GREDA 格瑞达

8-Bit 触摸按键式 Flash 单片机

BS45F1B

版本：V01　　日期：2013/06/14

www.greenmcu.com

目录

第1章　概述及其特性

1.1　特性

1.1.1　CPU 特性

- 工作电压：
  f_{SYS} = 12MHz：2.7V~5.5V
- 集成 16 触控按键功能 -- 不需要增加外接元件
- 暂停和唤醒功能，以降低功耗
- 振荡器类型
  外部高速晶体振荡器 – HXT
  内部 12MHz RC – HIRC
  内部 32kHz RC -- LIRC
- 多种工作模式：正常模式，低速模式，空闲模式和休眠模式
- 所有指令都可在 1 个或 2 个指令周期内完成
- 查表指令
- 63 条功能强大的指令系统
- 多达 8 层硬件堆栈
- 位操作指令

1.1.2　周边特性

- Flash 程序存储器：4K×16
- 数据存储器：384×8
- 看门狗定时器功能
- 多达 26 个双向 I/O 口
- 两个与 I/O 口复用的外部中断输入
- 2 个 8 位可编程定时/计数器
- 1 个时基功能，用于产生固定时间的中断信号
- 低电压复位功能
- 封装类型：16NSOP, 20/24/28 SOP

1.2 概述

该单片机是一款 8 位具有高性能精简指令集且完全集成触控按键功能的 Flash 单片机。此单片机含有触控按键功能和可多次编程的 Flash 存储器特性，为各种触控按键的应用提供了一种简单而又有效的实现方法。

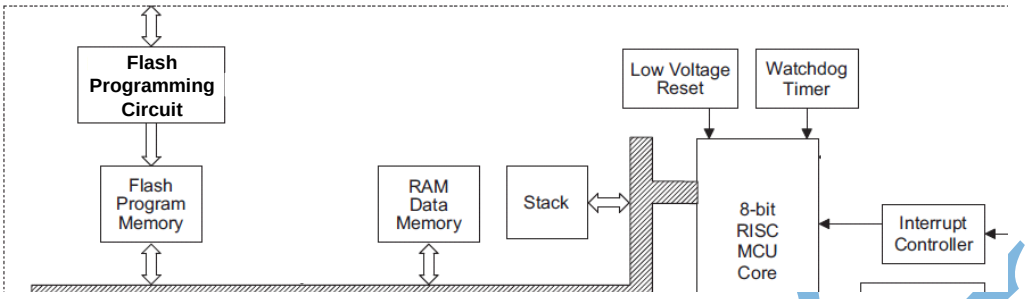
触控按键功能完全集成于单片机内，使用较少的外部元件便可实现触控按键的应用。该单片机除了 Flash 程序存储器，还包括 RAM 数据存储器。内部看门狗定时器和低电压保护功能具有良好的抗噪声和抗 ESD 保护功能，确保单片机在恶劣的电气环境中仍能保持稳定的操作。

该单片机内部集成了高/低速振荡器，在应用中不需增加外部元件。动态切换高低系统时钟的能力，为用户提供了优化单片机操作和降低功耗的能力。8-bit 定时器和其它特性增强了该单片机的功能和灵活性。

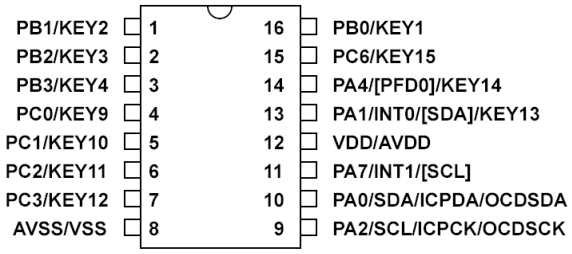
该触控按键单片机能广泛应用于各种触控按键产品中，例如仪器仪表，家用电器，电子控制工具等等。

正印	高速振荡器
BS45F1B-1	HIRC
BS45F1B-2	HXT

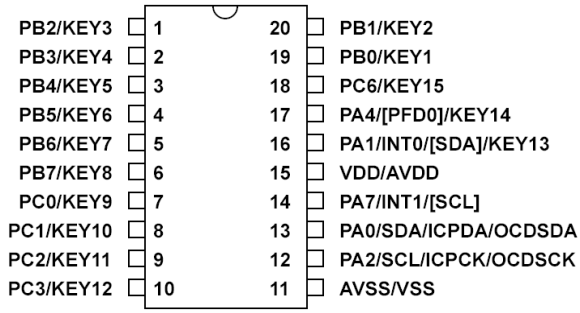
1.3 方框图



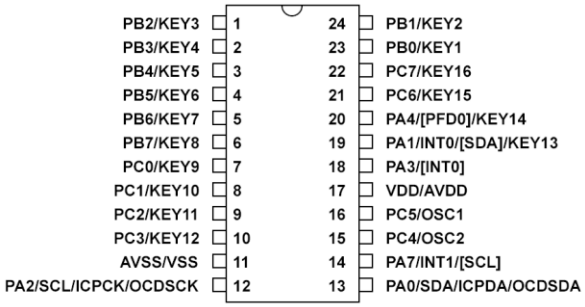
1.4 引脚图



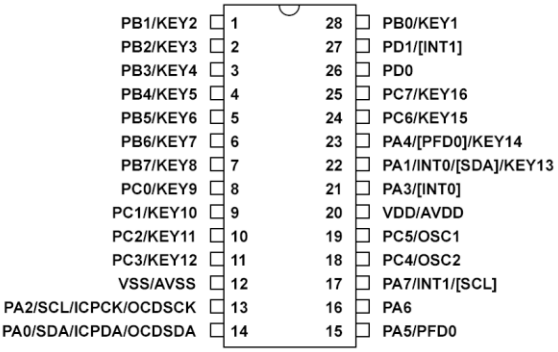
BS45F1BNS16



BS45F1BS20



BS45F1BS24



BS45F1BS28

1.5 引脚说明

下表中列出了每个引脚的功能，而每个引脚功能的细节将在文中其它章节有详细的描述。

引脚名称	功能	OP	I/T	O/T	说明
PA0/SDA/ICPDA/OCDSDA	PA0	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SDA	PRM	ST	NMOS	I^2C 数据
	ICPDA	—	ST	CMOS	在线烧录数据/地址引脚
	OCDSDA	—	ST	CMOS	片上调试数据/地址引脚
PA1/INT0/[SDA]/KEY13	PA1	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	INT0	INTEG	ST	—	外部中断 0
	SDA	PRM	ST	NMOS	I^2C 数据
	KEY13	TKM3C1	NSI	—	触控按键输入口
PA2/SCL/ICPCK/OCDSCK	PA2	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SCL	PRM	ST	NMOS	I^2C 时钟
	ICPCK	—	ST	—	在线烧录时钟引脚
	OCDSCK	—	ST	—	片上调试时钟引脚
PA3/[INT0]	PA3	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	INT0	INTEG	ST	—	外部中断 0
PA4/[PFD0]/KEY14	PA4	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	PFD0	TMR0C	—	CMOS	PFD 输出
	KEY14	TKM3C1	NSI	—	触控按键输入口
PA5/PFD0	PA5	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	PFD0	TMR0C	—	CMOS	PFD 输出
PA6	PA6	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
PA7/INT1/[SCL]	PA7	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	INT1	INTEG	ST	—	外部中断 1
	SCL	PRM	ST	NMOS	I^2C 时钟
PB0/KEY1~ PB3/KEY4	PB0~PB3	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	KEY1~ KEY4	TKM0C1	NSI	—	触控按键输入口
PB4/KEY5~ PB7/KEY8	PB4~PB7	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	KEY5~ KEY8	TKM1C1	NSI	—	触控按键输入口
PC0/KEY9~ PC3/KEY12	PC0~PC3	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	KEY9~ KEY12	TKM2C1	NSI	—	触控按键输入口

引脚名称	功能	OP	I/T	O/T	说明
PC4/OSC2	PC4	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OSC2	—	—	HXT	振荡器引脚
PC5/OSC1	PC5	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OSC1	—	HXT	—	振荡器引脚
PC6/KEY15	PC6	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	KEY15	TKM3C1	NSI	—	触控按键输入口
PC7/KEY16	PC7	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	KEY16	TKM3C1	NSI	—	触控按键输入口
PD0	PD0	PDPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
PD1/[INT1]	PD1	PDPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	INT1	INTEG	ST	—	外部中断 1
VDD	VDD	—	PWR	—	电源电压
VSS	VSS	—	PWR	—	地
AVDD	AVDD	—	PWR	—	触控按键电路电源
AVSS	AVSS	—	PWR	—	触控按键电路地

注： I/T：输入类型
O/T：输出类型
OP：通过设置寄存器来选择功能
PWR：电源
ST：斯密特触发输入
CMOS：CMOS 输出
NMOS：NMOS 输出
NSI：无标准输入

1.6 极限参数

电源供应电压 ..... Vss-0.3V ~ Vss +6.0V
端口输入电压 ..... Vss-0.3V ~ V_{DD}+0.3V
I_{OL} 总电流 100mA
总功耗 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。

1.7 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 (HIRC/HXT)	—	f_{SYS} = 12MHz	2.7	—	5.5	V
I_{DD1}	工作电流 (HIRC/HXT) ($f_{SYS}=f_H$, $f_S=f_{SUB}=f_{LIRC}$)	5V	无负载, f_H = 12MHz WDT 使能, LVR 使能	—	3.3	5.0	mA
I_{DD2}	工作电流 (LIRC) ($f_{SYS}=f_L=f_{LIRC}$, $f_S=f_{SUB}=f_{LIRC}$)	5V	无负载, WDT 使能, LVR 使能	—	70	150	μA
I_{IDLE0}	IDLE 0 模式静态电流	5V	无负载, LVR 除能	—	3.0	5.0	μA
I_{IDLE1}	IDLE 1 模式静态电流	5V	无负载, f_{SYS} = 12MHz on, LVR 除能	—	1.4	2.1	mA
I_{SLEEP}	SLEEP 模式静态电流	5V	无负载, LVR 除能	—	3.0	5.0	μA
V_{IL}	输入/输出口或输入引脚低电平输入电压	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	V
V_{IH}	输入/输出口或输入引脚高电平输入电压	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	V
V_{LVR}	低电压复位电压	—	LVR 使能, V_{LVR}=2.55V	-5%	2.55	+5%	V
I_{OL}	输入/输出口输出低电压	5V	$V_{OL}=0.1V_{DD}$	10	20	—	mA
I_{OH}	输入/输出口输出高电压	5V	$V_{OH}=0.9V_{DD}$	-5	-10	—	mA
R_{PH}	上拉电阻(输入/输出口)	5V	—	10	30	50	kΩ

1.8 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_{CPU}	工作时钟	—	2.7V~5.5V	DC	—	12	MHz
f_{HIRC}	系统时钟(HIRC)	5V	—	-2%	12	+2%	MHz
		5V	Ta = 0~70℃	-4%	12	+3%	MHz
		5V	Ta = -40~85℃	-8%	12	+3%	MHz
		3.0~5.5V	Ta = 0~70℃	-5%	12	+11%	MHz
		3.0~5.5V	Ta = -40~85℃	-8%	12	+11%	MHz
f_{LIRC}	系统时钟(LIRC)	5V	—	-10%	32	+10%	kHz
		2.2V～5.5V	Ta = -40~85℃	-50%	32	+60%	kHz
f_{TIMER}	定时器输入频率	—	—	—	—	1	f_{SYS}
t_{INT}	中断脉冲宽度	—	—	1	—	—	μs
t_{LVR}	低电压复位宽度	—	—	60	120	240	μs
t_{SST}	系统启动延时周期 (从 HALT 模式下唤醒)	—	f_{SYS}=HIRC	—	15~16	20	t_{SYS}
			f_{SYS}=LIRC	—	1~2	4	

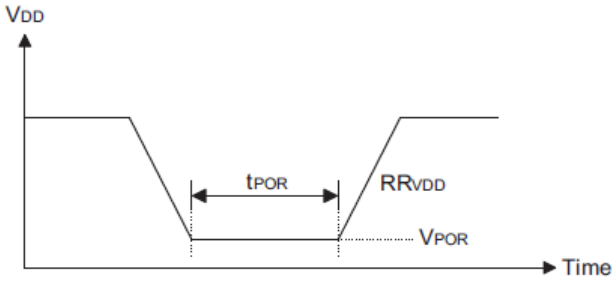
注： 1. $t_{SYS} = 1/f_{SYS}$

2.为了保持内部 HIRC 振荡器频率的准确性，需要在 VDD 和 VSS 之间接入一个 0.1μF 的去耦电容，并且尽可能地靠近单片机。

1.9 上电复位特性

Ta=25℃

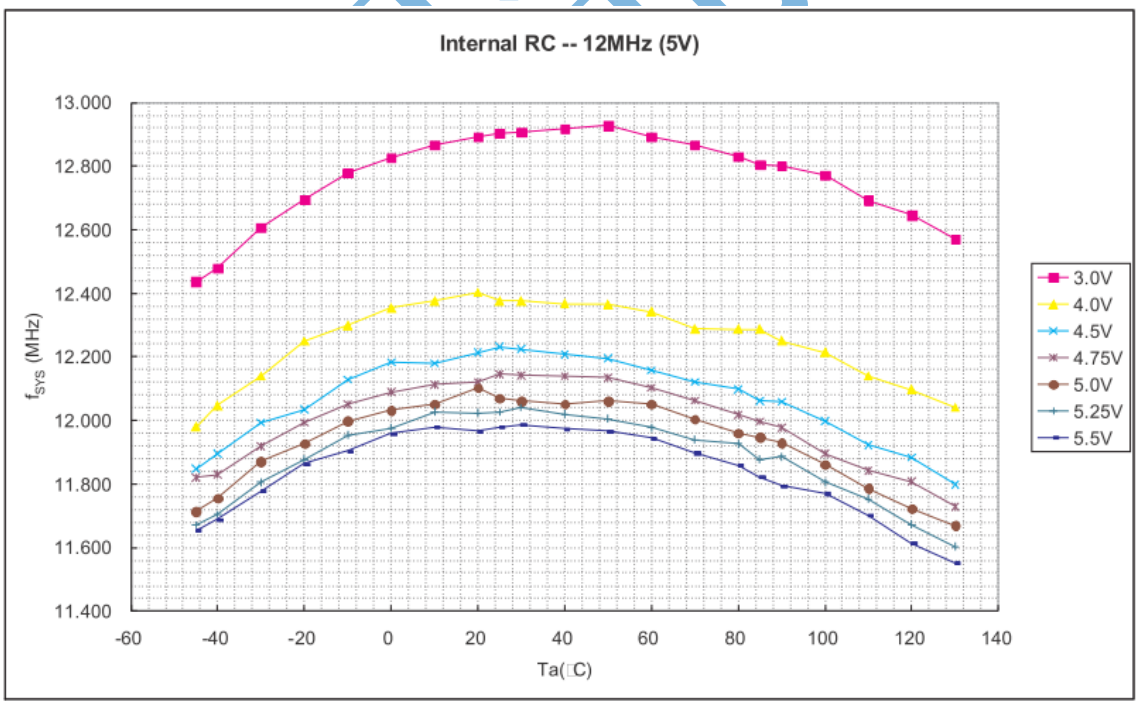
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{POR}	上电复位电压	—	—	—	—	100	mV
RR_{VDD}	上电复位电压速率	—	—	0.035	—	—	V/ms
t_{POR}	V_{DD} 保持为 V_{POR} 的最小时间	—	—	1	—	—	ms



1.10 振荡器温度/频率特性

下面的特性曲线图描述典型的振荡器行为。这里提出的数据是对一段时间内收集到的不同批次单位数据的统计汇总，仅供参考，在制造过程中并未进行测试。

在一些图表中，超出规定工作范围的数据仅供显示使用。该单片机只能在指定的范围内正常运行。



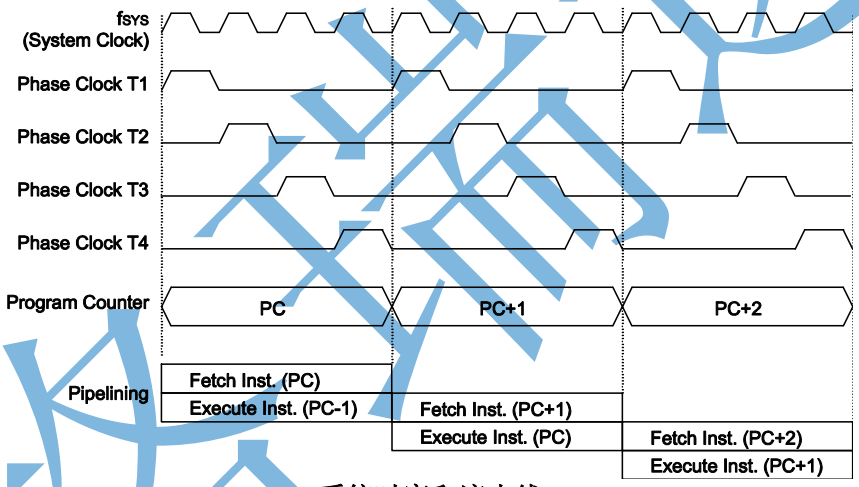
第2章 单片机系统

2.1 系统结构

内部系统结构是格瑞达单片机具有良好性能的主要因素。由于采用 RISC 结构，该单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的取得和执行同时进行，此举使得除了跳转和调用指令外，其它指令都能在一个指令周期内完成。8 位 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、递增、递减和分支等功能，而内部的数据路径则是以通过累加器或 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，确保了在提供具有最大可靠度和灵活性的实用性 I/O 控制系统时，仅需要少数的外部器件。这些特性使得该单片机适用于低成本，大批量生产的控制应用。

