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

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

技术相关信息

- 工具信息
- FAQs
- 应用范例
  - －HA0003S HT48 & HT46 MCU 与 HT93LC46 的通信
  - －HA0049S HT1380 的读写控制
  - －HA0052S HT46R47 之应用－电池充电器
  - －HA0075S MCU 重置电路及振荡电路应用

特性

- 工作电压：
  f_{SYS}=4MHz：2.2V~5.5V
  f_{SYS}=8MHz：3.3V~5.5V
- 最多 48 个双向输入/输出口
- 与输入/输出口共用引脚的外部中断输入
- 1 个 8 位定时/计数器
- 2 个 16 位可编程定时/计数器
- 32K×16 位程序存储器（4 个区块）
- 768×8 位数据存储器（4 个区块）
- 内置晶体和 RC 振荡电路
- 看门狗定时器
- 具有 PFD 功能，可用于产生音频
- 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
- 16 层硬件堆栈
- 8 通道 12 位分辨率的 A/D 转换器
- 4 通道 8 位与输入/输出共用引脚的 PWM 输出
- 具有 8 位预分频器的 RTC
- 通用异步接收/发送器(UART)
- I^2C 总线(slave 模式)
- SPI 总线
- 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
- 低电压复位功能
- 48/56-pin SSOP 封装

概述

HT46RU26/HT46CU26 是 8 位高性能精简指令集单片机，专门为需要 A/D 转换产品而设计，例如传感器信号输入。

单片机包含一个集成的多通道模数转换口和四个脉冲宽度调制输出。结合 HOLTEK 单片机的特性，如暂停和唤醒功能，振荡器选项，可编程分频等等，确保了使用者需要最少的外围设备的需求。

拥有着充分集成的 UART 功能，SPI 和 I^2C 接口，提供了单片机简单和有效地与外部 PC 和其他外围硬件的连接的便利。

低功耗、I/O 使用灵活、可编程分频器、计数器、振荡类型选择、多通道 A/D 转换、脉冲测量功能、I^2C 接口、UART 总线，暂停和唤醒功能，使这款单片机可以广泛应用于传感器的 A/D 转换、马达控制、工业控制、消费类产品，子系统控制器等。

HT46CU26 正在研发中，并将很快提供。

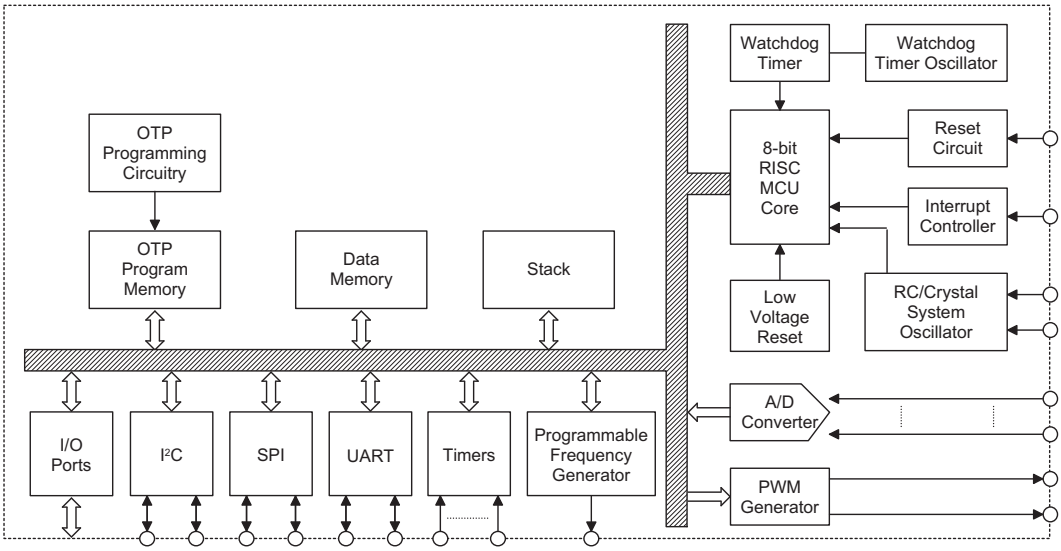
设备类型

在批号上带有“R”的单片机，显示它们为 OTP 单片机，使用盛群开发和程序设计工具，提供了简单和有效的程序资料的优势。此类单片机提供给设计者快速和低成本产品的开发周期的方法。在批号上带有“C”的单片机，显示它们为掩膜单片机。在设计者拥有成熟的设计程序和高要求的体积和低成本要求的应用下，此类单片机提供了补充的设备。

和 OTP 单片机完全一致的引脚和兼容的功能，对于超过开发周期和面临降低成本要求的产品，掩膜单片机提供了它们理想的替代品。

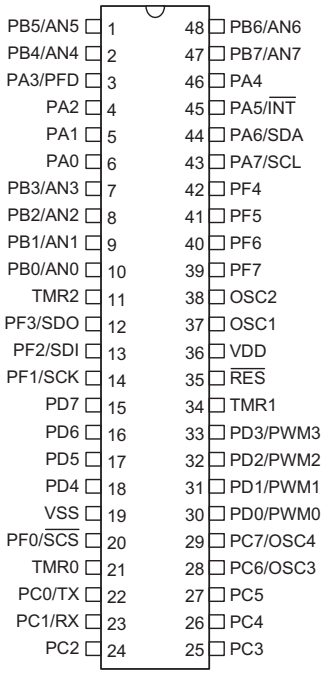
在此规格书内，为了简便，当描述单片机功能时，只有 OTP 型号的被涉及到，但是，相同的描述功能同样应用于掩膜单片机。

方框图

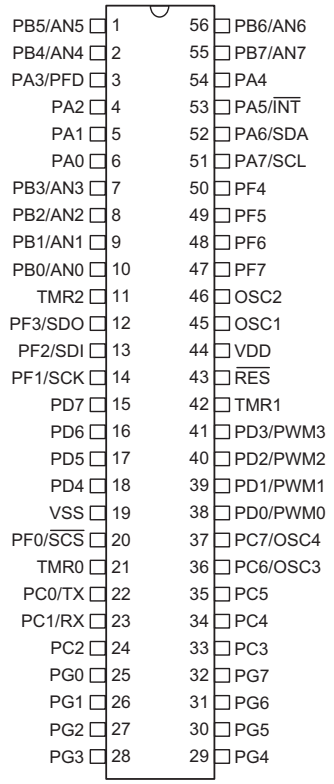


注意：本系统框线图为 OTP 单片机，至于 MASK 型单片机则没有 Device Programming Circuity。

引脚图



HT46RU26/HT46CU26
48 SSOP-A



HT46RU26/HT46CU26
56 SSOP-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 PA5/$\overline{INT}$ PA6/SDA PA7/SCL	输入/输出	上拉电阻 唤醒功能 PFD I^2C 总线	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、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。 PFD、$\overline{INT}$ 分别与 PA3、PA5 共用引脚。一旦使用 I^2C 总线功能，则与 PA6、PA7 相关的寄存器将被禁用。
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、或 A/D 输入。 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/TX PC1/RX PC2~PC5 PC6/OSC3 PC7/OSC4	输入/输出	上拉电阻 OSC3/OSC4	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：端口选择）的斯密特触发输入。 TX 和 RX 分别与 PC0 的 PC1 共用引脚，一旦使用 UART 功能，则与 PC0 和 PC1 相关的寄存器将被禁用。 RTC 振荡器引脚 OSC3 和 OSC4 分别与 PC6 和 PC7 共用引脚（由 OSC 相关掩膜选项决定）。一旦选择 RTC 振荡的掩膜选项，32768Hz 晶振将连接引脚 OSC3 和 OSC4。
PD0/PWM0 PD1/PWM1 PD2/PWM2 PD3/PWM3 PD4~PD7	输入/输出	上拉电阻 PWM	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：端口选择）的斯密特触发输入。 PWM0/PWM1/PWM2/PWM3 分别与 PD0/PD1/PD2/PD3 共用引脚（由 PWM 相关掩膜选项决定）。
PF0/$\overline{SCS}$ PF1/SCK PF2/SDI PF3/SDO PF0~PF7	输入/输出	上拉电阻 SIO	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：端口选择)的斯密特触发输入。SPI 接口引脚$\overline{SCS}$，SCK，SDO 和 SDI 分别与 PF0~PF3 共用引脚
PG0~PG7	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：端口选择)的斯密特触发输入。
TMR0	输入	—	定时/计数器 0 斯密特触发输入(不带上拉电阻)。
TMR1	输入	—	定时/计数器 1 斯密特触发输入(不带上拉电阻)。
TMR2	输入	—	定时/计数器 2 斯密特触发输入(不带上拉电阻)。
$\overline{RES}$	输入	—	斯密特触发复位输入，低电平有效。
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。
VSS	—	—	负电源，接地。
VDD	—	—	正电源。

注：单独的引脚能被选择为带上拉电阻。

引脚 PG 不存在于 48－pin 的封装。

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，RC 振荡)	3V	无负载,f_{SYS}=4MHz ADC 关闭,UART 关闭	—	1	2	mA
		5V		—	2.5	5	mA
I_{DD2}	工作电流 (晶体振荡，RC 振荡)	3V	无负载,f_{SYS}=4MHz ADC 关闭,UART 打开	—	1.5	3	mA
		5V		—	3	6	mA
I_{DD3}	工作电流 (晶体振荡，RC 振荡)	5V	无负载,f_{SYS}=8MHz ADC 关闭,UART 关闭	—	4	8	mA
I_{DD4}	工作电流 (晶体振荡，RC 振荡)	5V	无负载,f_{SYS}=8MHz ADC 关闭,UART 打开	—	5	10	mA
I_{DD5}	工作电流 (f_S ＝RTC 振荡)	3V	无负载 ADC 关闭,UART 关闭	—	0.3	0.6	mA
		5V			0.6	1	mA
I_{STB1}	静态电流(看门狗打开，f_S ＝WDT 振荡)	3V	无负载，系统 HALT	—	2	5	μA
		5V		—	6	10	μA
I_{STB2}	静态电流(看门狗打开，f_S ＝RTC 振荡)	3V	无负载，系统 HALT	—	2.5	5	μA
		5V		—	10	20	μA
I_{STB3}	静态电流(看门狗关闭)	3V	无负载，系统 HALT，UART 关闭	—	—	1	μA
		5V		—	—	2	μA
V_{IL1}	I/O、TMR 和 $\overline{INT}$ 引脚低电压输入	—	—	0	—	0.3 V_{DD}	V
V_{IH1}	I/O、TMR 和 $\overline{INT}$ 引脚高电压输入	—	—	0.7V_{DD}	—	V_{DD}	V
V_{IL2}	$\overline{RES}$ 引脚低电压输入	—	—	0	—	0.4V_{DD}	V
V_{IH2}	$\overline{RES}$ 引脚高电压输入	—	—	0.9V_{DD}	—	V_{DD}	V
V_{LVR}	低电压复位	—	掩膜选项：2.1V	1.98	2.10	2.22	V
			掩膜选项：3.15V	2.98	3.15	3.32	V
			掩膜选项：4.2V	3.98	4.20	4.42	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
I_{ADC}	打开 ADC 增加的功耗	3V	—	—	0.5	1	mA
		5V		—	1.5	3	mA
DNL	ADC 微分非线性	5V	t_{AD}=1μS	—	—	±2	LSB
INL	ADC 积分非线性	5V	t_{AD}=1μS	—	±2.5	±4	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	—	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}	看门狗振荡器周期	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	外部低电平复位脉宽	—	—	1	—	—	μs
t_{SST}	系统启动延迟时间	—	从 HALT 状态唤醒	—	1024	—	$*t_{SYS}$
t_{LVR}	低电压复位时间	—	—	0.25	1	2	ms
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}
t_{IIC}	I^2C 总线时钟周期	—	—	64	—	—	$*t_{SYS}$

注：$*t_{SYS}=1/f_{SYS}$

系统结构

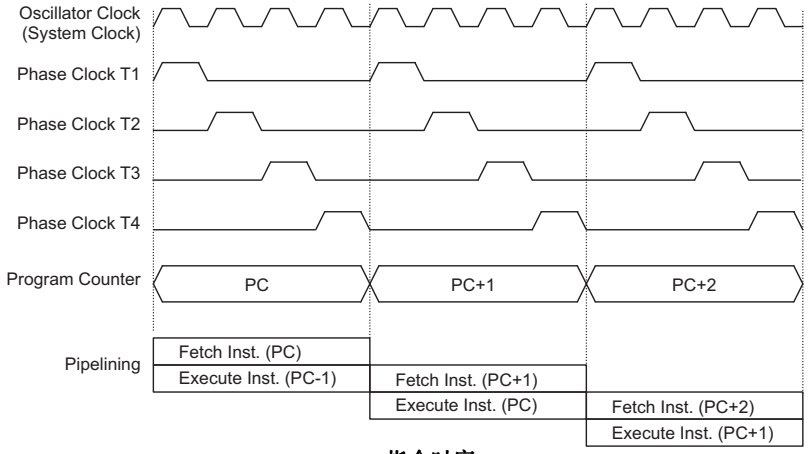
内部系统结构是 HOLTEK 单片机具有良好运行性能的主要因素。由于采用 RISC 结构，此系列单片机具有高运算速度和高性能的特性。通过流水线的方式，指令的取得和执行同时进行，此举使得除了分支和调用指令外，其它指令都能在一个指令周期内完成。8 位的 ALU 参与指令集中所有的运算，它可完成算术运算、逻辑运算、移位、加、减和分支等功能，而内部的数据存取则以通过累加器或 ALU 的方式加以简化。有些寄存器在数据存储器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供最大可靠度和灵活性的 I/O 和 A/D 控制系统时，仅需要少数的外部器件。这使这些有着外部模拟信号输入接口的单片机，譬如与传感器连接，适合在低成本高产量的控制应用上。

时序和流水线结构

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

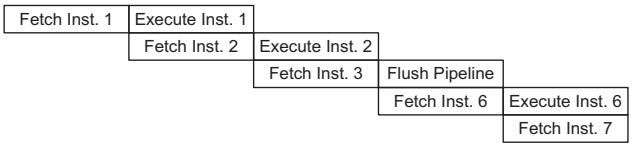
当使用 RC 振荡器时，OSC2 可以如同一个 T1 相时钟同步引脚一样地被使用，这个 T1 相时钟有 $f_{SYS}/4$ 的频率，拥有 1：3 高/低的占空比。

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



指令时序

```
1          MOV A,[12H]
2          CALL DELAY
3          CPL [12H]
4          :
5          :
6  DELAY:  NOP
```



指令捕捉

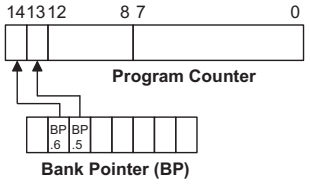
程序计数器 — PC

程序执行期间，程序计数器用来指向下一条要执行的指令地址。除了 JMP 或 CALL 这些要求跳转到一个非连续的程序存储器地址之外，它会在每条指令执行完后自动增加一。然而必须要注意只有低 8 位，即程序计数器低字节寄存器 PCL，是可以让使用者直接读写的。

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

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

程序计数器分在四个区块，注意的都是，由区块控制器的第 5 和第 6 位控制选择。这两位控制着程序计数器的最高地址，如下图所示。



模式	程序计数器														
	***14**	***13**	***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	0	0
外部，A/D 转换或 SPI 中断－选模选项选择	0	0	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	0	0	1	0	0	0
定时/计数器 1 溢出	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
UART 中断	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
I^2C 总线或 SPI 中断－掩膜选项选择	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0
多功能中断	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
条件跳转	PC+2（包括当前 bank）														
装载 PCL	*14	*13	*12	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
跳转，子程序调用	BP.6	BP.5	#12	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
子程序返回	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

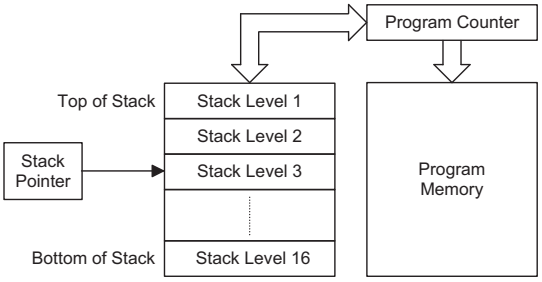
程序计数器

注： 掩膜选项选择部分中断向量功能
PC14~PC8：程序计数器当前位
@7~@0：PCL 字节
BP.5,BP.6：区块指针位
#12~#0：指令地址位
S14~S0：堆栈指针

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



算术逻辑单元 — ALU

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

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

程序存储器

程序存储器用来存放用户代码即储存程序。HT46RU26/HT46CU26 程序存储器是一次可编程（OTP），使用者可编写他们的应用代码到单片机中。使用适当的编程工具，OTP 单片机可以提供使用者灵活的方式来自由开发他们的应用，这对于除错或需要经常升级与改变程序的产品是很有帮助的。对于中小型量产，OTP 亦为极佳的选择。

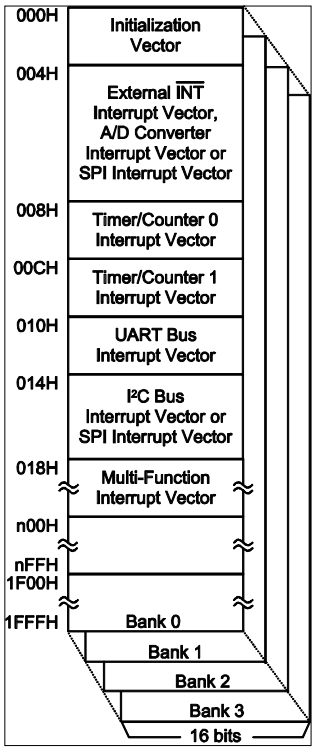
结构

16 位的程序存储器的容量是 32K。程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口，数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。程序存储器被细分成 4 个容量为 8K 的独立的区块。程序存储器区块通过使用 BP 选择。注意的是，当操作 BP 时，它也可用作控制数据存储器的数据指针。

特殊向量

程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。

- 地址 000H
  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，A/D 转换中断或 SPI 中断（由掩膜选项设置）服务程序保留。当单片机的$\overline{INT}$引脚为低电平，A/D 信号转换完成或 SPI 总线上 8 位数据已经被发送，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。
- 地址 010H
  该地址为 UART 中断服务程序保留。如果 UART 中断发生，由发生数据寄存器为空，接收寄存器状态，数据发送完成，过速错误或地址检测产生，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 I^2C 总线中断和 SPI 中断（由掩膜选项设置）服务程序保留。如果 I^2C 或 SPI 中断发生，由 I^2C 或 SPI 总线从器件地址匹配或收到一个 8 位的数据产生，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
  该地址为多功能中断服务程序保留。如果多功能中断发生，由定时/计数器 2 溢出，实时时钟溢出或时基溢出产生，如果中断允许且堆栈未满，则程序会跳转到 018H 地址开始执行。



程序存储器结构

多Bank管理

HT46RU26/HT46CU26 有多个程序存储Bank，一些特殊的事项需要注意。首先，伪指令ROMBANK 将程序段放置于不同的Bank 中。当执行"CALL"指令来调用位于不同Bank 的子程序，或者执行"JMP"指令来跳转到不同Bank 的地址时，必须先正确设置Bank 指针来确定目标Bank。伪指令ROMBANK 最佳用法如下所列。当执行"CALL"或"JMP"指令，BP 特殊寄存器中的Bank 指针会自动装载到程序计数器中。若子程序被位于另一个Bank 的主程序调用，当执这个子程序的"RET"指令时，程序会自动返回到原先主程序所在的Bank，然而，BP 的值不会改变，它仍保持着子程序所在Bank 的值。因此在不同的Bank 之间的转移过程中，BP 必须小心处理。

以下的范例就是母体HT46RU26/HT46CU26的程序，说明了如何在不同的Bank 之间使用"CALL"和"JMP"指令。

```
include HT46RU26.inc
 :
rombank 0 codesec0                  ;define rombank0
rombank 1 codesec1                  ;define rombank1
 :
codesec0 .section at 000h 'code'    ; locates the following program section into Bank0
clr bp                               ; re-initializing the BP
jmp start
 :
start:
 :
 :
lab0:
 :
 :
mov a,BANK routb1                    ; routine"routb1"is locate in Bank 1
mov bp,a                             ; load bank number for routb1 into BP
call roub1                           ; call subroutine located in Bank 1
clr bp .5                            ; program will return to this location
                                     ; after RET in Bank 1
 :                                   ;but BP will retain Bank 1 value;
 :                                   ; so clear the BP
codesec1 .section at 000h 'code'    ; locates following program section into Bank 1
 :
 :
routb1 proc
 :
ret                                  ; return program to Bank0 but BP will
                                     ; retain Bank 1 value
routb1 endp
 :
```

中断处理时，Bank指针的处理需十分小心。无论程序运行至哪个Bank，一旦中断发生，包括外部中断和内部中断，程序都会立刻跳转至相应的位于Bank 0的中断子程序入口。然而要注意的是，虽然无论什么情况下程序都会跳转到Bank0，但是Bank指针仍然保持原来的值，而不是指向Bank 0。因为这个原因，进入中断处理子程序后，除了保存累加器和状态寄存器之外，另一个重要的动作就是，保存Bank指针以及清Bank指针让它指向Bank 0，尤其是在Bank 0中执行调用子程序或跳转指令。在"RETI"指令执行以前，Bank指针和累加器、状态特殊寄存器一样必须被恢复，确保程序返回至正确的Bank地址，并指向此地址。以下范例说明了如何处理中断子程序：

```
include HT46RU26.inc
 :
 :
rombank 0 codesec0                   ; define rombank 0
```

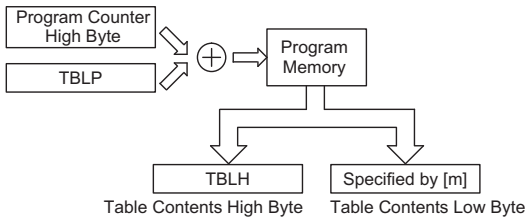
```
rombank 1 codesec1                  ; define rombank 1
  :
  :
codesec0 .section at 000h 'code'    ; locates the following program section into Bank 0
clr bp                               ; clear bank pointer after power-on reset
  :
  :
org 004h                            ; jump here from any bank when ext . int.
                                    ; occurs - BP retains original value
mov accbuf0,a                       ; backup accumulator
mov a,bp                             ; backup bank pointer
clr bp                               ; clear bp to indicate Bank 0 otherwise
                                     ; original BP value will remain and give
                                     ; rise to false jmp or call addresses
jmp ext_int                          ; jump to external interrupt subroutine
  :
  :
org 008h                             ; jump here from any bank when timer 0 int
                                    ; occurs - BP retains original value
mov accbuf1,a                        ; backup accumulator
mov a,bp                             ; backup bank pointer
clr bp                               ; clear bp to indicate Bank 0 otherwise
                                     ; original BP value will remain and give
                                     ; rise to false jmp or call addresses
jmp tim0_int                         ; jump to timer 0 interrupt subroutine
  :
  :
org 00Ch                             ; jump here from any bank when timer 1 int
                                    ; occurs - BP retains original value
  :
  :
ext_int:                             ; external interrupt subroutine
mov bp_exti,a                        ; backup bank pointer
mov a,status                        ; backup status register
mov statusbuf1,a                     ; backup status register
  :
  :
mov a,statusbuf0                     ; restore status register
mov status,a
mov a,bp_exti                       ; restore bank pointer
mov bp,a
mov a,accbuf1                        ; restore accumulator
reti                                 ; return to main program and original
  :
  :
```

查表

程序存储器中的任何地址都可以定义成一个表格，以便储存固定的数据。使用表格时，表格指针必须先行设定，其方式是将表格的低字节地址放在表格指针寄存器 TBLP 中，而高字节被传送到表格内容高字节寄存器 TBLH。在任何区块中，这些寄存器定义了全部的查表地址。

在设定完表格指针后，表格数据可以使用“TABRDC [m]”或“TABRDL [m]”指令从当前的程序所在的存储器页或存储器最后一页中来查表读取。当这些指令执行时，程序存储器中表格数据低字节，将被传送到使用者所指定的数据存储器[m]，程序存储器中表格数据的高字节，则被传送到 TBLH 特殊寄存器，而高字节中未使用的位将被读取为“0”。

下图是查表中寻址/数据流程：



查表范例

以下范例说明单片机中，表格指针和表格数据如何被定义和执行。这个例子使用的表格数据用ORG伪指令储存在存储器的最后一页，在此ORG伪指令中的值为000H，但这是相对Bank1起止地址而言，绝对地址是2000H。这里高字节表格指针是20H，低字节表格指针的初值则为05H。这可以保证从数据表格读取的第一笔数据位于程序存储器地址2005H，即ORG 伪指令定义地址后五个地址。执行“TABRDC [m]”指令，表格数据低字节“FFH”被送往指定的“temp”寄存器，而表格数据高字节“55H”将会被传送到TBLH 寄存器。

```
include HT46RU26.inc
 :
 :
data .section 'data'
temp db ?
 :
 :
rombank 0 codesec0                    ; Bank 0 definition
rombank 1 codesec1                    ; Bank 1 definition
 :
 :
codesec0 .section at 0 'code'
jmp start
 :
org 010h
start:
 :
 :
mov a,020h                            ; setup table high byte address
mov tbhp,a
```

```
mov a,005h                          ; setup table low byte address
mov tblp,a                          ; table pointer address is now 2005H
tabrdc temp                         ; read table data from PC address 2005H
nop                                  ; FFH will be placed in the temp
                                     ; register and 55H will be placed in the TBLH register
codesec1 .section at 000h 'code'     ; Bank 1 code located here
org 0000h                            ; this defines the offset from the start address of Bank 1
                                     ; which is 2000H
dc 000AAh, 011BBh, 022CCh, 033DDh, 044EEh, 055FFh
:
```

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

指令	表格地址								
	b14～b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC [m]	TBHP	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1111111	@7	@6	@5	@4	@3	@2	@1	@0

表格区

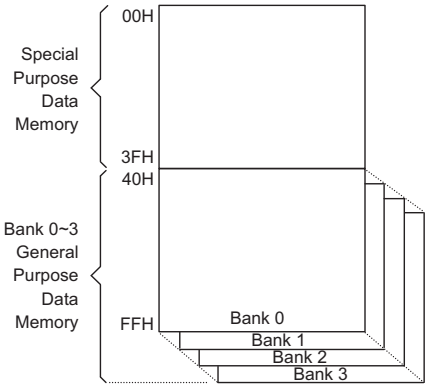
注：PC14~PC8：程序指针当前位
@7~@0：表格指针 TBLP 位

数据存储器

数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。分为两部份，第一部份是特殊功能寄存器，这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二部份数据存储器是作一般用途使用，都可在程序控制下进行读取和写入。作一般用途使用的数据存储器分为两个区块，即 BANK0～BANK3。区块之间的转换是通过设置区块指针成恰当的值完成的。

