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

技术相关信息

- 工具信息
- FAQs
- 应用范例
  - HA0003S HT48 & HT46 MCU 与 HT93LC46 EEPROM 的通信
  - HA0004S HT48 & HT46 MCU UART 的软件实现方法
  - HA0084S HT46R52 之应用－镍氢电池充电器展示板

特性

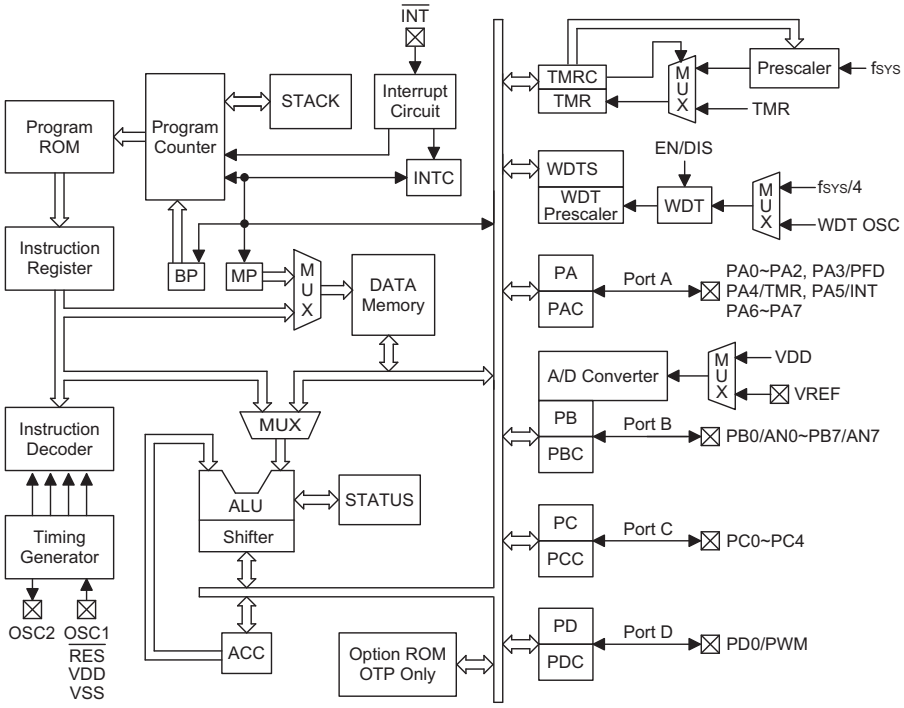
- 低功耗的全静态 CMOS 设计
- 工作电压：
- f_{SYS} =4MHz：2.2V~5.5V
- f_{SYS} =8MHz：3.3V~5.5V
- 程序存储器：
  2K×15 OTP（HT46R53A）
  4K×15 OTP（HT46R54A）
- 数据存储器：
  192×8 RAM（HT46R53A）
  280×8 RAM（HT46R54A）
- A/D 转换器：12bits×8 通道
  具有外部的 A/D 转换器的参考电压输入引脚
- 有 22 个双向输入/输出口
- 1 个与输入/输出口共用引脚的外部中断输入
- 8 位可编程定时/计数器 ，具有溢出中断和 7 级预分频器
- 内置晶体和 RC 振荡电路
- 6 层硬件堆栈
- 看门狗定时器
- 低电压复位功能
- HALT 和唤醒功能可降低功耗
- 在 V_{DD}=5V，系统频率为 8MHz 时，指令周期为 0.5μs
- 1 通道 8 位的 PWM 输出，与输入/输出口共用引脚
- PFD 功能
- 位操作指令
- 查表指令
- 63 条指令
- 指令执行时间为 1 或 2 个指令周期
- 28-pinSKDIP/SOP 封装

概述

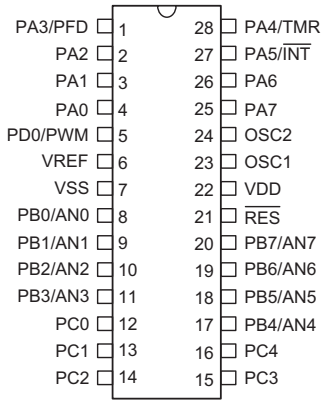
HT46R53A/ HT46R54A 是 8 位高性能精简指令集单片机，专门为需要 A/D 转换的产品而设计，例如传感器信号输入。

低功耗、I/O 使用灵活、计数器、振荡类型选择、多通道 A/D 转换、脉宽调制功能、暂停和唤醒功能，看门狗定时器，并且价格便宜，使这款单片机更具多功能性，可以广泛应用于传感器的 A/D 转换、充电器、电机控制、工业控制、消费类产品、子系统控制器等应用中。

方框图



引脚图



HT46R53A/HT4R54A
—28 SKDIP-A/SOP-A

引脚说明

引脚名称	输入/输出	掩膜选项	功能说明
PA0~PA2 PA3/PFD PA4/TMR PA5/$\overline{\text{INT}}$ PA6~PA7	输入/输出	上拉电阻 唤醒功能 PA3 或 PFD	8 位双向输入/输出口。每一位可由掩膜选项设置为唤醒输入。可由软件设置为 CMOS 输出或者斯密特触发输入。掩膜选项可以按位选择引脚带或不带上拉电阻。 PFD、TMR 和 $\overline{\text{INT}}$ 分别与 PA3、PA4 和 PA5 共用引脚。
PB0/AN0~ PB7/AN7	输入/输出	上拉电阻	8 位双向输入/输出口。可由软件设置为 CMOS 输出、带或不带上拉电阻(由上拉电阻选项决定：位选择)的斯密特触发输入。PB 口与 A/D 输入共用引脚。 一旦 PB 有一个口做为 A/D 输入(由软件设置)，则其输入/输出功能和上拉电阻会自动失效。
PC0~PC4	输入/输出	上拉电阻	5 位双向输入/输出口。可由软件设置为 CMOS 输出、带或不带上拉电阻(由上拉电阻选项决定：端口选择)的斯密特触发输入。
PD0/PWM	输入/输出	上拉电阻 PWM	1 位双向输入/输出口。可由软件设置为 CMOS 输出、带或不带上拉电阻(由上拉电阻选项决定：端口选择)的斯密特触发输入。 PWM 输出与 PD0 共用引脚。
OSC1 OSC2	输入 输出	晶体或 RC	OSC1 和 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。
$\overline{\text{RES}}$	输入	—	斯密特触发复位输入，低电平有效。
VDD	—	—	正电源。
VSS	—	—	负电源，接地。
VREF	输入	—	A/D 转换器参考电压输入引脚。将期望的 A/D 参考电压连接至此引脚。

极限参数

电源供应电压……V_{SS}-0.3V~V_{SS}+6.0V
端口输入电压……V_{SS}-0.3V~V_{DD}+0.3V
I_{OL} 总电流…… ……. …….150mA
总消耗电流……. ……. …….500mW

储存温度…….-50℃~125℃
工作温度…….-40℃~ 85℃
I_{OH} 总电流……. …. -100mA

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	V
I_{DD1}	工作电流(晶体振荡)	3V	无负载，f_{SYS}=4MHz ADC 关闭	—	0.6	1.5	mA
		5V		—	2	4	mA
I_{DD2}	工作电流(RC 振荡)	3V	无负载，f_{SYS}=4MHz ADC 关闭	—	0.8	1.5	mA
		5V		—	2.5	4	mA
I_{DD3}	工作电流	5V	无负载，f_{SYS}=8MHz ADC 关闭	—	4	8	mA
I_{STB1}	静态电流(看门狗打开)	3V	无负载，系统 HALT	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	静态电流 (看门狗和 A/D 关闭)	3V	无负载，系统 HALT	—	—	1	μA
		5V		—	—	2	μA
V_{IL1}	输入/输出口、TMR 和 $\overline{INT}$ 的低电平输入电压	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	输入/输出口、TMR 和 $\overline{INT}$ 的高电平输入电压	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	低电平输入电压($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	高电平输入电压($\overline{RES}$)	—	—	$0.9\ V_{DD}$	—	V_{DD}	V
V_{LVR}	低电压复位	—	掩膜选项：3V	2.7	3	3.3	V
I_{OL}	输入/输出口灌电流	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V		10	20	—	mA
I_{OH}	输入/输出口源电流	3V	V_{OH}=0.9V_{DD}	-2	-4	—	mA
		5V		-5	-10	—	mA
R_{PH}	上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
V_{AD}	A/D 输入电压	—	—	0	—	V_{REF}	V
V_{REF}	ADC 输入参考电压范围	—	—	1.2	—	V_{DD}	V
DNL	ADC 非线性微分	—	—	—	—	±2	LSB
INL	ADC 非线性积分	—	—	—	±2.5	±4	LSB
RESOLU	分辨率	—	—	—	—	12	Bits
I_{ADC}	打开 ADC 增加的功耗	3V	—	—	0.5	1	mA
		5V		—	1.5	3	mA

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）	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f_{TIMER}	定时器输入频率(TMR)	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
t_{WDTOSC}	看门狗振荡器	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	外部复位低电平脉宽	—	—	1	—	—	μs
t_{SST}	系统启动延迟时间	—	从 HALT 状态唤醒	—	1024	—	t_{SYS}
t_{INT}	中断脉冲宽度	—	—	1	—	—	μs
t_{AD}	A/D 时钟周期	—	—	1	—	—	μs
t_{ADC}	A/D 转换时间	—	—	—	80	—	t_{AD}
t_{ADCS}	A/D 采样时间	—	—	—	32	—	t_{AD}

