

## 技术相关信息

- [工具信息](#)
- [FAQs](#)
- [应用范例](#)
  - [HA0017S HT49 MCU 系列微控制器读写 HT24 系列应用范例](#)
  - [HA0024S HT49 MCU RTC 实时时钟的使用](#)
  - [HA0025S HT49 MCU 中的 Time base\(时基\)的使用说明](#)
  - [HA0026S HT49 MCU 输入/输出口的使用](#)
  - [HA0027S HT49 MCU 定时/计数器的使用](#)
  - [HA0075S MCU 复位电路和振荡电路的应用范例](#)

## 特性

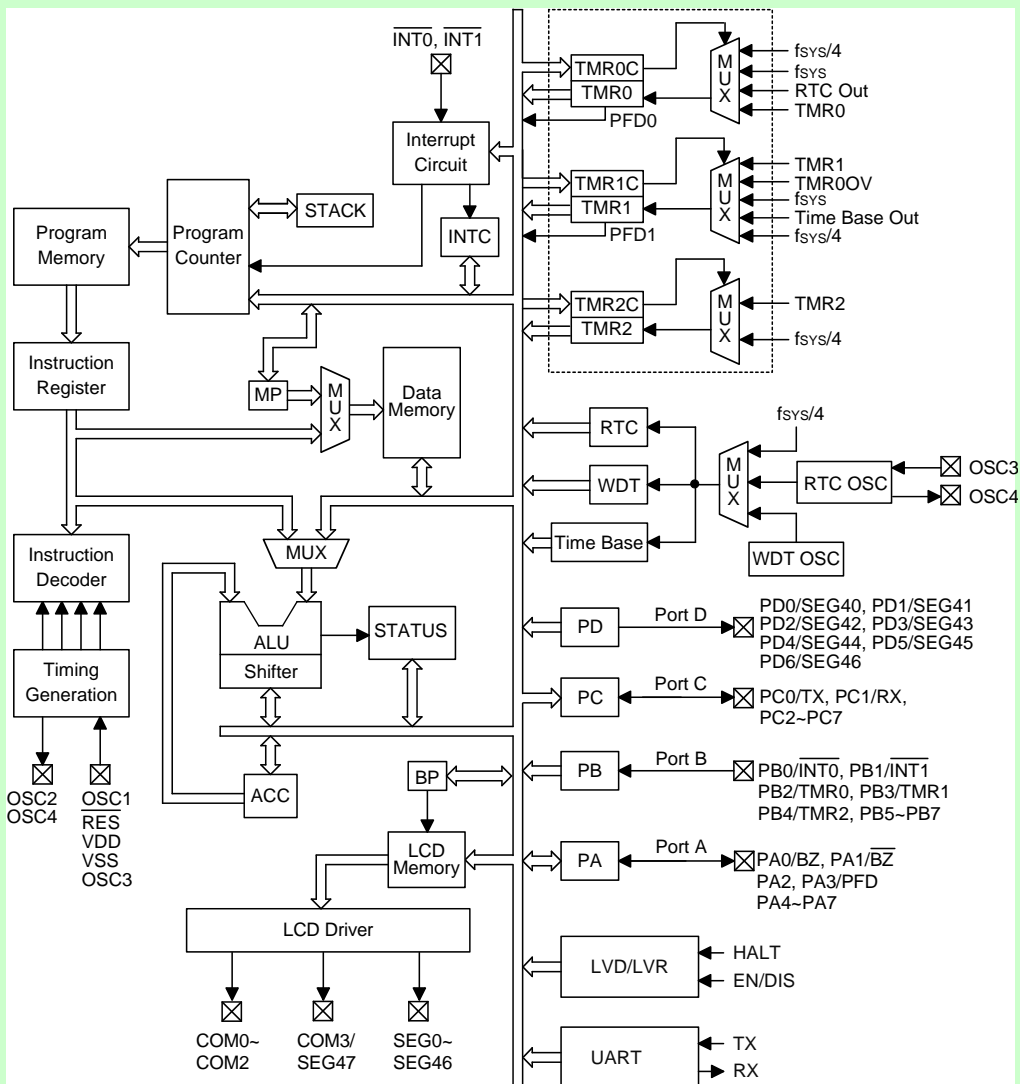
- 工作电压：  
fsys = 4MHz      2.2V~5.5V  
fsys = 8MHz      3.3V~5.5V
- 8 位输入口和 7 位输出口
- 16 个双向输入/输出口
- 2 个外部中断输入
- 具有预分频器及中断功能的一个 8 位和两个 16 位的可编程定时/计数器
- 48×2、48×3、47×4 段的 LCD 驱动器
- 16K×16 位的程序存储器 ROM
- 576×8 位的数据存储器 RAM
- 实时时钟 (RTC)
- 实时时钟 (RTC) 的 8 位预分频器
- 看门狗定时器
- 蜂鸣器输出
- 内置晶体、RC、32768Hz 晶体的振荡电路
- 提供暂停和唤醒功能，以降低功耗
- 16 层硬件堆栈
- 通用异步接收/发送器(UART)
- 位操作指令
- 16 位查表指令
- 系统时钟为 8MHz 时，指令周期为 0.5 μs
- 63 条指令
- 所有指令都可在 1 或 2 个指令周期内完成
- 具有低电压复位/检测功能
- 64-pin LQFP 和 100-pin LQFP 封装形式

## 概述

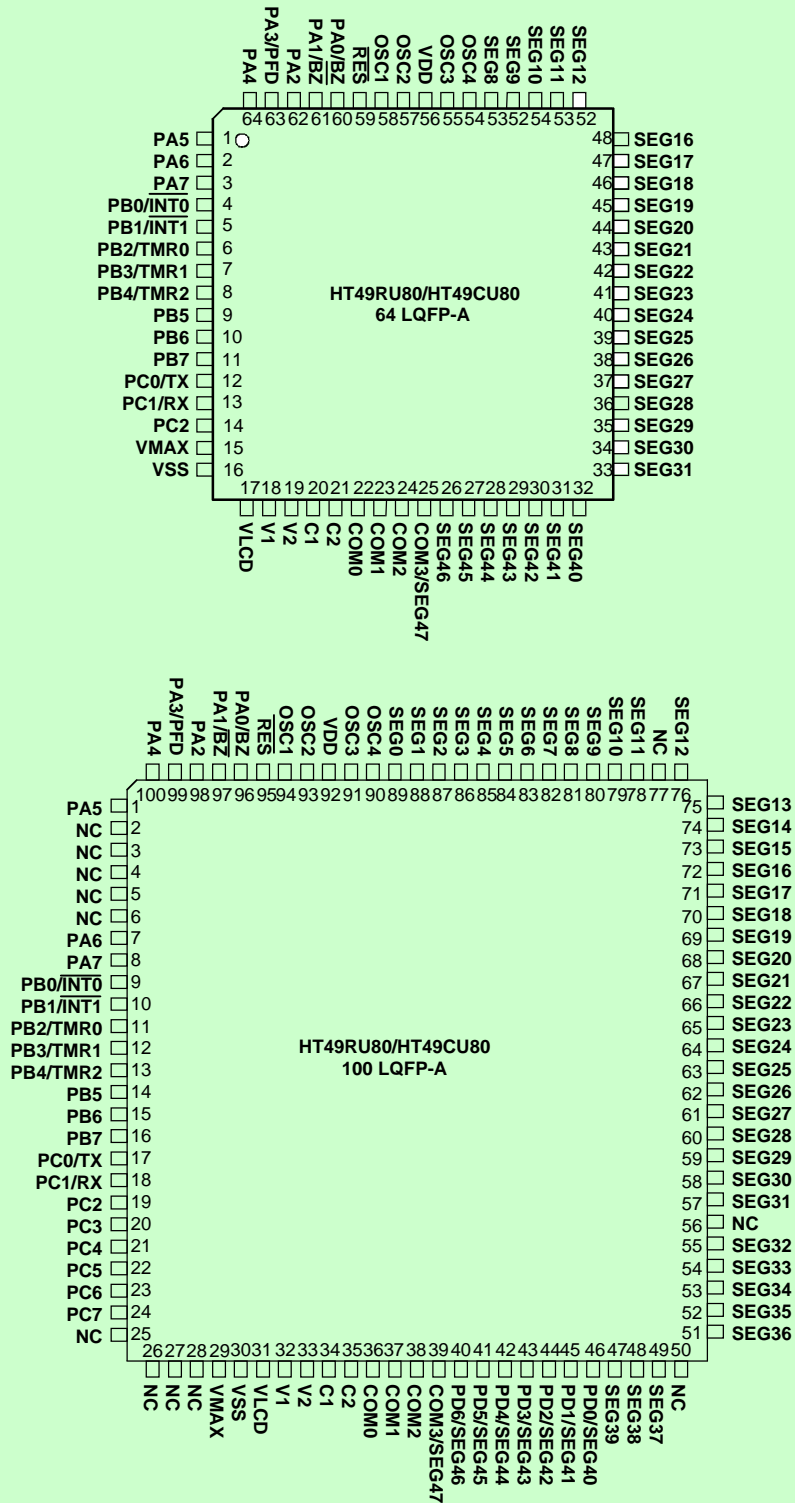
HT49RU80 是 8 位高性能精简指令集单片机，专门为需要 LCD 显示功能的产品而设计。MASK 版本 HT49CU80 与 OTP 版本 HT49RU80 引脚和功能完全相同。

具有低功耗、I/O 使用灵活、可编程分频器、计数器、振荡类型选择、暂停和唤醒、另有灵活的蜂鸣器输出以及 LCD 显示功能，使这款单片机功能更多样性。可以广泛应用于电子秤、电表、气表、定时器、计算器、遥控器和其他带 LCD 显示功能的工业或家用消费类产品等系统中。

方框图



引脚图



## 引脚说明

引脚名称	输入/输出	配置选项	说 明
PA0/BZ PA1/BZ PA2 PA3/PFD PA4~PA7	输入/输出	唤醒功能 CMOS 或 NMOS 上拉电阻 PA0/PA1 或 BZ/BZ PA3 或 PFD	PA0~PA7 是带斯密特触发输入的 8 位双向输入/输出端口。每一位能被配置选项设置为唤醒输入。PA0~PA3 为 CMOS 输出或 NMOS 输入/输出（由配置选项确定有无上拉电阻）。PA4~PA7 是带上拉电阻的 NMOS 输入/输出。PA0、PA1 和 PA3 分别与 BZ、BZ 和 PFD 共用引脚。
PB0/INT0 PB1/INT1 PB2/TMR0 PB3/TMR1 PB4/TMR2 PB5~PB7	输入	—	PB0~PB7 是 8 位带上拉电阻的斯密特触发输入口。PB0 和 PB1 分别与INT0和INT1共用引脚。而 PB2、PB3 和 PB4 分别与 TMR0、TMR1 和 TMR2 共用引脚。
PC0/TX PC1/RX PC2~PC7	输入/输出	上拉电阻 CMOS 或 NMOS	PC0~PC7 是双向 8 位的有斯密特触发输入的输入/输出口。PC0~PC3 和 PC4~PC7 可分别由配置选项配置为 CMOS 输出或 NMOS 输入/输出,若 PC0~PC3 或 PC4~PC7 选择为 NMOS 输入/输出,可分别由配置选项配置为带不带上拉电阻。PC0 和 PC1 分别与 UART 的 TX 和 RX 共用引脚。
PD0/SEG40~ PD6/SEG46	输出	CMOS 输出 或 SEG 输出	7 位输出端口。每个端口可由软件配置成 CMOS 输出或 SEG 输出。
VLCD	输入	—	LCD 电源输入。此电源只供给 LCD 使用。V <sub>LCD</sub> 电压可高于或低于 VDD 电压。
VMAX	—	—	IC 最高电压,接 VDD、VLCD 或 V1。
V1,V2,C1,C2	输入	—	电压泵。
COM0~COM2 COM3/ SEG47	输出	1/2,1/3 或 1/4 占空比	SEG47 可由配置选项被确定为 LCD 的段(segment)或公共端的输出驱动器。COM2~COM0 是作为 LCD 面板公共端的输出极。
SEG0~SEG39	输出	—	LCD 面板段驱动器输出。
OSC1 OSC2	输入 输出	晶体 或 RC	OSC1 和 OSC2 连接一个 RC 或一个晶体（由配置选项确定）来产生内部系统时钟。在 RC 方式下,OSC2 是一个系统时钟四分频的输出端。系统时钟也可来源于 RTC 振荡器。如果系统时钟来源于 RTC 振荡器,OSC1 和 OSC2 引脚悬空。
OSC3 OSC4	输入 输出	RTC 或系统时钟	实时时钟振荡器。OSC3 和 OSC4 连接到一个 32768Hz 的晶体振荡器来产生定时时间或连接到系统时钟源（由掩膜选择决定）。
RES	输入	—	斯密特触发复位输入,低电平有效。
VDD	—	—	正电源
VSS	—	—	负电源,接地

## 极限参数

电源供应电压.....V <sub>SS</sub> -0.3V~V <sub>SS</sub> +6.0V	储存温度.....-50℃~125℃
端口输入电压.....V <sub>SS</sub> -0.3V~V <sub>DD</sub> +0.3V	工作温度.....-40℃~85℃
I <sub>OL</sub> 总电流.....150mA	I <sub>OH</sub> 总电流..... -100mA
总消耗电流.....500mW	