结构

数据存储器的两个部份，即特殊和通用数据存储器，位于连续的地址。全部 RAM 为 8 位宽度。所有单片机的数据存储器的起始地址都是 00H。特殊数据存储器映射到各自的区块，因此能后在各自的区块内被读取。



数据存储器结构

注意：除部分特殊位，大部分数据存储区可以通过“SET [m].i”和“CLR [m].i”直接寻址。数据存储器也可以通过指针 MP0 和 MP1 间接寻址。

通用数据存储器

所有的单片机程序需要一个读/写的存储区，让临时数据可以被储存和再使用，该 RAM 区域就是通用数据存储器。这个数据存储区可让使用者进行读取和写入的操作。使用“SET [m].i”和“CLR [m].i”指令可对个别的位做置位或复位的操作，方便用户在数据存储器内进行位操作。由于通用数据存储器分为四个区块，有必要在存取通用数据存储器前设置区块指针为恰当的值。

特殊数据存储器

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

00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	RTCC
0AH	STATUS
0BH	INTC0
0CH	TMR0H
0DH	TMR0L
0EH	TMR0C
0FH	TMR1H
10H	TMR1L
11H	TMR1C
12H	PA
13H	PAC
14H	PB
15H	PBC
16H	PC
17H	PCC
18H	PD
19H	PDC
1AH	PWM0
1BH	PWM1
1CH	PWM2
1DH	PWM3
1EH	INTC1
1FH	TBHP
20H	HADR
21H	HCR
22H	HSR
23H	HDR
24H	ADRL
25H	ADRH
26H	ADCR
27H	ACSR
28H	PF
29H	PFC
2AH	PG
2BH	PGC
2CH	
2DH	TMR2
2EH	TMR2C
2FH	MFIC
30H	USR
31H	UCR1
32H	UCR2
33H	TXR/RXR
34H	BRG
35H	
36H	
37H	
38H	SBCR
39H	SBDR
3AH	
3BH	
3CH	
3DH	
3EH	
3FH	

: Unused, read as "00"

特殊数据存储器

特殊功能寄存器

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

间接寻址寄存器 — IAR0，IAR1

IAR0 和 IAR1 寄存器位于数据存储区，并没有实际的物理地址。间接寻址的方法准许使用间接寻址指针做数据操作，以取代定义实际存储器地址的直接存储器寻址方法。在间接寻址寄存器上的任何动作，将对间接寻址指针(MP0 和 MP1)所指定的存储器地址产生对应的读/写操作。直接读取 IAR0 和 IAR1 寄存器将返回 00H 的结果，而直接写入此寄存器则不做任何操作。

间接寻址指针 — MP0，MP1

间接寻址指针 MP0 和 MP1 位于数据存储器中，能象普通的寄存器一般被写入和操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由间接寻址指针所指定的地址。

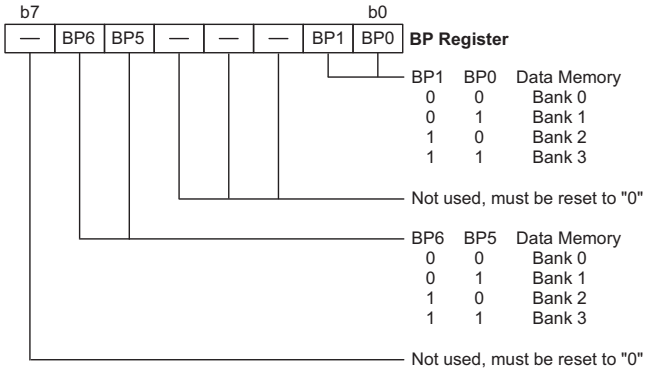
以下例子说明如何清除一个具有 4 RAM 地址的区块，它们已事先定义成地址 adres1 到 adres4。

```
data .section ´data´
adres1   db ?
adres2   db ?
adres3   db ?
adres4   db ?
block            db ?
code .section at 0 ´code´
org 00h
start:
    mov a,04h            ; setup size of block
    mov block,a
    mov a,offset adres1  ; Accumulator loaded with first RAM address
    mov mp0,a            ; setup memory pointer with first RAM address

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

continue:
```

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。

存储区指针

程序存储器和数据存储器都被分为四个独立的区块。可以使用存储区指针 BP 来选择对应的存储器区块。BP 的第 5 和 6 位选择程序存储器区块，同时 BP 的第 0 和 1 位选择程序存储器区块。复位后，通用数据存储器会初始化到 BANK0，但是在 HALT 模式下的 WDT 溢出复位，不会改变通用数据存储器的存储区块。请注意，特殊功能数据存储器不受存储区的影响，也就是说，不论是在 BANK0、BANK1 都能对特殊功能寄存器进行读写操作。



存储区指针

累加器 — ACC

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

程序计数器低字节寄存器 — PCL

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

表格寄存器 — TBLP，TBHP，TBLH

这三个特殊功能寄存器对储存在程序存储器中的表格进行操作。TBLP 和 TBHP 为表格低字节和高字节指针，指向表格数据全部的地址。它们的值必须在任何表格读取指令执行前加以设定，由于它们的值可以被如“INC”或“DEC”的指令所改变，这就提供了一种简单的方法对表格数据进行读取。表格读取数据指令执行之后，表格数据高字节存储在 TBLH 中。其中要注意的是，表格数据低字节会被传送到使用者指定的地址。

状态寄存器 — STATUS

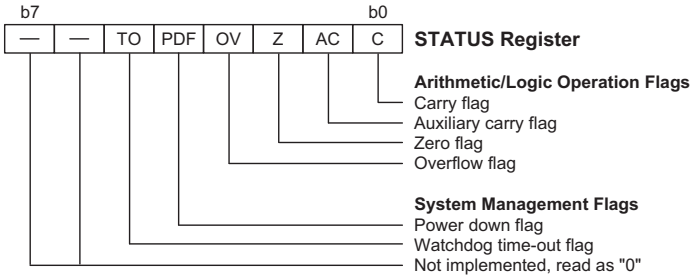
8 位的 STATUS 寄存器包含零标志位（Z）、进位标志位（C）、辅助进位标志位（AC）、溢出标志位（OV）、暂停标志位（PDF）和看门狗溢出标志位（TO），它同时记录算术/逻辑运算的和系统工作状态数据。

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

Z、OV、AC 和 C 标志位通常反映最近运算的状态：

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

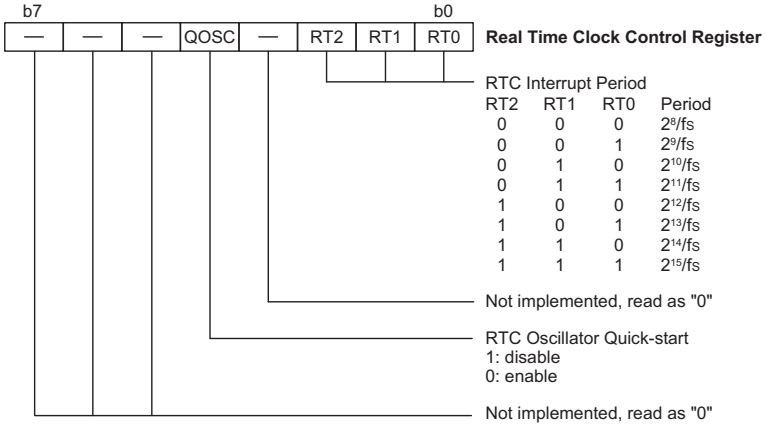
另外当进入一个中断程序或执行子程序调用时，状态寄存器不会自动压入到堆栈保存。假如状态寄存器的内容是重要的且子程序可能改变状态寄存器的话，则需谨慎的去做正确的储存。



状态寄存器

实时时钟控制寄存器 — RTCC

RTCC 寄存器控制两个内部功能，其中之一是 RTC 中断，用来提供一个有规律的内部中断。RTC 中断的驱动时钟来自于内部时钟源，即 f_S，其被进一步的分频以提供更久的时值，依次产生中断信号。分频率的值由 RTCC 寄存器的第 2～0 位，即 RT2～RT0 编程的值决定。通过直接写 RTCC 寄存器的这些位，将产生 $fs/2^8$～$fs/2^{15}$ 的溢出值。RTCC 寄存器也控制着 RTC 振荡器的快速启动功能。这个有 32768Hz 的固定频率的振荡器，能通过设置第 4 位，即 QOSC 位为 0 来以更快的速度启动。当单片机上电后，这一位将被清零，但是，由于一些额外能源的消耗，QOSC 位必须在大约 2 秒后被置位以降低功耗。



RTCC 寄存器

中断控制寄存器 — INTC0，INTC1

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

定时/计数器寄存器

该系列的单片机提供了一个 8 位和二个 16 位定时/计数器。每个定时/计数器都有相关的一个或一对寄存器，用来存放 8 位或 16 位的值。定时/计数器 2 有 TMR2 寄存器，其宽度为 8 位，定时/计数器 0/1 有一对 TMR0L/TMR0H 和 TMR1L/TMR1H 的寄存器，其宽度为 16 位。TMR0C，TMR1C 和 TMR2C 为三个控制寄存器，包含设置定时/计数器的选项，并确定定时/计数器的工作模式，同时包含着定时器打开/关闭的控制功能。

输入/输出控制寄存器

在特殊功能寄存器中，输入/输出寄存器和它们相对应的控制寄存器很重要。所有的输入/输出端口都有相对应的寄存器，标示为 PA，PB，PC，PD，PF 和 PG。如数据存储器结构图中所示，这些输入/输出寄存器映射到数据存储器的特定地址，用以传送端口上的输入/输出数据。每个输入/输出端口有一个相对应的控制寄存器，分别为 PAC，PBC，PCC，PDC，PFC 和 PGC，也同样映射到数据存储器的特定地址。这些控制寄存器设定引脚的状态，以决定哪些是输入口，哪些是输出口。要设定一个引脚为输入，控制寄存器对应的位必须设定成逻辑高，若引脚设定为输出，则控制寄存器对应的位必须设为逻辑低。程序初始化期间，在从输入/输出端口中读取或写入数据之前，必须先设定控制寄存器的位以确定引脚为输入或输出。使用“SET [m].i”和“CLR [m].i”指令可以直接设定这些寄存器的某一位。这种在程序中可以通过改变输入/输出端口控制寄存器中某一位而直接改变该端口输入/输出状态的能力是此系列单片机非常有用的特性。

脉宽调制寄存器 — PWM0，PWM1，PWM2，PWM3

每个 PWM 都具有自己独立的控制寄存器。这些 8 位的寄存器定义相应的脉冲宽度调制器调制周期的占空比。

A/D 转换寄存器 — ADRL，ADRH，ADCR，ACSR

单片机提供了 8 通道 12 位 A/D 转换器。A/D 转换需要涉及到 2 个数据存储器、1 个控制寄存器和 1 个时钟控制寄存器。A/D 转换结果寄存器为高位 ADRH 寄存器和低位 ADRL 寄存器。当一个模数转换周期结束后，转换出的数字量将保存在这些寄存器中。通道的选择和 A/D 转换器的设置通过寄存器 ADCR 控制，A/D 时钟频率由时钟源寄存器 ACSR 定义。

输入/输出端口

盛群单片机的输入/输出端口控制具有很大的灵活性。这体现在每一个引脚在使用者的程序控制下可以被指定为输入或输出、所有引脚的上拉电阻选项、以及指定引脚的唤醒选择，这些特性也使得此类单片机在广泛应用上都能符合开发的要求。

单片机提供从 48 个双向输入/输出口，标示为 PA、PB、PC，PD，PF 和 PG，这些输入/输出端口在数据存储器的对应指定地址如表所示。所有输入/输出口都可作为输入及输出之用。作为输入操作时，输入/输出引脚无锁存功能，输入数据必须在指令“MOV A,[m]”T2 上升沿准备好，m 表示端口地址。对于输出操作，所有数据是锁存的，而且持续到输出锁存被重写。

上拉电阻

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

PA 口唤醒

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

输入/输出端口控制寄存器

每一个输入/输出端口都具有各自的控制寄存器（PAC、PBC、PCC、PDC、PFC 和 PGC）控制输入/输出状态。利用此控制寄存器，每一个 CMOS 输出或者带或不带上拉电阻的输入，均可利用软件控制方式加以动态的重新设置。所有输入/输出端口的引脚都各自对应于输入/输出端口控制寄存器的某一位。若输入/输出引脚要实现输入功能，则对应的控制寄存器位必须设定为“1”，这时程序指令可以直接读出输入引脚的逻辑状态。如果引脚的控制寄存器位被设定为“0”，则此引脚被设置为 CMOS 输出。当引脚被设置为输出状态，程序指令读取的是输出端口寄存器的内容。请注意当输入/输出端口被设置为输出状态时，此时如果对输出口做读取的动作，则会读取到内部数据寄存器中的锁存值，而不是输出引脚实际的逻辑状态。

引脚共用功能

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

- 外部中断输入
  外部中断引脚 $\overline{\text{INT}}$ 与输入/输出引脚 PA5 共用。必须使能 INTC0 寄存器中的外部中断使能位才能将此引脚用作外部中断引脚。端口控制寄存器相应的位－PAC.5，也必须设置此引脚正确的外部中断操作的输入。如果引脚被用作外部中断，对此引脚做的任何上拉电阻掩膜选项的选择将保持有效。如果 PAC 控制寄存器已经设置此引脚为输出，那么即使 INTC0 寄存器中的外部中断使能位被使能，此引脚仍将会作为普通的逻辑输出运行。

- PFD 输出

  此单片机包含 PFD 功能，PFD 信号输出引脚与 PA3 共用。PFD 输出功能可以通过掩膜选项进行选择，并且在程序烧录后保持不变。需要注意的是，必须将 PAC.3 设为输出以使能 PFD 输出。若 PAC 设置为带上拉电阻的输入，即使选择了 PFD 输出功能，此引脚仍将作为带上拉电阻的一般输入引脚使用。

- PWM 输出

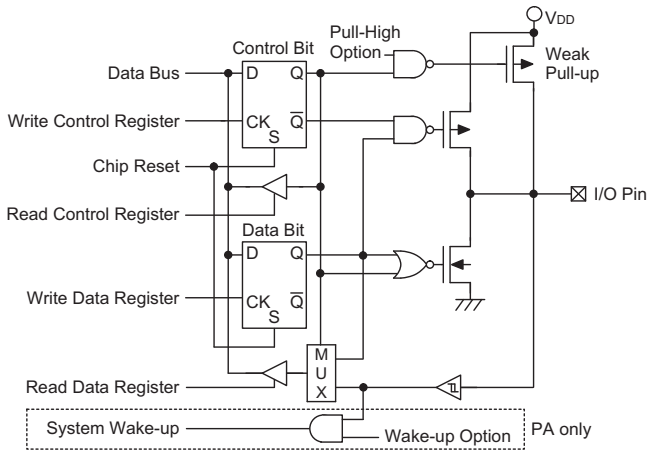
  此系列单片机均提供 PWM0，PWM1，PWM2 和 PWM3 输出，分别与 PD0，PD1，PD2 和 PD3 共用引脚。PWM 输出功能可以通过掩膜选项进行选择，并且在程序烧录后保持不变。需要注意的是，必须将 PDC 中相应的位设为输出以使能 PWM 输出。若 PDC 设置为带上拉电阻的输入，即使选择了 PWM 输出功能，此引脚仍将作为带上拉电阻的一般输入引脚使用。

- A/D 输入

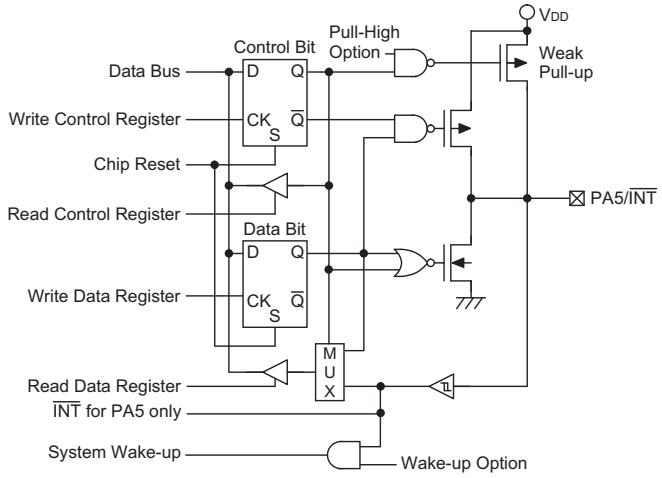
  此系列的单片机都具有 8 个 A/D 转换器输入。所有的模拟输入与 PB 口的 I/O 引脚共用。如果这些引脚被用来作为 A/D 输入而不是一般的 I/O 引脚，则 A/D 转换控制寄存器 ADCR 中相应的位必须被正确的设定。掩膜选项内不包含 A/D 功能。如果这些引脚作为 I/O 引脚使用，仍可以通过掩膜选项选择是否要接上拉电阻。然而如果作为 A/D 输入使用，则这些引脚上的上拉电阻会自动失效。

输入/输出端口结构

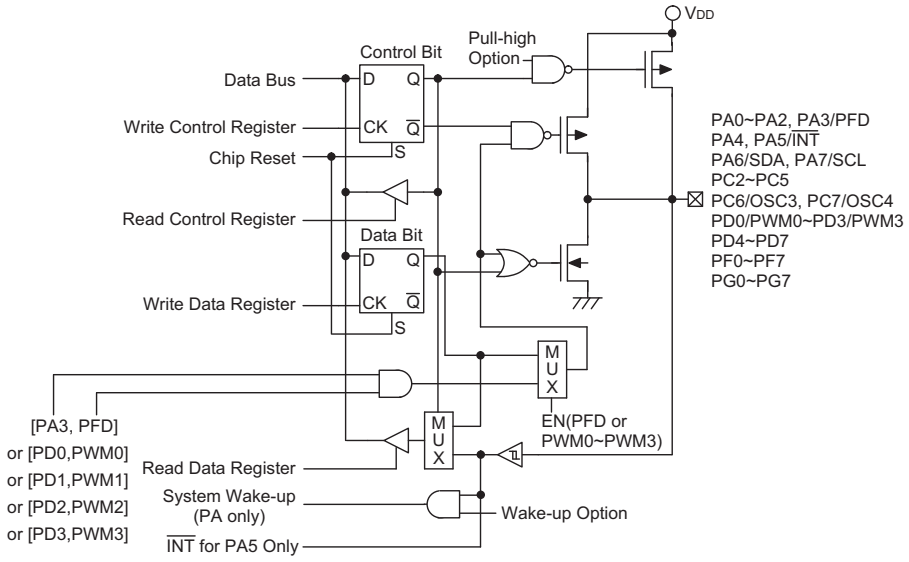
下列为输入/输出端口的内部结构图，可能与芯片内部实际的结构并不完全相同，只是用于帮助用户理解输入/输出端口。



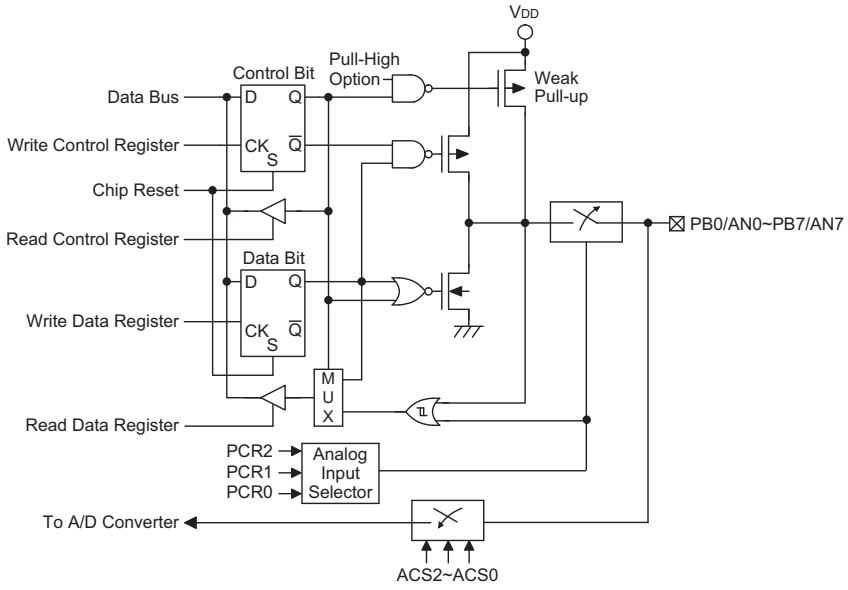
没有引脚共用功能的输入/输出端口



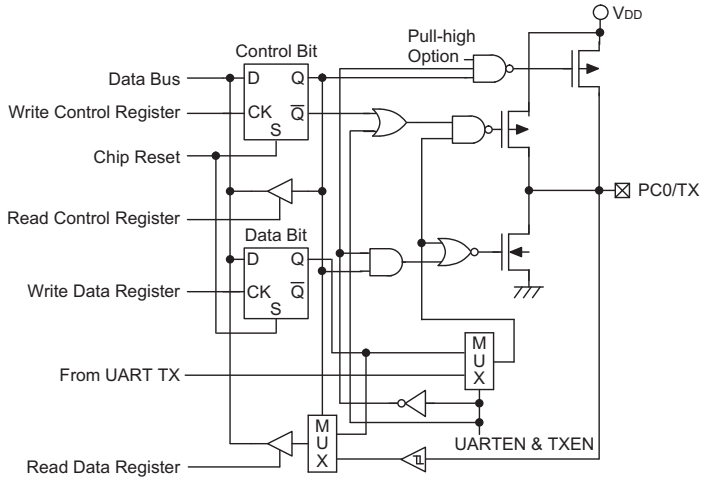
PA5 输入/输出端口



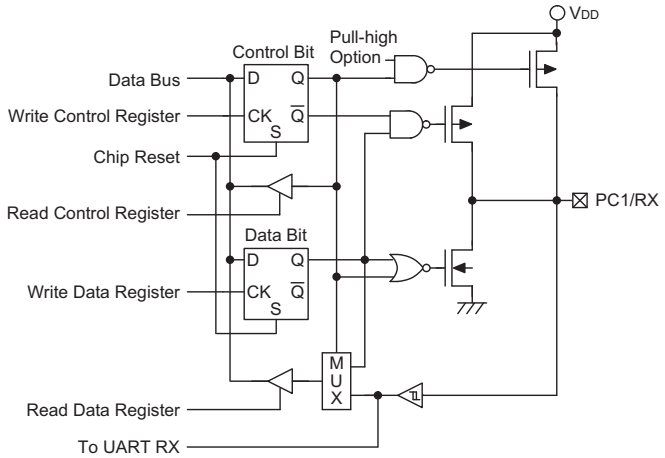
输入/输出端口



PB 输入/输出端口



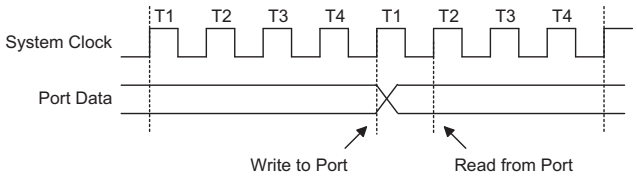
PC0/TX 输入/输出端口



PC1/RX 输入/输出端口

编程注意事项

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



读写时序

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

注意，有些型号芯片具有不同封装类型，导致某些端口没有引出。若这此引脚被设为输入，就会产生振荡并增加系统功耗，特别在 HALT 模式下更明显。建议在使用中将未引出的端口设为输出或设为带上拉的输入。

定时/计数器

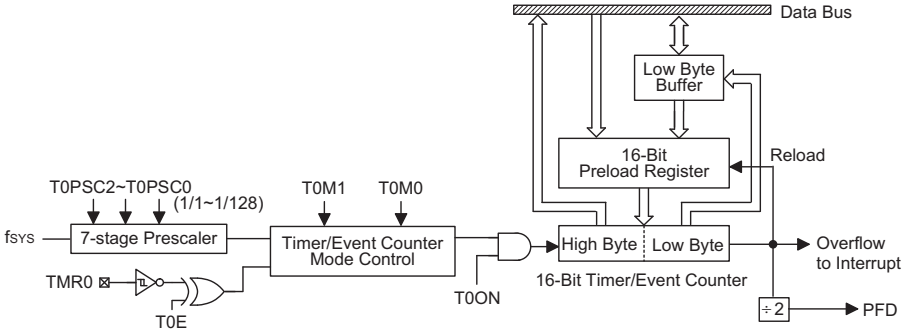
定时/计数器在任何单片机中都是一个很重要的部分，提供程序设计者一种实现和时间有关功能的方法。HT46RU26/HT46CU26 提供 2 个 16 位和 1 个 8 位的向上定时/计数器。定时/计数器有三种不同的工作模式，可以被当作普通定时器、外部事件计数器、或者脉冲宽度测量器使用。定时器里的 8 级预分频器扩大了定时的范围。

每个定时/计数器都有相关的一个或一对寄存器，用来存放 8 位或 16 位的值。定时/计数器 2 有 TMR2 寄存器，其宽度为 8 位，定时/计数器 0/1 有一对 TMR0L/TMR0H 和 TMR1L/TMR1H 的寄存器，其宽度为 16 位。TMR0C，TMR1C 和 TMR2C 为三个控制寄存器，包含设置定时/计数器的选项，并确定定时/计数器的工作模式，同时包含着定时器打开/关闭的控制功能。

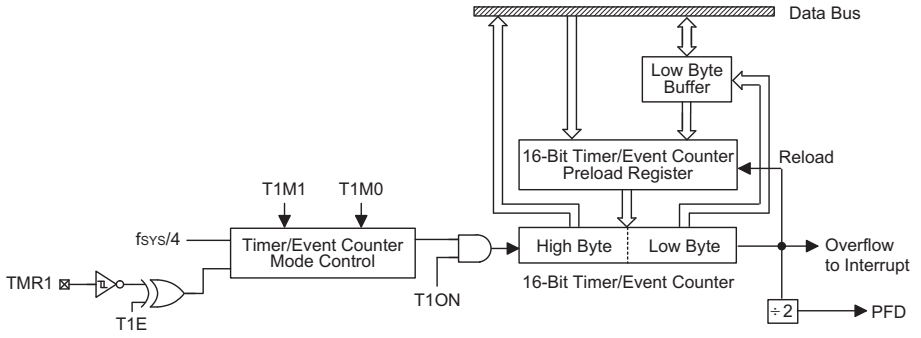
配置定时/计数器输入时钟源

无论系统时钟还是来子外部时钟源，都可以产生定时/计数器时钟源。当定时/计数器在定时器模式或者在脉冲宽度测量模式时，使用内部时钟作为计时来源。

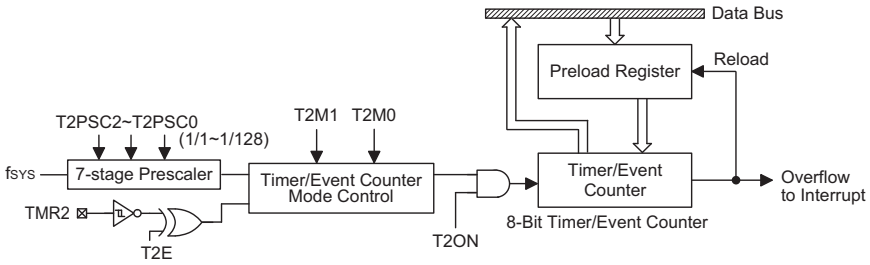
定时/计数器在事件计数器模式时使用外部时钟，由外部定时/计数器引脚 TMR0， TMR1 或 TMR2 提供。每次外部引脚由高电平到低电平或者由低电平到高电平（由 T0E，T1E 或 T2E 位决定）跳变时，计数器值加一。



