

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 © 2010 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](#)
 - [HT0002S MCU 大型表格的读取](#)
 - [HT0007S MCU 查表指令的使用](#)
 - [HT0019S HT48 MCU 的 WDT 的使用](#)
 - [HT0020S HT48 MCU 定时/计数器的使用](#)
 - [HT0075S MCU 重置电路和振荡电路应用](#)

特性

- 工作电压：
 - $f_{\text{SYS}} = 32768\text{Hz}$: 2.2V~5.5V
 - $f_{\text{SYS}} = 4\text{MHz}$: 2.2V~5.5V
 - $f_{\text{SYS}} = 8\text{MHz}$: 3.3V~5.5V
- 4k×15 程序存储器 ROM
- 192×8 数据存储器 RAM
- 8 个双向输入/输出
- 最多 16×16 LED 驱动输出
- 8 个复用的 LED 输入输出
- 24 个 LED 输出
- 1 个外部中断输入
- 2 个内部中断
- 2 个 8 位可编程定时/计数器
- 32768 实时时钟功能
- 6 层硬件堆栈
- 看门狗定时器 (WDT)
- 低电压复位功能 (LVR)
- PFD/Buzzer 驱动输出
- 外部 RC/XTAL 和 32768Hz 振荡器
- 双时钟系统提供 3 种工作模式
 - 常规模式: RC/XTAL 和 32768Hz 时钟都工作
 - 慢速模式: 只有 32768Hz 时钟工作
 - 暂停模式: 由看门狗时钟溢出周期性唤醒
- HALT 和唤醒功能可降低功耗
- 查表指令, 表格内容字长 15 位
- 63 条功能强大的指令
- 一个指令周期: 4 个系统时钟周期
- 所有指令执行时间皆为 1 或 2 个指令周期
- 位操作指令
- 系统频率为 8MHz 时, 指令周期为 0.5μs
- 44/52-pin QFP 和 44-pin LQFP 封装

概述

HT48R54A 是八位高性能精简指令集单片机, 专为多输入输出控制的产品设计。拥有低功耗、I/O 使用灵活、定时器功能、振荡选择、省电和唤醒功能、看门狗定时器以及低价位等优势, 使此款多功能芯片可以广泛地适用于各种应用, 例如工业控制、消费类产品、子系统控制器等。

The block diagram illustrates the internal architecture of the ATmega16 microcontroller. Key components and their interconnections include:

- System Clock and Timing:** The **RC or Crystal Oscillator** (OSC1, OSC2) provides the main clock to the **Timing Generator**. The **RTC Oscillator** (OSC3, OSC4) provides a lower-frequency clock for the **RTC**. The **Reset & LVR** block is connected to the **Timing Generator** and the **Reset** pin.
- Program Counter and Memory:** The **Program Counter** is connected to the **Address Decoder** and **Program Memory**. The **Stack** and **Stack Pointer** are also connected to the **Program Counter**. The **Interrupt Circuit** and **INTC** are connected to the **Program Counter** and the **Interrupt** pin.
- Instruction Processing:** The **Instruction Register** is connected to the **Instruction Decoder** and the **Timing Generator**. The **Instruction Decoder** outputs to the **Timing Generator** and the **ALU**. The **ALU** is connected to the **Shifter** and the **ACC** (Accumulator). The **Shifter** is connected to the **Memory Pointer** and the **MUX** (Multiplexer).
- Timers and Counters:** The **WDT Counter** (Watchdog Timer Counter) is connected to the **Timing Generator** and the **WDT Oscillator**. The **TMR0** (Timer/Counter 0) and **TMR1** (Timer/Counter 1) are connected to the **Timing Generator** and the **TMR0C** (Timer/Counter 0 Control) and **TMR1C** (Timer/Counter 1 Control) registers. The **Prescaler** is connected to the **TMR0** and the **Timing Generator**.
- Parallel I/O:** The **PAC** (Parallel Access Control) is connected to the **PORT A** and **PORT B**. The **PA** (Parallel Access) is connected to the **PORT A** and **PORT B**. The **PBC** (Parallel Access Control) is connected to the **PORT B** and **PORT C**. The **PC** (Parallel Access) is connected to the **PORT C** and **PORT D**. The **PD** (Parallel Access) is connected to the **PORT D** and **PORT E**. The **PE** (Parallel Access) is connected to the **PORT E** and **PORT F**.
- Serial I/O:** The **USART** (Universal Serial Asynchronous Receiver/Transmitter) is connected to the **Timing Generator** and the **USART** pins. The **SP** (Serial Peripheral Interface) is connected to the **Timing Generator** and the **SP** pins.
- Other Features:** The **RTC** (Real-Time Counter) is connected to the **Timing Generator** and the **RTC** pins. The **EEPROM** (Electrically Erasable Programmable Read-Only Memory) is connected to the **Timing Generator** and the **EEPROM** pins. The **EEPROM** is also connected to the **EEPROM** pins.

The image displays two pin diagrams for the HT48R54A and HT48R54A-52 QFP-A packages. The left diagram is for the HT48R54A - 44 QFP-A package, showing a 44-pin configuration. The right diagram is for the HT48R54A - 52 QFP-A package, showing a 52-pin configuration. Both diagrams include pin numbers, pin names, and a central label indicating the package type.

HT48R54A - 44 QFP-A

Pin 1: PA5
Pin 2: PA6
Pin 3: PA7/PFD
Pin 4: PE0
Pin 5: PE1
Pin 6: PE2
Pin 7: VSSE
Pin 8: PE3
Pin 9: PE4
Pin 10: PE5
Pin 11: PE6
Pin 12: PC7
Pin 13: PC6
Pin 14: PC5
Pin 15: PC4
Pin 16: VDDC
Pin 17: PC2
Pin 18: PC1
Pin 19: VSSD
Pin 20: PD0/INTT
Pin 21: PD0/INTT
Pin 22: PD0/INTT
Pin 23: PB0
Pin 24: PB1
Pin 25: PB2
Pin 26: PB3
Pin 27: VDDDB
Pin 28: VDDDB
Pin 29: PB4
Pin 30: PB5
Pin 31: PB6
Pin 32: PB7
Pin 33: VSS
Pin 34: OSC3
Pin 35: OSC4
Pin 36: VDD
Pin 37: OSC2
Pin 38: OSC1
Pin 39: RES
Pin 40: PA0/BZ
Pin 41: PA1/BZ
Pin 42: PA2/TMR0
Pin 43: PA3/TMR1
Pin 44: PA4

HT48R54A - 52 QFP-A

Pin 1: PA7/PFD
Pin 2: PE0
Pin 3: PE1
Pin 4: PE2
Pin 5: VSSE
Pin 6: PE3
Pin 7: PE4
Pin 8: PE5
Pin 9: PE6
Pin 10: PE7
Pin 11: PD0/INT
Pin 12: PD1
Pin 13: PD2
Pin 14: PC5
Pin 15: PC4
Pin 16: VDDDB
Pin 17: VDDDB
Pin 18: PC3
Pin 19: PC2
Pin 20: PC1
Pin 21: PD7
Pin 22: PD6
Pin 23: PD5
Pin 24: VSSS
Pin 25: VSSS
Pin 26: PD4
Pin 27: PD3
Pin 28: PC7
Pin 29: PC6
Pin 30: PC7
Pin 31: PB2
Pin 32: PB3
Pin 33: VDDDB
Pin 34: VDDDB
Pin 35: PB4
Pin 36: PB5
Pin 37: PB6
Pin 38: PB7
Pin 39: VSS
Pin 40: OSC3
Pin 41: OSC4
Pin 42: VDD
Pin 43: OSC2
Pin 44: OSC1
Pin 45: RES
Pin 46: PA0/BZ
Pin 47: PA1/BZ
Pin 48: PA2/TMR0
Pin 49: PA3/TMR1
Pin 50: PA4
Pin 51: PA5
Pin 52: PA6

引脚说明

引脚名称	输入/输出	掩膜选项	说 明
PA0/BZ PA1/ $\overline{\text{BZ}}$ PA2/TMR0 PA3/TMR1 PA4~PA6 PA7/PFD	输入/输出	上拉电阻 唤醒 Buzzer PFD	8 位双向输入/输出端口，可由软件指令设置为 CMOS 输出或斯密特触发输入，可由掩膜选项设置是否带上拉电阻，每一位可由掩膜选项设置为唤醒输入。 BZ 和 $\overline{\text{BZ}}$ 分别与 PA0 和 PA1 共用引脚。 TMR0 和 TMR1 分别与 PA2 和 PA3 共用引脚。 PFD 和 PA7 共用引脚，并由掩膜选项设置。
PB0~PB7	输入/输出	—	8 位双向输入/输出口，用作输出时为 PMOS 输出。
PC0~PC7	输出	—	PC0~PC7 是 PMOS 输出。
PD0/ $\overline{\text{INT}}$	输入/输出	—	外部中断输入，和 PD0 共用引脚，在上升沿或下降沿触发外部中断。PD0 为 NMOS 输出。
PD1~PD7	输出	—	PD1~PD7 是 NMOS 输出。
PE0~PE7	输出	—	PE0~PE7 是 NMOS 输出。
$\overline{\text{RES}}$	输入	—	斯密特触发复位输入端，低电平有效。
OSC1 OSC2	输入 输出	RC 或晶体	OSC1、OSC2 连接 RC 或晶体(由掩膜选项确定)以产生内部系统时钟。在 RC 振荡方式下，OSC2 是系统时钟四分频的输出口。
OSC3 OSC4	输入 输出	—	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实时系统时钟。
V _{DD}	—	—	正电源。
V _{DDB}	—	—	正电源，供 PB 口使用。
V _{DDC}	—	—	正电源，供 PC 口使用。
V _{SS}	—	—	负电源，接地。
V _{SSD} · V _{SSE}	—	—	负电源，供 PD、PE 口使用，接地。

