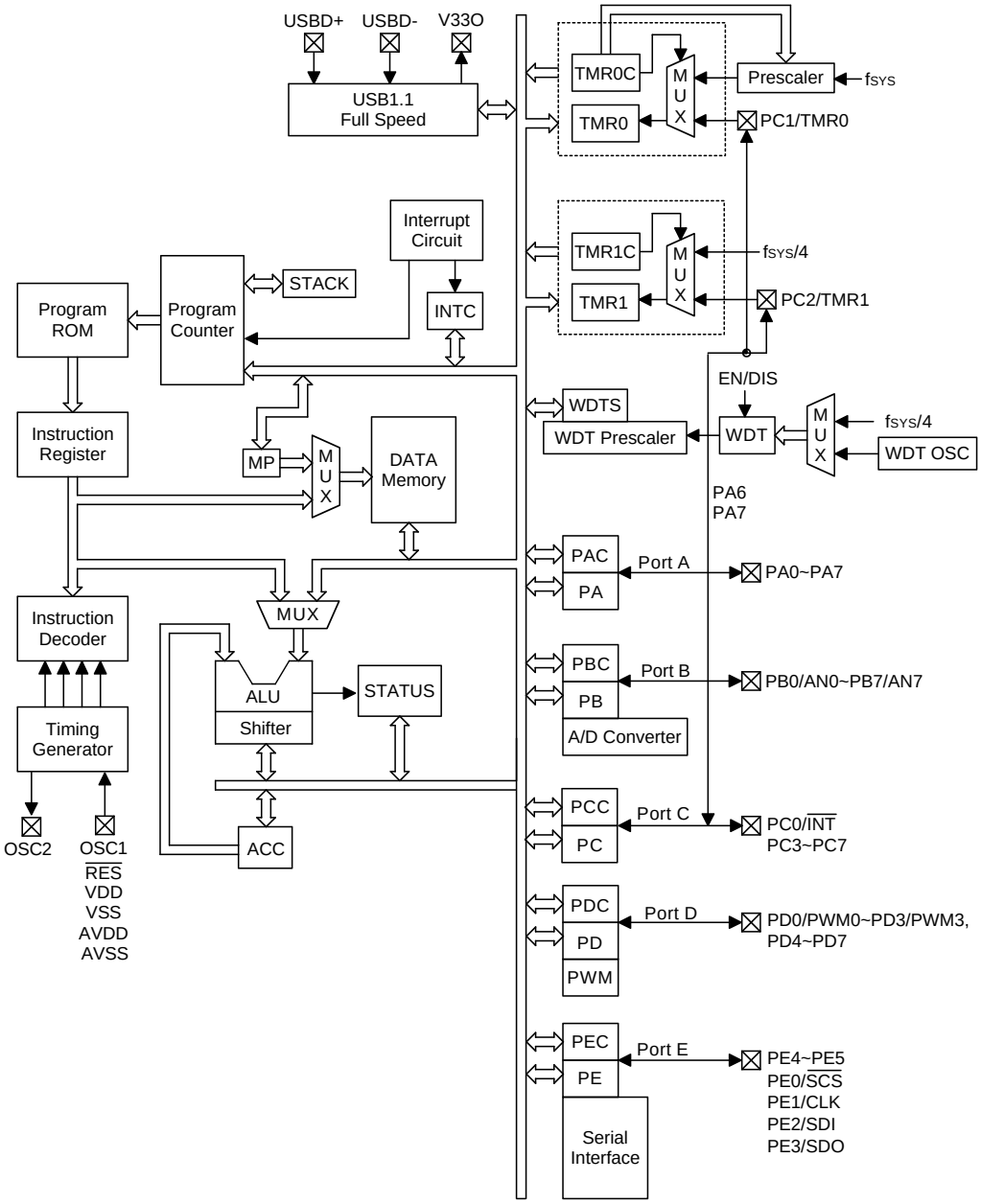
特性

- 工作电压：
  f_{SYS}=6MHz：2.2V~5.5V
  f_{SYS}=12MHz：2.7V~5.5V
  USB 总线电压：4.5V~5.5V
- 最多可有 38 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
- 两个 16 位可编程定时/计数器，具有溢出中断
- 晶体振荡（6MHz 或 12MHz）
- 看门狗定时器
- 8192×16 程序存储器 ROM
- 384×8 数据存储器 RAM
- HALT 和唤醒功能可降低功耗
- 在 V_{DD}=5V，系统频率为 12MHz 时，指令周期为 0.33μs
- 16 层硬件堆栈
- 8 通道 10 位解析度的 A/D 转换器
- 4 通道 8 位的 PWM 输出，与输入/输出口共用引脚
- SIO（同步串行口）功能
- 支持中断，控制和批量传输
- 兼容 USB 2.0 全速模式
- 支持 6 个端点（包括端点 0）
- 160 个字节 FIFO（EP0 到 EP5 分别为 8、8、8、64、8、64）
- 位操作指令
- 查表指令，表格内容字长 16 位
- 63 条指令
- 指令执行时间为 1 或 2 个指令周期
- 低电压复位功能
- 48-pin SSOP 封装

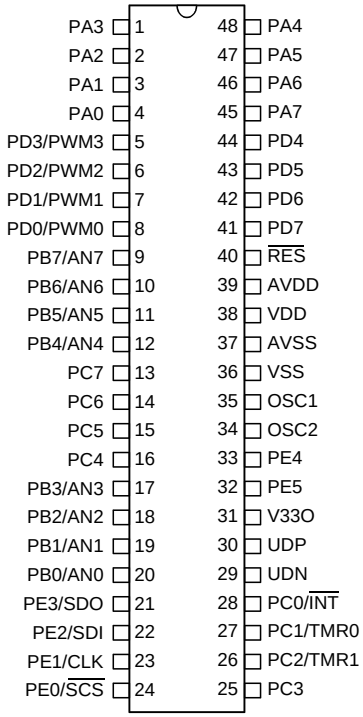
概述

HT46RB70 是 8 位高性能精简指令集单片机，专门为 USB 产品而设计。尤其适用于 USB 或 SPI 接口触控面板、USB 或 SPI 接口触控按键、PS2 摇杆、XBOX 摇杆、USB 鼠标键盘和摇杆。暂停模式可用于降低功耗。

方框图


USBD+
USBD-
V33O
USB1.1
Full Speed
TMR0C
TMR0
MUX
Prescaler
fSYS
PC1/TMR0
Interrupt
Circuit
TMR1C
TMR1
fSYS/4
PC2/TMR1
Program
ROM
Program
Counter
STACK
INTC
EN/DIS
WDTS
WDT Prescaler
WDT
WDT OSC
Instruction
Register
MP
DATA
Memory
PA6
PA7
PAC
PA
Port A
PA0~PA7
Instruction
Decoder
MUX
ALU
Shifter
STATUS
PBC
PB
Port B
PB0/AN0~PB7/AN7
A/D Converter
Timing
Generator
ACC
PCC
PC
Port C
PC0/INT
PC3~PC7
OSC2
OSC1
RES
VDD
VSS
AVDD
AVSS
PDC
PD
PWM
Port D
PD0/PWM0~PD3/PWM3,
PD4~PD7
PEC
PE
Port E
PE4~PE5
PE0/SCS
PE1/CLK
PE2/SDI
PE3/SDO
Serial
Interface

引脚图



HT46RB70
– 48 SSOP-A

引脚说明

引脚名称	输入/输出	配置选项	功能说明
PA0~PA7	输入/输出	上拉电阻 (位选择) 唤醒 (位选择)	8 位双向输入/输出口。每一位可由配置选项设置为唤醒输入。输入/输出模式由控制寄存器 PAC（PA 控制寄存器：位选择）决定。上拉电阻选项：PA0~PA7，位选择；唤醒选项：PA0~PA7。
PB0/AN0~PB7/AN7	输入/输出	上拉电阻 (位选择)	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 口可作为 AD 转换输入。
PC0/$\overline{\text{INT}}$ PC1/TMR0 PC2/TMR1 PC3~PC7	输入/输出	上拉电阻 (半字节选择)	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：半字节选择）的斯密特触输入。$\overline{\text{INT}}$、TMR0 和 TMR1 分别与 PC0、PC1 和 PC2 共用引脚。
PD0/PWM0~ PD3/PWM3 PD4~PD7	输入/输出	上拉电阻 (半字节选择)	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：半字节选择）的斯密特触输入。PWM0、PWM1、PWM2 和 PWM3 分别与 PD0、PD1、PD2 和 PD3 共用引脚。
PE0/$\overline{\text{SCS}}$	输入/输出	上拉电阻 (半字节选择)	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：半字节选择）的斯密特触输入。$\overline{\text{SCS}}$ 与 PE0 共用引脚。$\overline{\text{SCS}}$ 为串行接口芯片选择引脚，主模式为输出，从模式为输入。
PE1/CLK	输入/输出	上拉电阻 (半字节选择)	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：半字节选择）的斯密特触输入。CLK 与 PE1 共用引脚。CLK 为串行接口时钟输入/输出引脚（初始化为输入）。
PE2/SDI	输入/输出	上拉电阻 (半字节选择)	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：半字节选择）的斯密特触输入。SDI 与 PE2 共用引脚。SDI 为串行接口输入引脚。
PE3/SDO	输入/输出	上拉电阻 (半字节选择)	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：半字节选择）的斯密特触输入。SDO 与 PE3 共用引脚。SDO 为串行接口输出引脚。
PE4~PE5	输入/输出	上拉电阻 (半字节选择)	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：半字节选择）的斯密特触输入。
$\overline{\text{RES}}$	输入	—	斯密特触发复位输入，低电平有效。
VSS	—	—	负电源，接地。
AVSS	—	—	ADC 负电源，接地。
VDD	—	—	正电源。
AVDD	—	—	ADC 正源 28-pin：必外接 VDD。 48-pin：可连接到外部电源。
OSC1 OSC2	输入 输出	晶体或 RC	OSC1、OSC2 连接 6M 或 12M 的晶体振荡器/谐振器以产生内部系统时钟。
V33O	输出	—	3.3V 参考源输出。
UDP	输入/输出	—	UDP 为 USBD+引脚 USB 功能可通过软件控制。
UDN	输入/输出	—	UDN 为 USBD-引脚 USB 功能可通过软件控制。