16 位定时/计数器 0 结构



16 位定时/计数器 1 结构



8 位定时/计数器 2 结构

定时/计数寄存器 — TMR0L/ TMR0H，TMR1L/ TMR1H，TMR2

定时/计数器寄存器是位于专用数据存储器内的特殊功能寄存器，储存实际的定时器值。对于定时/计数器0和1，即16 位定时/计数器，需要用两个8位寄存器来储存16 位定时/ 计数器的值，这些成对的寄存器即为TMR0L/TMR0H 和TMR1L/ TMR1H。对于定时/计数器2，即8位定时/计数器，这个寄存器是TMR2。

当用作内部定时器模式时收到一个内部计时脉冲时，或用作外部计数模式时外部定时/计数器引脚发生电平转换时，定时/计数寄存器的值将会加一。定时器从预置寄存器所载入的值开始向上计数，直到8 位定时/计数器计到FFH，或16 位定时/计数器计到FFFFH，此时定时器发生溢出且会产生一个内部中断信号。定时器的值随后被预置寄存器的值重设，定时/计数器继续计数。

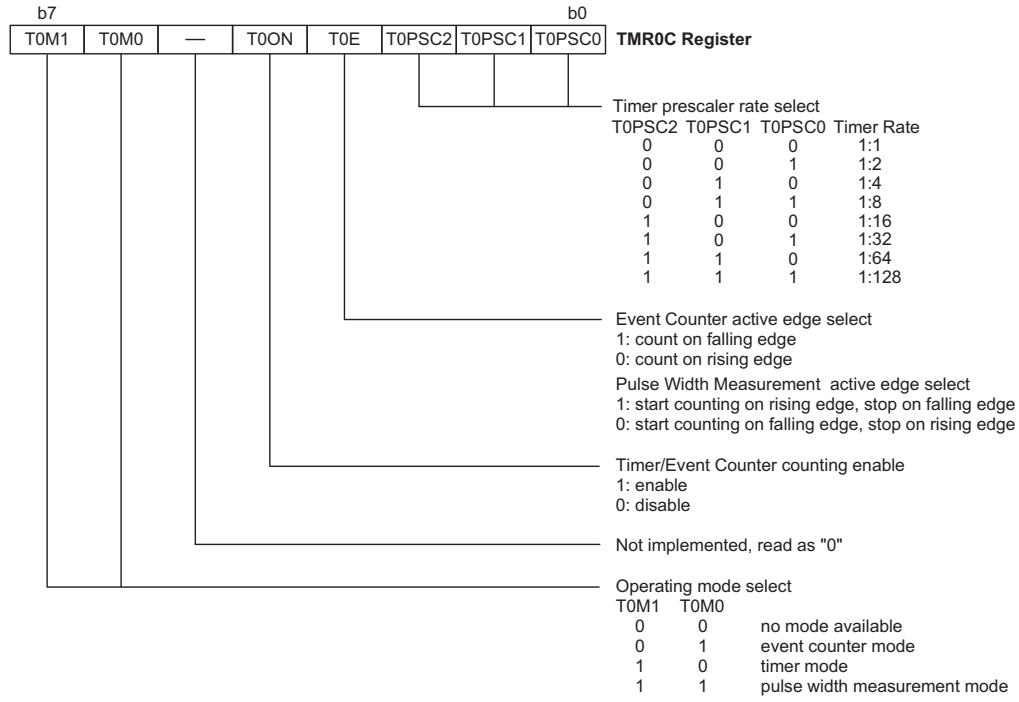
当预置寄存器被清为零，就可以得到00H到FFH 或 FFFFH 的最大计算范围。此时要注意的是，上电后预置寄存器中的数值处于未知状态。定时/计数器在OFF 条件下，如果把数据写入预置寄存器，这数据将被立即写入实际的定时器。然而如果定时/计数器已经被打开且正在计数，在这个周期内写入到预置寄存器的任何新数据将被保留在预置寄存器中，等到下一个溢出发生时才会被写入实际的定时器。

对于16 位内部定时/计数器，它有低字节与高字节两个定时/计数寄存器，访问这些寄存器需要以指定方式进行。必须要注意的是当使用指令载入数据到低字节寄存器，即TMR0L 或TMR1L 时，数据只被载入到低字节缓冲器而不是直接送到低字节寄存器。当数据写入相应高字节寄存器，即TMR0H 或TMR1H 时，低字节缓冲器中的数据才真正被写入低字节寄存器。换句话说，写入数据到高字节定时/计数寄存器时，数据会被直接写入到高字节寄存器。同时在低字节缓冲器里的数据将被写入相应低字节寄存器。所以当载入数据到 16 位定时/计数寄存器时，低字节数据应该先写入。另外要注意的是读取低字节寄存器的内容时，必须先读取高字节寄存器的内容，相应低字节寄存器中的内容就会载入低字节缓冲器中并被锁存。在此动作执行之后，低字节寄存器中的内容可使用一般的方式读取。请注意，读取定时/计数器低字节寄存器实际是读取先前锁存在低字节缓冲器中的内容，而非定时/计数器低字节寄存器的实际内容。

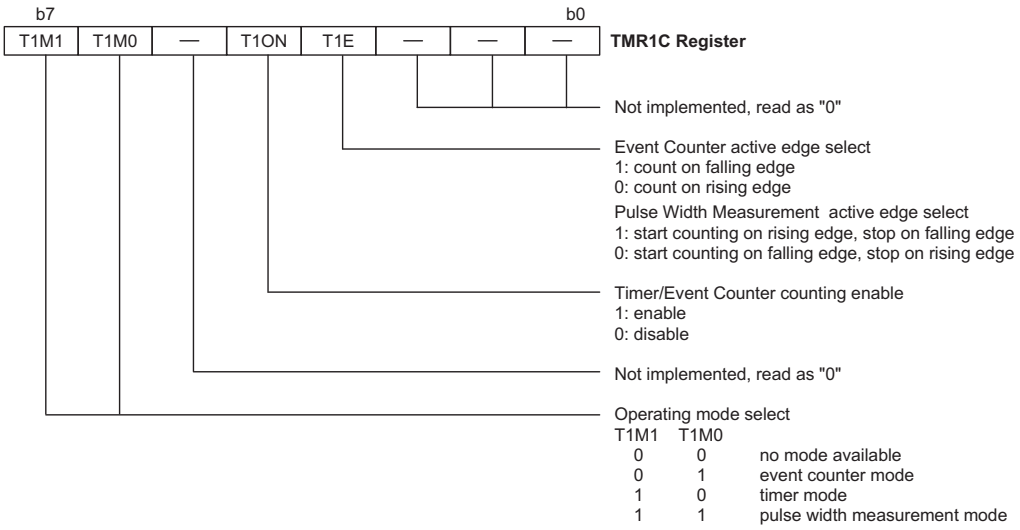
定时/计数控制寄存器 — TMR0C，TMR1C，TMR2C

HOLTEK MCU 灵活的定时/计数器特性，使其能工作在三种不同的模式，至于选择工作在哪一种模式则是由各自控制寄存器的内容决定，即 TMR0C，TMR1C 和 TMR2C。定时/计数控制寄存器和对应数据寄存器控制定时/计数器的全部操作。在使用定时器之前，必须正确地设定定时/计数控制寄存器，以便保证定时器能正确操作，而这个过程通常在程序初始化期间完成。

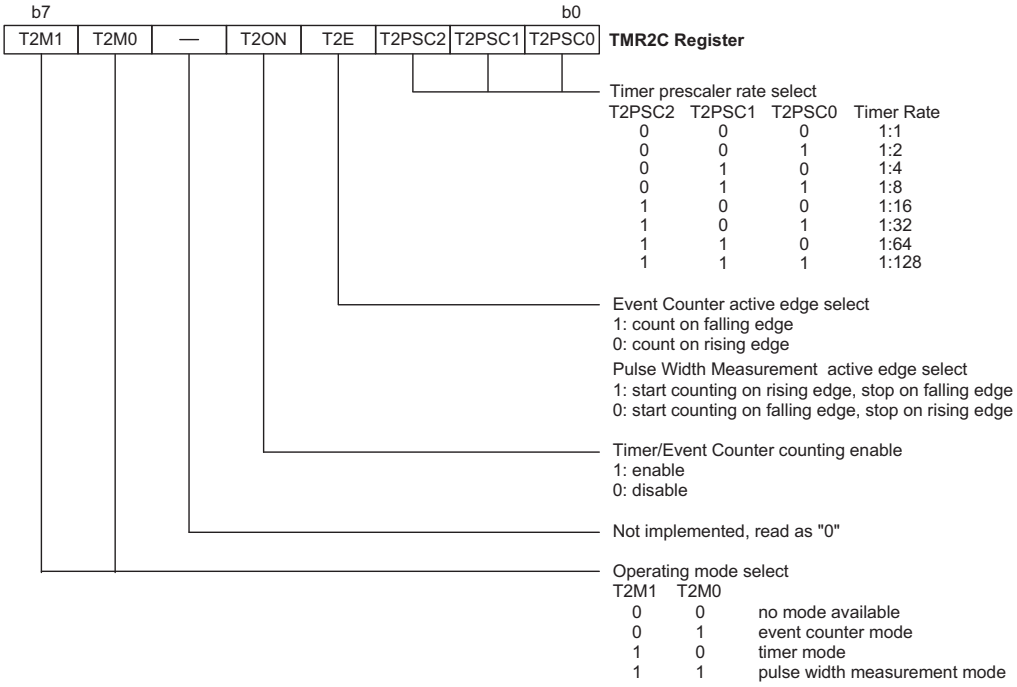
为了确定定时器工作在哪一种模式，定时器模式，外部事件计数模式或脉冲宽度测量模式，定时/计数控制寄存器位 7 和位 6，即 T0M0/T0M1，T1M0/T1M1 和 T2M0/T1M1 位必须设定到要求的逻辑电平。定时器打开位 T0ON，T1ON 或 T2ON，即定时/计数控制寄存器的第 4 位，是定时器控制的开关，设定为逻辑高时，计数器开始计数，而清零时则停止计数。TMR0C 和 TMR2C 的第 0~2 位决定输入定时预分频器中的分频系数。如果使用外部计时源，预分频器的位将不起作用。如果定时器工作在事件计数或脉冲宽度测量模式，T0E，T1E 或 T2E 的逻辑电平，即定时/计数控制寄存器的第 3 位将可用来选择上升沿或下降沿触发。



定时/计数控制寄存器 0



定时/计数控制寄存器 1



定时/计数控制寄存器 2

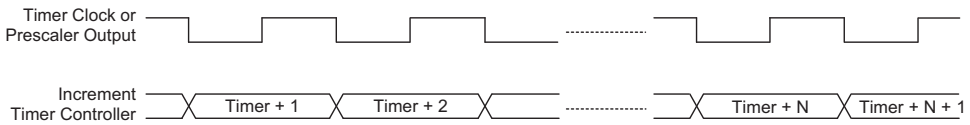
配置定时器模式

在这个模式下，定时器可以用来测量固定时间间隔，当定时器发生溢出时，就会产生一个内部中断信号。要工作在这个模式，两个工作模式选择位，T0M1/T0M0，T1M1/T1M0 或 T2M1/T2M0，在 TMRC 寄存器中必须设置如下：

位 7	位 6
1	0

定时器模式设置

在这个模式下，定时器的计数源来自内部时钟 f_{SYS} 或者 $f_{SYS}/4$。当设置完定时/计数控制寄存器其余位信息，将定时器使能位 T0ON，T1ON 或 T2ON，即定时/计数控制寄存器位 4 必须被设为逻辑高，才能使定时器工作。每次内部时钟由高到低的电平跳变都会使定时器值增加一。当定时器已满即溢出时，会产生中断信号且定时器会重新载入已经载入到预置寄存器的值，然后继续向上计数。若定时器中断允许，将会产生一个中断信号。寄存器 INTC 中 ETI 位清零，则定时器中断禁止。



定时器模式时序图

配置事件计数模式

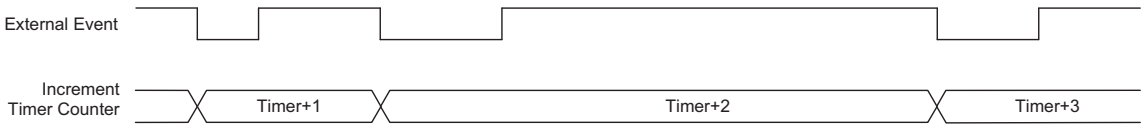
在这个模式下，发生在外部事件计数引脚改变的次数，可以通过内部计数器来记录。两个工作模式选择位，T0M1/T0M0，T1M1/T1M0 或 T2M1/T2M0，在 TMRC 寄存器中必须设置如下：

位 7	位 6
0	1

外部事件计数模式设置

系统工作在计数器模式下，时钟源来自外部事件计数引脚，TMR0，TMR1 或 TMR2，但预分频器不对外部时钟源分频。当设置完定时/计数控制寄存器其余位信息，将计数器使能位 T0ON，T1ON 或 T2ON，即定时/计数控制寄存器位 4 设为逻辑高，计数器开始计数。当边沿触发位 T0E，T1E 或 T2E，即定时/计数控制寄存器位 3 设为逻辑低，每次外部定时/计数器引脚接收到由低到高的电平跳变将使计数器值加一；而当其设为逻辑高时，每次外部定时/计数器引脚接收到由高到低的电平跳变将使计数器值加一。当定时器已满即溢出时，会产生中断信号且定时器会重新载入已经载入到预置寄存器的值，然后继续向上计数。定时器中断可以通过清除 INTC 寄存器中 ETI 位禁止。

注意的是，在事件计数模式下，即使系统进入 HALT 状态，定时/计数器仍可记录外部引脚的变化。定时/计数器溢出将会唤醒系统，若中断允许将会产生一个定时/计数器中断。



计数器模式时序图

配置脉冲宽度测量模式

在这个模式下，可以测量外部事件引脚上的外部脉冲宽度。要工作在这个模式，两个工作模式选择位，T0M1/T0M0，T1M1/T1M0 或 T2M1/T2M0，在 TMRC 寄存器中必须设置如下：

位 7	位 6
1	1

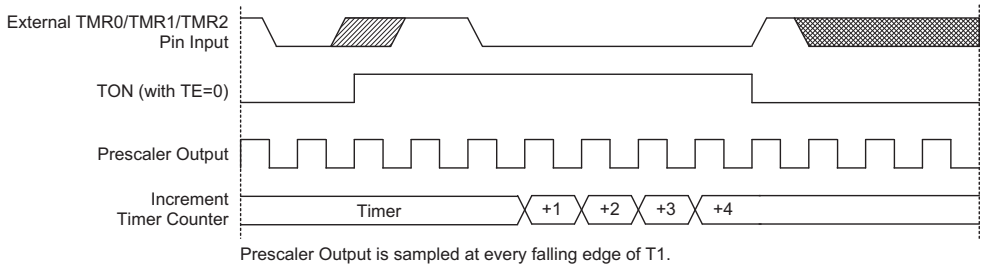
脉冲宽度测量模式设置

在这个模式下，定时器的计数源来自内部时钟 f_{SYS} 或者 $f_{SYS}/4$。当设置完定时/计数控制寄存器其余位信息，将定时器使能位 T0ON，T1ON 或 T2ON，即定时/计数控制寄存器位 4 必须被设为逻辑高，才能使定时器工作。但是，计数器实际上不会开始计数，直到外部事件引脚受到有效触发沿时。

如果有效边沿触发位 T0E，T1E 或 T2E，即定时/计数控制寄存器位 3 设置为逻辑低，当外部事件引脚接收到一个由高到低的电平跳变时，定时/计数器将开始计数直到外部事件引脚回到它原来的高电平，此时计数器使能位将自动清除为零，且停止计数。而如果有效边沿触发位设置为逻辑高，则当外部事件引脚接收到一个由低到高的电平跳变时，计数器开始计数直到外部事件引脚回到原来的低电平，如上所述，计数器使能位自动清 0，且停止计数。注意的是，在脉冲宽度测量模式中，当外部事件引脚上的外部控制信号回到它原来的电平时，计数器使能位将自动清 0。而在其它两种模式下，定时/计数器使能位只能在程序控制下被清 0。

这时定时/计数器中的值可被程序读取，并由此得知外部定时/计数器引脚接收到的脉冲的长度。当计数器使能位复位时，任何在外部事件计数引脚的电平跳变将被忽略，直到使能位再次被程序设定为逻辑高，计数器才开始脉冲宽度测量。换句话说，用这种方法可轻松地完成单个脉冲的测量。

注意的是，在这种模式下，计数器是通过外部事件计数引脚上的逻辑跳变来控制，而不是逻辑电平。当定时器已满即溢出时，会产生中断信号且定时器会重新载入已经载入到预置寄存器的值，然后继续向上计数。定时器中断可以通过清除 INTC 寄存器中 ETI 位禁止。



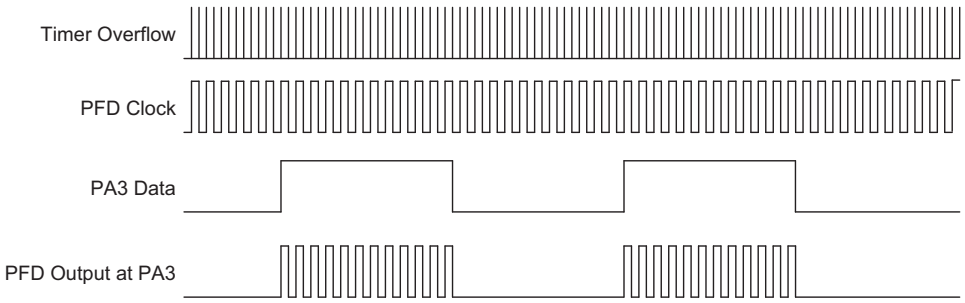
脉冲宽度测量模式时序图

可编程分频器 – PFD

PFD 输出引脚与 PA3 引脚共用。此功能通过掩膜选项选择，如果不选择该功能，则这个引脚作为普通的输入/输出引脚使用。PFD 电路使用定时器溢出信号作为它的时钟源。载入合适的值到定时器预分频器，可以产生需要时钟源的分频系数，由此来控制输出的频率。系统时钟被预分频器分频后的时钟源，进入定时器计时，定时器从预置寄存器的值开始往上计数，直到计数值满而产生溢出信号，导致 PFD 输出改变状态。定时器将自动地重新载入预置寄存器的值，并继续向上计数。

使 PFD 正确运作，必须将 PA 控制寄存器 PAC 位 3 设置为输出。如果把它设置为输入，则 PFD 输出不工作，该引脚仍是作为普通的输入引脚使用。只有将 PA3 置 1，PFD 输出引脚才会有输出。这个输出数据位被用作 PFD 输出的开/关控制。注意，如果 PA3 输出数据位被清 0，PFD 输出将为低电平。

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



PFD 输出控制

预分频器

定时/计数器 0 和定时/计数器 2 都拥有预分频器，其将输入时钟源分频以提供定时/计数器更高的范围。TMR0C 和 TMR2C 寄存器中 PSC0~PSC2 可以用来定义定时/计数器中内部时钟源的预分频系数。注意的是，系统工作在外部事件计数模式，预分频器定并无影响。

输入/输出接口

当工作在事件计数或脉冲宽度测量模式时，需要使用外部引脚以确保正确的动作。由于此端口为共用引脚，必须将其设置为定时/计数器的外部输入而不是普通的输入/输出，这需要设置定时/计数控制寄存器工作模式位为外部事件计数或脉宽测量模式。另外 PAC 控制寄存器的位必须为高，以保证被设为输入引脚。即使定时/计数器使用了此引脚，掩膜选项的上拉仍有效。

编程注意事项

当定时/计数器工作在定时器模式时，定时器的时钟源使用内部系统时钟且与单片机所有运算都能同步。在这个模式下，当定时器寄存器溢出时，单片机将产生一个内部中断信号，使程序进入相应的内部中断向量。对于脉冲宽度测量模式，定时器的时钟源也是使用内部系统时钟，但定时器只有在正确的逻辑条件出现在外部事件计数引脚时开始工作。由于外部事件和内部定时器时钟无法同步，只有当下一个定时器时钟到达时，单片机才会发现这个外部事件，因此在测量值上可能有小的差异，需要程序设计者在程序应用时加以注意。同样的情况发生在定时/计数器配置为外部事件计数模式时，它的时钟源是外部事件，与内部系统时钟或者定时器时钟不同步。

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

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

定时/计数器应用范例

这个例子说明了如何设置定时/计数器 0 的寄存器，如何设置和控制中断。另外还需注意的是，怎样通过 TMR0C 寄存器的第 4 位来启停定时/计数器。此应用范例设置定时/计数器 0 为定时模式，时钟来源于内部的系统时钟。

```
org 04h                    ; external interrupt vectors
reti
org 08h                    ; timer/event counter 0 interrupt vector
jmp tmrint0                ;jmp here when timer/event counter 0 overflows
    :
org 20h                    ; main program
                           ; internal timer/event counter 0 interrupt routine
tmrint0:
                           ;timer/event counter 0 main program placed here
    :
reti
    :
    :
begin:
                           ; setup timer registers
mov a,09bh                 ; setup timer preload value
mov tmr0,a
mov a,081h                 ; setup timer control register
mov tmr0c,a                ; timer mode and prescaler set to /4
                           ; setup interrupt register
mov a,005h                 ; enable master interrupt and timer interrupts
mov intc0,a
set tmr0c.4                ;start timer/event counter0– note mode bits must be previously setup
```

脉冲宽度调制器

此系列每款单片机都提供 4 个脉冲宽度调制(PWM)输出。这在马达速度控制等应用中十分有用，通过给相应的 PWM 寄存器设置特定的值，PWM 功能可提供占空比可调而频率固定的波形。

在数据存储器中，单片机为每一个 PWM 都指定了对应的寄存器。对于只有一个 PWM 输出的单片机，这个寄存器为 PWM。对于有两个 PWM 输出的单片机，寄存器则为 PWM0 和 PWM1。此寄存器为 8 位，表示输出波形中每个调制周期的占空比。为了提高 PWM 调制频率，每一个调制周期被调制两个或四个独立的调制子区段，即分别是 7＋1 或 6+2 模式。每个单片机可以通过设置适当的掩膜选项来选择使用哪种模式。当选择一种掩膜选项模式后，此模式将应用于此芯片的所有 PWM 输出。注意的是，当使用 PWM 时，只要将所需的值写入相应的 PWM 寄存器内，单片机的内部硬件会自动地将波形细分为子调制周期。

对所有的单片机而言，PWM 时钟源就是系统时钟 f_{SYS}。

将原始调制周期分成 2 个或 4 个子周期的方法，使产生更高的 PWM 频率成为可能，这样可以提供更广泛的应用。只要产生的 PWM 脉冲周期小于负载的时间常数，PWM 输出就比较合适，这是因为长时间常数负载将会平均 PWM 输出的脉冲。PWM 频率与 PWM 调制频率的不同之处如下表所示

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

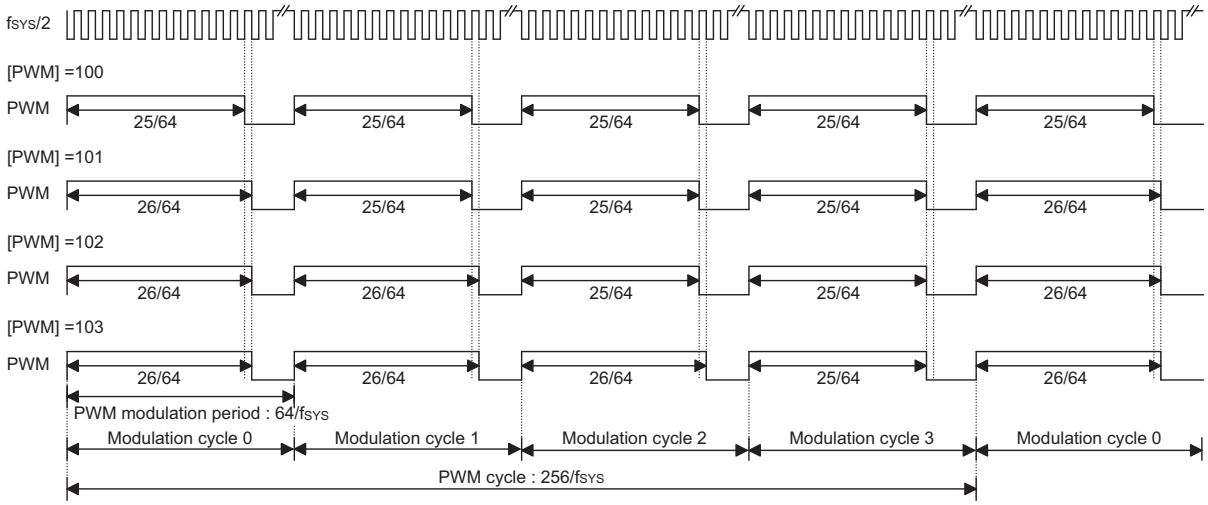
6+2 PWM 模式

通过一个 8 位的 PWM 寄存器控制，每个完整的 PWM 周期由 256 个时钟周期组成。在 6+2 PWM 模式中，每个 PWM 周期又被分成四个独立的子周期，称为调制周期 0~调制周期 3，在表格中以“i”表示。四个子周期各包含 64 个时钟周期。在这个模式下，得到以 4 为因数增加的调制频率。8 位的 PWM，PWM0，PWM1 寄存器被分成两个部分，这个寄存器的值表明整个 PWM 波形的占空比。第一部分包括第 2 位~第 7 位，表示 DC 值，第二部分为第 0 位~第 1 位，表示 AC 值。在 6+2 PWM 模式中，四个调制子周期的占空比，分别如下表所示。

