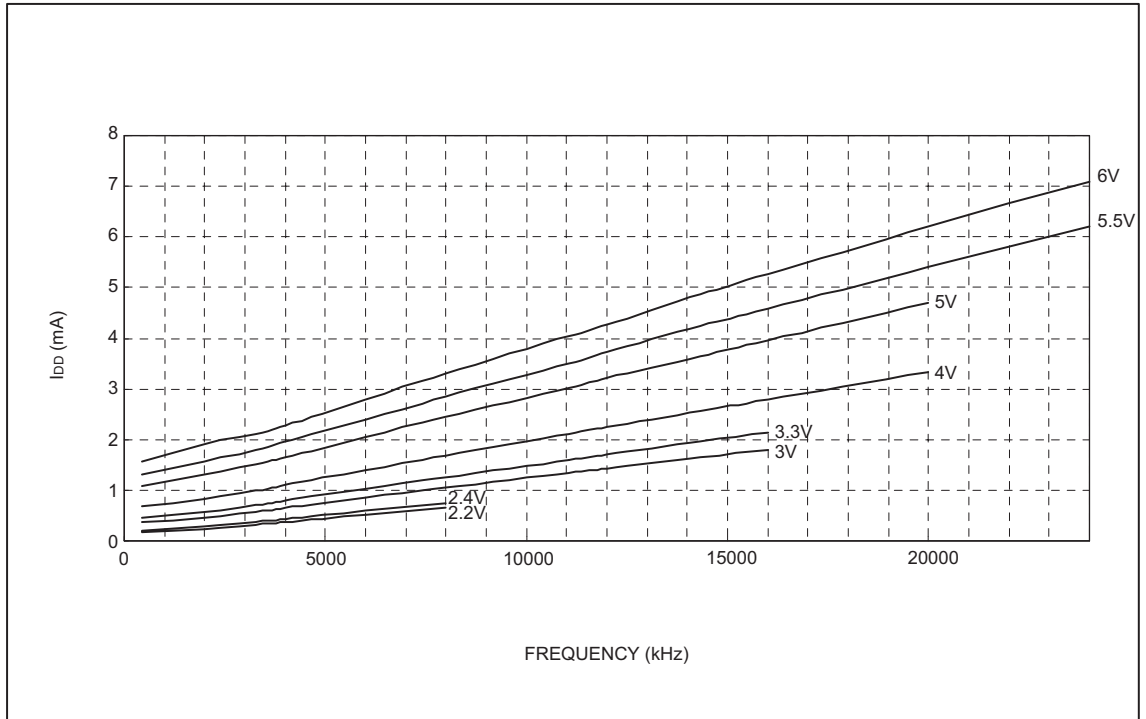
Typical  $I_{DD}$  vs. Frequency (External Clock,  $T_a = -40^\circ\text{C}$ )



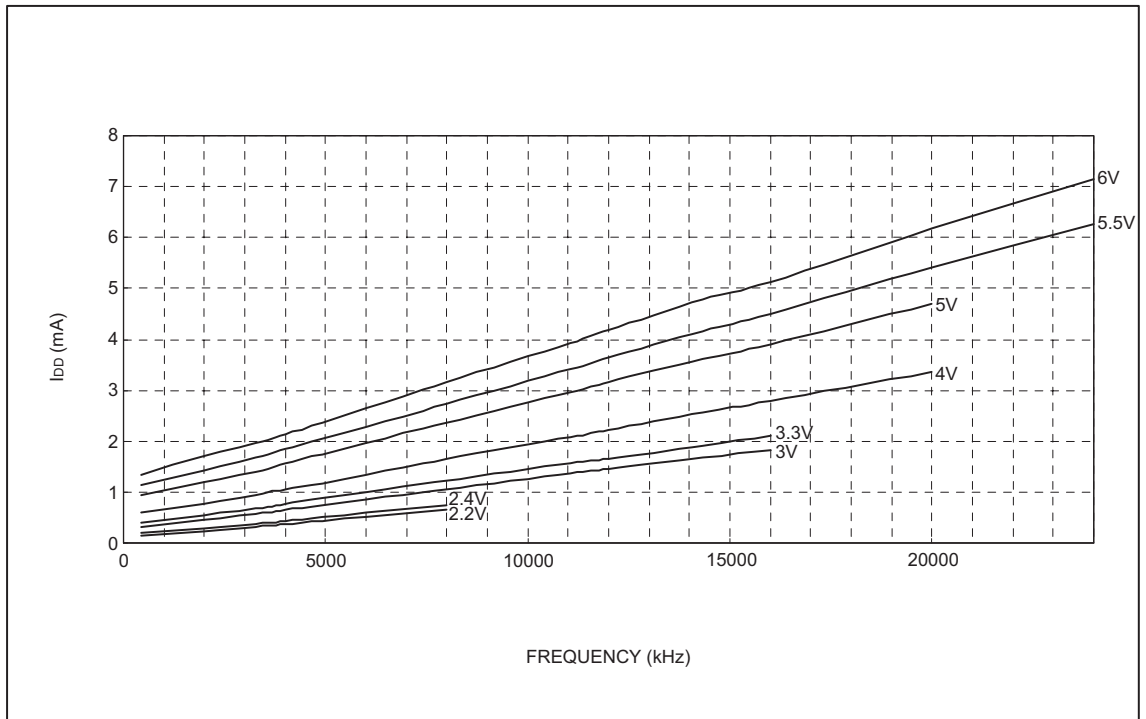
Typical  $I_{DD}$  vs. Frequency (External Clock,  $T_a=0^\circ\text{C}$ )