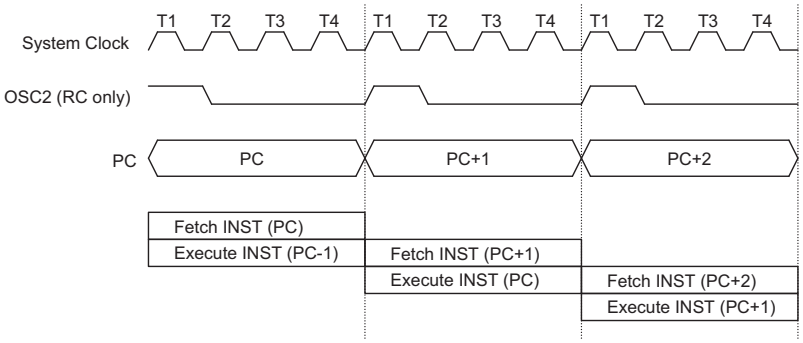
注：$t_{SYS}=1/f_{SYS}$

系统功能说明

指令执行时序

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

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



指令执行时序

程序计数器 — PC

HT46R53A 的程序计数器(PC)的宽度为 11 位，用来控制程序存储器 ROM 中指令执行的顺序，它可寻址 2048 个地址的范围。

HT46R54A 的程序计数器(PC)的宽度为 12 位，用来控制程序存储器 ROM 中指令执行的顺序，它可寻址 4096 个地址的范围。

模式	程序计数器											
	***b11**	***b10**	***b10**	***b8**	***b7**	***b6**	***b5**	***b4**	***b3**	***b2**	***b1**	***b0**
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/计数器中断	0	0	0	0	0	0	0	0	1	0	0	0
A/D 转换中断	0	0	0	0	0	0	0	0	1	1	0	0
条件跳跃	Program Counter+2											
装载 PCL	PC11	PC10	PC9	PC8	@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： *b11 ~ *b0 ： 程序计数器位
S11 ~ S0 ： 堆栈寄存器位
#11 ~ #0 ： 指令代码位
@7 ~ @0 ： PCL 位，PC11~ PC8：PC 计数器原始值，保持不变
对于 HT46R53A 来说，程序计数器有 11 位宽（b0 ~ b10），表格中的 b11 一栏是没有用的。
对于 HT46R54A 来说，程序计数器有 12 位宽，例如 b0 ~ b11

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

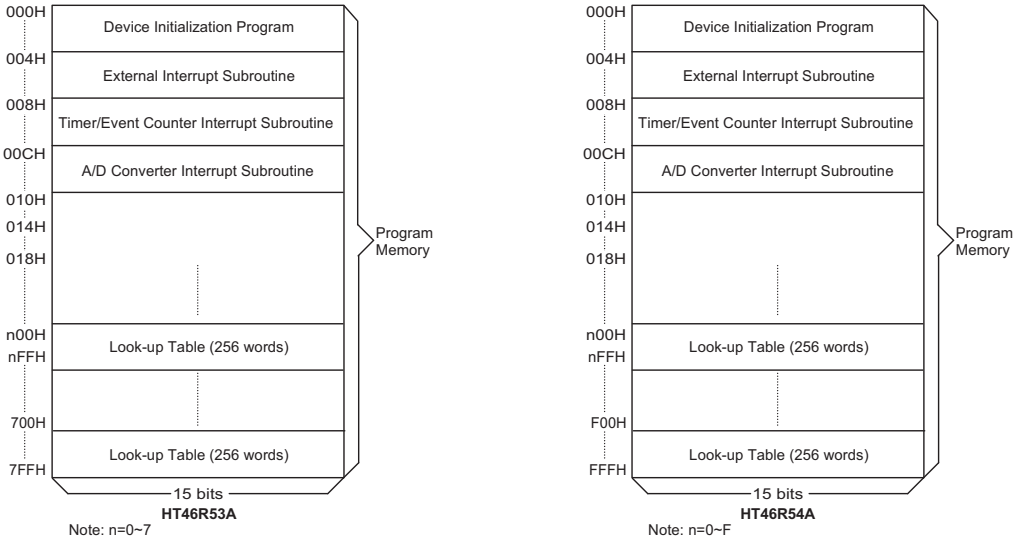
当遇到条件跳跃指令且符合条件时，当前指令执行过程中读取的下一条指令会被丢弃，取而代之的是一个空指令周期，随后才能取得正确的指令。反之，就会顺序执行下一条指令。

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

当遇到控制转移指令时，系统也会插入一个空指令周期。

程序存储器 — EPROM

程序存储器(EPROM)用来存放要执行的指令代码，以及一些数据、表格和中断入口。程序存储器有 2048×15 位（HT46R53A），或者 4096×15 位（HT46R54A）程序存储器空间可以用程序计数器或表格指针进行寻址。



程序存储器

以下列出的程序存储器地址是系统专为特殊用途而保留的：

- 地址 000H
  该地址为程序初始化保留。系统复位后，程序总是从 000H 开始执行。
- 地址 004H
  该地址为外部中断服务程序保留。当 $\overline{\text{INT}}$ 引脚有触发信号输入，如果中断允许且堆栈未满，则程序会跳转到 004H 地址开始执行。
- 地址 008H
  该地址为定时/计数器中断服务程序保留。当定时/计数器溢出，如果中断允许且堆栈未满，则程序会跳转到 008H 地址开始执行。
- 地址 00CH
  该地址为 A/D 转换中断服务程序保留。当 A/D 转换完成，如果中断允许且堆栈未满，则程序会跳转到 00CH 地址开始执行。
- 表格区
  ROM 空间的任何地址都可做为查表使用。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。

指令	表格区											
	b11	**b10**	**b9**	**b8**	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
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
P11~P8：当前程序指针位
@7~@0：表格指针位
对于 HT46R53A 来说，表格地址有 11 位宽（b0～b10），表格中的 b11 一栏是没有用的。
对于 HT46R54A 来说，表格地址有 12 位宽，从 b0～b11。

堆栈寄存器 — STACK

堆栈寄存器是特殊的存储器空间，用来保存 Program Counter 的值。系统有 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)来实现的，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(Program Counter)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。

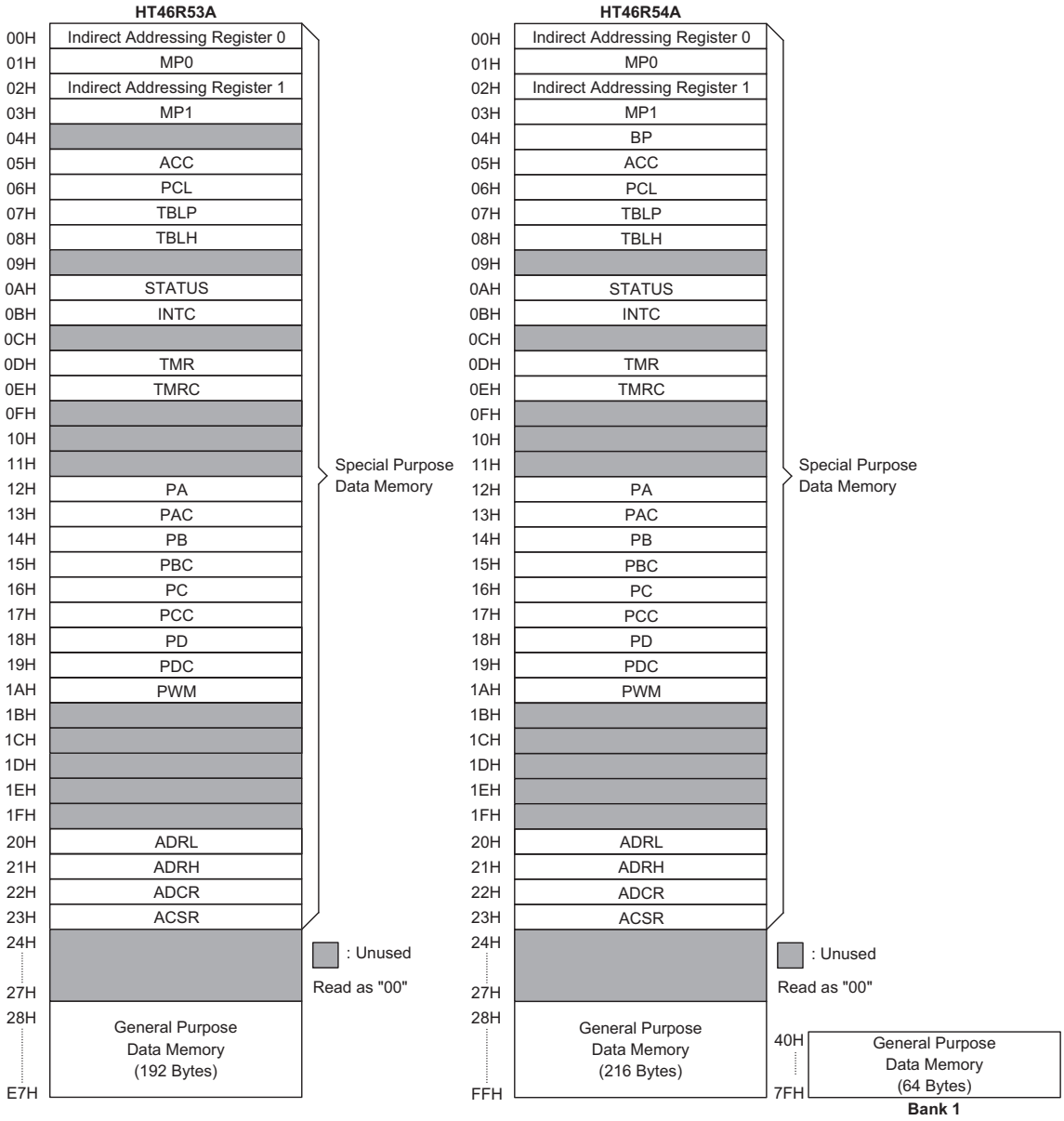
如果堆栈已满，并且发生了不可屏蔽的中断，那么只有中断请求标志会被记录下来，而中断响应会被抑制，直到堆栈指针(执行 RET 或 RETI 指令)发生递减，中断才会被响应。这个功能可以防止堆栈溢出，使得程序员易于使用这种结构。同样，如果堆栈已满，并且发生了子程序调用，那么堆栈会发生溢出，首先进入堆栈的内容将会丢失，只有最后的 6 个返回地址会被保留。