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				
V <sub>DD</sub>	工作电压	—	LVR 关闭, f <sub>sys</sub> = 4MHz	2.2	—	5.5	V
			LVR 关闭, f <sub>sys</sub> = 8MHz	3.3	—	5.5	V
V <sub>LCD</sub>	LCD 供电电源 (注*)	—	VA≤5.5V	2.2	—	5.5	V
I <sub>DD1</sub>	工作电流 (晶体振荡, RC 振荡)	3V	无负载, f <sub>sys</sub> =4MHz	—	1	2	mA
		5V	UART 关闭	—	3	5	mA
I <sub>DD2</sub>	工作电流 (晶体振荡, RC 振荡)	3V	无负载, f <sub>sys</sub> =4MHz	—	1.5	3.0	mA
		5V	UART 打开	—	3	6	mA
I <sub>DD3</sub>	工作电流 (晶体振荡, RC 振荡)	5V	无负载, f <sub>sys</sub> =8MHz UART 关闭	—	4	8	mA
I <sub>DD4</sub>	工作电流 (晶体振荡, RC 振荡)	5V	无负载, f <sub>sys</sub> =8MHz UART 打开	—	5	10	mA
I <sub>DD5</sub>	工作电流 (f <sub>sys</sub> = RTC 振荡)	3V	无负载, UART 关闭	—	0.3	0.6	mA
		5V		—	0.6	1	mA
I <sub>STB1</sub>	静态电流 (*f <sub>s</sub> = f <sub>sys</sub> /4)	3V	无负载, 系统 HALT HALT 时关闭 LCD, UART 关闭	—	—	1	μA
		5V	—	—	2	μA	
I <sub>STB2</sub>	静态电流 (*f <sub>s</sub> = RTC 振荡)	3V	无负载, 系统 HALT HALT 时开启 LCD, 电容型	—	2.5	5	μA
		5V	UART 关闭	—	10	20	μA
I <sub>STB3</sub>	静态电流 (*f <sub>s</sub> = WDT 振荡)	3V	无负载, 系统 HALT HALT 时开启 LCD, 电容型	—	2	5	μA
		5V	UART 关闭	—	6	10	μA
I <sub>STB4</sub>	静态电流 (*f <sub>s</sub> = RTC 振荡)	3V	无负载, 系统 HALT HALT 时开启 LCD, 电阻型	—	17	30	μA
		5V	1/2 偏压, UART 关闭	—	34	60	μA
I <sub>STB5</sub>	静态电流 (*f <sub>s</sub> = RTC 振荡)	3V	无负载, 系统 HALT HALT 时开启 LCD, 电阻型	—	13	25	μA
		5V	1/3 偏压, UART 关闭	—	26	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 时开启 LCD, 电阻型	—	14	25	μA
		5V	1/2 偏压, UART 关闭	—	28	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 时开启 LCD, 电阻型	—	10	20	μA
		5V	1/3 偏压, UART 关闭	—	20	40	μA
V <sub>IL1</sub>	I/O 口、TMR0、TMR1、 TMR2、 $\overline{\text{INT}}0$ 和 $\overline{\text{INT}}1$ 输入低电压	—	—	0	—	0.3V <sub>DD</sub>	V
V <sub>IH1</sub>	I/O 口、TMR0、TMR1、 TMR2、 $\overline{\text{INT}}0$ 和 $\overline{\text{INT}}1$ 输入高电压	—	—	0.7 V <sub>DD</sub>	—	V <sub>DD</sub>	V
V <sub>IL2</sub>	输入低电压( $\overline{\text{RES}}$ )	—	—	0	—	0.4 V <sub>DD</sub>	V
V <sub>IH2</sub>	输入高电压( $\overline{\text{RES}}$ )	—	—	0.9 V <sub>DD</sub>	—	V <sub>DD</sub>	V
I <sub>OL1</sub>	IO 口灌电流	3V	V <sub>OL</sub> =0.1 V <sub>DD</sub>	6	12	—	mA

		5V		10	25	—	mA
I <sub>OH1</sub>	IO 口源电流	3V	V <sub>OH</sub> =0.9 V <sub>DD</sub>	-2	-4	—	mA
		5V		-5	-8	—	mA
I <sub>OL2</sub>	LCD COM 和 SEG 口电流	3V	V <sub>OL</sub> =0.1 VA	210	420	—	μA
		5V		350	700	—	μA
I <sub>OH2</sub>	LCD COM 和 SEG 口电流	3V	V <sub>OH</sub> =0.9VA	-80	-160	—	μA
		5V		-180	-360	—	μA
R <sub>PH</sub>	上拉电阻	3V	—	20	60	100	kΩ
		5V		10	30	50	kΩ
V <sub>LVR</sub>	低电压复位电压	—	—	2.7	3.0	3.3	V
V <sub>LVD</sub>	低电压检测电压	—	—	3.0	3.3	3.6	V

注意：“\*” VA 电压请参考 LCD 驱动章节。

“\*fs”请参考 WDT 时钟选项。

## 交流特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		VDD	条件				
f <sub>SYS1</sub>	系统时钟 (晶体、RC 振荡)	—	2.2V~5.5V	400	—	4000	kHZ
		—	3.3V~5.5V	400	—	8000	kHZ
f <sub>SYS2</sub>	系统时钟 (32768Hz 晶体振荡)	—	—	—	32768	—	Hz
f <sub>RTCOSC</sub>	实时时钟频率	—	—	—	32768	—	Hz
f <sub>TIMER</sub>	定时器输入频率 (50% 占空比)	—	2.2V~5.5V	0	—	4000	kHZ
		—	3.3V~5.5V	0	—	8000	kHZ
t <sub>WDTOSC</sub>	看门狗振荡器周期	3V	—	45	90	180	μs
		5V		32	65	130	μs
t <sub>RES</sub>	外部复位低电平脉冲宽度	—	—	1	—	—	μs
t <sub>SST</sub>	系统启动延迟周期	—	从 HALT 状态唤醒	—	1024	—	*t <sub>SYS</sub>
t <sub>LVR</sub>	低电压复位宽度	—	—	0.25	1	2	ms
t <sub>INT</sub>	中断脉冲宽度	—	—	1	—	—	μs

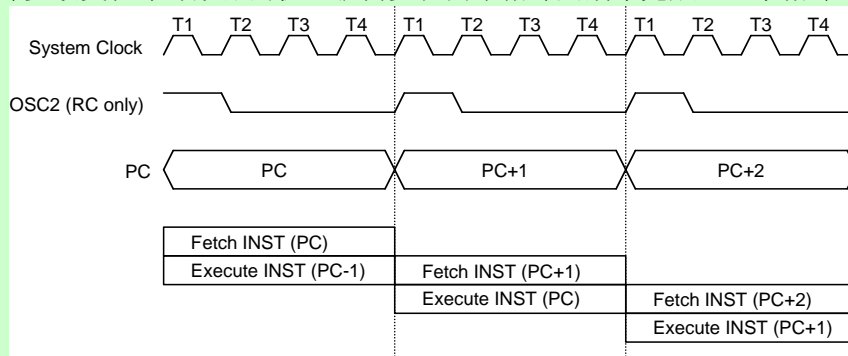
注意：\*t<sub>SYS</sub>= 1/f<sub>SYS1</sub> 或 1/f<sub>SYS2</sub>

## 功能说明

### 指令执行时序

系统频率由晶体振荡、RC 振荡或 32768Hz 晶体振荡产生。系统内部对此频率进行四分频，产生四个不重叠的时钟周期作为系统时钟，一个指令周期包含四个系统时钟周期。

指令读取与执行是以流水线方式来进行的。这种方式允许指令在前一个指令周期进行读取操作，而在当前指令周期里进行解码与执行。这种流水线方式能在一个指令周期里有效地执行每条指令。如果涉及到的指令要改变程序计数器的值，就需要花两个指令周期来完成这一条指令。



指令执行时序

### 程序计数器 — PC

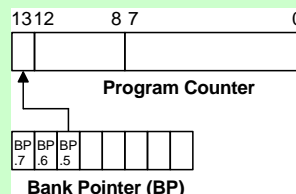
程序计数器为 14 位宽，是指令字的标志器，控制了程序的流程。单片机从程序计数器所指向的程序存储器里抓取指令来执行，程序计数器(PC)最大可以对 16384 个地址进行寻址。

取得指令字以后，程序计数器(PC)会自动指向下一个指令字的地址。PC 将指向包含下一指令码的存储器字。

但是如果执行跳转、条件跳转、写 PCL 寄存器、子程序调用、中断、复位初始化或由中断子程序返回等情况，程序计数器(PC)会载入相对应的地址，而非下一个地址来操作程序流程。

举例而言，如果执行条件单跳跃指令，当条件符合时，则在执行此条指令时装入的下一条指令会被抛弃，且一个空指令周期 (dummy cycle) 会被插入，程序计数器才会指向正确的指令字地址，因此需要两个指令周期。其他情况，程序计数器指向下一条指令字的地址。

程序计数器的低字节单元 (PCL) 是一个可读写的暂存器 (06H)，如果写入一个值将会产生一个短程跳跃动作，这个短程跳跃范围是 256 个地址。同样在此情况下也会插入一个空指令周期。



模 式	程序计数器									
	*13	*12~*8	*7	*6	*5	*4	*3	*2	*1	*0
初始化复位	0	00000	0	0	0	0	0	0	0	0
外部中断 0	0	00000	0	0	0	0	0	1	0	0
外部中断 1	0	00000	0	0	0	0	1	0	0	0
定时/计数器 0 溢出	0	00000	0	0	0	0	1	1	0	0
定时/计数器 1 溢出	0	00000	0	0	0	1	0	0	0	0
UART 中断	0	00000	0	0	0	1	0	1	0	0
多功能中断	0	00000	0	0	0	1	1	0	0	0
短跳转	PC+2									
装载 PCL	*13	*12~*8	@7	@6	@5	@4	@3	@2	@1	@0
跳转, 子程序调用	BP.5	#12~#8	#7	#6	#5	#4	#3	#2	#1	#0
从子程序返回	S13	S12~S8	S7	S6	S5	S4	S3	S2	S1	S0

**程序计数器**

注意: \*13 ~ \*0 : 程序计数器位  
 #12 ~ #0 : 指令代码位

S13~S0 : 堆栈寄存器位  
 @7 ~ @0 : PCL 位

## 程序存储器

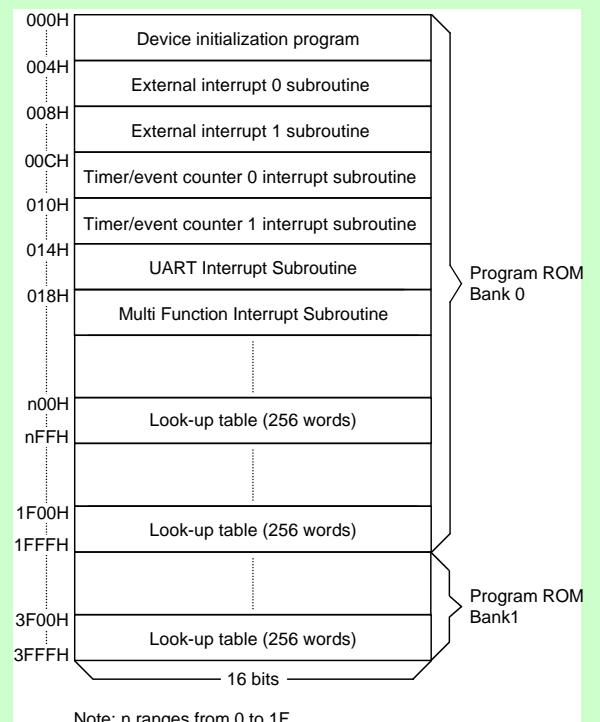
程序存储器 (ROM) 用来存放要执行的指令代码。还包括数据, 表格和中断入口。程序存储器为  $8192 \times 16 \times 2$  个 Bank。程序存储器的空间都可以由程序计数器或表格指针来寻址。BP 寄存器的第 5~7 位用于选择程序存储器。当 BP.7~BP.5 = 000B, 将选中程序存储器 Bank0(0000H~1FFFH); 当 BP.7~BP.5 = 001B, 将选中程序存储器 Bank1 (2000H ~ 3FFFH)。指令 JMP 和 CALL 只包含 13 位地址, 其跳转范围为一个 bank (即 8K)。当执行 JMP 或 CALL 指令, 必须设置 Bank 指针 BP.5, 以确保程序正确执行。若从中断或子程序返回时, 14 位 PC 将从堆栈弹出, 所以在执行 RET 或 RETI 指令时不需要改变 BP.5 值。

下所列出的程序存储器的地址是专为特殊用途而保留的。

- 地址 000H  
此地址保留给程序初始化之用。当系统复位时, 程序会从 000H 地址开始执行。
- 地址 004H  
此地址保留给外部中断服务使用。当中断允许, 且堆栈未滿, 则一旦  $\overline{\text{INT0}}$  输入脚有下降沿信号, 就能产生中断, 程序会从 004H 地址开始执行外部中断服务程序。
- 地址 008H  
此地址保留给外部中断服务程序使用。当中断允许, 且堆栈未滿, 则一旦  $\overline{\text{INT1}}$  输入脚有下降沿信号, 就能产生中断, 程序会从 008H 地址开始执行外部中断服务程序。
- 地址 00CH  
此地址保留给定时/计数器 0 中断服务使用。当中断允许, 且堆栈未滿, 则一旦定时/计数器 0 发生溢出时, 就能产生中断, 程序会从 00CH 地址开始执行中断服务程序。
- 地址 010H  
此地址保留给定时/计数器 1 中断服务使用。当中断允许, 且堆栈未滿, 则一旦定时/计数器 1 发生溢出时, 就能产生中断, 程序会从 010H 地址开始执行中断服务程序。
- 地址 014H  
此地址保留给 UART 中断服务使用。当中断允许, 且堆栈未滿, 则一旦 UART 的 TX 或 RX 产生中断, 程序会从 014H 地址开始执行中断服务程序。
- 地址 018H  
此地址保留给多功能中断服务使用。当多功能中断和相应的中断允许, 且堆栈未滿, 则一旦定时/计数器 2、时基或实时时钟中断产生, 程序会从 018H 地址开始执行中断服务程序。

### • 表格区

ROM 内的任何区域都可用来作为查表区使用。查表指令为 TABRDC [m] (地址由寄存器 TBHP 和 TBLP 指定) 与 TABRDL [m] (查最后一页)。[m] 为数据被存入的地址。在执行 TABRDC [m] 指令 (或 TABRDL [m] 指令) 后, 将会传送当前页 (或最后一页) 上的数据内容的低位字节到 [m], 而数据内容的高位字节传送到 TBLH (08H)。表格指针高位寄存器 TBLP (08H) 和表格指针低位寄存器 TBHP (1FH) 为可读写寄存器, 用来指定需要查表的地址。在访问表格以前, 通过对 TBHP 和 TBLP 寄存器赋值来指明表格地址。TBLH 为只读寄存器, 如果主程序和中断服务程序都用到查表指令, 主程序中 TBLH 的值可能会因为 ISR 中执行的查表指令而发生变化, 产生错误。也就是说, 要避免在主程序和中断服务程序中都使用查表指令。但如果必须这样做



程序存储器

的话，我们可以在查表指令前先将中断禁止，在保存了 TBLH 的值后再开放中断以避免发生错误。所有与表格有关的指令都需要两个指令周期的执行时间。这里提到的表格区都可以做为正常的程序存储器来使用。

指令	表格地址								
	*13~*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC[m]	TBHP	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL[m]	111111	@7	@6	@5	@4	@3	@2	@1	@0

表格区

注意：\*13~\*0 : 表格地址位                      TBHP : 表格指针高位  
 @7 ~ @0 : 表格指针低位

### 堆栈寄存器 — STACK

堆栈寄存器 (STACK) 是一个用来保存 PC 值的特殊存储单元，此系列有 16 层堆栈。堆栈寄存器既不是数据存储器的一部分，也不是程序存储器的一部分，而且它既不能读出，也不能写入。栈将要弹出的地址的由堆栈指针 (SP) 来索引，堆栈指针也不能读出与写入。一旦发生子程序调用或中断被响应时，程序计数器 (PC) 的值会被压入堆栈，在子程序调用结束或中断被响应结束时 (执行指令 RET 或 RETI)，堆栈将原先压入堆栈的内容弹出，重新装入程序计数器中。在系统复位后，堆栈指针会指向堆栈顶部。