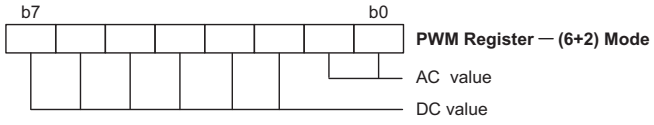
参数	AC (0~3)	DC (占空比)
调制周期 i (i=0~3)	i<AC	$\frac{DC+1}{64}$
	i≥AC	$\frac{DC}{64}$

6+2 模式调制周期值

下图表示 6+2 模式下 PWM 输出的波形。请特别注意单个的 PWM 周期是如何分成四个独立的调制周期以及 AC 值与 PWM 值的关系。



6+2 PWM 模式



6+2 模式 PWM 寄存器

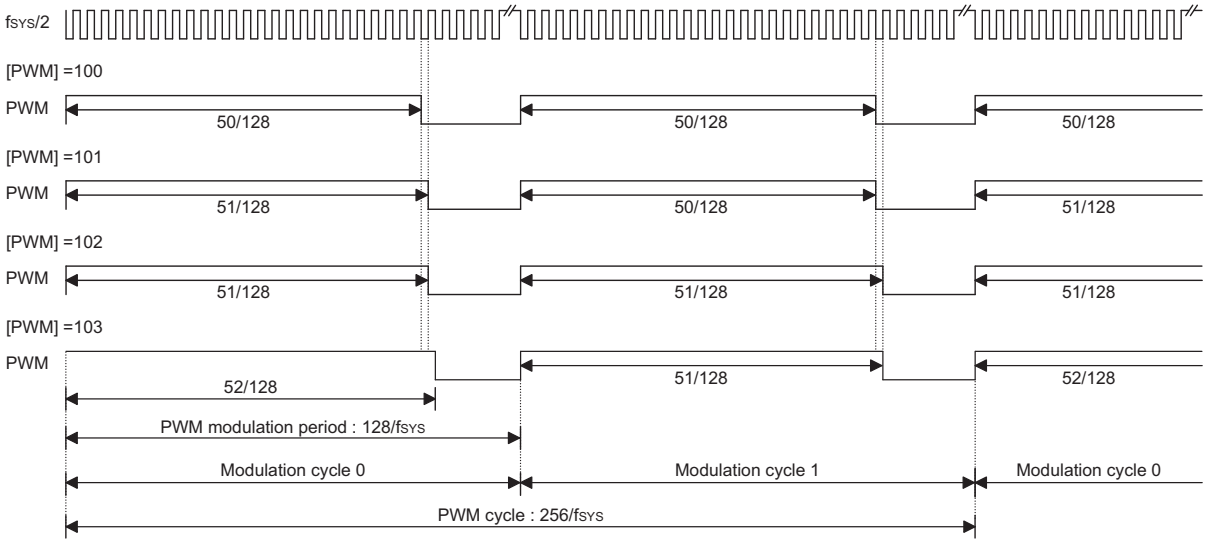
7+1 模式

通过一个 8 位的 PWM 寄存器控制，每个完整的 PWM 周期由 256 个时钟周期组成。在 7+1PWM 模式中，每个 PWM 周期又被分成两个独立的子周期，称为调制周期 0~调制周期 1，在表格中以“i”表示。两个子周期各包含 128 个时钟周期。在这个模式下，得到以 2 为因数增加的调制频率。8 位的 PWM，PWM0，PWM1 寄存器被分成两个部分，这个寄存器的值表明整个 PWM 波形的占空比。第一部分包括第 1 位~第 7 位，表示 DC 值，第二部分为第 0 位，表示 AC 值。在 7+1 PWM 模式中，两个调制子周期的占空比，分别如下表所示。

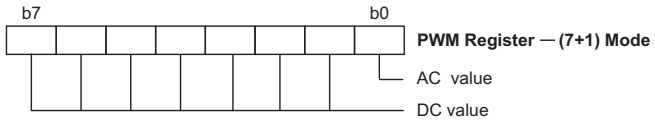
参数	AC (0~1)	DC (占空比)
调制周期 i (i=0~1)	i<AC	$\frac{DC+1}{128}$
	i≥AC	$\frac{DC}{128}$

7+1 模式调制周期值

下图表示 7+1 模式下 PWM 输出的波形。请特别注意单个的 PWM 周期是如何分成两个独立的调制周期以及 AC 值与 PWM 值的关系。



7+1 PWM 模式



7+1 模式 PWM 寄存器

PWM 输出控制

此系列所有单片机，PWM 输出与 PD0～PD3 端口引脚共用。要使某个引脚作为 PWM 输出而非普通的 I/O 引脚，必须选择正确的 PWM 掩膜选项。I/O 端口控制寄存器 PDC 中相应的位也必须写“0”，以确保所需要的PWM输出引脚设置为输出状态。在完成这两个初始化步骤，以及将所要求的PWM值写入PWM寄存器之后，将“1”写入到 PD 输出数据寄存器的相应位，则 PWM 数据就会出现在引脚上。将“0”写入到 PD 输出数据寄存器的相应位，则会除能 PWM 输出功能并强制输出低电平。通过这种方式，PD 数据输出寄存器作为 PWM 功能的开/关控制来使用。假如掩膜选项已经选择 PWM 功能，但是在 PDC 控制寄存器中相应的位又写入“1”，使其成为输入引脚，则此引脚仍是作为带上拉电阻的正常输入端使用。

PWM 应用范例

下面的范例程序表明如何设置和控制 PWM 输出。在使用相应 PWM 输出前需要在掩膜选项中使能 PWM 输出。

```
mov     a,64h           ; 设置 PWM 值 64H 为十进制的 100
mov     pwm0,a
clr     pdc.0           ; 设置 PD0 为输出
set     pd.0            ;pd.0=1;使能 PWM 输出
:       :
:       :
clr     pd.0            ; 禁能 PWM0 输出 – PD.0 将保持低电平
```

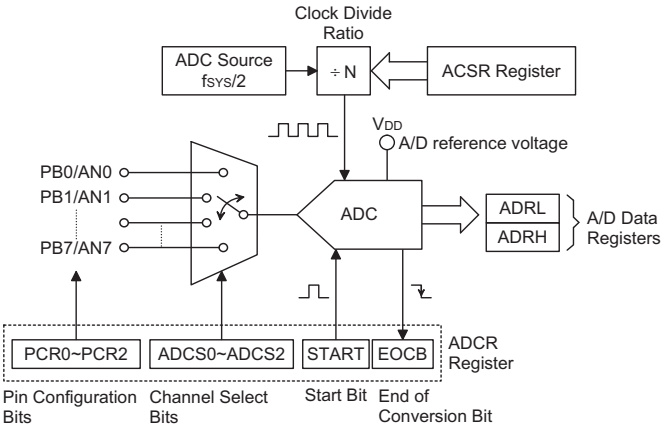
A/D 转换器

对于大多数电子系统而言，处理现实世界中模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先必须通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

A/D 简介

此系列每款单片机都包含了八个通道的 A/D 转换器，它们可以直接接入外部模拟信号（来自传感器或其它控制信号）并直接将这些信号转换成 12 位位的数字量。

下图显示了 A/D 转换器整个内部结构和相关的寄存器。



A/D 转换器结构

A/D 转换器数据寄存器 — ADRL，ADRH

在 A/D 转换完毕后，单片机可以直接读取这些寄存器以获得转换结果。对于具有 2 个 A/D 转换结果寄存器的单片机，要注意的是，只有高位寄存器 ADRH 完全利用了 8 位。而低位寄存器 ADRL 只利用了 8 位中的 4 位，它包含的只是 9 位转换值中低的 1 位。

在下表中，D0~D8 是 A/D 换转数据结果位。

寄存器	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRL	D3	D2	D1	D0	—	—	—	—
ADRH	D11	D10	D9	D8	D7	D6	D5	D4

A/D 数据寄存器

A/D 转换控制寄存器 – ADCR

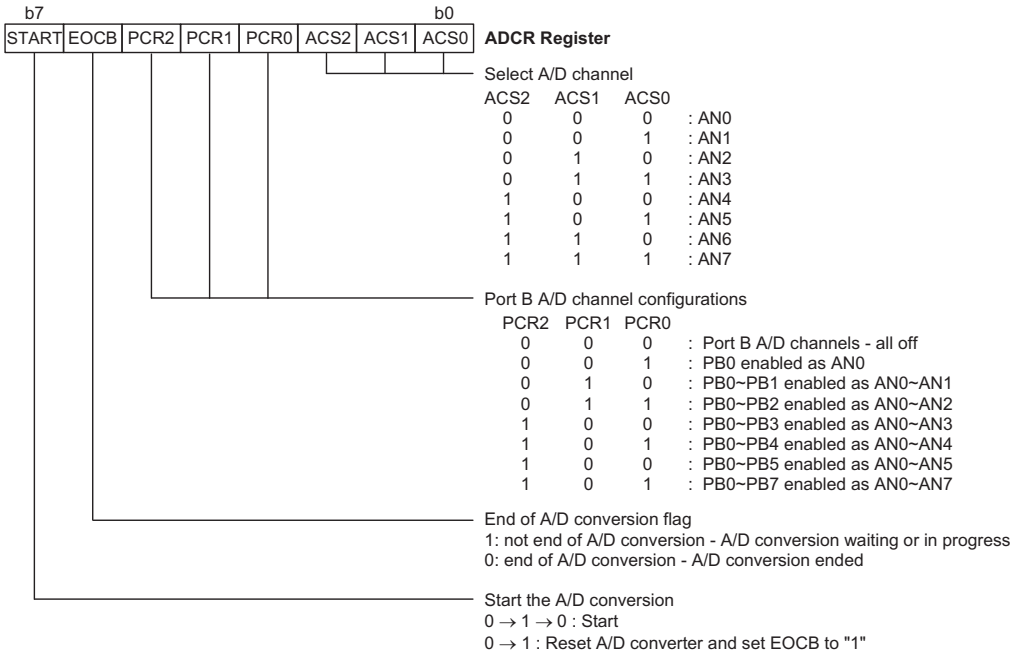
寄存器 ADCR 用来控制 A/D 转换器的功能和操作。这个 8 位的寄存器所定义的功能包括选择哪一个模拟通道连接至内部 A/D 转换器，哪个引脚是模拟输入，哪个引脚是正常 I/O，并控制和监视 A/D 转换器的开始和复位功能。

寄存器 ADCR 包含 ACS2~ACS0 位，它们定义通道的编号。由于每个单片机只包含一个实际的模数转换电路，因此这 4 个模拟输入中的每一个都必须分别被发送到转换器。ADCR 寄存器中 ACS2~ACS0 位的功能正是决定哪个模拟通道真正连接到内部 A/D 转换器。需要注意的是， ACS2 位必须保持为“0”。

ADCR 寄存器中的 PCR2~PCR0 位，用来定义 PB 端口上哪些引脚为 A/D 转换器的模拟输入，哪些引脚为正常的 I/O。注意的是，如果 PCR2~PCR0 全都设为“0”，则所有 PB 端口的引脚都被设定为正常的 I/O，这时内部 A/D 转换器电路的电源将被关闭以减少功耗。

ADCR 寄存器中的 START 位，用于打开和复位 A/D 转换器。当单片机设定此位从逻辑低到逻辑高，然后再到逻辑低，就会开始一个模数转换周期。当 START 位从逻辑低到逻辑高，但不再回到逻辑低时，ADCR 寄存器中的 EOCB 位置“1”，复位模数转换器。START 位用于控制内部模数换转器的开/关动作。

ADCR 寄存器中的 EOCB 位用于表明模数转换过程的完成。在转换周期结束后，EOCB 位会被单片机自动地置为“0”。此外，也会置位中断控制寄存器内相应的 A/D 中断请求标志位，如果中断使能，就会产生适当的内部中断信号。A/D 内部中断信号将引导程序到相应的 A/D 内部中断入口。如果 A/D 内部中断被禁止，单片机会轮询 ADCR 寄存器中的 EOCB 位，检查此位是否被清除，以作为另一种侦测 A/D 转换周期结束的方法。

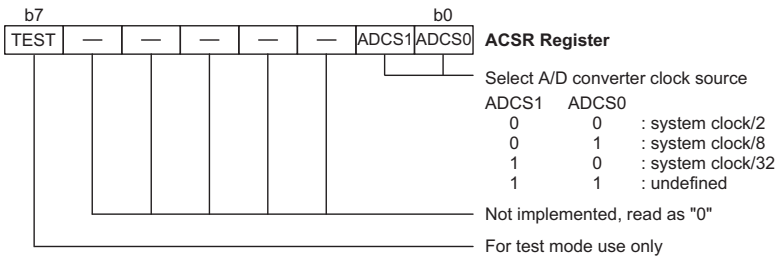


A/D 转换控制寄存器

A/D 转换时钟源寄存器 –ACSR

A/D 转换器时钟源为系统时钟 f_{SYS} 的分频，分频系数由 ACSR 寄存器中的 ADCS1 和 ADCS0 位决定。

虽然 A/D 时钟源是由系统时钟 f_{SYS}、ADCS1 和 ADCS0 决定，但可选择的最大 A/D 时钟源速度则有一些限制。



A/D 转换时钟控制寄存器

f_{SYS}	A/D 时钟周期(t_{AD})			
	ADCS1, ADCS0=00 (f_{SYS}/2)	**ADCS1, ADCS0=01 (f_{SYS}/8)**	**ADCS1, ADCS0=10 (f_{SYS}/32)**	**ADCS1, ADCS0=11**
1MHz	2μs	8μs	32μs	未定义
2MHz	1μs	4μs	16μs	未定义
4MHz	500ns*	2μs	8μs	未定义
8MHz	250ns*	1μs	4μs	未定义

A/D 时钟周期范例

A/D 输入引脚

所有的 A/D 模拟输入引脚都与 PB 端口的 I/O 引脚共用。ADCR 寄存器中的 PCR2~PCR0 位，决定是将输入引脚设置为普通的 PB 输入/输出引脚，还是将它们设置为模拟输入引脚，而不是由掩膜选项来决定。通过这种方式，引脚的功能可由程序来控制，从普通 I/O 操作功能到模拟输入，反过来也一样。当输入引脚作为普通 I/O 引脚使用时，可通过掩膜选项设置上拉电阻，若设置为 A/D 输入，则上拉电阻会自动断开。请注意，PB 端口控制寄存器并不需要为使能 A/D 输入，而先设定 A/D 引脚为输入引脚，当 PCR2~PCR0 位使能 A/D 输入时，不考虑端口控制寄存器的状态。电源脚 VDD 为 A/D 转换器的参考电压，模拟输入电压不可超过此电压值。另外须适当的量测 VDD，以确保该电压的稳定及减少噪声。

A/D 转换初始化

内部的 AD 转换器必须经过一个特殊的方法初始化。当 PB 端口转换通道选择位被改变，AD 转换器必须重新初始化。如果通道选择位改变后没有重新初始化，那么 EOCB 标志位可能会处于不确定的状态，这样可能会导致一个错误的转换结束信号。当通道选择位改变后，寄存器 ADCR 中的 START 位必须在 1~10 个指令周期内先置位再立即清零。这样可以保证 EOCB 被正确的置位。

A/D 转换步骤

下面总结实现 A/D 转换过程的各个步骤。

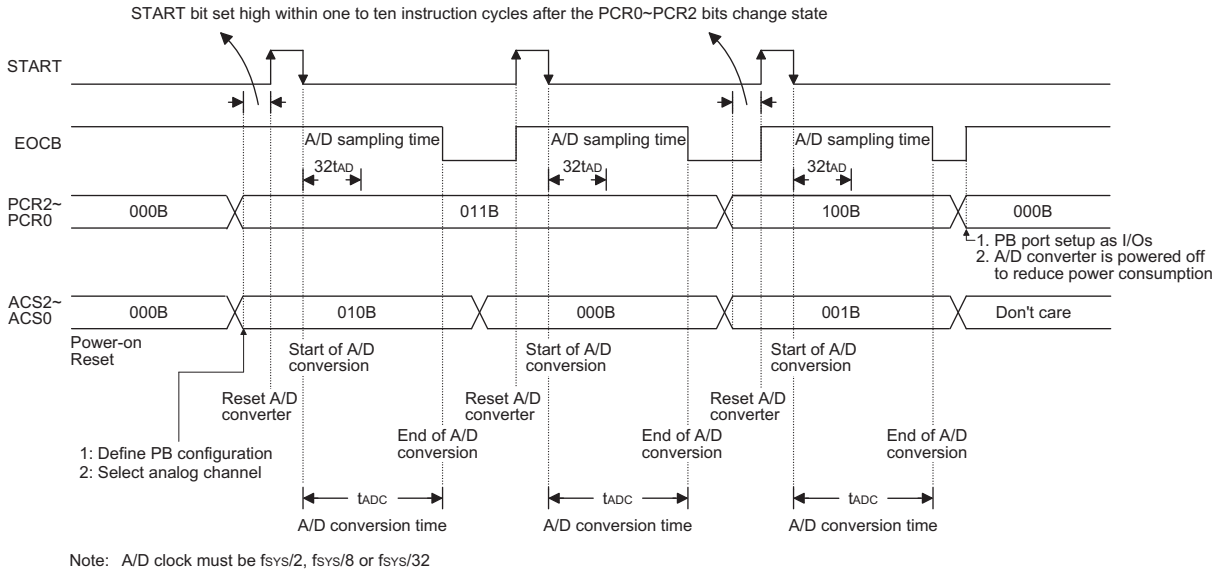
- 步骤 1
  通过 ACSR 寄存器中的 ADCS1 和 ADCS0 位，选择所需的 A/D 转换时钟。
- 步骤 2
  通过 ADCR 寄存器中的 ACS2~ACS0 位，选择连接至内部 A/D 转换器的通道。
- 步骤 3
  通过 ADCR 寄存器中的 PCR2~PCR0 位，选择 PB 端口的 A/D 输入引脚，并将它们设置为 A/D 输入引脚。此步骤也可在第二步写 ADCR 寄存器时完成。
- 步骤 4
  如果要使用中断，则中断控制寄存器必须正确地设置，以确保 A/D 功能的动作。中断控制寄存器 INTC 里总中断控制位 EMI 必须置位为“1”，A/D 转换器的中断使能位 EADI 也必须置位为“1”。
- 步骤 5
  通过设定 ADCR 寄存器中的 START 位从“0”到“1”再回到“0”，可以开始模数转换的过程。该位需初始化为“0”。

- 步骤 6

  可以轮询 ADCR 寄存器中的 EOCB 位，检查模数转换过程是否完成。当此位成为逻辑低时，表示转换过程已经完成。转换完成后，可读取 A/D 数据寄存器 ADRL 和 ADRH 获得转换后的值。另一种方法是，若中断使能且堆栈未满，则转换完成后，程序会进入 A/D 中断服务子程序。

注意：若使用轮询 ADCR 寄存器中 EOCB 位的状态的方法来检查转换过程是否结束时，步骤 4 可以省略。

下列时序图表示模数转换过程中不同阶段的图形与时序。



A/D 转换时序图

A/D 转换器没有对应的掩膜选项，它的功能设定与操作完全由应用程序控制。由应用程序控制开始 A/D 转换过程后，单片机的内部硬件就会开始进行转换，在这个过程中，程序可以继续其它功能。

编程注意事项

在编写程序时，必须特别注意寄存器 ADCR 中的 AD 转换通道选择位。如果这些全部为零，那么没有外部引脚连接到 AD 转换器上，此时的外部引脚可作为普通的 IO 口使用。这样可以减少 AD 转换部分的功耗，也降低了整个芯片的工作电流。关闭 AD 转换器以降低功耗，这一点对电池供电的系统非常重要。

另一个编程注意事项是，当 AD 转换通道选择位改变后，AD 转换器必须重新初始化。当通道选择位改变后，给寄存器 ADCR 的 START 位一个脉冲即可初始化 AD 转换器。当选择位全部被清零，由于不需要进行 AD 转换，可以不重新初始化 AD 转换器。

A/D 转换程序范例

下面两个范例程序说明如何设置和实现 A/D 转换。第一个范例通过不断扫描 ADCR 寄存器的 EOCB 位的方法来判断 A/D 转换是否完成，而第二个范例使用 A/D 中断方式判断转换完成。

例：通过扫描 EOCB 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作为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      EOCB            ; 扫描ADCR寄存器的EOCB位判断A/D转换是否完成
    jmp     polling_EOC     ; 继续扫描
    mov     a,ADRL          ; 从ADRH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
    mov     a,ADRH          ; 从ADRL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
            :
            :
    jmp     start_conversion ; 开始下一次A/D转换
```

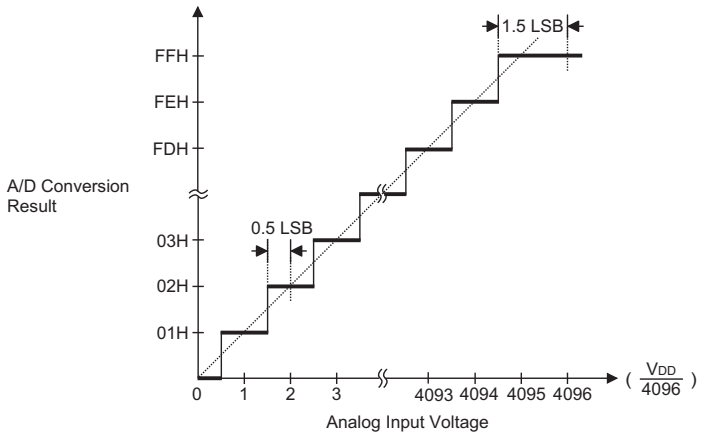
例：用中断方法判断 A/D 转换是否完成。