极限参数

电源供应电压	 V _{SS} - 0.3V 至 V _{SS} + 6.0V	储存温度	 -50℃至 125℃
端口输入电压	 V _{SS} - 0.3V 至 V _{DD} + 0.3V	工作温度	 -40℃至 85℃
端口总灌电流	 300mA	端口总源电流	 -200mA
总功耗	 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。

直流电气特性

Ta=25℃

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	
		—	f _{SYS} =32768Hz	2.2	—	5.5	
I _{DD1}	工作电流 (晶体振荡, RC 振荡)	3V	无负载	—	1.2	2	mA
		5V	f _{SYS} =4MHz	—	2.5	5	
I _{DD2}	工作电流 (晶体振荡, RC 振荡)	5V	无负载 f _{SYS} =8MHz	—	4	8	mA
I _{DD3}	工作电流 (*RTC 振荡打开, 晶体振荡关闭, RC 振荡关闭)	3V	无负载	—	20	40	μ A
		5V	f _{SYS} =32768Hz	—	50	100	
I _{STB1}	静态电流 (看门狗打开, *RTC 振荡打开)	3V	无负载	—	3	5	μ A
		5V	系统暂停模式	—	6	10	
I _{STB2}	静态电流 (看门狗关闭, *RTC 振荡打开)	3V	无负载	—	1	2	μ A
		5V	系统暂停模式	—	2	4	
I _{STB3}	静态电流 (看门狗打开, RTC 振荡关闭)	3V	无负载	—	2	4	μ A
		5V	系统暂停模式	—	4	8	
I _{STB4}	静态电流 (看门狗关闭, RTC 振荡关闭)	3V	无负载	—	—	1	μ A
		5V	系统暂停模式	—	—	2	
V _{IL1}	输入/输出口的低电平输入电压	—	—	0	—	0.3V _{DD}	V
V _{IH1}	输入/输出口的高电平输入电压	—	—	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 _{LVR}	低电压复位	—	2.1V 选项	1.98	2.1	2.22	V
			3.15V 选项	2.98	3.15	3.32	
			4.2V 选项	3.98	4.2	4.42	
I _{OL1}	输入/输出灌电流 PA	3V	V _{OL} =0.1V _{DD}	4	8	—	mA
		5V		10	20	—	
I _{OL2}	输入/输出灌电流 PD、PE	3V	V _{OL} =0.1V _{DD}	8	16	—	mA
		5V		20	40	—	
I _{OH1}	输入/输出源电流 PA	3V	V _{OH} =0.9V _{DD}	-2	-4	—	mA
		5V		-5	-10	—	
I _{OH2}	输入/输出源电流 PB、PC	3V	V _{OH} =0.9V _{DD}	-4	-8	—	mA
		5V		-10	-20	—	
R _{PH}	上拉电阻	3V	—	20	60	100	k Ω
		5V		10	30	50	

注: *RTC 振荡是慢起振荡器

交流电气特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
f _{SYS}	系统时钟 (晶体振荡, RC 振荡)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	
f _{TIMER}	定时器的输入频率	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	
t _{WDTOSC}	看门狗振荡器周期	3V	—	45	90	180	μ s
		5V	—	32	65	130	
t _{FSP1}	f _{SP} 溢出周期 (时钟来源: WDT 振荡)	—	预分频 (f _s /4096)	—	2 ²¹	—	t _{WDTOSC}
t _{FSP2}	f _{SP} 溢出周期 (时钟来源: RTC 振荡)	—	预分频 (f _s /4096)	—	2 ²¹	—	*t _{RTC}
t _{RES}	外部复位低电平脉冲宽度	—	—	1	—	—	μ s
t _{SST}	系统启动延时周期	—	上电或从暂停状态唤醒	—	1024	—	t _{sys}
t _{LVR}	低电压复位	—	—	0.25	1	2	m s
t _{INT}	中断脉冲宽度	—	—	1	—	—	μ s

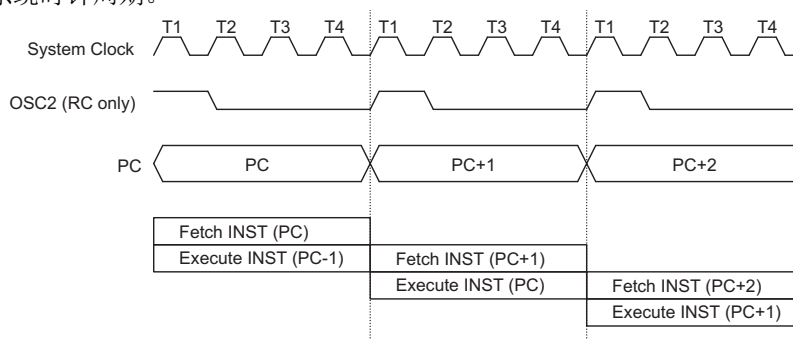
注: t_{sys} = 1/ f_{sys}

*t_{RTC} ≡ 30.5 μ s

系统功能说明

指令执行时序

系统时钟由晶体振荡器、RC 振荡或 RTC 振荡器产生，系统内部将此频率分为四个不重叠的时钟，一个指令周期为四个系统时钟周期。



指令执行时序

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

程序计数器

程序计数器(PC)控制程序存储器 ROM 中指令执行的顺序，它可寻址整个 ROM 的范围。

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

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

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

当遇到控制转移指令时，系统也会插入一个空指令周期。

模式	程序计数器											
	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
初始化复位	0	0	0	0	0	0	0	0	0	0	0	0
外部中断	0	0	0	0	0	0	0	0	0	1	0	0
定时/计数器 0 溢出	0	0	0	0	0	0	0	0	1	0	0	0
定时/计数器 1 溢出	0	0	0	0	0	0	0	0	1	1	0	0
条件跳转	PC+2											
装入 PCL	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
跳转、子程序调用	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
从子程序返回	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

程序计数器

注： *11~*0: 程序指针位
#11~#0: 指令指针位

S11~S0: 堆栈指针位
@7~@0: PCL 字节

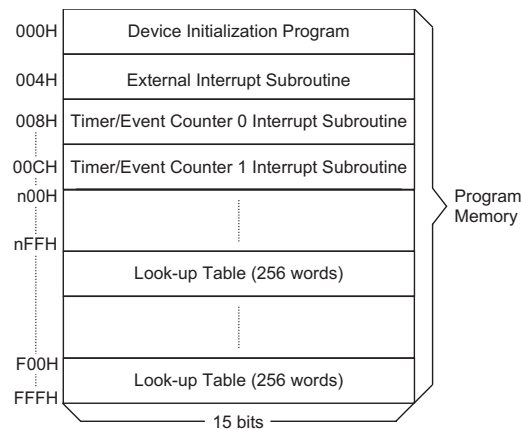
程序存储器

程序存储器用来存储待执行的程序指令，还包括数据、表格、中断入口地址。程序存储器由 4096×15 个位组成，由 PC 和表格指针确定其地址。

以下列出的程序存储器地址是系统专为特殊用途而保留的：

- 地址 000H
该地址保留给程序初始化。当系统复位时，程序会从 000H 地址开始执行。
- 地址 004H
该地址保留给外部中断服务程序。当 $\overline{\text{INT}}$ 引脚有触发信号输入，如果中断允许，且堆栈未满，程序将会跳转到 004H 地址，开始执行中断服务程序。
- 地址 008H
该地址保留给定时/计数器 0 中断服务。如果中断允许，且堆栈未满，一旦定时/计数器 0 发生溢出，就会产生中断，程序将会跳转到 008H 地址，开始执行中断服务程序。
- 地址 00CH
该地址保留给定时/计数器 1 中断服务。如果中断允许，且堆栈未满，一旦定时/计数器 1 发生溢出，就会产生中断，程序将会跳转到 00CH 地址，开始执行中断服务程序。
- 表格区

ROM 空间的任何地址都可作为查表使用。查表指令“TABRDC [m]”（查当前页表格，1 页=256 个字）和“TABRDL [m]”（查最后页表格），会把表格内容低字节传送给[m]，而表格内容高字节传送到 TBLH 寄存器(08H)。只有表格内容的低字节被传送到目标地址中，而高字节被传送到表格内容高字节寄存器 TBLH，并且 TBLH 的最高位始终为“0”。表格内容高字节寄存器 TBLH 是只读寄存器。表格指针(TBLP)是可读/写寄存器(07H)，用来指明表格地址。在查表之前，要先将表格地址写入 TBLP 中。如果主程序和中断服务程序(ISR)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。



Note: n ranges from 0 to 7

程序存储器

指令	表格地址											
	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P11	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

表格区

注：*11~*0：表格地址位
@7~@0：表格指针位

P11~P8：当前程序指针位

堆栈寄存器

堆栈寄存器是特殊的存储器空间，用来保存 PC 的值。系统提供 6 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。堆栈的使用是通过堆栈指针（SP）来指示的，堆栈指针也不能读出或写入。当发生子程序调用或中断响应时，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。

