

## 盛群知识产权政策

### 专利权

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

### 商标权

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

### 著作权

Copyright © 2008 by HOLTEK SEMICONDUCTOR INC.

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

## 技术相关信息

- [工具信息](#)
- [问答集](#)
- [应用范例](#)
  - [HA0029S HT47R20A-1 时基 \(Time Base\) 使用介绍](#)
  - [HA0030S HT47R20A-1 实时时钟 \(Real Time Clock\) 使用介绍](#)
  - [HA0034S HT47R20A-1 蜂鸣器 \(Buzzer\) 使用介绍](#)
  - [HA0036S HT47R20A-1 可编程分频器 \(PFD\) 使用介绍](#)
  - [HA0045S HT47 MCU 区分介绍](#)

## 特性

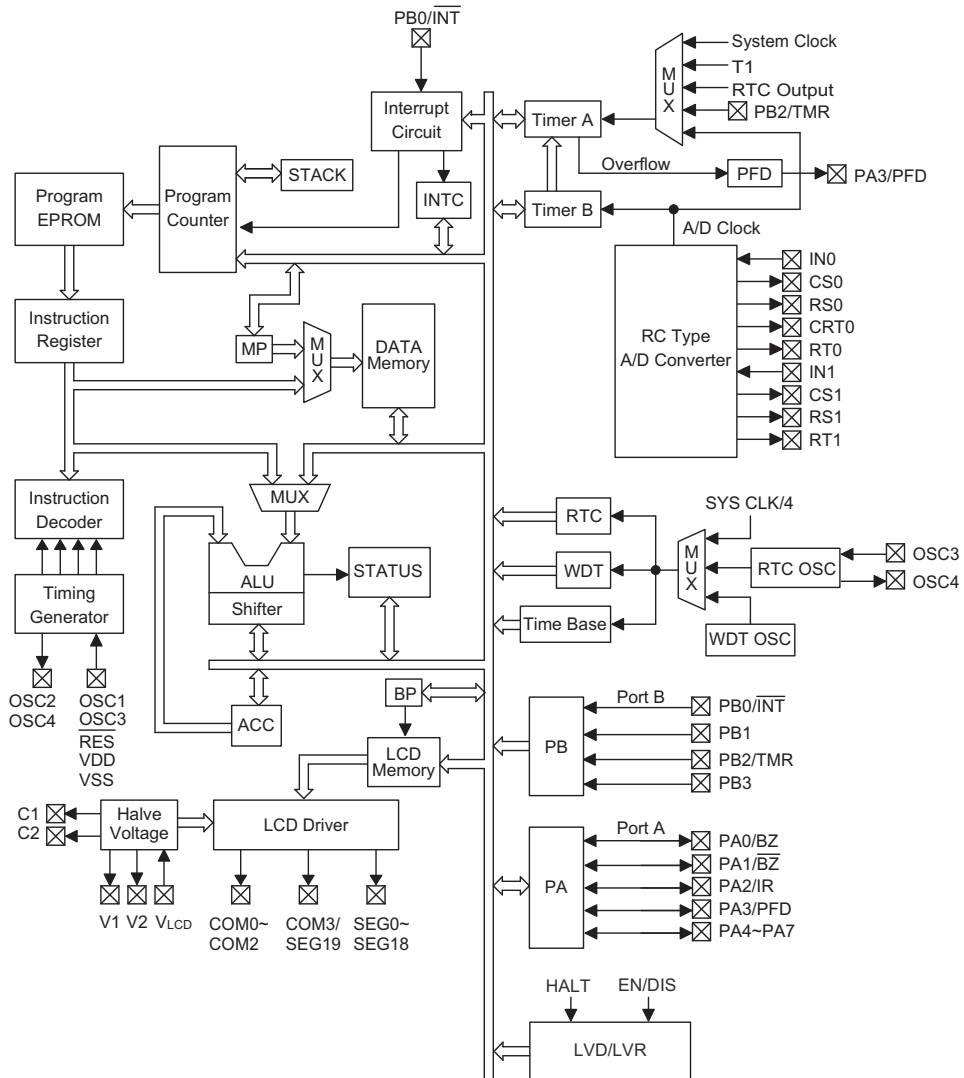
- 工作电压：2.2V~5.5V
- 8 个双向输入/输出口
- 4 个输入口
- 1 个外部中断输入口
- 1 个 16 位的可编程定时/计数器，并且具有 PFD(Programmable Frequency Divider)功能
- 内置晶体和 RC 振荡电路
- 内置 32768Hz 晶体振荡电路，可用于实时时钟或系统时钟
- 看门狗定时器
- 2K×16 程序存储器 ROM
- 64×8 数据存储器 RAM
- 一个实时时钟(RTC)
- 一个 8 位的实时时钟预分频器
- 低电压检测
- 低电压复位
- 一组蜂鸣器输出
- HALT 和唤醒功能可降低功耗
- 电容或电阻型 LCD 偏压方式
- 一个 20×2、20×3 或 19×4 段的液晶显示驱动电路
- 一个 IR 载波输出
- 两个 RC 型 A/D 转换通道
- 4 层硬件堆栈
- 位操作指令
- 查表指令，表格内容字长 16 位
- 当系统时钟为 4MHz 时，指令周期为 1μs
- 指令执行时间为 1 或 2 个指令周期
- 63 条指令
- 64-pin LQFP 封装

## 概述

HT47R20A-1/HT47C20-1 是 8 位高性能精简指令集单片机。专门为需要 A/D 转换的产品而设计，例如传感器信号输入。掩膜版本 HT47C20-1 与 OTP 版本 HT47R20A-1 引脚和功能完全相同。

低功耗、I/O 使用灵活、可编程分频器、计数器、振荡类型选择、2 通道 RC 型 A/D 转换、LCD 驱动、暂停和唤醒功能，使这款单片机可以广泛应用于电阻到频率转换的 A/D 测量，例如传感器测量、遥控测量、工业控制、消费类产品等系统中。

方框图





## 引脚说明

引脚名称	输入/输出	掩膜选项	功能说明
PA0/BZ PA1/ $\overline{\text{BZ}}$ PA2/IR PA3/PFD PA4~PA7	输入/输出	唤醒功能 上拉电阻 CMOS 或 NMOS	8 位双向输入/输出口。PA 低 4 位可由掩膜选项设置为 CMOS 或 NMOS 输出、带或不带上拉电阻(由上拉电阻选项决定)。NMOS 输出可以设置为带或不带上拉电阻的斯密特触发输入。NMOS 输出的每一位可由掩膜选项设为唤醒输入。PA0~PA1 可由掩膜选择设置为输入/输出或蜂鸣器输出。PA2 可由掩膜选择设置为输入/输出或 IR 载波输出。PA3 可由掩膜选项设置为输入/输出或 PFD 输出。
PB0/ $\overline{\text{INT}}$ PB1 PB2/TMR PB3	输入	—	4 位斯密特触发输入，具有上拉电阻。PB0 可由软件设置为输入引脚或外部中断输入引脚( $\overline{\text{INT}}$ )；PB2 可由软件设置为输入引脚或定时/计数器输入引脚。
VSS	—	—	负电源，接地。
IN1 CS1 RS1 RT1	输入 输出 输出 输出	—	振荡器输入引脚(通道 1) 参考电容连接引脚(通道 1) 参考电阻连接引脚(通道 1) 电阻传感器测量连接引脚(通道 1)
IN0 CS0 RS0 CRT0 RT0	输入 输出 输出 输出 输出	—	振荡器输入引脚(通道 0) 参考电容连接引脚(通道 0) 参考电阻连接引脚(通道 0) 电阻/电容传感器测量连接引脚(通道 0) 电阻传感器测量连接引脚(通道 0)
COM3/SEG19 COM2~COM0	输出	1/2 或 1/3 或 1/4 占空比	SEG19/COM3 可由掩膜选项设置为 LCD 显示的 Segment 或 Common 输出端。COM2~COM0 是 LCD 驱动的 Common 输出。
SEG18~SEG0	输出	—	LCD 驱动的 Segment 输出。
V1,V2,C1,C2	—	—	电压泵。
VLCD	输入	—	LCD 电源。
OSC4 OSC3	输出 输入	RTC 或系统 时钟	实时时钟振荡器 OSC3、OSC4 连接 32768Hz 的晶体振荡器，用于提供定时或系统时钟(由掩膜选项确定)。
VDD	—	—	正电源。
OSC2 OSC1	输出 输入	晶体或 RC	OSC1、OSC2 连接 RC 或晶体(由掩膜选项确定)以产生内部系统时钟。
$\overline{\text{RES}}$	输入	—	斯密特触发复位输入，低电平有效。
TEST1~3	输入	—	测试模式下输入引脚，正常使用时不必连接。

## 极限参数

电源供应电压.....  $V_{SS} -0.3V \sim V_{DD} 6.0V$   
 端口输入电压.....  $V_{SS} -0.3V \sim V_{DD} +0.3V$

储存温度..... $-50^{\circ}\text{C} \sim 125^{\circ}\text{C}$   
 工作温度..... $-40^{\circ}\text{C} \sim 85^{\circ}\text{C}$

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

## 直流电气特性

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

符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
V <sub>DD</sub>	工作电压	—	f <sub>SYS</sub> =4MHz	2.2	—	5.5	V
			f <sub>SYS</sub> =8MHz	3.3	—	5.5	V
I <sub>DD1</sub>	工作电流 (晶体振荡)	3V	无负载, f <sub>SYS</sub> =4MHz	—	0.7	1.5	mA
		5V		—	1.7	3	mA
I <sub>DD2</sub>	工作电流 (RC 振荡)	3V	无负载, f <sub>SYS</sub> =4MHz	—	0.7	1.5	mA
		5V		—	1.7	3	mA
I <sub>DD3</sub>	工作电流 (RC 振荡、晶体振荡)	5V	无负载, f <sub>SYS</sub> =8MHz	—	4	8	mA
I <sub>DD4</sub>	工作电流 (f <sub>SYS</sub> =32768Hz)	3V	无负载	—	0.25	0.5	mA
		5V		—	0.8	1.5	mA
I <sub>STB1</sub>	静态电流 (*f <sub>S</sub> =T1)	3V	无负载, 系统 HALT, HALT 时 LCD 关闭	—	—	1	μA
		5V		—	—	2	μA
I <sub>STB2</sub>	静态电流 (*f <sub>S</sub> =32768Hz)	3V	无负载, 系统 HALT, HALT 时 LCD 打开, 电容型偏压	—	2.5	5	μA
		5V		—	6	10	μA
I <sub>STB3</sub>	静态电流 (*f <sub>S</sub> =WDT RC 振荡)	3V	无负载, 系统 HALT, HALT 时 LCD 打开, 电容型偏压	—	2	5	μA
		5V		—	6	10	μA
I <sub>STB4</sub>	静态电流 (*f <sub>S</sub> =32768Hz)	3V	无负载, 系统 HALT, HALT 时 LCD 打开, 电阻型偏压, 1/2bias	—	17	30	μA
		5V		—	34	60	μA
I <sub>STB5</sub>	静态电流 (*f <sub>S</sub> =32768Hz)	3V	无负载, 系统 HALT, HALT 时 LCD 打开, 电阻型偏压, 1/3bias	—	13	25	μA
		5V		—	28	50	μA
I <sub>STB6</sub>	静态电流 (*f <sub>S</sub> =WDT RC 振荡)	3V	无负载, 系统 HALT, HALT 时 LCD 打开, 电阻型偏压, 1/2bias	—	14	25	μA
		5V		—	26	50	μA
I <sub>STB7</sub>	静态电流 (*f <sub>S</sub> =WDT RC 振荡)	3V	无负载, 系统 HALT, HALT 时 LCD 打开, 电阻型偏压, 1/3bias	—	10	20	μA
		5V		—	19	40	μA
V <sub>IL1</sub>	输入/输出、TMR 和 INT 的低电平输入电压	3V	—	0	—	0.3 V <sub>DD</sub>	V
		5V	—	0	—	0.3 V <sub>DD</sub>	V
V <sub>IH1</sub>	输入/输出、TMR 和 INT 的高电平输入电压	3V	—	0.7 V <sub>DD</sub>	—	V <sub>DD</sub>	V
		5V	—	0.7 V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>IL2</sub>	低电平输入电压(RES)	—	—	0	—	0.4 V <sub>DD</sub>	V
V <sub>IH2</sub>	高电平输入电压(RES)	—	—	0.9 V <sub>DD</sub>	—	V <sub>DD</sub>	V
I <sub>OL1</sub>	输入/输出灌电流	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	6	12	—	mA
		5V		10	25	—	mA
I <sub>OH1</sub>	输入/输出源电流	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-2	-4	—	mA
		5V		-5	-8	—	mA
I <sub>OL2</sub>	LCD Common 和 Segment 灌电流	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	210	420	—	μA
		5V		350	700	—	μA
I <sub>OH2</sub>	LCD Common 和 Segment 源电流	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-80	-160	—	μA
		5V		-180	-360	—	μA
I <sub>OL3</sub>	RC 振荡灌电流	3V	V <sub>OL</sub> =0.3V	5	10	—	mA
I <sub>OH3</sub>	RC 振荡源电流	3V	V <sub>OH</sub> =2.7V	-5	-10	—	mA