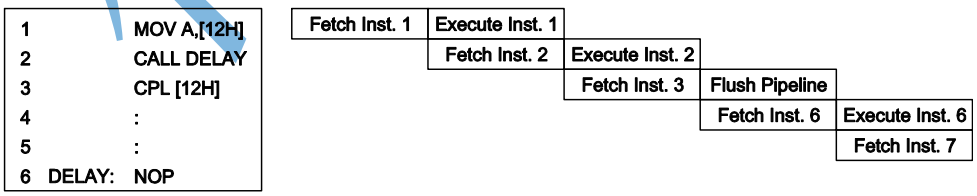
2.1.1 时序和流水线结构

主系统时钟由晶体/陶瓷振荡器，或 RC 振荡器提供，它被细分为 T1~T4 四个内部产生的非重叠时序。在 T1 时间，程序计数器自动加一并抓取一条新的指令。剩下的时间 T2~T4 完成译码和执行功能，因此，一个 T1~T4 时间周期构成一个指令周期。虽然指令的抓取和执行发生在连续的指令周期，但单片机流水线结构会保证指令在一个指令周期内被有效执行。除非程序计数器的内容被改变，如子程序的调用或跳转，在这种情况下指令将需要多一个指令周期的时间去执行。



系统时序和流水线

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，因此用户需要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。



指令捕捉

2.1.2 程序计数器

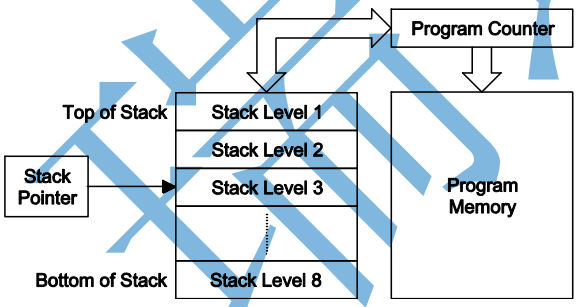
在程序执行期间，程序计数器用来指向下一个要执行的指令地址。除了“JMP”和“CALL”指令需要跳转到一个非连续的程序存储器地址之外，它会在每条指令执行完成以后自动加一。只有较低的 8 位，即所谓的程序低字节寄存器 PCL，可以被用户直接读写。

当执行的指令要求跳转到不连续的地址时，如跳转指令、子程序调用、中断或复位等，单片机通过加载所需要的位址到程序寄存器来控制程序，对于条件跳转指令，一旦条件符合，在当前指令执行时取得的下一条指令将会被舍弃，而由一个空指令周期来取代。

程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

2.1.3 堆栈

堆栈是一个特殊的存储器空间，用来保存程序计数器中的值。该单片机含有 8 层堆栈。堆栈寄存器既不是数据存储器的一部分，也不是程序存储器的一部分，而且它既不能读出，也不能写入。堆栈的使用是通过堆栈指针 SP 来指示的，堆栈指针也不能读出或写入。当发生子程序调用或中断响应时，程序计数器中的内容会被压入堆栈；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，则只有中断请求标志位会被置位，而中断响应会被禁止，直到堆栈指针发生递减(执行 RET 或 RETI 指令)，中断才会被响应。这个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 指令仍然可以执行，从而造成堆栈溢出。使用时应避免堆栈溢出的情况发生，因为这样会造成不可预期的程序分支指令的执行错误。

如果堆栈溢出，第一个保存在堆栈中的 PC 会丢失。

2.1.4 算术逻辑单元 — ALU

算术逻辑单元是单片机中很重要的部分，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：RRA、RR、RRCA、RRC、RLA、RL、RLCA、RLC
- 递增和递减：INCA、INC、DECA、DEC
- 分支判断：JMP、SZ、SZA、SNZ、SIZ、SDZ、SIZA、SDZA、CALL、RET、RETI

2.2Flash 程序存储器

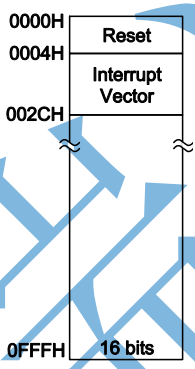
程序存储器用来存放用户代码即存储程序。该单片机提供可多次编程的存储器(Flash)，用户可以很方便的在同一个芯片中修改他们的应用代码。通过使用合适的编程工具，该 Flash 型单片机提供用户以灵活的方式自由开发他们的应用，这对于需要除错或需要经常升级和改变程序的产品是很有帮助的。

2.2.1 结构

程序存储器的容量为 4K×16。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。

2.2.2 特殊向量

程序存储器中某些地址保留用作诸如复位和中断的入口等特殊用途。000H 是保留用做单片机复位后的程序起始地址。在芯片初始化后，程序将会跳转到这个地址并开始执行。



Flash 程序存储器结构 2.2.3 查表

程序存储器中的任何地址都可以定义为一个表格，以便存储固定的数据。使用查表指令时，查表指针需要先行设定，其方式是将表格的地址放在表格指针寄存器 TBLP 和 TBHP 中。这两个寄存器定义的是表格总的地址。

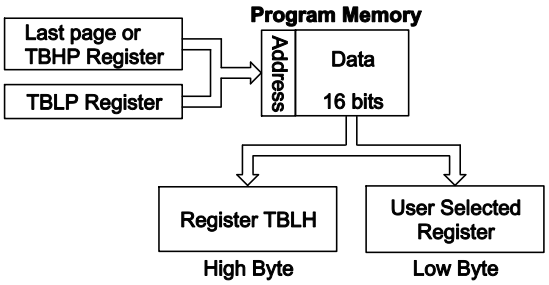
指令	表格地址											
	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRD [m]	@11	@10	@9	@8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

表格地址

注：*11~*0：表格地址位　　@7~@0：表格指针 (TBLP)位
@11~@8：表格指针(TBHP)位

在设定完表格指针后，表格数据可以使用“TABRD[m]”或“TABRDL[m]”指令从程序存储器中查表来读取。当这些指令执行时，程序存储器的表格的低字节，将会传输到用户所指定的数据存储器[m]中。程序存储器表格的高字节，将会传输到特殊寄存器 TBLH 中。传输数据中任何未定义的字节将会读取为“0”。

下图为查表寻址/数据流程图：



2.2.3 查表范例

以下范例说明在芯片中表格指针和表格数据如何被定义和执行。这个实例使用的表格数据用 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“F00H”，即 4K 程序存储器最后一页存储器的起始地址，而表格指针的初始值则为“06H”，这可保证从数据表格读取的第一笔数据位于程序存储器地址“F06H”，即最后一页起始地址后的第 6 个地址。值得注意的是，假如“TABRD [m]”指令被使用，则表格指针指向 TBHP+TBLP 位址。在这个例子中，表格数据的高字节等于零，而当“TABRD [m]”指令被执行时，此值将会自动的被传送到 TBLH 寄存器。

因为 TBLH 寄存器是只读寄存器，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，所有与表格相关的指令，都需要两个指令周期去完成操作。

2.2.4 表格读取程序范例

```
tempreg1 db ?                   ; temporary register #1
tempreg2 db ?                   ; temporary register #2
        :
        :
mov a,06h                       ; initialise table pointer - note that this address
mov tblp, a                     ; is referenced
mov a,0Fh                       ;initialise high table pointer
mov tbhp,a
        :
        :
tabrd    tempreg1               ; transfers value in table referenced by table pointer data at
                                ; program memory address “F06H” transferred to tempreg1 and TBLH
dec   tblp                      ; reduce value of table pointer by one
tabrd    tempreg2               ; transfers value in table referenced by table pointer data at
                                ; program memory address “F05H” transferred to tempreg2 and TBLH
                                ; in this example the data “1AH” is transferred to
                                ; tempreg1 and data “0FH” to register tempreg2
        :
        :
org F00h                        ; sets initial address of program memory

dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
        :
        :
```

2.2.5 在线烧录

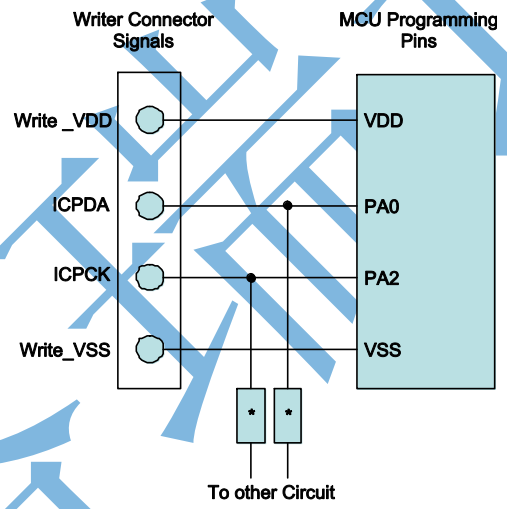
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。

另外，BS45F1B 单片机提供 4 线接口的在线烧录方式。用户可将进行过烧录或未经过烧录的单片机芯片连同电路板一起制成，最后阶段进行程序的更新和程序的烧写，在无需去除或重新插入芯片的情况下方便地保持程序为最新版。

烧录器	BS45F1B	引脚描述
引脚名称	引脚名称	
ICPDA	PA0	串行地址和数据——读/写
ICPCK	PA2	地址和时钟串行时钟输入脚
VDD	VDD	电源
VSS	VSS	地

芯片内部程序存储器可以通过 4 线的接口在线进行烧录。其中 PA0 用于数据串行下载或上传、PA2 用于串行时钟、两条用于提供电源。芯片在线烧写的详细使用说明超出此文档的描述范围，将由专门的参考文献提供。

在烧录过程中，烧录器会控制 PA0 和 PA2 脚进行数据和时钟烧录，用户必须确保这两个引脚没有连接至其它输出脚。



注：*可能为电阻或电容。若为电阻则其值必须大于 1kΩ，若为电容则其必须小于 1nF。

2.2.6 片上调试

EV IC 用于单片机仿真。此 EV IC 提供片上调试功能（OCDS—On-chip Debug Support）用于开发过程中的单片机调试。除了片上调试功能方面，EV IC 和实际 MCU 在功能上几乎是兼容的。用户可将 OCDSDA 和 OCDSCK 引脚连接至 HT-IDE 开发工具，从而实现 EV IC 对实际 IC 的仿真。OCDSDA 引脚为 OCDS 数据/地址输入/输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户用 EV IC 进行调试时，实际单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能无效。关于 OCDS 功能的详细描述，请参考“e-Link for 8-bit MCU OCDS User’s Guide”文件。

e-Link 引脚名称	EV IC 引脚名称	功能
OCDSDA	OCDSDA	片上调试串行数据/地址输入/输出
OCDSCK	OCDSCK	片上调试时钟输入
VDD	VDD	电源
VSS	VSS	地

2.3 数据存储器

数据存储器是内容可以更改的 8 位 RAM 内部存储器，用来存储临时数据。

2.3.1 结构

数据存储器分为两个部分，第一部分是特殊功能寄存器，这些寄存器有特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，而有些是被加以保护而不对用户开放。

第二部分是通用数据存储器，所有地址都可在程序的控制下进行读取和写入。

该单片机总的数据存储器被分为两个区。特殊功能数据寄存器均可在所有 Bank 被访问。切换不同区域可通过设置区域指针(BP)实现。所有单片机的数据存储器的起始地址都是“00H”。

	Bank0,1		Bank0,1
00H	IAR0	20H	PB
01H	MP0	21H	PBC
02H	IAR1	22H	PBPU
03H	MP1	23H	IICC0
04H	BP	24H	IICC1
05H	ACC	25H	IICD
06H	PCL	26H	IICA
07H	TBLP	27H	I2CTOC
08H	TBLH	28H	Unused
09H	TBHP	29H	Unused
0AH	STATUS	2AH	Unused
0BH	SMOD	2BH	Unused
0CH	Unused	2CH	Unused
0DH	INTEG	2DH	Unused
0EH	INTC0	2EH	Unused
0FH	INTC1	2FH	Unused
10H	INTC2	30H	Unused
11H	Unused	31H	Unused
12H	Unused	32H	Unused
13H	Unused	33H	Unused
14H	PA	34H	Unused
15H	PAC	35H	Unused
16H	PAPU	36H	Unused
17H	PAWU	37H	Unused
18H	CTRL	38H	PC
19H	LVRC	39H	PCC
1AH	WDTC	3AH	PCPU
1BH	TBC	3BH	PD
1CH	TMR0	3CH	PDC
1DH	TMR0C	3DH	PDPU
1EH	TMR1	3EH	PRM
1FH	TSCC	3FH	Unused

特殊功能数据存储器

2.4 特殊功能寄存器

大部分特殊功能寄存器的细节将在相关功能中描述，但有几个寄存器在此章节单独描述。

2.4.1 间接寻址寄存器 — IAR0，IAR1

间接寻址寄存器 IAR0 和 IAR1，位于数据存储区，并没有实际的物理地址。间接寻址方式是使用间接寻址寄存器和存储器指针对数据操作，以取代定义在实际存储器地址的直接存储器寻址方式。在间接寻址寄存器上的任何动作，将对间接寻址指针(MP0 或 MP1)所指定的存储器地址产生对应的读/写操作。IAR0 和 MP0，IAR1 和 MP1 对数据存储器中数据的操作是成对出现的，MP0 和 IAR0 用于访问 Bank 0，而 MP1 和 IAR1 可访问所有的 Bank。间接寻址寄存器不是实际存在的，直接读取 IAR 寄存器将会返回 00H 的结果，而直接写入此寄存器则不做任何操作。

2.4.2 间接寻址指针 — MP0, MP1

该单片机提供两个间接寻址指针，即 MP0 和 MP1。由于这些指针在数据存储器中能像普通的寄存器一样被写入和操作，因此提供了一个有效的间接寻址和数据追踪的方法。当对间接寻址寄存器进行任何操作时，单片机所指向的实际地址是由间接寻址指针所指定的地址。MP0 和 IAR0 用于访问 Bank 0，而 MP1 和 IAR1 可访问所有的 Bank。注意，对于该单片机，间接寻址指针 MP0 和 MP1 为 8 位寄存器，常与 IAR0、IAR1 搭配一起对数据存储器寻址。

以下范例说明如何清除一个具有 4 个 RAM 地址的区块，它们已经事先被定义成地址 adres1 到 adres4。

2.4.3 间接寻址程序范例

```
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:
```

在以上的例子中，没有提及具体的数据存储器地址。

2.4.4 存储区指针 — BP

该单片机数据存储器被分为两个部分，即 Bank 0 和 Bank 1。可以通过设置存储区指针（Bank Pointer）值来访问不同的数据存储区。位 0 用来选择数据存储器 Bank 0~1。

复位后，数据存储器会初始化到 Bank 0，但是在暂停模式下的 WDT 溢出复位，不会改变通用数据存储器的存储区号。应该注意的是特殊功能数据存储器不受存储区的影响，也就是说，不论是在哪一个存储区，都能对特殊功能寄存器进行读写操作。数据存储器的直接寻址总是访问 Bank 0，不影响存储区指针的值。要访问除 Bank 0 外的其它 Bank，则必须要使用间接寻址方式。

BP 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	DMBP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1　未定义，读为“0”

Bit 0　**DMBP0**：数据存储区选择

0：Bank 0

1：Bank 1

2.4.5 累加器 — ACC

对于任何单片机来说，累加器是相当重要的，且与 ALU 所完成的运算有密切关系，所有的 ALU 得到的运算结果都将暂存在累加器中，如果没有累加器，ALU 必须在每次进行如加法、减法和移位等运算时，将结果写入数据存储器中，这样会造成程序编写和时间的负担。另外，数据传输通常涉及到累加器的临时储存功能，如在用户定义的寄存器和另一个寄存器之间，由于两者之间的不能直接传送数据，因此需要通过累加器来传送数据。

2.4.6 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，程序计数器的低字节被设置在数据存储器的特殊功能区域，程序员可对此寄存器进行操作，很容易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专用程序存储器某一地址，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中跳转。注意，使用这种运算时，会插入一个空指令周期。

2.4.7 表格寄存器 — TBLP，TBHP，TBLH

这三个特殊功能寄存器对存储在程序存储器中的表格进行操作。TBLP 和 TBHP 为表格指针，指向表格的地址。它的值必须在任何表格读取指令执行前加以设定。由于它的值可以被如 INC 或 DEC 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用户指定的地址。

2.4.8 状态寄存器 — STATUS

这 8 位寄存器包括零标志位(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)。这些标志位同时记录单片机的状态数据和算术/逻辑运算。

除了 TO 和 PDF 标志位以外，状态寄存器的其它位像其它大多数寄存器一样可以被改变。但是任何数据写入状态寄存器将不会改变 TO 和 PDF 标志位。另外，执行不同指令操作后，与状态寄存器相关的运算将会得到不同的结果。TO 标志位只会受系统上电、看门狗溢出、或执行“CLR WDT”或“HALT”指令的影响。PDF 指令只会受执行“HALT”或“CLR WDT”指令或系统上电的影响。

