

盛群知识产权政策

专利权

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

商标权

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

著作权

Copyright © 2008 by HOLTEK SEMICONDUCTOR INC.

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

技术相关信息

- [工具信息](#)
- [FAQs](#)
- [应用范例](#)
 - [HA0003S HT48 & HT46 MCU 与 HT93LC46 的通信](#)
 - [HA0016S HT48读写 HT24 EEPROM](#)
 - [HA0018S HT48控制 HT1621 LCD](#)
 - [HA0049S HT48读写控制 HT1380](#)
 - [HA0075S MCU 复位与振荡电路应用](#)

特性

- 工作电压：
 - fsys = 4MHz : 2.2V – 5.5V
 - fsys = 8MHz : 3.3V – 5.5V
 - fsys = 12MHz : 4.5V – 5.5V
- 7 个双向输入/输出和一个输入口
- 与输入/输出引脚复用的外部中断输入
- 1 个 8 位可编程定时/计数器，具有溢出中断及 7 级预分频器
- 外部晶体振荡系统
- 内部集成 RC 振荡器可用于 3 种频率：4MHz, 8MHz 或 12MHz
- 看门狗定时器
- 支持蜂鸣器驱动
- HALT 和唤醒功能可降低功耗
- 在 V_{DD}=5V，系统时钟为 12MHz 时，指令周期为 0.33 μs
- 4 层硬件堆栈
- 位操作指令
- 查表指令
- 63 条功能强大的指令
- 所有指令在 1 个或 2 个周期内完成
- 低电压复位功能
- 10-pin MSOP 封装

概述

HT48R01A 是一款八位高性能精简指令集单片机，专为广泛应用的产品设计。

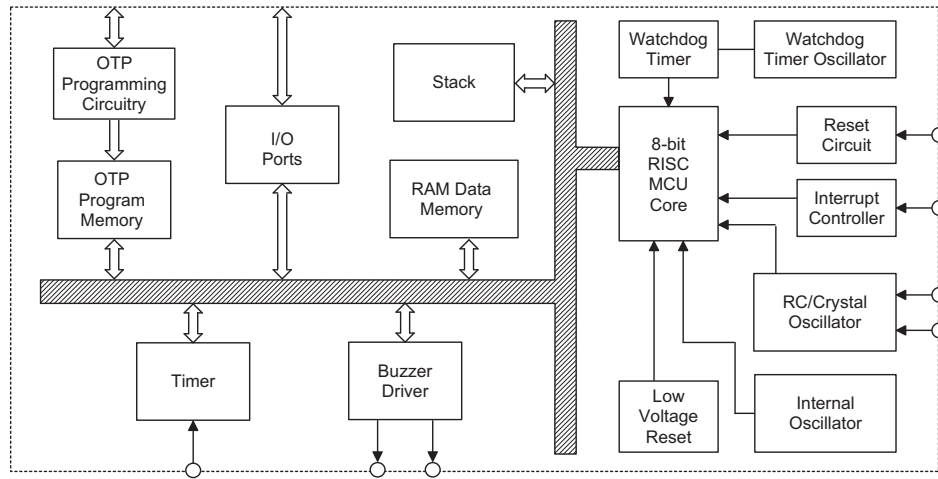
拥有低功耗、I/O 口灵活、定时功能、振荡选择、省电和唤醒功能、看门狗定时器、蜂鸣器驱动、以等一系列强大功能仍保持较低的成本，芯片集成 3 种频率可选的振荡系统，以及极小的 10-pin MSOP 封装，为开辟各种新的产品应用提供了可能，例如工业控制、消费类产品、家电子系统控制器等。

设备型号

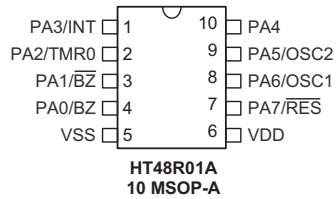
HT48R01A 分别具有三种不同的内部 RC 振荡频率 4MHz、8MHz 和 12MHz 分别用后缀 1、2 和三表示，如下表所示：

型号	设备型号	内部 RC 振荡频率
HT48R01A	HT48R01A1	4MHz
	HT48R01A2	8MHz
	HT48R01A3	12MHz

方框图



引脚图



引脚说明

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

注意: 除了 PA7 外的所有引脚的上拉电阻通过寄存器选项控制。

极限参数

电源供应电压 $V_{SS} - 0.3V$ 至 $V_{SS} + 6.0V$	储存温度 $-50^{\circ}C$ 至 $125^{\circ}C$
端口输入电压 $V_{SS} - 0.3V$ 至 $V_{DD} + 0.3V$	工作温度 $-40^{\circ}C$ 至 $85^{\circ}C$
IOL 总电流 150mA	IOH 总电流 -100mA
总消耗电流 500mW		

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

直流特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{DD}	工作电压	—	f _{sys} =4MHz	2.2	—	5.5	V
		—	f _{sys} =8MHz	3.3	—	5.5	V
		—	f _{sys} =12MHz	4.5	—	5.5	V
I _{DD1}	工作电流 (晶体振荡, RC 振荡)	3V	无负载	—	1	2	mA
		5V	f _{sys} =4MHz	—	2.5	5	
I _{DD2}	工作电流 (晶体振荡, RC 振荡)	3V	无负载	—	2	4	mA
		5V	f _{sys} =8MHz	—	4	8	
I _{DD3}	工作电流 (晶体振荡, RC 振荡)	5V	无负载 f _{sys} =12MHz	—	6	12	mA
I _{DD4}	工作电流 (内部 RC+RTC 振荡, 正常模式)	3V	无负载	—	1	2	mA
		5V	f _{sys} =4MHz	—	2.5	5	mA
I _{DD5}	工作电流 (内部 RC+RTC 振荡, 正常模式)	3V	无负载	—	2	4	mA
		5V	f _{sys} =8MHz	—	4	8	mA
I _{DD6}	工作电流 (内部 RC+RTC 振荡, 正常模式)	5V	无负载 f _{sys} =12MHz	—	6	12	mA
I _{DD7}	工作电流 (内部 RC+RTC 振荡, 低速模式)	3V	无负载	—	20	30	μA
		5V	f _{sys} =32768Hz	—	40	60	μA
I _{STB1}	静态电流 (看门狗打开, RTC 关闭)	3V	无负载	—	—	5	μA
		5V	暂停模式	—	—	10	
I _{STB2}	静态电流 (看门狗关闭, RTC 关闭)	3V	无负载	—	—	1	μA
		5V	暂停模式	—	—	2	
I _{STB3}	静态电流 (看门狗关闭, RTC 打开)	3V	无负载	—	—	5	μA
		5V	暂停模式 QOSC=1	—	—	10	
V _{IL1}	PA0~PA7,TMR0, INT 低电平输入电压	—	—	0	—	0.3V _{DD}	V
V _{IH1}	PA0~PA6,TMR0, INT 高电平输入电压	—	—	0.7V _{DD}	—	V _{DD}	V
V _{IL2}	低电平输入电压 (RES)	—	—	0	—	0.4V _{DD}	V
V _{IH2}	高电平输入电压 (RES)	—	—	0.9V _{DD}	—	V _{DD}	V
V _{LVR1}	低电压复位 1	—	掩模选项: 4.2V	3.98	4.2	4.42	V
V _{LVR2}	低电压复位 2	—	掩模选项: 3.15V	2.98	3.15	3.32	V
V _{LVR3}	低电压复位 3	—	掩模选项: 2.1V	1.98	2.1	2.22	V

IOL	输入/输出灌电流	3V	VOL=0.1VDD	4	8	—	mA
		5V		10	20	—	
IOH	输入/输出源电流	3V	VOH=0.9VDD	-2	-4	—	mA
		5V		-5	-10	—	
RPH	上拉电阻	3V	—	20	60	100	kΩ
		5V		10	30	50	

交流特性

Ta=25℃

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

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

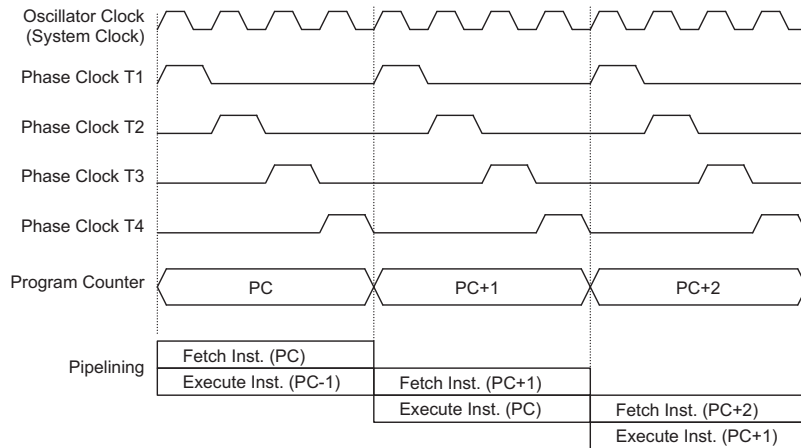
系统结构

内部系统结构是盛群单片机具有良好运行性能的主要因素。由于采用 RISC 结构，此系列单片机具有高运算速度和高性能的特性。通过流水线的方式，指令的取得和执行同时进行，此举使得除了分支和调用指令外，其它指令都能在一个指令周期内完成。8 位的 ALU 参与指令集中所有的运算，它可完成算术运算、逻辑运算、移位、加、减和分支等功能，而内部的数据路径则以通过累加器或 ALU 的方式加以简化。有些寄存器在数据存储器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供最大可靠度和灵活性控制系统时，仅需要少数的外部器件。

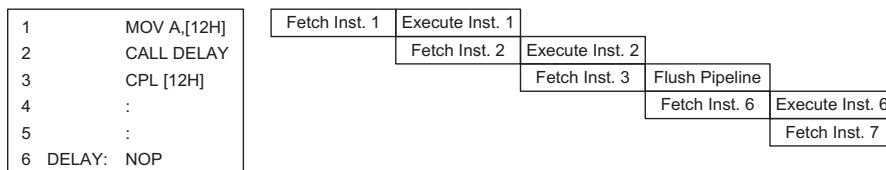
时序和流水线结构

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

如果指令牵涉到分支，例如跳转或调用等指令，则需要两个指令周期才能完成指令执行。需要一个额外周期的原因是程序先用一个周期取出实际要跳转或调用的地址，再用另一个周期去实际执行分支动作，因此用户必须特别考虑额外周期的问题，尤其是在执行时间要求较严格的时候。



系统时序和流水线



指令捕捉

程序计数器

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

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

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

程序计数器低字节可以由程序直接进行读取。操作 PCL 的可能导致程序分支，所以额外的周期需要预先取得。有关 PCL 寄存器更多的信息可在特殊功能寄存器部份查找。

模式	程序计数器									
	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
初始化复位	0	0	0	0	0	0	0	0	0	0
外部中断	0	0	0	0	0	0	0	1	0	0
定时/计数器溢出	0	0	0	0	0	0	1	0	0	0
条件跳跃	PC+2									
装载 PCL	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
跳转、子程序调用	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
从子程序返回	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

程序计数器

注： PC9~PC8：当前程序计数器位

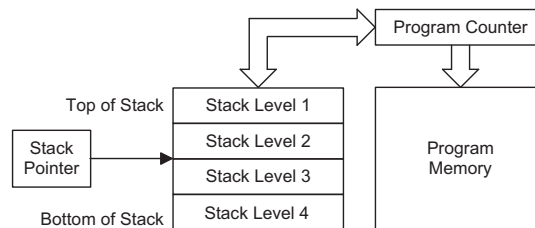
@7~@0： PCL 位

#9~#0： 指令码位

S9~S0： 堆栈寄存器位

堆栈

堆栈是存储器中一个特殊的部分，它只用来储存程序计数器中的内容。根据选择的单片机，堆栈可介于 2 或 4 层之间，它们既不是数据部份也不是程序空间部份，且既不可读取也不可写入。当前层由堆栈指针（Stack Pointer, SP）加以指示，同样也是不可读写的。在子程序调用或中断响应服务时，程序计数器的内容被压入堆栈。当子程序或中断服务程序结束时，返回指令（RET 或 RETI）使程序计数器从堆栈中返回它之前的值。当芯片复位之后，SP 将指向堆栈的顶部。



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

算术逻辑单元——ALU

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

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

程序存储器

程序存储器用来存放用户代码或储存程序。单片机提供一次可编程存储器（OTP），使用者可编写他们的应用代码到单片机中。使用适当的编程工具，OTP 单片机可以提供使用者灵活的方式来自由开发他们的应用，对于除错或需要经常升级与程序修改的产品是很有帮助的。对于中小型量产，OTP 亦为极佳的选择。