R <sub>PH</sub>	输入/输出口和 INT 上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
V <sub>LVR</sub>	低电压复位	—	—	2.5	3.2	3.6	V
V <sub>LVD</sub>	低电压检测	—	—	3.0	3.3	3.6	V

注: \*  $t_{SYS}=1/f_{SYS}$   
 \*\*为保护电压

**交流电气特性**

Ta=25°C

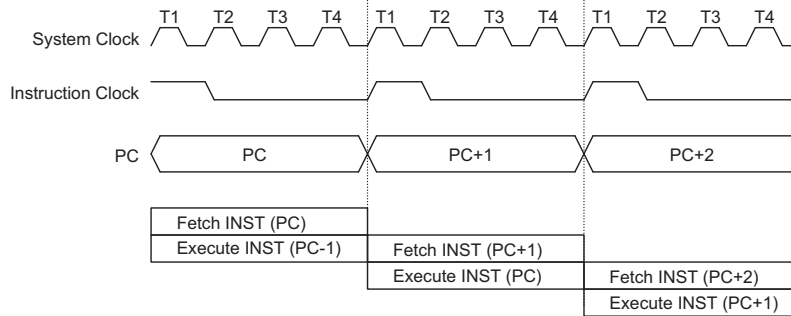
符号	参数	测试条件		最小	典型	最大	单位
		V <sub>DD</sub>	条件				
f <sub>SYS1</sub>	系统时钟(晶体振荡)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f <sub>SYS2</sub>	系统时钟(RC 振荡)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f <sub>SYS3</sub>	系统时钟 (32768Hz 晶体振荡)	—	—	—	32768	—	Hz
f <sub>RTCOSC</sub>	RTC 频率	—	—	—	32768	—	Hz
f <sub>TIMER</sub>	定时器输入频率(TMR)	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
t <sub>WDTOSC</sub>	看门狗振荡器	3V	—	45	90	180	μs
		5V	—	35	65	130	μs
t <sub>RES</sub>	外部复位低电平脉宽	—	—	1	—	—	μs
t <sub>SST</sub>	系统启动延迟时间	—	从 HALT 状态唤醒	—	1024	—	t <sub>SYS</sub>
t <sub>INT</sub>	中断脉冲宽度	—	—	1	—	—	μs

## 系统功能说明

### 指令执行时序

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

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



指令执行时序

### 程序计数器 — PC

11 位的程序计数器(PC)控制程序存储器 ROM 中指令执行的顺序，它可寻址整个 ROM 范围的 2048 个地址。

取得指令码以后，程序计数器会自动加一，指向下一个指令码的地址。但如果执行跳转、条件跳跃、向 PCL 赋值、子程序调用、初始化复位、内部中断、外部中断、子程序返回等操作时，PC 会载入与指令相关的地址而非下一条指令地址。

当遇到条件跳跃指令且符合条件时，当前指令执行过程中读取的下一条指令会被丢弃，取而代之的是一个空指令周期，随后才能取得正确的指令。反之，就会顺序执行下一条指令。

程序计数器的低字节(PCL)是一个可读写的寄存器(06H)。对 PCL 赋值将产生一个短跳转动作，跳转的范围为当前页 256 个地址。

当遇到控制转移指令时，系统也会插入一个空指令周期。

模式	程序计数器										
	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
初始化复位	0	0	0	0	0	0	0	0	0	0	0
外部中断	0	0	0	0	0	0	0	0	1	0	0
时基中断	0	0	0	0	0	0	0	1	0	0	0
实时时钟中断	0	0	0	0	0	0	0	1	1	0	0
定时/计数器中断	0	0	0	0	0	0	1	0	0	0	0
条件跳跃	PC+2										
装载 PCL	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
跳转、子程序调用	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
从子程序返回	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

程序计数器

注： \*10 ~ \*0：程序计数器位  
#10 ~ #0：指令代码位

S10 ~ S0：堆栈寄存器位  
@7 ~ @0：PCL 位

### 程序存储器 — EPROM

程序存储器用来存放要执行的指令代码，以及一些数据、表格和中断入口。程序存储器有 2048×16 位，程序存储器空间可以用程序计数器或表格指针进行寻址。

以下列出的程序存储器地址是系统专为特殊用途而保留的：

- 地址 000H

该地址为程序初始化保留。系统复位后，程序总是从 000H 开始执行。



输入/输出寄存器(PA; 12H, PB; 14H), 中断控制寄存器 1(INTC1; 1EH), 定时/计数器 A 高、低位字节寄存器(TMRAH; 20H, TMRAL; 21H), 定时/计数器控制寄存器(TMRC; 22H)。定时/计数器 B 高、低位字节寄存器(TMRBH; 23H, TMRBL; 24H), RC 振荡型 A/D 转换控制寄存器(ADCR; 25H)。其余在 40H 之前的空间保留给系统以后扩展使用, 读取这些地址的返回值为“00H”。通用数据寄存器地址从 40H 到 7FH, 用来存储数据和控制信息。

所有的数据存储器单元都能直接执行算术、逻辑、递增、递减和循环操作。除了一些特殊位外, 数据存储器的每一位都可通过“SET[m].i”置位或由“CLR[m].i”复位。而且都可以通过间接寻址指针 MP0、MP1 进行间接寻址。

### 间接寻址寄存器

地址 00H 和 02H 是间接寻址寄存器, 并无实际的物理区存在。任何对[00H]和[02H]的读/写操作, 都是访问由 MP0(01H)和 MP1(03H)所指向的 RAM 单元。间接读取 00H 或 02H 地址得到的值为 00H, 间接写入此地址, 不会产生任何操作。

间接寻址寄存器之间不支持数据传送功能。间接寻址指针 MP0 和 MP1 是 8 位寄存器。

MP0 只能用于数据存储器, 而 MP1 能用于数据存储器 and LCD 显示存储器。

### 累加器

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

### 算术逻辑单元 — ALU

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

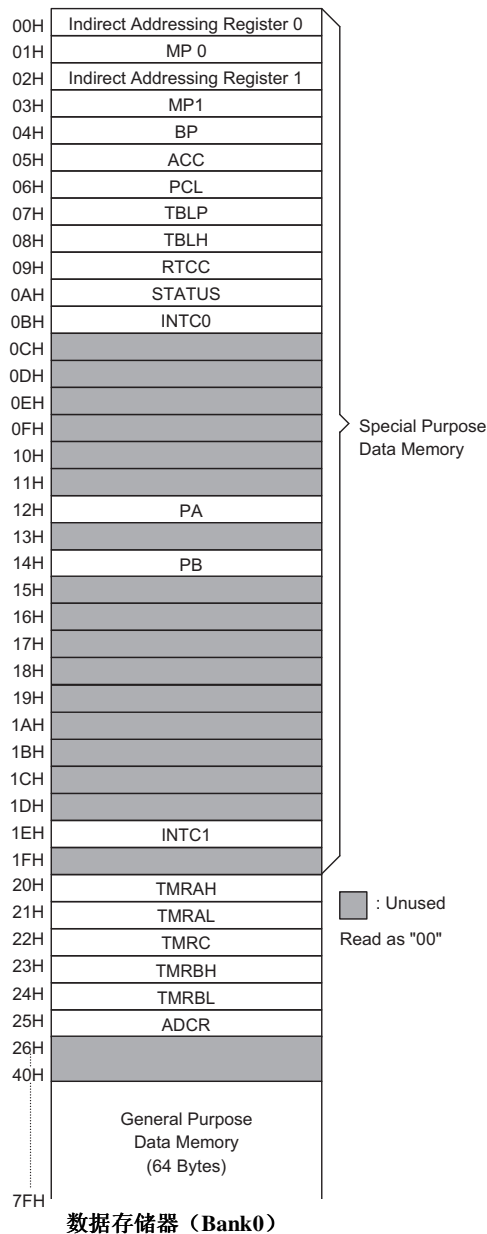
- 算术运算(ADD, ADC, SUB, SBC, DAA)
- 逻辑运算(AND, OR, XOR, CPL)
- 移位运算(RL, RR, RLC, RRC)
- 递增和递减(INC, DEC)
- 分支判断(SZ, SNZ, SIZ, SDZ...)

ALU 不仅可以储存数据运算的结果, 还会改变状态寄存器的值。

### 状态寄存器 — STATUS

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

符号	位	功能
C	0	如果在加法运算中结果产生了进位或在减法运算中结果不产生借位, 则 C 被置位; 反之, C 被清除。它也可被循环移位指令影响。
AC	1	如果在加法运算中低 4 位产生了进位或减法运算中低 4 位不产生借位, 则 AC 被置位; 反之, AC 被清除。
Z	2	如果算术或逻辑运算的结果为零, 则 Z 被置位; 反之, Z 被清除。
OV	3	如果运算结果向最高位进位, 但最高位并不产生进位输出, 则 OV 被置位, 反之亦然; 否则, OV 被清除



PDF	4	系统上电或执行“CLR WDT”指令，PDF 被清除；执行“HALT”指令，PDF 被置位。
TO	5	系统上电、执行“CLR WDT”或“HALT”指令，TO 被清除；WDT 定时溢出，TO 被置位。
—	6	未用，读出为“0”
—	7	未用，读出为“0”

**状态寄存器 (0AH)**

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

标志位 Z、OV、AC 和 C 反映的是最近一次操作的状态。

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

### 中断

HT47R20A-1/HT47C20-1 提供了一个外部中断、一个内部定时器/计数器中断、一个内部时基中断和一个内部实时时钟中断。中断控制寄存器 0(INTC0; 0BH)和中断控制寄存器 1(INTC1; 1EH)包含了所有中断控制位，用来控制中断允许/禁止以及中断请求标志。

位	标志	功能
0	EMI	控制总中断(1=允许; 0=禁止)
1	E EI	控制外部中断(1=允许; 0=禁止)
2	ETBI	控制时基中断(1=允许; 0=禁止)
3	ERTI	控制实时时钟中断(1=允许; 0=禁止)
4	EIF	外部中断请求标志(1=请求; 0=无)
5	TBF	时基中断请求标志(1=请求; 0=无)
6	RTF	实时时钟中断请求标志(1=请求; 0=无)
7	—	未定义，读取值为“0”

**INTC0 (0BH) 寄存器**

位	标志	功能
0	ETI	控制定时/计数器中断(1=允许; 0=禁止)
1~3	—	未定义，读取值为“0”
4	TF	定时/计数器中断请求标志(1=请求; 0=无)
5~7	—	未定义，读取值为“0”

**INTC1 (1EH) 寄存器**

只要有中断子程序被服务，其余的中断全部被禁止(通过清除 EMI 位)，这种做法的目的在于防止中断嵌套。在执行中断服务子程序期间，可能会发生其它的中断请求，但只有中断请求标志会被记录下来。如果要实现中断嵌套，使用者可以在中断子程序中置位 EMI 及 INTC0、INTC1 中对应的中断控制位。如果堆栈已满，即使有关的中断允许，该中断申请并不会被响应，一直到堆栈指针(SP)发生递减才会响应。如果需要立即服务，应避免让堆栈发生饱和。

所有的中断都具有唤醒功能，当有中断服务请求，会将程序计数器值压栈，然后再转至中断服务程序的入口。但要记住这时只有程序计数器的内容被压栈，如果寄存器和状态寄存器的内容被中断程序改变，而导致该中断服务程序会破坏预期控制序列的话，使用者应该事先将这些数据储存起来。

外部中断是由引脚从高电位转变为低电位时触发的，其相关的中断申请标志位(EIF)接着就会被置位。如果外部中断允许，而且此时堆栈尚未用满，而外部中断被触发时，会调用 04H 地址的子程序。该中断申请标志(EIF)和 EMI 位也因此而清除，以便禁止其他中断。

内部定时/计数器中断是由置位定时/计数器中断申请标志(TF; INTC1 的第 4 位)的方式来启动的，而

该中断申请标志是由定时器 A 或定时器 B 溢出产生的。如果中断允许，而此时堆栈尚未用满，而且 TF 也已置位的话，会调用地址 10H 子程序。此时不仅其相关的中断申请标志(TF)会被清除，而且 EMI 位也会被清除，以禁止其他中断。