Z、OV、AC 和 C 标志位通常反映最近的运算操作的状态。

- **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将不会自动压入到堆栈中保存。假如状态寄存器的内容很重要，且中断子程序会改变状态寄存器的内容，则需要保存备份以备恢复。

注意，状态寄存器的 0~3 位可以读取和写入。

STATUS寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	TO	PDF	OV	Z	AC	C
R/W	—	—	R	R	R/W	R/W	R/W	R/W
POR	—	—	0	0	x	x	x	x

“x”表示未知

Bit 7~6 未定义，读为“0”

Bit 5 **TO**：看门狗溢出标志位
0：系统上电或执行“CLR WDT”或“HALT”指令
1：WDT 溢出

Bit 4 **PDF**：暂停标志位
0：系统上电或执行“CLR WDT”指令
1：执行“HALT”指令将会置位 PDF 位。

Bit 3 **OV**：溢出标志位
0：不发生溢出时
1：当运算结果高两位的进位状态异或结果为 1 时

Bit 2 **Z**：零标志位
0：算数运算或逻辑运算的结果不为零时
1：算数运算或逻辑运算的结果为零时

Bit 1 **AC**：辅助进位标志位
0：没有辅助进位时
1：当低字节的加法造成进位或高字节的减法没有造成借位时

Bit 0 **C**：进位标志位
0：没有进位时
1：当加法造成进位或减法没有造成借位时，同时移位指令也会影响 C 标志位
C 也受循环移位指令的影响。

2.5 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实现更大范围的功能。振荡器的灵活性使得在速度和功耗之间可以达到最优化。振荡器选项是通过配置选项和寄存器共同完成的。

2.5.1 振荡器概述

所有振荡器都可以作为系统时钟源，低速振荡器还可以作为看门狗定时器，时基功能和定时/计数器的时钟源。外部振荡器需要一些外围器件，而集成的两个内部振荡器不需要任何外接器件。所有振荡器选项通过寄存器设置。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能/功耗比，此特性对功耗敏感的应用领域尤为重要。

振荡类型	名称	频率范围
外部晶振	HXT	400kHz~12MHz
内部高速 RC	HIRC	12MHz
内部低速 RC	LIRC	32kHz

振荡器类型

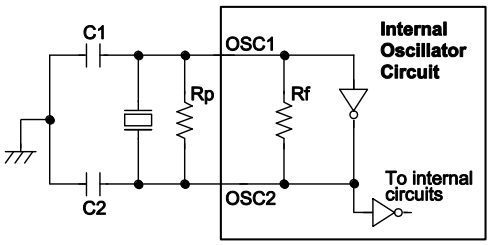
2.5.2 系统时钟配置

该单片机有三个方式产生系统时钟，两个高速时钟源和一个低速内部时钟源。高速振荡器有外部晶体/陶瓷振荡器–HXT，内部 12MHz RC 振荡器，低速振荡器为内部 32 kHz RC 振荡器。HIRC 和 LIRC 振荡器都是内部全集成的振荡器，无需外接器件。选择高速或低速振荡器作为系统振荡器，是通过 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位进行选择。

2.5.3 外部晶体/陶瓷振荡器－HXT

外部高频晶体/陶瓷振荡器可通过配置选项选择。对于晶体振荡器，只要简单地将晶体连接至 OSC1 和 OSC2，则会产生振荡所需的相移及反馈，而不需其它外部的器件。为保证某些低频率的晶

体振荡和陶瓷谐振器的振荡频率更精准，建议连接两个小容量电容 C1 和 C2 到 VSS，具体数值与客户选择的晶体/陶瓷晶振有关。



Note: 1. Rp is normally not required. C1 and C2 are required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

晶体/陶瓷振荡器-HXT

晶体振荡器 C1 和 C2 值		
晶体频率	**C1**	**C2**
12MHz	0pF	0pF
8MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	100pF	100pF
注：C1 和 C2 数值仅作参考用		

晶体振荡器电容推荐值

2.5.4 内部高速 RC 振荡器 — HIRC

内部高速 RC 振荡器是一个全集成的系统振荡器，不需其它外部器件。内部 RC 振荡器上电时的默认频率为 12MHz。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因 V_{DD}、温度以及芯片制成工艺不同的影响减至最低程度。

2.5.5 内部低速 RC 振荡器 — LIRC

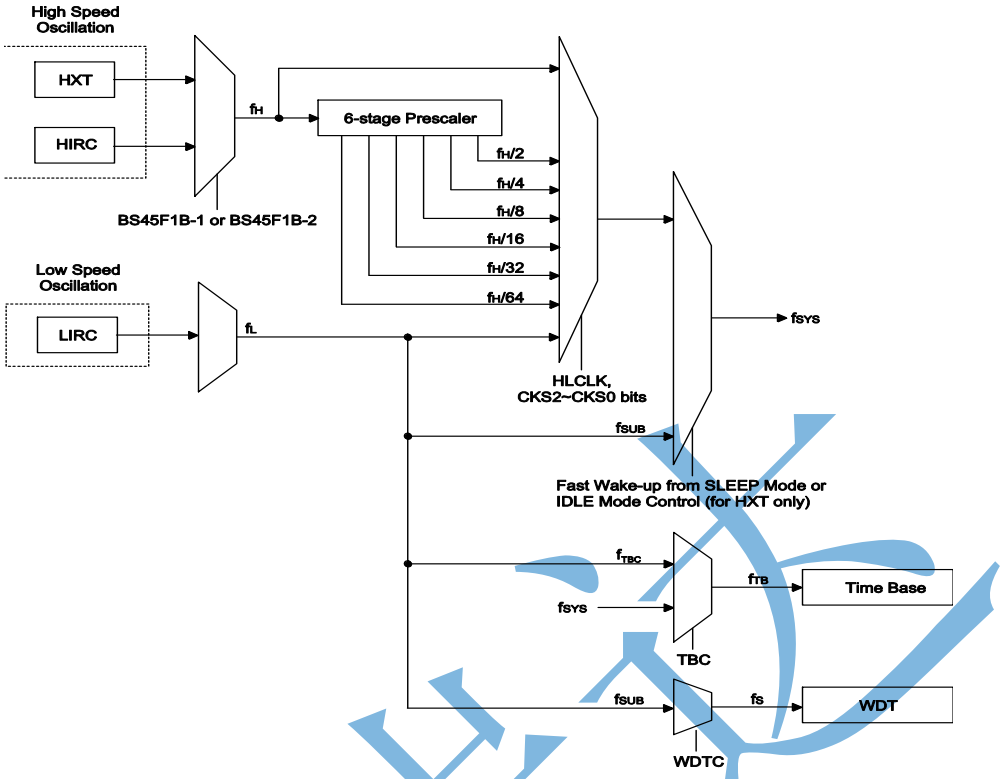
内部 32kHz 系统振荡器为低速振荡器。LIRC 是一个全集成的 RC 振荡器，无需外接器件，在常温 5V 条件下，振荡频率值为 32kHz。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因 V_{DD}、温度以及芯片制成工艺不同的影响减至最低程度。系统上电，LIRC 振荡器就使能，不存在将该振荡器除能的寄存器位。

2.6 工作模式和系统时钟

现今的应用要求单片机具有较高的性能及尽可能低的功耗，这种矛盾的要求在便携式电池供电的应用领域尤为明显。高性能所需要的高速时钟将增加功耗，反之亦然。此单片机提供高、低速两种时钟源，它们之间可以动态切换，用户可通过优化单片机操作来获得最佳性能/功耗比。

2.6.1 系统时钟

主系统时钟可来自高频时钟源 f_H 或低频时钟源 f_L，通过 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位进行选择。高频时钟来自 HXT 或 HIRC 振荡器，可通过选择不同正印的 MCU 决定，选择 BS45F1B-1 代表高频时钟来自 HIRC 振荡器，选择 BS45F1B-2 代表高频时钟来自 HXT 振荡器，低频时钟来自 LIRC 振荡器。其它系统时钟还有高速系统振荡器的分频 $f_H/2$~$f_H/64$。



系统时钟配置

2.6.2 控制寄存器

寄存器 SMOD 用于控制单片机内部时钟。

SMOD 寄存器

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	FSTEN	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
POR	0	0	0	0	0	0	1	1

Bit 7~5 **CKS2~CKS0**：当 HLCLK 为“0”时系统时钟选择位

000：f_L (f_{LIRC})
001：f_L (f_{LIRC})
010：$f_H/64$
011：$f_H/32$
100：$f_H/16$
101：$f_H/8$
110：$f_H/4$
111：$f_H/2$

这三位用于选择系统时钟源。除了 LIRC 振荡器提供的系统时钟源外，也可使用高频振荡器的分频作为系统时钟。

Bit 4 **FSTEN**：快速唤醒控制位（仅用于 HXT）

0：除能
1：使能

此位为快速唤醒控制位，用于决定单片机被唤醒后 f_{SUB} 是否开始工作。

Bit 3 **LTO**：低速振荡器就绪标志位

0：未就绪
1：就绪

此位为低速系统振荡器就绪标志位，用于表明低速系统振荡器在系统上电复位后何时稳定下来。

Bit 2 **HTO**：高速振荡器就绪标志位

0：未就绪
1：就绪

此位为高速系统振荡器就绪标志位，用于表明高速系统振荡器何时稳定下来。此标志在系统上电后经硬件

清零，高速系统振荡器稳定后变为高电平。因此，此位在单片机上电后由应用程序读取的总为“1”。该标志由休眠模式或空闲模式 0 中唤醒后会处于低电平状态，15~16 个时钟周期后改标志会处于高电平状态。

Bit 1 **IDLEN**：空闲模式控制位
0：除能
1：使能
此位为空闲模式控制位，用于决定 HALT 指令执行后发生的动作。若此位为高，当指令 HALT 执行后，单片机进入空闲模式。若 FSYSON 位为高，在空闲模式 1 中 CPU 停止运行，系统时钟将继续工作以保持外围功能继续工作；若 FSYSON 为低，在空闲模式 0 中 CPU 和系统时钟都将停止运行。若此位为低，单片机将在 HALT 指令执行后进入休眠模式。

Bit 0 **HLCLK**：系统时钟选择位
0：$f_H/2$~ $f_H/64$ 或 f_L
1：f_H
此位用于选择 f_H 或 $f_H/2$~$f_H/64$ 还是 f_L 作为系统时钟。该位为高时选择 f_H 作为系统时钟，为低时则选择 $f_H/2$~$f_H/64$ 或 f_L 作为系统时钟。当系统时钟由 f_H 时钟向 f_L 时钟转换时，f_H 将自动关闭以降低功耗。

CTRL 寄存器

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	—	LVRF	LRF	WRF
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	x	0	0

Bit 7 **FSYSON**：空闲模式时，f_{SYS} 控制位
0：除能
1：使能

Bit 6~3 未定义，读为“0”

Bit 2 **LVRF**：LVR 复位标志
0：未发生
1：发生
当低电压复位情况发生时此位设置为“1”。此位只能通过程序清零。

Bit 1 **LRF**：LVR 控制寄存器 LVRC 软件复位标志
0：未发生
1：发生
当 LVRC 寄存器包含任何未定义的 LVR 电压值时此位设置为“1”，用作软件复位功能。此位只能通过程序清零。

Bit 0 **WRF**：WDT 控制寄存器软件复位标志
0：未发生
1：发生
当 WDT 控制寄存器软件复位时此位设置为“1”。此位只能通过程序清零。

2.6.3 系统工作模式

该单片机有 5 种不同的工作模式，每种有它自身的特性，根据应用中不同的性能和功耗要求可选择不同的工作模式。单片机正常工作有两种模式：正常模式和低速模式。剩余的 3 种工作模式：休眠模式、空闲模式 0 和空闲模式 1 用于单片机 CPU 关闭时以节省耗电。

工作模式	说明			
	CPU	f_{SYS}	f_{LIRC}	f_{TBC}
正常模式	On	f_H~$f_H/64$	On	On
低速模式	On	f_L	On	On
空闲模式 0	Off	Off	On	On
空闲模式 1	Off	On	On	On
休眠模式	Off	Off	On	Off

正常模式

顾名思义，这是主要的工作模式之一，单片机的所有功能均可在此模式中实现且系统时钟由一个高速振荡器提供。该模式下单片机正常工作的时钟源来自 HIRC 振荡器。高速振荡器频率可被分为 1~64 的不等比率，实际的比率由 SMOD 寄存器中的 CKS2~CKS0 位及 HLCLK 位选择的。单片

机使用高速振荡器分频作为系统时钟可减少工作电流。

低速模式

此模式的系统时钟虽为较低速时钟源，但单片机仍能正常工作。单片机在此模式中运行所耗工作电流较低。在低速模式下，f_H 关闭。

休眠模式

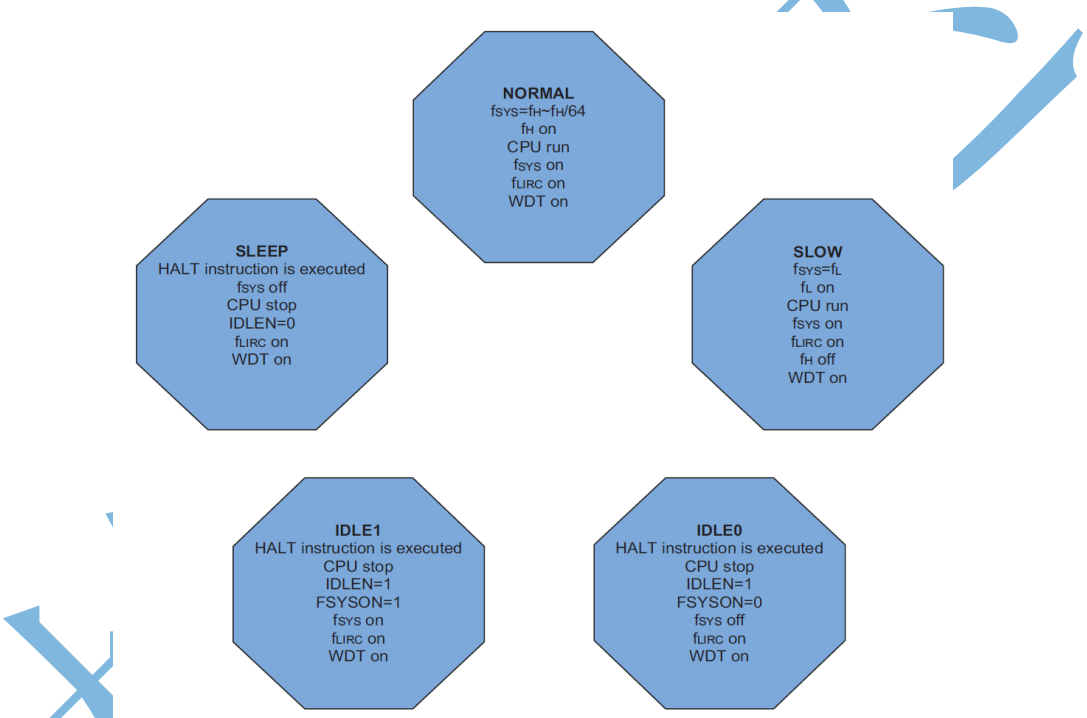
在 HALT 指令执行后且 SMOD 寄存器中 IDLEN 位为低时，系统进入休眠模式。在休眠模式中，CPU 停止运行。然而 f_{LIRC} 振荡器继续运行，看门狗定时器继续工作。

空闲模式 0

执行 HALT 指令后且 SMOD 寄存器中 IDLEN 位为高，CTRL 寄存器中 FSYSON 位为低时，系统进入空闲模式 0。在空闲模式 0 中，系统振荡器停止，CPU 停止工作。

空闲模式 1

执行 HALT 指令后且 SMOD 寄存器中 IDLEN 位为高，CTRL 寄存器中 FSYSON 位为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，但会提供一个时钟源给一些外围功能。在空闲模式 1 中，系统振荡器继续运行，该系统振荡器可以为高速或低速系统振荡器。

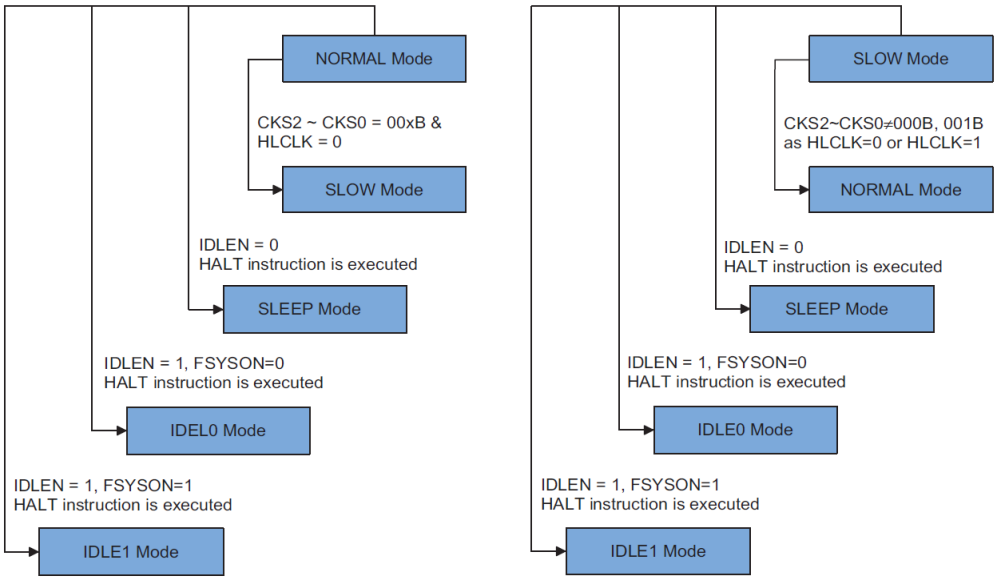


2.6.4 工作模式切换

单片机可在各个工作模式间自由切换，使得用户可根据所需选择最佳的性能/功耗比。用此方式，对单片机工作的性能要求不高的情况下，可使用较低频时钟以减少工作电流，在便携式应用上延长电池的使用寿命。