如果堆栈满了，并且发生了不可屏蔽的中断，那么中断请求标志将会被记录下来，而不会响应该中断，一旦堆栈指针 (由 RET 或 RETI) 发生递减时，才会响应中断。这个功能防止堆栈溢出，使得程序员易于使用这种结构。但是堆栈已满，接着又执行一个子程序调用 (CALL)，那么堆栈会产生溢出，而使首先进入堆栈的内容丢失。堆栈只会保留最近的 16 个返回地址。

## 数据存储器

不包括 LCD 存储器，数据存储器 RAM 由 608×8 位组成。它可分成两部分，即特殊功能寄存器和通用数据存储器。这两部分的大多数单元可以读写，但有一些是只读的。通用数据存储器分为 3 个 Bank(Bank0、Bank2 和 Bank3)，每个 Bank 为 192×8 位。Bank 指针 BP 用于选存储区段(Bank)，由于 BP 也用于选择程序存储器，所以在修改 BP 值必须注意。

BP	RAM Bank
00000	0
00001	1
00010	2
00011	3

通用数据存储器范围是从 40H~FFH (bank0,2,3)，用于指令中数据和控制信息的存储。

所有的 RAM 区单元都能直接执行算术、逻辑、递增、递减和移位等运算。除了某些特殊的位以外，RAM 中的每一位都可以由 SET[m].i 和 CLR[m].i 指令来置位和复位。这些 RAM 都可通过存储器指针寄存器 0(MP0; 01H)和存储器指针寄存器 1(MP1; 03H)来间接寻址。

Bank1 是 LCD 显示存储器，其中每一位对应到 LCD 面板上每一个点。

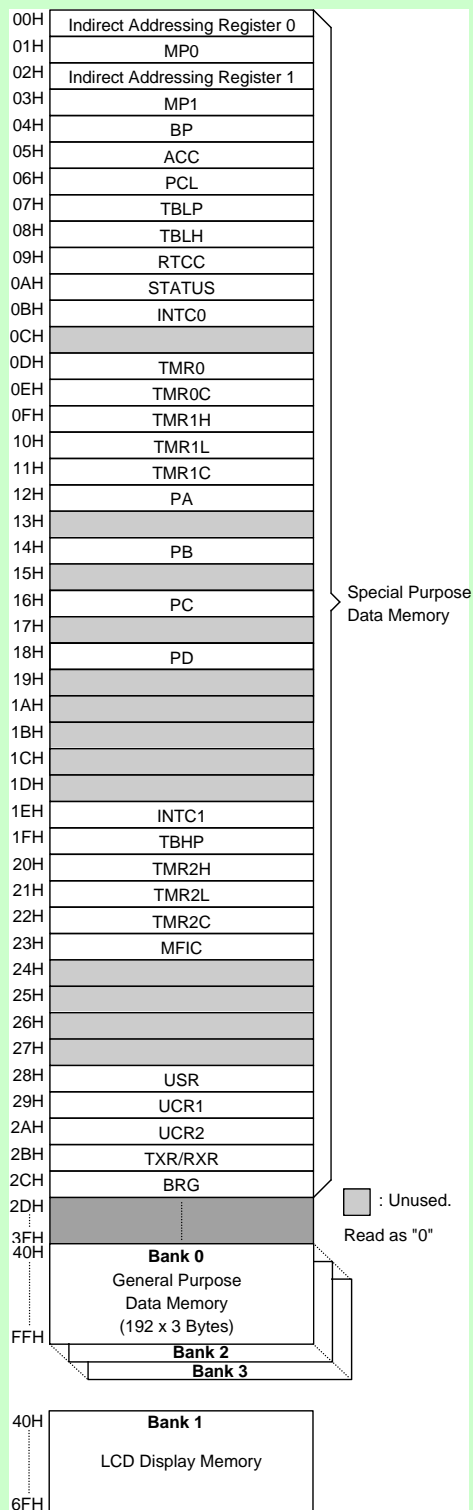
### 间接寻址寄存器

地址 00H 和 02H 作为间接寻址寄存器。它们都没有实际的物理结构。任何对[00H]和[02H]的读/写操作，都会访问由 MP0[01H]和 MP1[03H]所指向的 RAM 单元。间接地读取 00H 或 02H 单元，将会得到值 00H，而间接地写入 00H 或 02H 单元，则不会产生任何结果。

间接寻址寄存器之间不支持数据传送功能。间接寻址指针寄存器 MP0 和 MP1 的宽度都是 8 位，被用来访问由间接寻址寄存器所对应的 RAM。MP0 只能用于数据存储器，而 MP1 能用于数据存储器和 LCD 显示存储器。

### 累加器 — ACC

累加器 (ACC) 与算术逻辑单元 (ALU) 运算相联系。它对应于 RAM 的地址 05H，并能对立即数进行操作，在数据存储器间的数据传送都必须经过累加器。



数据寄存器

**算术逻辑单元 — ALU**

算术逻辑单元是执行 8 位算术，逻辑运算的电路。它提供如下的功能：

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

算术逻辑单元 ALU 不仅会保存运算的结果而且会改变状态寄存器的值。

**状态寄存器 — STATUS**

状态寄存器 (0AH) 的宽度为 8 位，由零标志位 (Z)、进位标志位 (C)、辅助进位标志位 (AC)、溢出标志位 (OV)、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。

除了 TO 和 PDF 以外，状态寄存器中的位都可用指令来改变。任何写到状态寄存器的数据不会改变 TO 或 PDF 标志位。注意与状态寄存器有关的操作可能会导致与预期不一样的结果。系统上电、看门狗定时器溢出、执行 HALT 指令或清除看门狗定时器都能改变 TO 和 PDF 标志位。而 Z、OV、AC 和 C 标志位都反映了最近一次操作的状态。

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

符号	位	功 能
C	0	如果在加法运算中结果产生了进位或在减法运算中结果不产生借位,那么被 C 置位; 反之, C 被清除。它也可被一个循环移位指令影响。
AC	1	在加法运算中低四位产生了进位或减法运算中在低四位不产生借位, AC 被置位; 反之, AC 被清除。
Z	2	算术运算或逻辑运算的结果为零则 Z 被置位; 反之, Z 被清除。
OV	3	如果运算结果次高位向最高位进位, 但最高位并不产生进位输出, 或情况相反, 则 OV 被置位; 反之, OV 被清除。
PDF	4	系统上电或执行了 CLR WDT 指令, PDF 被清除。执行 HALT 指令 TD 被置位。
TO	5	系统上电或执行了 CLR WDT 指令, 或 HALT 指令, TO 被清除。WDT 定时溢出, TO 被置位。
—	6	未定义, 读出为“0”
—	7	未定义, 读出为“0”

**STATUS(0AH) 寄存器**

## 中断

系统提供二个外部中断、三个内部定时/计数器中断、一个内部时基中断、一个实时时钟中断和一个 UART 中断。中断控制寄存器 0 (INTC0; 0BH) 和中断控制寄存器 1 (INTC1; 1EH) 包含了中断控制位以及中断请求标志位，中断控制位是用来设置中断允许/禁止。

一旦有中断响应，其它的中断都被禁止（通过清除 EMI 位）。这种方法目的在于防止中断嵌套。这时如有其它中断请求发生，只会把这个中断请求的标志记录下来。如果在一个中断服务程序中有另一个中断需要响应的话，程序员可以设置 EMI 位及 INTC0 或 INTC1 所对应位来允许中断嵌套服务。如果堆栈已满，即使相关的中断允许，该中断请求也不会被响应。直到堆栈指针发生递减时才会响应。如果需要立即得到中断服务，则必须避免堆栈饱和。

所有的中断都具有唤醒功能。当中断响应时，系统会将程序计数器 (PC) 压入堆栈，转移到指定的 ROM 地址的中断入口。只有程序计数器 (PC) 的内容压入堆栈。如果寄存器和状态寄存器的内容会被中断服务程序改变，从而破坏主程序的执行流程的话，那么程序员必须事先将这些数据保存起来。

外部中断是由  $\overline{\text{INT0}}$  或  $\overline{\text{INT1}}$  脚下降沿信号触发的，会置位相关的中断请求位 (EIF0, INTC0 的第 4 位, EIF1, INTC1 的第 5 位)。当中断允许，堆栈未满，外部中断触发时，那么将会产生 04H 或 08H 单元的子程序调用。系统将会清除中断请求标志 (EIF0 或 EIF1) 和 EMI 位以禁止其他的中断响应。

内部定时/计数器 0 中断产生，会置位定时/计数器 0 中断请求标志位 (T0F, INTC0 的第 6 位)，中断的请求标志一般是由定时/计数器 0 溢出产生的。当中断允许，堆栈未满，并且 T0F 已被置位，就会产生 0CH 单元的子程序调用。系统将会清除中断请求标志位 (T0F) 和 EMI 位，以禁止其他中断的响应。内部定时/计数器 1 的运作方式与之相同，相关的中断请求标志位是 T1F (INTC1 的第 4 位)，而它的子程序调用的地址是 10H 单元。

异步串行中断产生，会置位 UART 中断请求标志 (URF, INTC1 的第 5 位)，中断请求标志一般是由 UART 发送或接收完成产生的。当中断允许，堆栈未满，并且 URF 已被置位，就会产生 14H 单元的子程序调用。系统将清除中断请求标志位 (URF) 和 EMI 位，以禁止其他中断的响应。

多功能中断产生，会置位多功能中断请求标志 (MFF, INTC1 的第 6 位)，中断请求标志一般是由定时/计数器 2 溢出、时基信号或实时时钟信号产生的。当中断允许，堆栈未满，并且 MFF 已被置位，就会产生 18H 单元的子程序调用。系统将清除中断请求标志位 (MFF) 和 EMI 位，以禁止其他中断的响应。

单片机在执行中断子程序期间，会屏蔽其他的中断请求，直到执行 RETI 指令或是 EMI 位和相关的中断控制位都被置 1 (堆栈未满时)。若要从中断子程序返回时，只要执行 RET 或 RETI 指令。RETI 指令将会自动置位 EMI 允许中断服务，而 RET 则不会自动置位 EMI。

如果中断在两个连续的 T2 脉冲的上升沿之间发生，且中断被允许的话，那么在后来的二个 T2 周期，该中断会被响应。如果同时发生中断服务请求，那么下列表中列出了中断服务优先等级。这种优先权地位也可以通过 EMI 位的复位来屏蔽。

中 断 源	优 先 权	中 断
外部中断 0	1	04H
外部中断 1	2	08H
定时/计数器 0 中断	3	0CH
定时/计数器 1 中断	4	10H
UART 中断	5	14H
多功能中断 (定时器 2、时基或实时时钟)	6	18H

在中断子程序中建议不要使用“调用子程序”指令。中断通常发生在不可预期的情况或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使能，当“CALL”指令在中断子程序执行时，

将破坏原来的控制序列。

位	符号	功能
0	EMI	总中断控制位 允许=1, 禁止=0
1	EEI0	外部中断 0 控制位 允许=1, 禁止=0
2	EEI1	外部中断 1 控制位 允许=1, 禁止=0
3	ET0I	定时/计数器 0 中断控制位 允许=1, 禁止=0
4	EIF0	外部中断 0 请求标志位 有=1, 无=0
5	EIF1	外部中断 1 请求标志位 有=1, 无=0
6	T0F	内部定时/计数器 0 中断请求位 有=1, 无=0
7	—	未使用位, 读数为“0”

**INTC0(0BH) 寄存器**

位	符号	功能
0	ET1I	定时/计数器 1 中断控制位 允许=1, 禁止=0
1	EURI	UART 中断控制位 允许=1, 禁止=0
2	EMFI	多功能中断控制位 允许=1, 禁止=0
3	—	未使用位, 读数为“0”
4	T1F	定时/计数器 1 中断请求位 有=1, 无=0
5	URF	UART 中断请求位 有=1, 无=0
6	MFF	多功能中断请求位 有=1, 无=0
7	—	未使用位, 读数为“0”

**INTC1(1EH) 寄存器**

位	符号	功能
0	ET2I	定时/计数器 2 中断控制位 允许=1, 禁止=0
1	ETBI	时基中断控制位 允许=1, 禁止=0
2	ERTI	实时时钟中断控制位 允许=1, 禁止=0
3	—	未使用位, 读数为“0”
4	T2F	定时/计数器 2 中断请求位 有=1, 无=0
5	TBF	时基中断请求位 有=1, 无=0
6	RTF	实时时钟中断请求位 有=1, 无=0
7	—	未使用位, 读数为“0”

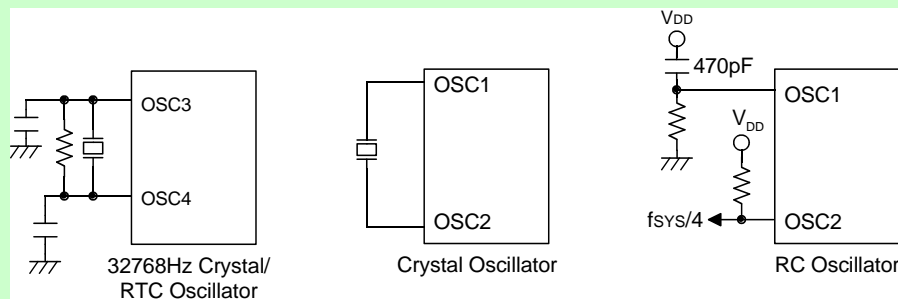
**MFIC(23H) 寄存器**

## 振荡器

系统提供三种振荡电路以供选择：外部晶体振荡、外部 RC 振荡器或外部 32768Hz 晶体振荡，可在配置选项中确定。不管用哪种振荡方式，都可以作为系统时钟来源。HALT 模式会停止系统振荡器（只是指 RC 振荡器和晶体振荡器）并忽视任何外部信号以降低功耗。如果选择 32768Hz 振荡做为系统振荡，在 HALT 模式下系统振荡不会停止，但是指令会停止运行。32768Hz 的晶体振荡还可以做为内部计数器的时钟源，即使进入 HALT 模式，这些计数器(RTC、时基、WDT)还会继续作用。

如果采用 RC 振荡，那么在 OSC1 与 VSS 之间要接一个外接电阻，其阻值范围为 24KΩ~1MΩ。在 OSC2 端接一个上拉电阻，可获得系统时钟四分频信号输出，它可以用于同步系统外部逻辑。RC 振荡是一种低成本方案，但是振荡频率会随 V<sub>DD</sub>、温度变化及芯片自身参数的漂移而产生误差。因此，它不能满足要求精确的振荡频率的应用场合。