时基中断是由置位时基中断申请标志位(TBF)的方式来启动的，而该中断申请标志是由时基信号产生的。如果中断允许，而此时堆栈尚未用满，而且 TBF 也已置位的话，会调用地址 08H 子程序，此时不仅其相关的中断申请标志(TBF)会被清除，而且 EMI 位也会被清除，以禁止任何进一步的中断。

实时时钟中断是由置位实时时钟中断申请标志位(RTF; INTC0 的第 6 位)的方式来启动的，而该中断申请标志是由实时时钟产生的。如果中断允许，而此时堆栈尚未用满，而且 RTF 也已置位的话，会调用地址 0CH 子程序。此时不仅其相关的中断申请标志(RTF)会被清除，而且 EMI 位也会被清除，用以禁止任何进一步的中断。

执行中断子程序期间，其他的中断响应会被暂停，直到执行 RETI 指令或是将 EMI 位和其相关中断控制位设为“1”(此时堆栈尚未用满)为止。若欲从此中断子程序返回，只要执行 RET 或 RETI 指令即可。其中，RETI 会设定 EMI 位，用以允许中断服务，而 RET 则否。

如果在两个连续的 T2 脉冲上升沿之间发生中断，则在这两个 T2 后面的 T2 脉冲该中断会被服务。而如果同时发生中断申请，其顺序会依照下表所显示，这个顺序可以通过设定各中断相关的控制位来改变顺序。

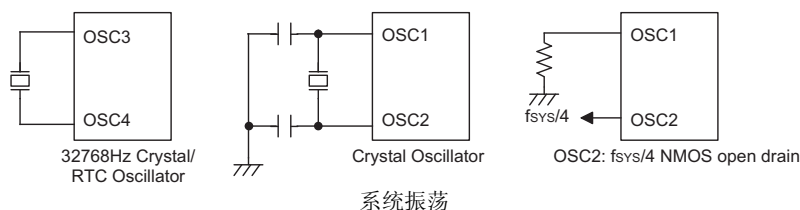
中断源	优先级	中断向量
外部中断	1	04H
时基中断	2	08H
实时时钟中断	3	0CH
定时/计数器中断	4	10H

中断控制寄存器 0(INTC0)是由外部中断申请标志位(EIF)、实时时钟中断申请标志位(RTF)、时基中断申请标志位(TBF)、外部中断允许位(EEI)、实时时钟中断允许位(ERTI)、时基允许位(ETBI)以及总中断允许位(EMI)组成的，其地址为数据存储器的 0BH。中断控制寄存器 1(INTC1)是由定时/计数器中断申请标志位(TF)、定时/计数器中断允许位(ETI)组成的，其地址为数据存储器的 1EH。EMI、EEI、ETI、ETBI 和 ERTI 是用来控制中断的允许/禁止状态。这些控制位可以防止正在进行中断服务时的中断申请。一旦中断申请标志(RTF、TBF、TF、EIF)置位之后，仍会继续保留在 INTC0 或 INTC1 寄存器中，直到全部中断都被服务或用软件指令清除为止。

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

## 振荡电路

HT47R20A-1/HT47C20-1 提供 3 种振荡器电路给系统时钟，根据掩膜选择分别为：RC 振荡器，晶体振荡器和 32768Hz 的振荡器。无论选择何种振荡源，其信号都提供系统时钟。HALT 模式停止系统振荡器(RC 和晶体振荡器)并忽略外部信号以节省电源。如果选择 32768Hz 的振荡器作为系统振荡器，那么在 HALT 模式下就不会停止系统振荡器，但是停止执行指令。由于这个振荡器(用于系统振荡器或 RCT 振荡器)也用于提供定时作用，因此在 HALT 模式下内部的定时操作(RTC，时基，WDT)仍然运行。



在这 3 种振荡器中，如果使用了 RC 振荡器则要在 OSC1 和 VSS 之间接一个外接的电阻，电阻的阻值范围为 24 kΩ到 1MΩ。从 OSC2 输出系统时钟 4 分频的信号，用于给外部逻辑器件提供同步时钟信号。RC 振荡器提供了最廉价的解决方案，然而振荡器的频率会随着 VDD，温度和芯片自身的参数发生变化。因此，它不适合需要精确的振荡器频率的计时操作的场合。

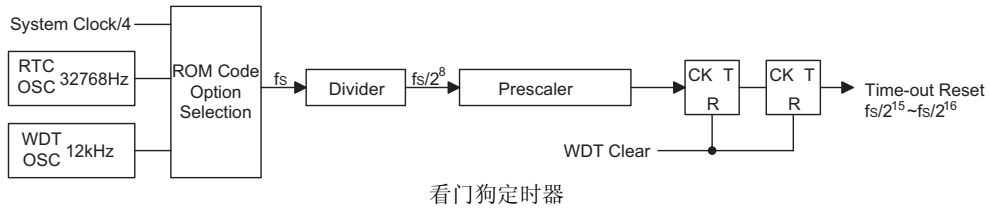
另一方面，如果选择了晶体振荡器，则要在 OSC1 和 OSC2 之间接一个晶体，用来提供晶体振荡器所需的回馈和相移，但除此之外就不再需要其它任何外部元件了。另外，可在 OSC1 与 OSC2 之间接一个陶瓷谐振器(Resonator) 来取代晶体振荡器用来得到频率产生，但是需要分别在 OSC1 和 OSC2 外接一个电容器。

还有一个振荡电路是专门设计给实时时钟(RTC)使用，只能采用 32.768KHZ 的晶体振荡器，只需把晶体连接 OSC3 与 OSC4 之间。RTC 振荡器，可由置位“QOSC”位(RTCC 的第四位)为 1，加快起振速度。建议在电源开启之后，启动加快起振功能，并在两秒后关闭它。

WDT 振荡器为一个 IC 内部自由振荡的 RC 型振荡器，并不需要连接任何外部元件。当系统进入暂停模式时，系统时钟会立即关闭，但 WDT 振荡器仍会运作。(其振荡周期大约为 90μs@3V)。在掩膜选择时，如欲节省电源，可关闭 WDT 振荡器。

### 看门狗定时器 — WDT

WDT 的时钟来源可以在掩膜选择时，置成由专用的 RC 振荡器(WDT 振荡器)、指令时钟(系统时钟 4 分频)或实时时钟振荡器(RTC)来提供。WDT 是用来防止程序的不正常运行或是跳到未知或不希望去的地址，而导致不可见的结果。WDT 可以被掩膜选择禁止。如果在关闭状态，所有与 WDT 有关的指令操作都是没有作用的。



如果 WDT 时钟源为内部 WDT 振荡器的话，那么溢出时间会因为温度、VDD 以及芯片参数的变化而变化。如果选择了指令时钟为时钟源，在 HALT 状态时，WDT 会停止计数而失去保护功能。只能由外部逻辑来重新启动系统。

因此若单片机工作在干扰很大的环境中，强烈建议使用片内的 RC 振荡器(WDT OSC)，因为 HALT 模式会使系统时钟停止运作。

看门狗定时器溢出在正常操作时不仅启动系统复位(Chip Reset)，并置位状态位 TO。在暂停模式中，这溢出会启动热复位(Warm Reset)，但只有程序计数器和状态指针会复位为零。要清除 WDT 的值可以有三种方法：外部复位(低电平输入到 RES 端)、清除看门狗指令或 HALT 指令。清除看门狗指令有“CLR WDT”和“CLR WDT1”、“CLR WDT2”二组指令。这两组指令中，只能选择其中一组，由掩膜选项决定。如果选择“CLR WDT”，那么只要执行“CLR WDT”指令就会清除 WDT。如果选择“CLR WDT1”和“CLR WDT2”，那么二条指令要交替使用才会清除 WDT，否则，WDT 会由于溢出而使系统复位。

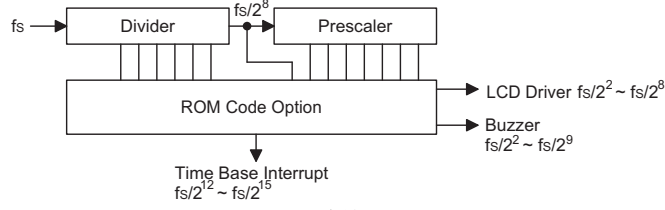
看门狗定时器的溢出时间周期为  $f_s/2^{15} \sim f_s/2^{16}$ 。因为“CLR WDT”或“CLR WDT1”和“CLR WDT2”指令只能清除 WDT 的最后两级锁存器。

### 多功能计时器

HT47R20A-1/HT47C20-1 包含一个多功能定时器，供看门狗定时器(WDT)、时基、实时时钟产生不同超时时间周期。此多功能定时器由一个八阶除法器及一个七位预分频器所组成，使用的时钟来自 WDT OSC、RTC OSC 或指令时钟(系统时钟四分频)。多功能定时器为 LCD 驱动电路提供可选择的频率信号(范围从  $f_s/2^2 \sim f_s/2^8$ )，并为蜂鸣器输出电路提供可选择的频率信号(范围从  $f_s/2^2 \sim f_s/2^9$ )，频率由掩膜选择。为了正确地显示，建议选择 4KHz 作为 LCD 驱动信号。

### 时基

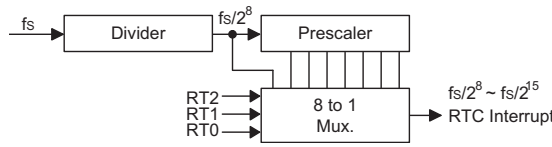
时基提供一个周期性溢出时间中断，它的溢出时间范围为  $f_s/2^{12} \sim f_s/2^{15}$ ，由掩膜选项决定。如果时基发生溢出现象，则其对应的中断请求标志(TBF)会被置位；如果中断允许，且此时堆栈尚有空间，则产生一个中断服务到 08H 的地址。进入 HALT 模式后，时基仍然工作(如果看门狗时钟来源是看门狗振荡器或 RTC 振荡器)，并且可以唤醒 HALT 模式。如果在进入 HALT 模式之前将“TBF”置“1”的话，则时基信号的溢出就不能唤醒系统。



时基

**实时时钟 — RTC**

实时时钟与时基的操作基本相同，可以产生有规律的内部中断。它的溢出时间范围为  $fs/2^8 \sim fs/2^{15}$ ，由软件设置决定。若将数据写入 RT2、RT1 和 RT0 (RTCC 第 2、第 1、第 0 位) 之中，会产生不同的溢出时间。如果实时时钟产生溢出的现象，则其对应的中断请求标志 (RTF; INTC 的第六位) 会被置位；如果中断允许，且此时堆栈尚有空间，则产生一个中断服务到 0CH 的地址。实时时钟的溢出时间信号，也可提供给定时/计数器做为时钟来源，以便得到更长的溢出时间周期。



实时时钟

RT2	RT1	RT0	RTC 分频级数
0	0	0	$2^8^*$
0	0	1	$2^9^*$
0	1	0	$2^{10}^*$
0	1	1	$2^{11}^*$
1	0	0	$2^{12}$
1	0	1	$2^{13}$
1	1	0	$2^{14}$
1	1	1	$2^{15}$

注：“\*” 不建议使用

**暂停模式 — HALT**

暂停模式是由 HALT 指令来实现的，具有下列功能：

- 关闭系统振荡器，但看门狗定时器振荡器仍会继续运行(如果选择的是看门狗定时器 RC 振荡器，或是实时时钟振荡器)
- 不改变 RAM 和寄存器的内容
- 清除并重新计数看门狗定时器(如果看门狗定时器的时钟来源为看门狗定时器 RC 振荡器，或是实时时钟振荡器)
- 所有输入/输出口都维持其原有状态
- 置位 PDF 标志并且清除 TO 标志
- 可由掩膜选择液晶显示驱动电路仍继续运作(如果选择的是看门狗定时器 RC 振荡器，或是实时时钟振荡器)

若欲离开暂停模式，可以执行外部复位、中断、PA 口下降沿的信号、或是看门狗定时器溢出。其中，外部复位会造成系统初始化(System Initialization)，看门狗定时器溢出则会发生热复位。我们可以通过检测 TO 和 PDF 标志的状态来了解系统复位的原因。欲清除 PDF 标志，可通过系统上电或是执行 CLR WDT 指令来达成。而若要置位 PDF 标志，则可执行 HALT 指令。如果发生看门狗定时器超时，不仅会置位 TO 标志，还会产生唤醒，并且复位程序计数器和状态指针，而其它的电路则继续维持其原有的状态。

PA 口唤醒和中断这两种唤醒方式可以视为正常运行的继续，PA 口上每个位在掩膜选择时都可以单独选择，用来唤醒系统。一旦从输入/输出口启动唤醒之后，程序即从下一条指令重新开始运行。但如果是从中断被唤醒的话，此时可能会发生两种情况。如果相关中断都被禁止，或该中断被允许且堆栈已用满的话，