简单来说，正常模式和低速模式间的切换仅需设置 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位即可实现，而正常模式/低速模式与休眠模式/空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SMOD 寄存器中的 IDLEN 位和 CTRL 寄存器中的 FSYSON 位决定的。

当 HLCLK 位变为低电平时，时钟源将由高速时钟源 f_H 转换成时钟源 $f_H/2$~$f_H/64$ 或 f_{LIRC}。若时钟源来自 f_{LIRC}，高速时钟源将停止运行以节省耗电。此时须注意，$f_H/16$ 和 $f_H/64$ 时钟源也将停止运行。所附流程图显示了单片机在不同工作模式间切换时的变化。



正常模式切换到低速模式

系统运行在正常模式时使用高速系统振荡器，因此较为耗电。可通过设置 SMOD 寄存器中的 HLCLK 位为“0”及 CKS2~CKS0 位为“000”或“001”使系统时钟切换至运行在低速模式下。此时将使用低速系统振荡器以节省耗电。用户可在对性能要求不高的操作中使用此方法以减少耗电。低速模式的时钟源来自 LIRC 振荡器。

低速模式切换到正常模式

在低速模式系统使用 LIRC 低速振荡器。切换到使用高速系统时钟振荡器的正常模式需设置 HLCLK 位为“1”，也可设置 HLCLK 位为“0”但 CKS2~CKS0 需设为“010”、“011”、“100”、“101”、“110”或“111”。高频时钟需要一定的稳定时间，通过检测 HTO 位的状态可进行判断。高速振荡器的稳定时间由所使用高速系统振荡器的类型决定。

进入休眠模式

进入休眠模式的方法仅有一种——应用程序中执行“HALT”指令前需设置寄存器 SMOD 中 IDLEN 位为“0”。在上述条件下执行该指令后，将发生的情况如下：

- 系统时钟停止运行，应用程序停止在“HALT”指令处，但 f_{LIRC} 时钟将继续运行。
- 数据存储器中的内容和寄存器将保持当前值。
- WDT 将被清零并重新开始计数。
- 输入/输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。

进入空闲模式 0

进入空闲模式 0 的方法仅有一种——应用程序中执行“HALT”指令前需设置寄存器 SMOD 中 IDLEN 位为“1”且 CTRL 寄存器中的 FSYSON 位为“0”。在上述条件下执行该指令后，将发生的情况如下：

- 系统时钟停止运行，应用程序停止在“HALT”指令处，时基时钟和 f_{LIRC} 时钟将继续运行。
- 数据存储器中的内容和寄存器将保持当前值。
- WDT 将被清零并重新开始计数
- 输入/输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。

进入空闲模式 1

进入空闲模式 1 的方法仅有一种——应用程序中执行“HALT”指令前需设置寄存器 SMOD 中 IDLEN 位为“1”且 CTRL 寄存器中的 FSYSON 位为“1”。在上述条件下执行该指令后，将发生的情况

如下：

- 系统时钟和 f_{LIRC} 开启，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- WDT 将被清零并重新开始计数。
- 输入/输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。

2.6.5 静态电流的注意事项

由于单片机进入休眠或空闲模式的主要原因是将 MCU 的电流降低到尽可能低，可能到只有几个微安的级别（空闲模式 1 除外），所以如果要将电路的电流降到最低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入/输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。

另外还需注意单片机设为输出的 I/O 引脚上的负载。应将它们设置在有最小拉电流的状态或将它们和其它的 CMOS 输入一样接到没有拉电流的外部电路上。

在空闲模式 1 中，系统时钟开启。若系统时钟来自高速系统振荡器，额外的静态电流也可能会有几百微安。

2.6.6 唤醒

系统进入休眠或空闲模式之后，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

若由外部复位唤醒，系统会经过完全复位的过程；若由 WDT 溢出唤醒，则会发生看门狗定时器复位。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，会清零 PDF；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，这种复位会重置程序计数器和堆栈指针，其它标志保持原有状态。

PA 口中的每个引脚都可以通过 PAWU 寄存器使能下降沿唤醒功能。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。第一种情况是：相关中断除能或是中断使能且堆栈已满，则程序会在“HALT”指令之后继续执行。这种情况下，唤醒系统的中断会等到相关中断使能或有堆栈层可以使用之后才执行。第二种情况是：相关中断使能且堆栈未满，则中断可以马上执行。如果在进入休眠或空闲模式之前中断标志位已经被设置为“1”，则相关中断的唤醒功能将无效。

系统振荡器	FSTEN 位	唤醒时间 (休眠模式)	唤醒时间 (空闲模式 0)	唤醒时间 (空闲模式 1)
HXT	0	1024 HXT 周期		1~2 HXT 周期
	1	1~2 个 f_{SUB} 周期(系统在 f_{SUB} 下运行 1024 个 HXT 周期后切换到 HXT 振荡器运行)		1~2 HXT 周期
HIRC		15~16 HIRC 周期		1~2 HIRC 周期
LIRC		1~2 LIRC 周期		1~2 LIRC 周期

唤醒时间

2.6.7 编程注意事项

高速和低速振荡器都使用 SST 计数器。例如，如果系统从休眠模式下唤醒，HIRC 或 HXT 振荡器起振需要一定的延迟时间。

如果单片机从休眠模式唤醒到正常模式，则高速振荡器需要 SST 的系统延迟。HTO 为高后，单片机会执行第一条指令。

2.7 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。

2.7.1 看门狗定时器时钟源

WDT 定时器时钟源来自于内部低速振荡器 f_{LIRC}。看门狗定时器的时钟源可分频为 2^8~2^{18} 以提供更大的溢出周期，分频比由 WDTC 寄存器中的 WS2~WS0 位来决定。电压为 5V 时内部振荡器 LIRC 的频率大约为 32kHz。

需要注意的是，这个特殊的内部时钟周期随 V_{DD}、温度和制成的不同而变化。

2.7.2 看门狗定时器控制寄存器

WDTC 寄存器用于控制 WDT 溢出周期选择。

WDTC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	WE4	WE3	WE2	WE1	WE0	WS2	WS1	WS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	0	1	1

Bit 7~3 **WE4~WE0**：WDT 软件控制
10101B或01010B：使能
其它值：MCU 复位
如果单片机复位且由干扰引起，需要 2~3 个 LIRC 周期响应复位。且在复位后 CTRL 寄存器中的 WRF 标志位会被置位。

Bit 2~0 **WS2~WS0**：选择看门狗溢出周期
000：$2^8/f_S$
001：$2^{10}/f_S$
010：$2^{12}/f_S$
011：$2^{14}/f_S$
100：$2^{15}/f_S$
101：$2^{16}/f_S$
110：$2^{17}/f_S$
111：$2^{18}/f_S$
这三位控制看门狗时钟的分频比，进而控制看门狗的溢出周期

2.7.3 看门狗定时器操作

该单片机的看门狗定时器时钟源来自 f_{LIRC} 振荡器，因此总是开启的。当 WDT 溢出时，它产生一个芯片复位的动作。这也就意味着正常工作期间，用户需在应用程序中看门狗定时器溢出前将看门狗定时器清零以防止其产生复位，可使用清看门狗指令实现。在程序运行过程由于某些无法预知的原因会使程序跳转到一个未知的地址或进入一个死循环，此时这些清除指令无法被正确执行，在这种情况下，看门狗定时器会计数溢出以使单片机复位。WDTC 寄存器中的 WE4~WE0 位可使能看门狗定时器。如果 WE4~WE0 为 10101B 或 01010B，则 WDT 使能；如果由于受到干扰，WE4~WE0 变为其它任意值，则经过 2~3 个 LIRC 时钟周期后单片机复位。

WE4 ~ WE0位	WDT功能
10101B	使能
01010B	使能
其它值	MCU复位

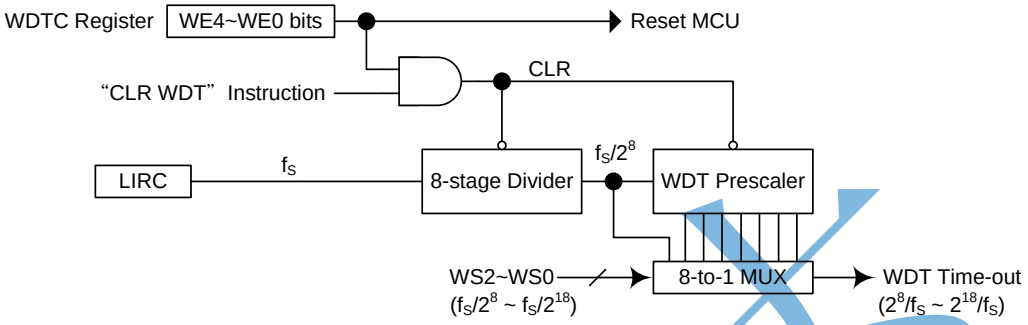
看门狗定时器使能/除能控制

程序正常运行时，WDT 溢出将导致芯片复位，并置位状态标志位 TO。若系统处于休眠或空闲模式，当 WDT 发生溢出时，状态寄存器中的 TO 标志位会被置位，且只有程序计数器 PC 和堆栈指针 SP 会被复位。有三种方法可以用来清除 WDT 的内容。第一种是 WDT 复位，即将 WE4~WE0 位

设置成除了 01010B 和 10101B 外的任意值；第二种是通过软件清除指令，而第三种是通过“HALT”指令。

该单片机用一条指令清除看门狗定时器，“CLR WDT”。因此只要执行“CLR WDT”清除 WDT。

当设置分频比为 2^{18} 时，溢出周期最大。例如，时钟源为 LIRC 振荡器，分频比为 2^{18} 时最大溢出周期约 8s，分频比为 2^8 时最小溢出周期约 7.8ms。



看门狗定时器

2.8 复位和初始化

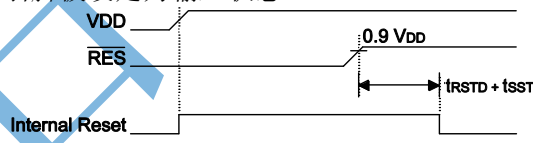
复位功能是任何单片机中基本的部分，使得单片机可以设定一些与外部参数无关的先置条件。最重要的复位条件是在单片机首次上电以后，经过短暂的延迟，内部硬件电路使得单片机处于预期的稳定状态并开始执行第一条程序指令。上电复位以后，在程序执行之前，部分重要的内部寄存器将会被设定为预先设定的状态。程序计数器就是其中之一，它会被清除为零，使得单片机从最低的程序存储器地址开始执行程序。

2.8.1 复位功能

包括内部和外部事件触发复位，单片机共有四种复位方式：

•上电复位

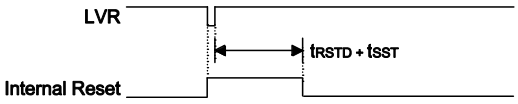
这是最基本且不可避免的复位，发生在单片机上电后。除了保证程序存储器从开始地址执行，上电复位也使得其它寄存器被设定在预设条件，所有的输入/输出端口控制寄存器在上电复位时会保持高电平，以确保上电后所有引脚被设定为输入状态。



上电复位时序图

•低电压复位 — LVR

单片机具有低电压复位电路，用来监测它的电源电压。低电压复位功能始终使能于特定的电压值，V_{LVR}。例如在更换电池的情况下，单片机供应的电压可能会在 0.9V~V_{LVR} 之间，这时 LVR 将会自动复位单片机且 CTRL 寄存器中的 LVRF 标志位置位。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 LVRC 寄存器中的 LVS 位设置固定为 2.55V。若由于受到干扰 LVS7~LVS0 变为其它值时，需经过 2~3 个 LIRC 周期响应复位。此时 CTRL 寄存器的 LRF 位被置位。上电后寄存器的值为 01010101B。正常执行时 LVR 会于休眠或空闲时自动除能关闭。



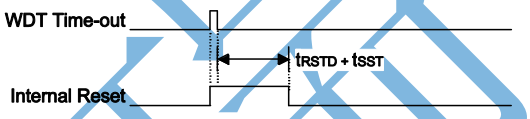
低电压复位时序图

LVRC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	LVS7	LVS6	LVS5	LVS4	LVS3	LVS2	LVS1	LVS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	1	0	1

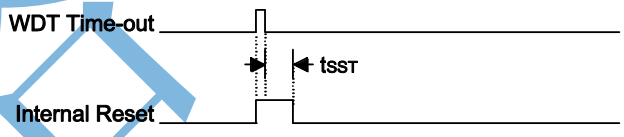
Bit 7~0 **LVS7~LVS0**：LVR 电压选择
01010101B：2.55V
00110011B：2.55V
10011001B：2.55V
10101010B：2.55V
其它值：MCU 复位(复位需要 2~3 个 LIRC 周期的响应时间)。
注：可以通过 S/W 写 00H~FFH 来控制 LVR 电压，也可复位单片机。如果单片机复位且由 LVRC 寄存器引起，则在复位后 CTRL 寄存器中的 LRF 标志位会被置位。

•正常运行时看门狗溢出复位
正常运行时看门狗溢出，标志位 TO 将被设为 1。



正常运行时看门狗溢出时序图

•空闲/休眠时看门狗溢出复位
空闲/休眠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，绝大部分的条件保持不变。图中 t_{SST} 的详细说明请参考交流电气特性。



空闲/休眠时看门狗溢出复位时序图

2.8.2 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，由空闲/休眠功能或看门狗计数器等几种控制器操作控制。复位标志位如下所示：

T0	PDF	复位条件
0	0	上电复位
u	u	正常模式或低速模式时的 LVR 复位
1	u	正常模式或低速模式时的 WDT 溢出复位
1	1	空闲模式或休眠模式时的 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	所有中断被除能
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
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

不同的复位形式对单片机内部寄存器的影响是不同的。为保证复位后程序能正常执行，了解寄存器在特定条件复位后的设置是非常重要的，下表即为不同方式复位后内部寄存器的状况。注意若芯片有多种封装类型，表格反应较大的封装的情况。

寄存器	上电复位	WDT 溢出复位 (正常模式)	LVR 复位 (正常模式)	LVR 复位 (空闲或休眠模式)	WDT 溢出复位 (空闲或休眠模式)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	---- xxxx	---- uuuu	---- uuuu	---- uuuu	---- uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
SMOD	0000 0011	0000 0011	0000 0011	0000 0011	uuuu uuuu
INTEG	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD	---- --11	---- --11	---- --11	---- --11	---- --uu
PDC	---- --11	---- --11	---- --11	---- --11	---- --uu
PDPU	---- --00	---- --00	---- --00	---- --00	---- --uu
CTRL	0--- -x00	0--- -000	0--- -000	0--- -000	u--- -uuu
LVRC	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu
WDTC	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu
TBC	--00 ----	--00 ----	--00 ----	--00 ----	--uu ----
TMR0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

TMR0C	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
TMR1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TSCC	-000 0-00	-000 0-00	-000 0-00	-000 0-00	-uuu u-uu
IICC0	---- 000-	---- 000-	---- 000-	---- 000-	---- uuu-
IICC1	1000 0001	1000 0001	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA	xxxx xxx-	xxxx xxx-	xxxx xxx-	xxxx xxx-	uuuu uuu-
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
------	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PRM	--00 -000	--00 -000	--00 -000	--00 -000	--uu -uuu

注： “-”表示未定义
“x”表示未知
“u”表示不改变

2.9 输入/输出端口

BS45F1B 单片机的输入/输出口控制具有很大的灵活性。大部分引脚都可在用户程序控制下被设定为输入或输出，所有引脚的上拉电阻设置以及指定引脚的唤醒设置也都由软件控制，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需求。

此单片机提供 PA~PD 双向输入/输出口。这些寄存器在数据存储器有特定的地址。所有 I/O 口用于输入输出操作。作为输入操作，输入引脚无锁存功能，也就是说输入数据必须在执行“MOV A，[m]”，T2 的上升沿准备好，m 为端口地址。对于输出操作，所有数据都是被锁存的，且保持不变直到输出锁存被重写。

2.9.1 输入/输出寄存器列表

寄存器名称	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	—	—	—	—	—	—	PD1	PD0
PDC	—	—	—	—	—	—	PDC1	PDC0
PDPU	—	—	—	—	—	—	PDPU1	PDPU0

2.9.2 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。为了免去外部上拉电阻，当引脚规划为输入时，可由内部连接到一个上拉电阻，这些上拉电阻可通过寄存器 PAPU~PDPU 来设置，它用一个 PMOS 晶体管来实现上拉电阻功能。

PAPU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PAPU**：PA 口 bit 7~bit 0 上拉电阻控制
0：除能
1：使能