数据存储器 — RAM

数据存储器(RAM)由 217×8 位(HT46R53A)或 306×8 位(HT46R54A)组成，分为两个功能区间：特殊功能寄存器(HT46R53A：25×8 位，HT46R54A：26×8 位)和通用数据存储器(HT46R53A：192×8 位，HT46R54A：Bank0 216×8 位，Bank1 64×8 位)，数据存储器单元大多数是可读/写的，但有些是只读的。

对于 HT46R53A，在 28H 之前的未用空间保留给系统以后扩展使用，读取这些地址的返回值为“00H”。通用数据寄存器地址从 28H 到 E7H，用来存储数据和控制信息。所有的数据存储器单元都能直接执行算术、逻辑、递增、递减和循环操作。除了一些特殊位外，数据存储器的每一位都可由“SET[m].i”置位或由“CLR[m].i”复位。而且都可以通过间接寻址指针(MP0；01H/MP1；03H)进行间接寻址。

对于 HT46R54A，在 28H 之前的未用空间保留给系统以后扩展使用，读取这些地址的返回值为“00H”。在每一个存储区段(BANK)的 40H 之前的地址都是重叠的。通用数据寄存器地址从 28H 到 FFH(Bank0;BP=00H)和 40H 到 7FH(Bank1;BP=01H)，用来存储数据和控制信息。所有的数据存储器单元都能直接执行算术、逻辑、递增、递减和循环操作。除了一些特殊位外，数据存储器的每一位都可由“SET[m].i”置位或由“CLR[m].i”复位。当 BP 值设为“01H”，Bank1 中的存储器可以通过间接寻址指针(MP1；03H)进行间接寻址。直接寻址将访问 Bank0 中的存储器。



数据存储器

间接寻址寄存器

地址 00H 和 02H 是间接寻址寄存器，并无实际的物理区存在。任何对[00H]或[02H]的读/写操作，都是访问由 MP0(01H)MP1(03H)或所指向的 RAM 单元。间接读取地址 00H 或 02H 得到的值为 00H，间接写入此地址，不会产生任何操作。存储器指针 MP0 和 MP1(7 位和 8 位)通过掩膜选项选择。若选择 7 位寄存器 ，那么存储器指针的第七位将不会执行。但必须注意的是，当单片机的存储器指针被读取时，第七位的结果是"1"。同时应注意到，如果 MP0 和 MP1 被选择为 7 位寄存器，那么数据存储器 80H 之后的地址，无法通过 MP0 和 MP1 存取。

累加器

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

算术逻辑单元 — ALU

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

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

ALU 不仅可以储存数据运算的结果，还会改变状态寄存器的值。

状态寄存器 — STATUS

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

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

标志位 Z、OV、AC 和 C 反映的是最近一次操作的状态。在进入中断程序或子程序调用时，状态寄存器不会被自动压入堆栈。如果状态寄存器的内容是重要的，而且子程序会影响状态寄存器的内容，那么程序员必须事先将 STATUS 的值保存好。

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，反之亦然；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，读出为“0”

STATUS(0AH) 寄存器

中断

系统提供一个外部中断、一个内部定时/计数器中断和一个 A/D 转换中断。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。

位	符号	功　　能
0	EMI	总中断控制位(1=允许；0=禁止)
1	EEI	外部中断控制位(1=允许；0=禁止)
2	ETI	定时/计数器中断控制位(1=允许；0=禁止)
3	EADI	A/D 转换中断控制位(1=允许；0=禁止)
4	EIF	外部中断请求标志(1=有；0=无)
5	TF	定时/计数器中断请求标志(1=有；0=无)
6	ADF	A/D 转换中断请求标志(1=有；0=无)
7	—	只作内部测试用 使用时必须写入‘0’；否则会发生不可预知的错误

INTC（0BH）寄存器

只要有中断子程序被服务，其余的中断全部都被自动禁止(通过清除 EMI 位)，这种做法的目的在于防止中断嵌套。这时如果有其它中断发生，只有中断请求标志会被记录下来。如果在中断服务程序中有另一个中断需要响应，程序员可以置位 EMI 和 INTC 所对应的位，以便进行中断嵌套。如果堆栈已满，则中断并不会被响应，一直到堆栈指针(SP)发生递减后才会响应。如果需要中断立即得到响应，应避免堆栈饱和。

所有的中断都具有唤醒能力。当有中断被服务，系统会将程序计数器值压入堆栈，然后再跳转至中断服务程序的入口。但这时只有程序计数器的内容被压入堆栈，如果其它寄存器和状态寄存器的内容会被中断程序改变，从而会破坏主程序的控制流程的话，程序员应该事先将这些数据保存起来。

外部中断是由$\overline{\text{INT}}$引脚下降沿信号触发的，其中断请求标志位(EIF；INTC 的第 4 位)会被置位。如果中断允许，且堆栈未满，当发生外部中断时，会产生地址 04H 的子程序调用；而中断请求标志 EIF 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

内部定时/计数器中断是由定时/计数器溢出触发的，其中断请求标志(TF；INTC 的第 5 位)会被置位。如果中断允许，且堆栈未满，当发生定时/计数器中断时，会产生地址 08H 的子程序调用；而中断请求标志 TF 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

A/D 转换中断是由 A/D 转换完成触发的，其中断请求标志(ADF；INTC 的第 6 位)会被置位。如果中断允许，且堆栈未满，当发生 A/D 转换中断时，会产生地址 0CH 的子程序调用；而中断请求标志位 ADF 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

在执行中断子程序期间，其它的中断请求会被屏蔽，直到执行 RETI 指令或 EMI 和相关中断控制位被置位(当然，此时堆栈未满)。如果要从中断子程序返回，只要执行 RET 或 RETI 指令即可。其中，RETI 指令会自动置位 EMI，以允许中断服务，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，那么在下两个 T2 脉冲之间，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；也可以通过设定各中断相关的控制位来改变优先级。

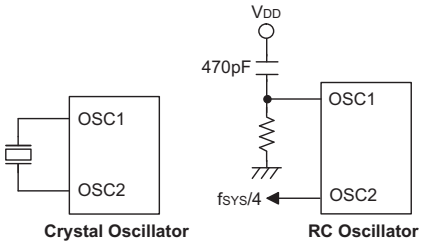
中断源	优先级	中断向量
外部中断	1	04H
定时/计数器中断	2	08H
A/D 转换中断	3	0CH

中断控制寄存器(INTC)，由定时/计数器中断请求标志(TF)、外部中断请求标志(EIF)、A/D 转换中断请求标志(ADF)、定时/计数器中断允许(ETI)、外部中断允许(EEI)、A/D 转换中断允许(EADI)和总中断允许(EMI)组成，其对应于数据存储器地址 0BH。EMI、EEI、ETI 和 EADI 用来控制中断的允许/禁止状态的。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(TF、EIF、ADF)被置位，会一直保留在 INTC 寄存器中，直到中断被响应或用软件指令清除为止。

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

振荡电路

系统有两种振荡方式，外部 RC 振荡和外部晶体振荡，可以通过掩膜选项设定，不管选用哪一种振荡方式，其信号都可以做为系统时钟。HALT 模式会停止系统振荡器，并忽视任何外部信号以降低功耗。



系统振荡器

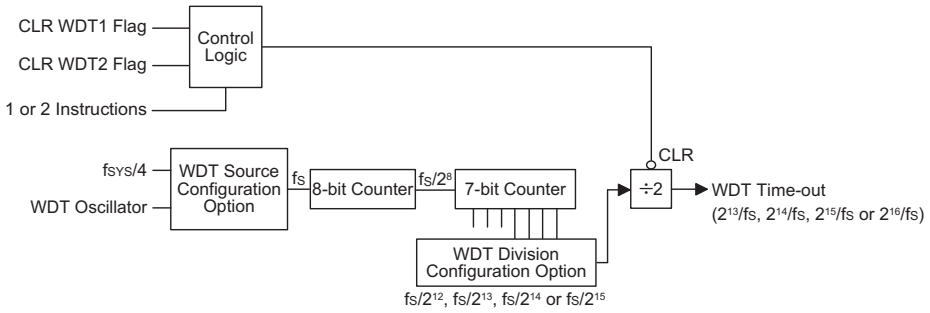
如果选用外部 RC 振荡方式，在 OSC1 与 VSS 之间需要接一个外部电阻，其阻值为 30kΩ～750kΩ；而 OSC2 上会输出带上拉的系统频率的 4 分频信号，可用于同步外部逻辑。RC 振荡方式是一种低成本的方案，但是，RC 振荡频率会随着 VDD、温度和芯片自身参数的漂移而产生误差。因此，在需要精确振荡频率做为计时操作的场合，并不适合使用 RC 振荡方式。

如果选用晶体振荡方式，在 OSC1 和 OSC2 之间需要连接一个晶体，用来提供晶体振荡器所需的反馈和相移，除此之外，不再需要其它外部元件。另外，在 OSC1 和 OSC2 之间也可使用谐振器来取代晶体振荡器，但是在 OSC1 和 OSC2 需要多连接两个电容(如果振荡频率小于 1MHz)。

WDT 振荡器是一个自由振荡内部 RC 振荡器，并不需要连接任何外部元件。当系统进入暂停模式时，系统时钟会停止，但 WDT 振荡器会继续工作，其振荡周期大约为 65μs/5V。如果要降低功耗，可在掩膜选项中关闭 WDT 振荡器。