程序会从下一条指令重新开始运行。但如果该中断被允许，但堆栈尚未用满，则会产生一般中断响应。

当进入“HALT”状态以前某个中断请求位被置位，那么系统不能用这个中断来唤醒。

若发生唤醒事件，必需额外花费  $1024t_{SYS}$ (系统时钟周期)的时间，才能重新正常运行。换言之，唤醒之后即会插入一个等待周期。如果唤醒是由于中断响应的话，实际中断子程序的执行会延时大约一个以上的周期。如果唤醒事件导致下一条指令执行，一旦插入的等待周期执行完成之后，会立即执行实际中断子程序。

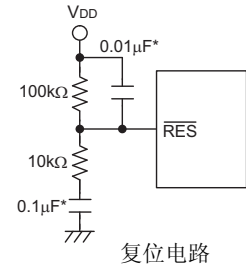
另外，为减少电源损耗，在进入暂停模式之前，应小心处理所有的输入/输出管脚状态。

**复位**

总共有四种方法会产生系统复位，如下所示：

- 正常操作时由  $\overline{RES}$  引脚发生复位
- 在暂停模式由  $\overline{RES}$  引脚发生复位
- 正常操作时由看门狗定时器溢出发生复位

暂停模式中的看门狗定时器溢出与其它系统复位状况不同，因为看门狗定时器溢出会执行热复位，用来重新设置程序计数器和堆栈指针，并保持其它电路原有的状态。少数寄存器在其它复位状态皆不会改变，大部分寄存器一旦符合复位条件时，会复位成初始的状态。通过检测 PDF 和 TO 这两个标志，程序即可区别出各种不同的系统复位。



复位电路  
注：“\*”连线应该尽量靠近  $\overline{RES}$  引脚，以减小干扰影响

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

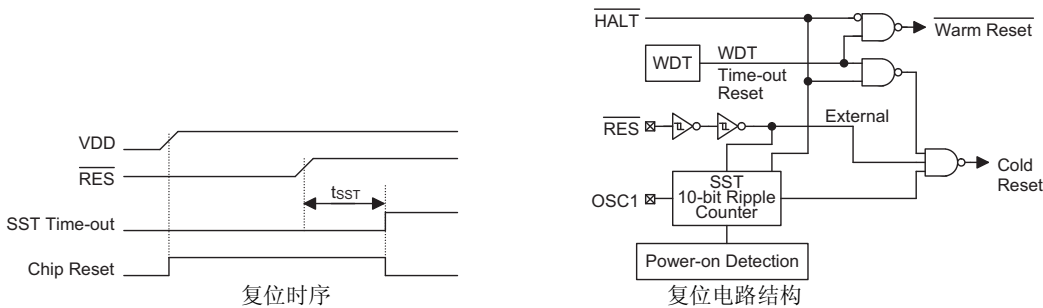
注：“u”表示不变

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

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

系统复位时各功能单元的状态如下所示：

程序计数器 PC	000H
中断	禁止
预分频器	清除
看门狗定时器、RTC、时基	清除、复位后定时器开始计数
定时/计数器	关闭
输入/输出口	输入模式
堆栈指针 SP	指向堆栈的顶端



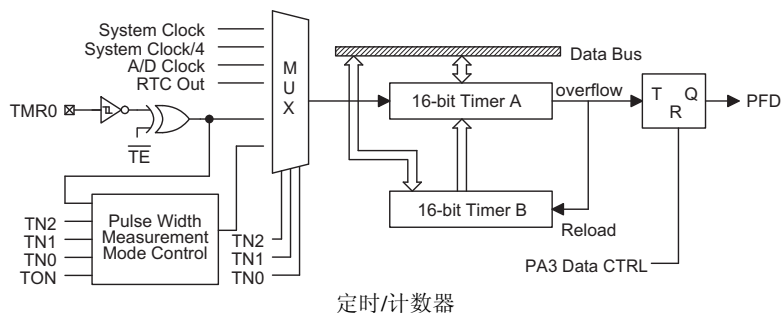
有关寄存器的状态如下

寄存器	复位(上电)	WDT 溢出 (正常运作)	RES复位 (正常运作)	RES复位 (暂停模式)	WDT 溢出 (暂停模式)
TMRAH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMRAL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMRC	0000 1---	0000 1---	0000 1---	0000 1---	uuuu u---
TMRBH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMRBL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ADCR	1xxx --00	1xxx --00	1xxx --00	1xxx --00	uuuu --uu
PC	000H	000H	000H	000H	000H*
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	---0 ---0	---0 ---0	---0 ---0	---0 ---0	---u ---u
RTCC	--00 0111	--00 0111	--00 0111	--00 0111	--uu uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu

注：“\*”表示“热复位” “u”表示“不变” “x”表示“未知”

### 定时/计数器

HT47R20A-1/HT47C20-1 有一个具有 PFD 输出功能的 16 位定时/计数器，可用作双通道的 RC 型 A/D 转换器。ADC/ $\overline{\text{TM}}$  位(ADCR 寄存器的第 1 位)用来决定定时器 A 和定时器 B 是用作 16 位的定时/计数器还是用作双通道的 RC 型 A/D 转换器。



定时/计数器

当 ADC/ $\overline{\text{TM}}$  为“0”时，TMRAL、TMRAH、TMRBL、TMRBH 组成了 16 位的定时/计数器。TMRBL 和 TMRBH 组成一个预置寄存器，分别用来存放定时/计数器初始值的低字节和高字节。

当定时/计数器采用内部时钟时，可以有三个时钟来源，分别是系统时钟、系统时钟/4 或 RTC 超时时钟；定时/计数器的时钟来源可以由外部输入。

外部时钟输入允许用户去计算外部事件，测量外部 RC 型的 A/D 时钟，测量时间长度或脉宽、或产生一个精确的时基信号。

总共有六个与定时/计数器工作模式有关的寄存器，分别是 TMRAH([20H])、TMRAL([21H])、TMRC([22H])、TMRBH([23H])、TMRBL([24H])和 ADCR([25H])。若写入 TMRBL 只能将数据写入低字节内部缓冲器中，但若写入的是 TMRBH 则可将数据和低字节内部缓冲器的内容同时写入定时/计数器预置寄存器(16 位)之中。改变定时/计数器预置寄存器的内容，只可被写入 TMRBH 之动作改变，但若写入 TMRBL 则可维持定时/计数器预置寄存器的内容不受改变。

同样的，若读取 TMRAH 则可将 TMRAL 传送至低字节内部缓冲器之中，以避免发生计时错误。然而，若读取 TMRAL，则只读回低字节内部缓冲器的内容。换言之，定时/计数器的低字节数据并不能直接读取。若欲读取该低字节的数据，必须先读取 TMRAH，以便将定时/计数器的低字节数据传送至内部低字节缓冲器之中。

如果定时/计数器正在计数，那么 TMRAH、TMRAL、TMRBH、TMRBL 不能读写。为了避免定时/计数器 A 和定时/计数器 B 之间产生冲突，应在定时/计数器关闭的情况下，使用指令“MOV”访问寄存器 TMRAH、TMRAL、TMRBH、TMRBL。

TMRC 为定时/计数器控制寄存器，用来定义定时/计数器的某些选项。定时/计数器的控制寄存器可以定义定时/计数器的工作模式、计数的允许或禁止以及计数的触发沿。

名称	位	功能
—	0~2	未定义，读取时为“0”
TE	3	定义定时/计数器 TMR 作用沿(0=上升沿作用, 1=下降沿作用)
TON	4	允许/禁止定时器计数(0=禁止, 1=允许)
TM0	5	定义操作方式(TM2, TM1, TM0) 000=定时器模式(系统时钟) 001=定时器模式(系统时钟/4) 010=定时器模式(RTC 输出)
TM1	6	011=A/D 时钟模式(由 ADCR 寄存器决定 RC 振荡)
TM2	7	100=计数器模式(外部时钟) 101=脉冲宽度测量模式(系统时钟/4) 110=未定义 111=未定义

TMRC (22H) 寄存器

写入定时器 B 就可以将定时/计数器的初始值放到预置寄存器中，而读取定时器 A 就可以得到定时/计数器的内容。定时器 B 是定时/计数器的初始值预置寄存器。

TM0、TM1 和 TM2 用来定义操作模式。计数器模式是用来计数外部事件，这表示时钟来源为外部 TMR 引脚的输入。A/D 时钟模式是用来计数外部 A/D 时钟，RC 振荡模式由寄存器 ADCR 来决定。定时器模式则作为普通定时器使用，其时钟来源为内部各种时钟。最后，脉冲宽度测量模式能够对外部引脚 TMR 的高电平或低电平的持续时间进行计数，计数的时钟来源为指令时钟。

在计数器、A/D 时钟或内部定时器模式下，一旦定时/计数器开始计数即从定时/计数器的现行内容(TMRAH 和 TMRAL)开始计数至 FFFFH。若发生溢出，计数器即从定时/计数器预置寄存器(TMRBH 和 TMRBL)重新装入加载值，并同时置位中断请求标志(TF; INTC1 的第四位)。

在脉冲宽度测量模式下，当 TON 和 TE 位的值都为 1 时，如果 TMR 收到由高电平到低电平(如果 TE 位的值为“0”，则为由低电平到高电平)的转变信号，计数器就会开始数，直到 TMR 引脚回到原来的电平为止，并且会将 TON 位清零。测量的结果会依然存放在定时/计数器之中，也就是说一次只能计数一次脉冲的宽度。而当 TON 位重新置位为“1”，只要 TMR 收到跳变脉冲，测量就会再次执行下去。在脉冲测量模式中，定时/计数器并不会根据逻辑电压来计数，其根据的标准为信号的转变沿。一旦发生计数溢出，计数器会从定时/计数器加载寄存器重新装入初值，同时还会发出中断请求，这种情况和定时和计数这两个模式一样。

若欲启动计数器运行，只要将定时器启动位(TON; TMRC 的第四位)的值设为“1”即可。在脉冲宽度测量模式中，TON 位在测量周期完成后，会自动被清除。但在其它三种模式中，TON 位只可以用指令清除。定时/计数器的溢出可作为唤醒的信号或由掩膜选项设定为 PA 的 PFD 输出。不管是何种操作模式，只要将“0”写入 ETI 位中即可将相对的中断服务禁止。当选择 PFD 功能时，执行“CLR PA.3”指令可以允许 PFD 输出；而执行“SET PA.3”指令则禁止 PFD 输出，并且 PA.3 输出为低电平。

若在定时/计数器关闭的情况下，将数据写入定时/计数器的预置寄存器同时也会将该数据重新载入定时/计数器之中。但若定时/计数器已经开启，写入定时/计数器的数据只会保存在定时/计数器的预置寄存器中。这时定时/计数器并不会马上被改变而会继续计数下去，直到发生溢出为止，此时再由加载寄存器装入新的初始值。

一旦定时/计数器(读取 TMRAH)的数据被读取，会将时钟禁止，以避免发生错误。将时钟禁止可能会导致计数错误，所以程序编写工程师必须考虑清楚才行。

我们强烈建议在打开定时/计数器前先将要加载的数据写入到 TMRBL、TMRBH、TMRAL 和 TMRAH 中去，因为在系统初始化后，TMRBL、TMRBH、TMRAL 和 TMRAH 的值是未知的。

如果定时/计数器是打开的，那么 TMRBL、TMRBH、TMRAL 和 TMRAH 最好不要进行读写操作。只有在定时/计数器关闭并且使用“MOV”指令时，才能对这四个寄存器进行读写操作。

下例为定时/计数器的定时模式(禁止中断):

```

clr tmrc
clr adcr.l           ; 设置为定时/计数器模式
clr intcl.4         ; 清除定时/计数器的中断请求标志位
mov a, low (65536-1000) ; 置定时器初值
mov tmrbl, a        ; 计数1000然后定时器溢出
mov a, high (65536-1000)
mov tmrbh, a

mov a, 00110000b    ; 定时器时钟来源为T1并且允许定时器计数
mov tmrc, a
    
```

P10:

```

clr wdt
snz intcl.4         ; 判断定时/计数器的中断请求标志位
jmp p10
clr intcl.4        ; 清除定时/计数器的中断请求标志位
                    ; 程序继续
    
```

### A/D 转换

HT47R20A-1/HT47C20-1 有两个 RC 型的 A/D 转换通道，包含两个可编程 16 位向上计数的计数器，计数器 A 的时钟来源可以是系统时钟、指令时钟或 RTC 输出时钟，计数器 B 的时钟来源可以是外部 RC 振荡电路。当 ADC/TM 位为“1”时(寄存器 ADRC 的第 1 位)，TMRAL、TMRAH、TMRBL、TMRBH 组成了 A/D 转换器。