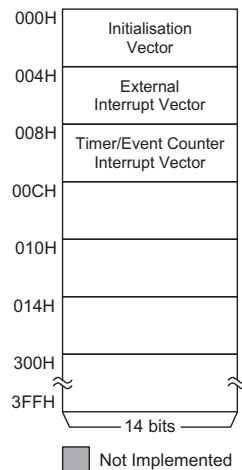
结构

14 位的程序存储器的容量是 1K，程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口，数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。

特殊向量

程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。

- 地址 000H
此地址保留给程序初始化之用。当系统复位时，程序会从 000H 地址开始执行。
- 地址 004H
该地址为外部中断服务程序保留。当 INT 引脚有触发信号输入，如果中断允许且堆栈未滿，则程序会跳转到 004H 地址开始执行服务程序。
- 地址 008H
此地址保留给定时/计数器 0 中断服务使用。当定时/计数器 0 溢出，如果中断允许且堆栈又未滿，则程序会从 008H 地址开始执行中断服务程序。



程序存储器结构

查表

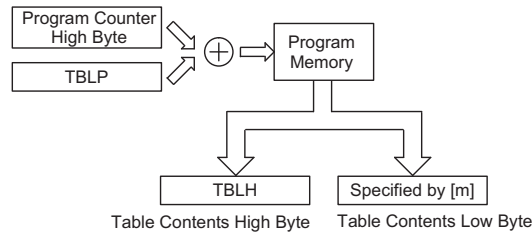
程序存储器中的任何地址都可以定义成一个表格，以便储存固定的数据。使用表格时，表格指针必须先设定，其方式是将表格的低字节地址放在表格指针寄存器 TBLP 中。这个寄存器定义表格较低的 8 位地址。在设定完表格指针后，表格数据可以使用“TABRDC [m]”或“TABRDL [m]”指令从当前的程序所在的存储器页或存储器最后一页中来查表读取。当这些指令执行时，程序存储器中表格数据低字节，将被传送到使用者所指定的数据存储器，程序存储器中表格数据的高字节，则被传送到 TBLH 特殊寄存器，而高字节中未使用的位将被读取为“0”。

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

表格区

注意： PC9 ~PC8: 当前 PC 位
 @7 ~ @0 : 表格指针位

下图是查表寻址/数据流程图：



查表程序范例

以下范例说明了如何定义表格指针、如何查表。这个例子使用的表格数据用 **ORG** 伪指令储存在存储器的最后一页。在此 **ORG** 伪指令中的值为 **300H**，即 **1K** 的程序存储器 **HT48R01A** 单片机最后一页存储器的开始的地址，而表格的初始值为 **06H**。这可保证从数据表格读取的第一笔数据位于程序存储器地址 **306H**，即最后一页开始地址后六个地址。值得注意的是假如“**TABRDC [m]**”指令被使用，则表格指针指向当前页。表格数据低字节被送往指定的寄存器，而表格数据高字节将自动被送往 **TBLH** 寄存器，在这个例子中，表格数据的高字节等于零。

```

Tempreg1 db ?           ; temporary register #1
tempreg2 db ?         ; temporary register #2
:
:
mov a,06h             ; initialize table pointer — note that this address
                    ; is referenced
mov tblp,a           ; to the last page or present page
:
:
tabrdl tempreg1      ; transfers value in table referenced by table pointer
                    ; to tempreg1
                    ; data at prog. Memory address – 306H– transferred to
                    ; tempreg1 and TBLH
dec tblp             ; reduce value of table pointer by one
tabrdl tempreg2      ; transfers value in table referenced by table pointer
                    ; to tempreg2
                    ; data at prog.memory address – 305H– transferred to
                    ; tempreg2 and TBLH
                    ; in this example the data – 1AH– is transferred to
                    ; tempreg1 and data – 0FH– to register tempreg2
                    ; the value – 00H– will be transferred to the high byte
                    ; register TBLH
:
:
org 300h             ; sets initial address of last page
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:

```

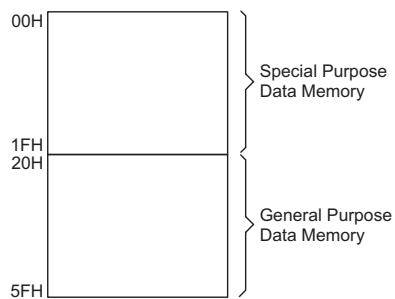
因为 **TBLH** 寄存器是只读寄存器，不能重新存储，若主程序和中断服务程序都使用表格读取指令，应该注意它的保护。若使用表格读取指令，中断服务程序可能会改变 **TBLH** 的值，若随后在主程序中再次使用这个值，则会发生错误。因此建议避免同时使用表格读取指令。然而在某些情况下，如果同时使用表格读取指令是不可避免的，则在执行任何主程序的表格读取前，中断应该先禁止，另外需要注意的是所有与表格相关的指令，都需要两个指令周期去完成操作。

数据存储器

数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。分为二个部份，第一部份是特殊功能寄存器，这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二部份数据存储器是作一般用途使用，都可在程序控制下进行读取和写入。

结构

特殊和通用数据存储器，位于连续的地址。全部 RAM 为 8 位宽度，但存储器长度因所选择的单片机而不同。所有单片机的数据存储器的起始地址都是 00H。常见的寄存器，如 ACC 和 PCL 等，全都具有相同的数据存储器地址。



数据存储器结构

注：除部分特殊位，大部分数据存储区可以通过“SET [m].i”和“CLR [m].i”直接寻址。数据存储器也可以通过指针间接寻址。

通用数据存储器

所有的单片机程序需要一个读/写的存储区，让临时数据可以被储存和再使用。该 RAM 区域就是通用数据存储器。这个数据存储区可让使用者进行读取和写入的操作。使用“SET [m].i”和“CLR [m].i”指令可对个别的位做置位或复位的操作，方便用户在数据存储器内进行位操作。

特殊数据存储器

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

特殊功能寄存器

为了确保单片机能成功的工作，数据存储器中设置了一些内部寄存器。这些寄存器确保内部功能（如定时器和中断等）和外部功能（如 I/O 数据和 AD 转换控制）的正确工作。在数据存储器中，这些寄存器以 00H 作为起始地址。在特殊功能寄存器和通用数据存储器的起始地址之间，有一些未定义的数据存储器，被保留用来做未来扩充，若从这些地址读取数据将返回 00H 值。

间接寻址寄存器 — IAR0, IAR1

IAR0 和 IAR1 寄存器位于数据存储器区，并没有实际的物理地址。间接寻址的方法准许使用间接寻址指针做数据操作，以取代定义实际存储器地址的直接存储器寻址方法。在间接寻址寄存器 IAR0 和 IAR1 上的任何动作，将对间接寻址指针 (MP 或 MP1) 所指定的存储器地址产生对应的读/写操作。直接读取 IAR 寄存器将返回 00H 的结果，而直接写入此寄存器则不做任何操作。

间接寻址指针 — MP0, MP1

对于该单片机，系统提供连个间接寻址指针 MP0 和 MP1。由于这些指针在数据存储器中能象普通的寄存器一般被写入和操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由间接寻址指针所指定的地址。寻址全部的存储空间 MP 的第 7 位是不需要的，需要注意的是读取 MP 的第 7 位返回值是 1。

下面的例子说明如何清除一个具有 4RAM 地址区块，它们已事先定义为地址 addr1 到 addr4:

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block db ?
code .section at 0 'code'
org 00h

start:
mov a,04h ; setup size of block
mov block,a
mov a,offset adres1 ; Accumulator loaded with first RAM address
mov mp0,a ; setup memory pointer with first RAM address

loop:
clr IAR0 ; clear the data at address defined by MP0
inc mp0 ; increment memory pointer
sdz block ; check if last memory location has been cleared
jmp loop