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℃~125℃
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}=6MHz	2.2	—	5.5	V
		—	f_{SYS}=12MHz	2.7	—	5.5	V
V_{USB}	USB SIE 工作电压	—	—	4.5	—	5.5	V
I_{DD1}	工作电流(6M 晶振)	5V	无负载，f_{SYS}=6MHz	—	6.5	12	mA mA
I_{DD2}	工作电流(12M 晶振)	3V	无负载，f_{SYS}=12MHz	—	3.6	10	mA
		5V		—	7.5	16	mA
I_{STB1}	静态电流（看门狗打开）	3V	无负载，系统 HALT USB 挂起	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	静态电流（看门狗关闭）	3V	无负载，系统 HALT USB 挂起	—	—	1	μA
		5V		—	—	2	μA
I_{STB3}	静态电流（看门狗关闭）	5V	无负载,系统 HALT USB 收发 3.3V 参考电压打开	—	150	200	μA
V_{IL1}	输入/输出口的低电平输入电压	—	—	0	—	0.3 V_{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（$\overline{RES}$）	—	—	0	—	0.4 V_{DD}	V
V_{IH2}	高电平输入电压（$\overline{RES}$）	—	—	0.9 V_{DD}	—	V_{DD}	V
I_{OL}	输入/输出口灌电流	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V	V_{OL}=0.1V_{DD}	10	20	—	mA
I_{OH}	输入/输出口源电流	3V	V_{OH}=0.9V_{DD}	-2	-4	—	mA
		5V	V_{OH}=0.9V_{DD}	-5	-10	—	mA
R_{PH}	上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
V_{LVR}	低电压复位	—	配置选项为 3.0V	2.7	3	3.3	V
V_{V33O}	3.3V 参考电压输出	5V	I_{V33O} =-5mA	3	3.3	3.6	V
E_{AD}	A/D 转换误差	—	—	—	±0.5	±1	LSB

交流电气特性

Ta=25℃

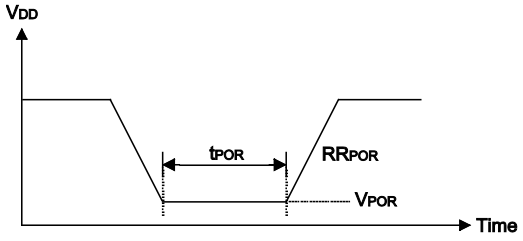
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{SYS}	系统时钟	—	2.2V~5.5V	400	—	6000	kHz
		—	3.3V~5.5V	400	—	12000	kHz
f_{TIMER}	定时器输入频率（TMR0/TMR1）	—	2.2V~5.5V	0	—	6000	kHz
		—	3.3V~5.5V	0	—	12000	kHz
t_{WDTOSC}	看门狗振荡器	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	外部复位低电平脉宽	—	—	1	—	—	μs
t_{SST}	系统启动延迟时间	—	从 HALT 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	—	—	—	76	—	t_{AD}
t_{ADCS}	A/D 采样时间	—	—	—	32	—	t_{AD}

注：$*t_{SYS}=1/f_{SYS}$

上电复位特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{POR}	上电复位电压	—	—	—	—	100	mV
RR_{POR}	上电复位电压速率	—	—	0.035	—	—	V/ms
t_{POR}	V_{DD}保持为V_{POR}的最小时间	—	—	1	—	—	ms

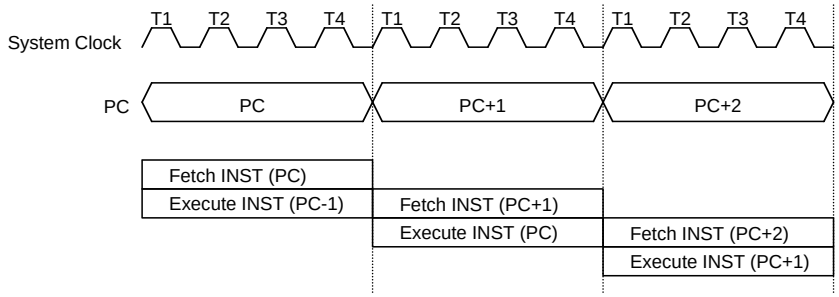


系统功能说明

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

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、条件跳跃、向 PCL 赋值、子程序调用、初始化复位、内部中断、外部中断、子程序返回等操作时，PC 会载入与指令相关的地址而非下一条指令地址。

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

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

当遇到控制转移指令时，系统也会插入一个空指令周期。

模式	程序计数器												
	***12**	***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	0
外部中断	0	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	0	1	0	0	0
定时/计数器 1 溢出	0	0	0	0	0	0	0	0	0	1	1	0	0
USB 中断	0	0	0	0	0	0	0	0	1	0	0	0	0
A/D 转换中断	0	0	0	0	0	0	0	0	1	0	1	0	0
串口接口中断	0	0	0	0	0	0	0	0	1	1	0	0	0
条件跳跃	PC+2												
装载 PCL	*12	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
跳转、子程序调用	#12	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
从子程序返回	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

程序计数器

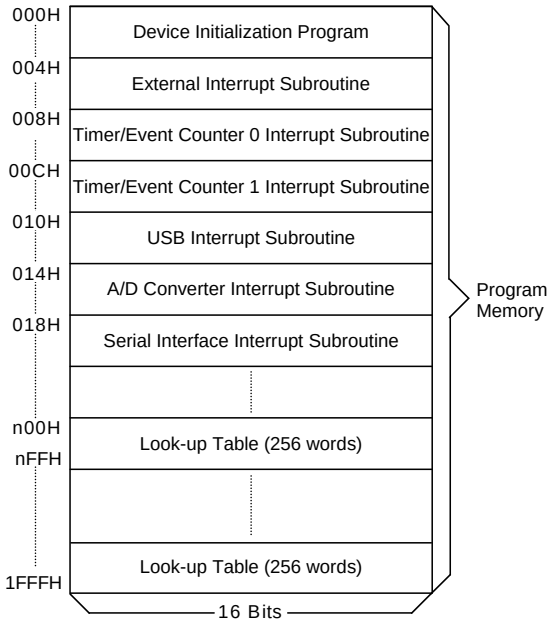
注： *12 ~ *0 ： 程序计数器位　　S12 ~ S0 ： 堆栈寄存器位
#12 ~ #0 ： 指令代码位　　@7 ~ @0 ： PCL 位

程序存储器 — ROM

程序存储器用来存放要执行的指令代码，以及一些数据、表格和中断入口。程序存储器有 8192×16 位，程序存储器空间可以用程序计数器或表格指针进行寻址。

以下列出的程序存储器地址是系统专为特殊用途而保留的：

- 地址 000H
  该地址为程序初始化保留。系统复位后，程序总是从 000H 开始执行。
- 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 0 中断服务程序保留。当定时/计数器 0 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
  该地址为定时/计数器 1 中断服务程序保留。当定时/计数器 1 溢出，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。



Note: n ranges from 0 to F

程序存储器

- 地址 010H
  该地址为 USB 中断服务程序保留。当发生 USB 中断，如果中断允许且堆栈未满，则程序会跳转到 010H 地址开始执行。
- 地址 014H
  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 014H 地址开始执行。
- 地址 018H
  该地址为串行接口中断服务程序保留。当 8 位数据接收或发送完成，如果中断允许且堆栈未满，则程序会跳转到 018H 地址开始执。
- 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。所有与表格有关的指令都需要两个指令周期的执行时间。这里提到的表格区都可以做为正常的程序存储器来使用。

指令	表格区												
	***12**	***11**	***10**	***9**	***8**	***7**	***6**	***5**	***4**	***3**	***2**	***1**	***0**
TABRDC[m]	P12	P11	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL[m]	1	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

表格区

注：*12~*0：表格地址位　　P12~P8：当前程序指针位　　@7~@0：表格指针位

堆栈寄存器 — STACK

堆栈寄存器是特殊的存储器空间，用来保存 PC 的值。HT46RB70 有 16 层堆栈，堆栈寄存器既不是数据存储器的一部分，也不是程序存储器的一部分，而且它既不能读出，也不能写入。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（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 16 个返回地址会被保留。

数据存储器 — RAM

数据存储器由 438×8 位组成，分为两个功能区间：特殊功能寄存器（51×8 位）和通用数据存储器（Bank0：192×8 位，Bank1：192×8 位），数据存储器单元大多数是可读/写的，但有些只读的。

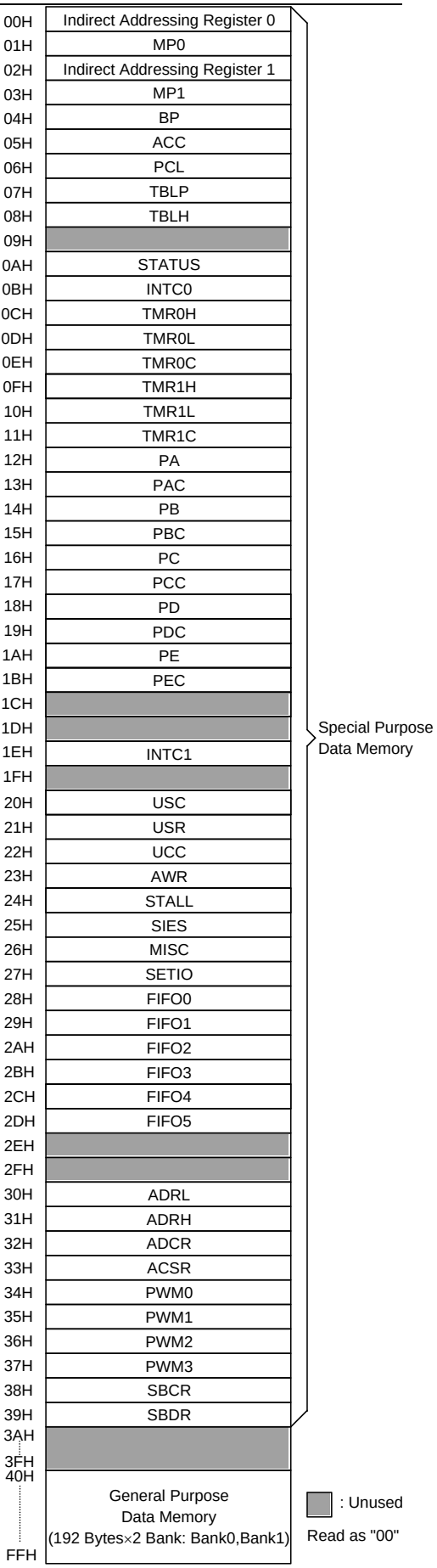
在 40H 之前的空间保留给系统以后扩展使用，读取这些地址的返回值为“00H”。通用数据寄存器地址从 40H 到 FFH(Bank0；BP=0 或 Bank1；BP=1)，用来存储数据和控制信息。