PBPU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PBPU**：PB 口 bit 7~bit 0 上拉电阻控制
0：除能
1：使能

PCPU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PCPU**：PC 口 bit 7~bit 0 上拉电阻控制
0：除能
1：使能

PDPU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PDPU1	PDPU0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”
Bit 1~0 **PDPU**：PD 口 bit 1~bit 0 上拉电阻控制
0：除能
1：使能

2.9.3PA 口唤醒

当使用暂停指令“HALT”迫使单片机进入空闲/休眠模式状态，单片机的系统时钟将会停止以降低功耗，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 口的其中一个引脚从高电平转为低电平。这个功能特别适合于通过外部开关来唤醒的应用。PA 口上的每个引脚是可以通过设置 PAWU 寄存器来单独选择是否具有唤醒功能。

PAWU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PAWU**：PA 口 bit 7~bit 0 唤醒控制
0：除能
1：使能

2.9.4 输入/输出端口控制寄存器

每一个输入/输出口都具有各自的控制寄存器(PAC~PDC)用来控制输入/输出状态。通过这些控制寄存器，每个 CMOS 输出或输入都可以通过软件动态控制。所有的 I/O 端口的引脚都各自对应于 I/O 端口控制寄存器的某一位。若 I/O 引脚要实现输入功能，则对应的控制寄存器的位需要设置为“1”，这时程序指令可以直接读取输入脚的逻辑状态。若控制寄存器相应的位被设定为“0”，则此引脚被设置为 COMS 输出。当引脚设置为输出状态时，程序指令读取的是输出端口寄存器的内容。注意，如果对输出口做读取动作时，程序读取到的是内部输出数据锁存器中的状态，而不是输出引脚上实际的逻辑状态。

PAC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PAC**：PA 口 bit 7~bit 0 输入/输出控制
0：输出
1：输入

PBC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PBC**：PB 口 bit 7~bit 0 输入/输出控制
0：输出
1：输入

PCC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PCC**：PC 口 bit 7~bit 0 输入/输出控制
0：输出
1：输入

PDC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PDC1	PDC0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	1	1

Bit 7~2 未定义，读为“0”
Bit 1~0 **PDC**：PD 口 bit 1~bit 0 输入/输出控制
0：输出
1：输入

2.9.5 引脚重置功能

引脚的多功能可以增加单片机应用的灵活性。有限的引脚个数将会限制设计者，而引脚的多功能将会解决很多此类问题。每个功能可单独选择所在的引脚，以及一个确定的优先级，使得引脚上多种功能可以同时使用。此外，一些引脚功能可以通过寄存器 PRM 进行设定。

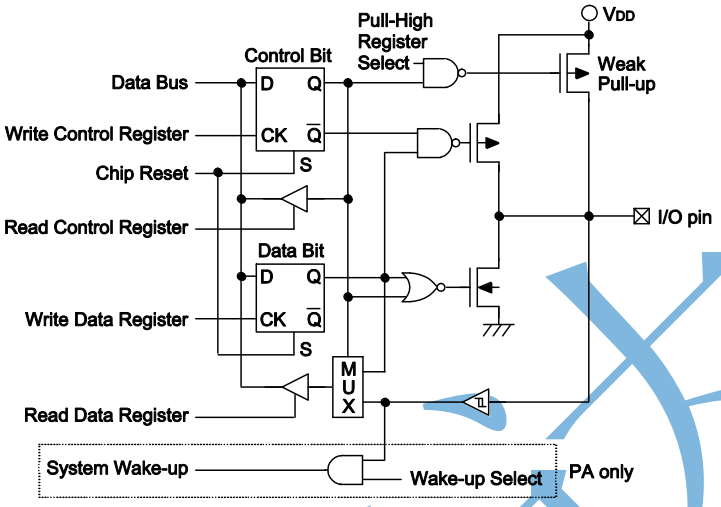
PRM 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	SDAPRM	SCLPRM	—	PFD0PRM	INT1PRM	INT0PRM
R/W	—	—	R/W	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7~6 未定义，读为“0”
Bit 5 **SDAPRM**：SDA 引脚重置控制位
0：SDA on PA0
1：SDA on PA1
Bit 4 **SCLPRM**：SCL 引脚重置控制位
0：SCL on PA2
1：SCL on PA7
Bit 3 未定义，读为“0”
Bit 2 **PFD0PRM**：PFD0 引脚重置控制位
0：PFD0 on PA5
1：PFD0 on PA4
Bit 1 **INT1PRM**：INT1 引脚重置控制位
0：INT1 on PA7
1：INT1 on PD1
Bit 0 **INT0PRM**：INT0 引脚重置控制位
0：INT0 on PA1
1：INT0 on PA3

2.9.6 输入/输出引脚结构

下图为输入/输出引脚的内部结构图。输入/输出引脚的准确逻辑结构图可能与此图不同，这里只是为了方便对功能的理解提供的一个参考。图中引脚共用并非针对所有单片机。



通用输入/输出端口结构

2.9.7 编程注意事项

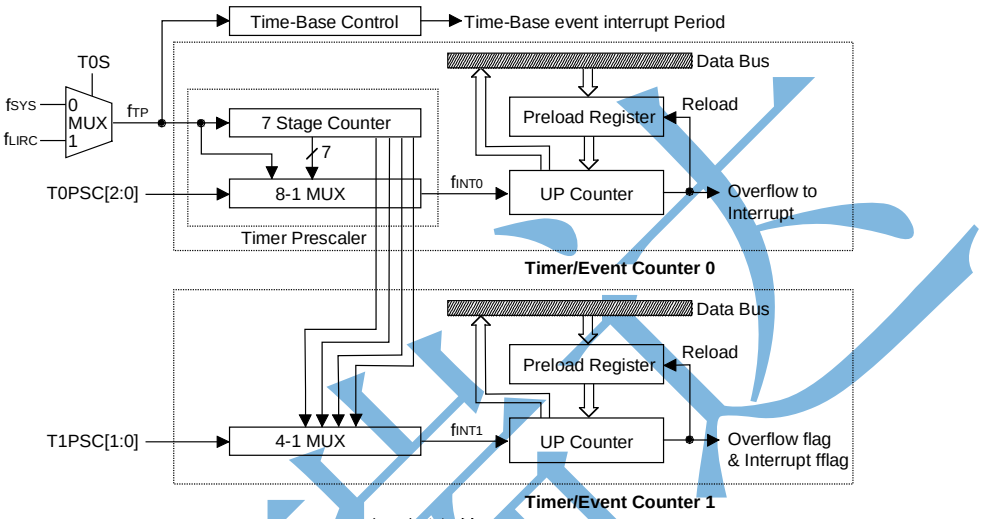
在编程中，最先要考虑的是端口的初始化。复位之后，所有的输入/输出数据及端口控制寄存器都将被设为逻辑高。所有输入/输出引脚默认为输入状态，而其电平则取决于其它相连接电路以及是否选择了上拉电阻。如果端口控制寄存器 PAC~PDC，某些引脚位被设定输出状态，这些输出引脚会有初始高电平输出，除非数据寄存器端口 PA~PD 在程序中被预先设定。设置哪些引脚是输入及哪些引脚是输出，可通过设置正确的值到适当的端口控制寄存器，或使用指令“SET [m].i”及“CLR [m].i”来设定端口控制寄存器中个别的位。注意，当使用这些位控制指令时，系统即将产生一个读-修改-写的操作。单片机需要先读入整个端口上的数据，修改个别的位，然后重新把这些数据写入到输出端口。

PA 口的每个引脚都带唤醒功能。单片机处于休眠或空闲模式时，有很多方法可以唤醒单片机，其中之一就是通过 PA 口任一引脚电平从高到低转换的方式，可以设置 PA 口一个或多个引脚具有唤醒功能。

2.10 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。该单片机具有 2 个 8 位的向上计数器。并且提供了一个内部时钟分频器，以扩大定时器的范围。

有两种和定时/计数器相关的寄存器。第一种类型的寄存器是用来存储实际的计数值，赋值给此寄存器可以设定初始值，读取此寄存器可获得定时/计数器的内容；第二种类型的寄存器为定时器控制寄存器，用来定义定时/计数器的定时设置。



定时/计数器 0/1

2.10.1 配置定时/计数器输入时钟源

定时/计数器的时钟源可以来自系统时钟 f_{SYS} 或 f_{LIRC} 振荡器，由 TMR0C 寄存器的 T0S 位选择使用哪种时钟源。内部时钟首先由分频器分频，分频比由定时器控制寄存器的位 T0PSC0~T0PSC2 或时隙控制寄存器 TSCC 中的 T1PSC0~T1PSC1 来确定。

2.10.2 定时/计数寄存器 — TMR0, TMR1

定时/计数寄存器 TMR0 和 TMR1 是位于特殊数据存储器内的特殊功能寄存器，用于储存定时器的当前值。当收到一个内部计数脉冲，寄存器的值将会加一。定时器将从预置寄存器所载入的值开始计数，到 FFH 时定时器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新载入并继续计数。

注意，上电后预置寄存器处于未知状态。为了得到定时器的最大计算范围 FFH，预置寄存器需要先清为零。注意，如果定时/计数器在关闭条件下，写数据到预置寄存器，会立即写入实际的定时器。而如果定时/计数器已经打开且正在计数，在这个周期内写入到预置寄存器的任何新数据将保留在预置寄存器，直到溢出发生时才被写入实际定时器。

2.10.3 定时/计数控制寄存器 — TMR0C, TSCC

定时/计数控制寄存器和时隙控制寄存器分别为 TMR0C、TSCC，分别配合 TMR0 和 TMR1 寄存器控制定时/计数器的全部操作。在使用定时器之前，需要先正确地设定定时/计数控制寄存器，以便保证定时器能正确操作，而这个过程通常在程序初始化期间完成。

定时/计数控制寄存器 TMR0C 和时隙控制寄存器 TSCC 的第 4 位即 T0ON 和 T1ON，用于定时器开关控制，设定为逻辑高时，计数器开始计数，而清零时则停止计数。定时/计数控制寄存器 TMR0C 的第 0~2 位和时隙控制寄存器的第 0~1 位用来控制输入时钟预分频器。T0S 位用来选择内部时钟源。

定时/计数器 1 也可用于触控按键功能，为时隙计数器提供时钟源。可通过 TSCC 寄存器设置定时/计数器 1 用于触控按键模式。

TMR0C 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	PFD0C	T0S	T0ON	—	T0PSC2	T0PSC1	T0PSC0
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7　未定义，读为“0”

Bit 6　**PFD0C**：I/O 或 PFD0 选择位
0：I/O
1：PFD0

Bit 5　**T0S**：定时器时钟源选择位
0：f_{SYS}
1：f_{LIRC}

Bit 4　**T0ON**：定时/计数器 0 控制位
0：除能
1：使能

Bit 3　未定义，读为“0”

Bit 2~0　**T0PSC2 ~ T0PSC0**：选择定时器 0 预分频比选择位
定时器 0 内部时钟 =
000：f_{TP}
001：$f_{TP}/2$
010：$f_{TP}/4$
011：$f_{TP}/8$
100：$f_{TP}/16$
101：$f_{TP}/32$
110：$f_{TP}/64$
111：$f_{TP}/128$

TSCC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	T1M	TSON	T1ON	TSCLR	—	T1PSC1	T1PSC0
R/W	—	R/W	R/W	R/W	R/W	—	R/W	R/W
POR	—	0	0	0	0	—	0	0

Bit 7　未定义，读为“0”

Bit 6　**T1M**：定时器 1 工作模式选择位
0：定时器模式
1：触控按键模式

Bit 5　**TSON**：6-bit 时隙计数器控制位
0：除能
1：使能

Bit 4　**T1ON**：定时/计数器 1 控制位
0：除能
1：使能

Bit 3　**TSCLR**：6-bit 时隙计数器清除控制位
0：计数器保持不变
1：清除计数器
此位须设置 1→0。

Bit 2　未定义，读为“0”

Bit 1~0　**T1PSC1 ~ T1PSC0**：选择定时器 1 预分频比选择位
定时器 1 内部时钟 =
00：$f_{TP}/8$
01：$f_{TP}/16$
10：$f_{TP}/32$
11：$f_{TP}/64$

2.10.4 定时器操作

定时/计数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。

f_{SYS} 或 f_{LIRC} 振荡器被用来当定时器的输入时钟源。然而，该定时器时钟源被预分频器进一步分频，分频比是由定时器控制寄存器 TMR0C 中的 T0PSC2~T0PSC0 位或时隙控制寄存器 TSCC 中的 T1PSC1~T1PSC0 位来确定。定时器使能位，即 T0ON 或 T1ON 位需要设为逻辑高，才能令定时器工作。每次内部时钟由高到低的电平转换都会使定时器值增加一。当定时器计数已满即溢出时，会产生中断信号且定时器会重新载入预置寄存器的值，然后继续计数。定时器溢出以及相应的内部中断产生也是唤醒暂停模式的一种方法，然而，通过设置中断寄存器中的定时器中断使能位为 0，可以禁止计数器中断。

2.10.5 预分频器

TMR0C 寄存器的 T0PSC0~T0PSC2 位和 TSCC 寄存器的 T1PSC1~T1PSC0 位用来确定定时/计数器 0/1 的内部时钟的分频比，从而能够设置更长的定时器溢出周期。

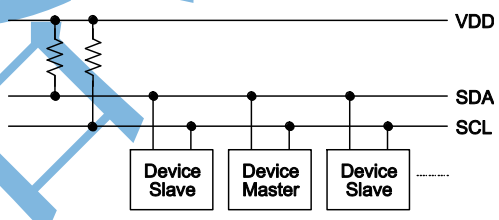
2.10.6 编程注意事项

当读取定时/计数器值或写数据到预置寄存器时，计数时钟会被禁止以避免发生错误，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需要正确的设置，否则相应定时/计数器内部中断仍然无效。在定时/计数器打开之前，需要确保先载入定时/计数寄存器的初始值，这是因为在上电后，定时/计数寄存器中的初始值是未知的。

定时/计数器初始化后，可以使用定时/计数器控制寄存器中的使能位来打开或关闭定时器。当定时/计数器产生溢出，中断控制寄存器中相应的中断请求标志将置位。若定时/计数器中断允许，将会依次产生一个中断信号。不管中断是否允许，在省电状态下，定时/计数器的溢出也会产生唤醒。若在省电模式下，不需要定时器中断唤醒系统，可以在执行“HALT”指令进入空闲/休眠模式之前将相应中断请求标志位置位

2.11 I^2C 接口

I^2C 可以和传感器，EEPROM 内存等外部硬件接口进行通信。最初是由飞利浦公司研制，是适用于同步串行数据传输的双线式低速串行接口。I^2C 接口具有两线通信，非常简单的通信协议和在同一总线上和多个设备进行通信的能力的优点，使之在很多的应用场合中大受欢迎。

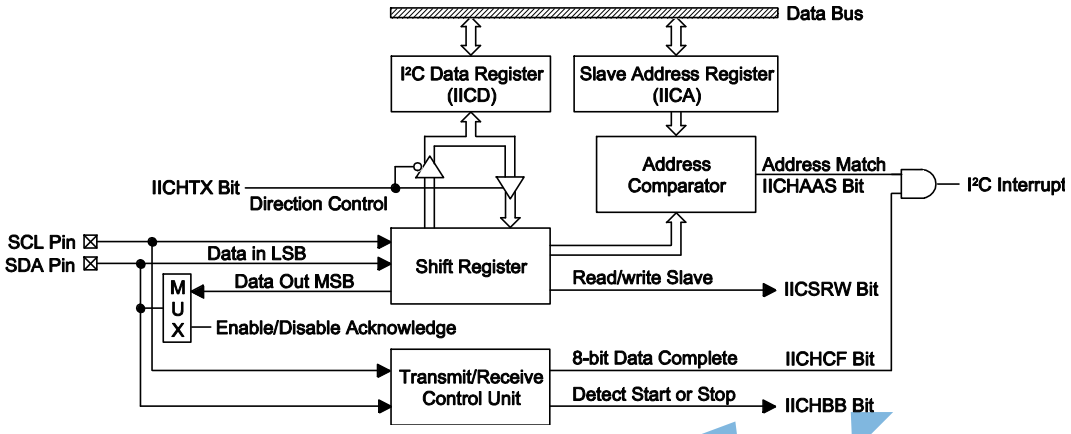


I^2C 主从总线连接图

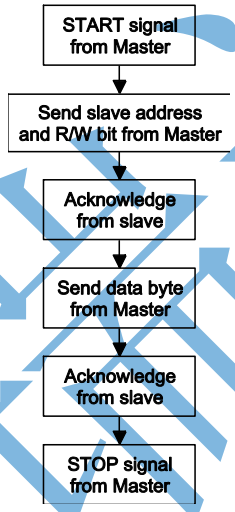
2.11.1 I^2C 接口操作

I^2C 串行接口是一个双线的接口，有一条串行数据线 SDA 和一条串行时钟线 SCL。由于可能有多个设备在同一条总线上相互连接，所以这些设备的输出都是开漏型输出。因此应在这些输出口上都应加上拉电阻。应注意的是：I^2C 总线上的每个设备都没有选择线，但分别与唯一的地址一一对应，用于 I^2C 通信。

如果有两个设备通过双向的 I^2C 总线进行通信，那么就存在一个主机和一个从机。主机和从机都可以用于传输和接收数据，但只有主机才可以控制总线动作。那些处于从机模式的设备，要在 I^2C 总线上传输数据只有两种方式，一是从机发送模式，二是从机接收模式。