```
    clr EADI                ; 禁止 A/D 中断
    mov a,00000001B
    mov ACSR,a              ; 设置 ACSR 寄存器，选择 fSYS/8 作为 A/D 转换时钟
    mov a,00100000B         ; 在 ADCR 寄存器中设置 Port PB0~PB3 作为 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 读取转换结果的高位
    mov adrh_buffer,a       ; 保存到用户自己定义的寄存器中
    mov a,ADRL              ; 从 ADRL 读取转换结果的低位
    mov adrl_buffer,a       ; 保存到用户自己定义的寄存器中
    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
```

A/D 转换功能

HT46RU26/HT46CU26 单片机含有一组 12 位的 A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于 VDD 的电压值，因此每一位可表示 $V_{DD}/256$ 的模拟输入值。而对于其它型号单片机均含有一组 9 位的 A/D 转换器，它们转换的最大值可达 1FFH，每一位表示 $V_{DD}/512$ 的模拟输入值。

下图显示 A/D 转换器模拟输入值和数字输出值之间理想的转换功能。



理想的 A/D 转换功能

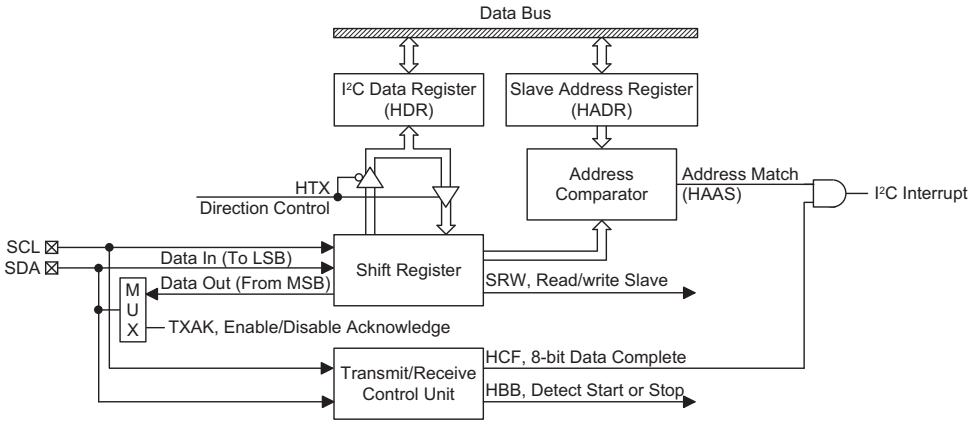
注意的是，为了减少量化错误，A/D 转换器输入端会加入 0.5LSB 的偏移量。除了数字化数值 0，其后的数字化数值会在精确点之前的 0.5LSB 处改变，而数字化数值的最大值将在 VDD 之前的 1.5 LSB 处改变。

I²C 总线接口

I²C 总线是最初有飞利浦半导体开发的双向两线通信接口。只通过 2 个引脚来实现数据的发送和接收的可能性，为为控制器的基础应用提供了许多新的应用的机会，因此，I²C 在此单片机上得以实现。I²C 总线功能由掩膜选项选择。

有两根引脚与 I²C 总线相连，第一个是串行数据线 SDA，第二个是串行时钟线 SCL。由于可能许多设备连接在相同的总线上，SDA 和 SCL 都是开漏输出。因此有必要外接上拉电阻与它们相连。注意的是，不存在片选线，每个 I²C 总线上的设备通过唯一的地址被识别，此地址会在 I²C 总线上被发送和接收。

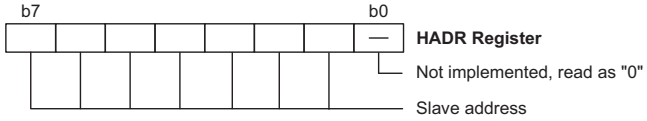
当两个设备在双向 I²C 总线上相互通信时，一个叫做主机，另一个叫做从机。主机和从机都可以进行数据的发送和接收，但是，主机具有总线过速控制能力。对于在从模式下运行的单片机，I²C 总线有两种数据传输方式，从器件发送模式和从器件接收模式。有四个与 I²C 总线相关寄存器，HADR、HCR、HSR、HDR。I²C 总线上的通信需要四步完成，一个起始信号，一个从器件地址发送，一个数据发送和一个最终的停止信号。



I²C 总线串行接口方框图

I²C 总线从器件地址寄存器－HADR

HADR 寄存器用来存放从器件地址。HADE 寄存器的第 1～7 位定义了 MCU 从器件地址，第 0 位没有用。如果主控制器发送的调用地址与该从器件地址匹配，则表明该器件被选中。



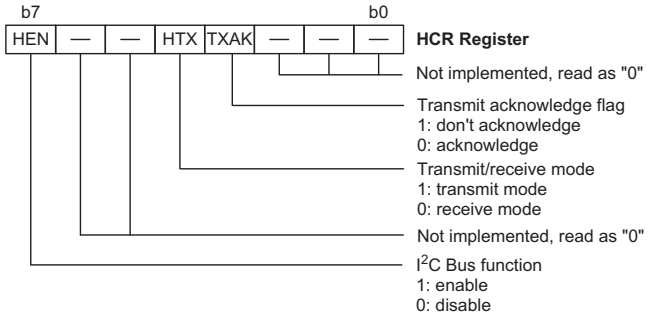
I²C 总线从器件地址寄存器

I²C 输入/输出数据寄存器－HDR

HDR 寄存器是 I²C 总线的数据输入/输出寄存器。在发送数据前，要先将发送的数据写到 HDR 寄存器；在接收数据前，要先从 HDR 虚拟读取数据。从 I²C 总线发送或接收数据都必须经过 HDR 寄存器。

I²C 总线控制寄存器－HCR

HCR 是 I²C 总线控制寄存器，用来控制 I²C 总线功能的开/关状态和 I²C 总线是工作于发送模式还是接收模式。HSR 是 I²C 总线状态寄存器，该寄存器反映了 I²C 总线的工作状态。I²C 总线控制寄存器包括 HCR3 位。第 7 位，即 HEN，控制 I²C 总线打开/关闭，如果数据想通过 I²C 总线传送，该位就必须置“1”。第 4 位，即 HTX，定义 I²C 总线是用于发送模式还是接收模式，如果要用于发送模式，该位就必须置“1”。第 3 位，即 TXAK，定义应答信号，当器件接收到 8 位数据后，在第 9 个时钟时，器件将该位送到 I²C 总线上，如果要继续接收下一个数据，在接收数据前该位必须清“0”。



I²C 总线控制寄存器

I²C 总线状态寄存器－HSR

I²C 总线寄存器 HSR 是 8 位的状态寄存器，包括 5 位可使用。第 7 位，即 HCF，在开始传送数据时被清“0” ，在数据传送结束后被置“1” 。第 6 位，即 HAAS，当从器件地址匹配时被置“1” ，同时 I²C 总线中断请求标志被置“1” ，如果中断允许且堆栈未满，则程序会跳到地址 14H 开始执行；写数据到 I²C 总线控制寄存器可以清除 HAAS；如果地址不匹配，HAAS 被清“0” 。

第 5 位，即 HBB，置“1”表示 I²C 总线忙，即系统检测到“START”信号；HBB 清“0” 表示 I²C 总线空闲，即系统检测到“STOP”信号，此时 I²C 总线空闲。第 2 位，即 SRW 或从器件读/写位，决定主控制器是发送数据还是从 I²C 总线接收数据。表示调用地址匹配时，器件的读/写状态。当 HAAS 被置“1”，可以通过检测 SRW 来确定器件是工作在发送模式还是接收模式。当 SRW 被置“1”，表示主控制器要从 I²C 总线读数据，从器件必须将数据写到 I²C 总线，即从器件为发送模式；当 SRW 被清“0” ，表示主控制器要写数据到 I²C 总线，从器件要从总线读取数据，即从器件为接收模式。

第 0 位，是接收应答位。当 RXAK 被复位“0” ，表示在发送完 8 位数据后，在第 9 个时钟时，接收到一个正确的应答信号。在发送模式，发送器件通过检测 RXAK 以确定接收器件是否要接收下一个数据，发送器件会一直写数据到 I²C 总线直到 RXAK 置“1”，同时发送器件释放 SDA 线，这样主控制器可以发送 STOP 信号来释放总线。

I²C 总线通信

I²C 总线上的通信需要四步完成，一个起始信号，一个从器件地址发送，一个数据发送和一个最终的停止信号。当起始信号被写入 I²C 总线时，所有的总线上的器件都会接收信号并且被通知总线上会立即有数据到达。数据的前 7 位是从器件地址，高位在前，低位在后。如果从器件地址匹配，系统会置位 HAAS，同时产生 I²C 总线中断。进入中断服务程序后，系统要检测 HAAS 位，以确定 I²C 总线中断是来自从器件地址匹配，还是来自 8 位数据传送完毕。

在数据传递中，注意的是，在 7 位从器件地址被发送后，接下来的一位，即第 8 位，是读/写控制位，该位的值会反映到 SRW。从器件通过检测 SRW 以确定主控制器是要进入发送模式还是接收模式。在 I²C 总线开始传送数据前，需要先初始化 I²C 总线，在初始化 I²C 总线时必须注意以下几点：

步骤 1

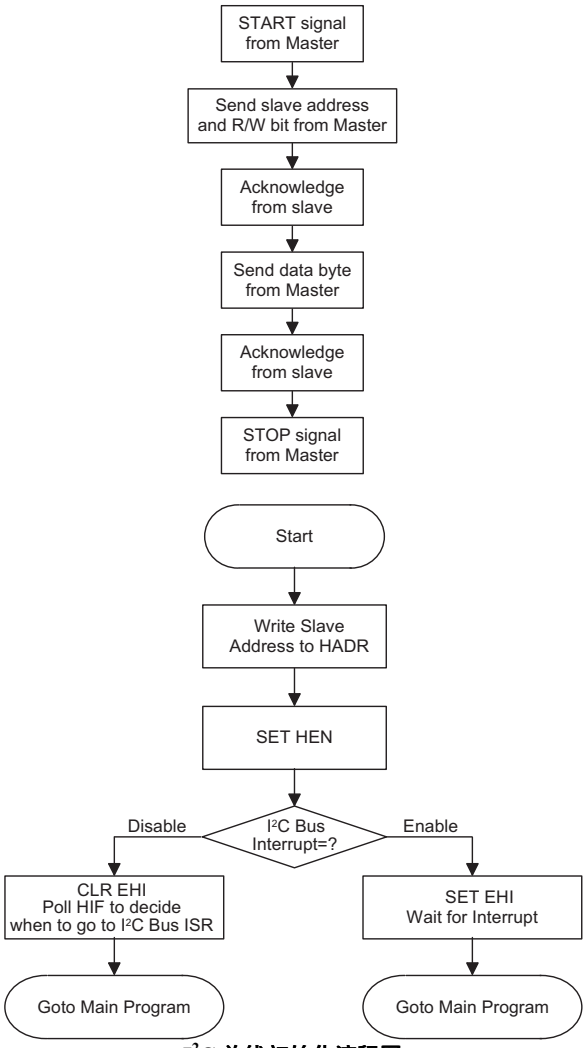
向 I²C 总线地址寄存器(HADR)写入从器件地址。

步骤 2

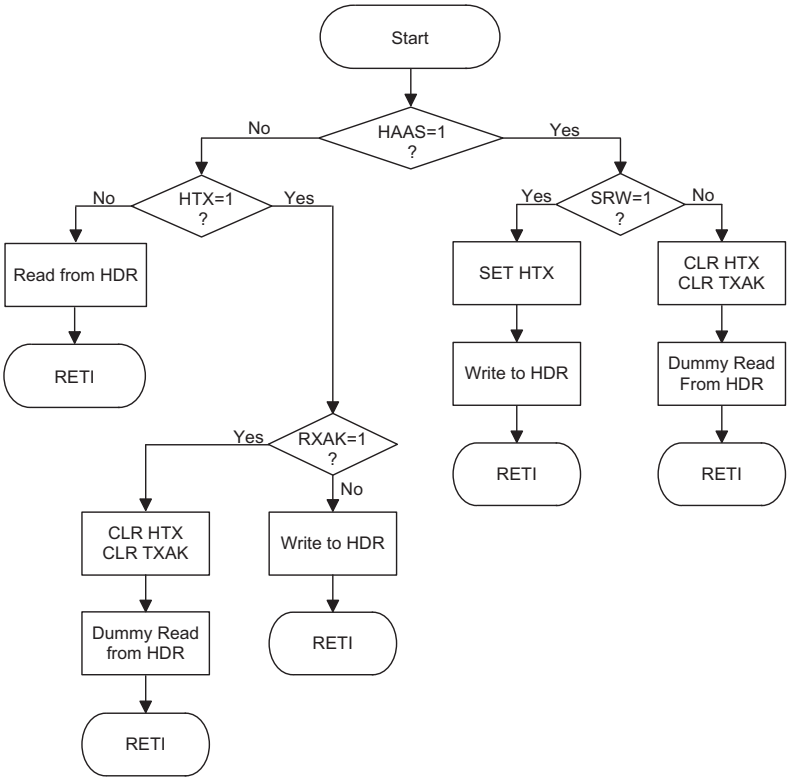
置位 I²C 总线控制寄存器(HCR)的 HEN 位，以打开 I²C 总线。

步骤 3

置位 I²C 总线中断控制寄存器 1(INTC1)的 EHI 位，以允许 I²C 总线中断。



I²C 总线初始化流程图



I²C 总线 ISR 流程图

起始信号

起始信号只能由连接 I²C 总线主控制器产生，而不是只是从器件的 MCU。总线上的所有器件必须侦测起始信号，当检测到后，表明 I²C 总线处于忙状态，因此会置位 HBB。起始信号是指在 SCL 为高时，SDA 发生电平从高到低的变化。

从器件地址

总线上的所有器件都会侦测由主机发起的起始信号的发送。发送起始信号后，主控制器必须发送从器件地址以选择要进行数据传输的从器件。所有在 I²C 总线上的从器件都会接收到这个从器件地址(7 位)，并与各自内部的从器件地址进行比较。如果从器件地址匹配，该从器件会产生一个中断，并将接下来的一位数据(即第 8 位)保存到 SRW 位，并发出一个应答信号，即第 9 位的低电平信号。当从器件地址匹配时，还会置位状态标志(HAAS)。

由于 I²C 总线有两个中断源，在中断服务子程序中，通过检测 HAAS 位可以确定 I²C 总线中断是来自从器件地址匹配，还是来自 8 位数据传送完毕。当是从器件地址匹配发生中断时，则器件必定是用于发送模式或接收模式，所以必须写数据到 HDR 或从 HDR 虚拟读取数据以释放 SCL 口线。

SRW 位

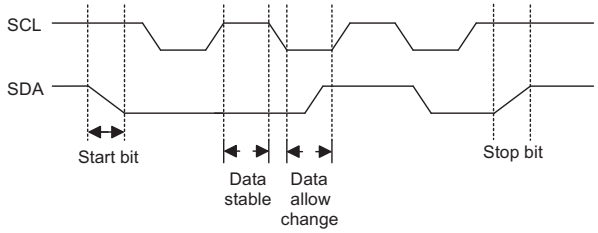
HSR 寄存器的 SRW 位表示主控制器是要从 I²C 总线上读取数据还是要将数据写到 I²C 总线上。从器件则通过检测该位以确定自己是作为发送器还是接收器。SRW 置“1”，表示主控制器要从 I²C 总线读数据，从器件必须将数据写到 I²C 总线，即从器件做为发送器；SRW 清“0”，表示主控制器要写数据到 I²C 总线，从器件要从总线读取数据，即从器件做为接收器。

应答位

在主控制器发送呼叫地址后，当 I^2C 总线上从器件内部地址与其匹配时，会发送一个应答信号。应答信号会通知主控制器从器件已经接收了呼叫地址。如果没有应答信号，主控制器必须发送停止(STOP)信号以结束通讯。当 I^2C 总线状态寄存器的第 6 位(HAAS)是高时，表示地址匹配，则从器件需检查 SRW，以确定自己是作为发送器还是作为接收器。如果 SRW 位为高，从器件须设置成发送器，这样会置位 HCR 寄存器的 HTX 位。如果 SRW 位为低，从器件须设置成接收器，这样会清零 HCR 寄存器的 HTX 位。

数据字节

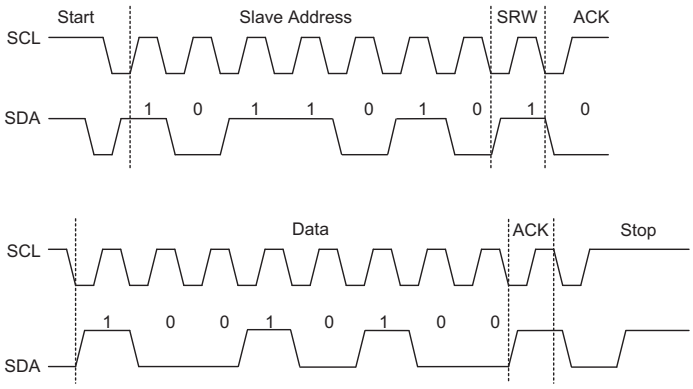
在从器件发出应答信号后，就会进行数据传输，一个数据长度为 8 位，高位在前，低位在后。接收器在接收到数据后会发出一个应答信号(“0”)以继续接收下一个数据。如果发送器没检测到应答信号，发送器将释放 SDA 线，同时，主控制器将发出 STOP 信号以释放 I^2C 总线。所传送的数据存储在 HDR 寄存器中。如果设置成发送器，从机必须将数据写到 HDR；如果设置成接收器，从机必须从 HDR 读取数据。



数据时序图

接收应答位

当接收器想要继续接收下一个数据时，必须在第 9 个时钟发出应答信号(TXAK)。发送器检测应答信号(RXAK)以决定是继续写数据到 I^2C 总线，还是改变为接收模式并虚读 HDR 寄存器以释放 SDA 线，同时主控制器发出停止信号。



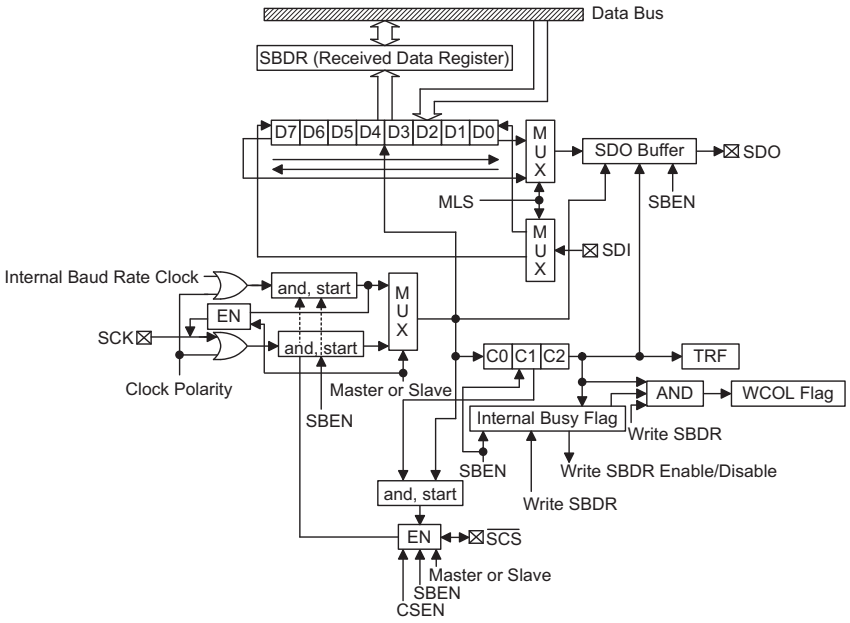
S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	--------	S	SA	SR	M	D	A	D	A	--------	P

I^2C 通讯时序图

SPI 串行接口

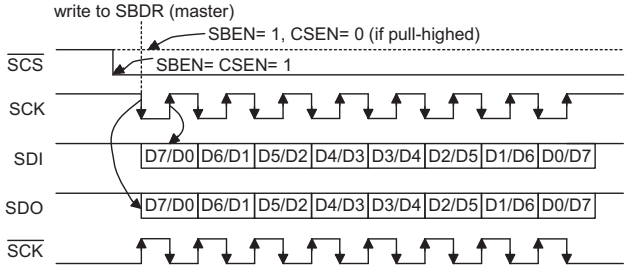
HT46RU26/HT46CU26 包含一个 SPI 串行接口。SPI 是一个全双工串行数据传输器，最初由摩托罗拉设计，其允许多种设备通过 SPI 总线进行相互通信。设备之间通过主/从技术，只有主机能够发起数据的传递。一个简单的四线信号总线被用来进行所有的通信，并且这些引脚与普通的 I/O 口共用引脚。SPI 功能由掩膜选项选择。



SPI 结构图

串行接口通信

串行接口功能有 4 个基本信号线，包含 SDI（串行数据输入），SDO（串行数据输出），SCK（串行时钟）和 SCS（从器件选择）。注意的是，从机选择线的条件是由 SBCR 控制寄存器内的 CSEN 位决定的。如果 CSEN 位被置位，SCS 线有效，但如果被清零，那么 SCS 线将处于浮空状态。下面的时序图描述了基本的 SPI 总线时序协议。



SPI 总线时序

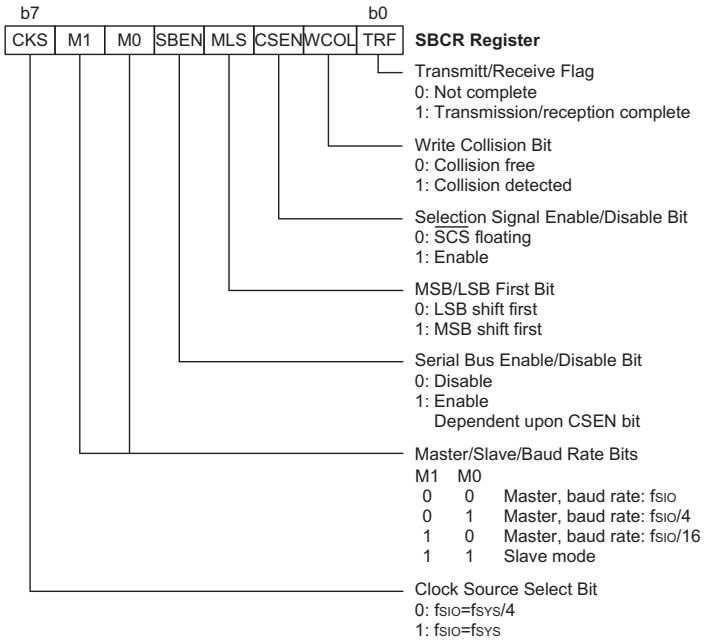
SPI 寄存器

有两个寄存器与 SPI 接口相关。SBCR 寄存器为控制寄存器，SBDR 为数据寄存器。SBCR 寄存器被用来设置 SPI 总线必须的设置参数和储存相关的操作标识位，同时，SCDR 寄存器被用作数据的存放。

上电后，SBDR 寄存器内的内容将处于未知状态，然后，SBCR 寄存器会形成如下默认的状态：

CKS	M1	M0	SBEN	MLS	CSEN	WCOL	TRF
0	1	1	0	0	0	0	@5

注意的是，写入 SBDR 寄存器的数据只会被写入到 TXRX 缓冲器，然而从 SBDR 读出的数据实际上就从此寄存器读出。



SPI 接口控制寄存器

SPI 总线 使能/禁止

使能 SPI 总线 CSEN=1，$\overline{SCS}$ =0 然后等待数据写入 SBDR（TXRX 缓冲）；在主模式下，将数据写入 SBDR（TXRX 缓冲）后自动开始发送/接收，数据发送完成后，系统将 TRF 标志位置 1。在从模式下：当收到 SCK 信号（$\overline{SCS}$ 取决于 CSEN）时，TXRX 缓冲内数据按位移出的同时 SDI 引脚上数据按位移进。

禁止 SPI 总线 SCK，SDI，SDO，$\overline{SCS}$ 应处于浮空状态。

SPI 操作

在主/从机模式下，所有的通信都通过使用这线接口得以实现。时序图显示了基本的总线操作。

SBCR 寄存器的 CSEN 位控制 SPI 接口的全部功能。CSEN=1→SCS 信号（可用来控制 SPI 接口）有效，将使能 SPI 接口。如果 CSEN=0，SPI 接口将被禁止，并且引脚将处于浮空状态，因此不会用来控制 SPI 接口。SBCR 寄存器的 SBEN 位也必须置位，其可以将 SDI 引脚设成浮空状态并且将 SDO 引脚置高。如果在主模式下，SCK 引脚将根据掩膜选项中的时钟极性被置高或置低。如果在从模式下，SCK 引脚将处于浮空状态。如果 SBEN 位清零，那么总线将被禁止，并且 SCS，SDI，SDO 和 SCK 将全都处于浮空状态。

在主模式下，主机将始终产生时钟信号。在数据被写入 SBDR 寄存器后，将启动时钟和数据的传递。在从模式下，数据的传递和接受将通过接收来自外部主机设备的时钟信号来启动。下面的步骤显示了在主/从模式下数据传递遵循的顺序。

- 主模式：
  - 步骤 1
    使用 SBCR 控制寄存器的 CKS 位来选择时钟源。
  - 步骤 2
    设置 SBCR 控制寄存器的 M0 和 M1 位来选择主模式和必须的波特率。00，01 和 10 值可被选择。
  - 步骤 3
    设置 CSEN 和 MLS 来选择数据从高位还是地位开始，从机必须保持一致。
  - 步骤 4
    设置 SBCR 控制寄存器的 SBEN 位来使能 SPI 接口。
  - 步骤 5
    写操作：将数据写入 SBDR→数据存入 TXRX 缓冲→输出 CLK（和）信号→使用 SCK 和 SCS 引脚输出数据→跳至步骤 6。
    读操作：SDI 数据移位写入 TXRX 缓冲→数据传输，TXRX 缓冲数据锁存至 SBDR。
  - 步骤 6
    检查 WCOL：WCOL=1→发生冲突错误，并跳至步骤 5。
    WCOL=0→跳至步骤 7。
  - 步骤 7
    检查 TRF 或等待 SBI 串行总线中断。
  - 步骤 8
    从 SBDR 寄存器读取数据。
  - 步骤 9
    清除 TRF 。
  - 步骤 10
    返回步骤 5。

- 从模式：
  - 步骤 1
    在从模式下，忽略 CKS。
  - 步骤 2
    设置的 M0 和 M1 位为 00 来选择从模式。忽略 CKS。
  - 步骤 3
    设置 CSEN 和 MLS 来选择数据从高位还是地位开始，主机必须保持一致。
  - 步骤 4
    设置 SBCR 控制寄存器的 SBEN 位来使能 SPI 接口。
  - 步骤 5
    写操作:将数据写入 SBDR→数据存入 TXRX 寄存器缓冲→等待主机时钟和 SCS 信号→跳至步骤 6。
    读操作：SDI 数据移位写入 TXRX 缓冲→数据传输，TXRX 缓冲数据锁存至 SBDR
  - 步骤 6
    检查 WCOL：WCOL=1→发生冲突错误，并跳至步骤 5；
    WCOL=0→跳至步骤 7；
  - 步骤 7
    检查 TRF 或等待 SBI 串行总线中断

- 步骤 8
  从 SBDR 寄存器读取数据
- 步骤 9
  清除 TRF
- 步骤 10
  返回步骤 5

SPI 掩膜选项

一些掩膜选项必须通过设备的编程来设置后用作 SPI 接口功能。一个掩膜选项用来在 SBCR 寄存器里使能 WCOL 的操作和写冲突位。另一个掩膜选项用来选择 SCK 引脚的时钟极性。掩膜选项同样用来禁止或使能 SBCR 寄存器中的 CSEN 位。如果掩膜选项禁止 CSEN 位，那么 CSEN 将不能用来影响 SPI 接口所有的控制。

错误检测

SBCR 寄存器的 WCOL 位用来在数据传送中提示错误。WCOL 位通过串行接口被置位，但必须通过应用程序来清零。当数据传递操作中出现写 SBCR 寄存器的现象时，WCOL 位会提示数据冲突并且防止继续进行写操作。WCOL 位将通过串行接口被置位，但必须通过用户应用程序来清零。WCOL 位的全部功能可以通过掩膜选项来禁止或使能。

编程注意事项

在设备进入暂停模式下，注意的是，数据的接收和传递将继续进行。在数据传递和接收后，TRF 位用来产生中断。

异步串行口 — UART

HT46RU26/ HT46CU26 具有一个全双工的异步串行通信口，可以很方便的与其它具有串行口的芯片通讯。UART 具有许多功能特性，发送或接收一个 8 位或 9 位数据帧的串行数据，当数据超速或数据帧不正确时，UART 可以检测出错误。UART 功能占用一个内部中断向量，当接收到数据或数据发送结束，触发 UART 中断。

- **UART 特性**

UART 具有以下功能：
- 全双工异步传输
- 8 位或 9 位传输格式
- 奇校验、偶校验或无校验
- 1 位或 2 位停止位
- 8 位预分频的波特率发生器
- 奇偶、帧、噪声和过速检测
- 支持地址匹配中断（最后一位=1）
- 独立的发送和接收允许
- 两层 FIFO 接收缓冲器
- 发送和接收中断
- 下列条件可触发中断：
  - 发送器为空
  - 发送器空闲
  - 接收器完成
  - 接收器过速
  - 地址匹配

- **UART 外部引脚**

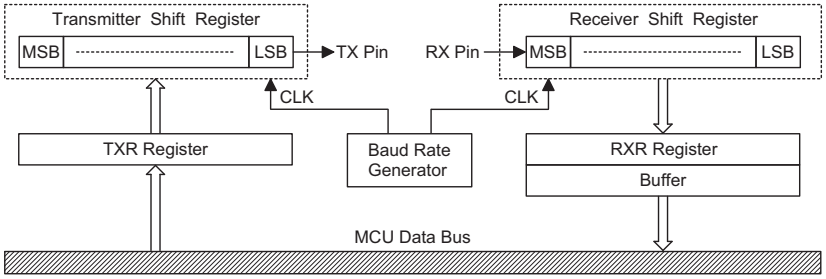
UART 是通过两个外部引脚 TX 和 RX 与外部芯片通讯。TX 引脚是 UART 的发送引脚，当 UCR2 寄存器的 TXEN 位清零，UART 发送功能被禁止，则 TX 引脚可作为普通 IO 口使用。RX 引脚是 UART 的接收引脚。同样的，当 UCR2 寄存器的 RXEN 位清零，UART 接收功能被禁止，则 RX 引脚可作为普通 IO 口使用。若 UARTEN、TXEN 和 RXEN 置位，这些 IO 口将自动转换成相应 TX 输出和 RX 输入，并且 RX 脚的上拉电阻将无效。

- **数据传送框图**

下图显示了 UART 的整体结构。需要发送的数据首先写入 TXR 寄存器，然后在波特率发生器的控制下将寄存器中数据以低位在前的方式一位位地移到 TX 引脚上。TXR 寄存器被映射到单片机的数据存储器中，而发送移位寄存器没有实际地址，所以发送移位寄存器不可直接操作。

数据在波特率发生器的控制下，数据以低位在前的方式从外部引脚 RX 进入接收移位寄存器。当移位寄存器已满，数据从接收移位寄存器移入可被用户程序操作的 RXR 寄存器。RXR 寄存器被映射到单片机数据存储器中，而接收移位寄存器没有实际地址，所以接收移位寄存器不可直接操作。

注意，上述发送寄存器 TXR 和接收寄存器 RXR，其实是共享一个地址的数据寄存器 TXR/RXR 寄存器。



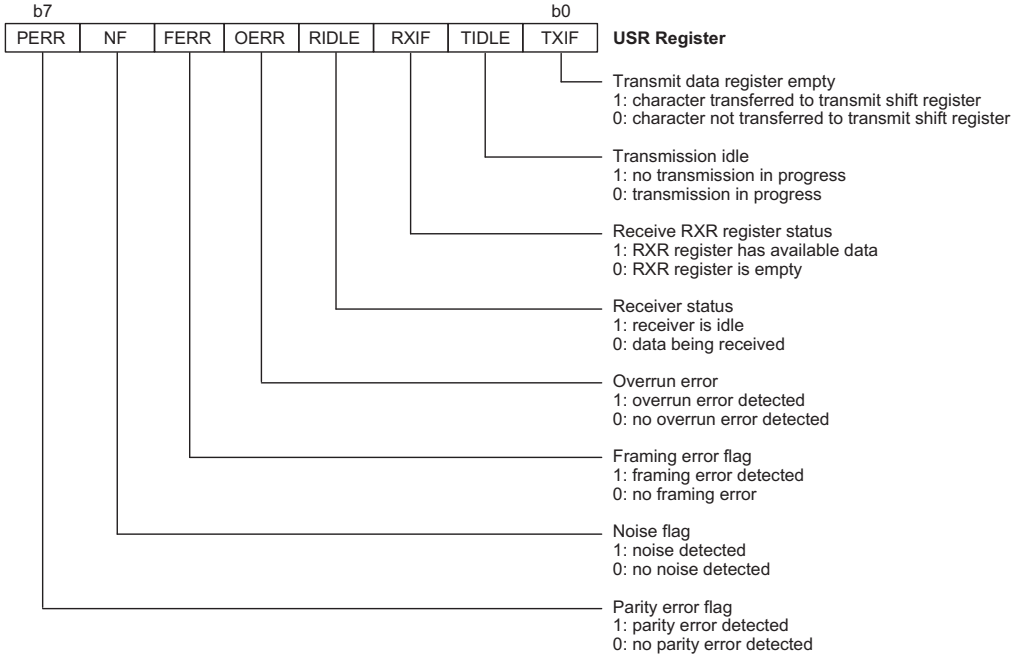
UART 数据发送接收图

- **UART 状态控制寄存器**

有 5 个寄存器与 UART 功能相关，寄存器 USR、UCR1 和 UCR2 全面控制 UART，而寄存器 BRG 控制波特率，发送和接收数据则通过寄存器 TXR/RXR。

- **USR 寄存器**

寄存器 USR 是 UART 的状态寄存器，可以通过程序读取 UART 当前状态。USR 中所有标志位为只读。USR 中所有标志位详细解释如下：



- TXIF

TXIF 是发送数据寄存器为空标志。若 TXIF=0，数据还没有从缓冲器加载到移位寄存器中；若 TXIF=1，数据已从 TXR 寄存器中加载到移位寄存器。读取 USR 寄存器再写 TXR 寄存器将清除 TXIF。当 TXEN 被置位，即使发送缓冲器未满，TXIF 也会被置位。