看门狗定时器 — WDT

看门狗定时器的时钟来源有两种：看门狗振荡器或指令时钟(系统时钟 4 分频)，由掩膜选项设置。看门狗定时器主要用来防止程序运行故障和程序跳入一死循环而导致不可预测的结果。看门狗定时器可由掩膜选项设置为打开或关闭，如果在关闭状态，所有与 WDT 有关的指令操作都是没有作用的。



看门狗定时器

WDT 时钟源(fs)被一个内部的计数器进一步分频以得到一个更长的看门狗溢出时间。在 HT46R53A/HT46R54A 中，分频系数可以由通过掩膜选项选择，掩膜选项的分频系数有 2^{12}~2^{15}。

如果 WDT 时钟源为内部 WDT 振荡(RC 振荡周期一般为 65μs/5V)，当 WDT 的溢出时间(time out)选为 2^{16}，溢出时间大约可达到 4.3s。溢出时间会因为温度、VDD 以及芯片参数的变化而变化。

WDT 时钟源如果不选择内部 WDT 振荡器的话，还可以使用指令时钟(系统时钟 4 分频)，只是在 HALT 时，WDT 会停止计数而失去保护功能；此时只能靠外部逻辑复位来重新启动系统。如果系统运用在强干扰的环境中，建议选用内部 WDT 振荡器，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。

在正常运行时，WDT 溢出会使系统复位并置位 TO 标志；但在 HALT 模式下，WDT 溢出只产生“热复位”，只有程序计数器 Program Counter 和堆栈指针 SP 被复位。要清除 WDT 的值可以有三种方法：外部复位(低电平输入到$\overline{\text{RES}}$端)、清除看门狗指令或 HALT 指令。清除看门狗指令有“CLR WDT”和“CLR WDT1”、“CLR WDT2”二组指令。这两组指令中，只能选择其中一组，由掩膜选项决定。如果选择“CLR WDT”，那么只要执行“CLR WDT” 指令就会清除 WDT。如果选择“CLR WDT1”和“CLR WDT2”，那么二条指令要交替使用才会清除 WDT，否则，WDT 会由于溢出而使系统复位。

如果 WDT 的分频系数选择为 $f_S/2^{16}$，那么 WDT 的溢出周期为固定为 $f_S/2^{16}$，因为“CLR WDT”和“CLR WDT1”、“CLR WDT2”指令能清除全部的 WDT 分频器。

暂停模式 — HALT

暂停模式是由 HALT 指令来实现的，暂停模式时系统状态如下：

- 系统振荡器停振，但 WDT 振荡器会继续振荡(如果选择 WDT 振荡器)。
- RAM 和寄存器内容保持不变。
- WDT 被清除并重新开始计数(如果 WDT 时钟来源为 WDT 振荡器)。
- 所有输入/输出口都保持其原有状态。
- 置位 PDF 标志，清除 TO 标志。

以下操作可以使系统离开暂停模式：外部复位、中断、PA 口下降沿信号或看门狗定时器溢出。其中，外部复位会使系统初始化，WDT 溢出则会发生“热复位”。通过检测 TO 和 PDF 标志，即可了解系统复位的原因。PDF 标志可由系统上电或执行“CLR WDT”指令清除，由 HALT 指令置位。TO 标志由 WDT 溢出置位，同时产生唤醒，但只有程序计数器 Program Counter 和堆栈指针 SP 被复位，其它都保持其原有的状态。

PA 口唤醒和中断唤醒可做为正常运行的继续。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。

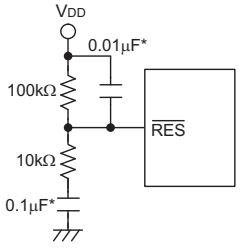
为减小功耗，在进入暂停模式之前，应小心处理所有的输入/输出口状态。

复位

总共有三种方法会产生初始复位：

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

暂停模式中的看门狗定时器溢出与其它系统复位状况不同，因为看门狗定时器溢出会执行“热复位”，只有程序计数器 Program Counter 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。



复位电路

注：“*”连线应该尽量靠近$\overline{\text{RES}}$

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

注：“u”表示不变

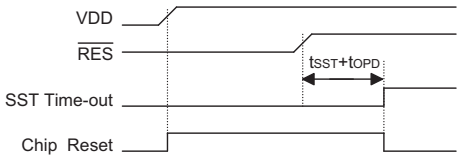
为了保证系统振荡器起振并稳定运行，系统复位(包括上电复位、WDT 溢出或由$\overline{\text{RES}}$端复位)或由暂停状态唤醒时，系统启动定时器(SST)提供了一个额外的延迟时间，共 1024 个系统时钟周期。

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

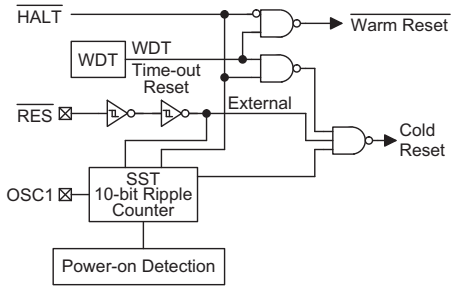
系统复位(包括上电复位、正常运行时 WDT 溢出或由$\overline{\text{RES}}$端复位)需要额外增加一个加载掩膜选项(Option)的时间。

系统复位时各功能单元的状态如下所示：

Program Counter	000H
中断	禁止
预分频器、分频器	清除
WDT	清除，在主系统复位后，WDT 开始计数
定时/计数器	停止
输入/输出口	输入模式
堆栈指针 SP	指向堆栈顶部



复位时序



复位电路结构

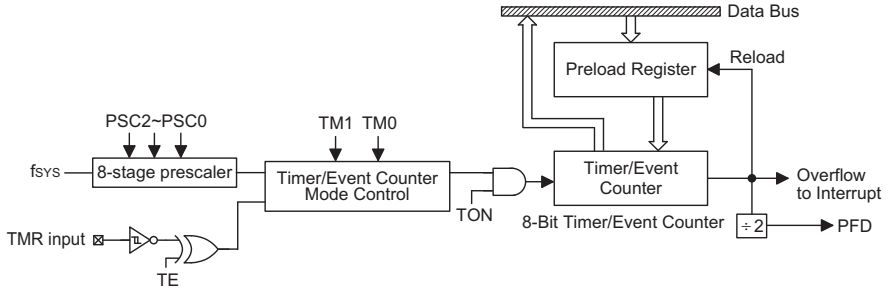
有关寄存器的状态如下：

寄存器	复位 (上电复位)	WDT 溢出 (正常运行)	$\overline{\text{RES}}$复位 (正常运行)	$\overline{\text{RES}}$复位 (暂停模式)	WDT 溢出 (暂停模式)*
MP0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP (HT46R54A)	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
Program Counter	0000H	0000H	0000H	0000H	0000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
STATUS	--00 xxxx	--lu uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	11111111	1111 1111	11111111	11111111	uuuu uuuu
PBC	11111111	11111111	11111111	11111111	uuuu uuuu
PC	---1 1111	---1 1111	---1 1111	---1 1111	---u uuuu
PCC	---1 1111	---1 1111	---1 1111	---1 1111	---u uuuu
PD	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u
PDC	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u
PWM	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	xxxx ----	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	---- --00	---- --00	---- --00	---- --00	---- --uu

注： 1.“*”表示“热复位； 2.“u”表示不变化； 3.“x”表示不确定。

定时/计数器

系统有一个定时/计数器(TMR)。定时/计数器是 8 位向上计数的，其时钟来源可以是外部信号输入或内部时钟，内部时钟为 f_{SYS}。外部信号输入可以用来计数外部事件、测量时间间隔、测量脉冲宽度或产生一个精确的时基信号。



定时/计数器

有两个与定时/计数器有关的寄存器，TMR(0DH)和 TMRC(0EH)。写入 TMR 会将特定值装入到定时/计数器的寄存器中，而读 TMR 则会取得定时/计数器的内容。TMRC 是定时/计数器控制寄存器，用来定义定时/计数器的操作模式、允许或禁止计数和计数的触发边缘。

位	符号	功能
0 1 2	PSC0 PSC1 PSC2	定义预分频器级数，PSC2，PSC1，PSC0= 000：$f_{INT}=f_{SYS}$ 001：$f_{INT}=f_{SYS}/2$ 010：$f_{INT}=f_{SYS}/4$ 011：$f_{INT}=f_{SYS}/8$ 100：$f_{INT}=f_{SYS}/16$ 101：$f_{INT}=f_{SYS}/32$ 110：$f_{INT}=f_{SYS}/64$ 111：$f_{INT}=f_{SYS}/128$
3	TE	定义定时/计数器 TMR 的触发方式 在计数模式（TM1，TM0）=（0，1） 1：下降沿计数 0：上升沿计数 在脉宽测量模式（TM1，TM0）=（1，1） 1：上升沿开始计数，下降沿结束计数 0：下降沿开始计数，上升沿结束计数
4	TON	打开/关闭定时/计数器(1=打开，0=关闭)
5	—	未用，读出为“0”
6 7	TM0 TM1	定义工作模式：TM1，TM0 01=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

TMRC（0EH）寄存器

TM0、TM1 用来定义定时/计数器的工作模式。外部事件计数模式是用来记录外部事件的，其时钟来源为外部 TMR 引脚输入。定时器模式是一个常用模式，其时钟来源为内部时钟。脉宽测量模式可以测量 TMR 引脚高/低电平的脉冲宽度，其时钟来源为内部时钟。