continue:
```

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。

累加器 — ACC

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

程序计数器低字节寄存器 — PCL

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

表格寄存器 — TBLP, TBLH

这两个特殊功能寄存器对储存在程序存储器中的表格进行操作。TBLP 为表格指针，指向表格数据的地址。它的值必须在任何表格读取指令执行前加以设定，由于它的值可以被如 INC 或 DEC 指令所改变，这就提供了一种简单的方法对表格数据进行读取。表格读取数据指令执行之后，表格数据高字节存储在 TBLH 中。其中要注意的是，表格数据低字节会被传送到使用者指定的地址。

状态寄存器 — STATUS

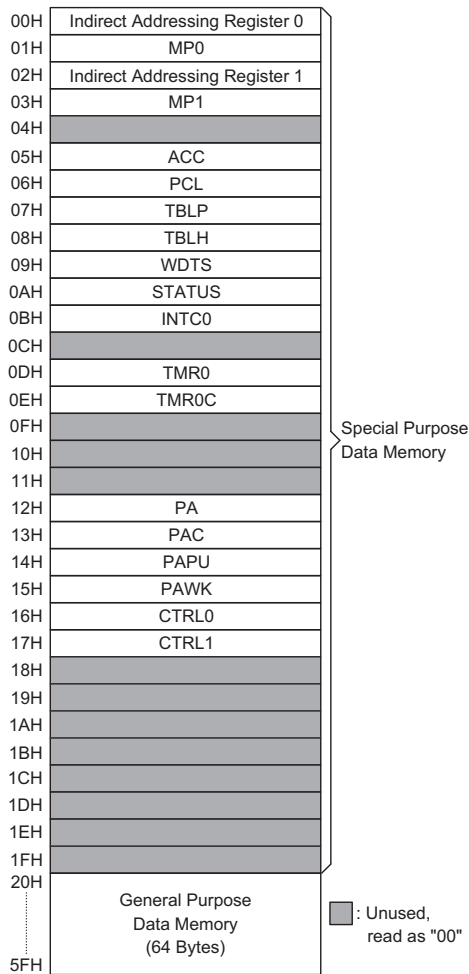
8 位的 STATUS 寄存器包含零标志位 (Z)、进位标志位 (C)、辅助进位标志位 (AC)、溢出标志位 (OV)、暂停标志位 (PDF) 和看门狗溢出标志位 (TO)，它同时记录算术/逻辑运算的和系统工作状态数据。

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

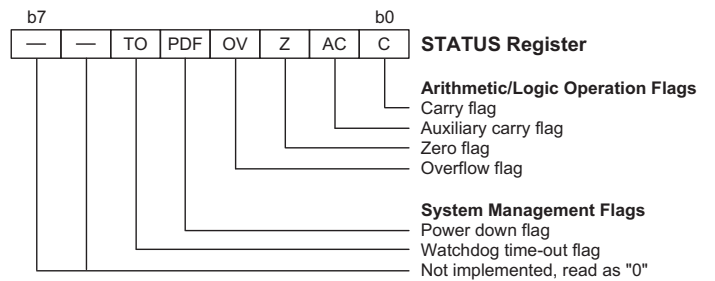
Z、OV、AC 和 C 标志位通常反映最近运算的状态。

- **C** 当加法运算的结果产生进位，或减法运算的结果没有产生借位时，则 C 被置位，否则 C 被清零，同时 C 也会被带进位/借位的移位指令所影响。
- **AC** 当低半字节加法运算的结果产生进位，或高半字节减法运算的结果没有产生借位时，AC 被置位，否则 AC 被清零。
- **Z** 当算术或逻辑运算结果是零时，Z 被置位，否则 Z 被清零。
- **OV** 当运算结果高两位的进位状态异或结果为 1 时，OV 被置位，否则 OV 被清零。
- **PDF** 系统上电或执行“CLR WDT”指令会清零 PDF，而执行“HALT”指令则会置位 PDF。
- **TO** 系统上电或执行“CLR WDT”或“HALT”指令会清零 TO，而当 WDT 溢出则会置位 TO。

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



特殊功能寄存器



状态寄存器

中断控制寄存器 — INTC0

8 位的 INTC0 寄存器用来控制外部和内部所有中断的动作。通过标准的位操作指令来设定这些寄存器的位，每个中断的使能/除能功能可分别被控制。INTC 寄存器内的总中断位 EMI 控制所有中断的使能/除能，用来设定所有中断使能位的开或关。当一个中断程序被响应时，就会自动屏蔽其它中断，EMI 位将被清除，而执行“RETI”指令则会置位 EMI 位。

定时/计数器寄存器

该系列的单片机提供了一个 8 位向上计数的定时/计数器。TMR0 寄存器用于存放 8 位设定值，可以预先写入固定的数据，以允许设定不同的时间中断。而 TMR0C 寄存器用来设置定时/计数器控制信息。

输入/输出端口和控制寄存器

在特殊功能寄存器中，PA 和它对应的控制寄存器很重要。这些输入/输出寄存器映射到数据存储器的特定地址，用以传送端口上的输入/输出数据。PAC 是输入/输出端口 PA 相对应的控制寄存器，用来设定引脚的状态，以决定是输入口还是输出口。要设定一个引脚为输入，必须先设定控制寄存器对应的位为高，若要引脚设定为输出，则控制器对应位必须设置为低。程序初始化期间，在从输入/输出端口中读取或写入数据前，必须先设定控制寄存器的位以确定引脚为输入或输出。使用“SET [m].i”和“CLR [m].i”指令可以直接设定这一寄存器的某一位。这种在程序中可以通过改变输入/输出端口控制寄存器中某一位而直接改变端口的输入/输出口状态的能力是此单片机非常有用的特性。

系统控制寄存器 —CTRL0

系统控制寄存器 CTRL0 是用来控制某些内部功能，包括系统时钟选项，蜂鸣器选择控制以及 RTC 快速起振功能。

当使用蜂鸣器功能时，BZ 或 \overline{BZ} 禁用对应的引脚将作为普通的 I/O 端口；如果其使能 BZ 或 \overline{BZ} 将作为对应的输出口。

系统控制寄存器 —CTRL1

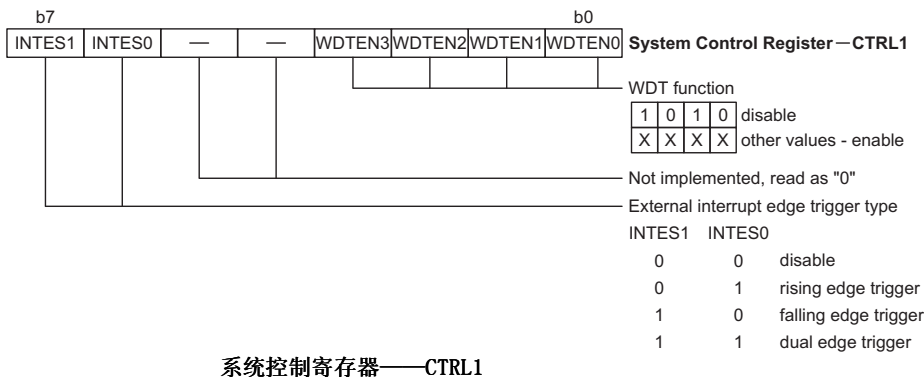
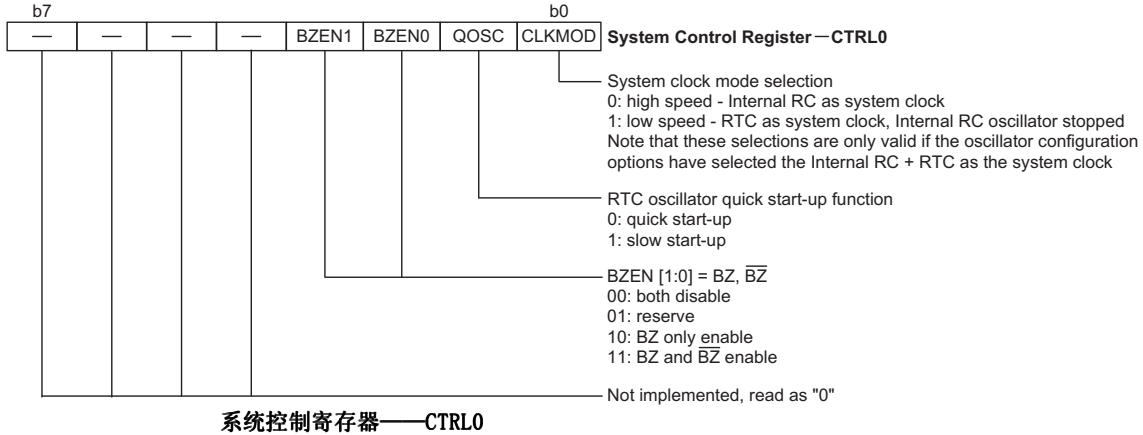
系统控制寄存器 CTRL1 是用来控制某些内部功能，包括外部中断边沿触发类型和 WDT 控制功能。

唤醒功能寄存器 —PAWK

当微控制器进入掉电模式，有多种方法使其唤醒并继续正常执行。其中的一种就是通过 I/O 口的下降沿来唤醒芯片。PAWK 是用来选择哪些 I/O 具有唤醒功能。

上拉寄存器 —PAPU

如果 I/O 口被定义为输入端口，其内部由一个上拉电阻而不需要外部上拉。PAPU 寄存器用来选择哪些 I/O 口具有内部上拉功能。



输入/输出端口

盛群单片机的输入/输出端口控制具有很大的灵活性。它有 7 个引脚在程序控制下可以被指定为输入或输出、所有引脚的上拉电阻选项、以及指定引脚的唤醒选择，这些特性也使得此类单片机在广泛应用上都能符合开发的要求。

每个器件都有一组 A 端口，对应内部的端口寄存器 PA，作为特殊功能寄存器映射到数据存储地址中。PA 口 7 个引脚可作为 I/O 口，一个只能作为输入口。作为输入操作时，输入/输出引脚无锁存功能，输入数据必须在指令“MOV A, [m]” T2 上升沿准备好，m 表示端口地址。对于输出操作，所有数据是锁存的，而且持续到输出锁存被重写。

上拉电阻

许多产品应用在端口处于输入状态时需要外加一个上拉电阻来实现上拉的功能。为了免去这个外加的电阻，PA0—PA6 端口，当引脚设置为输入时，可由内部连接上拉电阻，这些上拉电阻可通过掩膜选项来加以选择，它用一个 PMOS 晶体管来实现。

PA 口唤醒

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

输入/输出端口控制寄存器

PA 输入/输出端口的控制寄存器 PAC 用来控制输入/输出状态。利用此控制寄存器，PA0—PA6 每一个 CMOS 输出或者输入不管有没有上拉电阻设置，均可利用软件控制方式加以动态的重新设置。所有 PA0—PA6 输入/输出端口的引脚都各自对应于输入/输出端口控制寄存器的某一位。若输入/输出引脚要实现输入功能，则对应的控制寄存器位必须设定为“1”，这时程序指令可以直接读出输入引脚的逻辑状态。如果引脚的控制寄存器位被设定为“0”，则此引脚被设置为 CMOS 输出。当引脚被设置为输出状态，程序指令读取的是输出端口寄存器的内容。请注意当输入/输出端口被设置为输出状态时，此时如果对输出口做读取的动作，则会读取到内部数据寄存器中的锁存值，而不是输出引脚实际的逻辑状态。

引脚共用功能

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

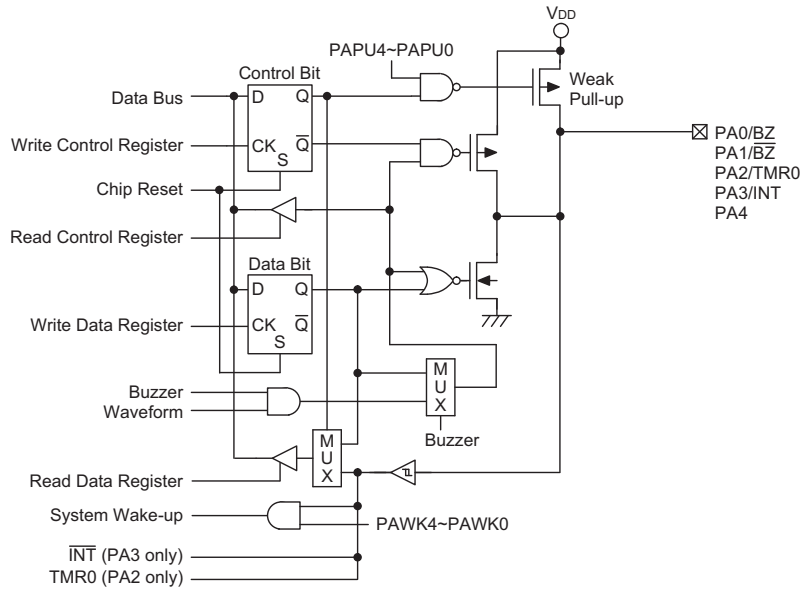
- 外部中断输入
外部中断引脚 INT 与输出引脚 PA3 共用。如果需要外部中断输入，INTCO 控制寄存器中的相应位必须正确的设置，相应的引脚也必须通过端口控制寄存器中的位设置为输入。需要注意的是即使作为外部中断输入引脚，通过相应的寄存器设置仍可以选择端口的上拉功能。
- 外部时钟输入
单片机有一个外部定时/计数器。外部时钟信号输入引脚 TMRO 与 PA2 共用。如果引脚被设定为计数器输入，需要定时/计数器设置为计数或脉冲测量模式，定时/计数控制寄存器中相应的控制位也必须正确设置。同时相应的引脚端口通过端口控制寄存器设置为输入。需要注意的是即使作为外部定时器中断输入引脚，通过相应的寄存器设置仍可以选择端口的上拉功能。

- 蜂鸣器输出

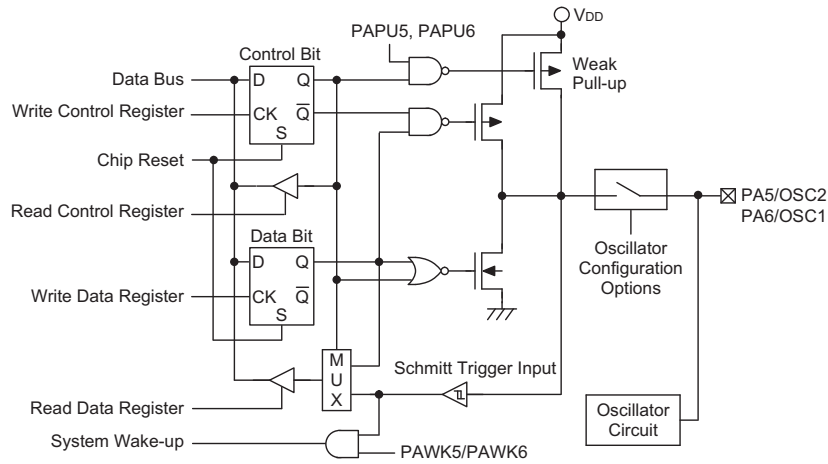
蜂鸣器输出引脚 BZ 或 $\overline{\text{BZ}}$ 分别与 PA0 和 PA1 口共用。蜂鸣器功能是通过设置 CTRL0 相应的位来实现。当要使能蜂鸣器输出功能时，对应引脚的端口控制寄存器相应的位设置为输出。需要注意的是，如果对应端口控制寄存器 PAC 相应的位设置为输入，即使选择了蜂鸣器功能，引脚仍将作为可以有上拉选项的普通输入口。

输入输出结构

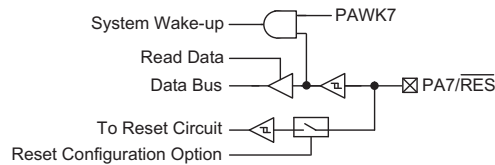
下列为输入/输出端口的内部结构图，可能与芯片内部实际的结构并不完全相同，只是用于帮助用户理解输入/输出端口。



PA0~PA4输入/输出端口



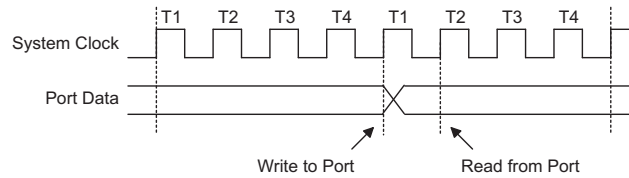
PA5~PA6 输入/输出端口



PA7输入端口

编程注意事项

在使用者的程序中，最先要考虑的是端口的初始化。复位之后，PA 输入/输出数据及端口控制寄存器都将被设为逻辑高。这意味着 PA 默认为输入状态，而其电平则取决于其它相连接电路以及是否选择了上拉选项。如果 PAC 某些引脚位被设定输出状态，这些输出引脚会有初始高电平输出，除非数据寄存器端口 PA 在程序中被预先设定。设置哪些引脚是输入及哪些引脚是输出，可通过设置正确的值到适当的端口控制寄存器，或者使用指令“SET [m].i”及“CLR [m].i”来设定端口控制寄存器中个别的位。注意的是当使用这些位控制指令时，一个读-修改-写的操作将会发生。单片机必须先读入整个端口上的数据，修改个别的位，然后重新把这些数据写入到输出端口。



读写时序

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

定时/计数器

定时/计数器在任何单片机中都是一个很重要的部分，提供程序设计者一种实现和时间有关功能的方法。器件包含一个8位的向上计数器。定时/计数器有三种不同的工作模式，可当作一个普通的定时器、外部事件计数器、或者脉冲宽度测量器来使用。内部设置预分频器加大了定时器的范围。

有两个和定时/计数器相关的寄存器，第一个寄存器用来存储实际的计数值，赋值给此寄存器可以设定初始计数值，读取此寄存器可获得定时/计数器的内容。第二个寄存器是定时/计数器的控制寄存器，此寄存器设置定时/计数器的选项，控制定时/计数器的工作模式。定时器的时钟源可来自内部时钟源或在外部定时器引脚。

定时器/计数器在事件计数模式下使用外部时钟源，而时钟源从外部计数器 TMR0 引脚输入，这些引脚与 I/O 口共用。每当外部定时/计数器输入引脚由高电平到低电平或者有低电平到高电平进行转换时（由定时器控制寄存器 TOE 位决定），将使计数器值加一。