- TIDLE

TIDLE 是数据发送完成标志位。若 TIDLE=0，表明数据传输中。当 TXIF=1 且数据发送完毕或暂停字被发送时，TIDLE 置位。TIDLE=1，TX 引脚空闲。读取 USR 寄存器再写 TXR 寄存器将清除 TIDLE 位。当数据字节或暂停字符排列好并准备发送时，TIDLE 不发生变化。

- RXIF

RXIF 是接收寄存器状态标志。当 RXIF=0，表明 RXR 寄存器为空；当 RXIF=1，表明 RXR 寄存器接收到新数据。当数据从移位寄存器加载到 RXR 寄存器，如果 UCR2 寄存器中的 RIE=1，则会触发中断。当接收数据时检测到一个或多个错误时，相应的标志位 NF、FERR 或 PERR 会在同一周期内置位。读取 USR 寄存器再读 RXR 寄存器，如果 RXR 寄存器中没有新的数据，那么将清除 RXIF 标志。

- RIDLE

RIDLE 是接收状态标志。若 RIDLE=0，表明正在接收数据；若 RIDLE=1，表明接收器空闲。在接收到停止位和下一个数据的起始位之间，RIDLE 被置位，表明 UART 空闲。

- OERR

OERR 是过速错误标志，表示接收缓冲器溢出。若 OERR=0，表明没有数据溢出；若 OERR=1，表明发生了过速错误，它将禁止下一组数据的接收。先读取 USR 寄存器再读 RXR 寄存器将清除此标志位。

- FERR

FREE 是帧错误标志位。若 FREE=0，表明没有帧错误发生；若 FREE=1，表明当前的数据发生了帧错误。先读取 USR 寄存器再读 RXR 寄存器将清除此位。

- NF

NF 是噪声干扰标志。若 NF=0，表明没有受到噪声干扰；若 NF=1，表明 UART 接收数据时受到噪声干扰。它与 RXIF 在同一个周期内置位，但不会与过速标志位同时置位。先读取 USR 寄存器再读 RXR 寄存器将清除此标志位。

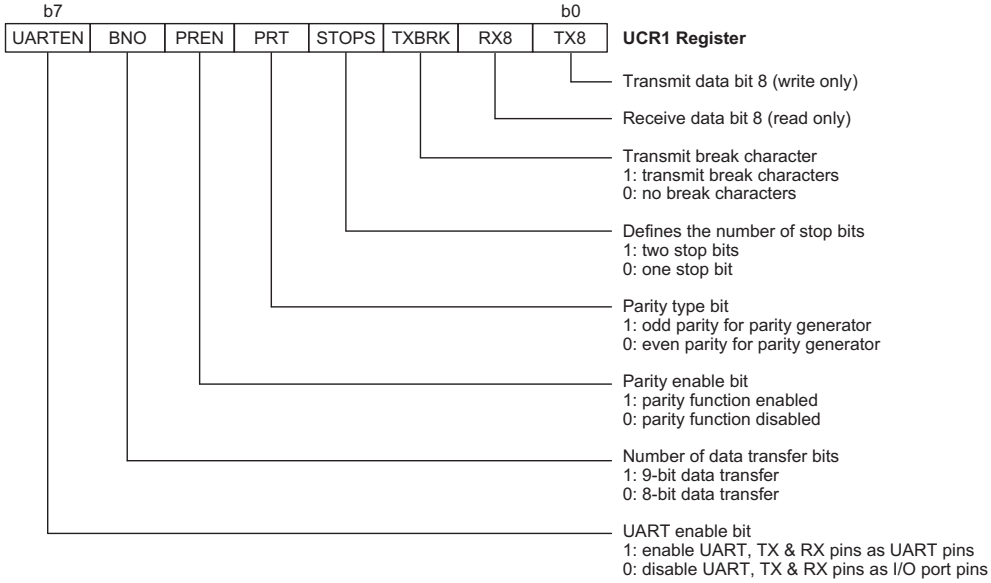
- PERR

PERR 是奇偶校验出错标志。若 PERR=0，表明奇偶校验正确；若 PERR=1，表明接收到的数据奇偶校验出错。只有使能了奇偶校验此位才有效。先读取 USR 寄存器再读 RXR 寄存器将清除此位。

- **UCR1 寄存器**

UCR1 和 UCR2 是 UART 的两个控制寄存器，用来定义各种 UART 功能，例如 UART 的使能与除能、奇偶校验控制和传输数据的长度等等。

UCR1 中标志位详细解释如下：



- TX8

此位只有在传输数据为 9 位的格式中有效，TX8 用来存储发送数据的第 9 位。BNO 是用来控制传输位数是 8 位还是 9 位。

- RX8

此位只有在传输数据为 9 位的格式中有效，RX8 用来存储接收数据的第 9 位。BNO 是用来控制传输位数是 8 位还是 9 位。

- TXBRK

TXBRK 是暂停字发送控制位。若 TXBRK=0，表明没有暂停字要发送，TX 引脚正常操作；若 TXBRK=1，表明将会发送暂停字，发送器将发送逻辑 0。若 TXBRK 为高，缓冲器中数据发送完毕后，发送器将至少保持 13 位宽的低电平直至 TXBRK 复位。

- STOPS

  此位用来设置 1 位或 2 位停止位的长度。若 STOPS=1，表明有两位停止位；若 STOP=0，表明只有一位停止位。

- PRT

  奇偶校验选择位。若 PRT=1，表明选择奇校验；若 PRT=0，表明选择偶校验。

- PREN

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

- BNO

  选择数据长度为 8 位或 9 位格式。若 BNO=1，表明传输数据为 9 位；若 BNO=0，表明传输数据为 8 位。若选择了 9 位数据传输格式，RX8 和 TX8 将分别存储接收和发送数据的第 9 位。

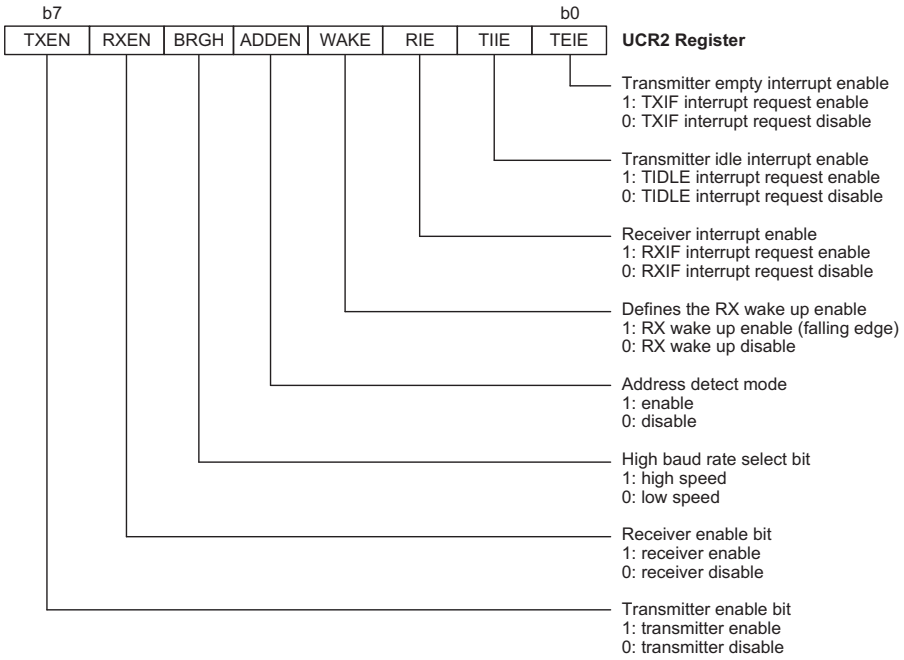
- UARTEN

  此位为 UART 的使能位。若 UARTEN=0，表明 UART 除能，RX 和 TX 可用作普通输入/输出口；若 UARTEN=1，表明 UART 使能，TX 和 RX 将分别由 TXEN 和 RXEN 控制。当 UART 除能，系统将清除缓冲器，所有缓冲器中的数据将被忽略，另外波特率计数器、错误和状态标志位被复位，TXEN、RXEN、TXBRK、RXIF、OERR、FERR，PERR 和 NF 清零；而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2 和 BRG 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零，所有发送和接收将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

- **UCR2 寄存器**

UCR2 是 UART 的另一个控制寄存器，它的主要功能是使能或除能发送和接收允许以及 UART 的各种中断源。它也可用来控制波特率，使能接收唤醒和地址侦测。

UCR2 中标志位详细解释如下：



- TEIE

  此位为发送寄存器为空时中断的使能或除能位。若 TEIE=1，当发送器为空时 TXIF 将置位，UART 的中断请求标志置位；若 TEIE=0，UART 中断请求标志不受 TXIF 的影响。

- TIIE

此位为发送器空闲时中断的使能或除能位。若 TIIE=1，当发送器空闲时 TIDLE 置位，UART 的中断请求标志置位；若 TIIE=0，UART 中断请求标志不受 TIDLE 的影响。

- RIE

此位为接收器中断使能或除能位。若 RIE=1，当接收器过速或接收数据有效时 OERR 或 RXIF 置位，UART 的中断请求标志置位；若 RIE=0，UART 中断请求标志不受 OERR 或 RXIF 影响。

- WAKE

此位为接收器唤醒功能的使能和除能位。若 WAKE=1 且在暂停模式下，RX 引脚的下降沿将唤醒单片机。若 WAKE=0 且在暂停模式下，RX 引脚的任何边沿都不能唤醒单片机。

- ADDEN

此位为地址检测使能和除能位。若 ADDEN=1，表明地址检测使能，此时数据的第 8 位（BON=0 时与 RX7 相应）或第 9 位（BON=1 时与 RX8 相应）为高，接到的是地址而非数据。若相应的中断使能且接收到的值最高位为 1，那么中断请求标志将会被置位，若最高位为 0，那么将不会产生中断且收到的数据也会被忽略。

- BRGH

此位为波特率发生器高低速选择位，它和 BRG 寄存器一起控制 UART 的波特率。若 BRGH=1，为高速模式；若 BRGH=0，为低速模式。

- RXEN

此位为接收使能位。若 RXEN=0，接收器将被除能，接收器停止工作。另外缓冲器将被复位，此时 RX 引脚可作普通输入/输出端口使用。若 RXEN=1 且 UARTEN=1，则接收将被使能，RX 引脚将由 UART 来控制。在数据传输时清除 RXEN 将中止数据接收且复位接收器，此时 RX 引脚可作为普通输入输出端口使用。

- TXEN

此位为发送使能位。若 TXEN=0，发送器将被除能，发送器停止工作。另外缓冲器将被复位，此时 TX 引脚可作为普通输入/输出端口使用。若 TXEN=1 且 UARTEN=1，则发送将被使能，TX 引脚将由 UART 来控制。在数据传输时清除 TXEN 将中止数据发送且复位发送器，此时 TX 引脚可作为普通的输入输出口使用。

- **波特率发生器**

UART 内建一个波特率发生器，可以设定数据传输速率。波特率是由一个独立的内部 8 位计数器产生，由 BRG 寄存器和 UCR2 寄存器的第 2 位 BRGH 来控制。BRGH 是决定波特率发生器处于高速模式还是低速模式，从而决定计算公式的选用。BRG 寄存器的值 N 可根据下表中的公式计算，范围是 0 到 255。

UCR2 的 BRGH 位	0	1
波特率	$\frac{f_{SYS}}{64(N+1)}$	$\frac{f_{SYS}}{16(N+1)}$

为了得到相应的波特率，首先需要设置 **BRGH** 来选择相应的计算公式从而算出 **BRG** 的值。由于 **BRG** 的值不连续，所以实际波特率和理论值之间有一个偏差。下面举例怎样计算 **BRG** 寄存器中的值 **N** 和误差。

波特率和误差的计算

系统选用 8M 晶振且 BRGH=0，若期望的波特率为 9600，计算它的 BRG 寄存器的值 N，实际波特率和误差。

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

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

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

取最接近的值，十进制 12 写入 BRG 寄存器，实际波特率如下

BR = $\frac{8000000}{64(12+1)}=9615$

误差 $\frac{9615-9600}{9600}=0.16\%$

下表给出 BRGH 取不同值时的实际波特率和误差。

波特率 K/BPS	BRGH=0											
	f_{SYS} =8MHz			f_{SYS} =7.159MHz			f_{SYS} =4MHz			f_{SYS} =3.579545MHz		
	BRG	Kbaud	Error	BRG	Kbaud	Error	BRG	Kbaud	Error	BRG	Kbaud	Error
0.3	-	-	-	-	-	-	207	0.300	0.00	185	0.300	0.00
1.2	103	1.202	0.16	92	1.203	0.23	51	1.202	0.16	46	1.19	-0.83
2.4	51	2.404	0.16	46	2.38	-0.83	25	2.404	0.16	22	2.432	1.32
4.8	25	4.807	0.16	22	4.863	1.32	12	4.808	0.16	11	4.661	-2.9
9.6	12	9.615	0.16	11	9.322	-2.9	6	8.929	-6.99	5	9.321	-2.9
19.2	6	17.857	-6.99	5	18.64	-2.9	2	20.83	8.51	2	18.643	-2.9
38.4	2	41.667	8.51	2	37.29	-2.9	1	-	-	1	-	-
57.6	1	62.5	8.51	1	55.93	-2.9	0	62.5	8.51	0	55.93	-2.9
115.2	0	125	8.51	0	111.86	-2.9	-	-	-	-	-	-

BRGH=0 时的波特率和误差

波特率 K/BPS	BRGH=1											
	f_{SYS} =8MHz			f_{SYS} =7.159MHz			f_{SYS} =4MHz			f_{SYS} =3.579545MHz		
	BRG	Kbaud	Error	BRG	Kbaud	Error	BRG	Kbaud	Error	BRG	Kbaud	Error
0.3	-	-	-	-	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	207	1.202	0.16	185	1.203	0.23
2.4	207	2.404	0.16	185	2.405	0.23	103	2.404	0.16	92	2.406	0.23
4.8	103	4.808	0.16	92	4.811	0.23	51	4.808	0.16	46	4.76	-0.83
9.6	51	9.615	0.16	46	9.520	-0.832	25	9.615	0.16	22	9.727	1.32
19.2	25	19.231	0.16	22	19.454	1.32	12	19.231	0.16	11	18.643	-2.9
38.4	12	38.462	0.16	11	37.287	-2.9	6	35.714	-6.99	5	37.286	-2.9
57.6	8	55.556	-3.55	7	55.93	-2.9	3	62.5	8.51	3	55.930	-2.9
115.2	3	125	8.51	3	111.86	-2.9	1	125	8.51	1	111.86	-2.9
250	1	250	0	-	-	-	0	250	0	-	-	-

BRGH=1 时的波特率和误差

- **UART 设置与控制**

  - 绪论

    UART 采用标准的不归零码传输数据，这种方法通常被称为 NRZ 法。它由 1 位起始位，8 位或 9 位数据位和 1 位或者 2 位停止位组成。奇偶校验是由硬件自动完成的，可设置成奇校验、偶校验和无校验三种格式。最常用的数据传输格式由 8 位数据位，无校验和 1 位停止位组成，用 8，N，1 表示，它是系统上电的默认格式。数据位数、停止位数和奇偶校验由 UCR1 寄存器的 BNO、PRT、PREN 和 STOPS 设定。用于数据发送和接收的波特率由一个内部的 8 位定时器产生，数据发送和接收时低位在前。尽管 UART 发送器和接收器在功能上相互独立，但它们使用相同的数据传输格式和波特率，在任何情况下，停止位是必须的。

  - UART 的使能和除能

    UART 功能是由 UCR1 寄存器的 UARTEN 位来使能和除能的。它的发送引脚 TX 和接收引脚 RX 分别与输入输出引脚复用，UARTEN 的一个基本功能就是控制这两个引脚。若 UARTEN、TXEN 和 RXEN 都为高，则 TX 和 RX 分别为 UART 的发送端口和接收端口，而不能作为普通的输入输出端口使用。若没有数据发送，TX 引脚默认状态为高电平。

    UARTEN 清零将除能 TX 和 RX，使其作为普通的输入输出端口使用。当 UART 被除能，缓冲器将复位为空状态，所有缓冲器中的数据将被忽略，另外错误和状态标志位被复位，TXEN、RXEN、TXBRK、RXIF、OERR、FERR，PERR 和 NF 清零，而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2 和 BRG 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零，所有发送和接收都将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

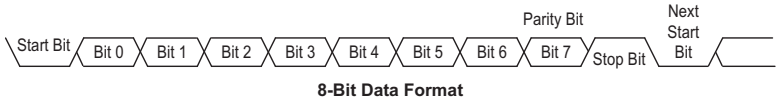
  - 数据位、停止位数以及奇偶校验的选择

    数据传输格式由数据长度、是否校验、校验类型，地址位以及停止位数组成，它们都由 UCR1 寄存器的各个位控制。BNO 决定数传输数据是 8 位还是 9 位；PRT 决定校验类型为奇校验或偶校验；PRTEN 决定是否选择奇偶校验；而 STOPS 决定选用 1 位或 2 位停止位。下表列出了各种数据传输格式。地址位用来确定此帧是否为地址。停止位为 1 位或两位与数据长度无关。

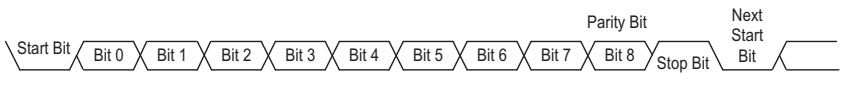
起始位	数据位	地址位	校验位	停止位
8 位数据格式				
1	8	0	0	1
1	7	0	1	1
1	7	1[1]	0	1
9 位数据格式				
1	9	0	0	1
1	8	0	1	1
1	8	1[1]	0	1

发送和接收数据格式

下图是传输 8 位和 9 位数据格式时的波形。



8-Bit Data Format



9-Bit Data Format

- UART 发送器

UCR1 寄存器的 BNO 位是控制传输数据的长度。当 BNO=1 时，其长度为 9 位，第 9 位为最高位，存储在 UCR1 寄存器的 TX8 中。发送器的核心是发送移位寄存器 TSR，它的数据由发送寄存器 TXR 提供，应用程序只须将发送数据写入 TXR 寄存器。上一组数据的停止位发出前，TSR 寄存器禁止写入。一旦停止位发出，TSR 寄存器仍可从 TXR 寄存器加载新的发送数据。注意的是，TSR 寄存器不像其它寄存器一样映射在数据存储器中，所以应用程序不能对其进行直接的读写操作。当 TXEN=1 时，发送使能，但若 TXR 寄存器没有数据或者波特率没有设置，发送器将不会工作。写 TXR 寄存器后再设置 TXEN 为高也会启动发送。当发送器开始工作，若 TSR 寄存器为空，数据写入 TXR 寄存器将会直接加载到 TSR 寄存器中。若数据发送过程中，TXEN 清零，发送器将立刻停止工作并且复位，此时 TX 引脚可作为普通的输入输出口使用。

- 发送数据

当 UART 发送数据时，数据从移位寄存器中以低位在前的方式移到 TX 引脚。在发送模式中，TXR 寄存器在内部总线和发送移位寄存器间形成一个缓冲。注意的是，选择 9 位数据传输格式，最高位 MSB 数据存储在 UCR1 寄存器的 TX8 中。

发送器初始化可由如下步骤完成：

--正确设置 BNO、PRT、PREN 和 STOPS 位以确定数据长度、校验类型和停止位长度。

--设置 BRG 寄存器，选择期望的波特率。

--设置 TXEN 为高，使 TX 引脚作为 UART 的发送端而非普通的输入输出端口。

--读取 USR 寄存器，然后将待发数据写入 TXR 寄存器，注意此步骤会清除 TXIF 标志位。

--如果要发送多个数据只需重复上一步骤。

注意的是，当 TXIF=0 时，数据将禁止写入 TXR 寄存器。软件中可以通过以下步骤来清除 TXIF：

1. 读取 USR 寄存器
2. 写 TXR 寄存器

只读标志位 TXIF 由 UART 硬件置位。若 TXIF=1，表明 TXR 寄存器为空，其它数据可以写入而不会覆盖以前的数据。若 TEIE=1，TXIF 标志位会产生中断。

在数据传输时，写 TXR 寄存器会将待发数据暂存在 TXR 寄存器中，当前数据发送完毕后，待发数据被加载到发送移位寄存器中。当发送器空闲时，写 TXR 寄存器会将数据直接加载到 TSR 寄存器中，数据传输立刻开始且 TXIF 置位。当一帧数据发送完毕（停止位或暂停字发出后），TIDLE 将被置位。

可以通过以下步骤来清除 TIDLE：

1. 读取 USR 寄存器
2. 写 TXR 寄存器

清除 TXIF 和 TIDLE 软件执行次序相同。

- 发送暂停字

若 TXBRK=1，下一帧将会发送暂停字。暂停字包含一个起始位、13*N（N=1，2 等）位逻辑 0 以及停止位组成。如果要发送暂停字，首先在应用程序中将 TXBRK 置为高，而将 TXBRK 清 0 将产生停止位，传输暂停字时不会产生中断。需要注意的是，暂停字至少 13 位宽。若 TXBRK 持续为高，那么发送器会一直发送暂停字；当应用程序中将 TXBRK 清 0，发送器将传输最后一帧暂停字再加上一位或者两位停止位。暂停字后的自动高电平保证了下一帧数据起始位的检测。

- **UART 接收器**

  - 绪论

    UART 接收器支持 8 位或 9 位数据的接收。若 BNO=1，数据长度为 9 位，而最高位 MSB 数据存放在 UCR1 寄存器的 RX8 中。接收器的核心是串行移位寄存器 RSR。RX 引脚上的数据送入数据恢复模块，它在 16 倍波特率的频率下工作，而串行移位器在正常波特率下工作。当在 RX 引脚上检测到停止位，如果接收寄存器为空，数据将从 RSR 寄存器加载到 RXR 寄存器。RX 引脚上的每一位数据会被采样三次以判断其逻辑状态。注意的是，RSR 不像其它寄存器一样映射在数据存储器，所以应用程序中不能直接对其进行读写操作。

  - 接收数据

    当 UART 接收数据时，数据以低位在前方式连续地从 RX 引脚输入。RXR 寄存器在内部总线和接收移位寄存器间形成一个缓冲。RXR 寄存器是一个两字节深度的 FIFO 缓冲器，它能保存两帧数据的同时接收第三帧数据，注意的是，应用程序必须保证在接收完第三帧前读取 RXR 寄存器，否则将忽略第三帧数据并且发生过速错误。

    接收器的初始化可由如下步骤完成：

    --正确设置 BNO、PRT、PREN 和 STOPS 位以确定数据长度、校验类型和停止位长度。

    --设置 BRG 寄存器，选择期望的波特率。

    --设置 RXEN 为高，使 RX 引脚作为 UART 的接收端而非普通的输入输出端口。

    此时接收器被使能并等待起始位。

    数据接收时将会发生如下事件：

    --当 RXR 寄存器中有一帧以上的数据时，USR 寄存器中的 RXIF 位将会置位。

    --若 RIE=1，数据从 RSR 寄存器加载到 RXR 寄存器中将产生中断。

    --若接收过程中检测到帧错误、噪声干扰错误、奇偶出错或过速错误，相应的错误标志位置高。

    可以通过如下步骤来清除 RXIF：

    1. 读取 USR 寄存器
    2. 读取 RXR 寄存器

  - 接收暂停字