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。

将 BP 寄存器赋值为“01H”后，就可以访问 Bank1 数据存储区。这些 Bank 内 40H~FFH 之间的数据寄存器可以用 MP1 间接寻址方式进行读/写操作。不管 BP 是何值，直接寻址方式只能对 Bank0 内的数据寄存器进行操作。

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。间接寻址指针 MP0（01H）和 MP1（03H）是 8 位寄存器。



累加器

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

算术逻辑单元 — ALU

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

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

ALU 不仅可以储存数据运算的结果，还会改变状态寄存器的值。

状态寄存器 — STATUS

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

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。

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

STATUS(0AH) 寄存器

中断

HT46RB70 提供一个外部中断、两个内部定时/计数器中断、一个 A/D 转换中断、一个串行接口中断和一个 USB 中断。中断控制寄存器 0（INTC0；0BH）和中断控制寄存器 1（INTC1；1EH）包含了中断控制位和中断请求标志，中断控制位用来设置中断允许/禁止。

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、INTC0 和 INTC1 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；INTC0 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 0 中断是由定时/计数器 0 溢出触发的，其中断请求标志（T0F；INTC0 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 T0F 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 1 中断是由定时/计数器 1 溢出触发的，其中断请求标志（T1F；INTC0 的第 6 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 0CH 的子程序调用；而中断请求标志 T1F 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

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	ET0I	定时/计数器中断控制位（1=允许；0=禁止）
3	ET1I	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	T0F	定时/计数器中断请求标志（1=有；0=无）
6	T1F	A/D 转换中断请求标志（1=有；0=无）
7	—	未用，读出为“0”

INTC0(0BH) 寄存器

位	符号	功　　能
0	EUI	USB 中断允许位（1=允许；0=禁止）
1	EADI	AD 转换中断允许位（1=允许；0=禁止）
2	ESII	串行接口中断允许位（1=允许；0=禁止）
3	—	未用，读出为“0”
4	USBF	USB 中断请求标志（1=有；0=无）
5	ADF	AD 转换中断请求标志（1=有；0=无）
6	SIF	串行接口中断请求标志（1=有；0=无）
7	—	未用，读出为“0”

INTC1(1EH) 寄存器

USB 中断是由以下 USB 事件触发的，其中断请求标志（USBF；INTC1 的第 4 位）会被置位。

- PC 访问相应的 USB FIFO
- PC 发出挂起信号
- PC 发出恢复信号
- USB 复位

如果中断允许，且堆栈未满，当发生 USB 事件中断时，会产生地址 10H 的子程序调用；而中断请求标志 USBF 和总中断控制位 EMI 会被清除，以禁止其它中断响应。当 PC 访问 HT46RB70 中 FIFO 时，寄存器 USR 中相应的请求标志会被置位，并触发 USB 中断。程序员可以很方便判断 PC 访问了哪个 FIFO，当相应的程序被服务过，相应的标志位必须由软件清除。当 HT46RB70 接收到 PC 发出挂起信号，挂起标志（SUSP；USC 的第 0 位）被置位并触发中断。同样，HT46RB70 接收到 PC 发出恢复信号，恢复标志（RESUME；USC 的第 3 位）被置位并触发中断。任何时候检测到 USB 复位信号都会触发 USB 中断。

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；INTC1 的第 5 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。

串行接口中断是由接收到或发送完完整的 8 位数据触发的，其中断请求标志位（SIF；INTC1 的第 6 位）会被置位。如果中断允许，且堆栈未满，当发生串行接口中断时，会产生地址 18H 的子程序调用；而中断请求标志位 SIF 和总中断控制位 EMI 会被清除，以禁止其它中断响应。

中断源	优先级	中断向量
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
USB 中断	4	10H
AD 转换中断	5	14H
串行接口中断	6	18H

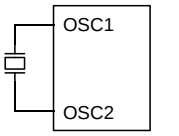
在执行中断子程序期间，其它的中断请求会被屏蔽，直到执行 RETI 指令或 EMI 和相关中断控制位被置位（当然，此时堆栈未满）。如果要从中断子程序返回，只要执行 RET 或 RETI 指令即可。其中，RETI 指令会自动置位 EMI，以允许中断服务，而 RET 则不会。

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。

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

HT46RB70 只有外部晶体振荡，其信号做为系统时钟。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。



Crystal Oscillator

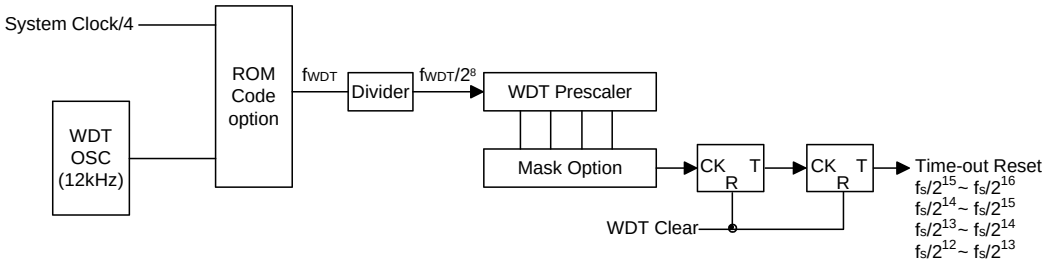
晶体振荡方式需要在 OSC1 和 OSC2 之间连接一个晶体振荡器，用来提供晶体振荡器所需的反馈和相移，除此之外，不再需要其它外部元件。另外，在 OSC1 和 OSC2 之间也可使用谐振器来取代晶体振荡器，但是在 OSC1 和 OSC2 需要多连接两个电容。

WDT 振荡器是一个内部 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。如果要降低功耗，可在配置选项中关闭 WDT 振荡器。

为了确保振荡器的稳定性及减少噪声和串扰的影响,晶体振荡器及其相关的电阻和电容以及他们之间的连线都应尽可能的接近单片机。

看门狗定时器

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)，由配置选项设置。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。看门狗定时器可由配置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

如果 WDT 时钟源为内部 WDT 振荡(RC 振荡周期一般为 65μs)，该频率可再加 2^{12}~2^{15}(通过配置选项：WDT time out)的分频系数。最小的 WDT 溢出周期大约是 300mS~600mS。溢出时间会因为温度、VDD 以及芯片参数的变化而变化。如果再用 WDT 预分频器，则可以得到更长的溢出周期。如果 WDT 的溢出时间(time out)选为 2^{15}，最大的溢出时间可达到 2.3s～4.7s(分频系数为 2^{15}~2^{16})。

WDT 时钟源除了使用内部 WDT 振荡器输出外，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。

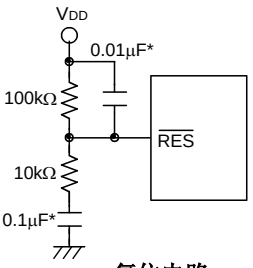
在正常运行时，WDT 溢出会使系统复位并置位 TO 标志；但在 HALT 模式下，WDT 溢出只产生“热复位”，只有程序计数器 PC 和堆栈指针 SP 被复位。要清除 WDT 的值可以有三种方法：外部复位(低电平输入到$\overline{\text{RES}}$端)、清除看门狗指令或 HALT 指令。清除看门狗指令有“CLR WDT”和“CLR WDT1”、“CLR WDT2”两组指令。这两组指令中，只能选择其中一组，由配置选项决定。如果选择“CLR WDT”，那么只要执行“CLR WDT” 指令就会清除 WDT。如果选择“CLR WDT1”和“CLR WDT2”，那么两条指令要交替使用才会清除 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 $f_S/2^{12}$(由配置选项决定)，则 WDT 的溢出周期为 $f_S/2^{12}$～$f_S/2^{13}$，因为“CLR WDT”和“CLR WDT1”、“CLR WDT2”指令只能清除最后两级 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。



复位电路

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

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 PC 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配置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 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

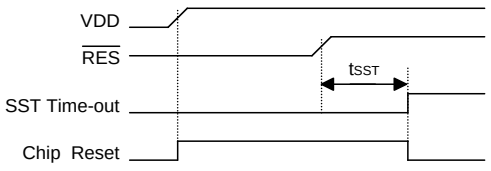
为了保证系统振荡器起振并稳定运行，系统复位(包括上电复位、WDT 溢出或由$\overline{\text{RES}}$端复位)或由暂停状态唤醒时，系统启动定时器(SST)提供了一个额外的延迟时间，共 1024 个系统时钟周期。

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。

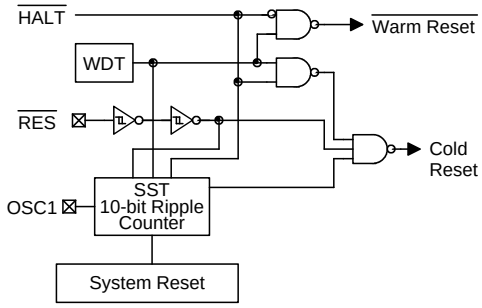
系统复位(包括上电复位、正常运行时 WDT 溢出或由$\overline{\text{RES}}$端复位)需要额外增加一个加载配置选项(Option)的时间。