在 OSC1 与 OSC2 之间连接一个晶体，用来提供晶体振荡所要的反馈 (Feedback) 和相移 (Phase Shift)。除此以外，不再需要其他外部元件。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。



系统振荡器

另外一种振荡电路被设计用于实时时钟 (RTC)。在这种振荡电路中，只能用 32.768KHz 的晶体。晶体被连接在 OSC3 与 OSC4 之间。

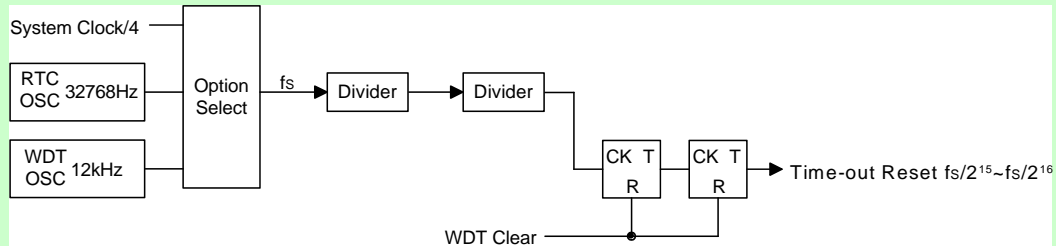
RTC 振荡器可以通过设置“QOSC”位 (RTCC 的第 4 位) 来控制快速振荡。系统上电时会启动快速振荡功能，建议在 2 秒钟后关闭它以降低功耗。

WDT 看门狗振荡器是一个内部的 RC 振荡器。不需连接任何外部元件。当系统进入暂停模式时，系统时钟会停止，但是 WDT 振荡器仍然以大约 65μs@5V 的周期工作着。如果要降低功耗，可在配置选项中关闭 WDT 振荡器。

## 看门狗定时器

WDT 的时钟源是由专用的 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。看门狗定时器主要用来防止程序运行故障和程序跳入一死循环而导致不可预测的结果。WDT 可以由配置选项设置为打开或关闭。如 WDT 关闭, 所有与 WDT 有关的执行都等同一个空操作。

在选定了 WDT 的时钟源后, 它的溢出周期为  $f_s/2^{15} \sim f_s/2^{16}$ 。



看门狗定时器

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、V<sub>DD</sub> 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。

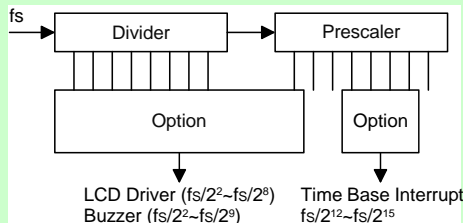
在正常运作下, WDT 溢出会使系统复位并置位 TO。但在 HALT 模式下, 溢出执行热复位, 只复位 PC 程序计数器和堆栈指针 SP。要清除 WDT 的值 (包括 WDT 预分频器) 可以有三种方法: 外部复位 (低电平输入到 RES 端)、软件指令和 HALT 指令三种。软件指令由 CLR WDT 和 CLR WDT1 及 CLR WDT2 二组指令组成。这两组指令中, 只能选取其中一种。选择的方式由配置选项的 CLR WDT 次数选项决定。如果“CLR WDT”被选择 (即 CLR WDT 次数为 1), 那么只要执行 CLR WDT 指令就会清除 WDT。在 CLR WDT1 和 CLR WDT2 被选择的情况下 (即 CLR WDT 次数为 2), 那么要交替执行二条指令才可清除 WDT, 否则, 会由于 WDT 的溢出而使系统复位。

## 多功能定时器

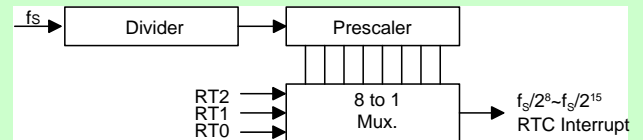
系统为 WDT、时基和 RTC 提供了具有不同溢出周期的多功能定时器。多功能定时器由一个七级预分频器和一个 8 位分频器组成。使用的时钟源来自 WDT OSC、RTC OSC 或指令时钟（系统时钟四分频）。多功能定时器为 LCD 驱动电路提供可选择的频率信号（范围从  $f_s/2^2 \sim f_s/2^8$ ），并为蜂鸣器输出电路提供可选择的频率信号（范围从  $f_s/2^2 \sim f_s/2^9$ ），频率由掩膜选择。为了正确地显示，建议选择 4kHz 左右的信号作为 LCD 驱动信号。

## 时基

时基指的是用一个周期性的溢出来产生一个有规律的内部中断。溢出周期的范围从  $f_s/2^{12} \sim f_s/2^{15}$ ，由配置选项确定。如果时基溢出产生，系统置位相关的中断请求标志位（TBF）。如果中断允许，堆栈未满，那么就产生一个地址在 18H 的子程序调用。时基溢出信号也能被作为定时/计数器 1 的时钟源来获得更长的溢出周期。



时基



实时时钟 (RTC)

## 实时时钟

实时时钟 (RTC) 的运作情况和时基是一样的。它是用来提供一个有规律的内部中断。它的溢出周期的范围从  $f_s/2^8 \sim f_s/2^{15}$ ，可通过软件编程实现。写数据到 RT2, RT1, RT0 (RTCC 的第 2, 1, 0 位; 09H) 将产生各种溢出周期。如果 RTC 溢出产生，相关的中断请求标志位 (RTF) 被置位。如果中断是允许的，并且堆栈未满，那么就产生一个地址在 18H 的子程序调用。实时时钟溢出信号也能被作为定时/计数器 0 的时钟源来获得更长的溢出周期。

RT2	RT1	RT0	RTC 实时时钟分频级数
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	$2^{15}$

注意：“\*”不建议使用

## 暂停模式

暂停模式是由 HALT 指令来实现的，有如下功能：

- 系统振荡器关闭，但 WDT 或 RTC 振荡器会继续运行（如果选择 WDT 振荡器或 RTC）。
- RAM 及寄存器的内容保持不变。
- WDT 被清除并重新计数（如果 WDT 的时钟是来自 WDT 振荡器或 RTC）。
- 所有的输入/输出端口都保持其原先状态。
- 置位 PDF 标志位，清除 TO 标志位。
- LCD 驱动器仍然运行（如果 WDT OSC 或 RTC OSC 被选择）。

外部复位、中断、外部输入一个下降沿的信号到 PA 口，或 WDT 溢出均可使系统唤醒。外部复位能使系统初始化，而 WDT 溢出唤醒复位为热复位。可以通过判断 TO 和 PDF 状态，确定系统复位的原因。系统上电复位和执行 CLR WDT 指令，可清除 PDF 标志位；而执行 HALT 指令，则会置位 PDF。如 WDT 产生溢出，置位 TO 标志位，还能产生唤醒，但只复位程序计数器 PC 和堆栈指针 SP，其他都保持原状态。

PA 口唤醒和中断唤醒可作为正常运行的继续，PA 口的每一位都可以通过配置选项来单独设定为对系统的唤醒。如果是输入输出唤醒的话，程序会从 HALT 后的下一条命令继续执行。如果是中断唤醒的话，则会产生二种情况：如果相关的中断被禁止或中断是允许的，但堆栈已满，程序会从 HALT 后的下一条命令继续执行。如果中断允许并且堆栈未滿，那么会相应这个中断。

如果进入暂停模式以前，某个中断请求位已经被置位，那么系统不能用这个中断来唤醒。当唤醒事件发生时，要延迟  $1024t_{SYS}$ （系统时钟周期），系统才会重新正常运行。这就是说，在唤醒后被插入了一个等待时间。如果是中断唤醒，那么实际中断程序的执行就被延迟了一个周期以上。如果唤醒导致下一条指令执行，那么在一个等待周期结束后指令就立即被执行。

为了减小功耗，在进入暂停模式之前必须要小心处理输入输出状态。

## 复位

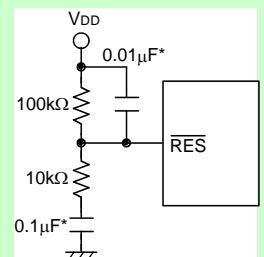
总共有三种方法会产生初始复位：

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

暂停模式中的看门狗定时器溢出与其它系统复位状况不同，因为看门狗定时器溢出会执行热复位，热复位只复位程序计数器 PC 和堆栈指针 SP，而系统其它部分都保持原有状态。在其它复位状态下，某些寄存器不会改变。在初始复位时，大部分寄存器会复位成初始的状态。通过检测 PDF 和 TO 标志，即可判断出各种不同的复位原因。

TO	PDF	复位条件
0	0	电源上电复位
u	u	正常运作时由 $\overline{RES}$ 发生复位
0	1	由 $\overline{RES}$ 唤醒暂停模式
1	u	正常运作时发生看门狗定时器超时
1	1	由看门狗定时器唤醒暂停模式

注意：“u”表示未变。



**复位电路**

注意：为了避免噪声干扰，连接  $\overline{RES}$  引脚的线请尽可能地短

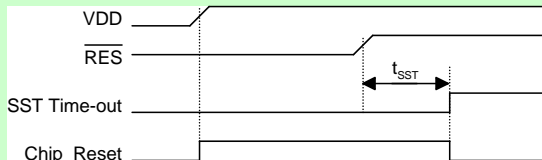
为了保证系统振荡器起振并稳定运行，系统复位(包括上电复位、看门狗定时器溢出或由  $\overline{\text{RES}}$  端复位)或由暂停状态唤醒时，系统启动定时器(SST)提供了一个额外的延迟时间，共 1024 个系统时钟周期。

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

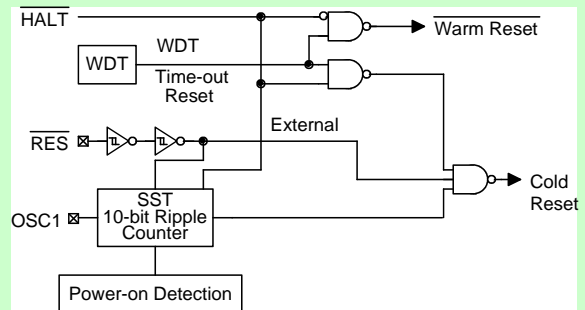
当系统上电、正常运行时 WDT 溢出或  $\overline{\text{RES}}$  脚复位，系统需要额外增加一个加载 Option 的时间。

系统复位时各功能单元的状态如下所示：

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



复位时序图



复位框图

**特殊功能寄存状态概述表**

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)*
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
Program Counter	0000H	0000H	0000H	0000H	0000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
RTCC	0000 0111	0000 0111	0000 0111	0000 0111	00uu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1---	0000 1---	0000 1---	0000 1---	uuuu u---
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	0000 1---	0000 1---	0000 1---	0000 1---	uuuu u---
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	-111 1111	-111 1111	-111 1111	-111 1111	-uuu uuuu
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
TBHP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR2H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2C	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---
MFIC	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
USR	0000 1011	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TXR/RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

注意：1.“\*”表示热复位。  
 2.“u”表示不变化。  
 3.“x”表示不确定。

## 定时/计数器

系统提供三个定时/计数器。一个 8 位可编程的向上计数的计数器；两个 16 位可编程的向上计数的计数器。定时/计数器 0 的时钟来源可以是系统时钟、指令时钟（四分频的系统时钟）、RTC 溢出信号或是外部的时钟源。系统时钟源或是系统时钟四分频通过配置选项来设定。定时/计数器 1 的时钟来源可以是来自 TMR0 的溢出、系统时钟、时基的溢出信号、系统时钟四分频或是外部时钟源，前三项时钟源由配置选项确定。外部时钟输入允许用户去计算外部事件，测量时间间隔或脉宽、或产生一个精确的时基信号；而使用内部时钟的话，则允许用户去产生一个精确的时基信号。定时/计数器 2 是 16 位向上计数器，其时钟来源可以是外部信号输入或内部时钟，内部时钟为系统时钟四分频。外部信号输入可以用来计数外部事件、测量时间间隔、测量脉宽或产生一个精确的时基信号。

有两个寄存器与定时/计数器 0 相关联，即 TMR0 ([0DH]) 和 TMR0C ([0EH])。写入 TMR0 会将初始值装入到定时/计数器的预置寄存器中，而读 TMR0 则会获得定时/计数器的内容。TMR0C 是定时/计数器控制寄存器，其定义操作模式、计时允许或禁止、触发边缘。

有三个寄存器与定时/计数器 1 相关联，即 TMR1H ([0FH])、TMR1L ([10H]) 和 TMR1C ([11H])。有两个物理寄存器对应 TMR1 的位置。若写入 TMR1L 只能将数据写入低字节内部缓冲器中，但若写入的是 TMR1H 则可将数据和低字节内部缓冲器的内容写入定时/计数器加载寄存器 (16 位) 之中。改变定时/计数器加载寄存器的内容，只可被写入 TMR1H 之动作改变，但若写入 TMR1L 则可维持定时/计数器加载寄存器的内容不受改变。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 TMR1H，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。TMR1C 是定时/计数器控制寄存器，其定义某些选项。

有三个寄存器与定时/计数器 2 相关联，即 TMR2H ([20H])、TMR2L ([21H]) 和 TMR2C ([22H])。有两个物理寄存器对应 TMR2 的位置。若写入 TMR2L 只能将数据写入低字节内部缓冲器中，但若写入的是 TMR2H 则可将数据和低字节内部缓冲器的内容写入定时/计数器加载寄存器 (16 位) 之中。改变定时/计数器加载寄存器的内容，只可被写入 TMR2H 之动作改变，但若写入 TMR2L 则可维持定时/计数器加载寄存器的内容不受改变。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 TMR2H，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。TMR2C 是定时/计数器控制寄存器，其定义某些选项。