I^2C 方框图



2.11.2 I^2C 寄存器

I^2C 总线的四个控制寄存器是 IICC0，IICC1，IICA 和 I2CTOC 及一个数据寄存器 IICD。IICD 寄存器用于存储正在传输和接收的数据，当单片机将数据写入 I^2C 总线之前，实际将被传输的数据存放在寄存器 IICD 中。从 I^2C 总线接收到数据之后，单片机就可以从寄存器 IICD 中读取这个数据。I^2C 总线上的所有传输或接收到的数据都必须通过寄存器 IICD。I^2C 引脚与 I/O 引脚共用，通过寄存器 IICC0 中的 I2CEN 位来使能。

寄存器名称	Bit							
	7	6	5	4	3	2	1	0
IICC0	—	—	—	—	I2CDBNC1	I2CDBNC0	I2CEN	—
IICC1	IICHCF	IICHAAS	IICHBB	IICHTX	IICTXAK	IICSRW	IICRNIC	IICRXAK
IICD	IICDD7	IICDD6	IICDD5	IICDD4	IICDD3	IICDD2	IICDD1	IICDD0
IICA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
I2CTOC	I2CTOEN	I2CTOF	I2CTOS5	I2CTOS4	I2CTOS3	I2CTOS2	I2CTOS1	I2CTOS0

I^2C 寄存器列表

IICC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	I2CDBNC1	I2CDBNC0	I2CEN	—
R/W	—	—	—	—	R/W	R/W	R/W	—

POR	—	—	—	—	0	0	0	—

Bit 7~4 未定义，读为“0”

Bit 3~2 **I2CDBNC1~I2CDBNC0**：I²C 抖动时间选择位
00：无抖动
01：2 个系统时钟的抖动
10：4 个系统时钟的抖动
11：4 个系统时钟的抖动

Bit 1 **I2CEN**：I²C 使能位
0：除能
1：使能

Bit 0 未定义，读为“0”

IICC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	IICHCF	IICHAAS	IICHBB	IICHTX	IICTXAK	IICSRW	IICRNIC	IICRXAK
R/W	R	R	R	R/W	R/W	R	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7 **IICHCF**：I²C 总线数据传输完成标志位
0：数据正在被传输
1：8 位数据传输完成
IICHCF 是数据传输标志位。数据正在传输时该位为低。当 8 位数据传输完成时，此位为高并产生一个中断。

Bit 6 **IICHAAS**：I²C 总线地址匹配标志位
0：地址不匹配
1：地址匹配
此标志位用于决定从机地址是否与主机发送地址相同。若地址匹配此位为高，否则此位为低。

Bit 5 **IICHBB**：I²C 总线忙标志位
0：I²C 总线闲
1：I²C 总线忙
当检测到 START 信号时 I²C 忙，此位变为高电平。当检测到 STOP 信号时 I²C 总线停止，该位变为低电平。

Bit 4 **IICHTX**：从机处于发送或接收模式标志位
0：从机处于接收模式
1：从机处于发送模式

Bit 3 **IICTXAK**：I²C 总线发送确认标志位
0：从机发送确认标志
1：从机没有发送确认标志
单片机接收 8 位数据之后会将该位在第九个时钟传到总线上。如果单片机想要接收更多的数据，则应在接收数据之前将此位设置为“0”。

Bit 2 **IICSRW**：I²C 从机读/写标志位
0：从机应处于接收模式
1：从机应处于发送模式
IICSRW 位是从机读写标志位。决定主机是否希望传输或接收来自 I²C 总线的数据。当传输地址和从机的地址相同时，IICHAAS 位会被设置为高，主机将检测 IICSRW 位来决定进入发送模式还是接收模式。如果 IICSRW 位为高时，主机会请求从总线上读数据，此时设备处于传输模式。当 IICSRW 位为“0”时，主机往总线上写数据，设备处于接收模式以读取该数据。

Bit 1 **IICRNIC**：I²C 内部时钟使用选择位
0：I²C 使用内部时钟运行
1：I²C 无需使用内部时钟运行
在休眠，空闲（慢速），正常（慢速）模式下，I²C 模块无需使用内部时钟运行，如果 I²C 中断使能，将产生中断，此时 MCU 应设成正常模式下，并将 IICRCIN 位设为“0”。

Bit 0 **IICRXAK**：I²C 总线接收确认标志位
0：从机接收到确认标志
1：从机没有接收到确认标志
IICRXAK 位是接收确认标志位。如果 IICRXAK 位被重设为“0”即 8 位数据传输之后，设备在第九个时钟有接受到一个正确的确认位。如果单片机处于发送状态，发送方会检查 IICRXAK 位来判断接收方是否愿意继续接收下一个字节。因此直到 IICRXAK 为“1”时，传输方停止发送数据。这时，传输方将释放 SDA 线，主机发出停止信号。

I2CTOC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	I2CTOEN	I2CTOF	I2CTOS5	I2CTOS4	I2CTOS3	I2CTOS2	I2CTOS1	I2CTOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **I2CTOEN**：I^2C 超时控制位
0：除能
1：使能

Bit 6 **I2CTOF**：I^2C 超时标志位
0：没有超时
1：超时发生

Bit 5~0 **I2CTOS5~I2CTOS0**：I^2C 超时时间定义位
I^2C 超时时钟源是 $f_{LIRC}/32$
I^2C 超时时间计算方法：[I2CTOS5 : I2CTOS0]+1) ×(32/f_{LIRC})

IICD 寄存器用于存储发送和接收的数据。在单片机尚未将数据写入到 I^2C 总线中时，要传输的数据应存在 IICD 中。I^2C 总线接收到数据之后，单片机就可以从 IICD 寄存器中读取。

IICD 寄存器

Bit	7	6	5	4	3	2	1	0
Name	IICDD7	IICDD6	IICDD5	IICDD4	IICDD3	IICDD2	IICDD1	IICDD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	×	×	×	×	×	×	×	×

“×”为未知

Bit 7~0 **IICDD7~IICDD0**：I^2C 数据缓冲区 bit 7~bit 0

IICA 寄存器

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	×	×	×	×	×	×	×	—

“×”为未知

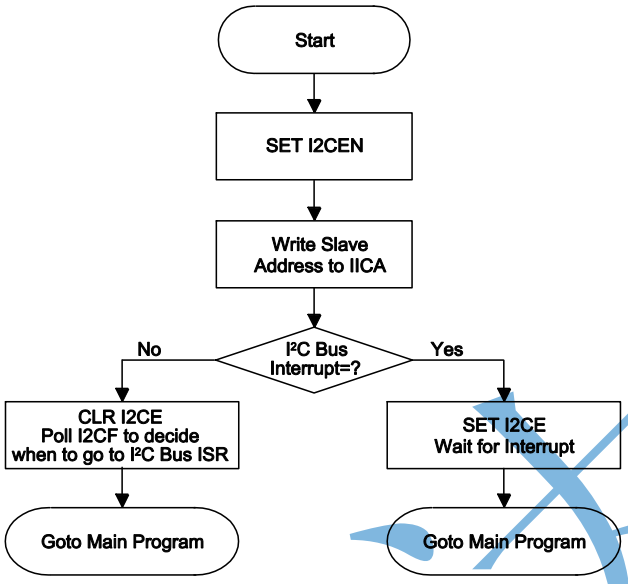
Bit 7~1 **IICA6~IICA0**：I^2C 从机地址位
IICA6~IICA0 是 I^2C 从机地址位 bit 6~bit 0 位。IICA 寄存器用于存放 7 位从机地址，寄存器 IICA 中的第 7~1 位是单片机的从机地址，位 0 未定义。如果接至 I^2C 的主机发送处的地址和寄存器 IICA 中存储的地址相符，那么就选中了这个从机。

Bit 0 未定义，读为“0”

2.11.3 I^2C 总线通信

I^2C 总线上的通信需要四步完成，一个起始信号，一个从机地址发送，一个数据传输，还有一个停止信号。当起始信号被写入 I^2C 总线时，总线上的所有从机都会接收到这个起始信号并且被通知总线上会即将有数据到达。数据的前 7 位是从机地址，高位在前，低位在后。如果发出的地址和从机地址匹配，IICC1 寄存器的 IICHAAS 位会被置位，同时产生 I^2C 中断。进入中断服务程序后，系统要检测 IICHAAS 位，以判断 I^2C 总线中断是来自从机地址匹配，还是来自 8 位数据传递完毕。在数据传递中，注意的是，在 7 位从机地址被发送后，接下来的一位，即第 8 位，是读/写控制位，该位的值会反映到 IICSRW 位中。从机通过检测 IICSRW 位以确定主控制器是要进入发送模式还是接收模式。在 I^2C 总线开始传送数据前，需要先初始化 I^2C 总线，初始化 I^2C 总线步骤如下：

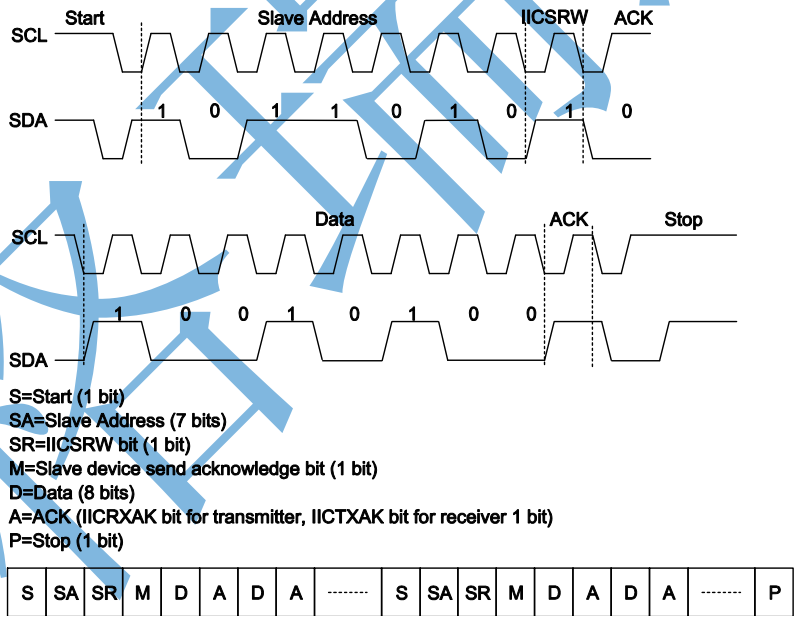
步骤 1
设置 IICC0 寄存器中 I2CEN 位为“1”，以使能 I^2C 总线
步骤 2
向 I^2C 总线地址寄存器 IICA 写入从机地址。
步骤 3
设置 I2CE 位，以使能 I^2C 中断。



I²C 总线初始化流程图

2.11.4 I²C 总线起始信号

起始信号只能由连接 I²C 总线主机产生，而不是由只做从机设备产生。I²C 总线上的所有从机都可以侦测到起始信号。如果有从机侦测到起始信号，则表明 I²C 总线处于忙碌状态，并会置位 IICHBB。起始信号是指在 SCL 为高电平时，SDA 线上发生从高到低的电平变化。



注：*当从机地址匹配时，单片机必须选择设置为发送模式还是接收模式。若设置为发送模式，需写数据至 IICD 寄存器；若设置为接收模式，需立即从 IICD 寄存器中虚读数据以释放 SCL 线。

I²C 通信时序图

2.11.5 从机地址

I²C 总线上的所有从机都会侦测由主机发出的起始信号。发送起始信号后，紧接着主机会发送从机地址以选择要进行数据传输的从机。所有在 I²C 总线上的从机接收到 7 位地址数据后，都会将其与各自内部的地址进行比较。如果从机从主机上接收到的地址与自身内部的地址相匹配，则会产生一个 I²C 总线中断信号。地址位接下来的一位为读/写状态位(即第 8 位)，将被保存到 IICC1 寄存器的 IICSRW 位，随后发出一个低电平应答信号（即第 9 位）。当单片机从机的地址匹配时，会将状

态标志位 IICHAAS 置位。

I^2C 总线有两个中断源，当程序运行至中断服务子程序时，通过检测 IICHAAS 位以确定 I^2C 总线中断是来自从机地址匹配，还是来自 8 位数据传递完毕。当是从机地址匹配发生中断时，则从机或是用于发送模式并将数据写进 IICD 寄存器，或是用于接收模式并从 IICD 寄存器中读取空值以释放 SCL 线。

2.11.6 I^2C 总线读/写信号

IICC1 寄存器的 IICSRW 位用来表示主机是要从 I^2C 总线上读取数据还是要将数据写到 I^2C 总线上。从机则通过检测该位以确定自己是作为发送方还是接收方。当 IICSRW 置 1，表示主机要从 I^2C 总线上读取数据，从机则作为发送方，将数据写到 I^2C 总线；当 IICSRW 清零，表示主机要写数据到 I^2C 总线上，从机则做为接收方，从 I^2C 总线上读取数据。

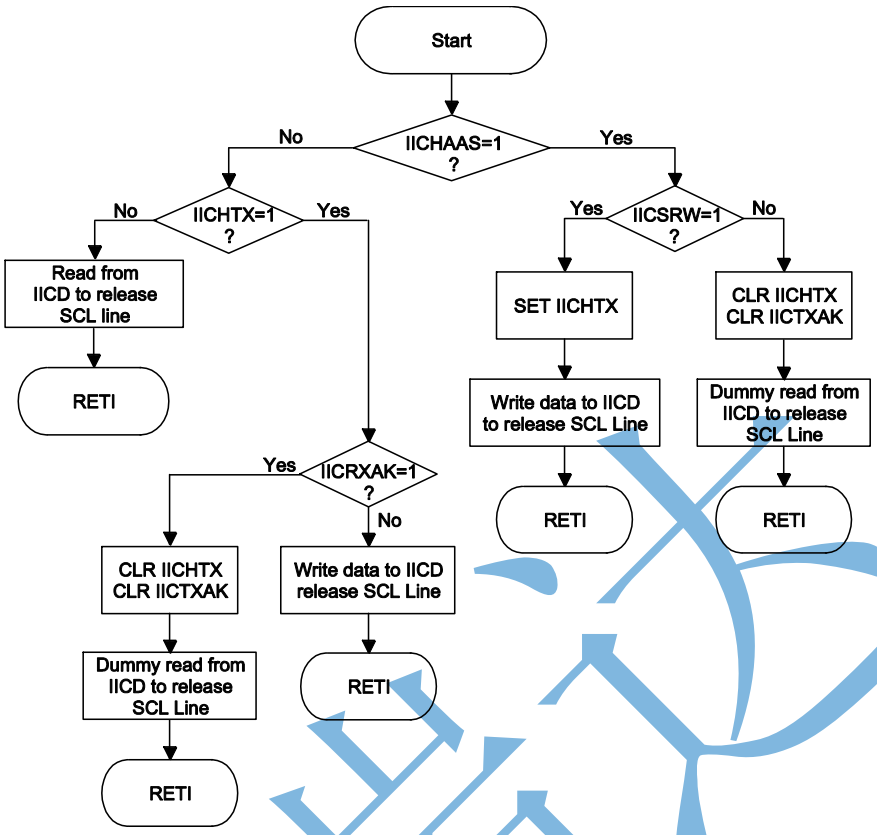
2.11.7 I2C 总线从机地址确认信号

主机发送呼叫地址后，当 I^2C 总线上的任何从机内部地址与其匹配时，会发送一个应答信号。此应答信号会通知主机有从机已经接收到了呼叫地址。如果主机没有收到应答信号，则主机必须发送停止(STOP)信号以结束通信。当 IICHAAS 为高时，表示从机接收到的地址与自己内部地址匹配，则从机需检查 IICSRW 位，以确定自己是作为发送方还是作为接收方。如果 IICSRW 位为高，从机须设置成发送方，这样会置位 IICC1 寄存器的 IICHTX 位。如果 IICSRW 位为低，从机须设置成接收方，这样会清零 IICC1 寄存器的 IICHTX 位。

2.11.8 I^2C 总线数据和确认信号

在从机确认接收到从地址后，会进行 8 位宽度的数据传输。这个数据传输顺序是的高位在前，低位在后。接收方在接收到 8 位数据后必须发出一个应答信号(“0”)以继续接收下一个数据。如果发送方没接收到应答信号，发送方将释放 SDA 线，同时，主机将发出 STOP 信号以释放 I^2C 总线。所传送的数据存储在 IICD 寄存器中。如果设置成发送方，从机必须先将欲传输的数据写到 IICD 寄存器中；如果设置成接收方，从机必须从 IICD 寄存器读取数据。

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(IICTXAK)。被设为发送方的从机将检测寄存器 IICC1 中的 IICRXAK 位以判断是否传输下一个字节的数据，如果单片机不传输下一个字节，那么它将释放 SDA 线并等待接收主机的停止信号。