无论是定时模式还是外部事件计数模式，一旦开始计数，定时/计数器会从寄存器当前值向上计到 0FFH。一旦发生溢出，定时/计数器会从预置寄存器中重新加载初值，并开始计数；同时置位中断请求标志(TF；INTC 的第 5 位)。在脉宽测量模式，当 TON 与 TE 是 1 时，只要 TMR 引脚有一个上升沿信号(如果 TE 是 0，则为下降沿信号)，定时/计数器就会开始计数，直到 TMR 脚电平恢复，同时 TON 被清零。测量的结果会保存在寄存器中，直到有新的测量开始。换句话说，一次只能测量一个脉冲宽度。重新置位 TON 后，可以继续测量。注意，在该模式下，定时/计数器是跳变触发而不是电平触发。当计数器溢出时，定时/计数器会从预置寄存器中重新加载初值，并置位中断请求标志，这与其它两种模式一样。

要启动计数器，只要置位定时器开启位(TON；TMRC 的第 4 位)。在脉宽测量模式下，TON 在测量结束后会被自动清除；但在另外两种模式中，TON 只能由指令来清除。定时/计数器的溢出可以做为唤醒信号，如果 PA3 选择为 PFD 输出，可以提供给 PFD (可编程分频输出)使用。不管是什么模式，只要写 0 到 ETI 即可禁止定时/计数器中断服务。当使用 PFD 功能时，执行“SET [PA].3”可以打开 PFD 输出，执行“CLR [PA].3”则关闭 PFD 输出。

在定时/计数器停止计数时，写数据到定时/计数器的预置寄存器中，同时会将该数据写入到定时/计数器。但如果在定时/计数器运行时这么做，数据只能写入到预置寄存器中，直到发生溢出时才会将数据从预置寄存器加载到定时/计数器寄存器。

读取定时/计数器时，计数会被停止，以避免发生错误；而计数停止会导致计数错误，程序员必须注意到这一点。特别要注意的，正确的操作是在打开相关的定时/计数器之前要先将期望的值加载到 TMR 寄存器中，因为 TMR 的初始值是未知的。鉴于定时/计数器的配置，程序员编程要特别注意的是在第一次使用定时/计数器时候要先打开再关闭定时/计数器以避免不可预期的情况发生，不管你此时是否需要使用定时/计数器功能。在这个程序之后，定时/计数器的功能才能正常操作。

TMR0C 的第 0~2 位用来定义内部时钟预分频级数，定义如上表所示。定时/计数器的溢出信号可做为 PFD 输出。定时/计数器的预分频器还可以用于 PWM 计数。

输入/输出口

系统有 22 个双向输入/输出口，记为 PA、PB、PC 和 PD，其分别对应 RAM 地址[12H]、[14H]、[16H]和[18H]，所有端口都可以进行输入/输出操作。输入时，端口没有锁存功能，输入信号必须在 MOV A，[m](m=12H、14H、16H 或 18H)指令的 T2 上升沿到来前准备好；输出时，端口有锁存功能，端口上的数据会保持不变直到执行下一个写入操作。

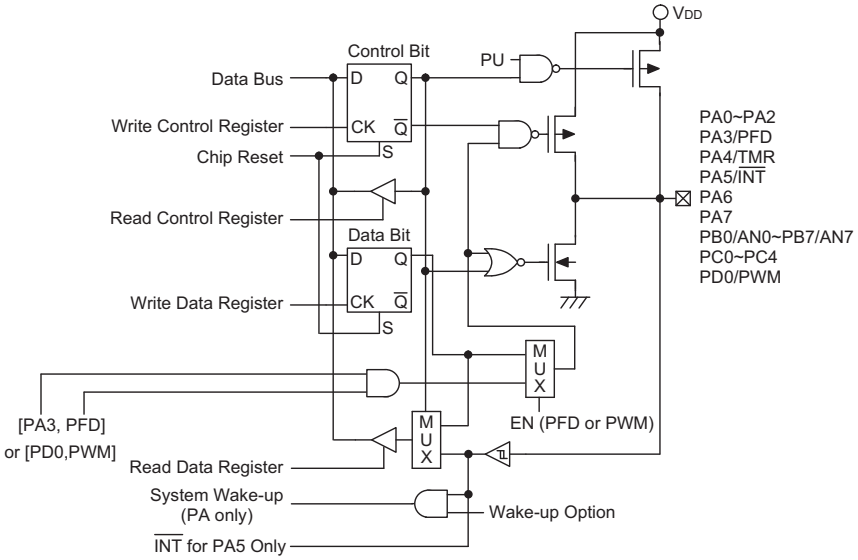
每个输入/输出口都有一个控制寄存器(PAC，PBC，PCC，PDC)，用来控制输入/输出状态。利用控制寄存器，可对 CMOS 输出、带或不带上拉电阻的斯密特触发输入通过软件动态地进行改变。做为输入时，对应的控制寄存器应设置为“1”。输入信号来源也取决于控制寄存器，如果控制寄存器的值为“1”，那么读取的是引脚状态；如控制寄存器的值为“0”，则读取的是内部锁存器的值。后者可能会在‘读-修改-写’指令中发生。

做为输出时，只能采用 CMOS 输出。控制寄存器对应 RAM 地址 13H、15H、17H 和 19H。

系统复位之后，这些输入/输出口会是高电平或浮空状态(由上拉电阻选项决定)。每一个输入/输出锁存位都能用“SET [m].i”或“CLR [m].i”指令置位或清除(m=12H、14H、16H 或 18H)。

有些指令会先输入数据，然后进行输出操作。例如：“SET [m].i”，“CLR [m].i”，“CPL [m]”，“CPLA[m]”这些指令会先将整个端口状态读入 CPU 中，接着执行所定义的运算(位操作)，然后再将结果写入锁存器或累加器中。

PA 的每一个口都具有唤醒系统的能力。所有的输入/输出口都有上拉电阻选项。一旦选择了上拉电阻选项，输入/输出口就加了上拉电阻。如果不选择上拉电阻，必须注意在输入模式下，若输入/输出口会产生浮空状态。



输入/输出口

PA3、PA4 和 PA5 分别与 PFD、TMR 和 $\overline{INT}$ 共用引脚。

如果选择 PFD 功能，则 PA3 在输出模式时的输出信号将是由定时/计数器的溢出信号产生的 PFD 信号，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 PA3 数据寄存器控制。向 PA3 数据寄存器写入“1”，则输出 PFD 信号；向 PA3 数据寄存器写入“0”，则 PA3 输出为“0”。 PA3 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)
PA3	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/计数器溢出频率的 1/2

PFD 输出频率和控制信号如下表：

定时/计数器	定时/计数器预置值	PA3 数据寄存器	PA3 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	$f_{INT}/[2\times(256-N)]$

注意：“X”表示未定义
“U”表示未知
“N”是定时/计数器的计数预装初值
“f_{TMR}”是定时/计数器的输入时钟频率

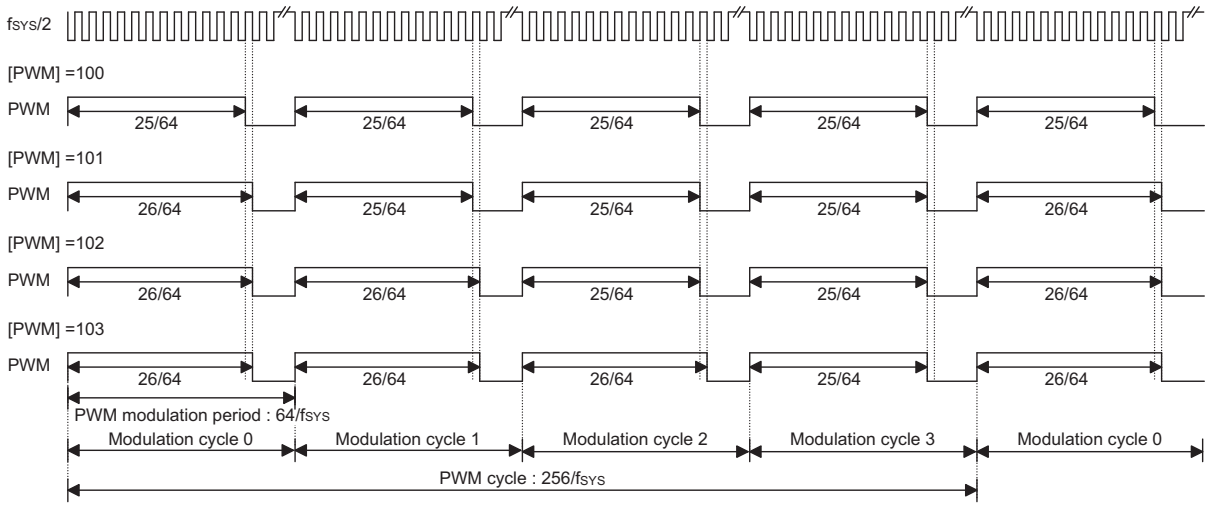
PB 口可以用做 A/D 转换输入，A/D 转换功能将在下面说明。PD0 与 PWM 输出共用引脚。如果选择 PWM 功能，则 PD0 口会有 PWM 信号输出(PD0 为输出模式)。向 PD0 数据寄存器写入“1”，则输出 PWM 信号；向 PD0 数据寄存器写入“0”，则 PD0 输出为“0”。PD0 的输入/输出如下所示：

I/O 模式	I/P(正常)	O/P(正常)	I/P(PWM)	O/P(PWM)
PD0	逻辑输入	逻辑输出	逻辑输入	PWM

建议用软件将未使用和没有外接的输入/输出口设置为输出模式，以防止这些端口在输入浮空时增加系统的功耗。

PWM

系统有 1 个通道(6+2)/(7+1)位的 PWM 输出(由掩膜选项决定)，与 PD0 共用引脚。PWM 通道由数据寄存器 PWM(1AH)来控制输出。PWM 计数器的时钟来源为系统时钟(f_{SYS})。PWM 有 1 个 8 位寄存器。PWM 的输出波形如图所示。一旦 PD0 选择为 PWM 输出，并且 PD0 为输出模式(PDC.0=“0”)，则向 PD0 寄存器写“1”能够产生 PWM 输出，向 PD0 寄存器写“0”会使 PD0 输出保持为“0”。