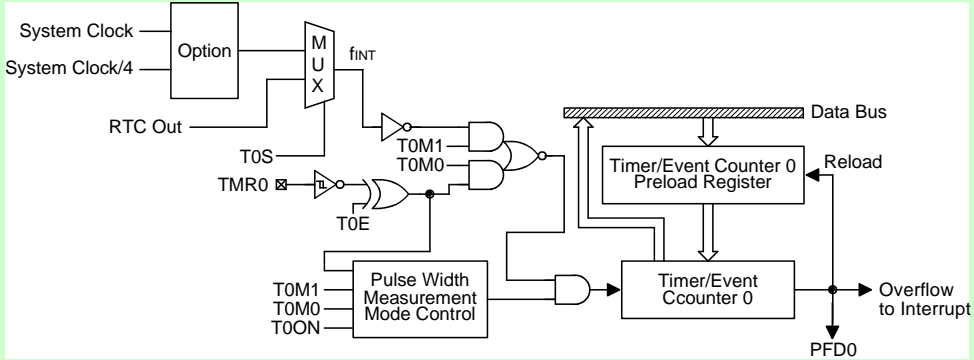
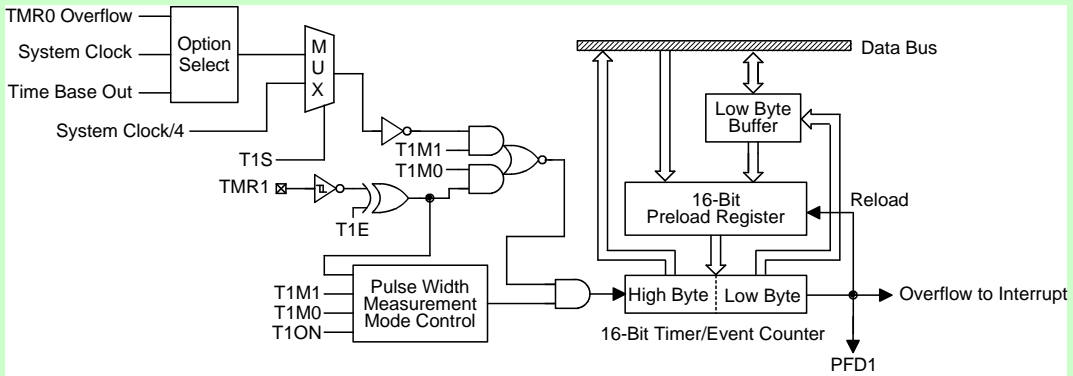
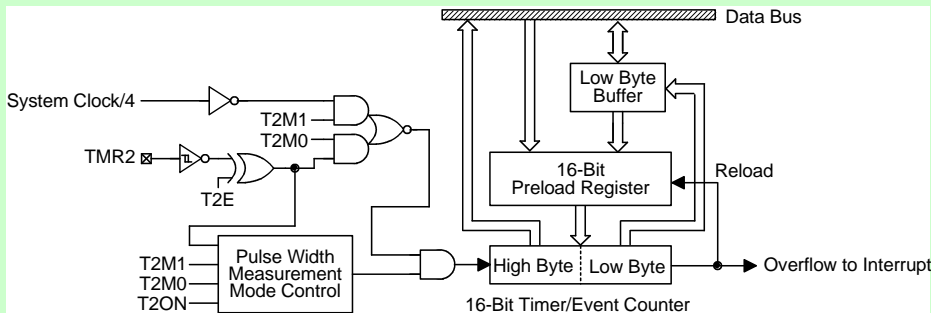
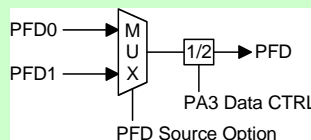
T0M0、T0M1 (TMR0C)，T1M0、T1M1 (TMR1C) 和 T2M0、T2M1 (TMR2C) 位，定义工作模式。外部事件计数模式用来记录外部事件，时钟来源由外部 (TMR0/TMR1/TMR2) 引脚输入。定时器模式是作为一个普通的定时器功能，时钟来源为内部时钟。脉冲宽度测量模式能用来测量外部引脚 (TMR0/TMR1/TMR2) 上的高电平或低电平的宽度，时钟来源是内部时钟。

在外部事件计数或定时器模式中，定时/计数器会从当前定时/计数器中的数值开始向上计数，到 OFFH (OFFFH) 结束。如果产生溢出，计数器会从定时/计数器预置寄存器重新装载计数值并且同时产生相应的中断请求标志 (T0F; INTC0 第 6 位，T1F; INTC1 的第 4 位，T2F; MFIC 的第 4 位)。

在脉冲宽度测量模式时，其 T0ON/T1ON/T2ON 和 T0E/T1E/T2E 位皆为 1 时，如果引脚 TMR0/TMR1/TMR2 接收到一个上升沿信号 (如 T0E/T1E/T2E 值为 0，则为下降沿信号) 时，计数器会开始计数直至 TMR0/TMR1/TMR2 引脚回到原来的电平为止，并且会将 T0ON/T1ON/T2ON 清零，只有这种模式 T0ON/T1ON/T2ON 会自动清零，其它模式 T0ON/T1ON/T2ON 位只可以用指令清除。计数器停止计数，测量的结果则保留在定时/计数器之中，而且即使再收到一个跳变信号也不会改变。换句话说，脉冲宽度测量模式一次只能测量一个脉冲。只要 T0ON/T1ON/T2ON 位又被置位，则当引脚 TMR0/TMR1/TMR2 接到跳变脉冲，测量周期会再次执行下去。在脉冲宽度测量模式中，定时/计数器并不会根据逻辑电平来计数，其根据的标准为信号的跳变沿。一旦发生计数器溢出，计数器会从定时/计数器加载寄存器重新装入，同时还会发出中断请求，这个情况和外部事件计数模式和定时器模式一样。

要启动计数运作，只要将定时器启动 ON 位 (T0ON/T1ON/T2ON; TMR0C/TMR1C/TMR2C 的第 4 位) 被置成为 1。在脉宽测量模式中 T0ON/T1ON/T2ON 在测量周期结束后自动被清除。但在另外两个模式中，T0ON/T1ON/T2ON 只能由指令来复位。定时/计数器的溢出是唤醒的信号源之一。

并且可由配置选项，设定 PA3 用作为 PFD 输出（可编程分频器）。掩膜只有一个 PFD 信号（PFD0 或 PFD1）提供给 PA3。不管处于何种模式，若写 0 到 ET0I/ET1I/ET2I 位即可禁止相应的中断服务。当 PFD 功能被选时，执行“CLR [PA].3”指令来允许 PFD 输出和执行“SET [PA].3”指令来禁止 PFD 输出。


**定时/计数器 0**

**定时/计数器 1**

**定时/计数器 2**

**PFD 时钟来源选择**

位	名称	功能
0~2	—	未定义, 读取时为“0”
3	T0E	定义定时/计数器 TMR0 作用沿: 外部事件计数器模式(T0M1, T0M0)=(0, 1) 1: 下降沿计数 0: 上升沿计数 脉冲宽度测量模式(T0M1, T0M0)=(1, 1) 1: 上升沿开始计数, 下降沿停止计数 0: 下降沿开始计数, 上升沿停止计数
4	T0ON	允许/禁止定时器计数 (0=禁止, 1=允许)
5	T0S	2 选 1 多通道选择, 控制定时/计数器的时钟来源 (0=RTC 输出,1=系统时钟或是系统时钟 4 分频)
6 7	T0M0 T0M1	定义工作模式 (T0M1, T0M0) 01=计数器模式 (外部时钟) 10=定时器模式 (系统时钟) 11=脉冲宽度测量模式 (外部时钟) 00=未定义

**TMR0C(0EH) 寄存器**

位	名称	功能
0~2	—	未定义, 读取时为“0”
3	T1E	定义定时/计数器 TMR1 作用沿: 外部事件计数器模式(T1M1, T1M0)=(0, 1) 1: 下降沿计数 0: 上升沿计数 脉冲宽度测量模式(T1M1, T1M0)=(1, 1) 1: 上升沿开始计数, 下降沿停止计数 0: 下降沿开始计数, 上升沿停止计数
4	T1ON	允许/禁止定时器计数 (0=禁止, 1=允许)
5	T1S	2 选 1 多通道选择, 控制定时/计数器的时钟来源 (0=配置选项时钟,1=系统时钟 4 分频)
6 7	T1M0 T1M1	定义操作方式 (T1M1, T1M0) 01=计数器模式 (外部时钟) 10=定时器模式 (内部时钟) 11=脉冲宽度测量模式 (外部时钟) 00=未定义

**TMR1C(11H) 寄存器**

在定时/计数器为关闭的状态下, 写数据到定时/计数器的预置寄存器之中, 同时也会将数据装入定时/计数器中。但是若定时/计数器处于工作状态, 写到定时/计数器的数据只会被保留在定时/计数器的预置寄存器中, 直到定时/计数器发生计数溢出才会将数据从预置寄存器加载到定时/计数器寄存器中。

如果要读取定时/计数器数据 (读 TMR0/TMR1/TMR2), 计数会被停止。计数停止会导致计数错误, 所以程序员必须注意到这一点。

强烈建议首先装载一个指定的值到 TMR0/TMR1/TMR2 寄存器, 然后启动定时/计数器作正常的运作。因为 TMR0/TMR1/TMR2 的初始值是不确定的。

鉴于定时/计数器的配置, 在任何要使用定时/计数器的时候, 为了避免不可预料的结果, 用户都必须特别注意, 第一次开启然后关闭定时/计数器。在这以后定时/计数器开始正常工作。下面给出一个例子, 用一个 8 位的定时器和一个 16 位的定时器叠加成一个 24 位的定时器。

位	名称	功能
0~2	—	未定义，读取时为“0”
3	T2E	定义定时/计数器 TMR2 作用沿： 外部事件计数器模式(T2M1, T2M0)=(0, 1) 1: 下降沿计数 0: 上升沿计数 脉冲宽度测量模式(T2M1, T2M0)=(1, 1) 1: 上升沿开始计数，下降沿停止计数 0: 下降沿开始计数，上升沿停止计数
4	T2ON	允许/禁止定时器计数 (0=禁止, 1=允许)
5	—	未定义，读取时为“0”
6 7	T2M0 T2M1	定义操作方式 (T2M1, T2M0) 01=计数器模式 (外部时钟) 10=定时器模式 (内部时钟) 11=脉冲宽度测量模式 (外部时钟) 00=未定义

**TMR2C(22H) 寄存器**

START:

```

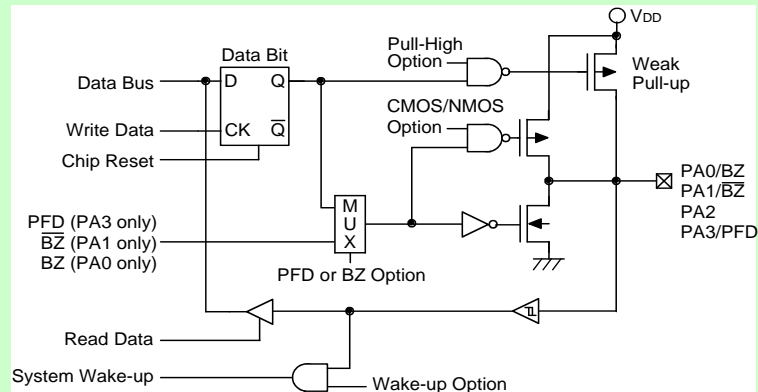
mov    a, 09h           ;置位 ETOI 和 ETI
mov    intc0, a         ;来开放定时/计数器 0 中断和主中断
mov    a, 01h           ;置位 ET1I
mov    intc1, a         ;来开放定时/计数器 1 中断
mov    a, 80h           ;设定定时/计数器 1 工作在计时模式
mov    tmr1c, a         ;并且选择掩膜时钟为时钟来源
mov    a, 0a0h          ;设定定时/计数器 0 工作在计时模式
mov    tmr0c, a         ;并且选择系统时钟 4 分频为时钟源
set    tmr1c.4          ;第一次开启定时/计数器 1
clr    tmr1c.4          ;第一次关闭定时/计数器 1
mov    a, 00h           ;为 TMR0/TMR1 赋初值
mov    tmr0, a
mov    a, 00h
mov    tmr1l, a
mov    tmr1h, a
set    tmr0c.4          ;正常工作
set    tmr1c.4

```

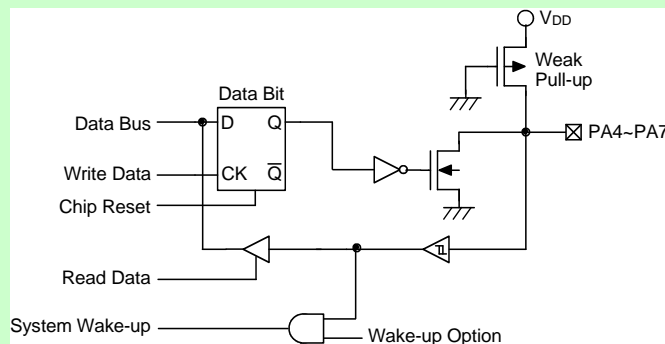
END

## 输入/输出端口

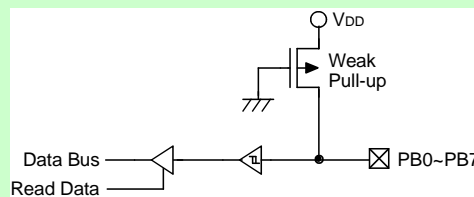
系统包含两个 8 位的双向输入/输出端口 PA 和 PC，一个 8 位的输入端口 PB 和一个 7 位输出端口 PD。PA、PB、PC 和 PD，其分别对应 RAM 的[12H]、[14H]、[16H]和[18H]。由配置选项确定，PA0~PA3 可以被设置成 CMOS 输出，或带或不带上拉电阻的 NMOS 输入/输出。PA4~PA7 总是带上拉电阻和作为 NMOS 输入/输出。如果选择为 NMOS 输入，PA 的每一位带都有唤醒功能。PB 只能作为输入端口。PC 可以被配置成 CMOS 输出，或带或不带上拉电阻的 NMOS 输入/输出。PD 只能作为 CMOS 输出端口使用。PA，PB 和 PC 作为输入端口时，不具有锁存功能，即输入数据必须在 MOV A, [m] (m=12H,14H 或 16H) 指令的 T2 上升沿被准备好。对 PA、PC 和 PD 输出端口而言，所有的数据被锁存并保持不变，直到输出锁存器被改写。