如果堆栈已满，并且发生了非屏蔽中断，那么只有中断请求标志会被记录下来，而中断响应会被禁止，直到堆栈指针发生递减（执行 RET 或 RETI 指令），中断才会被响应。这个功能可以防止堆栈溢出，使得程序员易于使用这种结构。同样，如果堆栈已满，并且发生了子程序调用，那么堆栈会发生溢出，首先进入堆栈的内容将会丢失，只有最后的 6 个返回地址会被保留。

数据存储器

数据存储器分为两个功能区间，特殊功能寄存器和通用数据存储器（192×8），数据存储器单元大多数是可读/写的，但有些为只读。

所有的 RAM 都可以直接执行算术、逻辑、递增、递减和移位等运算。除了一些少数指定的位之外，RAM 的每个位都可以由 SET[m].i 和 CLR[m].i 指令来置位和复位。这些 RAM 地址可以通过 MP 间接调用。

间接寻址寄存器

地址 00H 和 02H 是间接寻址寄存器，但并没有实际的物理地址。任何对[00H]（[02H]）的读/写操作，将对 MP0（MP1）所指定的存储器地址产生相应的读/写操作。直接读取地址 00H（02H），将返回 00H 的结果，而直接写入此地址，则不做任何操作。

间接寻址指针寄存器（MP0 和 MP1）是一个 8 位的寄存器。

累加器

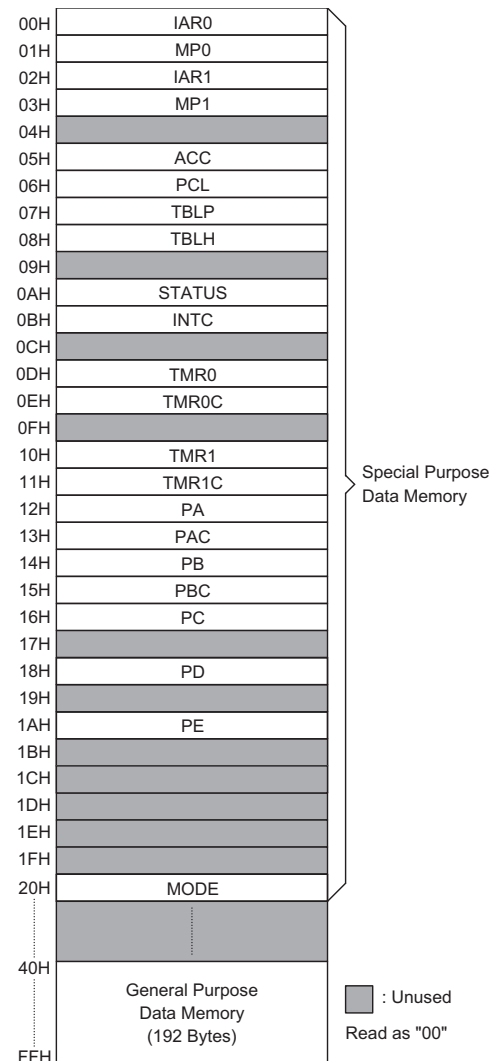
累加器（ACC）与算术逻辑单元（ALU）紧密联系。它对应于 RAM 的地址 05H，并能与立即数进行操作，存储器间的数据传送都必须经过累加器。

算术逻辑单元 — ALU

算术逻辑单元（ALU）执行 8 位的算术及逻辑运算，提供下列功能：

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

ALU 不仅可以储存数据运算的结果，还会改变状态寄存器的值。



数据存储器

状态寄存器 — STATUS

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

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

标志位 Z、OV、AC 和 C 反映的是最近一次操作的状态。

位	符号	功能
0	C	如果在加法运算中结果产生了进位或在减法运算中结果不产生借位，则 C 被置位；反之，C 被清除。它也可被循环移位指令影响。
1	AC	如果在加法运算中低 4 位产生了进位或减法运算中低 4 位不产生借位，则 AC 被置位；反之，AC 被清除。
2	Z	如果算术或逻辑运算的结果为零，则 Z 被置位；反之，Z 被清除。
3	OV	如果运算结果向最高位进位，但最高位并不产生进位输出，则 OV 被置位；反之，OV 被清除。
4	PDF	系统上电或执行“CLR WDT”指令，PDF 被清除； 执行“HALT”指令，PDF 被置位。
5	TO	系统上电、执行“CLR WDT”或“HALT”指令，TO 被清除；WDT 定时溢出，TO 被置位。
6, 7	—	未用，读数为“0”

STATUS(0AH) 寄存器

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

中 断

HT48R54A 提供一个外部中断和内部定时/计数器中断。中断控制寄存器 (INTC; 0BH) 包含了中断控制位和中断请求标志, 中断控制位用来设置中断允许/禁止。

一旦有中断子程序被响应, 所有其它的中断将被禁止 (通过清零 EMI 位)。这种机制能防止中断嵌套, 这时如有其它中断请求发生, 这个中断请求的标志会被记录下来。如果在一个中断服务程序中有另一个中断需要响应的话, 可以设置 EMI 位及 INTC 所对应的位来允许中断嵌套服务。如果堆栈已满, 该中断请求将不会被响应, 即使相关的中断被允许, 也要到堆栈指针发生递减时才会响应。如果需要立即得到中断响应, 必须避免使堆栈饱和。

所有的中断都具有唤醒功能。当一个中断被响应时, 会将程序计数器 (PC) 压入堆栈, 然后转移到中断服务程序的入口。只有程序计数器的内容能压入堆栈。如果寄存器和状态寄存器的内容会被中断服务程序改变, 从而破坏主程序的预定流程, 必须事先将这些数据保存起来。

外部中断是由 $\overline{\text{INT}}$ 脚上的上升沿或下降沿触发的, 相关的中断请求位 (EIF; INTC 的第 4 位) 会被置位。当中断允许, 堆栈也未满, 外部中断触发时, 将会产生地址 04H 的子程序调用。中断请求标志位 (EIF) 和 EMI 位将被清除, 以禁止其他中断的发生。

内部定时/计数器 0 中断是由定时/计数器 0 溢出触发的, 相关的中断请求位 (T0F; INTC 的第 5 位) 会被置位。当中断允许, 堆栈也未满, 定时/计数器 0 中断触发时, 将会产生地址 08H 的子程序调用。中断请求标志位 (T0F) 和 EMI 位将被清除, 以禁止其他中断的发生。

内部定时/计数器 1 中断是由定时/计数器 1 溢出触发的, 相关的中断请求位 (T1F; INTC 的第 6 位) 会被置位。当中断允许, 堆栈也未满, 定时/计数器 1 中断触发时, 将会产生地址 0CH 的子程序调用。中断请求标志位 (T1F) 和 EMI 位将被清除, 以禁止其他中断的发生。

在执行中断子程序期间, 其他的中断响应会被屏蔽, 直到执行 RETI 指令, 或者 EMI 位和相关的中断控制位都被置为 1 (当堆栈未滿时)。若要从中断子程序返回, 只要执行 RET 或 RETI 指令即可。RETI 指令将会自动置位 EMI 来再次允许中断服务, 而 RET 则不会。

如果中断在两个连续的 T2 脉冲的上升沿之间发生, 且中断响应允许, 那么在下一个 T2 脉冲, 该中断会被服务。如果同时发生中断请求, 其优先级如下表示; 这种优先级也可以通过 EMI 位的复位来屏蔽。

中断源	优先级	中断
外部中断	1	04H
定时/计数器 0 中断	2	08H
定时/计数器 1 中断	3	0CH

中断控制寄存器 (INTC) (0BH) 由定时/计数器 0 中断请求标志位 (T0F)、定时/计数器 1 中断请求标志位 (T1F)、外部中断请求标志位 (EIF)、定时/计数器 0 允许位 (ET0I)、定时/计数器 1 允许位 (ET1I), 外部中断允许位 (EEI), 和主中断控制允许位 (EMI) 组成。EMI、EEI, ET0I 和 ET1I 都是用来控制中断的允许/禁止。这些控制位可以用来屏蔽正在进行中断服务程序时发生的其它中断请求。一旦中断请求标志位被置位 (T0F, T1F 和 EIF), 它们将在 INTC 寄存器中被保留下来, 直到相关的中断被响应或由软件指令来清除。

位	符号	功 能
0	EMI	总中断控制位(1=允许, 0=禁止)
1	E EI	外部中断控制位(1=允许, 0=禁止)
2	ET0I	定时/计数器 0 中断控制位(1=允许, 0=禁止)
3	ET1I	定时/计数器 1 中断控制位(1=允许, 0=禁止)
4	EIF	外部中断请求标志位(1=有, 0=无)
5	T0F	定时/计数器 0 中断请求标志位(1=有, 0=无)
6	T1F	定时/计数器 1 中断请求标志位(1=有, 0=无)
7	—	未用, 读出为零

中断控制寄存器—INTC (0BH)