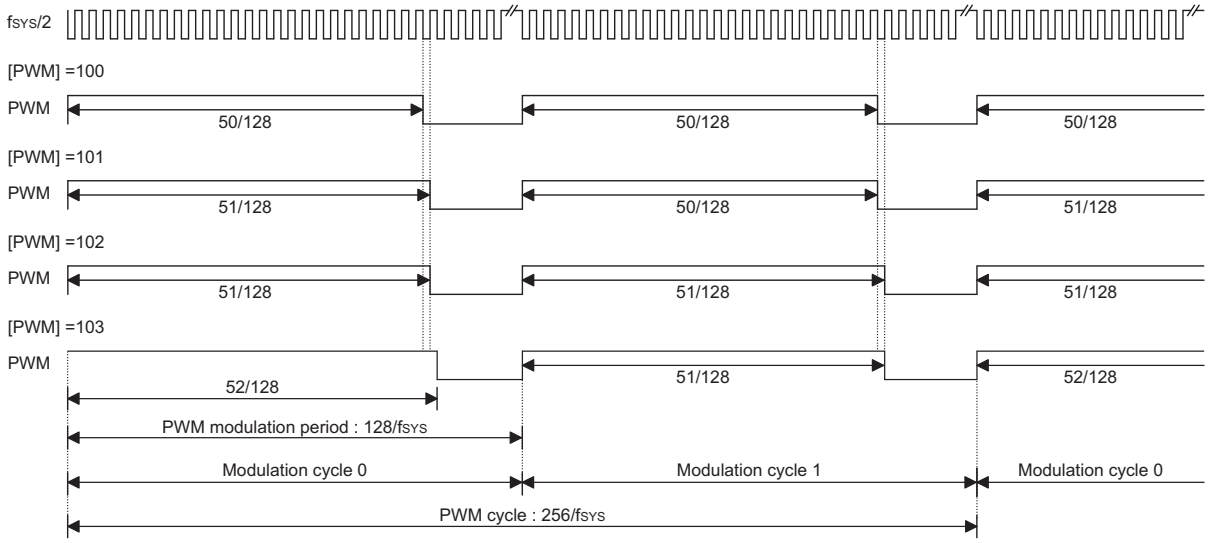


(6+2)位模式 PWM

在(6+2)位 PWM 模式中，一个 PWM 周期被分为四个调制周期(调制周期 0~调制周期 3)，每个调制周期有 64 个 PWM 输入时钟。在(6+2)位 PWM 模式中，PWM 寄存器被分为 2 个部分。第一部分是直流分量，由 PWM.7~PWM.2 控制；第二部分是交流分量，由 PWM.1~PWM.0 控制。

在(6+2)位 PWM 模式中，每个调制周期的占空比见下表：

参数	AC(0~3)	占空比
调制周期 i (i=0~3)	i<AC	$\frac{DC+1}{64}$
	i≥AC	$\frac{DC}{64}$



(7+1)位模式 PWM

在(7+1)位 PWM 模式中，一个 PWM 周期被分为两个调制周期(调制周期 0~调制周期 1)，每个调制周期有 128 个 PWM 输入时钟。在(7+1)位 PWM 模式中，PWM 寄存器被分为 2 个部分。第一部分是直流分量，由 PWM.7~PWM.1 控制；第二部分是交流分量，由 PWM.0 控制。

在(7+1)位 PWM 模式中，每个调制周期的占空比见下表：

参数	AC(0~1)	占空比
调制周期 i (i=0~1)	i<AC	$\frac{DC+1}{128}$
	i≥AC	$\frac{DC}{128}$

PWM 的调制频率、周期频率和占空比的关系总结如下：

PWM 调制频率	PWM 周期频率	PWM 占空比
$f_{SYS}/64$(6+2 模式) $f_{SYS}/128$(7+1 模式)	$f_{SYS}/256$	[PWM]/256

A/D 转换

系统有 8 个通道、12 位解析度(11 位精度)的 A/D 转换器。

与 A/D 转换有关的寄存器有 4 个：ADRL(20H)、ADRH(21H)、ADCR(22H)和 ACSR(23H)。ADRH 和 ADRL 是 A/D 转换结果的高字节和低字节寄存器，是只读寄存器。当完成 A/D 转换后，可从 ADRH 和 ADRL 读取 A/D 转换结果。ADCR 是 A/D 转换控制寄存器，用来定义 A/D 通道数量、模拟输入通道选择、A/D 转换开始控制和完成标志。如果要进行 A/D 转换，要先定义好 PB 口的设置，选择转换的模拟通道，然后给 START 控制位一个上升沿信号和一个下降沿信号(0→1→0)。完成 A/D 转换后，$\overline{EOC}$ 位会被清除，并且产生 A/D 转换中断(如果 A/D 转换允许)。ACSR 是 A/D 时钟控制寄存器，用来选择 A/D 的时钟来源。

位	符号	功能
0 1	ADCS0 ADCS1	选择 A/D 转换时钟源： 00=系统时钟/2 01=系统时钟/8 10=系统时钟/32 11=未定义
2~6	—	未用，读出为“0”
7	TEST	只做为内部测试用

ACSR（23H）寄存器

A/D 转换控制寄存器用来控制 A/D 转换。ADCR 的第 2~0 位用来选择模拟输入通道，总共有 8 个通道可以选择。ADCR 的第 5~3 位用来设置 PB 的工作模式，PB 可以做为模拟输入通道，或是数字输入/输出口，由这 3 位来决定。如果 PB 选择为模拟输入，则其输入/输出功能和上拉电阻将失效，而 A/D 转换电路会被使能。$\overline{EOC}$ 位(ADCR 的第 6 位)是 A/D 转换结束标志位。通过检测这个标志位可以知道 A/D 转换是否结束。ADCR 的 START 位用来开启 A/D 转换，给 START 位一个上升沿信号和一个下降沿信号可以开始 A/D 转换。为了确保 A/D 转换顺利完成，START 位应保持为“0”，直到 $\overline{EOC}$ 位变为“0”(A/D 转换完成信号)。

位	符号	功能
0 1 2	ACS0 ACS1 ACS2	选择模拟输入通道
3 4 5	PCR0 PCR1 PCR2	定义 PB 口的设置 如果 PCR0、PCR1 和 PCR2 都为 0，则 A/D 转换电路被关闭以减小功耗
6	$\overline{EOC}$	A/D 转换结束标志(0：A/D 转换结束) 每次 BIT3-5 状态的改变都必须通过 START 信号来初始化 A/D 转换器，否则 $\overline{EOC}$ 可能会处于不确定状态，具体可参照“A/D 转换初始化注意事项”
7	START	A/D 转换启始控制位 0→1→0：开始； 0→1：A/D 转换复位并且置 $\overline{EOC}$ 为“1”

ADCR（22H）寄存器

ACS2	ACS1	ACS0	模拟通道
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

模拟输入通道选择

PCR2	PCR1	PCR0	7	6	5	4	3	2	1	0
0	0	0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0	0	1	PB7	PB6	PB5	PB4	PB3	PB2	PB1	AN0
0	1	0	PB7	PB6	PB5	PB4	PB3	PB2	AN1	AN0
0	1	1	PB7	PB6	PB5	PB4	PB3	AN2	AN1	AN0
1	0	0	PB7	PB6	PB5	PB4	AN3	AN2	AN1	AN0
1	0	1	PB7	PB6	PB5	AN4	AN3	AN2	AN1	AN0
1	1	0	PB7	PB6	AN5	AN4	AN3	AN2	AN1	AN0
1	1	1	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0

PB 口的设置

ACSR 的第 7 位是内部测试用的，用户不能使用。ACSR 的第 1 位和第 0 位用来选择 A/D 转换的时钟来源。

当 A/D 转换完成时，A/D 中断标志被置位。当 START 标志由“0”置为“1”时，$\overline{EOC}$ 也置为“1”。

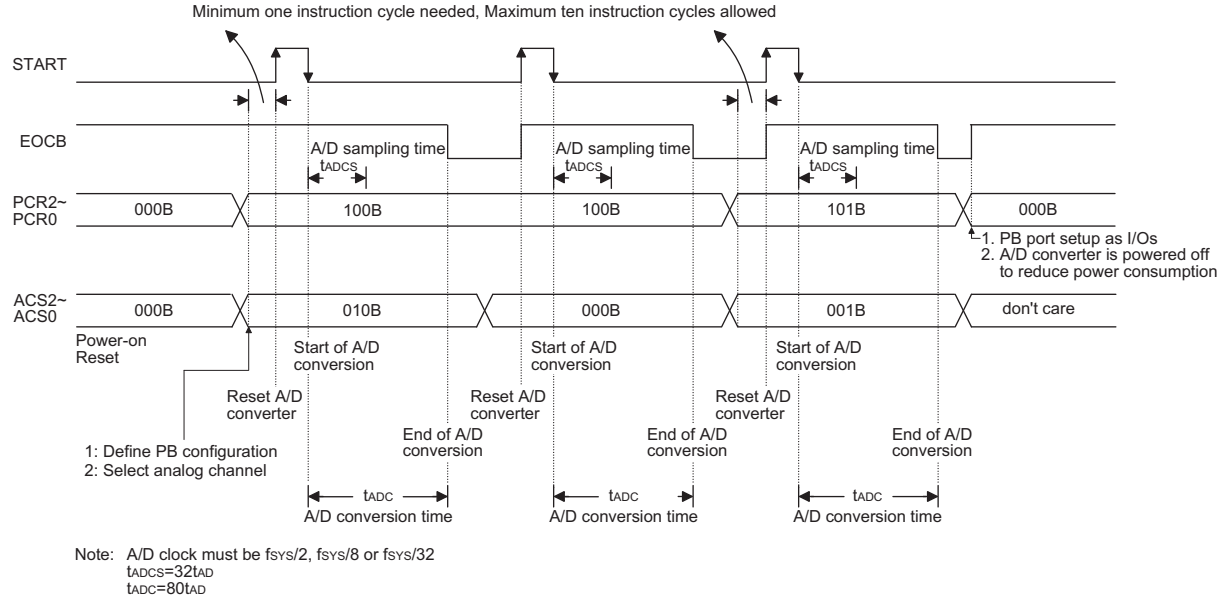
A/D 转换初始化注意事项：

每次改变模拟通道选择位后都要注意初始化 A/D 转换器，否则 $\overline{EOC}$ 可能处于不确定状态。在模拟通道选择位改变的 10 个指令周期内将 START 置 1 后清 0 来初始化 A/D 转换器。模拟通道选择位都清 0，可以不初始化 A/D。