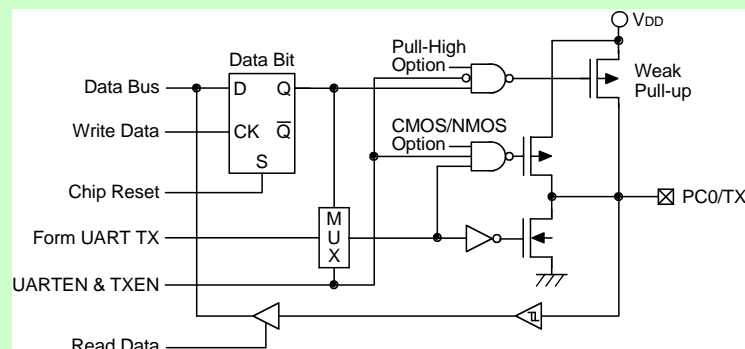
**PA0~PA3 输入/输出端口**



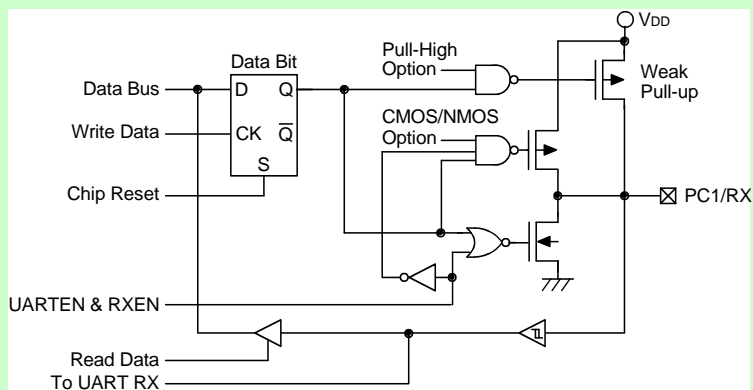
**PA4-PA7 输入/输出端口**



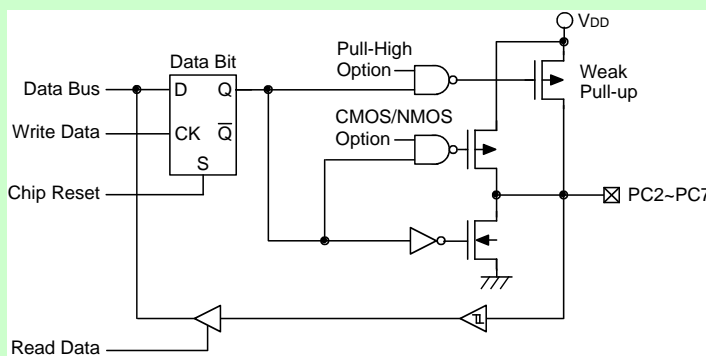
**PB 输入端口**



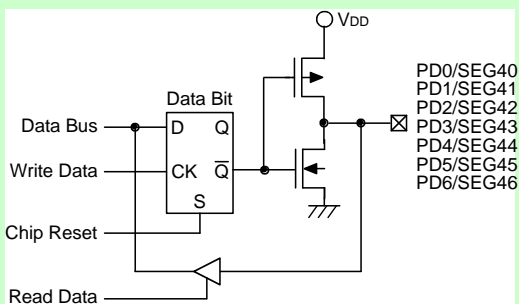
PC0/TX 输入/输出端口



PC1/RX 输入/输出端口



PC2-PC7 输入/输出端口



PD 输出端口

当 PA 和 PC 的结构为漏极开路 NMOS 型时，必须要注意：从引脚读数据前，应该先写“1”到相关位来禁止 NMOS 器件。既首先执行“SET[m].i”（i=0~7 仅对 PA 而言）来禁止相关位的 NMOS 器件，然后“MOV A, [m]”来获得稳定的数据。

某些指令先输入数据然后进行输出操作。例如，SET [m].i，CLR [m].i，CPL [m]和 CPLA [m] 指令，读取整个输入口的状态到 CPU，执行这个被指定的操作（位操作），然后将结果写回这个锁存器或累加器。当 PA 或 PC 口作为输入输出口时，相应的配置选项应该设为带或不带上拉电阻的 NMOS。一旦设为 CMOS 输出，就不能作为输入输出口使用了。

PA 或 PC 口输入是从相应的 PA 或 PC 管脚读取的。当作为带或不带上拉电阻的 NMOS 时，用户需要小心处理与之有关的读-改-写指令。因为读-改-写指令会先将整个口状态读入，进行相应的操作，然后将结果写回到数据存储器。当进行读操作时，默认的管脚状态（由于负载影响或悬空状态产生）会被读入，就会产生错误。

有三个功能脚与 PA 口复用：PA0/BZ，PA1/ $\overline{\text{BZ}}$ 和 PA3/PFD。BZ 和 $\overline{\text{BZ}}$ 是蜂鸣驱动输出端，PFD 是可编程分频输出端。如果用户需要使用 BZ/ $\overline{\text{BZ}}$ 或 PFD 功能，相应的 PA 口要设为 CMOS 输出。蜂鸣输出信号由 PA0 和 PA1 数据寄存器控制，如下表：

PA1 数据寄存器	PA0 数据寄存器	PA0/PA1 管脚状态
0	0	PA0=BZ, PA1= $\overline{\text{BZ}}$
1	0	PA0=BZ, PA1=0
X	1	PA0=0, PA1=0

注意：“X”表示未定义

PFD 输出信号由 PA3 数据寄存器和定时/计数器状态控制。PFD 输出信号的频率也依赖定时/计数器溢出周期。PFD 输出频率和控制信号如下表：

定时/计数器	定时/计数器预置值	PA3 数据寄存器	PA3 管脚状态	PFD 输出频率
关闭	X	0	U	X
关闭	X	1	0	X
打开	N	0	PFD	$f_{\text{INT}}/[2 \times (256-N)]$
打开	N	1	0	X

注意：“X”表示未定义

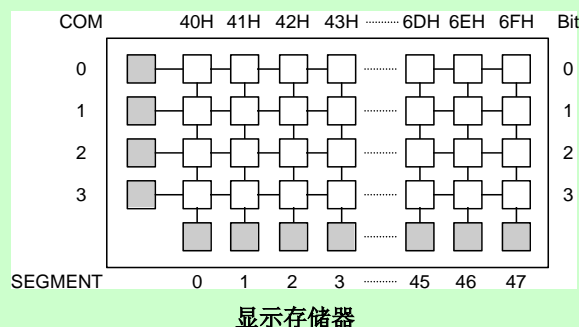
“U”表示未知

芯片复位后，这些输入口都会是高电平或浮空状态（可以由配置选项确定）。不建议采用“读-修改-写”的指令处理 IO 端口（会产生读错误）。使用“MOV”指令可以避免上述错误。PB 端口为 8 位带上拉电阻的施密特触发输入端口。

PA 的每个端口都有唤醒功能。PB0、PB1、PB2、PB3 和 PB4 分别与 $\overline{\text{INT0}}$ 、 $\overline{\text{INT1}}$ 、TMR0、TMR1 和 TMR2 引脚共用。

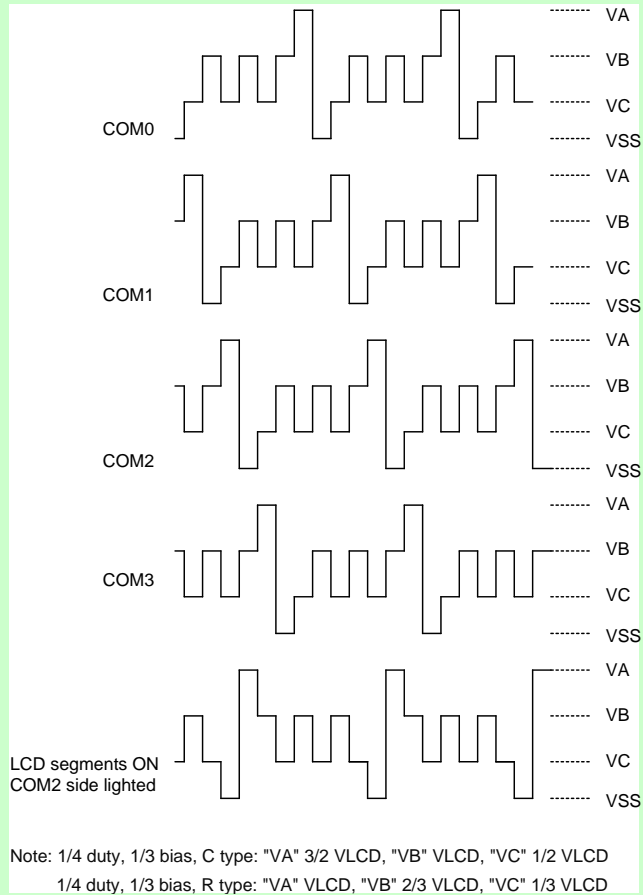
## LCD 显示存储器

系统为 LCD 显示提供一个嵌入式数据存储器区域。这个区域位于第一段数据存储器(bank 1)的 40H~6FH 单元。存储器段指针 BANK POINTER (BP; RAM 的 04H 单元) 是 RAM 和 LCD 显示存储器之间切换的开关。当 BP 设置“01”，通过 MP1 间接寻址写入 40H~6FH 的数据将会影响 LCD 的显示。当 BP 被清“0”，MP1 间接寻址写入的数据意味着访问一般意义上的数据存储器。LCD 显示存储器能被读出和写入，但是只能通过间接寻址模式，并使用 MP1 来进行。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，可以控制显示或不显示。下图为了显示存储器和 LCD 显示模块之间的映射关系。

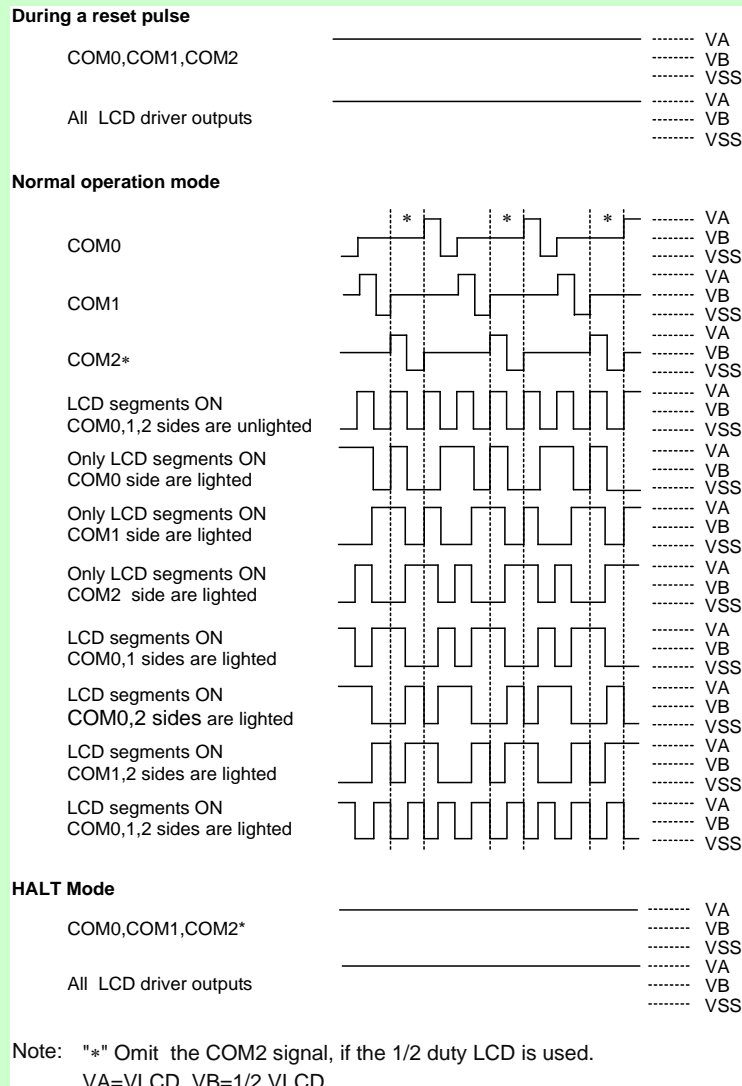


## LCD 驱动输出

LCD 驱动器的输出数目可以由配置选项确定为 48×2 或 48×3 或 47×4 (即 1/2 或 1/3 或 1/4 占空比)。LCD 驱动器偏压产生方式可以分为“R”型或“C”型，如果选为“R”型，不需要外接电容器；如果为“C”型，需要在 C1 和 C2 之间外接一个电容器。如果选择为 1/2 bias，则引脚 V2 到地需要接一个电容；如果选择为 1/3 bias 的话，则需要两个电容到地分别地接到引脚 V1、V2。建议选用 0.1μF 电容。



**LCD 驱动输出**



### LCD 驱动输出 (1/3 duty, 1/2 bias, R/C 型)

LCD 需要一个时钟源保证操作的正常进行，它由通用功能预分频器提供，其分频系数由配置选项控制。LCD 时钟频率必须选择接近 4kHz 的频率，LCD 时钟频率选项如下表所示：

LCD 时钟源	预分频系数
同看门狗时钟源 fs	$fs/2^2 \sim fs/2^8$

### LCD Segment 输出和逻辑输出

SEG40~SEG46 可由掩膜选择为 LCD Segment 输出或 PD 输出。若选择为 LCD Segment 输出， $V_{MAX}$  引脚连接将取决于 LCD 偏压和  $V_{LCD}$  电压。若选择为 PD 输出  $V_{MAX}$  必须连接至  $V_{DD}$ 。

LCD 偏压方式	R 型偏压		C 型偏压	
	1/2bias	1/3bias	1/2bias	1/3bias
$V_{MAX}$	如果 $V_{DD} > V_{LCD}$ ，则 $V_{MAX}$ 接到 $V_{DD}$ ，反之 $V_{MAX}$ 接到 $V_{LCD}$		如果 $V_{DD} > \frac{3}{2} V_{LCD}$ ，则 $V_{MAX}$ 接到 $V_{DD}$ ，反之 $V_{MAX}$ 接到 $V1$	

## 低电压复位/检测功能

系统具有低电压检测(LVD)和低电压复位(LVR)功能，这两个功能可以由配置选项选择是否打开。LVD 可以由配置选项打开或关闭。如果选择打开 LVD（低电压检测），用户可以通过 RTCC.3 来打开低电压检测，通过 RTCC.5 来读取低电压检测的状态。否则，低电压检测无效。

LVR 与外部复位信号有相同的功能，都会使芯片复位。在 HALT 状态下，LVR 是没有作用的。

RTCC 寄存器的定义如下表：

位	标号	读/写	功能
0~2	RT0~RT2	读/写	通过控制 8 选 1 多路器输入来选择实时钟预置寄存器的分频输出比例
3	LVDC	读/写	低电压检测打开/关闭 (1/0)
4	QOSC	读/写	32768Hz 晶振快速起振 0/1: 快速/慢速
5	LVDO	读	低电压检测输出 (1/0), 1: 检测到低电压
6~7	—	—	未定义, 读出为 0

**RTCC(09H) 寄存器**

## 异步串行口 — UART

HT49RU80/HT49CU80 具有一个全双工的异步串行通信口,可以很方便的与其它具有串行口的芯片通讯。UART 具有许多功能特性,发送或接收串行数据时,将数据组成一个 8 位或 9 位的数据块,连同数据特征位一并传输。当数据过多或数据特征位不正确时,UART 可以检测出错误。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 具有以下功能:

- 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、噪声和超速检测
- 支持中断和地址检测 (最后一位=1)
- 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
  - 接收完成
  - 超速错误
  - 地址匹配