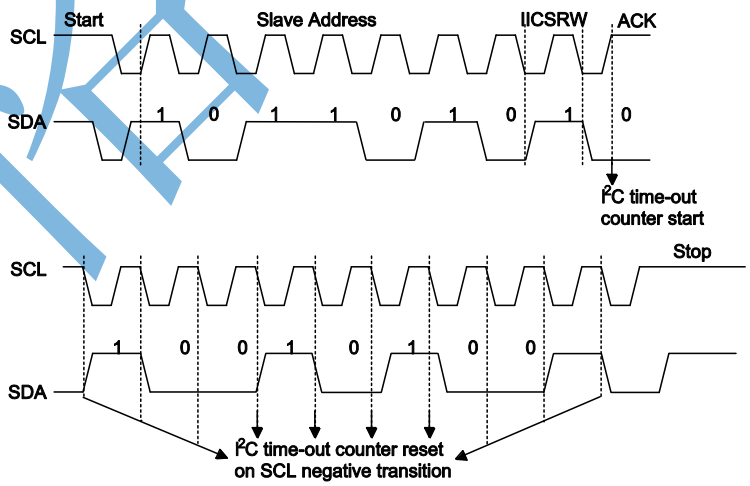


I²C 总线 ISR 流程图

2.11.9 I²C 超时控制

为了减少由于接收错误时钟源而产生的 I²C 锁定问题，系统提供了超时功能。在固定时间内如果 I²C 总线未接收到时钟源，则 I²C 电路和寄存器将会复位。

超时计数器在 I²C 总线接收到“START”信号和“地址匹配”条件时，超时计数器开始计数，并在 SCL 下降沿处清零。在下一个 SCL 下降沿来临之前，如果等待时间大于 I2CTOC 寄存器设定的超时时间，则会发生超时现象。当 I²C “STOP”条件发生时，超时计数器将停止计数。



I²C 超时时序图

当 I²C 超时计数器溢出时，计数器将停止计数，I2CTOEN 位被清零，且 I2CTOF 位被置高以表明超时计数器中断发生。超时计数器中断使用的也是 I²C 中断向量。当 I²C 超时发生时，I²C 内部电路会被复位，寄存器也将发生如下复位情况。

寄存器	I^2C 超时发生后
IICD,IICA,IICC0	保持不变
IICC1	复位至 POR

超时发生后的 I^2C 寄存器

I2CTOF 标志位可由应用程序清零。64 个超时时钟周期可由 I2CTOC 寄存器里的相应位来设置。超时时间的计算方法如下：

$((1\sim64)\times32)/f_{LIRC}$

这里给出了一个 1ms~64ms 的范围。注意，LIRC 振荡器是一直处于使能状态。

2.12 中断

中断是单片机一个重要功能。当发生外部中断或内部中断(如触控动作或定时/计数器溢出)，系统会中止当前的程序，而转到相对应的中断服务程序中。

该单片机提供两个外部中断和多个内部中断。外部中断由 INTn 引脚信号触发，而内部中断由触控按键，定时/计数器和时基等控制。

2.12.1 中断寄存器

中断控制基本上是在一定单片机条件发生时设置请求标志位，应用程序中中断使能位的设置是通过位于专用数据存储器中的一寄存器控制的。寄存器分为两类：第一类是 INTC0~INTC2 寄存器，用于设置基本的中断；第二类是一个 INTEG 寄存器，用于设置外部中断边沿触发类型。

2.12.2 中断寄存器列表

寄存器名称	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	T1F	TKMF	INT0F	T1E	TKME	INT0E	EMI
INTC1	T0F	M216CTF	M116CTF	M016CTF	T0E	M216CTE	M116CTE	M016CTE
INTC2	I2CF	M316CTF	INT1F	TBF	I2CE	M316CTE	INT1E	TBE

INTEG 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未定义，读为“0”

Bit 3~2　**INT1S1~INT1S0**：INT1 脚中断边沿控制位

00：除能

01：上升沿

10：下降沿

11：双沿

Bit 1~0　**INT0S1~INT0S0**：INT0 脚中断边沿控制位

00：除能

01：上升沿

10：下降沿

11：双沿

INTC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	T1F	TKMF	INT0F	T1E	TKME	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7　未定义，读为“0”

Bit 6 **T1F**：定时/计数器 1 中断请求标志位
0：无请求
1：中断请求
Bit 5 -
Bit 4 **INT0F**：INT0 中断请求标志位
0：无请求
1：中断请求
Bit 3 **T1E**：定时/计数器 1 中断控制位
0：除能
1：使能
Bit 2 -
Bit 1 **INT0E**：INT0 中断控制位
0：除能
1：使能
Bit 0 **EMI**：总中断控制位
0：除能
1：使能

INTC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	T0F	M216CTF	M116CTF	M016CTF	T0E	M216CTE	M116CTE	M016CTE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **T0F**：定时/计数器 0 中断请求标志位
0：无请求
1：中断请求
Bit 6 -
Bit 5 -
Bit 4 -
Bit 3 **T0E**：定时/计数器 0 中断控制位
0：除能
1：使能
Bit 2 -
Bit 1 -
Bit 0 -

INTC2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	I2CF	M316CTF	INT1F	TBF	I2CE	M316CTE	INT1E	TBE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **I2CF**：I^2C 中断请求标志位
0：无请求
1：中断请求
Bit 6 -
Bit 5 **INT1F**：INT1 中断请求标志位
0：无请求
1：中断请求
Bit 4 **TBF**：时基中断请求标志位
0：无请求
1：中断请求
Bit 3 **I2CE**：I^2C 中断控制位
0：除能
1：使能
Bit 2 -
Bit 1 **INT1E**：INT1 中断控制位
0：除能
1：使能
Bit 0 **TBE**：时基中断控制位
0：除能

1：使能

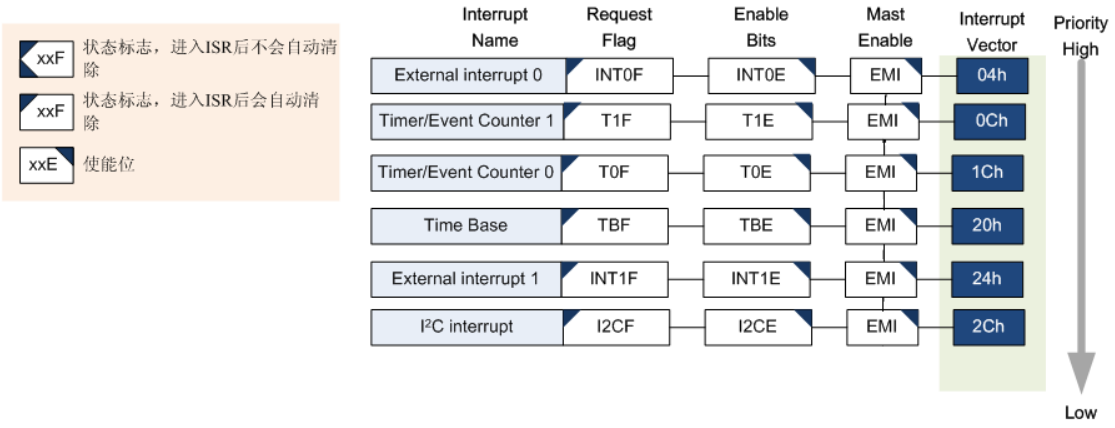
2.12.3 中断操作

若中断事件条件产生，如触控按键计数器溢出、时隙计数器溢出、定时/计数器溢出等等，相关中断请求标志将置起。中断标志产生后程序是否会跳转至相关中断向量执行是由中断使能位的条件决定的。若使能位为“1”，程序将跳至相关中断向量中执行；若使能位为“0”，即使中断请求标志置起中断也不会发生，程序也不会跳转至相关中断向量执行。若总中断使能位为“0”，所有中断都将除能。

当中断发生时，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，以继续执行原来的程序。

各个中断使能位以及相应的请求标志位，以优先级的次序显示在下图。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。其它中断请求可能发生在此期间，虽然中断不会立即响应，但是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时，有另一个中断要求立即响应，那么 EMI 位应在程序进入中断子程序后置位，以允许此中断嵌套。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。请求同时发生时，执行优先级如下流程图所示。所有被置起的中断请求标志都可把单片机从休眠或空闲模式中唤醒，若要防止唤醒动作发生，在单片机进入休眠或空闲模式前应将相应的标志置起。



中断结构

2.12.4 外部中断

通过 INTn 引脚上的信号变化可控制外部中断。当触发沿选择位设置好触发类型，INTn 引脚的状态发生变化，外部中断请求标志 INTnF 被置位时外部中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和相应中断使能位 INTnE 需先被置位。此外，必须使用 INTEG 寄存器使能外部中断功能并选择触发沿类型。外部中断引脚和普通 I/O 口共用，如果相应寄存器中的中断使能位被置位，此引脚将被作为外部中断脚使用。此时该引脚必须通过设置控制寄存器，将该引脚设置为输入口。当中断使能，堆栈未满并且外部中断脚状态改变，将调用外部中断向量子程序。当响应外部中断服务子程序时，中断请求标志位 INTnF 会自动复位且 EMI 位会被清零以除能其它中断。注意，即使此引脚被用作外部中断输入，其配置选项中的上拉电阻仍保持有效。

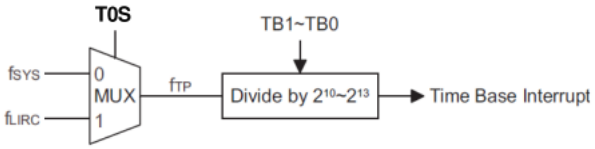
寄存器 INTEG 被用来选择有效的边沿类型，来触发外部中断。可以选择上升沿还是下降沿或双沿触发都产生外部中断。注意 INTEG 也可以用来除能外部中断功能。

2.12.5 时基中断

时基中断提供一个固定周期的中断信号，由定时器功能产生溢出信号控制。当中断请求标志 TBF 被置位时，中断请求发生。当总中断使能位 EMI 和时基使能位 TBE 被置位，允许程序跳转到各自

的中断向量地址。当中断使能，堆栈未满且时基溢出时，将调用它们各自的中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 TBF 会自动复位且 EMI 位会被清零以除能其它中断。

时基中断的目的是提供一个固定周期的中断信号，时钟源来自内部时钟源 f_{SYS} 或 f_{LIRC}。输入时钟首先经过分频器，分频率由程序设置 TBC 寄存器相关位获取合适的分频值以提供更长的时基中断周期。



时基结构

TBC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	TB1	TB0	—	—	—	—
R/W	—	—	R/W	R/W	—	—	—	—
POR	—	—	0	0	—	—	—	—

Bit 7~6　未定义，读为“0”

Bit 5~4　**TB1** ~ **TB0**：选择时基溢出周期位

00：$1024/f_{TP}$

01：$2048/f_{TP}$

10：$4096/f_{TP}$

11：$8192/f_{TP}$

Bit 3~0　未定义，读为“0”

2.12.6 定时/计数器中断

要使定时/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 TnE 必须先被置位。当定时/计数器溢出，相应的中断请求标志位 TnF 将置位并触发定时/计数器中断。中断使能，堆栈未满，当定时/计数器溢出发生中断时，将调用位于计数器中断向量处的子程序。当响应中断服务子程序时，中断请求标志位 TnF 会被自动复位且 EMI 位会被清零以除能其它中断。

2.12.7 I^2C 中断

当一个字节数据已由 I^2C 接口接收或发送完，中断请求标志 I2CF 被置位，I^2C 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI、串行接口中断使能位 I2CE 需先被置位。当中断使能，堆栈未满且一个字节数据已被传送或接收完毕时，可跳转至相关中断向量子程序中执行。当串行接口中断响应，中断请求标志位 I2CF 会被自动复位且 EMI 位会被清零以除能其它中断。

2.12.8 中断唤醒功能

每个中断都具有将处于休眠或空闲模式的单片机唤醒的能力。当中断请求标志由低到高转换时唤醒动作产生，其与中断是否使能无关。因此，尽管单片机处于休眠或空闲模式且系统振荡器停止工作，如有外部中断脚上产生外部边沿跳变，低电压改变都可能导致其相应的中断标志被置位，由此产生中断，因此必须注意避免伪唤醒情况的发生。若中断唤醒功能被除能，单片机进入休眠或空闲模式前相应中断请求标志应被置起。中断唤醒功能不受中断使能位的影响。

2.12.9 编程注意事项

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

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。

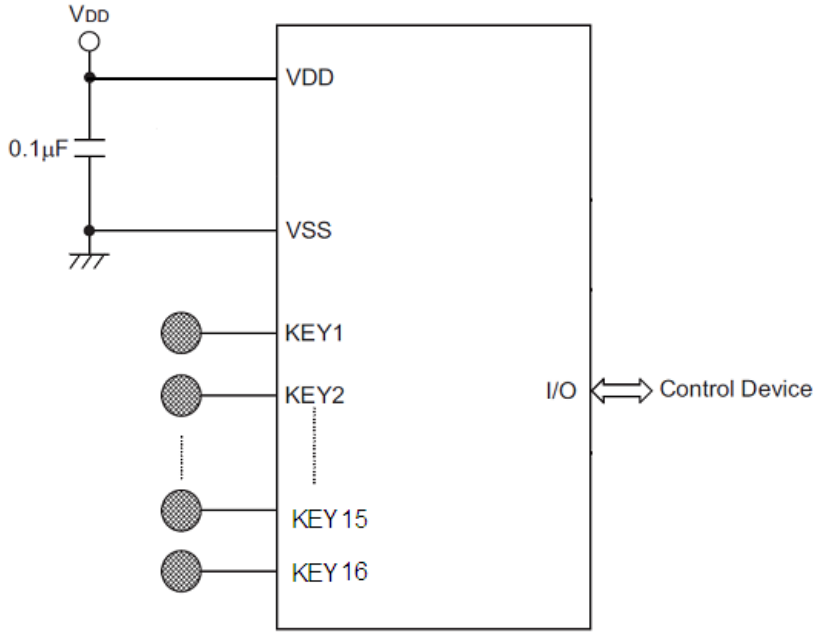
所有中断在休眠或空闲模式下都具有唤醒功能，当中断请求标志发生由低到高的转变时都可产

生唤醒功能。若要避免相应中断产生唤醒动作，在单片机进入休眠或空闲模式前需先将相应请求标志置为高。

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，如果中断服务程序会改变状态寄存器或其它的寄存器的内容而破坏控制流程，应事先将这些数据保存起来。

若从中断子程序中返回可执行 RET 或 RETI 指令。除了能返回至主程序外，RETI 指令还能自动设置 EMI 位为高，允许进一步中断。RET 指令只能返回至主程序，清除 EMI 位，除能进一步中断。

2.12.10 应用电路



第3章指令集

3.1 指令介绍

3.1.1 简介

任何单片机成功运作的核心在于它的指令集，此指令为一组程序指令码,用来指导单片机如何去执行指定的工作。在格瑞达单片机中，提供了丰富且灵活的指令集，共超过 60 条，程序设计师可以事半功倍地实现他们的应用。

为了更容易的了解各式各样的指令码，接下来按功能分组介绍它们。

3.1.2 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”。对于跳转命令必须注意，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。

3.1.3 数据的传送

单片机程序的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接收端口接收数据或传送数据到输出端口。

3.1.4 算术运算

算术运算和数据处理是大部分单片机应用所需具备的能力，在格瑞达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。

3.1.5 逻辑和移位运算

标准逻辑运算例如 AND、OR、XOR 和 CPL 全都包含在格瑞达单片机内部的指令集中，如同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。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。

3.1.6 分支和控制的转换

程序分支是采取使用 JMP 指令跳转到指定地址或使用 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 指令跳回。一个非常有用的分支指令是条件跳转，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，或是内部数据位的值。

3.1.7 位运算

提供数据存储器单一位的运算指令是格瑞达单片机的特性之一，这特性对于输出端口位的规划尤其有用，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-写出的程序现在则由位指令所取代。

3.1.8 查表运算

数据的储存通常由寄存器完成，然而当处理大量的数据时，其庞大与复杂的内容常造成对指定存储器储存上的不便，为了改善此问题，格瑞达单片机允许在程序存储器中设定一块数据可直接存取的区域，只需要一组简易的指令即可对数据进行查表。

3.1.9 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。

3.1.10 指令设定一览表

下表中说明了按功能分类的指令集，用户可以将该表作为基本的指令参考。

惯例

x：立即数
m：数据存储器地址
A：累加器
i：第0~7位
addr：程序存储器地址

助记符	说明	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、进位标志相减，结果放入 ACC	1	Z,C,AC,OV
SBCM A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	1 注	Z,C,AC,OV
DAA [m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	1 注	C
逻辑运算			
AND A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	1	Z
OR A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	1	Z
XOR A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	1	Z
ANDM A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	1 注	Z
ORM A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	1 注	Z
XORM A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	1 注	Z
AND A,x	ACC 与立即数做“与”运算，结果放入 ACC	1	Z
OR A,x	ACC 与立即数做“或”运算，结果放入 ACC	1	Z
XOR A,x	ACC 与立即数做“异或”运算，结果放入 ACC	1	Z
CPL [m]	对数据存储器取反，结果放入数据存储器	1 注	Z
CPLA [m]	对数据存储器取反，结果放入 ACC	1	Z
递增和递减			
INCA [m]	递增数据存储器，结果放入 ACC	1	Z
INC [m]	递增数据存储器，结果放入数据存储器	1 注	Z
DECA [m]	递减数据存储器，结果放入 ACC	1	Z
DEC [m]	递减数据存储器，结果放入数据存储器	1 注	Z
移位			

RRA [m]	数据存储器右移一位，结果放入 ACC	1	无
RR [m]	数据存储器右移一位，结果放入数据存储器	1 注	无
RRCA [m]	带进位将数据存储器右移一位，结果放入 ACC	1	C
RRC [m]	带进位将数据存储器右移一位，结果放入数据存储器	1 注	C
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