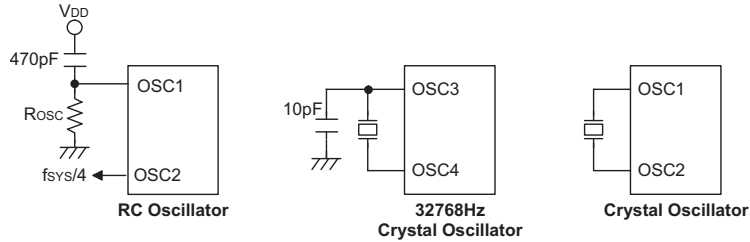
建议不要在中断服务程序中使用“CALL”指令来调用子程序。因为中断随时都可能发生, 而且在一些应用中需要立刻给予响应。如果只剩下一层堆栈, 而中断使能不能被很好地控制, 原先的控制序列很可能因为在中断子程序中执行“CALL”指令而使堆栈溢出, 从而破坏原先的控制序列。

振荡器

HT48R54A 提供三种振荡电路：外部晶体振荡、外部 RC 振荡和 32768Hz 晶体振荡。由掩膜选项设定为外部 RC 振荡和外部晶体振荡。

如果使用外部 RC 型振荡，在 OSC1 和 V_{SS} 之间需要一个外部电阻，其阻值范围为 $130k\Omega \sim 2.5M\Omega$ 。RC 振荡方式是一种低成本方案，但是振荡频率会随着 V_{DD} 、温度和制造漂移而不同。因此，在需要非常精确时基场合，不建议使用 RC 振荡。

另外一个振荡电路是专为实时时钟而设计。这种情况下，只能使用 32768Hz 的晶振。晶振需连接在 OSC3 和 OSC4 之间。



系统振荡器

RTC 振荡器可以通过设置“QOSC”位（MODE 的第 4 位）来控制快速振荡。建议在系统上电时启动快速振荡功能，直到 RTC 振荡器稳定，再过 2 秒钟后关闭，以降低能量消耗。

WDT 振荡器是 IC 内部 RC 型振荡器，不需要任何外部元件。即使系统进入暂停模式时，系统时钟关闭，WDT 振荡器仍然会运作。在 5V 工作电压下，其振荡周期大约为 $65\mu s$ 。WDT 振荡器可在掩膜选项中设置为关闭，以降低系统功耗。

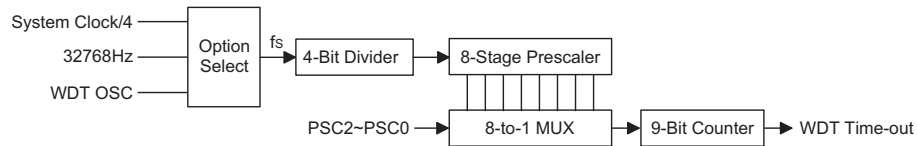
看门狗计时器

WDT 的时钟源有三种：WDT 振荡器、RTC 振荡器或是指令时钟（系统时钟 4 分频），由掩膜选项选择设置。看门狗主要用来避免程序因为运行故障和跳入死循环而导致不可预测的结果。看门狗可用掩膜选项设置为打开或关闭，如果在关闭状态，所有的 WDT 指令都是不起作用。

如果单片机工作在干扰很大的环境中，建议使用片内的 RC 振荡器（WDT OSC）或是 32768Hz 的晶体振荡器（RTC OSC），因为 HALT 模式会使系统时钟停止，看门狗也就失去了保护的功能。

如果选择了内部 WDT 振荡器，这个频率会先除以 16（4 级），再经过 TMR0C 的 8 级预分频器，最后再除以 512（9 级）产生溢出时间。使用 TMR0C 预分频器，可延长 WDT 溢出时间。设置 PSC2, PSC1, PSC0 会产生不同的溢出时间。WDT 振荡器振荡周期为 $65\mu s$ ，但会因为温度、 V_{DD} 以及芯片参数的漂移而变化。WDT 振荡器可以在芯片的任何工作模式下工作。

WDT 的时钟来源也可指令时钟（系统时钟 4 分频）和 32768Hz 实时时钟，其工作方式与 WDT 振荡器一样。



看门狗定时器

在常规和慢速工作模式下，WDT 溢出会使系统复位并置 TO 标志位。但在暂停模式下，WDT 溢出将产生“热复位”，只能使程序计数器 PC 和堆栈指针 SP 复位。

清除 WDT 的值（不包括 4 位分频器和 8 级预分频器），有外部复位（低电平输入到 $\overline{RES}$ 引脚），清除看门狗指令和 HALT 指令三种方法。

清除看门狗指令有“CLR WDT”和“CLR WDT1”、“CLR WDT2”两组指令。这两组指令中，只能选择其中一组，由掩膜选项决定。如果“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。“CLR WDT1”和“CLR WDT2”被选择的情况下（即 CLR WDT 次数为 2），那么二条指令要交替使用才会清除 WDT，否则，WDT 会由于溢出而使系统复位。

工作模式

HT48R54A 支持双时钟系统和三种工作模式。通过软件设置，系统时钟来源可以选择 RC/XTAL 振荡或者 32768Hz 实时时钟，工作模式可以是常规模式、慢速模式或者暂停模式。

位	符号	功 能
0	MODS	系统时钟高速/慢速模式选择位 0: 高速系统时钟 (RC/XTAL) 1: 慢速系统时钟 (32768Hz), RC/XTAL 振荡器停止 注: 当选择 32768Hz 系统时钟时, 只能选择 32768Hz 振荡器作为看门狗时钟源, 否则功能将出现无法预计的结果。
1,2,3	—	未定义, 读出为 “0”
4	QOSC	32768Hz 晶振快速起振 0/1: 快速/慢速
5,6,7	—	未定义, 读出为 “0”

MODE (20H) 寄存器

工作模式	系统时钟	HALT 指令	MODS	RC 振荡器	32768Hz
常规模式	RC/XTAL 振荡器	不执行	0	On	On
慢速模式	32768Hz	不执行	1	Off	On
暂停模式	暂停	执行	×	Off	On

工作模式

暂停模式

暂停模式由 HALT 指令来实现，暂停模式时系统状态如下：

- 系统振荡器停振，但 WDT 振荡器依然工作(如果选择 WDT 振荡器)。
- 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）。
- 所有的输入/输出口都保持其原先状态。
- 置位 PDF 标志，清除 TO 标志。

外部复位、中断或 PA 口下降沿信号或 WDT 溢出均可使系统脱离暂停状态。外部复位能使系统初始化，而 WDT 溢出执行“热复位”。通过检测 TO 和 PDF 标志，即可了解系统复位的原因。PDF 标志位由系统上电复位和执行“CLR WDT”指令清除，而执行“HALT”指令将它置位。如 WDT 产生溢出，使 TO 标志位置位，同时产生唤醒，使得程序计数的 PC 和堆栈指针复位。其他都保持原状态。

PA 口唤醒和中断唤醒可作为正常运行的继续。PA 口的每一位都可以由掩膜选项设置为唤醒功能。如果是由输入/输出口唤醒，程序会从下一条指令开始运行。如果是由中断唤醒，可能会发生两种情况：如果中断禁止或中断允许但堆栈已满，程序将会从下一条指令开始运行；如果中断允许且堆栈未满，则会产生一般的中断响应。如果在进入 HALT 模式之前，中断请求标志位已被置“1”，则中断唤醒功能被禁止。

当发生唤醒，系统需要额外花费 $1024t_{SYS}$ (系统时钟周期)的时间，才能重新正常运行，也就是说，唤醒之后会插入一个等待周期。如果唤醒是由中断产生的话，则实际中断子程序的执行会延迟一个以上的周期。如果唤醒导致下一条指令执行，那么在等待周期执行完成之后，会立即执行该指令。

为减小功耗，在进入暂停模式之前，应小心处理所有的输入/输出口状态。

在 HALT 模式下，如果选用 RTC 振荡器，RTC 振荡器仍然工作。

复位

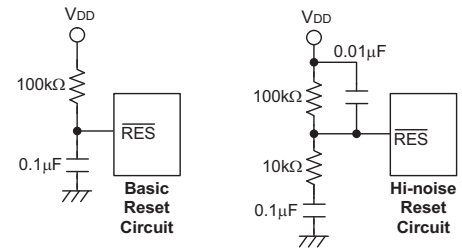
有三种方法可以产生复位：

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

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”，只有程序计数器 PC 和堆栈指针 SP 被复位，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。

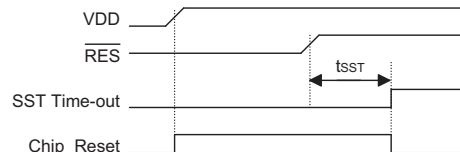
TO	PDF	复位条件
0	0	电源上电时由 $\overline{\text{RES}}$ 发生复位
u	u	正常运作时由 $\overline{\text{RES}}$ 发生复位
0	1	由 $\overline{\text{RES}}$ 唤醒暂停模式
1	u	正常运作时看门狗定时器溢出
1	1	看门狗定时器溢出唤醒暂停模式

注：u 表示不变

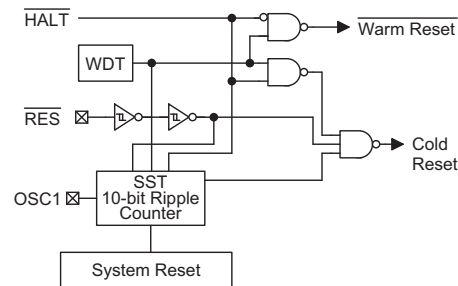


复位电路

注意：“*” 为了避免噪声干扰，连接 $\overline{\text{RES}}$ 引脚的线应尽量短



复位时序



复位电路结构

为了保证系统振荡器起振并稳定运行，系统复位（包括上电复位、WDT 溢出或由 $\overline{\text{RES}}$ 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 被加到复位延时中。任何来自 HALT 的唤醒都将产生 SST 延迟。