寄存器	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRL（20H）	D3	D2	D1	D0	0	0	0	0
ADRH（21H）	D11	D10	D9	D8	D7	D6	D5	D4

注：D0~D11 是 A/D 转换结果的低位~高位

ADRL(20H) 寄存器 ADRH(21H) 寄存器



A/D 转换时序

下面举两个例子说明如何启动和实现 A/D 转换。第一个例子是不断扫描 ADCR 寄存器的$\overline{EOC}$位来判断 A/D 转换是否完成；而第二个例子直接用中断的方法来判断 A/D 转换是否完成。

例 1：通过扫描$\overline{EOC}$位判断 A/D 转换是否完成。

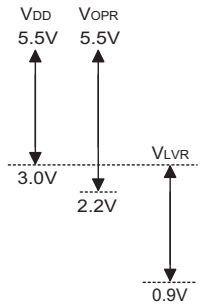
```
    clr EADI                ;禁止A/D中断
    mov a,00000001B
    mov ACSR,a              ; 设置ACSR寄存器，选择fSYS/8做为A/D转换时钟
    mov a,00100000B         ; 在ADCR寄存器中设置Port PB0~PB3做为A/D输入
    movADCR,a               ; 设置AN0进行A/D转换


                            ; 当模拟通道选择位改变后，START信号（0-1-0）必须在10个
                            ; 指令周期内发出
Start_conversion:
    clr ADCR.7
    set ADCR.7              ; A/D转换复位
    clr ADCR.7              ; 开始A/D转换
Polling_EOC:
    sz ADCR.6               ; 扫描ADCR寄存器的EOC位判断A/D转换是否完成
    jmp polling_EOC         ; 继续扫描
    mov a,ADRH              ; 从ADRH寄存器读取A/D转换结果的高位字节
    mov adrh_buffer,a       ; 将结果放入用户定义的寄存器中
    mov a,ADRL              ; 从ADRL寄存器读取A/D转换结果的低位字节
    mov adrl_buffer,a       ; 将结果放入用户定义的寄存器中
        :
        :
    jmp start_conversion    ; 开始下一次A/D转换
```

例 2：用中断方法判断 A/D 转换是否完成。

```
        clr EADI                ;禁止A/D中断
        mov a,00000001B
        mov ACSR,a              ; 设置ACSR寄存器，选择fSYS/8做为A/D转换时钟
        mov a,00100000B
        mov ADCR,a              ; 在ADCR寄存器中设置Port PB0~PB3做为A/D输入
                                ; 设置AN0进行A/D转换
start_conversion:
        clr   ADCR.7
        set ADCR.7              ; A/D转换复位
        clr ADCR.7              ; 开始A/D转换
        clr ADF                 ; 清除AD中断请求标志
        set INTC0.0             ; 打开总中断
        set INTC1.0             ; 在中断控制寄存器中设置A/D中断允许
            :
            :
; 中断服务子程序
ADC_ISR:
        mov acc_stack,a         ; 将ACC保存到用户定义的寄存器中
        mov a,STATUS
        mov status_stack,a      ; 将STATUS保存到用户
            :
            :
        mov a,ADRH              ; 从ADRH寄存器读取A/D转换结果的高位字节
        mov adrh_buffer,a       ; 将结果放入用户定义的寄存器中
        mov a,ADRL              ; 从ADRL寄存器读取A/D转换结果的低位字节
        mov adrl_buffer,a       ; 将结果放入用户定义的寄存器中
        clr ADCR.7
        set ADCR.7              ; A/D转换复位
        clr ADCR.7              ; 开始A/D转换
            :
            :
        EXIT_INT_ISR:
        Mov a,status_stack
        Mov STATUS,a            ; 将STATUS从暂存器中读出
        mov a,a_buffer          ; 将ACC从暂存器中读出
        reti
```

低电压复位—LVR

为了监控器件的工作电压，系统提供低电压复位功能。如果器件的工作电压在 0.9V～V_{LVR} 之间，例如电池电压的变化，那么 LVR 会自动使器件产生内部复位。

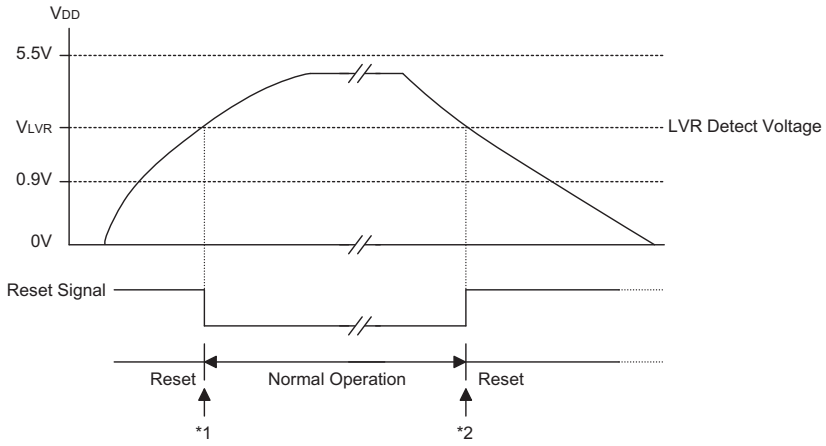


注：V_{OPR} 是在系统时钟为 4MHz 时，使得芯片正常运行的电压值

LVR 功能说明如下：

- 低电压(0.9V～V_{LVR})的状态必须持续 1ms 以上。如果低电压的状态没有持续 1ms 以上，那么 LVR 会忽视它而不去执行复位功能。
- LVR 通过与外部$\overline{RES}$信号的“或”的功能来执行系统复位。

V_{DD} 与 V_{LVR} 之间的关系如下所示：



低电压复位

注：*1：要保证系统振荡器起振并稳定运行，在系统进入正常运行以前，SST 提供额外的 1024 个系统时钟周期的延迟。

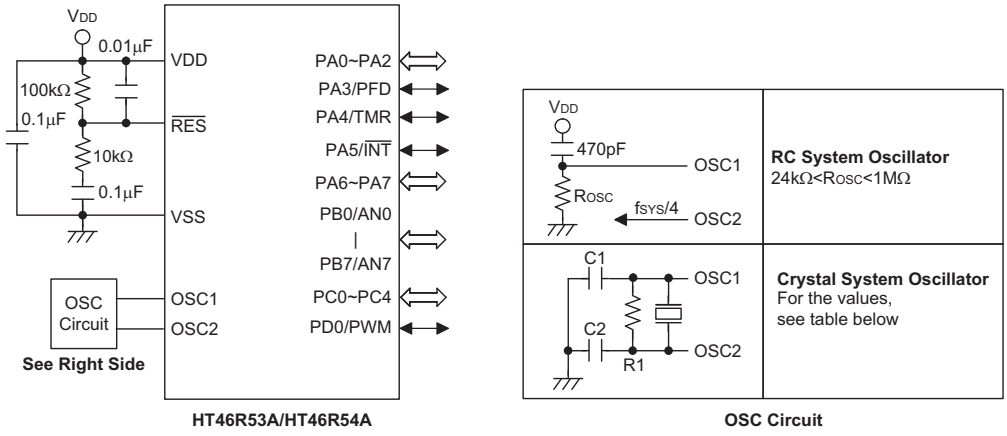
*2：因为低电压状态必须保持 1ms 以上，因此进入复位模式就要有 1ms 的延迟。

掩膜选项

下表列出了所有掩膜选项。所有选项必须正确定义，以保证统正常运行。

选项
振荡类型选择。 该选项用来定义系统时钟来源为 RC 振荡还是晶体振荡。
WDT 时钟源选择。 有三种选择：内部 WDT 振荡、指令时钟或关闭。
清除 WDT 指令选择。 该选项用来定义清除 WDT 的指令条数。“1 条指令”表示“CLR WDT”就能清除 WDT；“2 条指令”表示要同时使用“CLR WDT1”和“CLR WDT2”才能清除 WDT。
唤醒选择。 该选项用来定义唤醒功能，外部输入/输出口(只有 PA)都具有将系统从 HALT 模式中唤醒的能力。(按位定义)
上拉选择。 该选项用来定义输入/输出口做为输入时，是否带有内部上拉电阻；PA 可以按位定义，PB、PC 和 PD 按端口定义。
PFD 选择。 PA3：电平输出或 PFD 输出。
PWM 选择：(7+1)或(6+2)模式。 PD0：电平输出或 PWM 输出。
WDT 预分频选择。 有四种分频选择：2^{12}、2^{13}、2^{14} 和 2^{15}。
低电压复位功能：打开/关闭。
MP0/MP1 7 位或者 8 位选择。 如果 MP0 和 MP1 被选择为 7 位寄存器，那么数据存储器 80H 之后的地址，无法通过 MP0 和 MP1 存取.。