### • UART 外部引脚

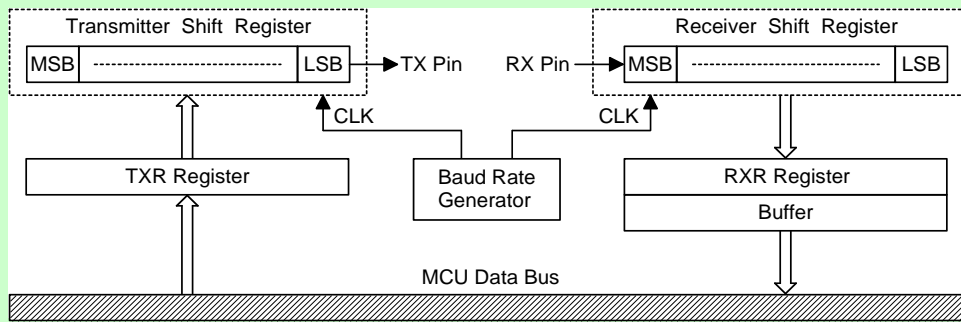
UART 是通过两个外部引脚 TX 和 RX 与外部芯片通讯。TX 引脚是 UART 的发送引脚。当 UCR2 寄存器的 TXEN 位清零,UART 发送功能被禁止,则 TX 引脚可作为普通 IO 口使用。RX 引脚是 UART 的接收引脚。同样的,当 UCR2 寄存器的 RXEN 位清零,UART 接收功能被禁止,则 RX 引脚可作为普通 IO 口使用。若 UARTEN、TXEN 和 RXEN 置位,这些 IO 口将自动转换成相应 TX 输出和 RX 输入,并且 RX 脚的上拉电阻将无效。

### • 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。TXR 寄存器被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,所以发送移位寄存器不可直接操作。

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。RXR 寄存器被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,所以接收移位寄存器不可直接操作。

需要注意的是,上述发送寄存器 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 数据发送接收图**

- **UART 状态控制寄存器**

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，发送和接收数据则通过寄存器 TXR/RXR。

- **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：

- **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 是数据发送完成标志位。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。

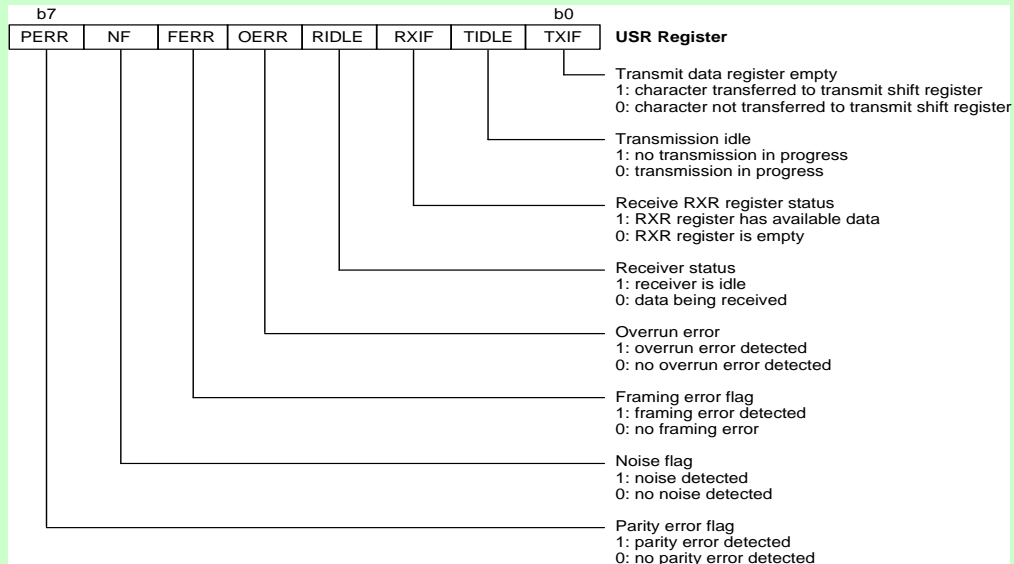
- **NF**

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。



- 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

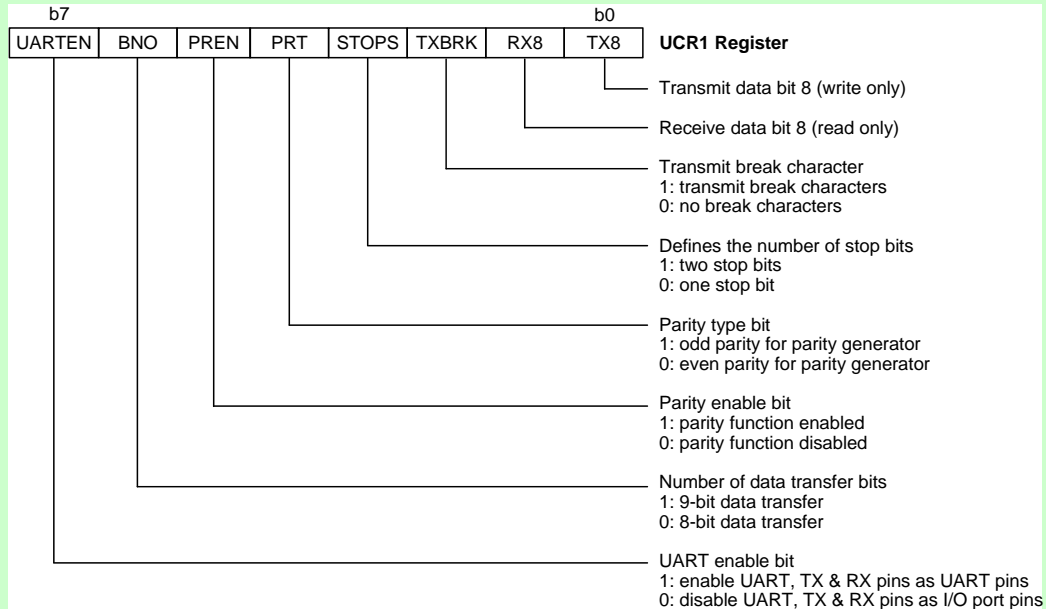
此位为奇偶校验使能位。PREN=1，使能奇偶校验；PREN=0，除能奇偶校验。

- 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。

• **TIEE**

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



• 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。

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

$$\text{转换后的公式 } N = \frac{f_{SYS}}{(BR \times 64)} - 1$$

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

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 K/BPS	BRGH=0											
	f <sub>SYS</sub> =8MHz			f <sub>SYS</sub> =7.159MHz			f <sub>SYS</sub> =4MHz			f <sub>SYS</sub> =3.579545MHz		
	BRG	Kbaud	Error	BRG	Kbaud	Error	BRG	Kbaud	Error	BRG	Kbaud	Error
0.3	-	-	-	-	-	-	207	0.300	0.00	185	0.300	0.00
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	18.64	-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 K/BPS	BRGH=1											
	f <sub>SYS</sub> =8MHz			f <sub>SYS</sub> =7.159MHz			f <sub>SYS</sub> =4MHz			f <sub>SYS</sub> =3.579545MHz		
	BRG	Kbaud	Error	BRG	Kbaud	Error	BRG	Kbaud	Error	BRG	Kbaud	Error
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，这种方法通常被称为 NRZ 法。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 PC6 和 PC7 复用，UARTEN 的一个基本功能就是控制这两个引脚。若 UARTEN、TXEN 和 RXEN 都为高，则 PC6 和 PC7 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，使其可作为普通的输入输出端口使用。当 UART 被除能清除缓冲器，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。

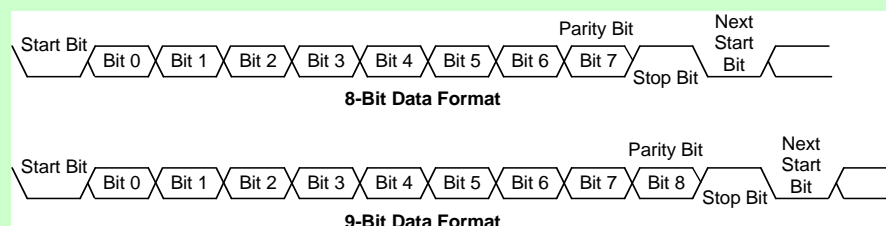
### · 数据位、停止位位数以及奇偶校验的选择

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 决定数据传输是 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
<b>8 位数据位</b>				
1	8	0	0	1
1	7	0	1	1
1	7	1 <sup>1</sup>	0	1
<b>9 位数据位</b>				
1	9	0	0	1
1	8	0	1	1
1	8	1 <sup>1</sup>	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区，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、噪声干扰错误、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。

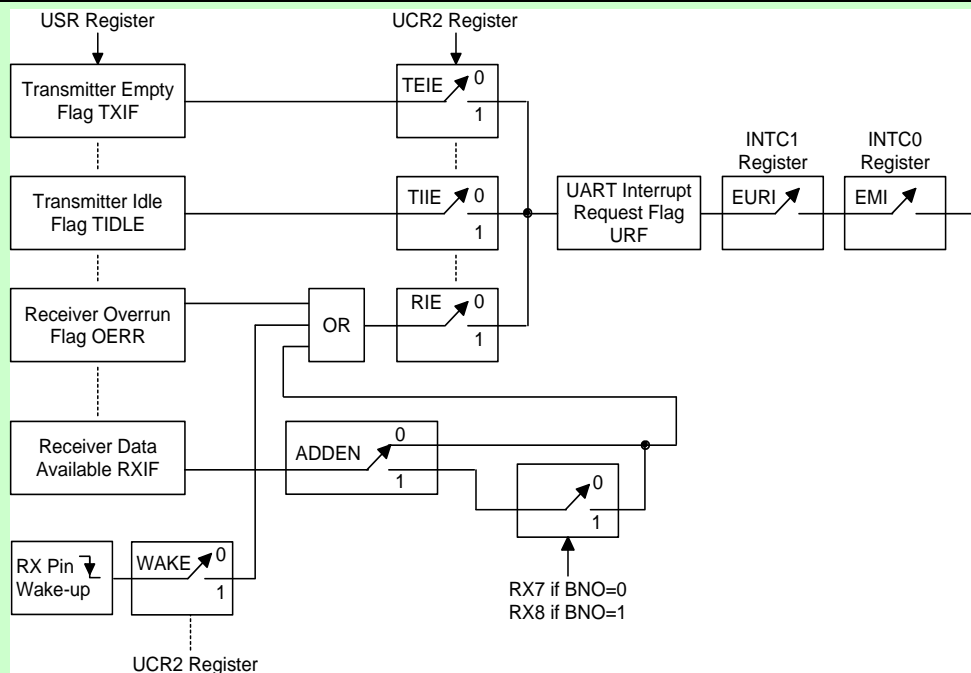
- 奇偶校验错误——**PERR** 标志

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 拥有单独的内部中断和独立的中断变量。发送寄存器为空、发送器空闲、接收器数据有效、过速和地址检测和 **RX** 引脚唤醒都会产生中断。若 UART 中断允许且堆栈未滿，程序将会跳转到相应的中断向量执行中断服务程序，而后再返回主程序。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产生中断。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个别的 UART 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** 中的 **WAKE** 和 **RIE** 位被置位，**RX** 引脚上有下降沿可以唤醒单片机。应注意，**RX** 唤醒中断发生时，系统必须延时 1024 个系统时钟才能正常工作。



UART 中断框图

• 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，同时必须将奇偶检验使能位清零，除能奇偶校验。

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

ADDEN 位功能

• 暂停模式下的 UART 功能

当 MCU 进入暂停模式，UART 将停止工作。当芯片进入暂停模式，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。当单片机进入暂停模式，USR、UCR1 和 UCR2、接收/发送寄存器、BRG 寄存器都不会受到影响。

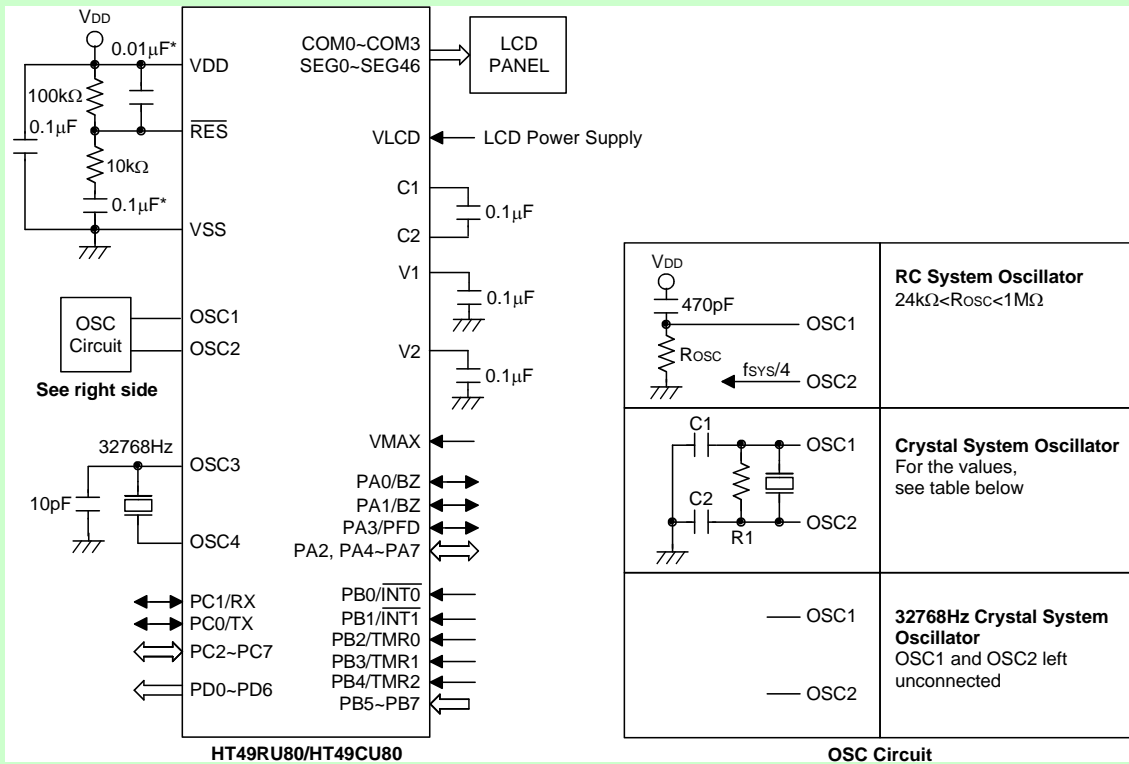
UART 功能中包括了 RX 引脚的唤醒功能，由 UCR2 寄存器中 WAKE 位控制。进入暂停模式前，若该标志位与 UART 允许位 UARTEN、接收器允许位 RXEN 和接收器中断位 RIE 都被置位，则 RX 引脚的下降沿可唤醒单片机。唤醒后系统需延时 1024 个系统时钟才能正常工作，在此期间，RX 引脚上的任何数据将被忽略。