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

系统复位时各功能单元的状态如下所示：

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

有关寄存器的状态如下：

寄存器	复位 (上电复位)	WDT 溢出 (正常运行)	RES复位 (正常运行)	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
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	--00 -000	--00 -000	--00 -000	--00 -000	--uu -uuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	0000 1000	0000 1000	0000 1000	0000 1000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PE	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
MODE	---0 ---0	---0 ---0	---0 ---0	---0 ---0	---u ---u

注：“*”表示“热复位”
“u”表示“不变化”
“x”表示“不确定”

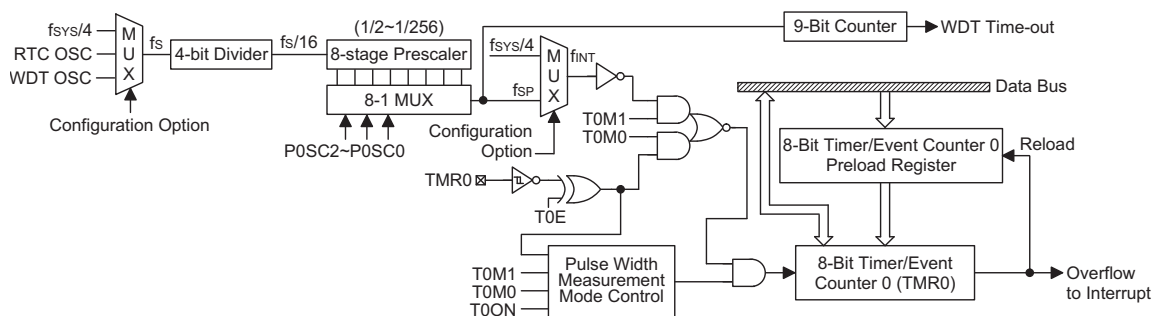
定时/计数器 0

HT48R54A 提供两个定时/计数器。TMR0 是一个 8 位可编程的向上计数的计数器，该定时/计数器的时钟来源可以是外部时钟源、系统时钟四分频或 f_{SP} 。

f_{SP} 时钟源可以在掩膜选项中设定，选择 WDT OSC、外部 32768Hz (f_{RTC}) 或指令时钟（系统时钟 4 分频）中的一种。选择设置后时钟源频率信号首先会被除以 16（4 级），然后被 TMR0C 预分频器（8 级）分频得到 f_{SP} 输出周期。通过使用 TMR0C 预分频器，可以延长溢出时间。设置不同的 PSC2, PSC1 以及 PSC0 值，可以产生不同的 f_{SP} 输出周期。

如果采用内部时钟源，定时/计数器 0 有两个相应的时基，由掩膜选项选择设定内部时钟源为 $f_{SYS}/4$ 或 f_{SP} 。使用外部时钟输入，可以用来计数外部事件、测量时间间隔、脉冲宽度或产生一个精确的时基。使用内部时钟用于产生精确的时基。

有两个与定时/计数器 0 相关的寄存器，分别为 TMR0 ([0DH]) 和 TMR0C ([0EH])。写 TMR0 会将初始值装入到定时/计数器 0 的预置寄存器中，而读 TMR0 则会获得定时/计数器 0 的内容。TMR0C 是定时/计数器 0 的控制寄存器，用来设置某些选项。



定时/计数器 0

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

在外部事件计数或定时器模式中，一旦定时/计数器 0 开始计数，它将会从当前定时/计数器 0 中的数值向上计数到 0FFH。一旦产生溢出，计数器会从定时/计数器 0 预置寄存器重新装载初值，同时产生相应的中断请求状态位 (T0F ; INTC 的第 5 位)。

在脉冲宽度测量模式中，将 T0ON 和 T0E 置为“1”，如果 TMR0 接收到上升沿（如果 T0E 位是“0”，下降沿），就开始计数，直到 TMR0 返回到原来的电平，同时复位 T0ON 位。测量的结果被保留在定时/计数器 0 中，即使电平再发生一次跳变，结果也不会改变。也就是说，一次只能测量一个脉冲宽度。当 T0ON 重新被置位，只要再接收到跳变信号，那么测量过程会再次执行。要注意的是，在这个操作模式中，定时/计数器 0 的启动计数不是根据逻辑电平，而是依据信号的边沿跳变触发。一旦发生计数器溢出，计数器会从定时/计数器 0 的预置寄存器重新装入，并发出中断请求，这种情况与其另外两个模式一样。要启动计数，必须将定时器启动位 (T0ON; TMR0C 的第 4 位) 置 1。在脉宽测量模式中，T0ON 在测量周期结束后自动被清零。但在另外两个模式中，T0ON 只能由指令来复位。定时/计数器 0 的溢出是唤醒的信号之一。不管何种模式，若写 0 到 E0TI 位，即可禁止相应的中断服务。

在定时/计数器 0 为关闭的状态下，写数据到定时/计数器 0 的预置寄存器，同时也会将数据装入定时/计数器 0 中。如果定时/计数器 0 已经开启，写到定时/计数器 0 的数据只会被保留在定时/计数器 0 的预置寄存器中，直到定时/计数器 0 发生计数溢出为止，再由预置寄存器加载新的值。当定时/计数器 0 的数据被读取时，计数会被停止，以防出错。停止计数会导致计数错误，所以程序员必须仔细加以考虑。

TMR0C 的 0~2 位被用于定义定时/计数器 0 的内部时钟源的预分频级数，定义如表所示：

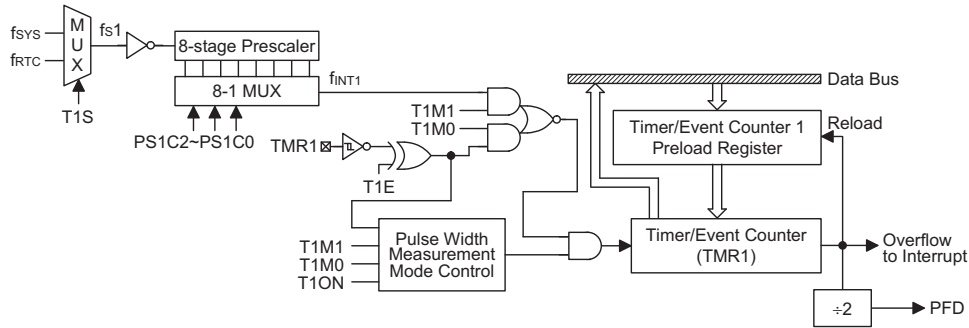
位	符号	功能
0 1 2	P0SC0 P0SC1 P0SC2	定义预分频器级数，P0SC2, P0SC1, P0SC0= 000: fsp= fs/32 001: fsp= fs/64 010: fsp= fs/128 011: fsp= fs/256 100: fsp= fs/512 101: fsp= fs/1024 110: fsp= fs/2048 111: fsp= fs/4096
3	T0E	定义定时/计数器 TMR0 的触发方式 外部事件计数模式 (T0M1, T0M0) = (0, 1) 1: 下降沿触发 0: 上升沿触发 脉冲宽度测量模式 (T0M1, T0M0) = (1, 1) 1: 上升沿开始，下降沿结束 0: 下降沿开始，上升沿结束
4	T0ON	打开/关闭定时/计数器 0 (1=打开, 0=关闭)
5	—	未用，读出为“0”
6 7	T0M0 T0M1	定义工作模式，T0M1, T0M0= 01=外部事件计数模式(外部时钟) 10=定时模式(内部时钟) 11=脉冲宽度测量模式 00=未用

TMR0C (0EH) 寄存器

定时/计数器 1

使用内部时钟源，定时/计数器 1 有两个相应的时基，由 T1S 位选择设定 f_{SYS} 或 f_{RTC} 。使用外部时钟输入，可以用来计数外部事件、测量时间间隔、脉冲宽度，产生一个精确的时基和作为 PFD 信号。

有两个与定时/计数器 1 相关的寄存器，分别为 TMR1 ([10H]) 和 TMR1C ([11H])。在定时/计数器 1 模式 (T1ON=1)，写 TMR1 会将初始值装入到定时/计数器 1 的预置寄存器中，而读 TMR1 会获得定时/计数器 1 的内容。TMR1C 是定时/计数器 1 的控制寄存器，用来定义工作模式、计数功能打开或关闭、计数的触发沿。



定时/计数器 1

T1M0 和 T1M1 用来定义 TMR1 的工作模式。外部事件计数模式用来记录外部事件，它的时钟来自外部 TMR1 引脚。定时器模式是一个常用模式，它的时钟源来自 fINT1 时钟。脉冲宽度测量模式用来计算引脚 TMR1 上的外部脉冲的高/低电平的宽度，计数是基于 fINT1 时钟。

在外部事件计数或定时器模式中，一旦定时/计数器 1 开始计数，它将会从当前定时/计数器 1 中的数值向上计数到 0FFH。一旦产生溢出，计数器会从定时/计数器 1 预置寄存器重新装载初值，同时产生相应的中断请求状态位 (T1F ; INTC 的第 6 位)。