配置定时/计数器输入时钟源

内部定时/计数器的时钟源可以来自不同的时钟源，它取决于所用的芯片与使用的定时器。当定时/计数器在定时器或脉冲宽度测量模式时，使用内部时钟作为时钟源。定时/计数器的内部时钟源首先经过一个预分频器，其分频比受定时/计数器控制寄存器的 T0PSC0~T0PSC2 位的值决定。

当定时/计数器的工作在计数模式时，使用外部时钟作为时钟源。外部时钟源来根据芯片或使用定时器的不同源于外部引脚 TMR0 或 TMR1。根据 TOE 位的设置，外部计数引脚每一个上升沿或下降沿都会使计数值加1。

定时/计数器寄存器 —TMR0

定时/计数器寄存器是位于专用数据存储器的特殊功能寄存器，储存实际的定时器值。芯片的定时/计数器寄存器是 TMR0。当用作内部定时器模式时收到一个内部计时脉冲时，或用作外部计数模式时外部定时/计数器引脚发生电平转换时，定时/计数寄存器的值将会加一。定时器从预置寄存器所载入的值开始向上计数，直到8位定时/计数器计到 FFH，此时定时器发生溢出且会产生一个内部中断信号。定时器的值随后被预置寄存器的值重设，定时/计数器继续计数。

注意当预置寄存器被清为零，就可以得到8位定时/计数器 FFH 的最大计算范围。。此时要注意的是，上电后预置寄存器中的数值处于未知状态。定时/计数器在 OFF 条件下，如果把数据写入预置寄存器，这数据将被立即写入实际的定时器。然而如果定时/计数器已经被打开且正在计数，在这个周期内写入到预置寄存器的任何新数据将被保留在预置寄存器中，等到下一个溢出发生时才会被写入实际的定时器。

定时/计数器控制寄存器 —TMROC

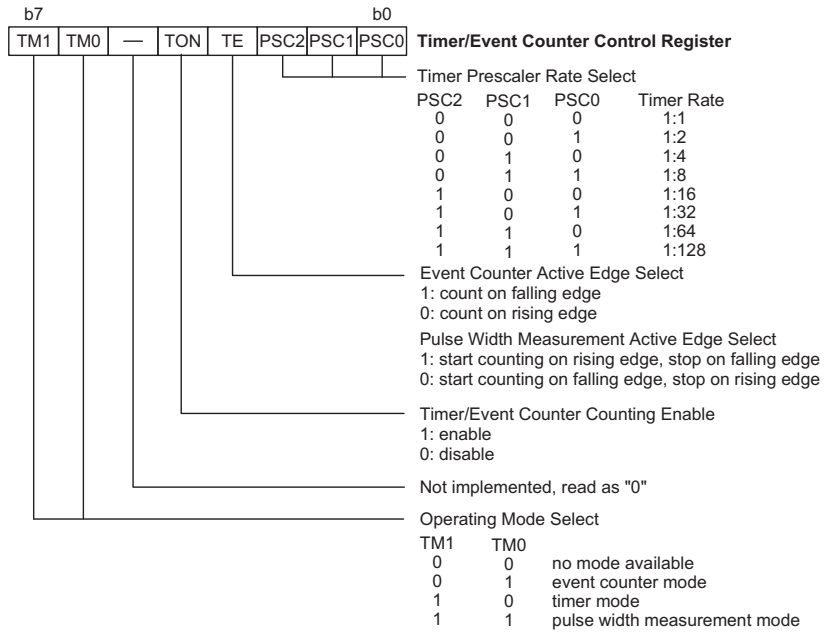
定时/计数器能工作在三种不同的模式，至于选择工作在哪一种模式则是由各自的控制寄存器的内容决定。

定时/计数器寄存器 TMROC 和定时/计数控制寄存器控制计时/事件计数器的全部操作。在定时器使用之前必须先正确地设定定时/计数控制寄存器，以便保证定时器能正确操作，而这个过程通常在程序初始化期间完成。

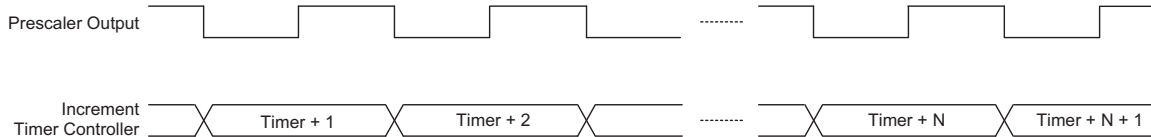
定时器工作模式有三种，定时器模式，事件计数模式和脉冲宽度测量模式。为了确定定时器工作在哪一种模式，TMROC 寄存器的第7位和第6位 (TOM1/TOM0) 必须设定到要求的逻辑电平。定时器打开位 TOON，即定时/计数控制寄存器的第4位，是定时器控制的开关，设定逻辑高时定时器开始计数，而清零时则定时器停止计数。对8位定时/计数器而言，定时/计数控制寄存器的位0~2 决定输入定时预分频器 (Prescaler) 中的分频比例。如果使用外部计时源，预分频器 (Prescaler) 将不作用。如果定时器工作在事件计数或脉冲宽度测量模式，TOE 的逻辑电平即定时器控制寄存器的第3位将可用来选择上升或下降沿触发。

配置定时器模式

在这个模式，定时/计数器可以用来测量固定时间间距，当定时器发生溢出时，就会提供一个内部中断信号。要工作在这个模式，位TM1/TM0必须分别设为1和0。在这个模式，内部时钟源被用来当定时器的计时源。定时/计数器的输入计时频率是 f_{SYS} 或 f_{RTC} 除以定时器预分频器(Prescaler)的值，预分频器的值是由相应的寄存器的TOPSC0~TOPSC2位来决定。定时器打开位TON 必须被设为逻辑高才能让定时器开始工作。每次内部时钟由高到低的电平转换都会使定时器值增加一；当定时器已满即溢出时，会产生中断信号且定时器会重新载入已经载入到预置寄存器的值，然后继续向上计数。定时器溢出是中断的一种，也是唤醒暂停模式的一种方法，然而，内部中断也可以被屏蔽，方法是将相关中断寄存器的E0TI 置零。



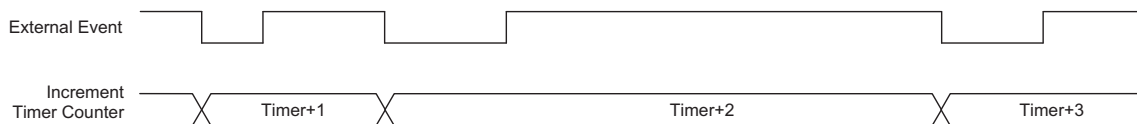
TMROC寄存器



定时器模式时序图

配置事件计数模式

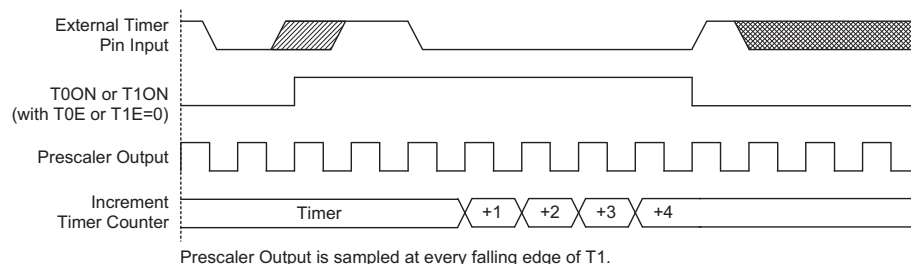
在这个模式，发生在外部引脚的外部逻辑事件改变的数目，可以通过内部定时/计数器来记录。为使定时/计数器工作于事件计数模式，位TOM1/TOM0必须分别设为0 和1。定时器打开位TOON 必须设为逻辑高，令定时器开始计数。当TOE为逻辑低时，每次外部定时/计数器引脚接收到由低到高电平的转换将使计数器加一。而当TOE为逻辑高时，每次外部定时/计数器引脚接收到由高到低电平的转换将使计数器加一。与另外两个模式一样，当计数器计满时，将会发生溢出且产生一个内部中断信号，同时定时/计数器重新载入一个已经载入到预置寄存器的值。如果外部计数输入与其它I/O 共用引脚，为确保事件计数模式正常工作，除了TOM1/TOM0位需设定在事件计数模式，还需通过输入/输出端口控制寄存器将该 I/O 设定在输入模式。计数器溢出是中断的一种，也是唤醒暂停模式的一种方法。应当注意的是，定时器的溢出是中断的一种，也是唤醒暂停模式的一种。注意， INTC0的ET0I位置零可以屏蔽中断。



外部计数模式时序图

配置脉冲宽度测量模式

在这个模式，可以测量外部定时/计数器引脚的外部脉冲宽度。在脉冲宽度测量模式中，定时/计数器时钟源由内部时钟提供，而位TOM1/TOM0则必须都设为逻辑高。如果TOE位是逻辑低，当外部定时/计数器引脚接收到一个由高到低电平的转换时，定时/计数器将开始计数直到外部定时/计数器引脚回到它原来的高电平，此时TOON位将自动清除为零且定时/计数器将停止计数。而如果TOE位是逻辑高，则当外部定时/计数器引脚接收到一个由低到高电平的转换时，定时/计数器开始计数直到外部定时/计数器引脚回到原来的低电平。如上所述， TOON位将自动清除为零，且定时/计数器停止计数。请注意，在脉冲宽度测量模式中， TOON位将自动地清除为零且定时/计数器会停止计数，而在其他两种模式下， TOON位要在程序控制下才会被清除为零。这时定时/计数器中的剩下的值被程序读取，可以由此得知外部计数引脚接收到的脉冲的长度。TOON位被清零时，任何在外部定时/计数器引脚的进一步转换将被忽略，TOON位再次被程序设定为逻辑高，定时/计数器才又开始脉冲宽度测量。利用这种方法可以轻松完成单一脉冲测量。要注意的是在这种模式下，定时/计数器是通过外部定时/计数器引脚上的逻辑转换来控制，而不是通过逻辑电平。与另外两个模式一样，当定时/计数器已满，将溢出且产生一个内部中断信号，定时/计数器也将清零并载入预置寄存器的值。如果外部计数输入与其他I/O 共用引脚，为确保脉冲宽度测量模式正常工作，要注意两点。第一点是要将TOM1/TOM0位设定在脉冲宽度测量模式，第二点是确定此引脚的输入/输出口控制寄存器对应位被设定为输入状态。这种定时器溢出是中断的一种，也是唤醒暂停模式的一种方法。注意， INTC0的ET0I位置零可以屏蔽中断。



脉冲宽度测量模式时序图

预分频器

定时/计数器控制寄存器的第0位~第2位T0PSC0~T0PSC2可以用来定义定时/计数器中内部时钟源的预分频级数。

蜂鸣器功能

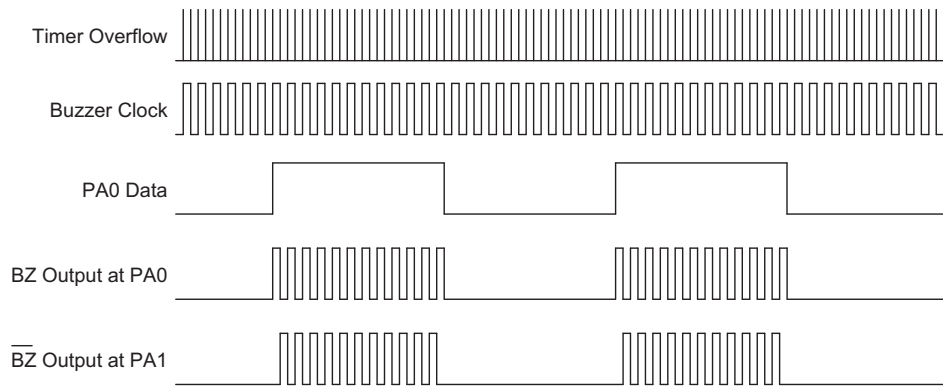
与可编程分频器的作用相似，单片机中蜂鸣器的功能在于能提供可变频率输出，以适用于像压电蜂鸣器驱动、或其它需要精确频率输出的场合。

BZ 和 $\overline{\text{BZ}}$ 是一对和输入/输出引脚PA0 与PA1 共用的蜂鸣器输出。可通过CTRL0选择单一BZ 输出或是BZ 和 $\overline{\text{BZ}}$ 输出。如果不选择为蜂鸣器输出，其功能即如正常的输入/输出引脚。要注意的是 $\overline{\text{BZ}}$ 引脚是BZ 引脚的反向输出，两者配合可以产生更强的输出功率驱动蜂鸣器等器件。通过CTRL0寄存器可以选择单个BZ输出或BZ与 $\overline{\text{BZ}}$ 输出。

蜂鸣器电路使用定时器溢出信号作为其时钟源。载入合适的值到定时器预分频器(Prescaler)，可以产生需要的时钟源分频比例，由此来控制输出的频率。系统时钟被预分频器(Prescaler)分频后的时钟源，进入定时器计时，定时器从预置寄存器的值开始往上计算，直到计数值满并产生溢出信号，并改变BZ/ $\overline{\text{BZ}}$ 输出状态。定时器将自动地重新载入预置寄存器的值，并继续往上计数。

如果寄存器CTRL0选择了蜂鸣器功能，那么蜂鸣器输出引脚将工作。本质上是PA口控制寄存器PAC的0位与1位将设置位输出。如果一个引脚设置成蜂鸣器输出那么另外一个引脚仍可以设置为普通的输入口；但是如果两个引脚均设置为输入那么蜂鸣器功能将不起作用。只有在PA0口输出“1”时蜂鸣器功能才起作用。这个数据作为控制蜂鸣器输出的控制位。需要注意：如果PA0口输出数据位被清零，蜂鸣器输出口BZ/ $\overline{\text{BZ}}$ 为低电平。PA1口的输出数据对BZ/ $\overline{\text{BZ}}$ 的输出无影响。

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



蜂鸣器功能

输入输出口

当运行在事件计数器或脉冲宽度测量模式时，定时/计数器需要使用外部定时/计数器引脚以确保正确的动作。要实现这些必须正确地配置定时/计数器控制寄存器模式选择的位来选择是工作在计数或是脉宽测量模式。另外端口控制寄存器PAC相应的位必须设置引脚为输入，即使这些脚设置为事件计数输入与引脚相关的上拉电阻仍然存在。

编程注意事项

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

当读取定时/计数器或写数据到预置寄存器时，计数时钟会被阻隔以避免错误发生，但这样做可能会导致计数错误，所以程序设计师应该考虑到这点。在第一次使用定时/计数器之前，要仔细确认有没有正确地设定初始值。中断控制寄存器中的定时器使能位必须正确的设置，否则相应定时/计数器内部中断仍然无效。定时/计数器控制寄存器中的边沿触发选择、定时/计数器工作模式和时钟源控制位也必须正确的设定，以确保定时/计数器按照应用需求而正确的配置。在定时/计数器打开之前，必须确保先载入定时/计数器寄存器的初始值，这是因为在上电后，定时/计数器寄存器中的初始值是未知的。定时/计数器初始化后，可以使用定时/计数器控制寄存器中的使能位来打开或关闭定时器。

当定时/计数器发生溢出，相应的中断请求标志将置位。若中断允许，将会产生一个中断信号。不管中断是否允许，在 HALT 状态下，定时/计数器的溢出也会产生唤醒，这种情况可能发生在外部信号变化的计数模式中，定时/计数器向上计数直至溢出并唤醒系统。若在 HALT 模式下，不需要定时器中断唤醒系统，可以在进入 HALT 前将相应中断请求标志位置 1。

定时/计数器应用范例

这个例子说明了如何设置定时/计数器 0 的寄存器，如何设置和控制中断。另外还需注意的是，怎样通过 TMROC 寄存器的第 4 位来启停定时/计数器。此应用范例设置定时/计数器 0 为定时模式，时钟来源于内部的系统时钟。