系统复位时各功能单元的状态如下所示：

PC	000H
中断	禁止
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

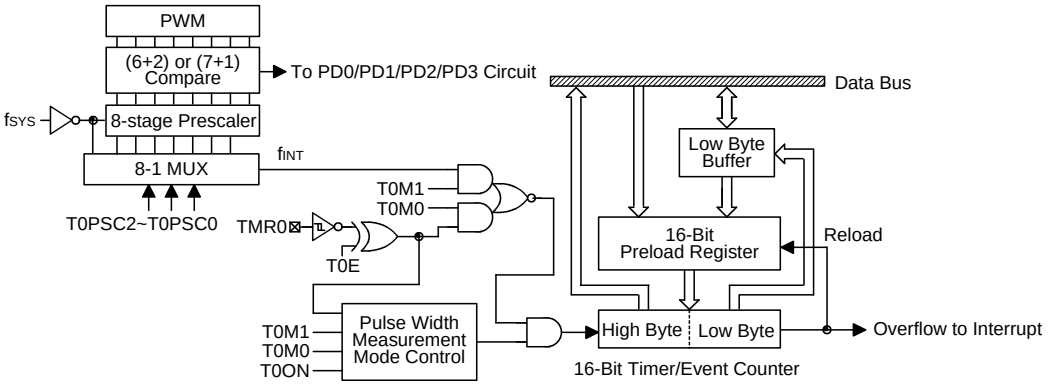
有关寄存器的状态如下：

寄存器	复位 (上电复位)	WDT 溢出 (正常运行)	$\overline{RES}$ 复位 (正常运行)	$\overline{RES}$ 复位 (暂停模式)	WDT 溢出 (暂停模式)*	USB 复位 (正常运行)	USB 复位 (暂停模式)
TMR0H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
TMR0L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu	00-0 1000	00-0 1000
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
TMR1C	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---	00-0 1---	00-0 1---
PC	000H	000H	000H	000H	000H	000H	000H
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
STATUS	--00 xxxx	--lu uuuu	--uu uuuu	--01 uuuu	--11 uuuu	--uu uuuu	--01 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 0000
INTC1	-000 -000	-000 –000	-000 –000	-000 –000	-uuu -uuu	-000 -000	-000 -000
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PE	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PEC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
AWR	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STALL	--11 1110	--uu uuuu	--11 1110	--11 1110	--uu uuuu	--11 1110	--11 1110
MISC	0xx- -000	uxx- -uuu	0xx- -000	0xx- -000	uxx- -uuu	000- -000	000- -000
SETIO	--11 1110	--uu uuuu	--11 1110	--11 1110	--uu uuuu	--11 1110	--11 1110
FIFO0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO3	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO4	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO5	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
USC	1-00 0000	u-uu uuuu	1-00 0000	1-00 0000	u-uu uuuu	u-00 0100	u-00 0100
USR	--00 0000	--uu uuuu	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
UCC	-000 0000	-uuu uuuu	-000 0000	-000 0000	-uuu uuuu	-uu0 u000	-uu0 u000
SIES	0100 0000	uuuu uuuu	0100 0000	0100 0000	uuuu uuuu	0100 0000	0100 0000
ADRL	xx-- ----	xx-- ----	xx-- ----	xx-- ----	uu-- ----	xx-- ----	xx-- ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu	0100 0000	0100 0000
ACSR	1--- --00	1--- --00	1--- --00	1--- --00	u--- --uu	1--- --00	1--- --00
PWM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PWM1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PWM2	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PWM3	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
SBCR	0110 0000	0110 0000	0110 0000	0110 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
SBDR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu

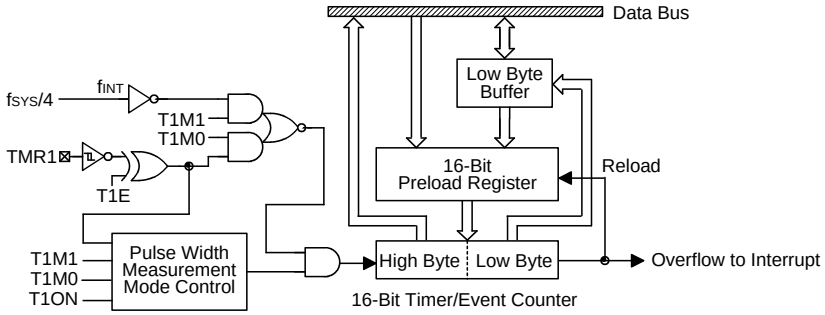
注： 1.“*”表示“热复位； 2.“u”表示不变化； 3.“x”表示不确定。

定时/计数器

HT46RB70 提供两个定时/计数器（TMR0，TMR1）。定时/计数器 0 是 16 位可编程向上计数的，其时钟来源可以是外部信号或内部时钟，内部时钟为 f_{SYS}。定时/计数器 1 是 16 位可编程向上计数的，其时钟来源可以是外部信号或内部时钟，内部时钟为 $f_{SYS}/4$。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 0



定时/计数器 1

有六个与定时/计数器有关的寄存器，TMR0H(0CH)，TMR0L(0DH)，TMR0C(0EH)，TMR1H(0FH)，TMR1L(10H)和 TMR1C(11H)。写入 TMR0L(TMR1L)只能将数据写到低字节缓冲器(8 位)， 而写入 TMR0H(TMR1H)会把指定数据和低字节缓冲器的数据分别写到 TMR0H(TMR1H)和 TMR0L(TMR1L)预置寄存器中。读取 TMR0H(TMR1H)会把 TMR0H(TMR1H)的内容送至目标单元，而 TMR0L(TMR1L)的值被送至低字节缓冲器中。读 TMR0L(TMR1L)将读取低字节缓冲器的值。TMR0C（TMR1C）是定时/计数器 0(1)控制寄存器，用来定义定时/计数器一些选项。

T0M0、T0M1 和 T1M0、T1M1 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，其时钟来源为外部引脚输入(TMR0，TMR1)。定时器模式是一个常用模式，其时钟来源为内部时钟 f_{INT}。脉宽测量模式可以测量 TMR0 或 TMR1 引脚高/低电平的脉冲宽度，其时钟来源为内部时钟 f_{INT}。

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 0FFFFH。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(T0F；INTC0 的第 5 位，T1F；INTC0 的第 6 位)。

在脉宽测量模式，当 T0ON/T1ON 与 T0E/T1E 都是 1 时，只要 TMR0(TMR1)引脚有一个上升沿信号(如果 T0E/T1E 是 0，则为下降沿信号)，定时/计数器就会开始计数，直到 TMR0(TMR1)脚电平恢复，同时 T0ON/T1ON 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 T0ON/T1ON 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。

位	符号(TMR0C)	功能
0~2	T0PSC0~T0PSC2	定义预分频器级数，T0PSC2，T0PSC1，T0PSC0= 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	T0E	定义定时/计数器 TMR 的触发方式 0：上升沿计数 1：下降沿计数
4	T0ON	打开/关闭定时/计数器(1=打开，0=关闭)
5	—	未用，读出为“0”
6 7	T0M0 T0M1	定义工作模式： 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

TMR0C(0EH) 寄存器

位	符号(TMR1C)	功能
0~2，5	—	未用，读出为“0”
3	T1E	定义定时/计数器 TMR 的触发方式 0：上升沿计数 1：下降沿计数
4	T1ON	打开/关闭定时/计数器(1=打开，0=关闭)
6 7	T1M0 T1M1	定义工作模式，T1M1，T1M0： 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

TMR1C(11H) 寄存器

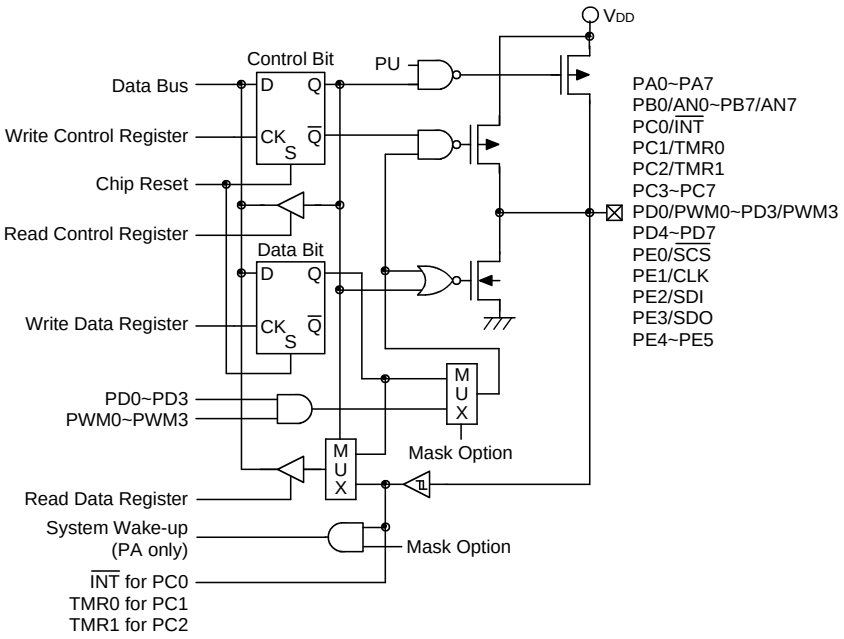
要启动计数器，只要置位定时/计数器开关 (T0ON：TMR0C 的第 4 位；T1ON：TMR1C 的第 4 位)。在脉宽测量模式下，T0ON/T1ON 在测量结束后会被自动清除；但在另外两种模式中，T0ON/T1ON 只能由指令来清除。定时/计数器溢出可以做为唤醒信号。不管是什么模式，只要写 0 到 ET0I 或 ET1I 即可禁止定时/计数器中断服务。

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；计数停止会导致计数错误，程序员必须注意到这一点。建议在打开定时/计数器前，先将相应的值写入 TMR0/TMR1 寄存器内，否则定时/计数器将从不确定值开始计数。

TMRC 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。

输入/输出口

HT46RB70 有 38 个双向输入/输出口，记为 PA、PB、PC、PD 和 PE，其分别对应 RAM 地址[12H]、[14H]、[16H]、[18H]和[1AH]，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、18H 和 1AH)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。



输入/输出口

每个输入/输出口都有一个控制寄存器(PAC，PBC，PCC，PDC，PEC)，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、19H 和 1BH。

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、18H 或 1AH）。

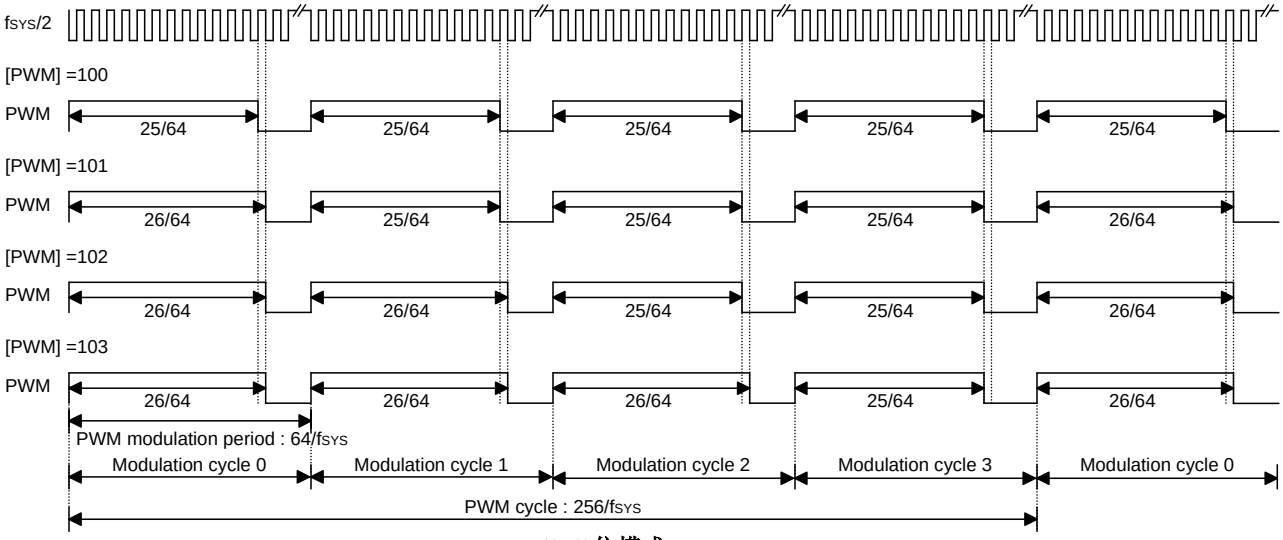
有些指令会先输入数据，然后进行输出操作。例如：“SET [m].i”，“CLR [m].i”，“CPL [m]”，“CPLA[m]”这些指令会先将整个端口状态读入 CPU 中，接着执行所定义的运算(位操作)，然后再将结果写入锁存器或累加器中。