在脉冲宽度测量模式中，将 T1ON 和 T1E 置为“1”，如果 TMR1 接收到上升沿（如果 T1E 位是“0”，下降沿），就开始计数，直到 TMR1 返回到原来的电平，同时复位 T1ON 位。测量的结果被保留在定时/计数器 1 中，即使电平再发生一次跳变，结果也不会改变。也就是说，一次只能测量一个脉冲宽度。当 T1ON 重新被置位，只要再接收到跳变信号，那么测量过程会再次执行。注意的是，在这个操作模式中定时/计数器 1 的启动计数不是根据逻辑电平，而是依据信号的边沿跳变触发。一旦发生计数器溢出，计数器会从定时/计数器 1 的预置寄存器重新装入，并发出中断请求，这种情况与其另外两个模式一样。

要启动计数，必须将定时器启动位 (T1ON; TMR1C 的第 4 位) 置 1。在脉宽测量模式中，T1ON 在测量周期结束后自动被清零。但在另外两个模式中，T1ON 只能由指令来复位。定时/计数器 1 的溢出是唤醒的信号之一。不管何种模式，若写 0 到 E1TI 位，即可禁止相应的中断服务。

在定时/计数器 1 为关闭的状态下，写数据到定时/计数器 1 的预置寄存器，同时也会将数据装入定时/计数器 1 中。如果定时/计数器 1 已经开启，写到定时/计数器 1 的数据只会被保留在定时/计数器 1 的预置寄存器中，直到定时/计数器 1 发生计数溢出为止，再由预置寄存器加载新的值。

当定时/计数器 1 的数据被读取时，计数会被停止，以防出错。停止计数会导致计数错误，所以程序员必须仔细加以考虑。

TMR1C 的 0~2 位被用于定义定时/计数器 1 的内部时钟源的预分频级数，定义如表所示：

位	符号	功能
0 1 2	P1SC0 P1SC1 P1SC2	定义预分频器级数，P1SC2, P1SC1, P1SC0= 000: $f_{INT1} = f_{S1}/2$ 001: $f_{INT1} = f_{S1}/4$ 010: $f_{INT1} = f_{S1}/8$ 011: $f_{INT1} = f_{S1}/16$ 100: $f_{INT1} = f_{S1}/32$ 101: $f_{INT1} = f_{S1}/64$ 110: $f_{INT1} = f_{S1}/128$ 111: $f_{INT1} = f_{S1}/256$
3	T1E	定义定时/计数器 TMR1 的触发方式 外部事件计数模式 (T1M1, T1M0) = (0, 1) 1: 下降沿触发 0: 上升沿触发 脉冲宽度测量模式 (T1M1, T1M0) = (1, 1) 1: 上升沿开始, 下降沿结束 0: 下降沿开始, 上升沿结束
4	T1ON	打开/关闭定时/计数器 1 (1=打开, 0=关闭)
5	T1S	选择 TMR1 时钟源 (0= f_{SYS} ; 1= f_{RTC})
6 7	T1M0 T1M1	定义工作模式, T1M1, T1M0= 01=外部事件计数模式(外部时钟) 10=定时模式(内部时钟) 11=脉冲宽度测量模式 00=未用

TMR1C (11H) 寄存器

输入/输出

此单片机具有 16 个双向输入输出(PA,PB)，8 个 PMOS (PC) 输出和 16 个 NMOS (PD, PE) 输出，标号从 PA 到 PE，分别对应 RAM 的[12H], [14H], [16H], [18H]和[1AH]。PA、PB 的所有的引脚都能被作为输入和输出使用。

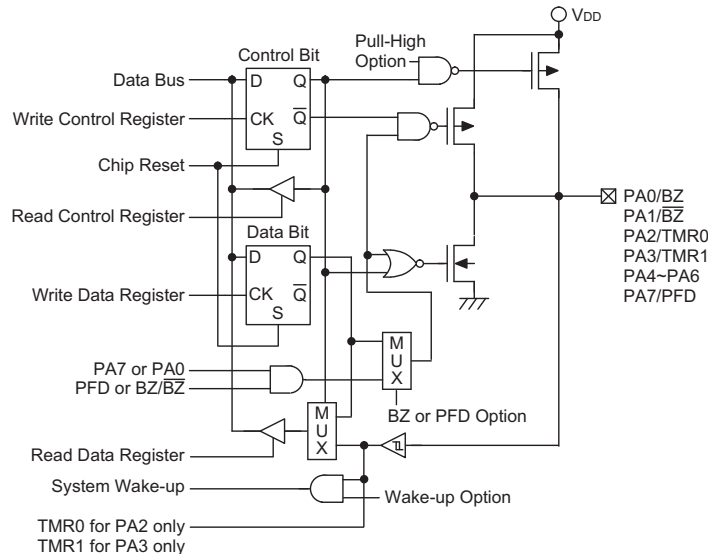
作输入口时，这些端口没有锁存功能，输入数据必须在“MOV A, [m]”(m=12H, 14H) 指令的 T2 上升沿准备好。作输出口时，所有的数据被锁存并保持不变，直到输出锁存器被改写。

PA 和 PB 口都有各自的控制寄存器 (PAC, PBC)，分别对应 RAM 的[13H], [15H]。利用此控制寄存器，CMOS/PMOS 输出或斯密特触发器输入，均可利用软件控制方式加以动态地重新设置。控制寄存器设定哪些引脚为输入，哪些引脚为输出。要实现输入功能，对应的控制寄存器必须设定为“1”；若要设置成输出，则需设定为“0”。

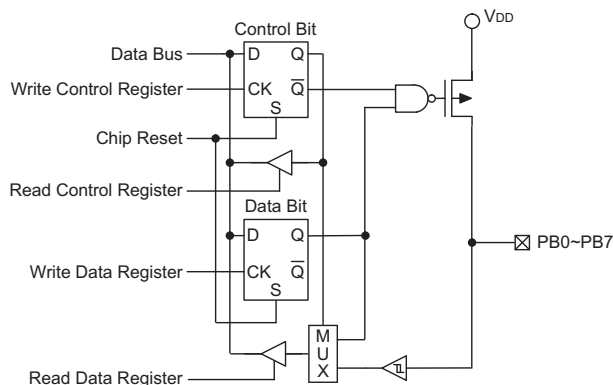
PC 只能作为输出引脚。设置对应输出寄存器为“0”，使其对应的 PMOS 输出晶体管成为高阻态；设置对应寄存器为“1”，则输出高电平。

外部中断引脚 $\overline{\text{INT}}$ 与输出引脚 PD0 共用一个引脚。

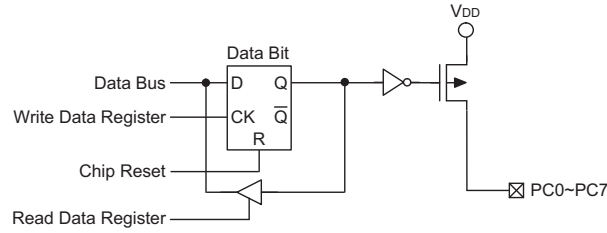
PD 和 PE 只能作为输出引脚，设置对应输出寄存器为“1”，使其对应的 NMOS 输出晶体管成为高阻态；设置对应输出寄存器为“0”，则输出低电平。



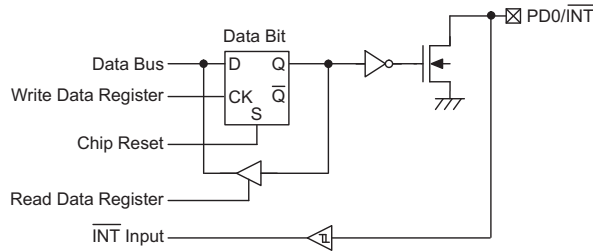
PA 输入/输出口



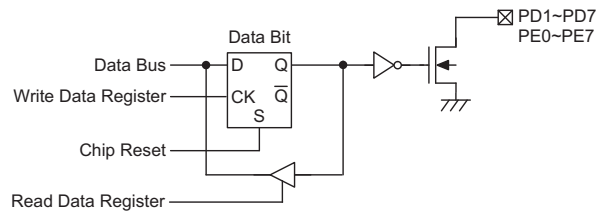
PB 输入/输出口



PC 输出口



PD0 输入/输出口



PD1~PD7, PE 输出口

芯片复位后，这些输入/输出口都保持高电平或浮空状态（取决于上拉电阻的选项）。每一个输入/输出锁存位都能被 SET [m].i 或 CLR [m].i 指令置位或清零（m=12H, 14H, 16H, 18H 或 1AH）。

某些指令需要先输入数据，然后进行输出操作。例如，“SET [m].i”、“CLR [m].i”、“CPL [m]”和“CPLA [m]”指令，读取输入口的状态到 CPU，执行对应位操作指令，然后将数据写回锁存器或累加器。

PA 的每一个口都具有唤醒功能。

PA 的输入/输出口都可以选择上拉电阻。一旦选择上拉电阻，所有的输入/输出口都具有上拉电阻。否则，上拉电阻是高阻状态。必须注意的是没有上拉电阻的输入/输出口，在输入模式时会产生浮空状态。

Buzzer 输出

PA0 和 PA1 分别与 BZ 和 $\overline{\text{BZ}}$ 共用引脚，如果选择 BZ/ $\overline{\text{BZ}}$ 功能，则 PA0/PA1 在输出模式时的 Buzzer 信号将由定时/计数器的溢出信号产生，而在输入模式始终保持其原来的功能。一旦选择 BZ/ $\overline{\text{BZ}}$ 功能，Buzzer 信号的输出只受 PA0 数据寄存器控制。