```

::
org 04h           ; external interrupt vector
org 08h           ; Timer Counter 0 interrupt vector
jmp tmr0int      ; jump here when Timer 0 overflows
::
org 20h           ; main program
::
;internal Timer 0 interrupt routine
tmr0int:
:
; Timer 0 main program placed here
:
:
begin:
;setup Timer 0 registers
mov a,09bh       ; setup Timer 0 preload value
mov tmr0,a
mov a,081h       ; setup Timer 0 control register
mov tmr0c,a      ; timer mode and prescaler set to /2
;setup interrupt register
mov a,00dh       ; enable master interrupt and both timer interrupts
mov intc0,a
::
set tmr0c.4      ; start Timer 0
::

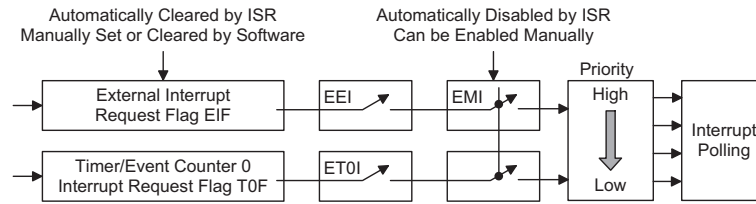
```

中断

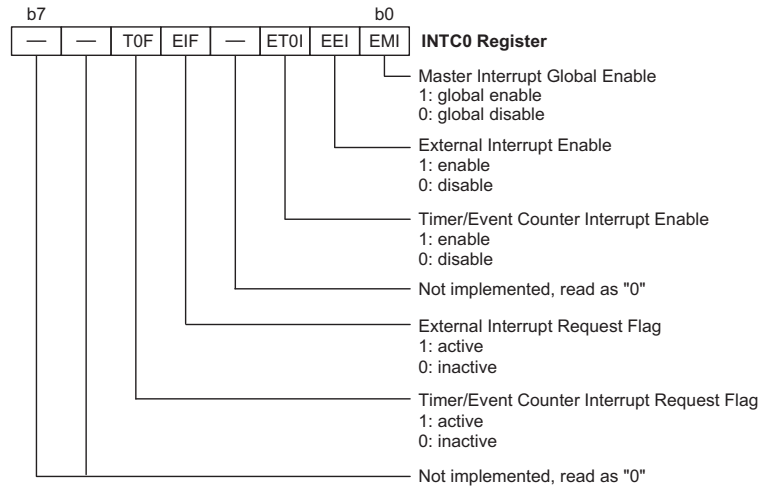
中断是单片机的一项重要功能。当外部事件中断或内部中断如定时/计数器有效，系统会中止当前的程序，而转到相对应的中断服务程序。单片机一个外部中断和一个内部中断，外部中受外部中断引脚的状态控制，内部中断受定时/计数器的溢出控制。

中断寄存器

所有的中断允许和中断请求标志均由数据存储寄存器内的 INTC0 寄存器控制。通过控制相应的中断允许位使能或禁止相关的中断。当发生中断，相应中断请求标志将被置位。总中断请求标志清零将禁止所有中断允许。



中断示意图



中断控制寄存器INTC0

中断操作

如果中断使能，定时/计数器溢出或外部中断边沿有效将产生中断请求标志位。当中断发生，下一条指令的地址将被压入堆栈，相应的中断向量地址加载至PC中，系统将从此向量取下条指令。中断向量处通常为跳转指令，以跳转到相应的中断服务程序。中断服务程序必须以RETI指令返回至主程序，以执行原来的程序。

各种中断使能及它们的中断请求标志、优先级图形所示。

一旦中断子程序被响应，系统将自动清除 EMI 位，所有其它的中断将被屏蔽，这个方式可以防止任何进一步的中断嵌套。如果其它中断请求可能发生在此期间，只有中断请求标志位会被记录。如果某个中断服务子程序正在执行，此时有另一个中断要求响应，EMI 位可以被置位，以允许此中断被响应。如果堆栈已满，即使此中断使能，中断请求也不会被响应，直到 SP 减少为止。如果要求立刻动作，则堆栈必须避免成为储满状态。

中断优先级

中断发生在两个连续的 T2 脉冲上升沿之间，如果相应的中断请求被允许，中断将在后一个 T2 脉冲响应。下表指出同时提出请求的情况下所提供的优先级，可以通过重新设定 EMI 位来加以屏蔽。

中断源	HT48R01A
外部中断	1
定时/计数器 0 溢出中断	2

假使外部和内部中断均被使能，且外部和内部中断同时发生，则外部中断永远优先处理，首先被响应。使用控制寄存器适当地屏蔽个别中断，可以防止中断同时发生的情况。

外部中断

使外部中断发生，总中断控制位 EMI、外部中断使能位 EEI 必须先被置位。当外部中断有效触发，控制寄存器中 EIF 位将被设定。由于外部中断可以双边沿触发，当 INT 引脚出现上升沿触发或下降沿触发时，EIF 标志位将被设置为 1。中断的触发类型（上升沿，下降沿或双边沿触发）取决于控制寄存器 CTRL1 的第 6 位 INTES0 和第 7 位 INTES1。这两位也可以屏蔽掉中断。

INTES1	INTES0	边沿触发类型
0	0	屏蔽
0	1	上升沿触发
1	0	下降沿触发
1	1	多边沿触发

外部中断和 PA3 共享引脚，使用外部中断时必须将 INTC0 寄存器中外部中断使能位设置为 1，系统控制寄存器 CTRL1 设置相应的触发边沿，同时将 PAC.3 位设置为 1；如果外部中断使能位未设置为 1，当中断使能、堆栈未满且外部中断产生时，将调用位于地址 04H 处的子程序。当响应外部中断服务子程序时，中断请求标志位 EIF 会被复位且 EMI 位会被清零以除能其它中断。需要注意的是即使外部中断使能与外部中断输入相连接的内部上拉电阻依然存在。

定时/计数器中断

使定时/计数器中断发生，总中断控制位 EMI、INTCO 寄存器中定时/计数器中断使能位 ETOI 必须先被置位。当定时/计数器发生溢出，INTCO 寄存器中 TOF 中断请求标志位置位并触发定时/计数器中断。若中断使能，堆栈未满，当发生定时/计数器中断时，将调用位于地址 08H 处的子程序。当响应定时中断服务子程序时，中断请求标志位 TOF 会被复位且 EMI 位会被清零以除能其它中断。

编程注意事项

通过禁止中断使能位，可以屏蔽中断请求，然而，一旦中断请求标志位被设定，它们会被保留在 INTCO 中断控制寄存器内，直到相应的中断服务子程序执行或被软件指令清除。

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

所有中断都具有唤醒功能。

当进入中断服务程序，系统仅将程序计数器的内容压入堆栈，如果有影响主程序流程的寄存器或状态寄存器被中断服务程序改变，应事先将这些数据保存起来。

复位和初始化

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

除了上电复位外，即使单片机正在执行状态，有些情况的发生也迫使单片机必须加以复位。其中一个例子是当提供电源给单片机以执行程序后， $\overline{\text{RES}}$ 引脚被强制拉至低电平。这个例子为正常操作复位，单片机中只有一些寄存器受影响，而大部分寄存器不受影响，以便复位引脚回复至高电平后，单片机仍可以正常操作。复位的另一种形式是看门狗定时器溢出而复位单片机，所有复位操作类型导致不同的寄存器条件被加以设定。

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

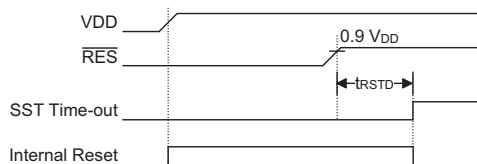
复位功能

通过内部与外部事件触发复位，单片机共有五种复位方式：

- 上电复位

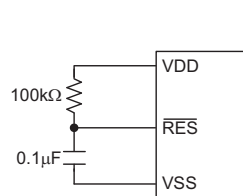
这是最基本且不可避免的复位，发生在单片机上电后。除了保证程序存储器会从起始地址开始执行，上电复位也使得其它寄存器被设定在预设条件，所有的输入/输出端口寄存器和输入/输出端口控制寄存器在上电复位时会保持高电平，以确保上电后所有引脚被设为输入状态。

虽然单片机有一个内部 RC 复位功能，如果电源上升缓慢或刚上电时电源不稳定，内部 RC 振荡可能会导致芯片复位不良，所以推荐使用和 $\overline{\text{RES}}$ 引脚连接的外部 RC 电路。由 RC 电路所造成的时间延迟使得 $\overline{\text{RES}}$ 引脚在电源供应稳定前的一段延长周期内保持在低电平。在这段时间内，单片机的正常操作是被禁止的。 $\overline{\text{RES}}$ 引脚达到一定电压值后，再经过延迟时间 t_{RSTD} ，单片机可开始进行正常操作。下图中 SST 是系统延迟周期 System Start-up Timer 的缩写。

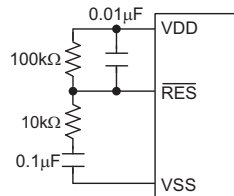


上电复位时序图

在许多应用场合，可以在 VDD 与 $\overline{\text{RES}}$ 之间连接电阻，而在 $\overline{\text{RES}}$ 与 VSS 之间连接电容作为复位电路，为了减少干扰影响，至 $\overline{\text{RES}}$ 引脚的连线应尽量短。



基本型复位电路



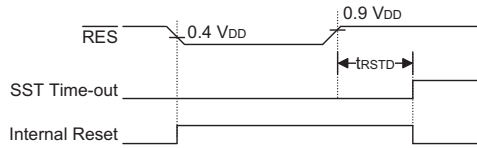
增强型复位电路

当系统在较强干扰的场合工作时，强烈建议使用增强型复位电路。

欲知更多外部复位电路的相关信息可参考 HOLTEK 网站应用范例 HA0075S。

● **RES** 引脚复位

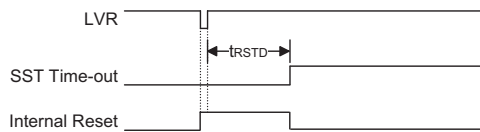
当单片机正常工作，**RES** 引脚通过外部硬件(如外部开关)强迫拉至低电平时，此种复位形式即会发生。这种复位形式与其它复位一样，程序计数器会被清除为零且程序从头开始执行。



RES 引脚复位时序图

● 低电压复位

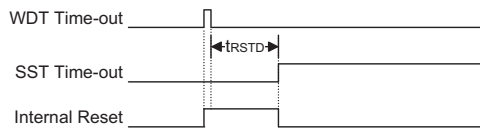
单片机具有低电压复位电路，用来监测它的电源电压，可通过掩膜选项进行选择。例如在更换电池的情况下，单片机供应的电压可能会落在 $0.9V \sim V_{LVR}$ 的范围内，这时 **LVR** 将会自动复位单片机。**LVR** 包含以下的规格：有效的 **LVR** 信号，即在 $0.9V \sim V_{LVR}$ 的低电压状态的时间，必须超过交流电气特性中 t_{LVR} 参数的值。如果低电压存在不超过 t_{LVR} 参数的值，则 **LVR** 将会忽略它且不会执行复位功能。 V_{LVR} 参数值可通过掩膜选项进行设定。



低电压复位时序图

● 正常操作时看门狗溢出复位

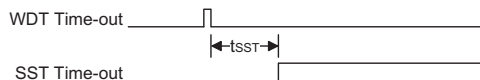
除了看门狗溢出标志位 **TO** 将被设为 1 之外，正常操作时看门狗溢出复位和 **RES** 复位相同。



正常操作时看门狗溢出复位时序图

● 暂停时看门狗溢出复位

暂停时看门狗溢出复位和其它种类的复位有些不同，除了程序计数器与堆栈指针将被清 0 及 **TO** 位被设为 1 外，绝大部份的条件保持不变。图中 t_{SST} 的详细说明请参考交流电气特性。



暂停时看门狗溢出复位时序图

注：如果系统时钟为外部晶体，**SST** 可以使能（1024 个系统时钟）或除能（仅需 2 个系统时钟）

复位初始状态

不同的复位形式以不同的途径影响复位标志位。这些标志位，即 PDF 和 T0 位存放在状态寄存器中，由暂停功能或看门狗计数器等几种控制器操作控制。复位标志位如下所示：

T0	PDF	复位条件
0	0	上电时的 $\overline{\text{RES}}$ 复位
u	u	正常运行时的 $\overline{\text{RES}}$ 或 LVR 复位
1	u	正常运行时的 WDT 溢出复位
1	1	暂停时的 WDT 溢出复位

注：“u”代表不改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

项目	复位后情况
程序计数器	清除为零
中断	所有中断被禁止
看门狗定时器	WDT 清除并重新计时
定时/计数器	定时/计数器停止
预分频器	定时/计数器之预分频器内容清除
输入/输出口	所有 I/O 设为输入模式
堆栈指针	堆栈指针指向堆栈顶端

不同的复位形式以不同的途径影响单片机中的内部寄存器。为保证复位发生后程序正常执行，了解单片机特定的复位后的情况是非常重要的。下表描述了不同复位影响单片机的内部寄存器。

寄存器	复位 (上电复位)	$\overline{\text{RES}}$ 或 LVR 复位	WDT 溢出 (正常运行)	WDT 溢出 (暂停模式)
MP0	1xxx xxxx	uuuu uuuu	1uuu uuuu	1uuu uuuu
MP1	1xxx xxxx	uuuu uuuu	1uuu uuuu	1uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu
WDTS	---- -111	---- -111	---- -111	---- -uuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1000	0000 1000	0000 1000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	-000 0000	-000 0000	-000 1000	-uuu uuuu