标志	位	功能
OVB/ $\overline{\text{OVA}}$	0	在 RC 型 A/D 转换模式下，该位用来定义定时/计数器中断来自定时器 A 溢出或定时器 B 溢出(0=定时器 A 溢出，1=定时器 B 溢出) 在定时/计数器模式下，该位空缺
ADC/ $\overline{\text{TM}}$	1	设定定时/计数器或 RC 型 A/D 转换器允许(0=定时/计数器允许，1=A/D 转换器允许)
—	2~3	未定义，读取时为“0”
M0 M1 M2 M3	4 5 6 7	定义 A/D 转换器的工作模式(M3,M2,M1,M0) 0000=IN0 外部时钟输入模式 0001=RS0~CS0 振荡器(参考电阻和参考电容) 0010=RT0~CS0 振荡器(传感器电阻和参考电容) 0011=CRT0~CS0 振荡器(传感器电阻和参考电容) 0100=RS0~CRT0 振荡器(参考电阻和传感器电容) 0101=RS1~CS1 振荡器(参考电阻和参考电容) 0110=RT1~CS1 振荡器(传感器电阻和参考电容) 0111=IN1 外部时钟输入模式 1xxx=未定义

ADCR (25H) 寄存器

A/D 转换定时器 B 的时钟来源可以来自通道 0(IN0 外部时钟输入模式、RS0~CS0 振荡器、RT0~CS0 振荡器、CRT0~CS0 振荡器(CRT0 为一电阻)或 RS0~CRT0 振荡器(CRT0 为一电容))或来自通道 1(RS1~CS1 振荡器、RT1~CS1 振荡器或 IN1 外部时钟输入)。定时器 A 的时钟来源可以通过寄存器 TMRC 选择为系统时钟、指令时钟或是 RTC 预分频输出时钟。

总共有六个与 A/D 转换器有关的寄存器，分别是 TMRAH、TMRAL、TMRC、TMRBH、TMRBL 和

ADRC。内部定时器时钟输入到 TMRAH 和 TMRAL 中, A/D 时钟输入到 TMRBH 和 TMRBL 中。OV $\overline{B}$ /OV $\overline{A}$  位(ADCR 寄存器的第 0 位)用来设置定时器 A 或定时器 B 溢出作为定时/计数器中断信号。在 A/D 转换模式下, 当定时器 A 或定时器 B 溢出时 TON 位被清除并且计数器停止计数。写入 TMRAH/TMRBH 就是对定时器 A/定时器 B 设置初值, 读取 TMRAL/TMRBL 就是读取定时器 A/定时器 B 的内容, 写入 TMRAL/TMRBL 只能将数据写入内部缓冲器的低位字节, 但若写入的是 TMRAH/TMRBH 则可将数据和低字节内部缓冲器的内容写入定时器 A/定时器 B(16 位)之中。改变定时 A/定时器 B 的内容, 只可被写入 TMRAH/TMRBH 之动作改变, 但若写入 TMRAL/TMRBL 则可维持定时 A/定时器 B 的内容不改变。

若读取 TMRAL/TMRBL 则可将 TMRAL/TMRBL 传送至低字节内部缓冲器之中, 以避免发生计时错误。然而, 若读取 TMRAL/TMRBL, 则只读回低字节内部缓冲器的内容。换言之, 定时器 A/定时器 B 的低字节数据并不能直接读取。若欲读取该低字节的数据, 必须先读取 TMRAH/TMRBH, 以便将定时/计数器的低字节数据传送至内部低字节缓冲器之中。

如果定时/计数器是打开的, 那么 TMRBL、TMRBH、TMRAL 和 TMRAH 不能进行读写操作。为了避免定时/计数器 A 和定时/计数器 B 发生冲突, 应在定时/计数器关闭的情况下, 使用指令“MOV”访问寄存器 TMRAH、TMRAL、TMRBH、TMRBL。

寄存器 ADCR 的 4~7 位用来决定选取哪一组电阻、电容来组成 TMRBH 和 TMRBL 的振荡输入。

寄存器 TMRC 的 TM0、TM1、TM2 用来决定定时器 A 的时钟来源。定时器 A 的时钟来源可以是系统时钟、指令时钟或实时时钟(Real Time Clock)分频器时钟。

当 TON 位(TMRC 的第 4 位)置为“1”时, 定时器 A 和定时器 B 就开始计数, 直到定时器 A 或定时器 B 发生溢出, 此时, 定时/计数器便置位中断请求标志(TF; INTC1 的第 4 位), 同时计数器 A 和计数器 B 停止计数并 TON 位被清为“0”。

当 TON 位(TMRC 的第 4 位)置为“1”时, 那么 TMRBL、TMRBH、TMRAL 和 TMRAH 不能进行读写操作。只有在定时/计数器关闭并且使用“MOV”指令时, 才能对这四个寄存器进行读写操作。

下例是 RC 型 AD 转换模式(定时器 A 溢出):

```

clr tmrc
clr   adcr.1           ; 设置定时器模式
clr   intc1.4         ; 清除定时/计时器中断请求标志位
mov   a, low (65536-1000) ; 置TIMER A初值
mov   tmrbl, a        ; 计数1000后溢出
mov   a, high (65536-1000)
mov   tmrbh, a
mov   a, 00010010b    ; RS0~CS0; 设置RC型ADC模式; 设置TIMER A溢出
mov   adcr,a
mov   a, 00h          ; 置TIMER B初值
mov   tmrbl, a
mov   a, 00h
mov   tmrbh, a
mov   a, 00110000b    ; TIMER A的时钟来源为T1并且允许计数
mov   tmrc, a

p10:
clr   wdt
snz   intc1.4         ; 判断定时/计数器中断请求标志位
jmp   p10
clr   intc1.4         ; 清除定时/计数器中断请求标志位
; 程序继续

```

下例是 RC 型 AD 转换模式(定时器 B 溢出):

```

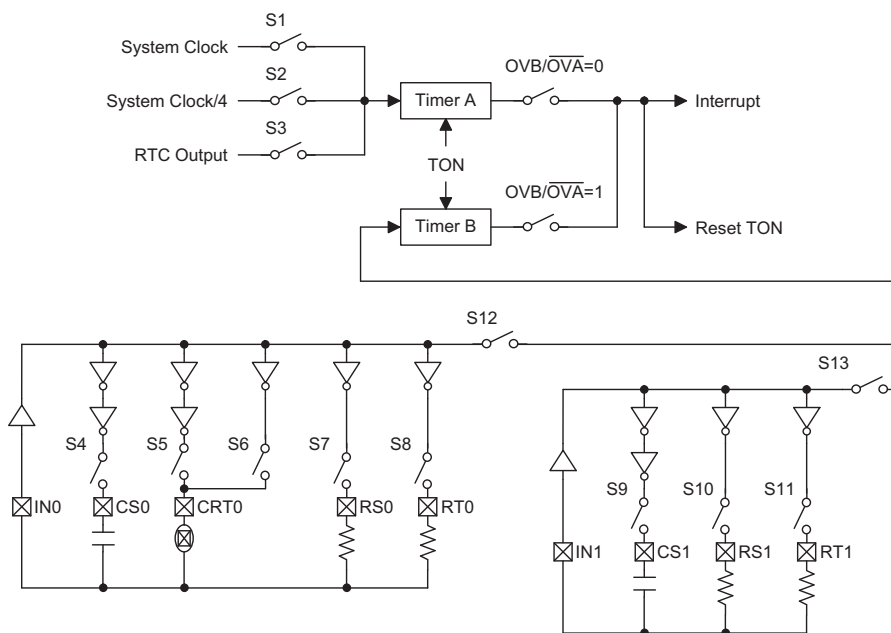
clr   tmrc
clr   adcr.1           ; 设置定时器模式
clr   intc1.4         ; 清除定时/计数器中断请求标志位
mov   a, 00h          ; 置TIMER A初值
mov   tmrbl, a
mov   a, 00h
mov   tmrbh, a

```

```

mov    a, 00010011b      ; RS0~CS0; 设置RC型ADC模式; 设置TIMERB溢出
mov    adcr, a
mov    a, low (65536-1000) ; 置TIMERB初值
mov    tmrbl, a          ; 计数1000后溢出
mov    a, high (65536-1000)
mov    tmrbh, a
mov    a, 00110000b      ; TIMERA的时钟来源为T1并且允许计数
mov    tmrc, a

p10:
clr    wdt
snz    intcl.4           ; 判断定时/计数器中断请求标志位
jmp    p10
clr    intcl.4          ; 清除定时/计数器中断请求标志位
; 程序继续
    
```



TN2	TN1	TN0	S1	S2	S3
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	1
Other			0	0	0

Note: 0=off, 1=on

M3	M2	M1	M0	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	0	0	1	0	0	0	0	1	0
0	0	1	0	1	0	0	0	1	0	0	0	1	0
0	0	1	1	1	0	1	0	0	0	0	0	1	0
0	1	0	0	0	1	0	1	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	1	1	0	0	1
0	1	1	0	0	0	0	0	0	1	0	1	0	1
0	1	1	1	0	0	0	0	0	0	0	0	0	1
1				0	0	0	0	0	0	0	0	0	0

Note: 0=off, 1=on

RC 型 A/D 转换器

输入/输出口

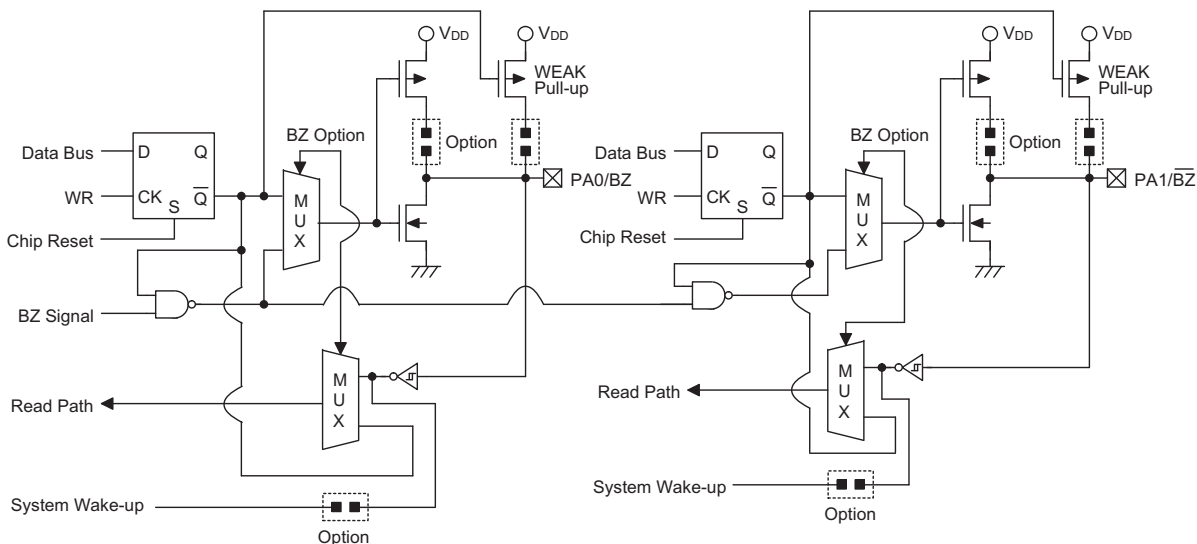
HT47R20A-1/HT47C20-1 有一个 8 位双向输入/输出口和一个 4 位的输入口，分别标号为 PA 和 PB，分别对应 RAM 中的[12H]和[14H]。PA 的高半字节引脚经常是带有上拉电阻的 NMOS 输入/输出，PA 的低半字节引脚可由掩膜选择设定为 NMOS 输入/输出或 CMOS 输出，PA 的每个引脚都可以具有唤醒功能，并且 PA 的低字节位可由掩膜选择设定为有或无上拉电阻。PB 只能用于输入，且每个引脚都带有上拉电阻。就 PA、PB 口作为输入而言，并不具有锁存功能，也就是说，所有输入在 MOV A, [m]指令(m=12H 或 m=14H)的 T2 上升沿时会被重新读入。就 PA 口作为输出而言，所有数据被锁存住，而且不受任何影响，直到输出