PA0/PA1 的输入/输出功能如下所示：

PAC 寄存器 PAC0	PAC 寄存器 PAC1	PA 数据寄存器 PA0	PA 数据寄存器 PA1	输出功能
0	0	1	X	PA0=BZ, PA1= $\overline{\text{BZ}}$
0	0	0	X	PA0=0, PA1=0
0	1	1	X	PA0=BZ, PA1=Input
0	1	0	X	PA0=0, PA1=Input
1	0	X	D	PA0= Input , PA1=0
1	1	X	X	PA0= Input, PA1=Input

注：“X”表示任意
“D”表示数据“0”或“1”

PFD 输出

PA7 与 PFD 输出共用引脚，一旦选择 PFD 功能，则 PA7 在输出模式时的 PFD 信号将由定时/计数器 1 的溢出信号产生，而在输入模式始终保持其原来的功能。一旦选择 PFD 功能，PFD 的输出信号只受 PA7 数据寄存器控制。PA7 的输入/输出功能如下所示：

I/O 模式	I/P (正常)	O/P (正常)	I/P (PFD)	O/P (PFD)
PA7	逻辑输入	逻辑输出	逻辑输入	PFD (定时/计数器开启)

注：PFD 的输出频率是定时/计数器 1 溢出频率的 1/2。

PFD 的控制信号和输出频率如下所示：

定时/计数器	定时/计数器 预置值	PA7 数据 寄存器	PA7 引脚 状态	PFD 输出频率
关闭	X	0	0	X
关闭	X	1	U	X
开启	N	0	0	X
开启	N	1	PFD	$f_{\text{TMR1}}/[2 \times (M-N)]$

注：“X”表示未定义
“U”表示未知
“M”为 PFD 等于“65536”
“N”定时/计数器 1 初始值
“ f_{TMR1} ”定时/计数器 1 输入频率

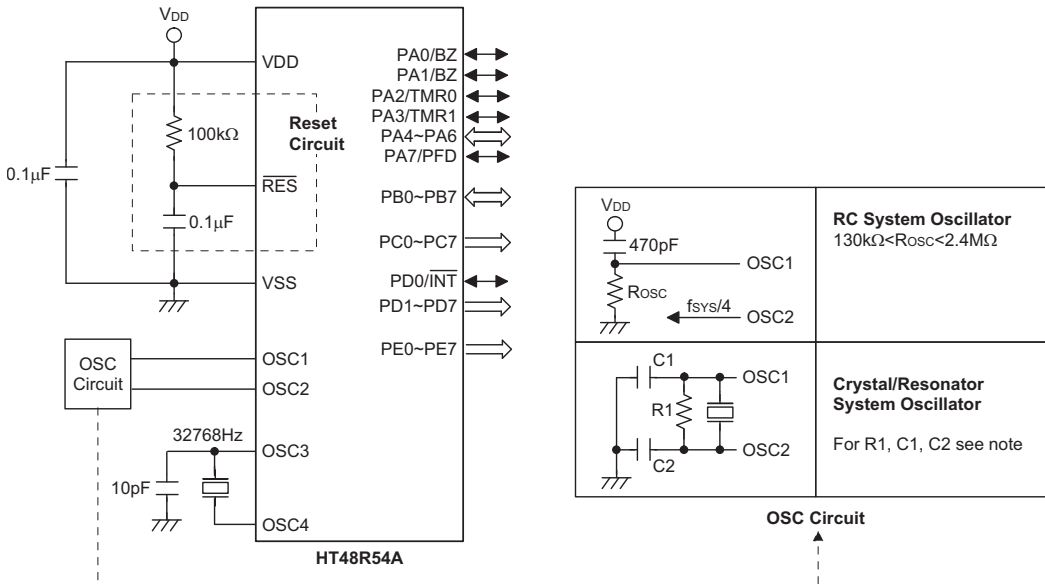
2010-04-28

掩膜选项

下表列出了单片机的所有掩膜选项，所有的掩膜选项必须正确设定，以保证系统功能的实现：

编号	选 项
1	系统振荡：RC 振荡/晶体振荡
2	PA0~PA7 位唤醒：有/无（以位为单位）
3	PA 选择上拉电阻：有/无（以端口为单位）
4	WDT 时钟源：WDT 振荡或 $f_{SYS}/4$ 或 32768Hz 振荡
5	WDT 功能：开/关
6	CLR WDT 指令：1 条/2 条指令
7	定时/计数器 0 时钟来源： $f_{SYS}/4$ 或 f_{SP}
8	LVR 功能：开/关
9	LVR 电压：2.1V，3.15V 或 4.2V
10	Buzzer 功能：BZ 打开，BZ 和 $\overline{BZ}$ 都打开或都关闭
11	Buzzer 频率： $f_s/2$ 、 $f_s/4$ 、 $f_s/8$ 、 $f_s/16$
12	PA7：普通输出/输出或 PFD 输出

应用电路



注： 1、晶体/谐振器

对于晶体振荡器来说，为了保证振荡频率，仅有少许晶体振荡器需要应用 C1 和 C2。而对于谐振器来说，通常情况下必须应用 C1 和 C2 以产生振荡。许多应用场合 R1 也必须添加。当 LVR 禁止，如果系统需要在 VDD 低于单片机的最低工作电压的时候确保关闭振荡，建议添加 R1。C1 和 C2 的数值必须参考晶体/谐振器厂商规格书进行选择。

2、复位电路

复位电路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。

3、当应用电路中复位电路受到噪声干扰及需要振荡电路外部元件的详细资料时，可参考应用范例 HA0075S 获得更多的信息。

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出口的 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
MOV	[m],A	将 ACC 送至数据存储器	1 ^注
MOV	A,x	将立即数送至 ACC	1
位运算			
CLR	[m].i	清除数据存储器的位	1 ^注
SET	[m].i	置位数据存储器的位	1 ^注
转移			
JMP	addr	无条件跳转	2
SZ	[m]	如果数据存储器为零，则跳过下一条指令	1 ^注
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
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
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
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
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
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
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
CALL	addr	子程序调用	2
RET		从子程序返回	2
RET A,x		从子程序返回，并将立即数放入 ACC	2
RETI		从中断返回	2
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
TABRDL	[m]	读取最后页的 ROM 内容，并送至数据存储器 and TBLH	2 ^注
其它指令			
NOP		空指令	1
CLR	[m]	清除数据存储器	1 ^注
SET	[m]	置位数据存储器	1 ^注
CLR	WDT	清除看门狗定时器	1
CLR	WDT1	预清除看门狗定时器	1
CLR	WDT2	预清除看门狗定时器	1
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
SWAPA	[m]	交换数据存储器的高低字节，结果放入 ACC	1
HALT		进入暂停模式	1

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内容和累加器内容相加，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内容和累加器内容相加，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: 将累加器中的数据和指定数据存储器内容做逻辑与，结果存放到累加器。

运算过程: $ACC \leftarrow ACC \text{ "AND" } [m]$

影响标志位: Z

AND A, x Logical AND immediate data to the accumulator

说明: 将累加器中的数据和立即数做逻辑与，结果存放到累加器。

运算过程: $ACC \leftarrow ACC \text{ "AND" } x$

影响标志位: Z

ANDM A, [m] Logical AND data memory with the accumulator

说明: 将指定数据存储器内容和累加器中的数据做逻辑与，结果存放到数据存储器。

运算过程: $[m] \leftarrow ACC \text{ "AND" } [m]$

影响标志位: Z

CALL	addr	Subroutine call
说明:	无条件的调用指定地址的子程序，此时程序计数器先加 1 获得下一个要执行的指令地址并压入堆栈，接着载入指定地址并从新地址执行程序。由于指令需要额外的运算，所以此指令为 2 个周期。	
运算过程:	$Stack \leftarrow Program\ Counter + 1$ $Program\ Counter \leftarrow addr$	
影响标志位:	无	
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; 如果高四位值大于 “9” 或 C=1, 那么 BCD 调整就执行对原值加 “6”。BCD 转换实质上是根据累加器和标志位执行 00H, 06H, 60H 或 66H 的加法运算, 结果存放于数据存储器。只有进位标志位 C 受影响, 用来指示原始 BCD 的和是否大于 100, 并可以进行双精度十进制数的加法运算。
操作:		$[m] \leftarrow ACC+00H$ 或 $[m] \leftarrow ACC+06H$ $[m] \leftarrow ACC+60H$ 或 $[m] \leftarrow ACC+66H$
影响标志位:		C
DEC	[m]	Decrement data memory
说明:		将指定数据寄存器的内容减 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:		将指定数据寄存器的内容减 1, 把结果存放回累加器并保持指定数据寄存器的内容不变。
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:		将指定数据寄存器的内容加 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:		将指定数据寄存器的内容加 1, 结果存放回累加器并保持指定的数据寄存器内容不变。
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:	$PC \leftarrow \text{addr}$
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:	将累加器中的数据和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:	将累加器中的数据和立即数逻辑或，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和累加器逻辑或，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:	$PC \leftarrow \text{Stack}$
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 ←[m].(i+1) (i=0~6) ACC.7 ←[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 ←[m].(i+1) (i=0~6) [m].7 ←C C ← [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 ←[m].(i+1) (i=0~6) ACC.7 ←C C ← [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:	$ACC \leftarrow ACC - [m] - \overline{C}$	
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:	$ACC \leftarrow ACC - [m] - \overline{C}$	
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] ← [m] -1, 如果[m]=0 跳过下一条指令执行	
影响标志位:	无	