    UART 接收到任何暂停字都会当作帧出错处理。接收器只根据 BNO 和 STOPS 位确定一帧数据的长度。若暂停字位数比 13 大得多，当接收到 BNO 和 STOPS 位指定的长度时，接收器认为接收已完毕，RXIF 和 FERR 置位，RXR 寄存器清 0，若相应的中断允许且 RIDLE 为高将会产生中断。若暂停字较长，接收器收到起始位、数据位和有效停止位时，将会置位 FERR 标志，且在下一起始位前接收器将等待有效的停止位。接收器不会将暂停字认为是下一帧数据的起始位,暂停字仅包含 0 同时 FERR 标志被置高。暂停字被加载到缓冲器中，在接收到停止位前不会再接收数据。注意的是，没有检测到停止位也会置位只读标志位 RIDLE。

    UART 接收到暂停字会产生以下事件：

    --帧错误标志位 FERR 置位。

    --RXR 寄存器清零。

    --OERR、NF、PERR、RIDLE 或 RXIF 可能会置位。

  - 空闲状态

    当 UART 接收数据时，即在起始位和停止位之间，USR 寄存器中接收标志位 RIDLE 清零。在停止位和下一帧数据的起始位之间，RIDLE 被置位，表示接收器空闲。

- 接收中断

USR 寄存器的只读标志位 RXIF 由接收器的边沿触发置位。若 RIE=1，数据从移位寄存器 RSR 加载到 RXR 寄存器时产生中断，同样地，过速也会产生中断。

- **接收错误处理**

UART 会产生几种接收错误，下面部分将描述各种错误以及他们的处理。

- 过速出错——OERR 标志位

RXR 寄存器是一个两字节深度的 FIFO 缓冲器，它能保存两帧数据的同时接收第三帧数据，应用程序必须保证在接收完第三帧前读取 RXR 寄存器，否则发生过速错误。

产生过速错误时将会发生以下事件：

--USR 寄存器中 OERR 被置位。

--RXR 寄存器中数据不会丢失。

--RSR 寄存器数据会被覆盖。

--若 RIE=1，将会产生中断。

读取 USR 寄存器后再读 RXR 寄存器将 OERR 标志位清 0。

- 噪声干扰——NF 标志位

数据恢复时多次采样可以有效鉴别噪声干扰。当检测数据受到噪声干扰时将会发生以下事件：

--在 RXIF 上升沿，USR 寄存器中 NF 置位。

--数据从 RSR 寄存器加载到 RXR 寄存器。

--不产生中断，此位置位的同时由 RXIF 请求中断。

读取 USR 寄存器后再读取 RXR 寄存器将复位 NF 标志位。

- 帧错误——FERR 标志位

若在停止位上检测到 0，USR 寄存器中 FERR 置位。若选择 2 位停止位，2 位停止都必须为高，否则将置位 FERR。它同数据一起存储在缓冲器中，可被任何复位清除。

- 奇偶校验错误——PERR 标志位

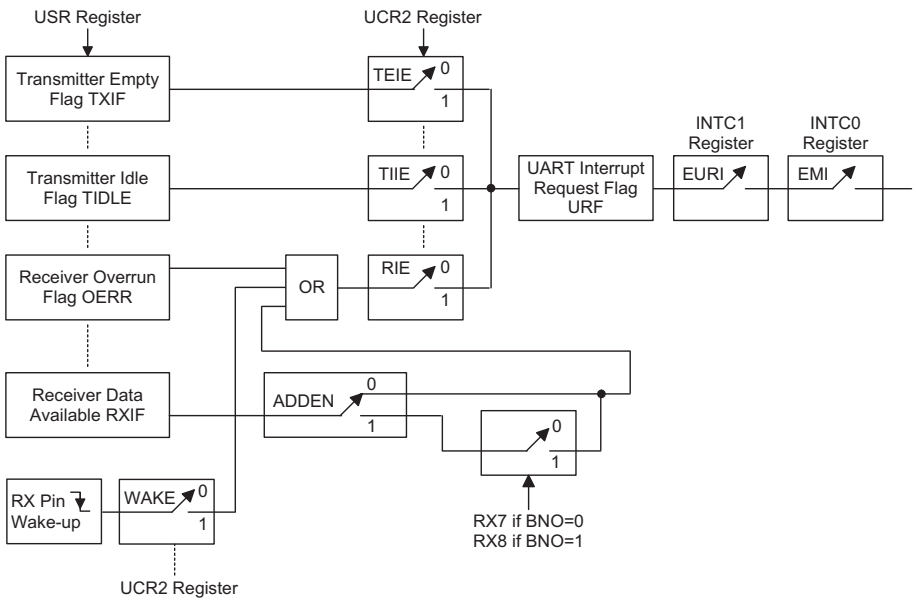
若接收到的数据出现奇偶校验错误，USR 寄存器中 PERR 置位。只有使能了奇偶校验，PREN=1，同时选择校验类型，此标志位才有效。它同数据一起存储在缓冲器中，可被任何复位清除。注意的是，FERR 和 PERR 与相应的数据一起存储在缓冲器中，在读取数据之前必须先访问错误标志位。

- **接收中断图解**

UART 拥有单独的内部中断和独立的中断向量。发送寄存器为空、发送器空闲、接收器数据有效、接收过速，地址匹配和 RX 引脚唤醒都会产生中断。若 UART 中断允许且堆栈未满，程序将会跳转到相应的中断向量执行中断服务程序，而后再返回主程序。上述四种中断条件中，若 UCR2 寄存器中相应中断允许位被置位，USR 寄存器中标志位将会产生中断。发送器有两个相应的中断允许位而接收器两个中断共用一个中断允许位。这些允许位可用于禁止个别的 UART 中断源。

地址匹配也是 UART 的一种中断源，它没有相应的标志位，若 UCR2 寄存器中 ADDEN=1，当检测到匹配地址时将会产生 UART 中断。RX 引脚唤醒也可以产生 UART 中断，它没有相应的标志位，当 UXR2 中的 WAKE 和 RIE 位被置位，RX 引脚上触发下降沿可以唤醒单片机。注意的是，RX 唤醒中断发生时，系统必须延时 1024 个系统时钟才能正常工作。

值得注意的是，USR 寄存器中标志位为只读，应用程序中无法清 0 或置 1，和其他中断一样，即使响应对应中断服务程序时也无法清 0。当 UART 响应某种动作时，标志位会自动清 0，详细请参考 UART 寄存器部分。INTC1 寄存器中 EURI 位用来打开/关闭 UART 中断，以禁止 UART 中断发生。



UART 中断框图

- **地址检测模式**

置位 UCR2 寄存器中的 ADDEN 将启动地址检测模式。若 ADDEN 有效，只有在接收到有效数据时，产生中断请求标志 RXIF。当 ADDEN 使能，仅在接收到有效数据最高位为 1 时，产生中断。中断允许位 EURI 和 EMI 也必须使能才会产生中断。地址的最高位为第 9 位（BNO=1）或第 8 位（BNO=0），若此位为高，则接收到的是地址而非数据。只有接收的数据的最高位为 1 才会产生中断。若 ADDEN 除能，每接收到一个有效数据便会置位 RXIF，而不用考虑数据的最高位。地址检测和奇偶校验在功能上相互排斥，因此如果使能地址检测模式，必须保证操作的正确，同时必须将奇偶检验使能位清零，以除能奇偶校验。

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

ADDEN 位功能

- **暂停模式下的 UART 功能**

当 MCU 进入暂停模式，UART 将停止工作。当芯片进入暂停模式，模块的所有时钟关闭。当 UART 传送数据时，MCU 进入暂停模式，发送将停止并且 TX 引脚保持高电平。同样地，当 MCU 接收数据时进入暂停模式，数据接收也会停止。当单片机进入暂停模式，USR、UCR1 和 UCR2、发送/接收寄存器、BRG 寄存器都不会受到影响。

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

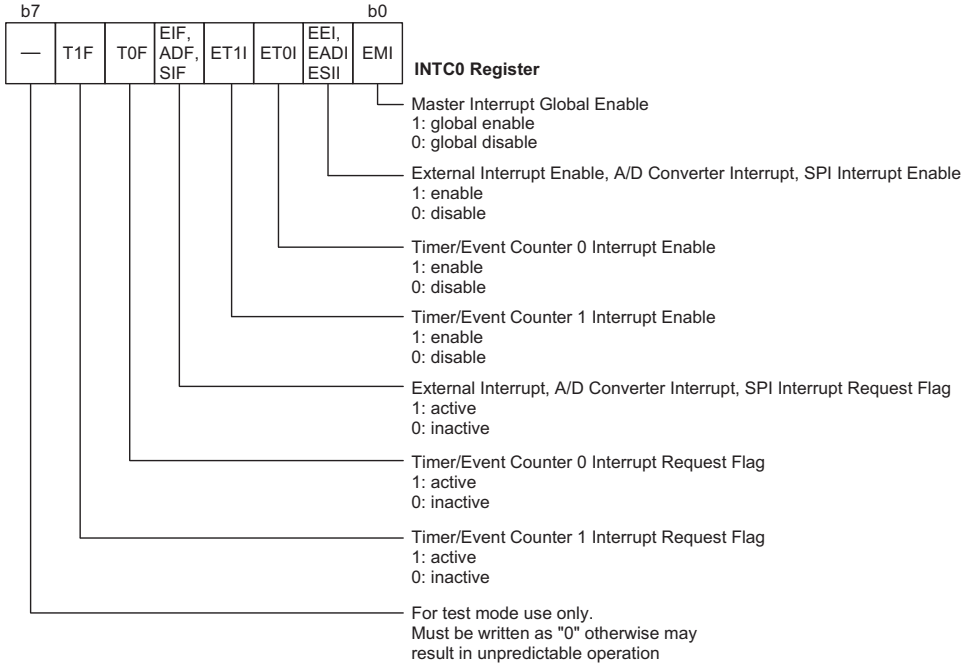
若要唤醒并产生 UART 中断，全局中断允许位 EMI 和 UART 中断允许 EURI 也必须置位；若这两标志位没有被置位，那么，单片机将可以被唤醒但不会产生中断。同样唤醒后系统需延时 1024 个系统时钟才能正常工作，然后才会产生 UART 中断。

中断

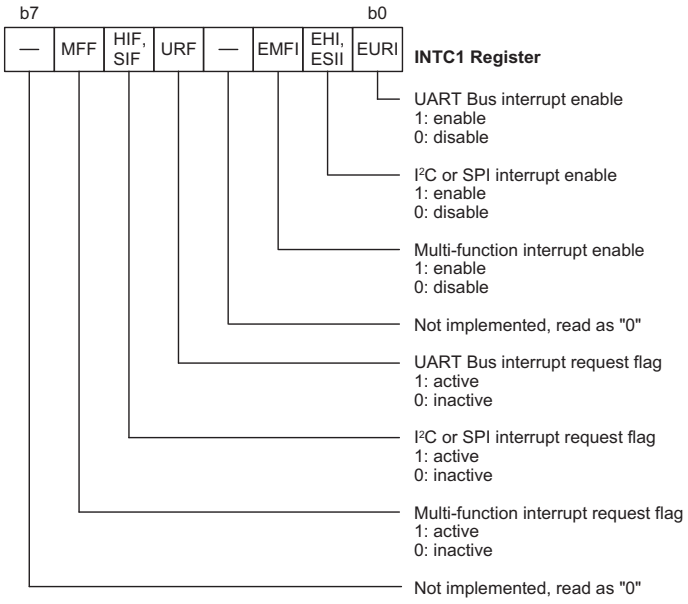
中断是单片机一个重要功能。当外部中断或内部中断如定时/计数器，UART，I^2C，SPI 总线，时基，实时时钟或 A/D 转换有效，系统会中止当前的程序，而转到相对应的中断服务程序。外部中断与 INT 引脚相关。

中断寄存器

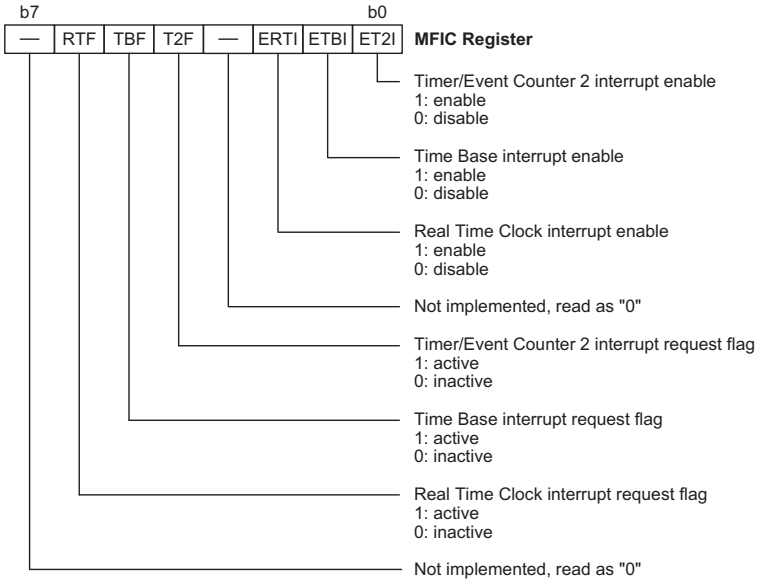
所有中断允许和请求标志均由数据存储器内 INTC0，INTC1 和 MFIC 寄存器控制。通过控制相应的中断允许位使能或禁止相关的中断。当发生中断，相应有中断请求标志将被置位。总中断请求标志清零将禁止所有中断允许。



中断控制寄存器 0



中断控制寄存器 1

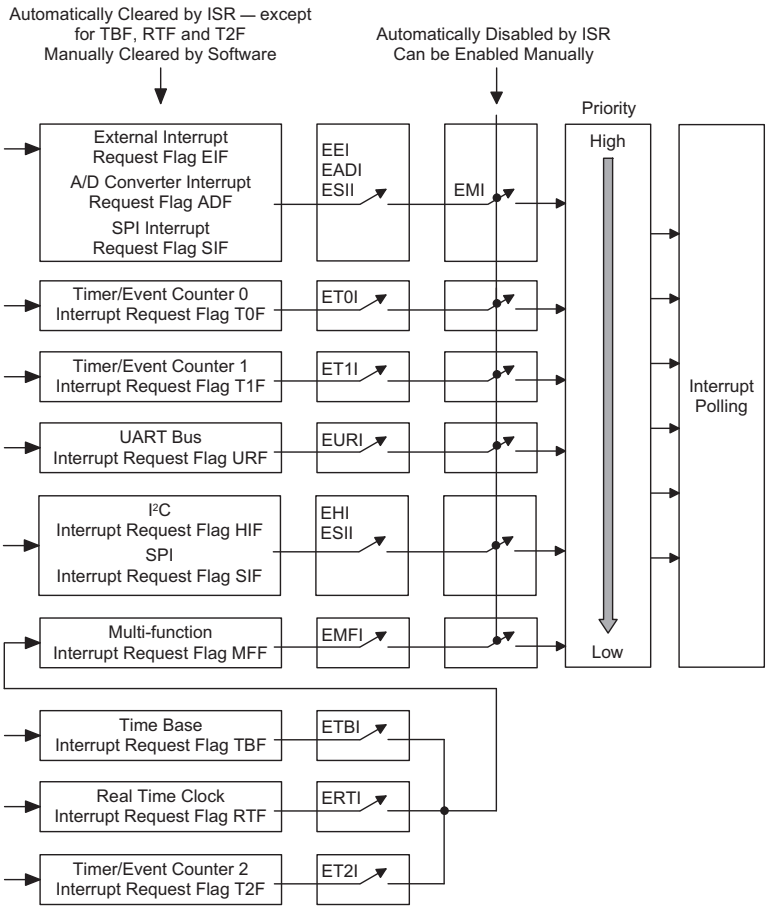


多功能中断控制寄存器

中断操作

如果相应的中断允许，定时/计数器溢出、A/D 转换结束，外部中断引脚拉低，UART，SPI，I^2C 总线或多功能中断都会产生一个中断请求，下条指令的地址将被压入堆栈，相应的中断向量地址加载至 PC 中，系统将从此向量取下条指令。中断向量处通常为跳转指令，以跳转到相应的中断服务程序。中断服务程序必须以 RETI 指令返回至主程序，以执行原来的程序。

各个中断使能位以及相应的请求标志位，以优先级的顺序如下图所示。



中断结构图

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

中断优先级

中断发生在两个连续的 T2 脉冲上升沿之间，如果相应的中断请求被允许，中断将在后一个 T2 脉冲响应。下表指出同时提出请求的情况下所提供的优先级，可以通过重新设定 EMI 位来加以屏蔽。下表中，对于 04H 和 14H 的向量地址，多个中断共享相同的地址，通过掩膜选项可以选择有效的中断。

中断源	优先级	中断向量
外部中断 A/D 转换中断 SPI 中断	1	04H
定时/计数器 0 溢出中断	2	08H
定时/计数器 1 溢出中断	3	0CH
UART 总线中断	4	10H
I^2C 总线中断 SPI 中断	5	14H
多功能中断： －定时/计数器 2 溢出 －实时时钟溢出 －时基溢出	6	18H

假使外部和内部中断均被使能，且外部和内部中断同时发生，则外部中断永远优先处理，首先被响应。使用 INTC0 和 INTC1 控制寄存器适当地屏蔽个别中断，可以防止中断同时发生的情况。

外部中断

使外部中断发生，总中断控制位 EMI、外部中断使能位 EEI 必须先被置位。当外部中断有效触发，INTC0 寄存器中 EIF 位将被设定。由于外部中断可以双边沿触发，当 INT 引脚出现上升沿触发或下降沿触发时，EIF 标志位将被设置为 1。外部中断和 PA5 共享引脚，使用外部中断时必须将 INTC 寄存器中外部中断使能位设置为 1。同时通过设置引脚的控制寄存器相应的位，即 PAC.5 来将 PA5 设置为输入。当中断使能、堆栈未满且外部中断产生时，将调用位于地址 04H 处的子程序。当响应外部中断服务子程序时，中断请求标志位 EIF 会被复位且 EMI 位会被清零以除能其它中断。注意到，即使此引脚被用作外部中断输入，其掩膜选项中的上拉电阻仍保持有效。由于此中断向量地址与其他中断共享，因此应该由掩膜选项选择后才能有效。

定时/计数器中断

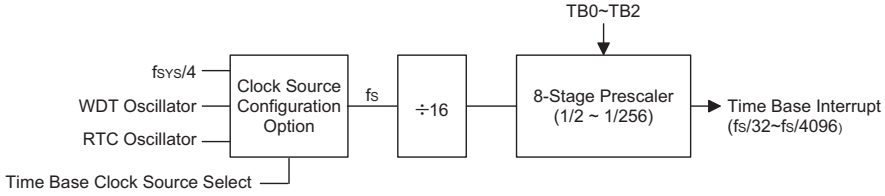
使定时/计数器中断发生，总中断控制位EMI和相应的内部中断使能位必须先被置位。定时/计数器0 的中断使能位ET0I为INTC0 寄存器的第2 位，定时/计数器1 的中断使能位ET1I 为INTC0 寄存器的第3 位，而定时/计数器2的中断使能位ET2I 为MFIC 寄存器的第0 位。当定时器溢出后，就会置位定时/计数器中断请求标志位，产生定时/计数器中断。对于具有一组定时器的单片机，请求标志位TF 为INTC 寄存器的第5 位。定时/计数器0 的请求标志位T0F 为INTC0 寄存器的第5 位，定时/计数器1 的请求标志位T1F 为INTC0 寄存器的第6 位，定时/计数器2 的请求标志位T2F 为MFIC寄存器的第4位。因为定时/计数器2的中断向量包含在多功能中断内，对于由定时/计数器2产生的中断，也必须通过设置INTC1寄存器中的EMFI位来使能多功能中断。然后，定时/计数器2的溢出也会造成INTC1 寄存器的第6位，多功能中断请求的标志位，即MFF被置位，并且依次产生中断。

当主设备的中断使能位被置位、堆栈未满且对应的内部中断使能位被置位时，一旦定时/计数器溢出，就会产生内部中断。由定时/计数器0 产生的中断，调用地址08H 处子程序，由定时/计数器1 产生的中断，调用地址0CH 处子程序，而由定时/计数器2 产生的中断，调用地址018H 处的子程序。需注意的是，定时/计数器2的中断向量包含在多功能中断内，它和其他中断共用向量。在进入定时器中断执行程序后，相应的中断请求标志位TF、T0F 会被复位且EMI 位会被清零以除能其它中断。对于定时器2，当其中断发生后，EMI 位会被清零以除能其它中断，但是只有MFF中断请求标志位会被复位。由于T2F标志位不会自动复位，它必须通过应用程序来清零。

时基中断

使时基中断发生，INTC0 寄存器中总中断控制位 EMI，和相应内部中断使能位 ETBI（MFIC 寄存器的第 1 位）必须先被置位。当时基中断发生时，INTC1 寄存器中 TBF（MFIC 寄存器的第 5 位）中断请求标志位将置位并触发时基中断。因为时基中断向量包含在多功能中断内，对于由时基产生的中断，也必须通过设置 INTC1 寄存器中的 EMFI 位来使能多功能中断。然后，时基的溢出也会造成 INTC1 寄存器的第 6 位，多功能中断请求的标志位，即 MFF 被置位，并且依次产生中断。

当主设备的中断使能位被置位、堆栈未满且对应的时基中断使能位被置位时，一旦时基产生溢出信号，就会产生内部时基中断。然后将调用地址018H 处的子程序。需注意的是，时基中断向量包含在多功能中断内，它和其他中断共用向量。当时基中断发生后，EMI 位会被清零以除能其它中断，但是只有MFF中断请求标志位会被复位。由于T2F标志位不会自动复位，它必须通过应用程序来清零。时基中断的目的是提供一个固定周期的中断信号。时基中断时钟源由内部时钟源f_S产生。f_S输入时钟首先经过分频器，分频率由掩膜选项选择来提供更长久的时基中断周期。时基中断溢出范围是$2^{12}/f_S$~$2^{15}/f_S$。产生f_S的时钟源，依次控制着时基中断周期，由三种不同的振荡源产生，RTC振荡器，看门狗振荡器或者系统振荡器的四分频，振荡源的选择由f_S时钟源的掩膜选项决定。注意的是，如果选择RTC振荡器为系统时钟，那么f_S和相应的时基中断，也会将RTC振荡器作为它的时钟源。



时基中断

实时时钟中断

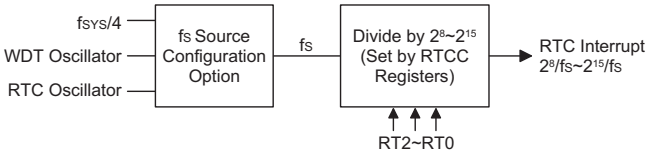
使实时中断中断发生，INTC0 寄存器中总中断控制位 EMI，和相应内部中断使能位 ERTI（MFIC 寄存器的第 2 位）必须先被置位。当时基中断发生时，INTC1 寄存器中 RTF（MFIC 寄存器的第 6 位）中断请求标志位将置位并触发时基中断。

当主设备的中断使能位被置位、堆栈未满且对应的实时时钟中断使能位被置位时，一旦溢出信号，就会产生内部时基中断，然后将调用地址 018H 处的子程序。因为实时时钟中断向量包含在多功能中断内，对于由时基产生的中断，也必须通过设置 INTC1 寄存器中的 EMFI 位来使能多功能中断。然后，实时时钟的溢出也会造成 INTC1 寄存器的第 6 位，多功能中断请求的标志位，即 MFF 被置位，并且依次产生中断。当时基中断发生后，EMI 位会被清零以除能其它中断，但是只有 MFF 中断请求标志位会被复位。由于 T2F 标志位不会自动复位，它必须通过应用程序来清零。重要的是，不要将 RTC 中断和 RTC 振荡器混淆。

和时基中断的中断相似，RTC 中断的目的是也提供一个固定周期的中断信号。RTC 中断时钟源由内部时钟源 f_S 产生。f_S 输入时钟首先经过分频器，分频率由设计 RTCC 寄存器相应的位来选择以获取更长久的 RTC 中断周期，其范围是 $2^8/f_S$~$2^{15}/f_S$。产生 f_S 的时钟源，依次控制着时基中断周期，由三种不同的振荡源产生，RTC 振荡器，看门狗振荡器或者系统振荡器的四分频，振荡源的选择由 f_S 时钟源的掩膜选项决定。注意的是，如果选择 RTC 振荡器为系统时钟，那么 f_S 和相应的时基中断，也会将 RTC 振荡器作为它的时钟源。

注意的是，RTC 中断周期由掩膜选项和一个内部共同控制。掩膜选项选择作为内部时钟 f_S 的时钟源，RTCC 寄存器的 RT2，RT1 和 RT0 位选择分频率。注意的是，实际的分频率能被编程在 2^8~2^{15}。实际的 RTC 中断周期的细节，请参考 RTCC 寄存器章节。

注意的是，在唤醒后，系统需要 1024 个时钟周期来占用正差的操作。如果 32768Hz 的 RTC 振荡器也被选择为系统时钟源，那么唤醒后灵敏的时序是作为 RTC 中断的应用，当选择 2^8，2^9 和 2^{10}RTC 中断分频时，需采取防范措施。对于这些分频率，在唤醒后，在 1024 个时钟循环周期内，一些在 RTC 中断后的事件会被错过。



RTC 中断

A/D 转换中断

使 A/D 转换中断发生，总中断控制位 EMI、和相应的中断使能位 EADI 必须先被置位。当 A/D 转换结束，INTC1 寄存器中 ADF 中断请求标志位将置位并触发 A/D 转换中断。若中断使能，堆栈未满，当发生 A/D 转换中断时，将调用位于地址 04H 处的子程序。当响应 AD 转换中断服务子程序时，中断请求标志位 ADF 会被复位且 EMI 位会被清零以禁止其它中断。由于此中断和其他中断共用向量地址，因为必须选择好掩膜选项才能使其有效。

UART 中断

要使UART中断发生，相应的内部中断使能位EURI 必须被置位，此位为INTC1 的第一位。当UART 产生暂停信号时，就会置位中断请求标志URF（即INTC1 的第4位），产生UART 中断。当主中断使能位被置位，堆栈未满且相应的UART 中断使能位EURI 被置位时，一旦UART 产生请求，就会产生UART 中断，将会调用地址010H 处的子程序。当UART 中断响应时，中断请求标志位URF 会被复位且EMI 位将被清零以除能其它中断。

产生UART 中断有多种情况，例如，发送数据寄存器为空（TXIF），收寄存器状态（RXIF），数据发送完成(TIDLE)，过速错误（OERR）和地址检测，这些状态反映在UART 的状态寄存器USR 中。UART 控制寄存器UCR2 决定是否产生中断信号。更多关于这两个寄存器的介绍可查阅UART 相关章节。

I^2C 总线中断

使 I^2C 中断发生，总中断控制位 EMI、和相应的中断使能位 EHI 必须先被置位，此位为 INTC1 的第一位。当 I^2C 中断请求标志位 HIF（INTC1 寄存器中第 5 位）被置位，就会产生实际的 I^2C 中断。当主设备的中断使能位被置位、堆栈未满且对应的 SPI 中断使能位被置位时，一旦接收到 I^2C 从机地址的匹配信号或者 I^2C 数据位发送的完成，就会产生内部时基中断。然后将调用地址 014H 处的子程序。当响应 I^2C 中断服务子程序时，中断请求标志位 HIF 会被复位且 EMI 位会被清零以禁止其它中断。由于此中断和其他中断共用向量地址，因为必须选择好掩膜选项才能使其有效。

SPI 中断

使 SPI 中断发生，总中断控制位 EMI、和相应的中断使能位 EHI 必须先被置位，此位为 INTC0/INTC1 的第一位。当 I^2C 中断请求标志位 SIF（INTC0/INTC1 寄存器中第 4/5 位）被置位，就会产生实际的 I^2C 中断。当主设备的中断使能位被置位、堆栈未满且对应的 SPI 中断使能位被置位时，就会产生一个内部中断。然后将调用地址 014H 或 04H 处的子程序。当响应 SPI 中断服务子程序时，中断请求标志位 SIF 会被复位且 EMI 位会被清零以禁止其它中断。由于此中断和其他中断共用向量地址，因为必须选择好掩膜选项才能使其有效。

多功能中断

这里提供了附加的中断，即多功能中断，与其他中断不同，它没有独立源，但由另外的现有的中断源构成，即时基中断，实时时钟中断和定时/计数器 2 中断。当 EMFI 位（INTC1 寄存器中第 2 位）被置位，就使能多功能中断。当多功能中断请求标志位 MFF（INTC1 寄存器中第 6 位）被置位，就会产生实际的多功能中断。当主设备的中断使能位被置位、堆栈未满且对应的 SPI 中断使能位被置位时，无论是时基溢出，实时时钟溢出或者定时/计数器 2 溢出发生，都会产生一个多功能内部中断。然后将调用地址 018H 处的子程序。当响应 SPI 中断服务子程序时，中断请求标志位 MFF 会被复位且 EMI 位会被清零以禁止其它中断。但必须注意的是，来自多功能中断的源头，即时基，实时时钟和定时/计数器 2 的请求标志位不会自动复位，并且必须由使用者手动复位。

编程注意事项

通过禁止中断使能位，可以屏蔽中断请求，然而，一旦中断请求标志位被设定，它们会被保留在 INTC0 或 INTC1 中断控制寄存器内，直到相应的中断服务子程序执行或被软件指令清除。

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

所有中断在暂停模式下都具有唤醒功能。

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

复位和初始化

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

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

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

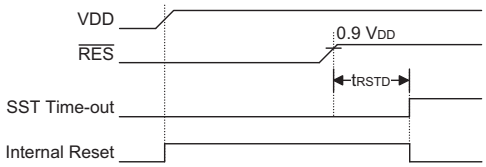
复位功能

通过内部与外部事件触发复位，单片机共有五种复位方式：

- 上电复位

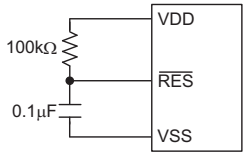
  这是最基本且不可避免的复位，发生在单片机上电后。除了保证程序存储器会从起始地址开始执行，上电复位也使得其它寄存器被设定在预设条件，所有的输入/输出端口寄存器和输入/输出端口控制寄存器在上电复位时会保持高电平，以确保上电后所有引脚被设为输入状态。

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

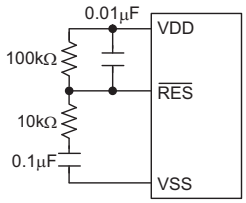


上电复位时序图

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



基本复位电路



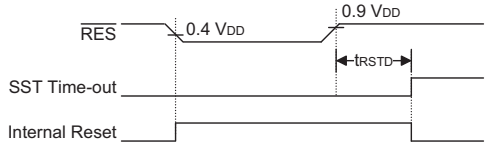
增强型复位电路