锁存被写入新的值为止。

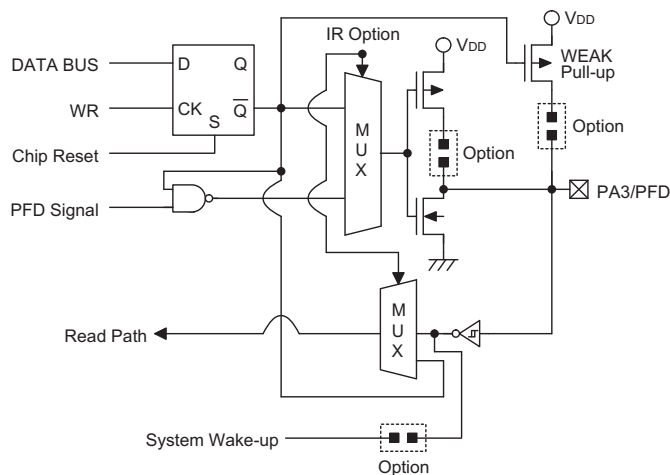
PA 口某个引脚在当成输入线使用时必须先设定相对应的为“1”，以便关闭 NMOS；也就是先执行“MOV A,0FFH”和“MOV [12H],A”指令关闭 NMOS，然后执行“MOV A,[12H]”指令来读取寄存器的数据。

在系统复位之后，这些引脚输入若不是高电平即为浮接(由掩膜选项决定)。

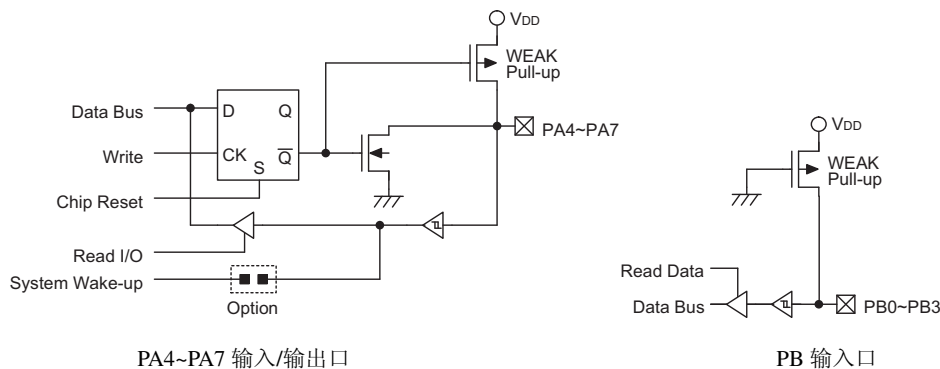
有些指令会先输入数据，然后才输出运行结果。举例来说：“SET [m].i”，“CLR [m].i”，“CPL [m]”和“CPLA[m]”这些指令会先将整个口状态读入 CPU 中，接着执行所定义的运算，最后再将执行的结果写入锁存或是累加器中。对 PA 口的输出锁存器不能使用这些指令，因为这些指令可能会引起输入和输出引脚的混乱(当输入引脚为低电平时)。



PA0/BZ、PA1/ $\overline{BZ}$  输入/输出口



PA2/IR、PA3/PFD 输入/输出口

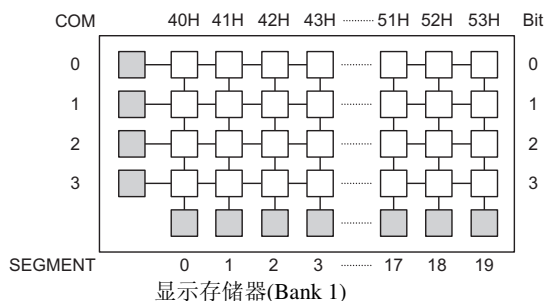


PA4~PA7 输入/输出口

PB 输入口

**LCD 液晶显示存储器**

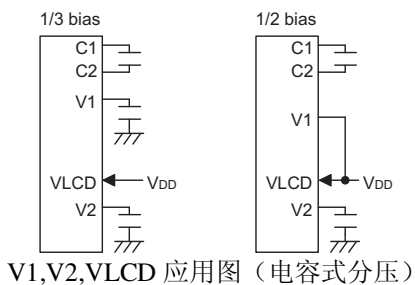
HT47R20A-1/HT47C20-1 为液晶显示提供一块嵌入式的数据存储区。LCD 的显示寄存器设计为 20×3bits。如果系统设置 LCD 为 19×4 的输出显示，那么 LCD 显示存储区域的 53H 无法读写。它的存储区域为 RAM 的 BANK1 内的 40H~53H，存储器段指针(BP; RAM 的 04H 单元)是在 RAM 和 LCD 显示存储器之间的切换开关。当 BP 被置“1”，任何数据写入 40H~53H 将会影响 LCD 的显示。当 BP 被清“0”，任何数据写入 40H~53H 意味着访问通用数据存储区。LCD 只能通过 MP1 进行间接寻址来访问。当数据被写入显示数据区域，这些数据自动地被 LCD 驱动器读取来产生相应的 LCD 驱动信号。把“1”或“0”写入显示存储器的相应位，来控制显示或不显示。图示表达了显示存储器和 HT47R20A-1/HT47C20-1 上 LCD 显示模式之间的对应显示关系。



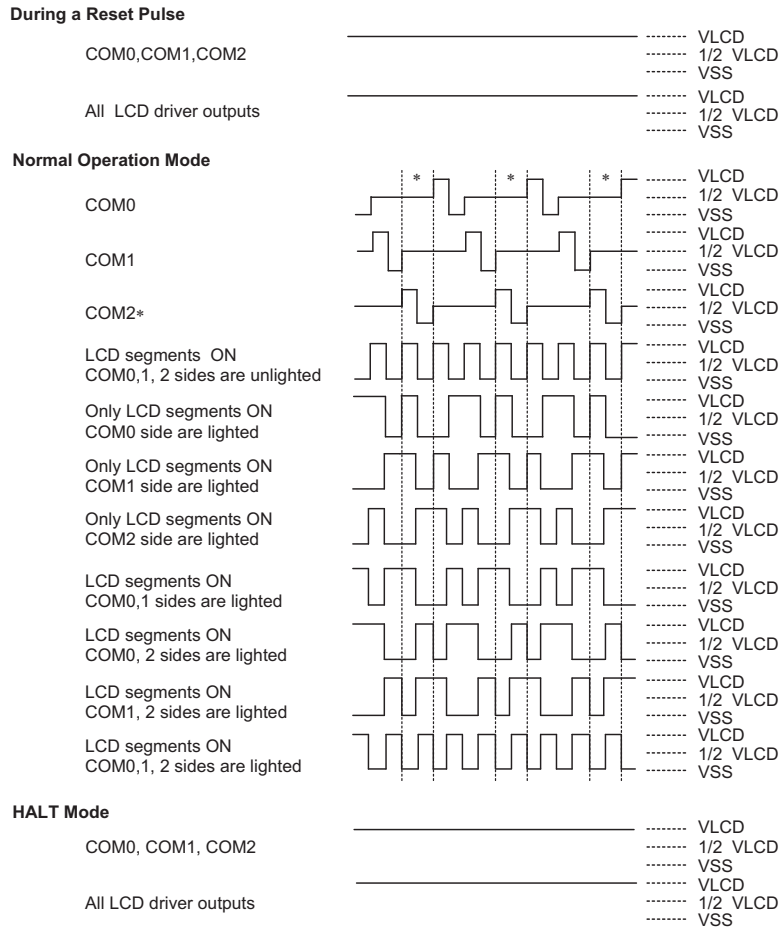
**液晶显示驱动输出**

HT47R20A-1/HT47C20-1 的 LCD 液晶显示驱动器数，可通过掩膜选择为 20×2、20×3 或 19×4(1/2、1/3、1/4 占空比)。

LCD 的驱动偏压方式可以为电容式偏压或电阻式偏压。如果选择了电容式分压，在 C1 和 C2 引脚之间要接上一个电容。可以通过掩膜选择设置为 1/2 偏压还是 1/3 偏压。如果选择了 1/2 偏压，则要在 V2 脚和地之间接上一个电容，如果是 1/3 偏压，则要在 V1 和 V2 脚上分别接上一个电容。参见参考电路。如果选择了电阻式偏压，则不需要外接电容。

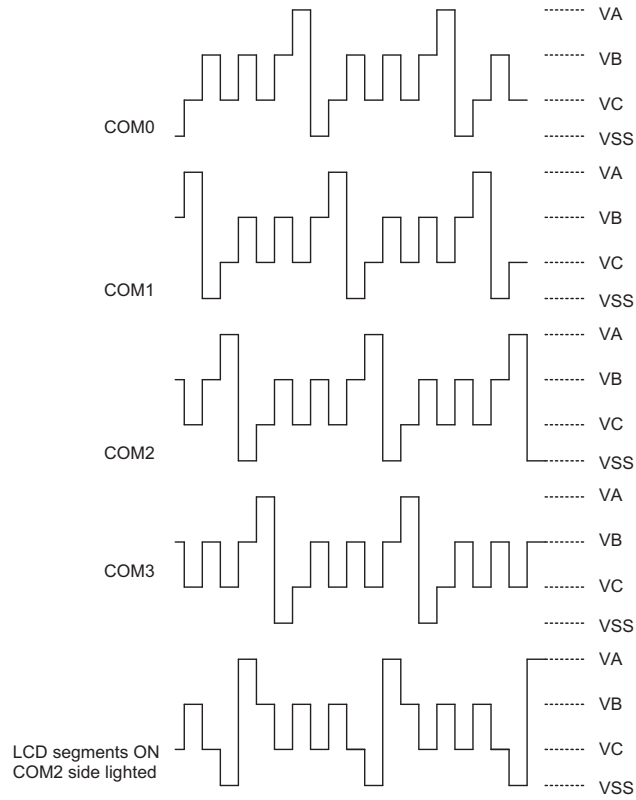


V1,V2,VLCD 应用图 (电容式分压)



Note: "\*" Omit the COM2 signal, if the 1/2 duty LCD is used.

LCD 驱动输出(1/3 duty、1/2bias、电阻/电容分压)



Note: 1/4 duty, 1/3 bias, C type: "VA" 3/2 VLCD, "VB" VLCD, "VC" 1/2 VLCD  
 1/4 duty, 1/3 bias, R type: "VA" VLCD, "VB" 2/3 VLCD, "VC" 1/3 VLCD

LCD 驱动输出

### 低电压复位/检测

HT47R20A-1/HT47C20-1 有低电压检测(LVD)和低电压复位(LVR)电路。可以通过掩膜设置来允许或禁止这两个功能。可以通过掩膜选择来设置 LVD 功能。一旦通过掩膜选择允许了 LVD 功能，则用户可以通过 RTCC.3 来允许/禁止(1/0)使用 LVD 电路，并且可以通过 RTCC.5 来读取 LVD 检测的状态；否则的话，LVD 功能是无效的。

当 LVR 执行系统复位功能时其功能和外部复位信号的功能是一样的。在 HALT 模式中，LVR 是无效的。

寄存器	位号	标号	读/写	复位	功能
RTCC (09H)	0~2	RT0~RT2	读/写	1	8选1的多路选择器控制输入以选择实时时钟的前置分频输出
	3	LVDC*	读/写	0	低电压检测使能位：“0”除能，“1”使能
	4	QOSC	读/写	0	32768振荡器是否快速起振：0/1 快/慢起振
	5	LVDO	读/写	0	低电压检测输出(1/0) 1：检测到低电压
	6~7	—	—	—	未使用，读数为“0”

RTCC (09H) 寄存器

注：“\*”位如果这项功能有效的那么参考电压也是有效的，否则参考电压由 LVR 掩膜选择决定。

### 蜂鸣器

HT47R20A-1/HT47C20-1 单片机提供了一对与 PA0、PA1 共用引脚的蜂鸣器输出，分别是 BZ 及  $\overline{BZ}$ 。该蜂鸣器的输出允许与输出频率，均可由掩膜选项决定。

如果蜂鸣器输出允许，用软件指令同时置 PA.0 和 PA.1 为“0”则启动蜂鸣器输出；如果置 PA.0=1 则关闭此蜂鸣器；如果置 PA.0=0 且 PA.1=1，则只有蜂鸣器 BZ 输出，而蜂鸣器  $\overline{BZ}$  被关闭。

PA1	PA0	功能
0 (CLR PA.1)	0 (CLR PA.0)	PA0=BZ PA1= $\overline{BZ}$
1 (SET PA.1)	0 (CLR PA.0)	PA0= $\overline{BZ}$ PA1=0
X	1 (SET PA.0)	PA0=0 PA1=0

蜂鸣器

### 红外载波发射

HT47R20A-1/HT47C20-1 具有红外载波发射功能，可以很方便的与红外线二极管相连，其输出与 PA2 共用一个引脚，其输出允许由掩膜选项决定。

如果红外载波输出允许，置 PA2 为“0”可以打开红外载波输出，置 PA2 为“1”则关闭红外载波输出并且 PA2 口输出为低电平。红外载波的输出频率为系统时钟的 12 分频，其占空比为 1/4。