PA 的每一个口都具有唤醒系统的能力。

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

HT46RB70 有 4 个通道(6+2)/(7+1)位的 PWM 输出(由配置选项决定)，与 PD0/PD1/PD2/PD3 共用引脚。PWM 通道由数据寄存器 PWM0(34H)、PWM1(35H)、PWM2(36H)和 PWM3(37H)来控制输出。PWM 计数器的时钟来源为系统时钟(f_{SYS})。PWM 有 4 个 8 位寄存器。PWM 的输出波形如图所示。一旦 PD0/PD1/PD2/PD3 选择为 PWM 输出，并且 PD0/PD1/PD2/PD3 为输出模式(PDC.0/PDC.1/PDC.2/PDC.3= “0”)，则向 PD0/PD1/PD2/PD3 寄存器写“1”能够产生 PWM 输出，向 PD0/PD1/PD2/PD3 寄存器写“0”会使 PD0/PD1/PD2/PD3 输出保持为“0”。

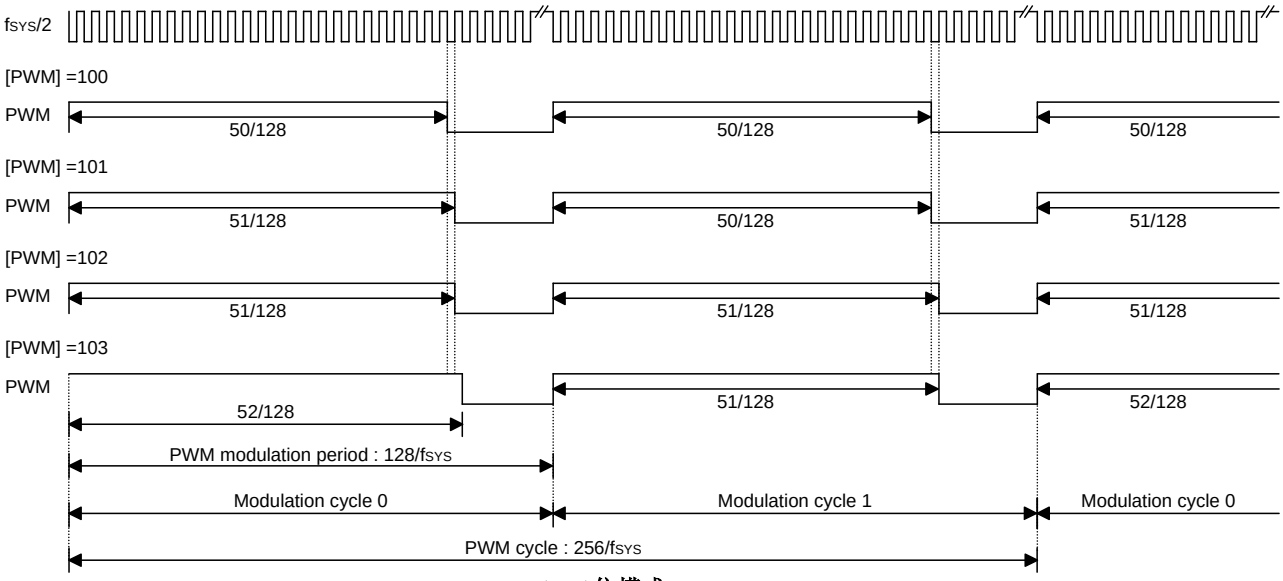


(6+2)位模式 PWM

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

HT46RB70 有 8 个通道、10 位解析度(9 位精度)的 A/D 转换器。其参考电压为 VDD。与 A/D 转换有关的寄存器有 4 个：ADRL(30H)、ADRH(31H)、ADCR(32H)和 ACSR(33H)。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。

符号(ACSR)	位	功能
ADCS0 ADCS1	0 1	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”
TEST	7	只做为内部测试用

ACSR(33H) 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)。

符号(ADCR)	位	功能
ACS0 ACS1 ACS2	0 1 2	选择模拟输入通道
PCR0 PCR1 PCR2	3 4 5	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
$\overline{EOC}$	6	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”
START	7	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(32H) 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 中断请求标志被置位。当 START 标志由“0”置为“1”时，$\overline{EOC}$ 也置为“1”。

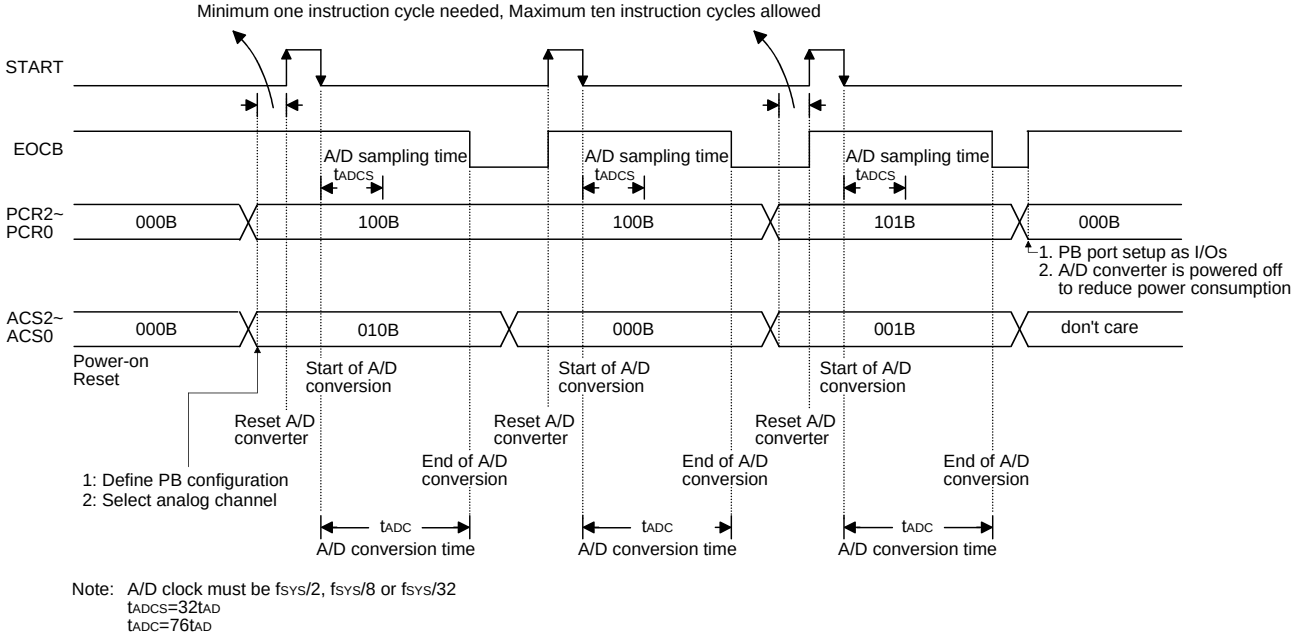
A/D 转换初始化注意事项：

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	D1	D0	—	—	—	—	—	—
ADRH	D9	D8	D7	D6	D5	D4	D3	D2

注：D0~D9 是 A/D 转换结果的低位~高位

ADRL(30H) 寄存器 ADRH(31H)寄存器



Note: A/D clock must be $f_{SYS}/2$, $f_{SYS}/8$ or $f_{SYS}/32$
$t_{ADCS}=32t_{AD}$
$t_{ADC}=76t_{AD}$

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。

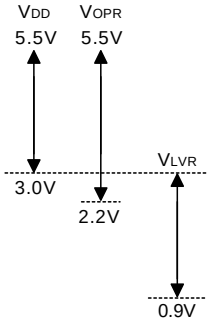
```
        clr     EADI            ;禁止A/D中断
        mov     a,00000001B
        mov     ACSR,a          ; 设置ACSR寄存器，选择fSYS/8做为A/D转换时钟
        mov     a,00100000B     ; 在ADCR寄存器中设置Port PB0~PB3做为A/D输入
        mov     ADCR,a          ; 设置AN0进行A/D转换
                :
                :
                :               ; 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     START
        set     START           ;  A/D转换复位
        clr     START           ; 开始A/D转换
Polling_EOC:
        sz      EOC             ; 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     ; 在ADCR寄存器中设置Port PB0~PB3做为A/D输入
    mov     ADCR,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     START
    set     START           ;  A/D转换复位
    clr     START           ; 开始A/D转换
    clr     ADF             ; 清除AD中断请求标志
    set     EADI            ; 打开 A/D 中断
    set     EMI             ; 打开总中断
            :
                :
; 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保存到用户定义的寄存器中
            :
            :
    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     START
    set     START           ;  A/D转换复位
    clr     START           ; 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,acc_stack     ; 将ACC从暂存器中读出
    reti
```