PAWK	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTRL0	---- 0000	---- 0000	---- 0000	---- uuuu
CTRL1	10—1010	10—1010	10—1010	uu—uuuu

注：
 “*”表示热复位
 “u”表示未变化。
 “x”表示未确定。
 “-”表示未使用。

振荡器

不同的振荡器选择可以让使用者在不同的应用需求中获得更多范围的功能。有四种系统时钟可供选择，而看门狗定时器又有多种时钟源选项，提供了使用者最大的灵活性。所有振荡器的选择，是通过配置选项寄存器来实现的。

有四种方法产生系统时钟：

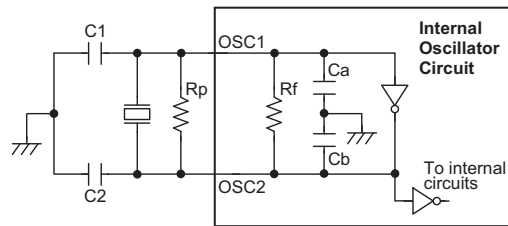
- 外部晶体/陶瓷振荡器
- 外部 RC 振荡器
- 内部 RC 振荡器
- 结合内部 RC 振荡和实时时钟

四种时钟选择必须通过选项来选择其中之一。

欲知更多振荡器电路的相关信息可参考 HOLTEK 网站应用范例 HA0075S。

外部晶体/陶瓷振荡器

对于晶体振荡器的结构，只要简单地将晶体连接至 OSC1 和 OSC2，则会产生所需的相移及反馈，而不需其它外部的器件。然而为了保证某些低频率的晶体振荡和所有陶瓷共振器的应用，建议使用两个小容量电容 C1 和 C2，具体数值与客户选择的晶体/陶瓷晶振有关。外部并联的反馈电阻，即 R_p 在某些场合并不需要，仅用来辅助系统启动。



Note: 1. R_p is normally not required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

外部晶体/陶瓷振荡器

内部 C_a , C_b , R_f 典型值@5V, 25°C		
C_a	C_b	R_f
11pF~13pF	13pF~15pF	270kΩ

振荡器内部元件值

晶体振荡器 C_1 和 C_2 值			
晶体频率	C_1	C_2	C_L
12MHz	TBD	TBD	TBD
8MHz	TBD	TBD	TBD
4MHz	TBD	TBD	TBD
1MHz	TBD	TBD	TBD

注：1、 C_1 , C_2 值仅作为参考。
2、 C_L 为晶体规格测试时的负载电容。

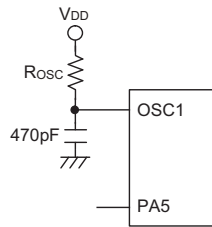
晶体振荡器电容推荐值

陶瓷振荡器 C1 和 C2 值		
陶瓷振荡频率	C1	C2
3.58MHz	TBD	TBD
1MHz	TBD	TBD
455KHz	TBD	TBD
注：C1, C2 值仅作为参考。		

陶瓷振荡器电容推荐值

外部 RC 振荡器

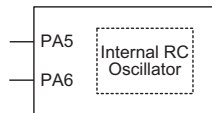
使用外部 RC 电路作为系统振荡器，需要在 OSC1 和 VDD 之间连接一个阻值约在 24 kΩ 到 1.5MΩ 之间的电阻，OSC1 与 VSS 之间连接一个电容。虽然此振荡器配置成本较低，但振荡频率会因 VDD、温度和芯片本身的制成而改变，因此不适合用来做计时严格或需要精确振荡器频率的场合。对于外部电阻 R_{osc} 的阻值，请参考附录章节中典型 RC 振荡器对温度以及对 V_{DD} 特性曲线分析。外部 RC 振荡器仅使用 OSC1 引脚，OSC1 与 PA6 引脚共用，留下 PA5 作为普通输入/输出使用。注意的是系统频率由内部电路和外部电阻 R_{osc} 共同决定，图中显示的外部电容并不会影响振荡器的频率值。



外部 RC 振荡器

内部 RC 振荡器

内部 RC 振荡是一个完全集成在系统内部，不需要接外部器件。内部 RC 振荡器有三种固定的频率：4MHZ, 8MHZ, 12MHZ. 封装型号的后缀决定了选择哪一种内部 RC 频率。注意，由于如果选择了内部的时钟，无需额外的引脚；那么 PA6 与 PA5 可以作为普通的 IO 口。

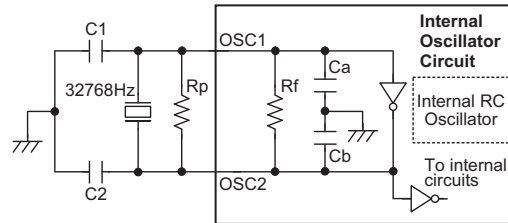


Note: PA5/PA6 used as normal I/Os

内部 RC 振荡器

内部 RC+外部 RTC 振荡器

当系统进入暂停模式，系统时钟关闭以降低功耗。然而在某些应用，比如暂停模式下保持定时/计数功能，必须提供额外的系统时钟。要做此配置选项，存在允许内部 RC 振荡与 RTC 一起的振荡器。下面是与 PA5 和 PA6 共用的 OSC1 与 OSC2 引脚，与外部 32768HZ 晶振相连实现内部的 RTC 功能。对于一些晶体，为了保证起振和频率的精确，建议使用两个小容量电容 C1 和 C2，具体数值与客户选择的晶体规格有关。外部并联的反馈电阻，即 R_p 在某些场合并不需要，仅用来辅助系统启动。



Note: 1. R_p is normally not required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

内部 RC+外部 RTC 振荡器

内部 Ca, Cb, Rf 典型值 @ 5V, 25°C		
Ca	Cb	Rf
11-13pF	13-15pF	270KΩ

实时时钟振荡器内部元件值

实时时钟振荡器 C1 和 C2 值			
晶体频率	C1	C2	CL
32768Hz	TBD	TBD	TBD

注：1、C1, C2 值仅作为参考。
2、CL 为晶体规格测试时的负载电容。

32768Hz 晶体振荡器电容推荐值

对于使用 RTC 振荡器的应用，系统时钟可以选择内部 RC 或 RTC 振荡器。使用 CTRL0 寄存器的 CLKMOD 位来进行选择，如果被置位，系统的时钟来源于 32768 的晶体振荡器；如果被清零则系统的时钟源于内部 RC 振荡器。当进入暂停模式，系统时钟不考虑 CLKMOD 是选择了 RTC 或内部 RC 振荡器，它将停止。

下表说明了各种振荡和 CLKMOD 之间的关系：

操作模式	CLKMOD 位	内部 RC	RTC	系统时钟
正常操作	0	On	On	RC 振荡
	1	On	On	32786Hz
暂停模式	X	Off	On	停止

系统上电后，RTC 振荡器启动会有一些的延时。可以通过设置 CTRL0 寄存器第 1 位 QOSC 位缩短延时来控制快速振荡。系统上电时 QOSC 位清 0 以快速启动 RTC 振荡器，为了降低快速启动的功耗，建议系统上电 2 秒钟后，应用程序设置 QOSC 位为 1。注意的是，无论 QOSC 位是否设置，RTC 振荡器将保持正常工作，仅影响快速启动时的系统功耗。

看门狗定时振荡器

WDT 振荡器是一个完全独立且自由动作的内建振荡器，它在 5V 时的周期时间典型值为 65 μ s 且不需外部的器件搭配。当芯片进入暂停模式，系统时钟停止振荡，但 WDT 振荡器仍继续工作。然而在某些应用中，为了降低功耗，WDT 振荡器可以通过掩膜选项来关闭。

暂停和唤醒

暂停模式

所有 HOLTEK 单片机都具有暂停功能。当单片机进入此模式，由于系统停止振荡，芯片的工作电流会降到极低静态模式。由于单片机保存了当前工作的一些内部条件，它可以被唤醒并继续运行，而不需要重新进行复位。这个特性对于电池供电系统等供电容量受限但需要单片机保存当前工作状态的应用场合特别重要。

进入暂停模式

单片机进入暂停模式仅能通过应用程序执行“HALT”指令实现，且造成如下结果：

- 系统振荡器将被关闭，应用程序将停止在“HALT”指令处
- 如过 RTC 振荡器的配置选项启用，则实时时钟继续运行。
- 在 RAM 芯片和寄存器上的内容保持不变
- 如果 WDT 时钟源是来自 WDT 振荡器，则 WDT 将被清除然后再重新计数；若来源于系统时钟，则停止计数
- 所有输入/输出端口状态保持不变
- STATUS 寄存器中 PDF 标志位被置位，T0 标志位被清零

静态电流

要使系统静态电流降到最小，为毫安级，除了需要单片机进入 HALT 模式，还要考虑到电路的设计。特别要注意输入/输出口的状态。所有高阻抗输入引脚必须接高电平或低电平，否则会造成引脚浮空而引起内部振荡。另外还需要注意单片机输出端口上的负载，当外部电路为 CMOS 输入时，可设置为拉电流。

如果配置选项使能看门狗振荡器，当进入暂停模式，振荡器继续振荡，并继续消耗能量。对电源消耗敏感的应用，使用系统时钟作为 WDT 时钟是更好的选择。如果配置 RTC 使能，当进入暂停模式时也将消耗一定的能量。若要使 RTC 消耗的能量保持在一个较低的水平，那么 CTRL0 中控制快速启动的位 QOSC 应被置位。

唤醒

当系统进入 HALT 模式下，可以通过以下几种方式唤醒：

- 外部复位
- PA 口下降沿
- 系统中断
- WDT 溢出

若由外部 RES 引脚唤醒，系统会经过完全复位的过程。若由 WDT 溢出唤醒，则看门狗计数器将被复位清零。这两种唤醒方式都会使系统复位，可以通过状态寄存器中 T0 和 PDF 位来判断它的唤醒源。系统上电或执行清除看门狗的指令，PDF 被清零；执行 HALT 指令，PDF 将被置位。看门狗计数器溢出将会置位 T0 标志并唤醒系统，同时复位程序计数器和堆栈指针，其它标志保持原有状态。

端口 PA0~PA7 中的每个位都可以通过 PAWUK 寄存器独立选择唤醒功能。PA 端口唤醒后，程序将在“HALT”指令后继续执行。

如果系统是通过中断唤醒，则有两种可能发生，假如中断除能或中断使能但堆栈已满，程序将在“HALT”指令下一条处继续执行，但相应的中断服务程序只有在中断使能或堆栈空闲后被执行；假如中断使能且堆栈未满，则正常的中断响应将会发生。假设在进入暂停模式之前外部中断请求标志位被设为“1”，则相关中断的唤醒功能将无效。

一旦唤醒事件发生，单片机回到正常运行将需要 1024 个系统时钟周期。如果唤醒由中断响应，则实际的中断子程序执行将延迟一个或数个周期；如果唤醒后接着去执行下一条指令，则它在 1024 个系统周期结束后立刻执行。

看门狗定时器

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

通过配置选项和两个内部的寄存器 WDTN3 和 CTRL1 可以设置不同的 WDT 选项。通过配置选项和内部数据存储器中特殊功能寄存器 CTRL1 的 WDTEN 位，来使能看门狗定时器。

配置选项	CTRL1 寄存器	WDT 功能
禁止	禁止	Off
使能	禁止	0n
禁止	使能	0n
使能	使能	0n

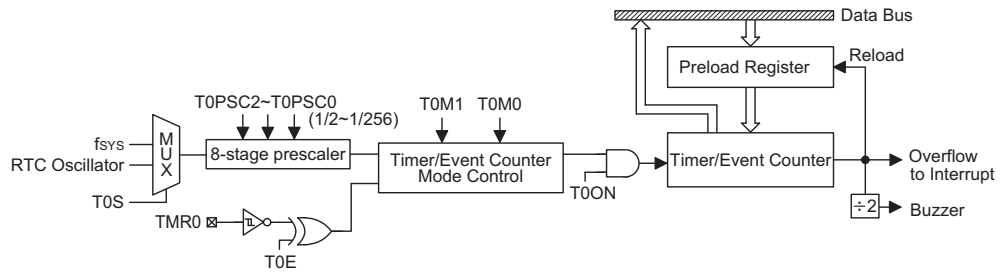
看门狗开/关控制

如果 WDT 配置选项和 CTRL1 的 WDTEN3~WDTEN0 位被写入 1010B，看门狗定时器将被禁止。这是设备加电时的情况。虽然向 CTRL1 的 WDTEN3~WDTEN0 位写入其他的任何数字都可确保看门狗定时器使能，但为了最大程度保护它，建议向这些位写入 0101B。

通过配置选项可以选择三种不同看门狗定时器的时钟来源：集成的内部 WDT 振荡器，RTC 时钟或 $f_{SYS}/4$ 。WDT 的专用内部时钟源是内部振荡器，在供应电压为 5V 时周期近似为 65 μ s。必须注意的是，这个内部时钟的周期可以随着 VDD、温度和制作工艺而改变。看门狗定时器另外两种时钟源是 $f_{SYS}/4$ 时钟和 RTC。值得注意的是配置选项选择 $f_{SYS}/4$ 时钟作为 WDT 的时钟源，当进入暂停模式时，指令时钟会停止且 WDT 将失去其保护目的。对于工作在噪声环境的应用，推荐使用 WDT 振荡器和 RTC 振荡器作为 WDT 的时钟源。

不论选择哪一种时钟源，其时钟信号要通过一个 8 阶计数器除以 256，随后通过一个 7 级的预分频器用以产生更长的溢出周期。分频比由 WDTN3 寄存器的第 0 位 WS0、WS1 和 WS2 位来决定。如果 WS0、WS1 和 WS2 都置 1，分频比例为 1:128 且产生大约 2.1s 的最大溢出周期。

系统在正常运行状态下，WDT 溢出将导致芯片复位，并置位状态标志位 T0。但是在系统处于暂停模式时，如果 WDT 发生溢出，系统将从暂停中唤醒并且它只复位程序计数器 PC 和 SP。有三种方法可以用来清除 WDT 的内容，第一种是外部硬件复位 (\overline{RES} 引脚低电平)，第二种是通过软件指令，而第三种是通过“HALT”指令。使用软件指令有两种方法去清除看门狗寄存器，必须由掩膜选项选择。第一种选择是使用单一“CLR WDT”指令，而第二种是使用“CLR WDT1”和“CLR WDT2”两个指令。对于第一种选择，只要执行“CLR WDT”便清除 WDT。而第二种选择，必须交替执行“CLR WDT1”和“CLR WDT2”两者才能成功的清除 WDT。关于第二种选择要注意的是，如果“CLR WDT1”正被使用来清除 WDT，接着再执行这条指令将是无效的，只有执行“CLR WDT2”指令才能清除 WDT。同样的“CLR WDT2”指令已经执行后，只有接着执行“CLR WDT1”指令才可以清除看门狗定时器。