PA2	功能
0 (CLR PA.2)	PA2=红外载波输出
1 (SET PA.2)	PA2=0

### 可编程分频器 — PFD

PFD 的输出与 PA3 共用一个引脚，由掩膜选项决定。

如果 PFD 输出允许，置 PA3 为“0” (CLR PA.3)可以打开 PFD 输出，置 PA3 为“1”则关闭 PFD 输出并且 PA3 口输出为低电平。

$$\text{PFD 输出频率} = \frac{1}{2} \times \frac{1}{\text{定时器溢出时间}}$$

PA3	功能
0 (CLR PA.3)	PA3=PFD 输出端
1 (SET PA.3)	PA3=0

### 掩膜选择

下表列出了 HT47R20A-1/HT47C20-1 的掩膜选择，这些选择必须设定清楚，以确保系统运作正常。

编号	掩膜选择
1	振荡器类型选择。 决定系统时钟为 RC 振荡器、晶体振荡器还是 RTC 振荡器。
2	看门狗定时器、实时时钟、时基的振荡源。 总共有三种不同的选择，分别为：WDT 振荡器、RTC 振荡器或系统时钟/4。
3	看门狗定时器允许/禁止选择。 看门狗定时器可设置为允许或禁止。
4	清除看门狗定时器次数选择，定义如何以指令清除看门狗定时器。“One time”表示 CLR WDT 指令能够清除看门狗定时器。“Two times”表示需要交互执行 CLR WDT1 和 CLR WDT2 这两个指令才可清除看门狗定时器。
5	时基溢出的周期选择。 溢出周期范围为 $f_s/2^{12} \sim f_s/2^{15}$ ， $f_s$ 为 WDT 的时钟来源。
6	蜂鸣器输出频率选择。 共有八种蜂鸣器输出频率，为 $f_s/2^2 \sim f_s/2^9$ ， $f_s$ 为 WDT 的时钟来源。
7	唤醒选择。定义唤醒功能的选择。 所有 PA 输入/输出管脚都可设置成下降沿将系统从暂停模式唤醒。
8	上拉电阻选择。 决定在 PA 口低半字节是否带有上拉电阻。
9	PA 口 CMOS 或 NMOS 架构选择。PA 口的低半字节可分别选择为 CMOS 或是 NMOS 架构。如果是 CMOS 架构，只能当作输出使用，如果是 NMOS 架构，则可以当作双向输入/输出使用。
10	输入/输出管脚与其它功能共用选择： PA0/BZ、PA1/BZ：PA0 和 PA1 可设置为一般的输入/输出管脚，或是蜂鸣器的输出口 PA2/IR：PA2 可设置为一般输入/输出管脚或红外线输出口 PA3/PFD：PA3 可设置为一般的输入/输出管脚或 PFD 的输出口
11	LCD common 选择，有三种形式可以选择：2common (1/2 周期)、3 common(1/3 周期)或 4common(1/4 周期)。如果选择 4 common，则 segment 输出端 SEG32 要设为 common 输出。
12	LCD 驱动时钟选择。 共有七种频率可供 LCD 选择： $f_s/2^2 \sim f_s/2^8$ ， $f_s$ 表示 WDT 的时钟源。
13	暂停模式下 LCD 的开关状态选择。 LCD 在暂停模式下的开关状态由掩膜选项设定
14	LVD 允许/禁止。
15	LVR 允许/禁止。



## 指令集

### 简介

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

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

### 指令周期

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

### 数据的传送

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

### 算术运算

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

### 逻辑和移位运算

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

### 分支和控制的转换

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设置尤其有用，其中个别的位或端口的引脚可以使用“SET [m].i”或“CLR [m].i”指令来设定其为高位或低位。如果没有这特性，程序设计师必须先读入输入输出的 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建立一个表格作为数据可直接存储的区域，只需要一组简易的指令即可对数据进行查表。

### 其它运算

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

### 指令集概要

#### 惯例

x: 立即数

m: 数据存储器地址

A: 累加器

i: 第 0~7 位

addr: 程序存储器地址

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

助记符	说明	指令周期	影响标志位
<b>移位</b>			
RRA [m]	数据存储器右移一位, 结果放入 ACC	1	无
RR [m]	数据存储器右移一位, 结果放入数据存储器	1 <sup>注</sup>	无
RRCA [m]	带进位将数据存储器右移一位, 结果放入 ACC	1	C
RRC [m]	带进位将数据存储器右移一位, 结果放入数据存储器	1 <sup>注</sup>	C
RLA [m]	数据存储器左移一位, 结果放入 ACC	1	无
RL [m]	数据存储器左移一位, 结果放入数据存储器	1 <sup>注</sup>	无
RLCA [m]	带进位将数据存储器左移一位, 结果放入 ACC	1	C
RLC [m]	带进位将数据存储器左移一位, 结果放入数据存储器	1 <sup>注</sup>	C
<b>数据传送</b>			
MOV A,[m]	将数据存储器送至 ACC	1	无
MOV [m],A	将 ACC 送至数据存储器	1 <sup>注</sup>	无
MOV A,x	将立即数送至 ACC	1	无
<b>位运算</b>			
CLR [m].i	清除数据存储器的位	1 <sup>注</sup>	无
SET [m].i	置位数据存储器的位	1 <sup>注</sup>	无

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

注: 1、对跳转指令而言, 如果比较的结果牵涉到跳转即需 2 个周期, 如果没有发生跳转, 则只需一个周期。

2、任何指令若要改变 PCL 的内容将需要 2 个周期来执行。

3、对于“CLR WDT1”或“CLR WDT2”指令而言, TO 和 PDF 标志位也许会受执行结果影响, “CLR WDT1”和“CLR WDT2”被连续地执行后, TO 和 PDF 标志位会被清除, 除此之外 TO 和 PDF 标志位保持不变。

## 指令定义

- ADC A, [m]** Add data memory and carry to the accumulator  
 说明: 将指定的数据存储器、累加器内容以及进位标志相加, 结果存放到累加器。  
 运算过程:  $ACC \leftarrow ACC + [m] + C$   
 影响标志位: OV、Z、AC、C
- ADCM A, [m]** Add the accumulator and carry to the accumulator  
 说明: 将指定的数据存储器、累加器内容和进位标志位相加, 结果存放到指定的数据存储器。  
 运算过程:  $[m] \leftarrow ACC + [m] + C$   
 影响标志位: OV、Z、AC、C
- ADD A, [m]** Add data memory to the accumulator  
 说明: 将指定的数据存储器 and 累加器内容相加, 结果存放到累加器。  
 运算过程:  $ACC \leftarrow ACC + [m]$   
 影响标志位: OV、Z、AC、C
- ADD A, x** Add immediate data to the accumulator  
 说明: 将累加器和立即数相加, 结果存放到累加器。  
 运算过程:  $ACC \leftarrow ACC + x$   
 影响标志位: OV、Z、AC、C
- ADDM A, [m]** Add the accumulator to the data memory  
 说明: 将指定的数据存储器 and 累加器内容相加, 结果存放到指定的数据存储器。  
 运算过程:  $[m] \leftarrow ACC + [m]$   
 影响标志位: OV、Z、AC、C
- AND A, [m]** Logical AND accumulator with data memory  
 说明: 将累加器中的数据和指定数据存储器内容做逻辑与, 结果存放到累加器。  
 运算过程:  $ACC \leftarrow ACC \text{ "AND" } [m]$   
 影响标志位: Z
- AND A, x** Logical AND immediate data to the accumulator  
 说明: 将累加器中的数据和立即数做逻辑与, 结果存放到累加器。  
 运算过程:  $ACC \leftarrow ACC \text{ "AND" } x$   
 影响标志位: Z
- ANDM A, [m]** Logical AND data memory with the accumulator  
 说明: 将指定数据存储器内容和累加器中的数据做逻辑与, 结果存放到数据存储器。  
 运算过程:  $[m] \leftarrow ACC \text{ "AND" } [m]$   
 影响标志位: Z
- CALL addr** Subroutine call  
 说明: 无条件的调用指定地址的子程序, 此时程序计数器先加 1 获得下一个要执行的指令地址并压入堆栈, 接着载入指定地址并从新地址执行程序。由于指令需要额外的运算, 所以此指令为 2 个周期。  
 运算过程:  $Stack \leftarrow Program\ Counter + 1$   
 $Program\ Counter \leftarrow addr$   
 影响标志位: 无

<b>CLR</b>	<b>[m]</b>	Clear data memory 说明: 将指定数据存储器的内容清零。 运算过程: $[m] \leftarrow 00H$ 影响标志位: 无
<b>CLR</b>	<b>[m].i</b>	Clear bit of data memory 说明: 将指定数据存储器的 i 位内容清零。 运算过程: $[m].i \leftarrow 0$ 影响标志位: 无
<b>CLR</b>	<b>WDT</b>	Clear Watchdog Timer 说明: WDT 计数器、暂停标志位 PDF 和看门狗溢出标志位 TO 清零。 运算过程: $WDT \leftarrow 00H$ $PDF \ \& \ TO \leftarrow 0$ 影响标志位: TO、PDF
<b>CLR</b>	<b>WDT1</b>	Preclear Watchdog Timer 说明: PDF 和 TO 标志位都被清 0。必须配合 CLR WDT2 一起使用清除 WDT 计时器。当程序仅执行 CLR WDT1, 而没有执行 CLR WDT2 时, PDF 与 TO 保留原状态不变。 运算过程: $WDT \leftarrow 00H$ $PDF \ \& \ TO \leftarrow 0$ 影响标志位: TO、PDF
<b>CLR</b>	<b>WDT2</b>	Preclear Watchdog Timer 说明: PDF 和 TO 标志位都被清 0。必须配合 CLR WDT1 一起使用清除 WDT 计时器。当程序仅执行 CLR WDT2, 而没有执行 CLR WDT1 时, PDF 与 TO 保留原状态不变。 运算过程: $WDT \leftarrow 00H$ $PDF \ \& \ TO \leftarrow 0$ 影响标志位: TO、PDF
<b>CPL</b>	<b>[m]</b>	Complement data memory 说明: 将指定数据存储器中的每一位取逻辑反, 相当于从 1 变 0 或从 0 变 1。 运算过程: $[m] \leftarrow \overline{[m]}$ 影响标志位: Z
<b>CPLA</b>	<b>[m]</b>	Complement data memory 说明: 将指定数据存储器中的每一位取逻辑反, 相当于从 1 变 0 或从 0 变 1, 结果被存放回累加器且数据寄存器的内容保持不变。 运算过程: $ACC \leftarrow \overline{[m]}$ 影响标志位: Z
<b>DAA</b>	<b>[m]</b>	Decimal-Adjust accumulator for addition 说明: 将累加器中的内容转换为 BCD (二进制转成十进制) 码。如果低四位的值大于“9”或 AC=1, 那么 BCD 调整就执行对原值加“6”, 否则原值保持不变; 如果高四位的值大于“9”或 C=1, 那么 BCD 调整就执行对原值加“6”。BCD 转换实质上是根据累加器和标志位执行 00H, 06H, 60H 或 66H 的加法运算, 结果存放于数据存储器。只有进位标志位 C 受影响, 用来指示原始 BCD 的和是否大于 100, 并可以进行双精度十进制数的加法运算。 操作: $[m] \leftarrow ACC+00H$ 或 $[m] \leftarrow ACC+06H$ $[m] \leftarrow ACC+60H$ 或 $[m] \leftarrow ACC+66H$ 影响标志位: C

<b>DEC</b>	<b>[m]</b>	Decrement data memory 将指定数据存储器的内容减 1。
说明:		
运算过程:		$[m] \leftarrow [m]-1$
影响标志位:		Z
<b>DECA</b>	<b>[m]</b>	Decrement data memory and place result in the accumulator 将指定数据存储器的内容减 1, 把结果存放回累加器并保持指定数据存储器的内容不变。
说明:		
运算过程:		$ACC \leftarrow [m]-1$
影响标志位:		Z
<b>HALT</b>		Enter power down mode
说明:		此指令终止程序执行并关掉系统时钟, RAM 和寄存器的内容保持原状态, WDT 计数器和分频器被清“0”, 暂停标志位 PDF 被置位 1, WDT 溢出标志位 TO 被清 0。
运算过程:		$PDF \leftarrow 1$ $TO \leftarrow 0$
影响标志位:		TO、PDF
<b>INC</b>	<b>[m]</b>	Increment data memory 将指定数据存储器的内容加 1。
说明:		
运算过程:		$[m] \leftarrow [m]+1$
影响标志位:		Z
<b>INCA</b>	<b>[m]</b>	Increment data memory and place result in the accumulator 将指定数据存储器的内容加 1, 结果存放回累加器并保持指定的数据存储器内容不变。
说明:		
运算过程:		$ACC \leftarrow [m]+1$
影响标志位:		Z
<b>JMP addr</b>		Directly jump
说明:		程序计数器的内容无条件地由被指定的地址取代, 程序由新的地址继续执行。当新的地址被加载时, 必须插入一个空指令周期, 所以此指令为 2 个周期的指令。
运算过程:		$PC \leftarrow addr$
影响标志位:		无
<b>MOV A, [m]</b>		Move data memory to the accumulator 将指定数据存储器的内容复制到累加器。
说明:		
运算过程:		$ACC \leftarrow [m]$
影响标志位:		无
<b>MOV A, x</b>		Move immediate data to the accumulator 将 8 位立即数载入累加器。
说明:		
运算过程:		$ACC \leftarrow x$
影响标志位:		无
<b>MOV [m], A</b>		Move the accumulator data to memory 将累加器的内容复制到指定的数据存储器。
说明:		
运算过程:		$[m] \leftarrow ACC$
影响标志位:		无
<b>NOP</b>		No operation
说明:		空操作, 顺序执行下一条指令。
运算过程:		$PC \leftarrow PC+1$
影响标志位:		无