应用电路



下表是不同晶体频率时，C1、C2 和 R1 的不同取值。

晶体或共振器	C1、C2	R1
4MHz 晶体	0pF	10kΩ
4MHz 共振器	10pF	12kΩ
3.58MHz 晶体	0pF	10kΩ
3.58MHz 共振器	25pF	10kΩ
2MHz 晶体和共振器	25pF	10kΩ
1MHz 晶体	35pF	27kΩ
480kHz 共振器	300pF	9.1kΩ
455kHz 共振器	300pF	10kΩ
429kHz 共振器	300pF	10kΩ
R1 的作用是在低电压的时候确保关闭振荡，此低电压值低于单片机的最低工作电压。需要注意的是如果 LVR 使能，可以不加 R1。		

注： 电阻和电容值选取的原则是使 VDD 保持稳定并在$\overline{\text{RES}}$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$\overline{\text{RES}}$引脚的线请尽可能地短

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 0.5μs 中执行完成，而分支或调用操作则将在 1μs 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

下列表格根据功能描述了指令集的分类，利用下表列出的惯例可以作为基本的指令参考。

表格惯例：

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
SUB A,x	ACC 与立即数相减，结果放入 ACC	1	Z,C,AC,OV
SUB A,[m]	ACC 与数据存储器相减，结果放入 ACC	1	Z,C,AC,OV
SUBM A,[m]	ACC 与数据存储器相减，结果放入数据存储器	1 注	Z,C,AC,OV
SBC A,[m]	ACC 与数据存储器、进位标志的反相减，结果放入 ACC	1	Z,C,AC,OV
SBCM A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	1 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
RET	A,x	从子程序返回，并将立即数放入 ACC	2	无
RETI		从中断返回	2	无
查表				
TABRDC	[m]	读取当前页的 ROM 内容，并送至数据存储器和 TBLH	2 注	无
TABRDL	[m]	读取最后页的 ROM 内容，并送至数据存储器和 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←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]←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←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←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]←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←ACC “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←ACC “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]←ACC “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←Program Counter+1
Program Counter ← 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] ←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 ←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 ← 00H
PDF & TO ← 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 ← 00H
PDF & TO ← 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 ← 00H
PDF & TO ← 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] ← [$\overline{m}$]
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 ← [$\overline{m}$]
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] ←ACC+00H 或 [m] ←ACC+06H
[m] ←ACC+60H 或 [m] ←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] ← [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 ← [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 ← 1
TO ← 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] ← [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 ← [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 ← 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 ← [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 ← 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] ← 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 ← 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←ACC “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←ACC “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]←ACC “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 ← 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 ← Stack
ACC ← 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 ← Stack
EMI ← 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) ←[m].i (i=0~6)
[m].0 ←[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) ←[m].i (i=0~6)
ACC.0 ←[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) ←[m].i (i=0~6)
[m].0 ←C
C ← [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) ←[m].i (i=0~6)
ACC.0 ←C
C ← [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 ←[m].(i+1) (i=0~6)
[m].7 ←[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←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←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] ← [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] ←[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 ←[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

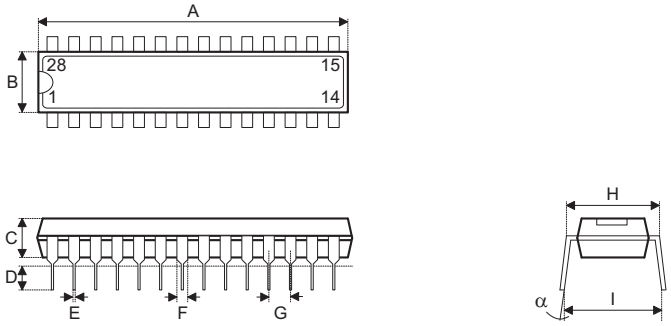
XOR A, [m] Logical XOR accumulator with data memory
说明： 将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。
运算过程： ACC←ACC “XOR” [m]
影响标志位： Z

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

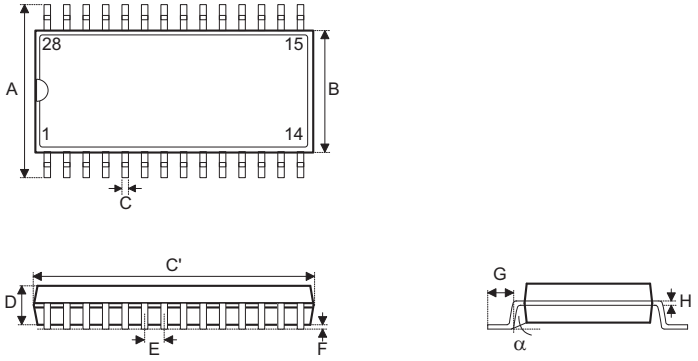
封装尺寸

28-pin SKDIP (300mil)外形尺寸



符号	尺寸 (单位：mil)		
	最小	典型	最大
A	1375	—	1395
B	278	—	298
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	—	—	375

28-pin SOP (300mil)外形尺寸

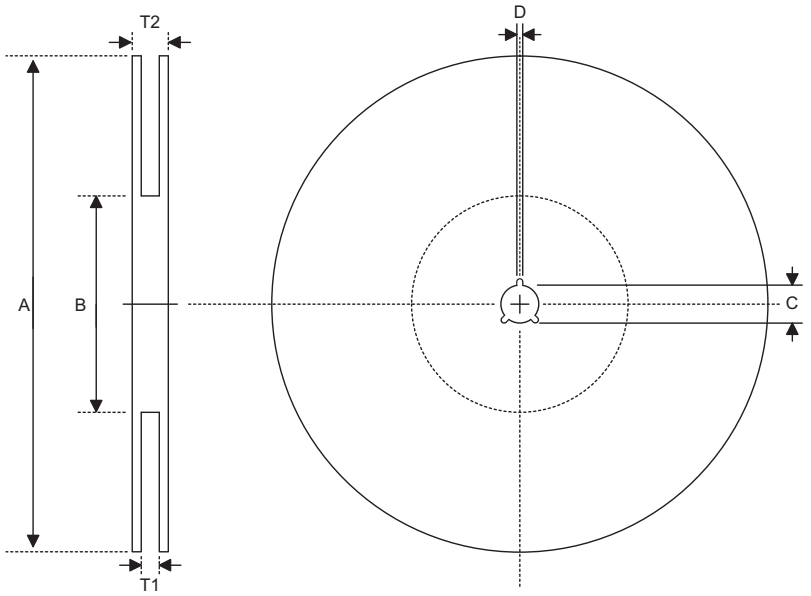


• MS-013

符号	尺寸 (单位：mil)		
	最小	典型	最大
A	393	—	419
B	256	—	300
C	12	—	20
C’	697	—	713
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

包装带和卷轴规格

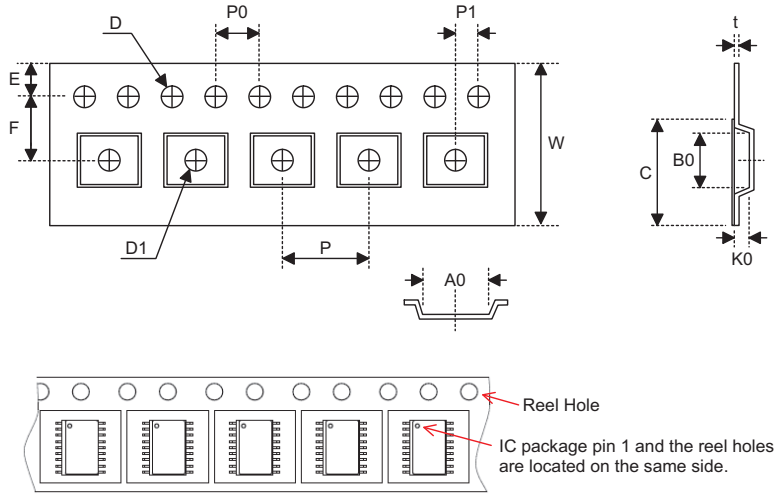
卷轴尺寸



SOP 28W (300mil)

符号	说明	尺寸 (单位：mm)
A	卷轴外圈直径	330.0±1.0
B	卷轴内圈直径	100.0±1.5
C	轴心直径	$13.0^{+0.5/-0.2}$
D	缝宽	2.0±0.5
T1	轮缘宽	$24.8^{+0.3/-0.2}$
T2	卷轴宽	30.2±0.2

运输带尺寸



SOP 28W

符号	说明	尺寸（单位：mm）
W	运输带宽	24.0±0.3
P	空穴间距	12.0±0.1
E	穿孔位置	1.75±0.10
F	空穴至穿孔距离(宽度)	11.5±0.1
D	穿孔直径	$1.5^{+0.1/-0.0}$
D1	空穴中之小孔直径	$1.50^{+0.25/-0.00}$
P0	穿孔间距	4.0±0.1
P1	空穴至穿孔距离(长度)	2.0±0.1
A0	空穴长	10.85±0.1
B0	空穴宽	18.34±0.1
K0	空穴深	2.97±0.1
t	传输带厚度	0.35±0.01
C	覆盖带宽度	21.3±0.1

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)

合泰半导体有限公司 (东莞业务处)
中国东莞松山湖翠竹路 4 号新竹苑 10 幢 (总部壹号 10 号楼)
电话 : 86-0769-2626-1300
传真 : 86-0769-2626-1311，86-0769-2626-1322

Holtek Semiconductor(USA), Inc. (北美业务处)
46729 Fremont Blvd., Fremont, CA 94538, USA
电话 : 1-510-252-9880
传真 : 1-510-252-9885
网站 : www.holtek.com