低电压复位—LVR

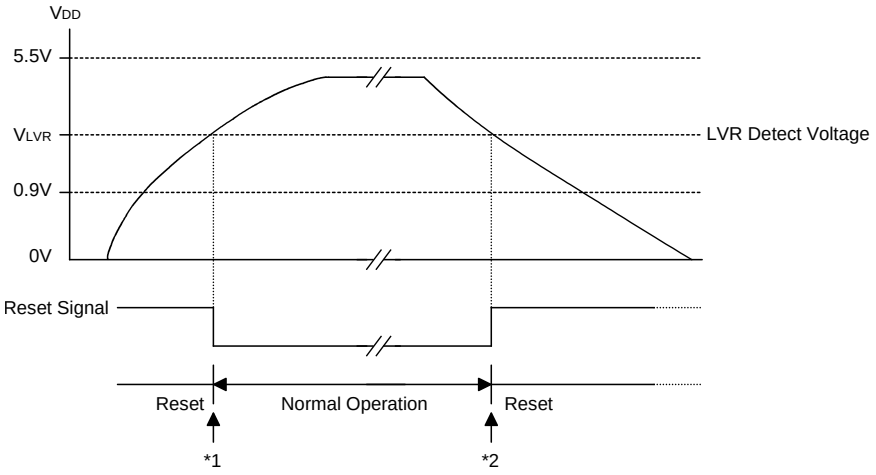
为了监控器件的工作电压，HT46RB70 提供低电压复位功能。如果器件的工作电压在 0.9V～V_{LVR} 之间，例如电池电压的变化，那么 LVR 会自动使器件产生内部复位。



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：

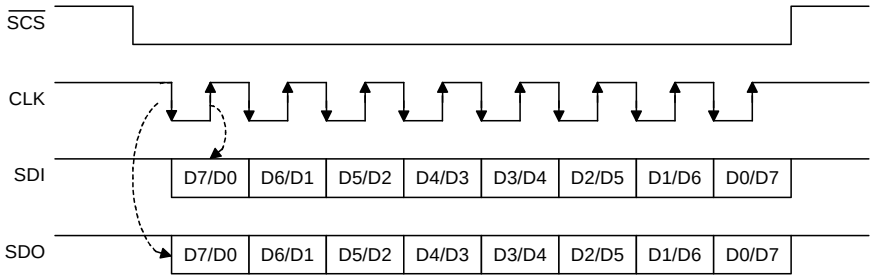


注：*1：要保证系统振荡器起振并稳定运行，在系统进入正常运行以前，SST 提供额外的 1024 个系统时钟周期的延迟。

*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。

串行接口

有四个与串行接口相关的信号线，SDI(串行数据输入线)、SDO(串行数据输出线)、SCK(串行时钟)和$\overline{SCS}$(从模式片选)。



	D7	D6	D5	D4	D3	D2	D1	D0	
SBCR	CKS	M1	M0	SBEN	MLS	CSEN	WCOL	TRF	SBCR : SERIAL BUS
DEFAULT	0	1	1	0	0	0	0	0	CONTROL REGISTER
SBDR	D7	D6	D5	D4	D3	D2	D1	D0	SBDR : SERIAL BUS
DEFAULT	U	U	U	U	U	U	U	U	DATA REGISTER

Note: "U" means unchanged.

有两个与串行接口相关的寄存器，SBCR 和 SBDR。

- SBCR：串行接口控制寄存器

第 7 位(CKS)，时钟源选择：$f_{SIO} = f_{SYS}/4$，选择为 0

第 6 位(M1)，第 5 位(M0)：主从模式及波特率选择

00 → 主模式，波特率为 f_{SIO}

01 → 主模式，波特率为 $f_{SIO}/4$

10 → 主模式，波特率为 $f_{SIO}/16$

11 → 从模式

第 4 位(SBEN)，串行接口使能/禁止位(1/0)

使能：($\overline{SCS}$取决于 CSEN)

禁止→使能：SCK、SDI、SDO、$\overline{SCS}$ =0(SCKB=“0”)，等待数据写入 SBDR 寄存器

主模式：当数据写入 SBDR 寄存器，硬件会自动将数据发送出去

主模式：当数据发送完毕，置位 TRF

从模式：当检测到有数据需要接收时，SDI 引脚上数据在 SCK 作用下移入缓冲器

禁止：SCK、SDI、SDO、$\overline{SCS}$处于浮空状态

第 3 位(MLS)，高位前导(MSB)或低位前导(LSB)控制位(1/0)

第 2 位(CSEN)，串行接口片选功能使能/禁止($\overline{SCS}$)，当 CSEN=0，SCSB 浮空

第 1 位(WCOL)，当数据写入 SBDR 寄存器且正在发送时置位，正在发送数据时，写 SBDR 将被忽略

第 0 位(TRF)，数据发送或接收用于产生中断

注意：系统进入 HALT，串行接收电路仍处于工作状态。

- SBDR：串行接口数据寄存器

写 SBDR 寄存器，数据将被加载到 TXRX 寄存器；读 SBDR 寄存器，将返回 SBDR 寄存器值。

操作模式描述：

主发送模式：写 SBDR 寄存器将触发数据的发送

从发送模式：接收到有效的时钟将触发发送

从接收模式：接收到有效的时钟将触发接收

时钟极性=上升沿($\overline{CLK}$)下降沿(CLK)：1 或 0

模式	操作
主模式	1、 选择 CKS 和 M1，M0=00，01，10 2、 选择 CSEN、MLS(同从模式) 3、 置位 SBEN 4、 写数据至 SBDR 寄存器 → 数据存储在 TXRX 寄存器中 → 输出 CLK 信号→ 到第 5 步 → (SIO 内部操作 → 数据存储在 TXRX 缓冲器中，SDI 上数据信号移入 TXRX 寄存器 → 数据传输完毕，TXRX 缓冲器数据加载到 SBDR 寄存器 5、 检查 WCOL；WCOL=1 → 清除 WCOL 并且跳转到第 4 步；WCOL=0 → 跳到第 6 步 6、 检查 TRF 或等待 SBI(串行接口中断) 7、 从 SBDR 寄存器读取数据 8、 清除 TRF 9、 跳转到第 4 步
从模式	1、 选择 M1，M0=11 2、 选择 CSEN、MLS(同从模式) 3、 置位 SBEN 4、 写数据至 SBDR 寄存器 → 数据存储在 TXRX 寄存器中 → 等待主机时钟(和$\overline{SCS}$)：CLK → 跳转到第 5 步 → (SIO 内部操作 → CLK()接收 → 输出数据存储在 TXRX 寄存器内，SDI 上数据移位到 TXRX 缓冲器内 → 数据发送完毕，TXRX 缓冲器数据加载到 SBDR 寄存器 5、 检查 WCOL；WCOL=1 → 清除 WCOL 并且跳转到第 4 步；WCOL=0 → 跳到第 6 步 6、 检查 TRF 或等待 SBI(串行接口中断) 7、 从 SBDR 寄存器读取数据 8、 清除 TRF 9、 跳转到第 4 步

WCOL：主/从模式下，若正在发送数据或接收数据，写 SBDR 寄存器将会置位 WCOL，且写入数据被忽略。WCOL 功能可由配置选项使能或禁止。WCOL 由硬件置位，软件清 0。

系统进入 HALT，串行接收和发送电路仍处于工作状态。

CPOL 用于选择 CLK 极性，可通过配置选项设定。

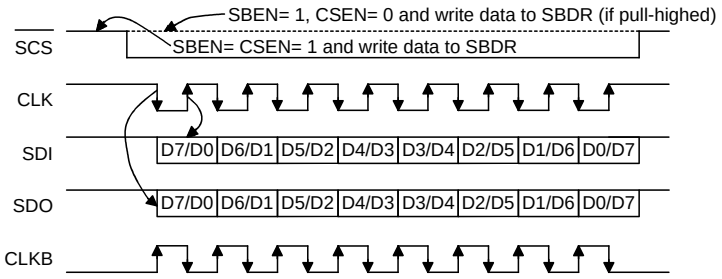
MLS：MSB 或 LSB 选择位。

CSEN：串行接口片选功能使能/禁止。若 CSEN=1，$\overline{SCS}$功能有效。主模式下，在 CLK 信号输出前先输出$\overline{SCS}$信号；而在从模式下，接收到$\overline{SCS}$信号前(后)，数据传输被禁止(使能)。若 CSEN=0，$\overline{SCS}$功能无效，$\overline{SCS}$(主模式和从模式)引脚必须浮空。有两个选项与 CSEN 相关：CSEN 功能可由配置选项使能/禁止。若 CSEN 由配置选项禁止，寄存器中 CSEN 将无效；若 CSEN 由配置选项使能，寄存器中 CSEN 有效。

当 SBEN=1，串行接口作于工作状态；$\overline{SCS}$(CSEN=1)=1；$\overline{SCS}$=浮空(CSEN=0)；SDI=浮空；SDO=1；主模式下 CLK=1/0(取决于 CPOL 配置选项)，从模式下 CLK=浮空。

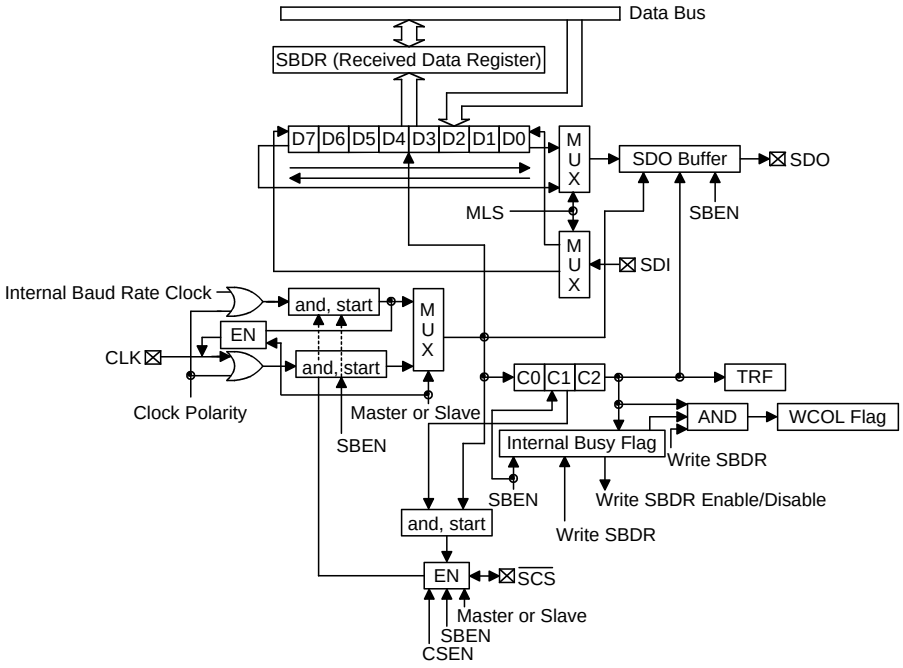
当 SBEN=0，串行接口禁止；$\overline{SCS}$=SDI=SDO=CLK=浮空。

TRF 由 SIO 置位，由用户软件清 0。当数据传输(发送和接收)完毕，TRF 将置位并触发串行接口中断。



	D7	D6	D5	D4	D3	D2	D1	D0
SBCR	CKS	M1	M0	SBEN	MLS	CSEN	WCOL	TRF
Default	0	1	1	0	0	0	0	0
SBDR	D7	D6	D5	D4	D3	D2	D1	D0
Default	u	u	u	u	u	u	u	u

Note: "u" means unchanged.