看门狗定时器

b7				b0			
INTES1	INTES0	—	—	WDTEN3	WDTEN2	WDTEN1	WDTEN0
CTRL1 Register							
Watchdog Timer Function							
1	0	1	0	Disable			
0	1	0	1	Enable - recommended value			
X	X	X	X	Other values - all enable			

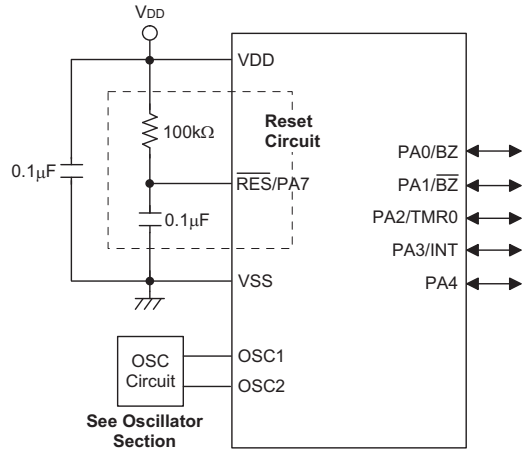
b7					b0			
—	—	—	—	—	WS2	WS1	WS0	
WDTS Register								
WDT prescaler rate select								
					WS2	WS1	WS0	WDT Rate
					0	0	0	1:1
					0	0	1	1:2
					0	1	0	1:4
					0	1	1	1:8
					1	0	0	1:16
					1	0	1	1:32
					1	1	0	1:64
					1	1	1	1:128
Not used								

掩膜选项

掩膜选项在烧写程序时写入芯片。通过HT-IDE 的软件开发环境，使用者在开发过程中可以选择掩膜选项。当掩膜选项烧入单片机后，无法再通过应用程序修改。所有位必须按系统的需要定义，具体内容可参考下表：

编号	选 项
1	看门狗定时器：打开或关闭
2	看门狗定时器时钟源：看门狗振荡器、 $f_{SYS}/4$ 或 RTC 振荡器
3	清除看门狗指令：1 条或 2 条
4	系统振荡器：内部 RC、内部 RC 和外部 RTC、外部晶体或外部 RC
5	LVR 功能：使能或禁止
6	LVR 电压：2.1V, 3.15V 或 4.2V
7	\overline{RES} 或 PA7 选择
8	系统启动延时时间：使能（1024 个时钟）或禁止（2 个时钟）

应用电路



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 μ s 中执行完成，而分支或调用操作则将在 1 μ 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
算术运算			
ADD A,[m]	ACC 与数据存储器相加, 结果放入 ACC	1	Z,C,AC,OV
ADDM A,[m]	ACC 与数据存储器相加, 结果放入数据存储器	1 ^注	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 ^注	Z,C,AC,OV
SUBA,x	ACC 与立即数相减, 结果放入 ACC	1	Z,C,AC,OV
SUBA,[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 ^注	Z,C,AC,OV
SBCA,[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 ^注	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 ^注	C
逻辑运算			
AND A,[m]	ACC 与数据存储器做“与”运算, 结果放入 ACC	1	Z
OR A,[m]	ACC 与数据存储器做“或”运算, 结果放入 ACC	1	Z
XOR A,[m]	ACC 与数据存储器做“异或”运算, 结果放入 ACC	1	Z
ANDM A,[m]	ACC 与数据存储器做“与”运算, 结果放入数据存储器	1 ^注	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 ^注	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 ^注	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 ^注	Z
CPLA [m]	对数据存储器取反, 结果放入 ACC	1	Z
递增和递减			
INCA [m]	递增数据存储器, 结果放入 ACC	1	Z
INC [m]	递增数据存储器, 结果放入数据存储器	1 ^注	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 ^注	Z

助记符	说明	指令周期	影响标志位
移位			
RRA [m]	数据存储器右移一位, 结果放入 ACC	1	无
RR [m]	数据存储器右移一位, 结果放入数据存储器	1 ^注	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 ^注	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 ^注	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 ^注	C
数据传送			
MOV A,[m]	将数据存储器送至 ACC	1	无
MOV [m],A	将 ACC 送至数据存储器	1 ^注	无
MOV A,x	将立即数送至 ACC	1	无
位运算			
CLR [m].i	清除数据存储器的位	1 ^注	无
SET [m].i	置位数据存储器的位	1 ^注	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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	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 ^注	无
CALL addr	子程序调用	2	无
RET	从子程序返回	2	无
RETA,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 ^注	无
TABRDL [m]	读取最后页的 ROM 内容, 并送至数据存储器 and TBLH	2 ^注	无
其它指令			
NOP	空指令	1	无
CLR [m]	清除数据存储器	1 ^注	无
SET [m]	置位数据存储器	1 ^注	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 ^注	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和累加器内容相加，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和累加器内容相加，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 and 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 and 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

CLR	[m]	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
CLR	[m].i	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
CLR	WDT	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
CLR	WDT1	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
CLR	WDT2	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
CPL	[m]	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
CPLA	[m]	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, 结果被存放在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
DAA	[m]	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, 结果存放在到数据存储器。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

DEC	[m]	Decrement data memory 说明: 将指定数据存储器的内容减 1。 运算过程: $[m] \leftarrow [m]-1$ 影响标志位: Z
DECA	[m]	Decrement data memory and place result in the accumulator 说明: 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。 运算过程: $ACC \leftarrow [m]-1$ 影响标志位: Z
HALT		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
INC	[m]	Increment data memory 说明: 将指定数据存储器的内容加 1。 运算过程: $[m] \leftarrow [m]+1$ 影响标志位: Z
INCA	[m]	Increment data memory and place result in the accumulator 说明: 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。 运算过程: $ACC \leftarrow [m]+1$ 影响标志位: Z
JMP	addr	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
MOV	A, [m]	Move data memory to the accumulator 说明: 将指定数据存储器的内容复制到累加器。 运算过程: $ACC \leftarrow [m]$ 影响标志位: 无
MOV	A, x	Move immediate data to the accumulator 说明: 将 8 位立即数载入累加器。 运算过程: $ACC \leftarrow x$ 影响标志位: 无
MOV	[m], A	Move the accumulator data to memory 说明: 将累加器的内容复制到指定的数据存储器。 运算过程: $[m] \leftarrow ACC$ 影响标志位: 无

NOP	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
OR A, [m]	Logical OR accumulator with data memory
说明:	将累加器中的数据 and 指定的数据存储器内容逻辑或, 结果存放到累加器。
运算过程:	$ACC \leftarrow ACC \text{ "OR" } [m]$
影响标志位:	Z
OR A, x	Logical OR immediate data to the accumulator
说明:	将累加器中的数据 and 立即数逻辑或, 结果存放到累加器。
运算过程:	$ACC \leftarrow ACC \text{ "OR" } x$
影响标志位:	Z
ORM A, [m]	Logical OR data memory with accumulator
说明:	将存在指定数据存储器中的数据 and 累加器逻辑或, 结果放到数据存储器。
运算过程:	$[m] \leftarrow ACC \text{ "OR" } [m]$
影响标志位:	Z
RET	Return from subroutine
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。
运算过程:	$PC \leftarrow Stack$
影响标志位:	无
RET A, x	Return and place immediate data in the accumulator
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。
运算过程:	$PC \leftarrow Stack$ $ACC \leftarrow x$
影响标志位:	无