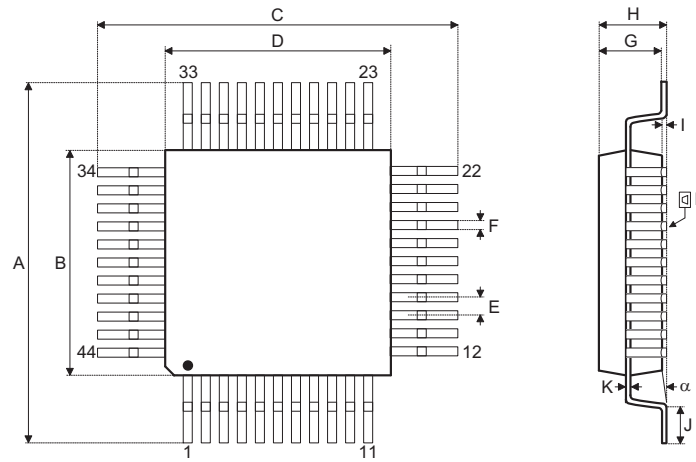
SDZA	[m]	Decrement data memory and place result in ACC,skip if 0
说明:	将指定数据存储器内容减 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果将存放到累加器，但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。	
运算过程:	ACC ← [m]-1，如果 ACC=0 跳过下一条指令执行	
影响标志位:	无	
SET	[m]	Set data memory
说明:	将指定数据存储器的每一位设置为 1。	
运算过程:	[m] ← 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 ←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] ←[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 ←[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寄存器的数据，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中的数据, 结果存放到指定的数据存储器。如果结果为负, C 标志位清除为 0, 反之结果为正或 0, C 标志位设置为 1。	
运算过程:	$[m] \leftarrow ACC - [m]$	
影响标志位:	OV、Z、AC、C	
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:	$ACC \leftarrow ACC - x$	
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:	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$	
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:	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$	
影响标志位:	无	
SZ	[m]	Skip if data memory is 0
说明:	判断指定数据存储器内容是否为 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内容是否为 0, 若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期, 所以此指令为 2 个周期的指令。如果结果不为 0, 则程序继续执行下一条指令。	
运算过程:	$ACC \leftarrow [m]$, 如果 $[m] = 0$, 跳过下一条指令执行	
影响标志位:	无	
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] $\leftarrow$ 程序代码（低字节） TBLH $\leftarrow$ 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] $\leftarrow$ 程序代码（低字节） TBLH $\leftarrow$ 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 $\leftarrow$ 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] $\leftarrow$ 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 $\leftarrow$ ACC “XOR” x
影响标志位:	Z

封装尺寸

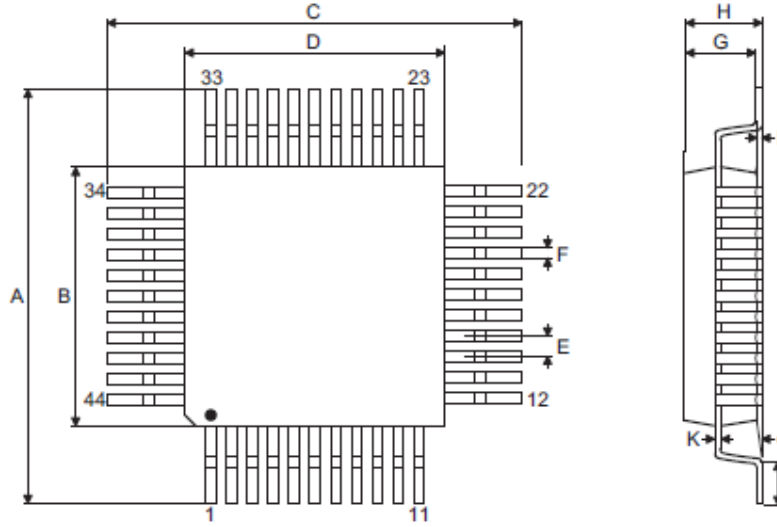
44-pin QFP (10×10) 外形尺寸



符号	尺寸 (单位: inch)		
	最小	一般	最大
A	0.512	—	0.528
B	0.390	—	0.398
C	0.512	—	0.528
D	0.390	—	0.398
E	—	0.031	—
F	—	0.012	—
G	0.075	—	0.087
H	—	—	0.106
I	0.010	—	0.020
J	0.029	—	0.037
K	0.004	—	0.008
L	—	0.004	—
α	0°	—	7°

符号	尺寸 (单位: mm)		
	最小	一般	最大
A	13	—	13.4
B	9.9	—	10.1
C	13	—	13.4
D	9.9	—	10.1
E	—	0.8	—
F	—	0.3	—
G	1.9	—	2.2
H	—	—	2.7
I	0.25	—	0.5
J	0.73	—	0.93
K	0.1	—	0.2
L	—	0.1	—
α	0°	—	7°

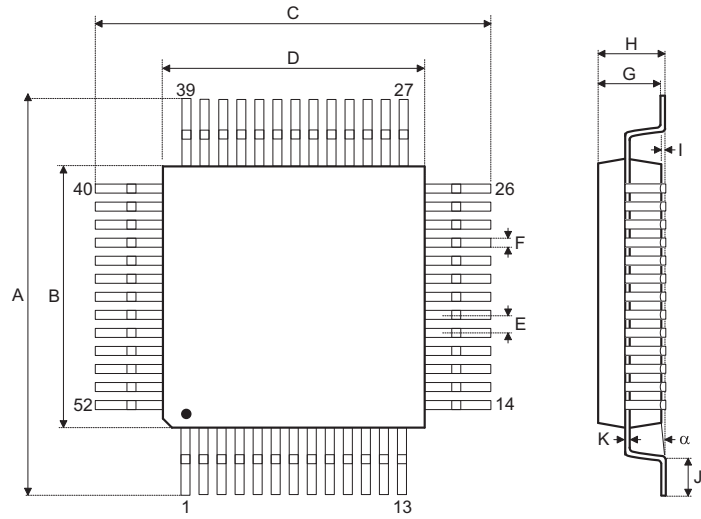
44-pin LQFP (10×10) 外形尺寸



符号	尺寸 (单位: inch)		
	最小	一般	最大
A	0.512	0.520	0.528
B	0.390	0.394	0.398
C	0.512	0.520	0.528
D	0.390	0.394	0.398
E	—	0.031	—
F	—	0.012	—
G	0.053	0.055	0.057
H	—	—	0.063
I	0.004	—	0.010
J	0.041	0.047	0.053
K	0.004	—	0.008
α	0°	—	7°

符号	尺寸 (单位: mm)		
	最小	一般	最大
A	13.00	13.20	13.40
B	9.90	10.00	10.10
C	13.00	13.20	13.40
D	9.90	10.00	10.10
E	—	0.80	—
F	—	0.30	—
G	1.35	1.40	1.45
H	—	—	1.60
I	0.10	—	0.25
J	1.05	1.20	1.35
K	0.10	—	0.25
α	0°	—	7°

52-pin QFP (14×14) 外形尺寸



符号	尺寸 (单位: inch)		
	最小	一般	最大
A	0.681	—	0.689
B	0.547	—	0.555
C	0.681	—	0.689
D	0.547	—	0.555
E	—	0.039	—
F	—	0.016	—
G	0.098	—	0.122
H	—	—	0.134
I	—	0.004	—
J	0.029	—	0.041
K	0.004	—	0.008
L	—	0.004	—
α	0°	—	7°

符号	尺寸 (单位: mm)		
	最小	一般	最大
A	17.3	—	17.5
B	13.9	—	14.1
C	17.3	—	17.5
D	13.9	—	14.1
E	—	1	—
F	—	0.4	—
G	2.5	—	3.1
H	—	—	3.4
I	—	0.1	—
J	0.73	—	1.03
K	0.1	—	0.2
α	0°	—	7°

盛群半导体股份有限公司（总公司）

新竹市科学工业园区研新二路3号

电话: 886-3-563-1999

传真: 886-3-563-1189

网站: www.holtek.com.tw**盛群半导体股份有限公司（台北业务处）**

台北市南港区园区街3之2号4楼之2

电话: 886-2-2655-7070

传真: 886-2-2655-7373

传真: 886-2-2655-7383 (International sales hotline)

盛扬半导体有限公司（上海业务处）

上海市宜山路2016号合川大厦1号楼3楼G室201103

电话: 021-5422-4590

传真: 021-5422-4596

网站: www.holtek.com.cn**盛扬半导体有限公司（深圳业务处）**

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

电话: 0755-8616-9908, 8616-9308

传真: 0755-8616-9533

ISDN: 0755-8615-6181

盛扬半导体有限公司（北京业务处）

北京市西城区宣武门西大街甲129号金隅大厦1721室100031

电话: 010-6641-0030, 6641-7751, 6641-7752

传真: 010-6641-0125

盛扬半导体有限公司（成都业务处）

成都市东大街97号香槟广场C座709室610016

电话: 028-6653-6590

传真: 028-6653-6591

Holmate Semiconductor, Inc.（北美业务处）

46712 Fremont Blvd., Fremont, CA 94538

电话: 510-252-9880

传真: 510-252-9885

网站: www.holmate.com

Copyright © 2010 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>