WCOL: set by SIO cleared by users
CSEN: enable/disable chip selection function pin
1. master mode 1/0: with/without SCSB output function
2. slave mode 1/0: with/without SCSB input control function
SBEN : enable/disable serial bus (0: initialize all status flags)
1. When SBEN= 0, all status flags should be initialized
2. When SBEN= 0, all SIO related function pins should stay at floating state
TRF 1: data transmitted or received, 0: data is transmitting or still not received
CPOL 1/0 : clock polarity rising/falling edge : mask option

唤醒

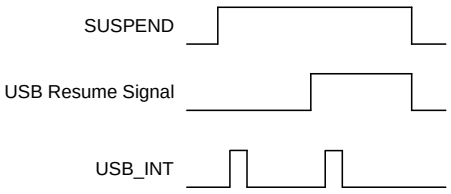
如果总线上 3ms 没有信号，HT46RB70 会进入挂起模式(消耗电流为 500μA)，挂起标志位(USC 寄存器第 0 位)将被置位，并触发 HT46RB70 产生 USB 中断。

为达到 500μA 的挂起电流，程序必须清除 USBCKEN(UCC 寄存器第 3 位)以禁止 USB 时钟，此时挂起电流为 400μA。

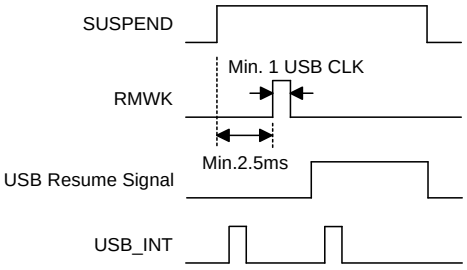
通过置位 SUSP2(UCC 寄存器第 4 位)，可将挂起电流进一步降至 250μA。但是如果 SUSP2 置位，必须保证配置选项中 LVR 功能关闭，否则 HT46RB70 将发生复位。

当检测到主机发出的恢复信号，HT46RB70 被唤醒，恢复标志位(USC 寄存器第 3 位)被置位并且触发 USB 中断。为使 HT46RB70 正常工作，程序必须置位 USBCKEN(UCC 寄存器第 3 位)和清除 SUSP2(UCC 寄存器第 4 位)。恢复标志位会在挂起信号发出前清为“0”。当 MCU 检测到有挂起信号时，恢复信号也必须注意。

当恢复信号完全发出，挂起标志位将被清零并触发 USB 中断，其时序图如下：



HT46RB50 可以通过 RMWK(USC 的第 1 位)发送信号来唤醒 USB 主机。当 USB 主机接收到 HT46RB50 发送的远程唤醒信，它会发出一个恢复位信号给设备。其时序图如下：



USB 接口

HT46RB70 有 6 个端点(EP0~EP5)。EP0~EP2 和 EP4 支持中断型传输，EP3 和 EP5 支持批量型传输。

有 14 个与 USB 功能相关的寄存器，它们分别为 USC(20H)、USR(21H)、UCC(22H) 、AWR(23H) 、STALL(24H) 、SIES(25H) 、MISC(26H) 、SETIO(27H) 、FIFO0(28H) 、FIFO1(29H) 、FIFO2(2AH)、FIFO3(2BH)、FIFO4(2CH)和 FIFO5(2DH)。

其 FIFO 大小分别为 8 字节(FIFO0)、8 字节(FIFO1)、8 字节(FIFO2)、64 字节(FIFO3)、8 字节(FIFO4)和 64 字节(FIFO5)，共 160 个字节。

URD(USC 的第 7 位)是 USB 复位信号控制位。

位	标号	R/W	功能
0	SUSP	R	只读，USB 挂起标志。当 USB 进入挂起状态，此位会被 SIE 置位。此位的任何变化都会触发中断。
1	RMWK	R/W	USB 远程唤醒命令。MCU 可置位此位，使 USB 主机离开挂起状态。置位 RMWK 使能远程唤醒。为确保 SIE 接收到命令，置位 RMWK 后必须有 2μs 延时。
2	URST	R/W	USB 复位标志。此位是由 SIE 置位/清除。若 URST 位被置位，表明发生了 USB 复位并且会触发 USB 中断。
3	RESUME	R	USB 恢复标志。当 USB 离开挂起状态，此位会被 SIE 置位并且持续 20ms，等待 MCU 检测。当 RESUME 位被 SIE 置位，MCU 会产生中断并唤醒 MCU。为了检测到挂起状态，MCU 必须置 USBCKEN 位和 SUSP2 位以使能 SIE 检测功能。SUSP 位清零同时 RESUME 也会被清除。当 MCU 检测 SUSP 时，必须考虑 RESUME 位。
4	V33C	R/W	0/1：关闭/打开 V33O 输出
5	PLL	R/W	0：打开 PLL(默认值)；1：关闭 PLL
6	—	—	未用，读出为“0”
7	URD	R/W	USB 复位信号控制位 1：USB 复位将复位 MCU 0：USB 复位不复位 MCU

USC(20H) 寄存器

USR(端点中断状态寄存器)寄存器，用来表明哪个端点被访问过。端点请求标志位(EP0IF、EP1IF、EP2IF、EP3IF、EP4IF 和 EP5IF)用来表明哪个端点被访问过。如果一个端点被访问过，相关的端点请求标志位被置“1”并触发 USB 中断，当 USB 中断被服务时，相应的端点请求标志位必须清零。

位	标号	R/W	功能
0	EP0IF	R/W	若此位被 SIE 置位，表明端点 0 被访问过并且触发 USB 中断。当进入中断服务程序，必须通过程序清除此位。
1	EP1IF	R/W	若此位被 SIE 置位，表明端点 1 被访问过并且触发 USB 中断。当进入中断服务程序，必须通过程序清除此位。
2	EP2IF	R/W	若此位被 SIE 置位，表明端点 2 被访问过并且触发 USB 中断。当进入中断服务程序，必须通过程序清除此位。
3	EP3IF	R/W	若此位被 SIE 置位，表明端点 3 被访问过并且触发 USB 中断。当进入中断服务程序，必须通过程序清除此位。
4	EP4IF	R/W	若此位被 SIE 置位，表明端点 4 被访问过并且触发 USB 中断。当进入中断服务程序，必须通过程序清除此位。
5	EP5IF	R/W	若此位被 SIE 置位，表明端点 5 被访问过并且触发 USB 中断。当进入中断服务程序，必须通过程序清除此位。
6~7	—	—	未用，读出为“0”

USR(21H) 寄存器

HT46RB70 还提供了一个系统时钟控制寄存器。它包括 USB 时钟控制位(USBCKEN)、挂起模式控制位(SUSP2)和系统时钟选择位(SYSCLK)。

可以通过下表中 EPS2、EPS1 和 EPS0 位选择端点。

位	标号	R/W	功能
0~2	EPS0~EPS2	R/W	选择所需操作的 FIFO，EPS2、EPS1、EPS0： 000：选择端点 0 的 FIFO 001：选择端点 1 的 FIFO 010：选择端点 2 的 FIFO 011：选择端点 3 的 FIFO 100：选择端点 4 的 FIFO 101：选择端点 5 的 FIFO 110：保留给以扩展，不能使用 111：保留给以扩展，不能使用 若选择的端点不存在，相关的功能无效。
3	USBCKEN	R/W	USB 时钟控制位。若此位置位，表示打开 USB 时钟；否则将关闭 USB 时钟。
4	SUSP2	R/W	在挂起模式中，可以通过此位进一步降低功耗。 正常模式，清除此位(默认值) 在暂停模式，置起此位可以降低功耗。
5	f_{SYS} (24MHz)	R/W	此位用于定义 MCU 系统时钟是来自外部振荡器还是来自 PLL 输出的 24MHz。 0：来自外部振荡器 1：来自 PLL 输出的 24MHz
6	SYSCLK	R/W	此位用于表明 MCU 使用的晶振频率。若使用 6M 晶振，必须置位；若使用 12M 晶振，必须清零(默认值)。
7	—	—	未用，读出为“0”

UCC(22H) 寄存器

AWR 寄存器由当前地址和远程唤醒功能控制位组成。AWR 寄存器的初始值为“00H”。写 AWR 寄存器直到 SETUP 阶段结束才会生效。

位	标号	R/W	功能
0	WKEN	R/W	远程唤醒功能打开/关闭
7~1	AD6~AD0	R/W	USB 设备地址

AWR(23H) 寄存器

STALL 寄存器用于表明相应的端点工作是否正常。若端点工作存在异常，STALL 寄存器相应的位必须置“1”。USB 复位将会清除 STALL 寄存器。

位	标号	R/W	功能
5~0	STL5~STL0	R/W	当相关 USB 端点停顿时，这些位由用户设置。通过 USB 复位信号及 Setup Token 事件来清零。
7~6	—	—	未用，读出为“0”

STALL(24H) 寄存器

位	标号	R/W	功能
0	ASET	R/W	SIE 自动从 AWR 寄存器中加载设备地址控制位。若此位由程序置位，当主机成功地从设备中读取到数据时，SIE 将会从 AWR 寄存器中加载地址。否则，若此位清零，写入 AWR 寄存器中地址将会立即生效。所以，当接收到 SETUP 封包，程序必须清除此位。
1	ERR	R/W	访问 FIFO0 时出现错误。由 SIE 置位，软件清零。
2	OUT	R/W	接收到 OUT 封包(除 0 封包)。读取 OUT 数据后，由程序清除此位。在 SIE 接收到下次有效的 SETUP 封包也会清除此位。
3	IN	R	接收到 IN 封包。(1=IN 封包；0=没有 IN 封包)
4	NAK	R/W	在 IN 和 OUT 的状态阶段，SIE 返回 NAK 信号给主机。(1=NAK 信号；0=没有 NAK 信号)
5	CRCF	R/W	CRC、PID 和未接收到完整的封包标志位。CRCF 标志由 SIE 置位，软件清零。
6	EOT	R	封包标志位，低有效。
7	NMI	R/W	NAK 封包中断控制位。若此位置位，当设备发送 NAK 封包给主机，将不会触发中断。否则，当设备发送 NAK 封包给主机，将触发中断。