若要唤醒并产生 UART 中断，全局中断允许位 EMI 和 UART 中断允许 EURI 也必须置位；若这两标志位没有被置位，那么，单片机将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。

## 配置选项

下面的表格列出了配置选项，所有选项必须正确定义以确保系统正确的功能。摸膜选项涉及到 MCU 的具体功能，在烧录程序的同时烧入芯片。它在开发过程中，通过软件 HT-IDE 选择。由编程器烧入芯片，并且不能通过应用程序修改。

配置选项
<b>输入/输出选项</b>
PA0~PA7 唤醒功能使能/禁止
PA0~PA3 CMOS/NMOS 选择
PA0~PA3 上拉电阻使能/禁止
PC0~PC3 CMOS/NMOS 选择
PC4~PC7 CMOS/NMOS 选择
PC0~PC3 上拉电阻使能/禁止
PC4~PC7 上拉电阻使能/禁止
<b>看门狗选项</b>
WDT 功能：打开或关闭
CLRWDT 指令：一条或两条指令
<b>振荡器选项</b>
OSC 类型选项：RC 或晶体振荡
f <sub>SYS</sub> 时钟源：OSC 或 RTC
f <sub>S</sub> 内部时钟源：f <sub>SYS</sub> /4，RTC 振荡或 WDT 振荡
<b>PFD 选项</b>
PFD 输出：使能或禁止
PFD 时钟源：定时/计数器 0 或定时/计数器 1
<b>定时器选项</b>
定时/计数器 0 时钟源：f <sub>SYS</sub> /4 或 f <sub>SYS</sub>
定时/计数器 1 时钟源：定时/计数器 0 溢出频率、时基溢出频率或 f <sub>SYS</sub>
<b>时基选项</b>
时基频率：f <sub>S</sub> /2 <sup>12</sup> ~f <sub>S</sub> /2 <sup>15</sup>
<b>蜂鸣器选项</b>
蜂鸣器输出：使能或禁止
蜂鸣器频率：f <sub>S</sub> /2 <sup>2</sup> ~f <sub>S</sub> /2 <sup>9</sup>
<b>LVD/LVR 选项</b>
LVR 功能：使能或禁止
LVD 功能：使能或禁止
<b>LCD 选项</b>
LCD 时钟：f <sub>S</sub> /2 <sup>2</sup> ~f <sub>S</sub> /2 <sup>8</sup>
LCD 占空比：1/2、1/3 或 1/4
LCD 偏压：1/2 或 1/3
LCD 偏压方式：R 型或 C 型
HALT 模式下 LCD 打开/关闭：使能或禁止
SEG40~SEG46 为 LCD Segment 输出或 PD0~OD6 输出
HALT 模式下系统时钟：打开或关闭

**应用电路**


下表所示为根据不同的晶振值选择 R1、C1 和 C2 值（仅供参考）。

晶体或共振器	C1、C2	R1
4MHz 晶体	0pF	12kΩ
4MHz 共振器	10pF	12kΩ
3.58MHz 晶体	0pF	12kΩ
3.58MHz 共振器	25pF	12kΩ
2MHz 晶体和共振器	25pF	12kΩ
1MHz 晶体	35pF	14kΩ
480kHz 共振器	100pF	14kΩ
455kHz 共振器	200pF	12kΩ
429kHz 共振器	200pF	12kΩ
400kHz 共振器	300pF	12kΩ

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。

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

## 指令集摘要

助记符	说明	指令周期	影响标志位
<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>(1)</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>(1)</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>(1)</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
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>(1)</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>(1)</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>(1)</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>(1)</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>(1)</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>(1)</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>(1)</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>(1)</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>(1)</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>(1)</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>(1)</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>(1)</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>(1)</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>(1)</sup>	无
SET [m].i	置位数据存储器的位	1 <sup>(1)</sup>	无

助记符	说明	指令周期	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>(2)</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>(2)</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>(2)</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>(2)</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>(3)</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>(3)</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>(2)</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>(2)</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>(1)</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>(1)</sup>	无
<b>其它指令</b>			
NOP	空指令	1	无
CLR [m]	清除数据存储器	1 <sup>(1)</sup>	无
SET [m]	置位数据存储器	1 <sup>(1)</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 <sup>(4)</sup> ,PDF <sup>(4)</sup>
CLR WDT2	预清除看门狗定时器	1	TO <sup>(4)</sup> ,PDF <sup>(4)</sup>
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>(1)</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： x: 立即数

m: 数据存储器地址

A: 累加器

i: 第 0~7 位

addr: 程序存储器地址

√: 影响标志位

—: 不影响标志位

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

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

(3): (1)和(2)

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

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

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
0	0	—	—	—	—

**CLR WDT1** 预清除看门狗定时器

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

运算过程： $WDT \leftarrow 00H^*$   
 $PDF \& TO \leftarrow 0^*$

影响标志位

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

**CLR WDT2** 预清除看门狗定时器

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

运算过程： $WDT \leftarrow 00H^*$   
 $PDF \& TO \leftarrow 0^*$

影响标志位

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

**CPL [m]** 对数据存储器取反，结果放入数据存储器

说明：本指令是将数据存储器内保存的数值取反。

运算过程： $[m] \leftarrow \overline{[m]}$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**CPLA [m]** 对数据存储器取反，结果放入累加器

说明：本指令是将数据存储器内保存的值取反后，结果存放在累加器中。

运算过程： $ACC \leftarrow \overline{[m]}$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

影响标志位

TO	PDF	OV	Z	AC	C
0	1	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**NOP**                    空指令  
 说明:                    本指令不作任何运算, 而只将程序计数器加一。  
 运算过程:                 $\text{Program Counter} \leftarrow \text{Program Counter} + 1$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

**RET**                    从子程序返回  
 说明:                    本指令是将堆栈寄存器中的程序计数器值送回程序计数器。  
 运算过程:                 $\text{Program Counter} \leftarrow \text{Stack}$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**RETI**            从中断返回

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

运算过程：        Program Counter  $\leftarrow$  Stack  
                       EMI  $\leftarrow$  1

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**RL**            [m]        数据存储器左移一位，结果放入数据存储器

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

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

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**RLA**          [m]        数据存储器左移一位，结果放入累加器

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

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

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**RLC**          [m]        带进位将数据存储器左移一位，结果放入数据存储器

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

运算过程：        [m].(i+1)  $\leftarrow$  [m].i; (i=0~6)  
                       [m].0  $\leftarrow$  C  
                       C  $\leftarrow$  [m].7

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

**RLCA**        [m]        带进位将数据存储器左移一位，结果放入累加器

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

运算过程：        ACC.(i+1)  $\leftarrow$  [m].i; (i=0~6)  
                       ACC.0  $\leftarrow$  C  
                       C  $\leftarrow$  [m].7

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

**RR** [m] 数据存储器右移一位，结果放入数据存储器  
 说明：本指令是将存储器内的数值循环右移，第 0 位移到第 7 位，结果送回数据存储器。  
 运算过程： $[m].7 \leftarrow [m].0$ ,  $[m].i \leftarrow [m].(i+1)$ ; (i=0~6)  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**RRA** [m] 数据存储器右移一位，结果放入累加器  
 说明：本指令是将数据存储器内的数值循环右移，第 0 位移到第 7 位，结果送回累加器，而数据存储器内的数值不变。

运算过程： $ACC.7 \leftarrow [m].0$ ,  $ACC.i \leftarrow [m].(i+1)$ ; (i=0~6)  
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**RRC** [m] 带进位将数据存储器右移一位，结果放入数据存储器  
 说明：本指令是将存储器内的数值加进位标志循环右移，第 0 位取代进位标志，进位标志移到第 7 位，结果送回存储器。

运算过程： $[m].i \leftarrow [m].(i+1)$ ; (i=0~6)  
 $[m].7 \leftarrow C$   
 $C \leftarrow [m].0$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

**RRCA** [m] 带进位将数据存储器右移一位，结果放入累加器  
 说明：本指令是将数据存储器内的数值加进位标志循环右移，第 0 位取代进位标志，进位标志移到第 7 位，结果送回累加器，数据存储器内的数值不变。

运算过程： $ACC.i \leftarrow [m].(i+1)$ ; (i=0~6)  
 $ACC.7 \leftarrow C$   
 $C \leftarrow [m].0$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

**SBC** A,[m] 累加器与数据存储器、进位标志相减，结果放入累加器  
 说明：本指令是把累加器值减去数据存储器值以及进位标志的取反，结果放到累加器。

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

影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**SIZ** [m] 数据存储器加 1，如果结果为“0”，则跳过下一条指令

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

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

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

运算过程：如果  $[m].i \neq 0$ ，跳过下一行指令。

影响标志位

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

**SUB** A, [m] 累加器与数据存储器相减，结果放入累加器

说明：本指令是把累加器值、数据存储器值相减，结果放到累加器。

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

影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

**SUB** A, x 累加器与立即数相减，结果放入累加器

说明：本指令是把累加器值、立即数相减，结果放到累加器。

运算过程： $ACC \leftarrow ACC + \overline{x} + 1$

影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

**SUBM A, [m]** 累加器与数据存储器相减，结果放入数据存储器  
 说明： 本指令是把累加器值、存储器值相减，结果放到存储器。  
 运算过程： $[m] \leftarrow ACC + [\overline{m}] + 1$   
 影响标志位

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

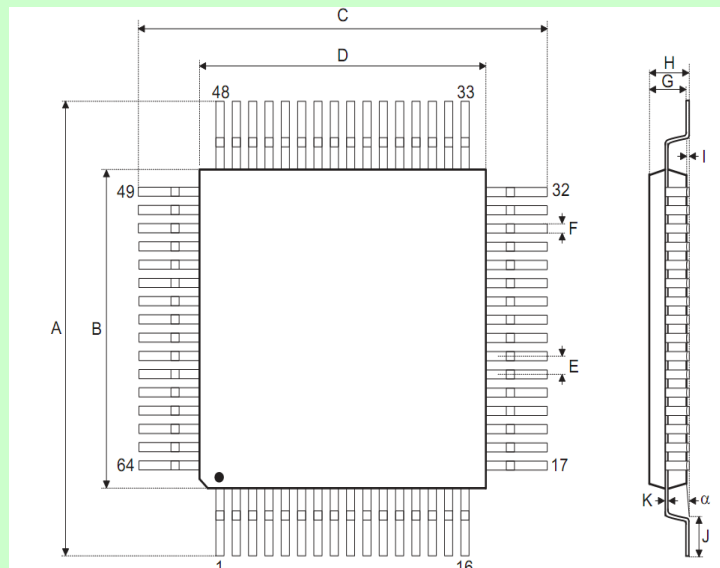
TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

## 封装尺寸

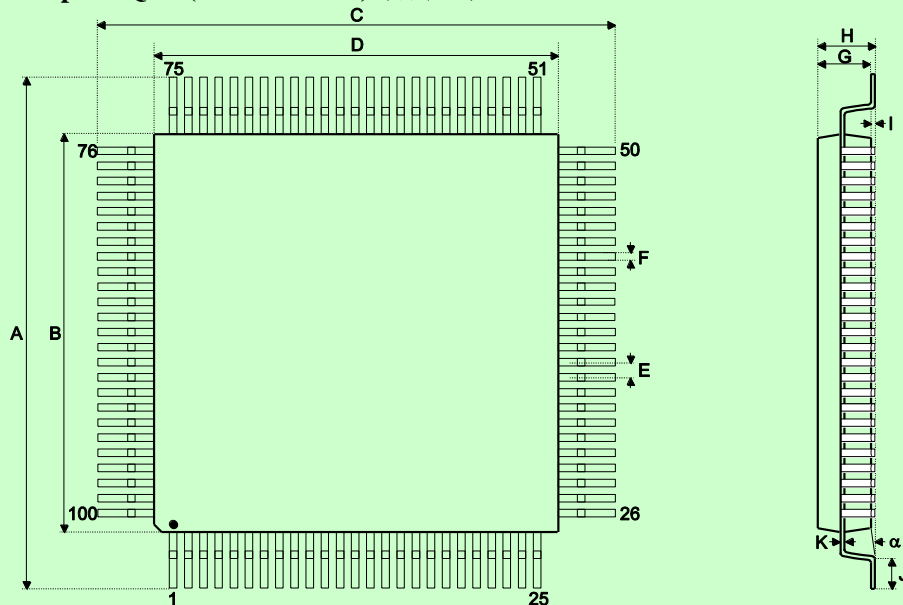
### 64-pin LQFP (7mm×7mm) outline dimensions



符号	尺寸 (单位: inch)		
	最小	一般	最大
A	0.350	—	0.358
B	0.272	—	0.280
C	0.350	—	0.358
D	0.272	—	0.280
E	—	0.016	—
F	0.005	—	0.009
G	0.053	—	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	—	0.030
K	0.004	—	0.008
$\alpha$	0°	—	7°

符号	尺寸 (单位: mm)		
	最小	一般	最大
A	8.90	—	9.10
B	6.90	—	7.10
C	8.90	—	9.10
D	6.90	—	7.10
E	—	0.40	—
F	0.13	—	0.23
G	1.35	—	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	—	0.75
K	0.09	—	0.20
$\alpha$	0°	—	7°

100-pin LQFP (14mm×14mm)外形尺寸



符号	尺寸(单位: inch)		
	最小	正常	最大
A	0.626	—	0.634
B	0.547	—	0.555
C	0.626	—	0.634
D	0.547	—	0.555
E	—	0.020	—
F	—	0.008	—
G	0.053	—	0.057
H	—	—	0.063
I	—	0.004	—
J	0.018	—	0.030
K	0.004	—	0.008
$\alpha$	0°	—	7°

符号	尺寸(单位: mm)		
	最小	典型	最大
A	15.90	—	16.10
B	13.90	—	14.10
C	15.90	—	16.10
D	13.90	—	14.10
E	—	0.50	—
F	—	0.20	—
G	1.35	—	1.45
H	—	—	1.60
I	—	0.10	—
J	0.45	—	0.75
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](http://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（深圳业务处）**

深圳市南山区科技园科技中三路与高新中二道交汇处生产力大楼A单元五楼 518057

电话: 86-755-8616-9908, 86-755-8616-9308

传真: 86-755-8616-9722

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

46729 Fremont Blvd., Fremont, CA 94538, USA

电话: 1-510-252-9880

传真: 1-510-252-9885

网站: [www.holtek.com](http://www.holtek.com)

Copyright © 2011 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>