RETI	Return from interrupt
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。
运算过程:	$PC \leftarrow Stack$ $EMI \leftarrow 1$
影响标志位:	无
RL [m]	Rotate data memory left
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。
运算过程:	$[m].(i+1) \leftarrow [m].i \quad (i=0\sim6)$ $[m].0 \leftarrow [m].7$
影响标志位:	无
RLA [m]	Rotate data memory left and place result in the accumulator
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。
运算过程:	$ACC.(i+1) \leftarrow [m].i \quad (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
影响标志位:	无
RLC [m]	Rotate data memory left through carry

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:	$[m].(i+1) \leftarrow [m].i \quad (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位:	C
RLCA [m]	Rotate left through carry and place result in the accumulator
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:	$ACC.(i+1) \leftarrow [m].i \quad (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位:	C
RR [m]	Rotate data memory right
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:	$[m].i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $[m].7 \leftarrow [m].0,$
影响标志位:	无
RRA [m]	Rotate right and place result in the accumulator
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:	$ACC.i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
影响标志位:	无
RRC [m]	Rotate data memory right through carry
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:	$[m].i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位:	C
RRCA [m]	Rotate right through carry and place result in the accumulator
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:	$ACC.i \leftarrow [m].(i+1) \quad (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位:	C
SBC A,[m]	Subtract data memory and carry from the accumulator
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

SBCM	A,[m]	Subtract data memory and carry from the accumulator
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。
运算过程:		$ACC \leftarrow ACC - [m] - \bar{C}$
影响标志位:		OV、Z、AC、C
SDZ	[m]	Skip if decrement data memory is 0
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。
运算过程:		$[m] \leftarrow [m] - 1$, 如果 $[m]=0$ 跳过下一条指令执行
影响标志位:		无
SDZA	[m]	Decrement data memory and place result in ACC, skip if 0
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。
运算过程:		$ACC \leftarrow [m] - 1$, 如果 $ACC=0$ 跳过下一条指令执行
影响标志位:		无
SET	[m]	Set data memory
说明:		将指定数据存储器的每一位设置为 1。
运算过程:		$[m] \leftarrow FFH$
影响标志位:		无
SET	[m].i	Set bit of data memory
说明:		将指定数据存储器的第 i 位设置为 1。
运算过程:		$[m].i \leftarrow 1$
影响标志位:		无
SIZ	[m]	Skip if increment data memory is 0
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。
运算过程:		$[m] \leftarrow [m] + 1$, 如果 $[m]=0$ 跳过下一条指令执行
影响标志位:		无
SIZA	[m]	Increment data memory and place result in ACC, skip if 0
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。
运算过程:		$ACC \leftarrow [m] + 1$, 如果 $ACC=0$ 跳过下一条指令执行
影响标志位:		无

SNZ [m]. i	Skip if bit I of the data memory is not 0
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。
运算过程:	如果[m].i≠0, 跳过下一条指令执行
影响标志位:	无
SUB A, [m]	Subtract data memory from the accumulator
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←ACC-[m]
影响标志位:	OV、Z、AC、C
SUBM A, [m]	Subtract data memory from the accumulator
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]←ACC-[m]
影响标志位:	OV、Z、AC、C
SUB A, x	Subtract immediate data from the accumulator
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←ACC-x
影响标志位:	OV、Z、AC、C
SWAP [m]	Swap nibbles within the data memory
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~[m].0↔[m].7~[m].4
影响标志位:	无
SWAPA [m]	Swap data memory and place result in the accumulator
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~ACC.0← [m].7~[m].4 ACC.7~ACC.4← [m].3~[m].0
影响标志位:	无
SZ [m]	Skip if data memory is 0
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
SZA [m]	Move data memory to ACC, skip if 0
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←[m], 如果[m] = 0, 跳过下一条指令执行
影响标志位:	无

SZ	[m]. i	Skip if bit I of the data memory is 0
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
TABRDC	[m]	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
TABRDL	[m]	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
XOR	A, [m]	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
XORM	A, [m]	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
XOR	A, x	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

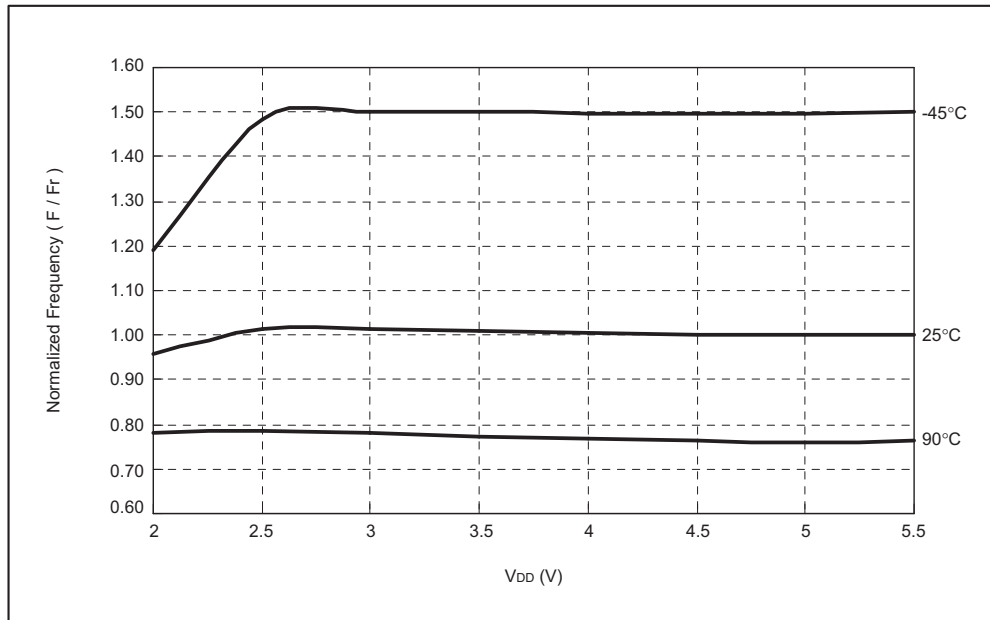
设备特性图表

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

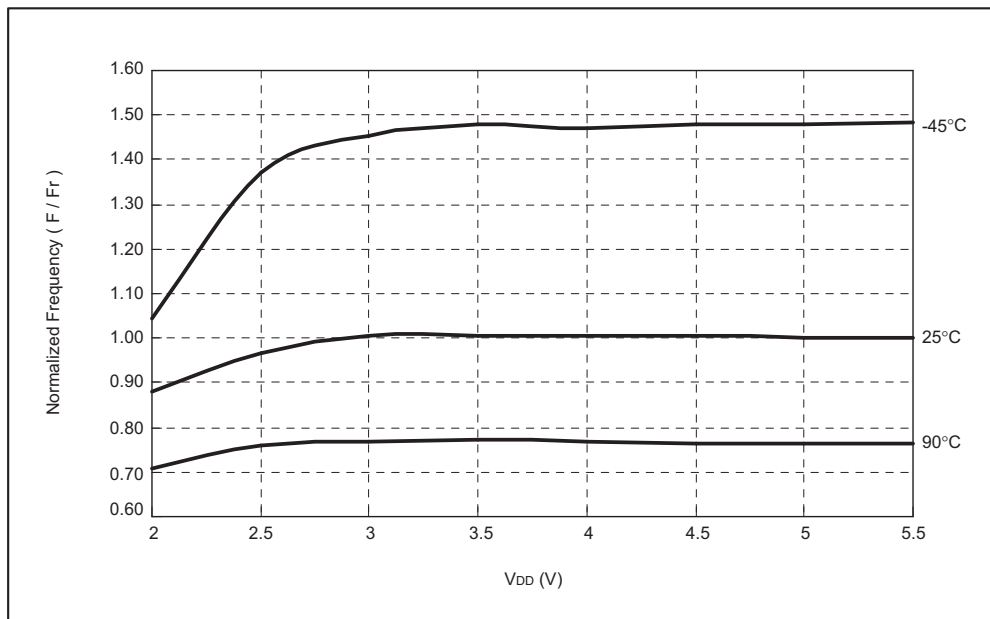
在一些图表中显示的，超出指定的运行范围的数据仅作为参考。设备只有在指定的范围内才能正常的运行。

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

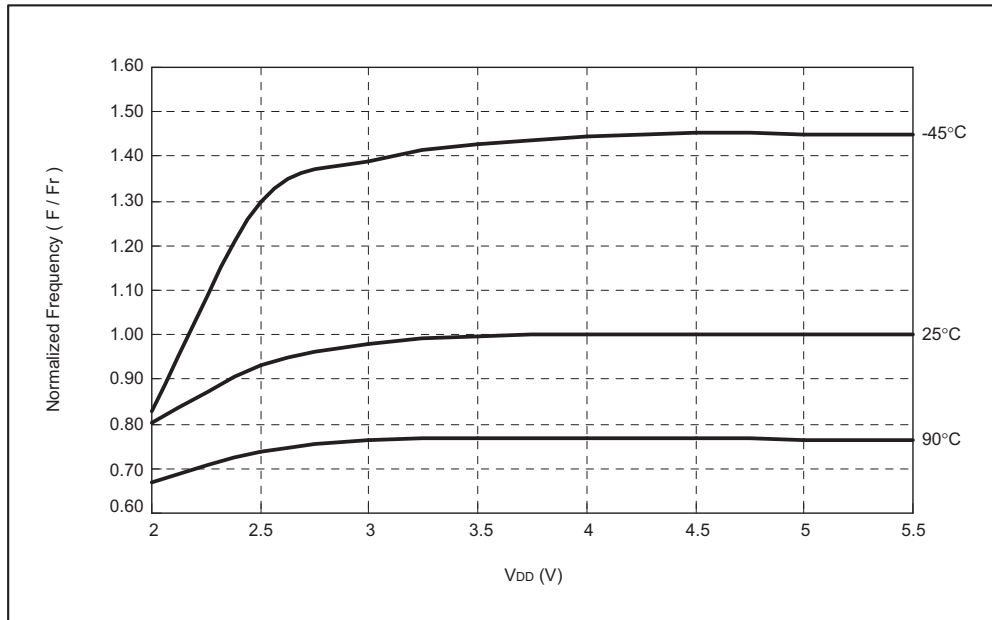
4MHZ IRC 频率 vs.VDD



8MHZ IRC 频率 vs.VDD

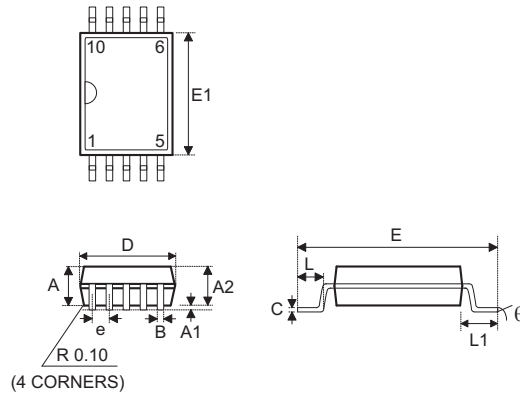


12MHZ IRC 频率 vs.VDD



封装信息:

10-pin MSOP 外形尺寸



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

盛群半导体股份有限公司（总公司）

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

盛群半导体股份有限公司（台北业务处）

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

盛扬半导体有限公司（上海业务处）

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

盛扬半导体有限公司（深圳业务处）

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

盛扬半导体有限公司（北京业务处）

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

盛扬半导体有限公司（成都业务处）

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

Holmate Semiconductor, Inc.（北美业务处）

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

Copyright © 2008 by HOLTEK SEMICONDUCTOR INC.

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