SIES(25H) 寄存器

MISC 寄存器由命令和状态组成，用于控制端点 FIFO 的读写并反映 FIFO 的状态。USB 复位将清除 MISC 寄存器。

位	标号	R/W	功能
0	REQUEST	R/W	选择所需端点并将此位置高，便可访问相应的 FIFO。操作相应的 FIFO 后，此位必须清零。
1	TX	R/W	方向控制位。若置位，MCU 可以写数据到 FIFO 内，操作相应的 FIFO 后，必须在清除 REQUEST 位前将其清零；若清零，MCU 可以从 FIFO 内读取数据，同样在清除 REQUEST 位前需将其清零。
2	CLEAR	R/W	用于清除所选择的 FIFO，即使 FIFO 尚未准备好。
3~4	—	—	保留
5	SETCMD	R/W	表示 FIFO 内数据为 SETUP 命令。此位将会保持到下次 FIFO 被访问。(1=SETCMD 封包；0=没有 SETCMD 封包)
6	READY	R	所选择 FIFO 已准备好。 (1=已准备好；0=未准备好)
7	LED0	R/W	表示主机发送了 0 封包给 MCU。读取相应的 FIFO 后必须清零。(1=主机发送了 0 封包)

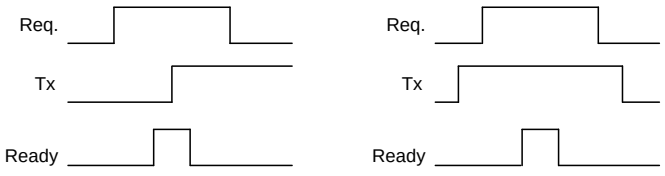
MISC(26H) 寄存器

读写 FIFO 时必须按照一定时序进行。通过设定 MISC 寄存器位，MCU 对 FIFO 可以执行读、写和清除动作。下表为一些读、写和清除 FIFO 的时序。

动作	MISC 设置和状态
读 FIFO0 时序	00H→01H→延时 2μs，检查是否为 41H(准备好)或 01H(未准备好)→从 FIFO0 寄存器读取*→03H→02H
写 FIFO0 时序	02H→03H→延时 2μs，检查是否为 43H(准备好)或 03H(未准备好)→写数据到 FIFO0 寄存器*→01H→00H
检查 FIFO0 是否可读	00H→01H 延时 2μs，检查是否为 41H(准备好)或 01H(未准备好)→00H
检查 FIFO0 是否可写	02H→03H 延时 2μs，检查是否为 43H(准备好)或 03H(未准备好)→02H
从 FIFO0 内读取 0 封包	00H→01H→延时 2μs，检查是否为 81H→读取(01H) →03H→02H
写 0 封包到 FIFO0 内	02H→03H→延时 2μs，检查是否为 03H→07H→06H→00H

读或写 FIFO

注意：*：在两次读与写之间必须存在 2μs 延时



Read FIFO Timing **Write FIFO Timing**

位	标号	R/W	功能
0	DATATG*	R/W	触发此位，所有 DATA 封包都会从 DATA0 开始
1	SETIO1**	R/W	设置端点 1 为输入或输出(1/0)，默认为输入(1)
2	SETIO2**	R/W	设置端点 2 为输入或输出(1/0)，默认为输入(1)
3	SETIO3**	R/W	设置端点 3 为输入或输出(1/0)，默认为输入(1)
4	SETIO4**	R/W	设置端点 4 为输入或输出(1/0)，默认为输入(1)
5	SETIO5**	R/W	设置端点 5 为输入或输出(1/0)，默认为输入(1)
6~7	—	—	未用，读出为“0”

SETIO(27H) 寄存器

注意：*USB 设置：当主机发出“set Configuration”，数据管道会先发出 DATA0。所以，当设备接收到“set Configuration”封包，必须设置此位以确保数据从 DATA0 开始。

**必须设置数据管道为输入管道还是输出管道。主要用于防止主机错误的发送 IN 或 OUT 封包或禁止端点。

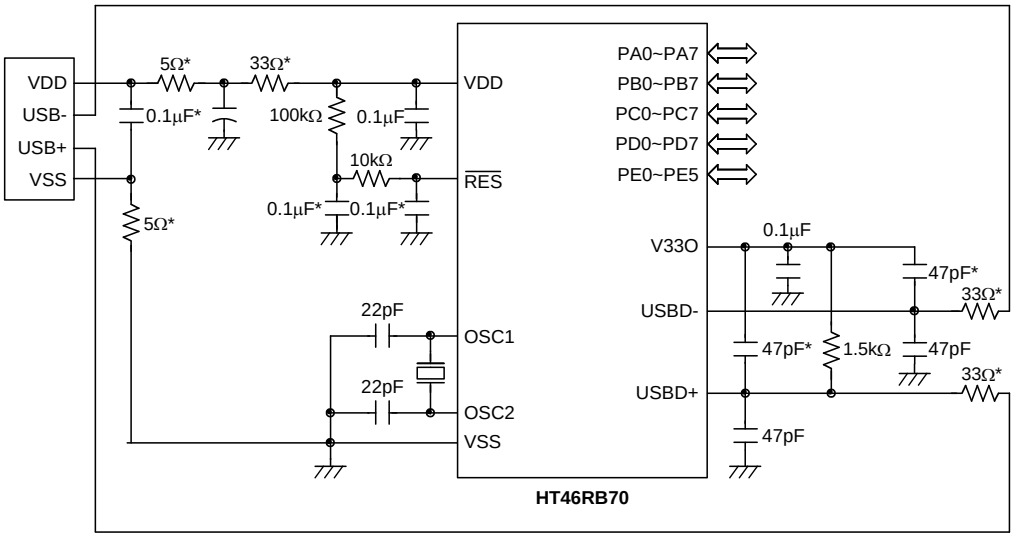
配置选项

下表列出了所有配置选项。所有选项必须正确定义，以保证统正常运行。

编号	选项
1	PA0~PA7 上拉电阻使能位(按位定义)
2	PB0~PB7 上拉电阻使能位(按位定义)
3	PC0~PC7 上拉电阻使能位(半字节定义)
4	PD0~PD7 上拉电阻使能位(半字节定义)
5	PE0~PE5 上拉电阻使能位(半字节定义)
6	LVR 使能或禁止位
7	PWM 选择：(7+1)或(6+2)模式 PD0：电平输出或 PWM 输出 PD1：电平输出或 PWM 输出 PD2：电平输出或 PWM 输出 PD3：电平输出或 PWM 输出
8	SIO(串行接口)使能或禁止(若 SIO 使能，PE0~PE3 输入/输出功能将禁止)
9	SIO_CPOL：时钟极性 1/0：上升沿有效或下降沿有效
10	SIO_WCOL：使能/禁止
11	SIO_CSEN：使能/禁止，CSEN 选项用于使能或/禁止 CSEN 功能。
12	WDT 使能/禁止
13	WDT 时钟源：f_S /4 或看门狗振荡器
14	WDT 预分频选择：$2^{12}/f_S$~$2^{13}/f_S$、、$2^{13}/f_S$~$2^{14}/f_S$ 、$2^{14}/f_S$~$2^{15}/f_S$ 和 $2^{15}/f_S$~$2^{16}/f_S$
15	清除看门狗指令条数：1 条或 2 条
16	PA0~PA7 唤醒使能/禁止(按位定义)
17*	EP1~EP5 数据管道使能：EP1、EP2、EP3、EP4、EP5 使能/禁止。(默认为使能)

注意：*：此配置选项可用于使能/禁止相关的端点。

应用电路



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。
X1 可使用 6MHz 或 12MHz，X1 尽量靠近 OSC1 和 OSC2
“*”为了避免噪声干扰，连接$\overline{\text{RES}}$引脚的线请尽可能地短
若使用谐振器必须连接 22pF 的电容

指令集摘要

下表中说明了按功能分类的指令集，用户可以将该表作为基本的指令参考。

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
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

助记符		说明	指令周期	影响标志位
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				
TABRD	[m]	读取特定页的 ROM 内容，并送至数据存储器和 TBLH	2 注	无
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，否则 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

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： TO ← 0
PDF ← 1
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： [m]←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

TABRD [m] Move the ROM code(specific page) to TBLH and data memory
说明： 将表格指针 TBLP 和 TBHP 所指的程序代码低字节（特定页）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

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

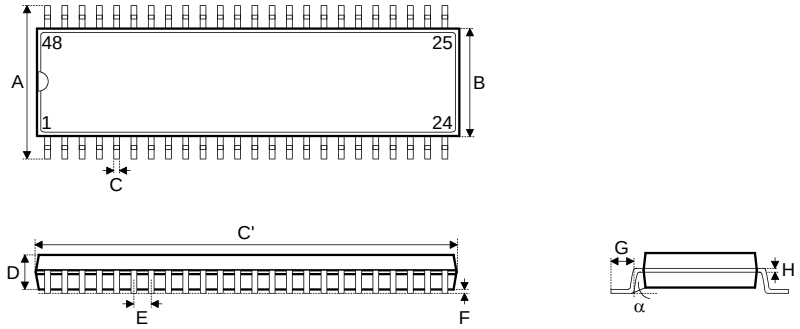
封装尺寸

请注意，这里提供的封装信息仅作为参考。由于这个信息经常更新，提醒用户咨询 Holtek 网站以获取最新版本的封装信息。

封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

- 封装信息（包括外形尺寸、包装带和卷轴规格）
- 封装材料信息
- 纸箱信息

48-pin SSOP (300mil)外形尺寸



符号	尺寸 (单位：inch)		
	最小	典型	最大
A	0.395	—	0.420
B	0.291	0.295	0.299
C	0.008	—	0.014
C’	0.620	0.625	0.630
D	0.095	0.102	0.110
E	—	0.025 BSC	—
F	0.008	0.012	0.016
G	0.020	—	0.040
H	0.005	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小	典型	最大
A	10.03	—	10.67
B	7.39	7.49	7.59
C	0.20	—	0.34
C’	15.75	15.88	16.00
D	2.41	2.59	2.79
E	—	0.64 BSC	—
F	0.20	0.30	0.41
G	0.51	—	1.02
H	0.13	—	0.25
α	0°	—	8°

Copyright © 2017 by HOLTEK SEMICONDUCTOR INC.

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