当系统在较强干扰的场合工作时，强烈建议使用增强型复位电路。

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

- **$\overline{\text{RES}}$**引脚复位

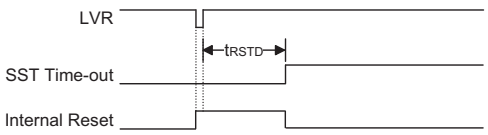
当单片机正常工作，$\overline{\text{RES}}$引脚通过外部硬件(如外部开关)强迫拉至低电平时，此种复位形式即会发生。这种复位形式与其它复位一样，程序计数器会被清除为零且程序从头开始执行。



$\overline{\text{RES}}$引脚复位时序图

- 低电压复位

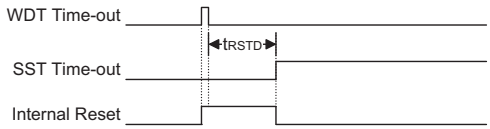
单片机具有低电压复位电路，用来监测它的电源电压，可通过掩膜选项进行选择。例如在更换电池的情况下，单片机供应的电压可能会落在 0.9V~V_{LVR} 的范围内，这时 LVR 将会自动复位单片机。LVR 包含以下的规格：有效的 LVR 信号，即在 0.9V~V_{LVR} 的低电压状态的时间，必须超过交流电气特性中 t_{LVR} 参数的值。如果低电压存在不超过 t_{LVR} 参数的值，则 LVR 将会忽略它且不会执行复位功能。V_{LVR} 参数值可通过掩膜选项进行设定。



低电压复位时序图

- 正常操作时看门狗溢出复位

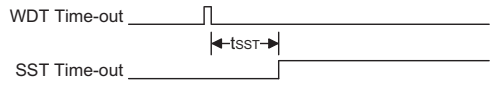
  除了看门狗溢出标志位 TO 将被设为 1 之外，正常操作时看门狗溢出复位和 $\overline{RES}$ 复位相同。



正常操作时看门狗溢出复位时序图

- 暂停时看门狗溢出复位

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



暂停时看门狗溢出复位时序图

复位初始状态

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

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

注：“u”代表不改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

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

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

寄存器	复位 (上电复位)	$\overline{\text{RES}}$ 或 LVR 复位	WDT 溢出 (正常运行)	WDT 溢出 (暂停模式)
MP0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	0000 0000	0000 0000	0000 0000	00u0 00uu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RTCC	---0 -111	---0 -111	---0 -111	--uu uuuu
STATUS	--00 xxxx	--uu uuuu	--lu uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0H	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	uuuu uuuu
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	00-0 1---	00-0 1---	00-0 1---	uu-u u---
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PWM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM2	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM3	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
INTC1	-000 -000	-000 -000	-000 -000	-uuu -uuu
TBHP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
HADR	xxxx xxx-	xxxx xxx-	xxxx xxx-	uuuu uuu-
HCR	0--0 0---	0--0 0---	0--0 0---	u--u u---
HSR	100- -0-1	100- -0-1	100- -0-1	uuuu uuuu
HDR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- --00	1--- --00	1--- --00	u--- --uu
PF	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PG	1111 1111	1111 1111	1111 1111	uuuu uuuu
PGC	1111 1111	1111 1111	1111 1111	uuuu uuuu
TMR2	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
MFIC	-000 -000	-000 -000	-000 -000	-uuu -uuu
USR	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
TXR/RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu xxxx
SBCR	0110 0000	0110 0000	0110 0000	uuuu uuuu

SBDR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

注： “u”表示未变化。
“x”表示未确定。
“–”表示未使用。

振荡器

不同的振荡器选择可以让使用者在不同的应用需求中获得更多范围的功能。有两种系统时钟可供选择，而看门狗定时器又有多种时钟源选项，提供了使用者最大的灵活性。

有两种方法产生系统时钟：

外部晶体/陶瓷振荡器

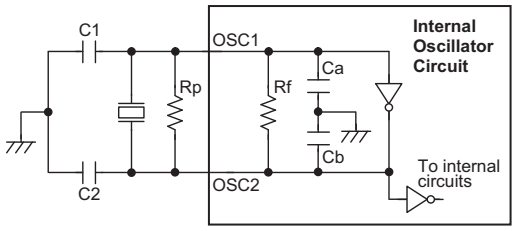
外部 RC 振荡器

外部晶体/陶瓷振荡器或 RC 系统振荡器都需通过掩膜选项进行设置。

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

外部晶体/陶瓷振荡器

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



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

外部晶体/陶瓷振荡器

内部 Ca，Cb，Rf 典型值@5V，25℃		
Ca	Cb	Rf
11pF~13pF	13pF~15pF	470kΩ

振荡器内部元件值

晶体振荡器 C1 和 C2 值			
晶体频率	C1	C2	CL
8MHz	TBD	TBD	TBD
4MHz	TBD	TBD	TBD
1MHz	TBD	TBD	TBD
注：1、C1，C2 值仅作为参考。 2、CL 为晶体规格测试时的负载电容。			

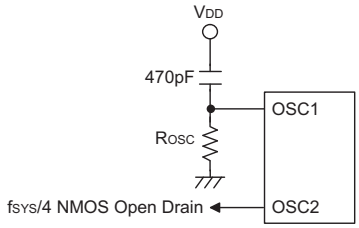
晶体振荡器电容推荐值

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

陶瓷振荡器电容推荐值

外部 RC 振荡器

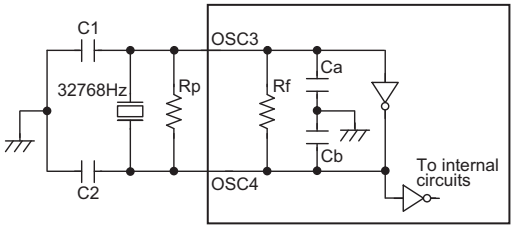
使用外部 RC 电路作为系统振荡器，需要在 OSC1 和 VDD 之间连接一个阻值约在 24KΩ到 1MΩ之间的电阻，OSC1 与地之间连接一个电容。产生的系统时钟 4 分频后提供给 OSC2 作输出，以达到与外部同步化的目的。由于 OSC2 为 NMOS 开漏输出，测量 RC 振荡频率时，则需要加上拉电阻。虽然此振荡器配置成本较低，但振荡频率会因 VDD、温度和芯片本身的制成而改变，因此不适合用来做计时严格或需要精确振荡器频率的场合。对于外部电阻 R_{OSC} 的阻值，请参考附录章节中典型 RC 振荡器对温度以及对 V_{DD} 特性曲线分析。注意：内部电容和外部电阻 R_{OSC} 共同作用决定频率值，图中显示的外部电容并不会影响振荡器的频率值。



外部 RC 振荡器

外部 RTC 振荡器

当系统进入暂停模式，系统时钟关闭以降低功耗。然而在某些应用，比如暂停模式下保持定时/计数功能，必须提供 32768Hz 振荡器的实时时钟。此时 OSC3 和 OSC4 引脚必须连接 32768Hz 晶体振荡器。为了保证低频率晶体振荡的应用，建议使用两个小容量电容 C1 和 C2，具体数值与客户选择的晶体规格有关。外部并联的反馈电阻，即 R_P 在某些场合并不需要，仅用来辅助系统启动。系统使用 32768Hz 晶体振荡器时的慢速模式可降低功耗。



Note: 1. Rp is normally not required.
2. Although not shown OSC3/OSC4 pins have a parasitic capacitance of around 7pF.

内部 RC 振荡+外部 RTC 振荡

内部 Ca，Cb，Rf 典型值@5V，25℃		
Ca	Cb	Rf
TBD	TBD	TBD

RTC 振荡器内部元件值

RTC 振荡器 C1 和 C2 值			
晶体频率	C1	C2	CL
32768Hz	TBD	TBD	TBD
注：1、C1，C2 值仅作为参考。 2、CL 为晶体规格测试时的负载电容。			

32768Hz 晶体振荡器电容推荐值

当系统进入暂停模式，32768Hz 振荡器继续运行，如果 32768Hz 振荡器作为时钟和看门狗时钟源，将保持计数功能。

系统上电后，RTC 振荡器启动会有一定的延时。可以通过设置 MODE 寄存器中 QOSC 位缩短延时来控制快速振荡。系统上电时 QOSC 位清 0 以快速启动 RTC 振荡器，为了降低快速启动的功耗，建议系统上电 2 秒钟后，应用程序设置 QOSC 位为 1。注意的是，无论 QOSC 位是否设置，RTC 振荡器将保持正常工作，仅影响快速启动时的系统功耗。

看门狗定时振荡器

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

暂停模式下的暂停和唤醒

暂停模式

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

进入暂停模式

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

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

静态电流

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

唤醒

当系统进入 HALT 模式下，可以通过以下几种方式唤醒：

- 外部复位
- PA 口下降沿
- 系统中断
- WDT 溢出

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

端口 PA 中的每个位都可以通过掩膜选项独立选择唤醒功能。PA 端口唤醒后，程序将在“HALT”指令后继续执行。

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

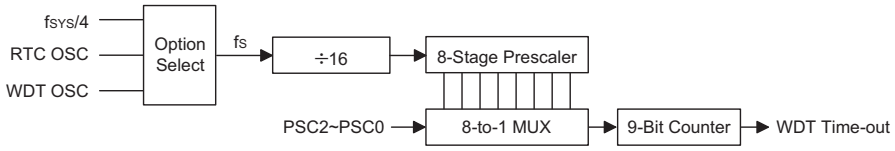
一旦唤醒事件发生，单片机回到正常运行将需要 1024 个系统时钟周期。如果唤醒由中断响应，则实际的中断子程序执行将延迟一个或数个周期；如果唤醒后接着去执行下一条指令，则它在 1024 个系统周期结束后立刻执行。

看门狗定时器

看门狗定时器的功能在于防止如电磁的干扰等外部不可控制事件，所造成的程序不正常动作或跳转到未知的地址。当 WDT 溢出时，它产生一个芯片复位的动作。WDT 时钟源可以通过掩膜选项进行设置，选择三个时钟源中一个提供，内建 WDT 振荡器，$f_{SYS}/4$ 或 RTC 振荡器。注意的是，假如 WDT 掩膜选项设为除能，则任何相关的指令将无效。

在此单片机中，所有看门狗定时器的选项，如使能/除能、WDT 时钟源和清除看门狗指令条数都是通过掩膜选项来选择，没有与 WDT 相关的内部寄存器。WDT 的时钟源之一是内部振荡器，在供应电压为 5V 时周期近似为 65μs。必须注意的是，这个内部时钟的周期可以随着 VDD、温度和制作工艺而改变。看门狗溢出周期为 $2^{14}/f_S \sim 2^{21}/f_S$。

如果选择 $f_{SYS}/4$ 为看门狗时钟源，必须注意的是，当系统进入暂停模式后，指令时钟会停止且 WDT 将失去其保护目的。在这种情况下，系统只能通过外部逻辑重新复位。当系统操作在干扰严重的环境时，建议使用内部 WDT 振荡器。



看门狗定时器

系统在正常运行状态下，WDT 溢出将导致芯片复位，并置位状态标志位 TO。然而如果系统处于暂停模式，则只有一个从暂停模式来的 WDT 溢出复位发生，它只复位程序计数器和 SP。有三种方法可以用来清除 WDT 的内容，第一种是外部硬件复位($\overline{\text{RES}}$引脚低电平)，第二种是通过软件指令，而第三种是通过“HALT”指令。

使用软件指令有两种方法去清除看门狗寄存器，必须由掩膜选项选择。第一种选择是使用单一“CLR WDT”指令，而第二种是使用“CLR WDT1”和“CLR WDT2”两个指令。对于第一种选择，只要执行“CLR WDT”便清除 WDT。而第二种选择，必须交替执行“CLR WDT1”和“CLR WDT2”两者才能成功的清除 WDT。关于第二种选择要注意的是，如果“CLR WDT1”正被使用来清除 WDT，接着再执行这条指令将是无效的，只有执行“CLR WDT2”指令才能清除 WDT。同样的“CLR WDT2”指令已经执行后，只有接着执行“CLR WDT1”指令才可以清除看门狗定时器。

时基 — Time Base

时基指的是用一个周期性的溢出来产生一个有规律的内部中断。溢出周期的范围为 $2^{12}/fs \sim 2^{15}/fs$，由掩膜选项确定。当时基发生溢出，其中断请求标志(TBF；MFIC 的第 5 位、MFF；INTC1 的第 6 位)会被置位，如果中断允许，且堆栈未满，那么就会产生一个地址 018H 的子程序调用。TBF 不会自动清除，需要程序员通过软件清除。

实时时钟 — RTC

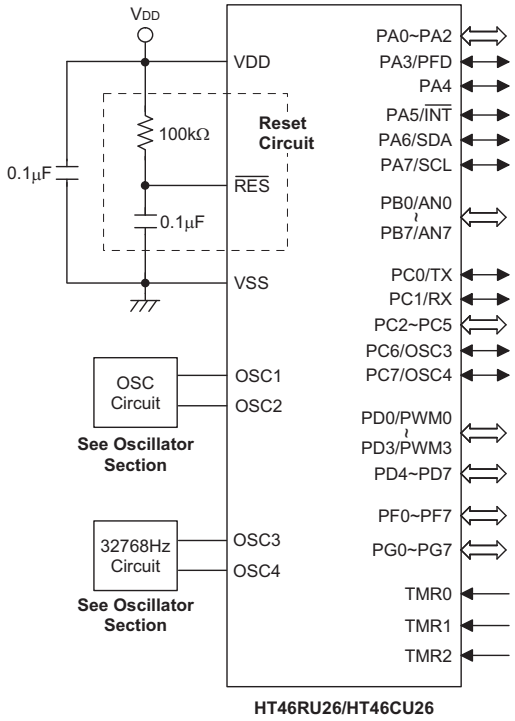
实时时钟(RTC)的工作情况和时基一样。它是用来提供一个有规律的内部中断。它的溢出周期范围为 $fs/2^8 \sim fs/2^{15}$，可通过软件编程实现。写数据到 RT2、RT1、RT0（RTCC 的第 2、1、0 位；09H）将产生各种溢出周期。当 RTC 发生溢出，其中断请求标志(RTF；MFIC 的第 6 位、MFF；INTC1 的第 6 位)被置位，如果中断允许，且堆栈未满，那么就会产生一个地址 018H 的子程序调用。RTF 不会自动清除，需要程序员通过软件清除。

掩膜选项

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

编号	选项
输入/输出选项	
1	PA0~PA7：唤醒 使能/禁止（位选择）
2	PA0~PA7：选择上拉电阻 使能/禁止
3	PB0~PB7：选择上拉电阻 使能/禁止
4	PC0~PC7：选择上拉电阻 使能/禁止
5	PD0~PD7：选择上拉电阻 使能/禁止
6	PF0~PF4 ：选择上拉电阻 使能/禁止
7	PG0~PG7：选择上拉电阻 使能/禁止
振荡器选项	
8	震荡类型选择：晶体或 RC
9	f_{SYS} 时钟源：OSC 或 RTC 振荡器
10	f_S 内部时钟源：RTC 振荡器，WDT 振荡器或 $f_{SYS}/4$
时基选项	
11	时基溢出周期： $2^{12}/f_S$， $2^{13}/f_S$， $2^{14}/f_S$，$2^{15}/f_S$
PFD 选项	
12	PA3：正常输入/输出或 PFD 输出
13	PFD 时钟选择：定时/计数器 0 或者定时/计数器 1
PWM 选项	
14	PD0~PD3：PWM0~PWM3 功能选择
15	PWM 模式：6+2 或 7+1 模式
看门狗选项	
16	看门狗功能：使能/禁止
17	清除看门狗指令：1 条或 2 条
18	看门狗溢出周期选择。 看门狗溢出周期： $2^{12}/f_S$~$2^{13}/f_S$ ， $2^{13}/f_S$~$2^{14}/f_S$ ， $2^{14}/f_S$~$2^{15}/f_S$ ， $2^{15}/f_S$~$2^{16}/f_S$
中断选项	
19	中断向量选择 04H： $\overline{INT}$，14H：I^2C 04H： $\overline{INT}$，14H：SPI 04H：A/D，14H：I^2C 04H：A/D，14H：SPI 04H：SPI， 14H：I^2C
I^2C 总线选项	
20	I^2C 总线功能：使能/禁止
SPI 选项	
21	SPI 功能：使能/禁止
22	SPI WCOL 功能：使能/禁止
23	SPI CSEN 功能：使能/禁止
24	SPI CPOL 功能：使能/禁止
LVR 选项	
25	LVR 功能：使能/禁止

应用电路


VDD
VDD
100kΩ
0.1μF
RES
0.1μF
VSS
Reset Circuit
OSC Circuit
OSC1
OSC2
See Oscillator Section
32768Hz Circuit
OSC3
OSC4
See Oscillator Section
PA0~PA2
PA3/PFD
PA4
PA5/INT
PA6/SDA
PA7/SCL
PB0/AN0
PB7/AN7
PC0/TX
PC1/RX
PC2~PC5
PC6/OSC3
PC7/OSC4
PD0/PWM0
PD3/PWM3
PD4~PD7
PF0~PF7
PG0~PG7
TMR0
TMR1
TMR2
HT46RU26/HT46CU26

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

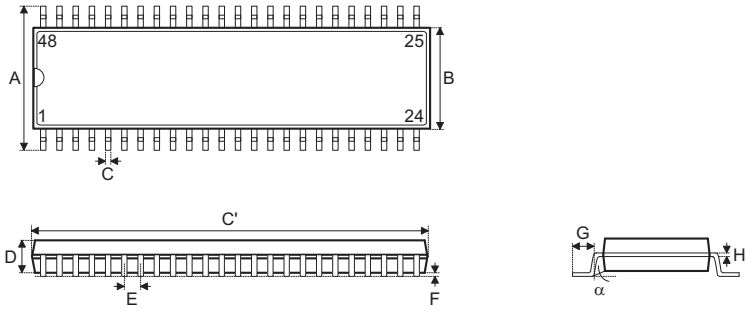
XOR A, [m] Logical XOR accumulator with data memory
说明： 将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。
运算过程： ACC←ACC “XOR” [m]
影响标志位： Z

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

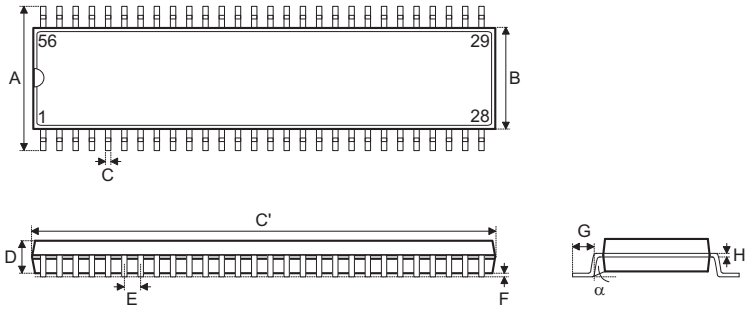
封装尺寸

48-pin SSOP (300mil)外形尺寸



标号	尺寸（单位：mil）		
	最小	典型	最大
A	395	--	420
B	291	--	299
C	8	--	12
C’	613	--	637
D	85	--	99
E	--	25	--
F	4	--	10
G	25	--	35
H	4	--	12
α	0°	--	8°

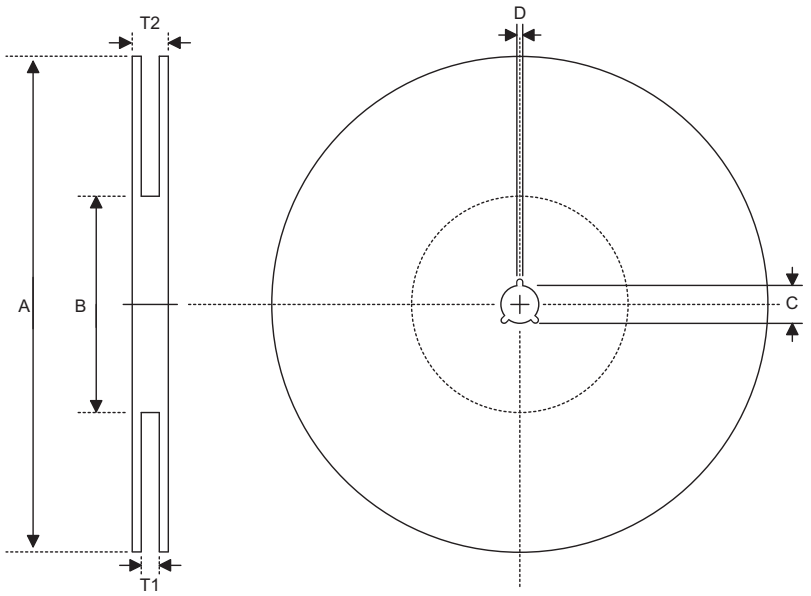
56-pin-SOP (300mil)外形尺寸



标号	尺寸（单位：mil）		
	最小	典型	最大
A	395	--	420
B	291	--	299
C	8	--	12
C’	720	--	730
D	89	--	99
E	--	25	--
F	4	--	10
G	25	--	35
H	4	--	12
α	0°	--	8°

包装带和卷轴规格

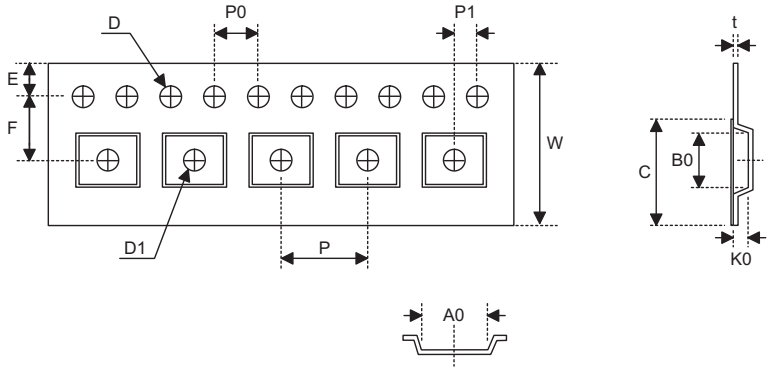
卷轴尺寸



SSOP 48W

符号	说明	尺寸 (mm)
A	卷轴外圈直径	330±1
B	卷轴内圈直径	100±0.1
C	轴心直径	13+0.5 -0.2
D	缝宽	2±0.5
T1	轮缘宽	32.2+0.3 -0.2
T2	卷轴宽	38.2±0.2

运输带尺寸



SSOP 48W

符号	说明	尺寸 (mm)
W	运输带宽	32±0.3
P	空穴间距	16±0.1
E	穿孔位置	1.75±0.1
F	空穴至穿孔距离(宽度)	14.2±0.1
D	穿孔直径	2Min.
D1	空穴中之小孔直径	1.5+0.25
P0	穿孔间距	4±0.1
P1	空穴至穿孔距离(长度)	2±0.1
A0	空穴长	12±0.1
B0	空穴宽	16.2±0.1
K1	空穴深	2.4±0.1
K2	空穴深	3.2±0.1
t	传输带厚度	0.35±0.05
C	覆盖带宽度	25.5