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

<b>RLC</b>	<b>[m]</b>	<b>Rotate data memory left through carry</b>
说明:		将指定数据存储器的内容连同进位标志左移 1 位, 第 7 位取代进位标志且原本的进位标志移到第 0 位。
运算过程:		$[m].(i+1) \leftarrow [m].i \quad (i=0\sim 6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位:		C
<b>RLCA</b>	<b>[m]</b>	<b>Rotate left through carry and place result in the accumulator</b>
说明:		将指定数据存储器的内容连同进位标志左移 1 位, 第 7 位取代进位标志且原本的进位标志移到第 0 位, 移位结果送回累加器, 但是指定数据寄存器的内容保持不变。
运算过程:		$ACC.(i+1) \leftarrow [m].i \quad (i=0\sim 6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位:		C
<b>RR</b>	<b>[m]</b>	<b>Rotate data memory right</b>
说明:		将指定数据存储器的内容循环右移 1 位且第 0 位移到第 7 位。
运算过程:		$[m].i \leftarrow [m].(i+1) \quad (i=0\sim 6)$ $[m].7 \leftarrow [m].0,$
影响标志位:		无
<b>RRA</b>	<b>[m]</b>	<b>Rotate right and place result in the accumulator</b>
说明:		将指定数据存储器的内容循环右移 1 位, 第 0 位移到第 7 位, 移位结果存放到累加器, 而指定数据存储器的内容保持不变。
运算过程:		$ACC.i \leftarrow [m].(i+1) \quad (i=0\sim 6)$ $ACC.7 \leftarrow [m].0$
影响标志位:		无
<b>RRC</b>	<b>[m]</b>	<b>Rotate data memory right through carry</b>
说明:		将指定数据存储器的内容连同进位标志右移 1 位, 第 0 位取代进位标志且原本的进位标志移到第 7 位。
运算过程:		$[m].i \leftarrow [m].(i+1) \quad (i=0\sim 6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位:		C
<b>RRCA</b>	<b>[m]</b>	<b>Rotate right through carry and place result in the accumulator</b>
说明:		将指定数据存储器的内容连同进位标志右移 1 位, 第 0 位取代进位标志且原本的进位标志移到第 7 位, 移位结果送回累加器, 但是指定数据寄存器的内容保持不变。
运算过程:		$ACC.i \leftarrow [m].(i+1) \quad (i=0\sim 6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位:		C
<b>SBC</b>	<b>A,[m]</b>	<b>Subtract data memory and carry from the accumulator</b>
说明:		将累加器减去指定数据存储器的内容以及进位标志的反, 结果存放到累加器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:		$ACC \leftarrow ACC - [m] - \bar{C}$
影响标志位:		OV、Z、AC、C

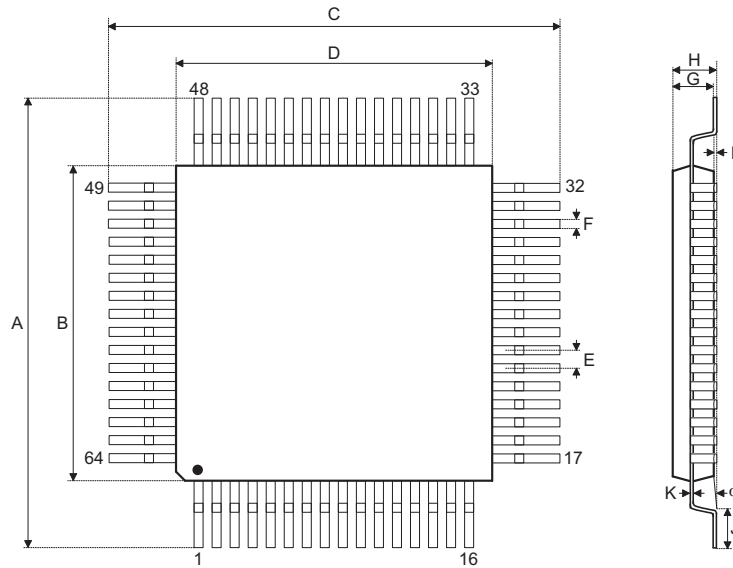
<b>SBCM</b>	<b>A,[m]</b>	<p><b>Subtract data memory and carry from the accumulator</b></p> <p>说明: 将累加器减去指定数据存储器的内容以及进位标志的反, 结果存放到数据存储器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。</p> <p>运算过程: <math>ACC \leftarrow ACC - [m] - \overline{C}</math></p> <p>影响标志位: OV、Z、AC、C</p>
<b>SDZ</b>	<b>[m]</b>	<p><b>Skip if decrement data memory is 0</b></p> <p>说明: 将指定的数据存储器的内容减 1, 判断是否为 0, 若为 0 则跳过下一条指令, 由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。</p> <p>运算过程: <math>[m] \leftarrow [m] - 1</math>, 如果 <math>[m]=0</math> 跳过下一条指令执行</p> <p>影响标志位: 无</p>
<b>SDZA</b>	<b>[m]</b>	<p><b>Decrement data memory and place result in ACC, skip if 0</b></p> <p>说明: 将指定数据存储器内容减 1, 判断是否为 0, 如果为 0 则跳过下一条指令, 此结果将存放到累加器, 但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。</p> <p>运算过程: <math>ACC \leftarrow [m] - 1</math>, 如果 <math>ACC=0</math> 跳过下一条指令执行</p> <p>影响标志位: 无</p>
<b>SET</b>	<b>[m]</b>	<p><b>Set data memory</b></p> <p>说明: 将指定数据存储器的每一位设置为 1。</p> <p>运算过程: <math>[m] \leftarrow FFH</math></p> <p>影响标志位: 无</p>
<b>SET</b>	<b>[m].i</b>	<p><b>Set bit of data memory</b></p> <p>说明: 将指定数据存储器的第 i 位设置为 1。</p> <p>运算过程: <math>[m].i \leftarrow 1</math></p> <p>影响标志位: 无</p>
<b>SIZ</b>	<b>[m]</b>	<p><b>Skip if increment data memory is 0</b></p> <p>说明: 将指定的数据存储器的内容加 1, 判断是否为 0, 若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。</p> <p>运算过程: <math>[m] \leftarrow [m] + 1</math>, 如果 <math>[m]=0</math> 跳过下一条指令执行</p> <p>影响标志位: 无</p>
<b>SIZA</b>	<b>[m]</b>	<p><b>Increment data memory and place result in ACC, skip if 0</b></p> <p>说明: 将指定数据存储器的内容加 1, 判断是否为 0, 如果为 0 则跳过下一条指令, 此结果会被存放到累加器, 但是指定数据存储器的内容不变。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。</p> <p>运算过程: <math>ACC \leftarrow [m] + 1</math>, 如果 <math>ACC=0</math> 跳过下一条指令执行</p> <p>影响标志位: 无</p>
<b>SNZ</b>	<b>[m].i</b>	<p><b>Skip if bit I of the data memory is not 0</b></p> <p>说明: 判断指定数据存储器的第 i 位, 若不为 0, 则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果为 0, 则程序继续执行下一条指令。</p> <p>运算过程: 如果 <math>[m].i \neq 0</math>, 跳过下一条指令执行</p> <p>影响标志位: 无</p>

<b>SUB</b>	<b>A, [m]</b>	<b>Subtract data memory from the accumulator</b>
说明:		将累加器的内容减去指定的数据存储器的数据, 把结果存放到累加器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:		$ACC \leftarrow ACC - [m]$
影响标志位:		OV、Z、AC、C
<b>SUBM</b>	<b>A, [m]</b>	<b>Subtract data memory from the accumulator</b>
说明:		将累加器的内容减去指定数据存储器的数据, 结果存放到指定的数据存储器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:		$[m] \leftarrow ACC - [m]$
影响标志位:		OV、Z、AC、C
<b>SUB</b>	<b>A, x</b>	<b>Subtract immediate data from the accumulator</b>
说明:		将累加器的内容减去立即数, 结果存放到累加器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。
运算过程:		$ACC \leftarrow ACC - x$
影响标志位:		OV、Z、AC、C
<b>SWAP</b>	<b>[m]</b>	<b>Swap nibbles within the data memory</b>
说明:		将指定数据存储器的低 4 位和高 4 位互相交换。
运算过程:		$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
影响标志位:		无
<b>SWAPA</b>	<b>[m]</b>	<b>Swap data memory and place result in the accumulator</b>
说明:		将指定数据存储器的低 4 位和高 4 位互相交换, 再将结果存放到累加器且指定数据寄存器的数据保持不变。
运算过程:		$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
影响标志位:		无
<b>SZ</b>	<b>[m]</b>	<b>Skip if data memory is 0</b>
说明:		判断指定数据存储器的内容是否为 0, 若为 0, 则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:		如果 $[m] = 0$ , 跳过下一条指令执行
影响标志位:		无
<b>SZA</b>	<b>[m]</b>	<b>Move data memory to ACC, skip if 0</b>
说明:		将指定数据存储器内容复制到累加器, 并判断指定数据存储器的内容是否为 0, 若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:		$ACC \leftarrow [m]$ , 如果 $[m] = 0$ , 跳过下一条指令执行
影响标志位:		无
<b>SZ</b>	<b>[m]. i</b>	<b>Skip if bit I of the data memory is 0</b>
说明:		判断指定数据存储器的第 i 位是否为 0, 若为 0, 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。
运算过程:		如果 $[m].i = 0$ , 跳过下一条指令执行
影响标志位:		无

<b>TABRDC [m]</b>	Move the ROM code(current page) to TBLH and data memory
说明:	将表格指针 TBLP 所指的程序代码低字节（当前页）移至指定的数据存储器且将高字节移至 TBLH。
运算过程:	[m] ←程序代码（低字节） TBLH←程序代码（高字节）
影响标志位:	无
<b>TABRDL [m]</b>	Move the ROM code(last page) to TBLH and data memory
说明:	将表格指针 TBLP 所指的程序代码低字节（最后一页）移至指定的数据存储器且将高字节移至 TBLH。
运算过程:	[m] ←程序代码（低字节） TBLH←程序代码（高字节）
影响标志位:	无
<b>XOR A, [m]</b>	Logical XOR accumulator with data memory
说明:	将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。
运算过程:	ACC←ACC “XOR” [m]
影响标志位:	Z
<b>XORM A, [m]</b>	Logical XOR data memory with accumulator
说明:	将累加器的数据和指定的数据存储器内容逻辑异或，结果放到数据存储器。
运算过程:	[m]←ACC “XOR” [m]
影响标志位:	Z
<b>XOR A, x</b>	Logical XOR immediate data to the accumulator
说明:	将累加器的数据与立即数逻辑异或，结果存放到累加器。
运算过程:	ACC←ACC “XOR” x
影响标志位:	Z

封装信息

64-pin LQFP (7mm×7mm)外形尺寸



符号	尺寸 (单位: mm)		
	最小	典型	最大
A	8.9	—	9.1
B	6.9	—	7.1
C	8.9	—	9.1
D	6.9	—	7.1
E	—	0.4	—
F	0.13	—	0.23
G	1.35	—	1.45
H	—	—	1.6
I	0.05	—	0.15
J	0.45	—	0.75
K	0.09	—	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2016号1号楼3楼G室 201103  
电话: 021-5422-4590  
传真: 021-5422-4596  
网站: www.holtek.com.cn

**盛扬半导体有限公司（深圳业务处）**

深圳市南山区科技园科技中三路与高新中二道交汇处生产力大楼A单元五楼 518057  
电话: 0755-8616-9908, 8616-9308  
传真: 0755-8616-9722

**盛扬半导体有限公司（北京业务处）**

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

**盛扬半导体有限公司（成都业务处）**

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

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

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

Copyright © 2008 by HOLTEK SEMICONDUCTOR INC.

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