Typical  $I_{DD}$  vs. Frequency (External Clock,  $T_a=+25^\circ\text{C}$ )



Typical  $I_{DD}$  vs. Frequency (External Clock,  $T_a=+85^{\circ}\text{C}$ )



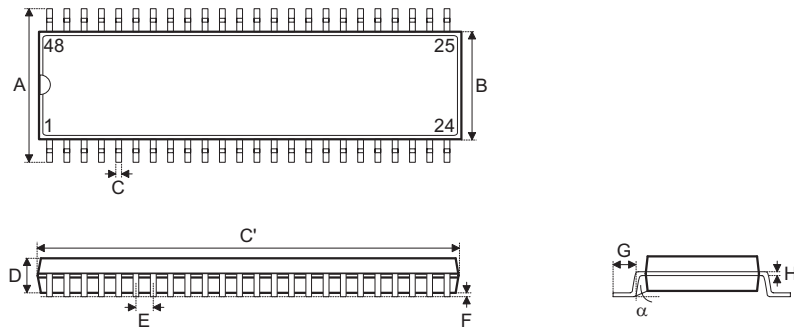


附录 B

封装信息

B

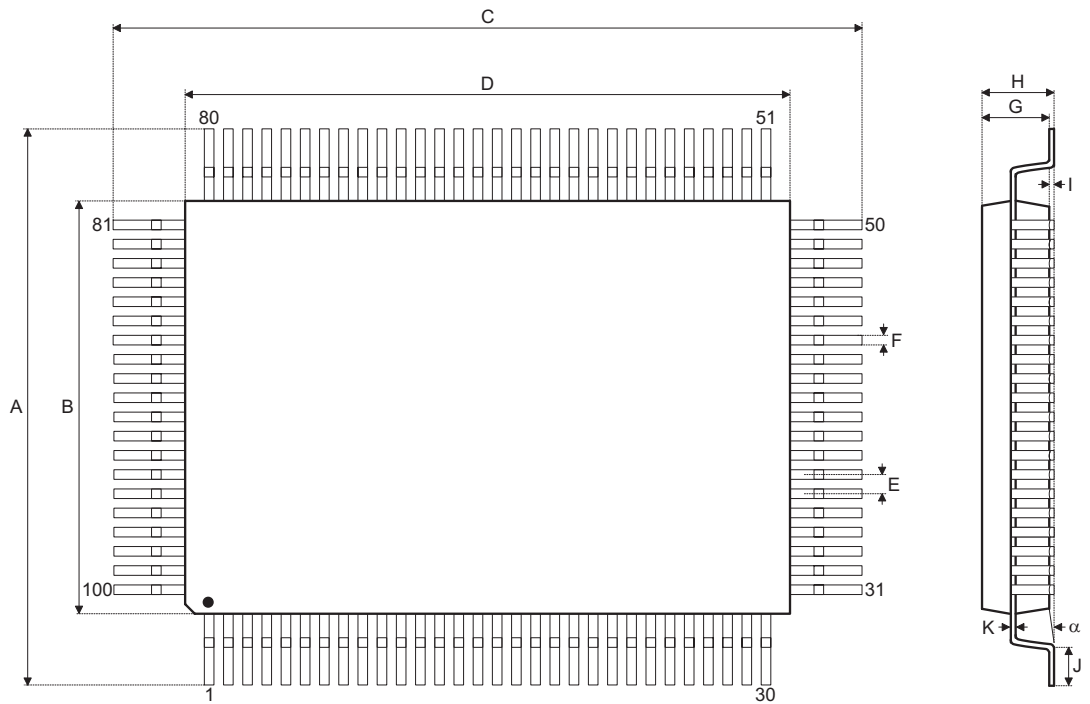
48-pin SSOP (300mil)外观尺寸



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	395	—	420
B	291	—	299
C	8	—	12
C'	613	—	637
D	85	—	99
E	—	25	—
F	4	—	10
G	25	—	35
H	4	—	12
$\alpha$	0°	—	8°



100-pin QFP (14×20)外观尺寸



Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	18.50	—	19.20
B	13.90	—	14.10
C	24.50	—	25.20
D	19.90	—	20.10
E	—	0.65	—
F	—	0.30	—
G	2.50	—	3.10
H	—	—	3.40
I	—	0.10	—
J	1	—	1.40
K	0.10	—	0.20
$\alpha$	0°	—	7°



**盛群半导体股份有限公司（总公司）**

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

**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

**盛扬半导体有限公司（深圳业务处）**

深圳市南山区科技园科技中三路与高新中二道交汇处生产力大楼A单元五楼 518057  
电话: 0755-8616-9908, 8616-9308  
传真: 0755-8616-9533  
ISDN: 0755-8615-6181

**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.holmate.com

Copyright © 2006 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应用。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>