助记符	说明	周期	影响标志位
数据传送			
MOV A,[m]	将数据存储器送至 ACC	1	无
MOV [m],A	将 ACC 送至数据存储器	1 注	无
MOV A,x	将立即数送至 ACC	1	无
位运算			
CLR [m].i	清除数据存储器的位	1 注	无
SET [m].i	置位数据存储器的位	1 注	无
转移			
JMP addr	无条件跳转	2	无
SZ [m]	如果数据存储器为零，则跳过下一条指令	1 注	无
SZA [m]	数据存储器送至 ACC，如果内容为零，则跳过下一条指令	1 注	无
SZ [m].i	如果数据存储器的第 i 位为零，则跳过下一条指令	1 注	无
SNZ [m].i	如果数据存储器的第 i 位不为零，则跳过下一条指令	1 注	无
SIZ [m]	递增数据存储器，如果结果为零，则跳过下一条指令	1 注	无
SDZ [m]	递减数据存储器，如果结果为零，则跳过下一条指令	1 注	无
SIZA [m]	递增数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	1 注	无
SDZA [m]	递减数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	1 注	无
CALL addr	子程序调用	2	无
RET	从子程序返回	2	无
RET A,x	从子程序返回，并将立即数放入 ACC	2	无
RETI	从中断返回	2	无
查表			
TABRD [m]	读取当前页的 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]	清除数据储存器	1 注	无
SET [m]	置位数据储存器	1 注	无
CLR WDT	清除看门狗定时器	1	TO，PDF
SWAP [m]	交换数据储存器的高低字节，结果放入数据存储器	1 注	无
SWAPA [m]	交换数据储存器的高低字节，结果放入 ACC	1	无
HALT	进入空闲/休眠模式	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，如果没有跳转发生，则只需一个周期即可。
2. 任何指令若要改变PCL的内容将需要2个周期来执行。

3.2 指令定义

3.2.1 ADC A, [m] Add Data Memory to ACC with Carry

指令说明 将指定数据存储器、累加器和进位标志位的内容相加后，把结果储存回累加器。
功能表示 ACC ← ACC + [m] + C
影响标志位 OV , Z , AC , C

3.2.2 ADCM A, [m] Add ACC to Data Memory with Carry

指令说明 将指定数据存储器、累加器和进位标志位的内容相加后，把结果储存回指定数据存储器。
功能表示 [m] ←ACC + [m] + C
影响标志位 OV , Z , AC , C

3.2.3 ADD A, [m] Add Data Memory to ACC

指令说明 将指定数据存储器和累加器的内容相加后，把结果储存回累加器。
功能表示 ACC ← ACC + [m]
影响标志位 OV , Z , AC , C

3.2.4 ADD A, x Add immediate data to ACC

指令说明 将累加器和立即数的内容相加后，把结果储存回累加器。
功能表示 ACC ← ACC + x
影响标志位 OV , Z , AC , C

3.2.5 ADDM A, [m] Add ACC to Data Memory

指令说明 将指定数据存储器和累加器的内容相加后，把结果储存回指定数据存储器。
功能表示 [m] ←ACC + [m]
影响标志位 OV , Z , AC , C

3.2.6 AND A, [m] Logical AND Data Memory to ACC

指令说明 将存在累加器和指定数据存储器中的数据作AND的运算，然后把结果储存回累加器。
功能表示 ACC ← ACC“AND”[m]
影响标志位 Z

3.2.7 AND A, x Logical AND immediate data to ACC

指令说明 将存在累加器中的数据和立即数作AND的运算，然后把结果储存回累加器。
功能表示 ACC ← ACC“AND”x
影响标志位 Z

3.2.8 ANDM A, [m] Logical AND ACC to Data Memory

指令说明 将存在指定数据存储器和累加器中的数据作AND的运算，然后把结果储存回数据存储器。
功能表示 [m] ← ACC“AND”[m]
影响标志位 Z

3.2.9 CALL addr Subroutine call

指令说明 无条件地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继续执行程序，由于此指令需要额外的运算，所以为一个2周期的指令。

功能表示 Stack ← Program Counter + 1
Program Counter ← addr

影响标志位 None

3.2.10 CLR [m] Clear Data Memory

指令说明 指定数据存储器中的每一位均清除为0。

功能表示 [m] ← 00H

影响标志位 None

3.2.11 CLR [m].i Clear bit of Data Memory

指令说明 指定数据存储器中的i位清除为0。

功能表示 [m].i ← 0

影响标志位 None

3.2.12 CLR WDT Clear Watchdog Timer

指令说明 将TO、PDF 标志位和WDT全都清零。

功能表示 WDT cleared
TO ← 0
PDF ← 0

影响标志位 TO , PDF

3.2.13 CPL [m] Complement Data Memory

指令说明 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0变1。

功能表示 [m] ← $\overline{[m]}$

影响标志位 Z

3.2.14 CPLA [m] Complement Data Memory with result in ACC

指令说明 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0变1，而结果被储存回累加器且数据存储器中的内容不变。

功能表示 ACC←$\overline{[m]}$

影响标志位 Z

3.2.15 DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

指令说明 将存在累加器中的内容数值转换为BCD(二进制转成十进制)数值，如果低4位大于9或AC标志位被置位，则在低4位加上一个6，不然低4位的内容不变，如果高4位大于9或C标志位被置位，则在高4位加上一个6，十进制的转换主要是依照累加器和标志位状况，分别加上00H、06H、60H或66H，只有C标志位也许会被此指令影响，它会指出原始BCD数是否大于100，并可以进行双精度十进制数相加。

功能表示 [m] ← ACC + 00H 或
[m] ← ACC + 06H 或
[m] ← ACC + 60H 或
[m] ← ACC + 66H

影响标志位　　C

3.2.16 DEC [m]　　Decrement Data Memory

指令说明　　将在指定数据存储器内的数据减1。
功能表示　　[m] ← [m] – 1
影响标志位　　Z

3.2.17 DECA [m]　　Decrement Data Memory with result in ACC

指令说明　　将在指定数据存储器内的数据减1，把结果储存回累加器且数据存储器中的内容不变。
功能表示　　ACC ← [m] – 1
影响标志位　　Z

3.2.18 HALT　　Enter power down mode

指令说明　　此指令停止程序的执行并且关闭系统时钟，但数据存储器和寄存器的内容仍被保留，WDT和预分频器(Prescaler)被清零，暂停标志位PDF被置位且WDT溢出标志位TO 被清零。
功能表示　　TO ← 0
PDF ← 1
影响标志位　　TO , PDF

3.2.19 INC [m]　　Increment Data Memory

指令说明　　将指定数据存储器内的数据加1。
功能表示　　[m] ← [m] + 1
影响标志位　　Z

3.2.20 INCA [m]　　Increment Data Memory with result in ACC

指令说明　　将指定数据存储器内的数据加1，把结果储存回累加器且数据存储器中的内容不变。
功能表示　　ACC ← [m] + 1
影响标志位　　Z

3.2.21 JMP addr　　Jump unconditionally

指令说明　　程序计数器的内容被指定地址所取代，程序由新地址继续执行，当新地址被加载入时，必须插入一个空指令周期，所以此指令为2个周期的指令
功能表示　　Program Counter ← addr
影响标志位　　None

3.2.22 MOV A, [m]　　Move Data Memory to ACC

指令说明　　将指定数据存储器的内容复制到累加器中。
功能表示　　ACC← [m]
影响标志位　　None

3.2.23 MOV A, x　　Move immediate data to ACC

指令说明　　将立即数载入至累加器中。
功能表示　　ACC ← x

影响标志位 None

3.2.24 MOV [m], A Move ACC to Data Memory

指令说明 将累加器的内容复制到指定数据存储器。
功能表示 [m] ← ACC
影响标志位 None

3.2.25 NOP No operation

指令说明 空操作，接下来顺序执行下一条指令。
功能表示 No operation
影响标志位 None

3.2.26 OR A, [m] Logical OR Data Memory to ACC

指令说明 将存在累加器和指定数据存储器中的数据作OR的运算，然后把结果储存回累加器。
功能表示 ACC □ ACC“OR”[m]
影响标志位 Z

3.2.27 OR A, x Logical OR immediate data to ACC

指令说明 将存在累加器中的数据和立即数作OR的运算，然后把结果储存回累加器。
功能表示 ACC ← ACC“OR”x
影响标志位 Z

3.2.28 ORM A, [m] Logical OR ACC to Data Memory

指令说明 将存在指定数据存储器和累加器中的数据作OR的运算，然后把结果储存回数据存储器。
功能表示 [m] ← ACC“OR”[m]
影响标志位 Z

3.2.29 RET Return from subroutine

指令说明 将堆栈区的数据取回至程序计数器，程序由取回的地址继续执行。
功能表示 Program Counter←Stack
影响标志位 None

3.2.30 RET A, x Return from subroutine and load immediate data to ACC

指令说明 将堆栈区的数据取回至程序计数器且累加器载入立即数，程序由取回的地址继续执行。
功能表示 Program Counter ← Stack
ACC←x
影响标志位 None

3.2.31 RETI Return from interrupt

指令说明 将堆栈区的数据取回至程序计数器且中断功能通过EMI位重新被使能，EMI是控制中断使能的主中断位(寄存器INTC的第0位)，如果在执行RETI指令之前还有中断未被响应，则这个中断将在返回主程序之前被响应。
功能表示 Program Counter ←Stack

EMI ← 1
影响标志位 None

3.2.32 RL [m] Rotate Data Memory left

指令说明 将指定数据存储器的内容向左移1个位，且第7位移回第0 位。
功能表示 [m].(i+1) ← [m].i ; (i = 0~6)
[m].0 ← [m].7
影响标志位 None

3.2.33 RLA [m] Rotate Data Memory left with result in ACC

指令说明 将指定数据存储器的内容向左移1个位，且第7位移回第0位，而移位的结果储存回累加器且数据存储器中的内容不变。
功能表示 ACC.(i+1) ← [m].i ; (i = 0~6)
ACC.0 ←[m].7
影响标志位 None

3.2.34 RLC [m] Rotate Data Memory Left through Carry

指令说明 将指定数据存储器的内容连同进位标志位向左移1个位，第7位取代进位位且原本的进位标志位移至第0位。
功能表示 [m].(i+1) ← [m].i ; (i = 0~6)
[m].0 ← C
C ← [m].7
影响标志位 C

3.2.35 RLCA [m] Rotate Data Memory left through Carry with result in ACC

指令说明 将指定数据存储器的内容连同进位标志位向左移1个位，第7位取代进位位且原本的进位标志位移至第0位，而移位的结果储存回累加器且数据存储器中的内容不变。
功能表示 ACC.(i+1) ← [m].i ; (i = 0~6)
ACC.0 ← C
C ← [m].7
影响标志位 C

3.2.36 RR [m] Rotate Data Memory right

指令说明 将指定数据存储器的内容向右移1个位，且第0位移回第7位。
功能表示 [m].i ← [m].(i+1) ; (i = 0~6)
[m].7 ← [m].0
影响标志位 None

3.2.37 RRA [m] Rotate Data Memory right with result in ACC

指令说明 将指定数据存储器的内容向右移1个位，且第0位移回第7位，而移位的结果储存回累加器且数据存储器中的内容不变。
功能表示 ACC.i ← [m].(i+1) ; (i = 0~6)
ACC.7 ← [m].0
影响标志位 None

3.2.38 RRC [m] Rotate Data Memory right through Carry

指令说明 将指定数据存储器的内容连同进位标志位向右移1个位，第0位取代进位位且原本的进位标志位移至第7位。

功能表示 [m].i ← [m].(i+1) ; (i = 0~6)
[m].7← C
C ← [m].0

影响标志位 C

3.2.39 RRCA [m] Rotate Data Memory right through Carry with result in ACC

指令说明 将指定数据存储器的内容连同进位标志位向右移1个位，第0位取代进位位且原本的进位标志位移至第7位，而移位的结果储存回累加器且数据存储器中的内容不变。

功能表示 ACC.i ← [m].(i+1) ; (i = 0~6)
ACC.7 ← C
C ← [m].0

影响标志位 C

3.2.40 SBC A, [m] Subtract Data Memory from ACC with Carry

指令说明 将累加器中的数据与指定数据存储器内容和进位标志位的反相减，把结果储存回累加器。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。

功能表示 ACC ← ACC – [m] – $\overline{C}$

影响标志位 OV , Z , AC , C

3.2.41 SBCM A, [m] Subtract Data Memory from ACC with Carry and result in Data Memory

指令说明 将累加器中的数据与指定数据存储器内容和进位标志位的反相减，把结果储存回数据存储器。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。

功能表示 [m] ← ACC – [m] – $\overline{C}$

影响标志位 OV , Z , AC , C

3.2.42 SDZ [m] Skip if Decrement Data Memory is 0

指令说明 将指定数据存储器的内容先减去1后，如果结果为0，则程序计数器再加1跳过下一条指令，由于取得下一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，则程序继续执行下面的指令。

功能表示 [m] ← [m] – 1
Skip if [m] = 0

影响标志位 None

3.2.43 SDZA [m] Skip if decrement Data Memory is zero with result in ACC

指令说明 将指定数据存储器的内容先减去1后，如果结果为0，则程序计数器再加1 跳过下一条指令，此结果会被储存回累加器且指定数据存储器中的内容不变，由于取得下一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，则程序继续执行下面的指令。

功能表示 ACC ← [m] – 1
Skip if ACC = 0

影响标志位 None

3.2.44 SET [m] Set Data Memory

指令说明 将指定数据存储器的每一个位置位为1。
功能表示 [m] ← FFH
影响标志位 None

3.2.45 SET [m].i Set bit of Data Memory

指令说明 将指定数据存储器的第i位置位为1。
功能表示 [m].i ← 1
影响标志位 None

3.2.46 SIZ [m] Skip if increment Data Memory is 0

指令说明 将指定数据存储器的内容先加上1后，如果结果为0，则程序计数器再加1 跳过下一条指令，由于取得下一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，则程序继续执行下面的指令。
功能表示 [m] ← [m] + 1
Skip if [m] = 0
影响标志位 None

3.2.47 SIZA [m] Skip if increment Data Memory is zero with result in ACC

指令说明 将指定数据存储器的内容先加上1后，如果结果为0，则程序计数器再加1跳过下一条指令，此结果会被储存回累加器且指定数据存储器中的内容不变，由于取得下一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，则程序继续执行下面的指令。
功能表示 ACC← [m] + 1
Skip if ACC = 0
影响标志位 None

3.2.48 SNZ [m].i Skip if bit i of Data Memory is not 0

指令说明 如果指定数据存储器的第i位不为0，则程序计数器再加1跳过下一条指令，由于取得下一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，程序继续执行下面的指令。
功能表示 Skip if [m].i ≠ 0
影响标志位 None

3.2.49 SUB A, [m] Subtract Data Memory from ACC

指令说明 将累加器中内容减去指定数据存储器的数据，把结果储存回累加器。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。
功能表示 ACC ← ACC – [m]
影响标志位 OV , Z , AC , C

3.2.50 SUBM A, [m] Subtract Data Memory from ACC with result in Data Memory

指令说明 将累加器中内容减去指定数据存储器的数据，把结果储存回数据存储器。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。

功能表示　[m] ← ACC – [m]
影响标志位　OV , Z , AC , C

3.2.51 SUB A, x　Subtract immediate Data from ACC

指令说明　将累加器中内容减去立即数，把结果储存回累加器。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。
功能表示　ACC ← ACC – x
影响标志位　OV , Z , AC , C

3.2.52 SWAP [m]　Swap nibbles of Data Memory

指令说明　将指定数据存储器的低4位与高4位互相交换。
功能表示　[m].3 ~ [m].0 ↔ [m].7 ~ [m].4
影响标志位　None

3.2.53 SWAPA [m]　Swap nibbles of Data Memory with result in ACC

指令说明　将指定数据存储器的低4位与高4位互相交换，然后把结果储存回累加器且数据存储器的内容不变。
功能表示　ACC.3 ~ ACC.0 ← [m].7 ~ [m].4
ACC.7 ~ ACC.4 ← [m].3 ~ [m].0
影响标志位　None

3.2.54 SZ [m]　Skip if Data Memory is 0

指令说明　如果指定数据存储器的内容为0，则程序计数器再加1跳过下一条指令，由于取得下一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，程序继续执行下面的指令。
功能表示　Skip if [m] = 0
影响标志位　None

3.2.55 SZA [m]　Skip if Data Memory is 0 with data movement to ACC

指令说明　将指定数据存储器的内容复制到累加器，如果值为0，则程序计数器再加1跳过下一条指令，由于取得下一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，程序继续执行下面的指令。
功能表示　ACC ← [m]
Skip if [m] = 0
影响标志位　None

3.2.56 SZ [m].i　Skip if bit i of Data Memory is 0

指令说明　如果指定数据存储器第i 位为0，则程序计数器再加1 跳过下一条指令，由于取得下一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，程序继续执行下面的指令。
功能表示　Skip if [m].i = 0
影响标志位　None

3.2.57 TABRD [m]　Read table (current page) to TBLH and Data Memory

指令说明　将表格指针TBLP所指的程序代码低字节(当前页)移至指定数据存储器且将高字节移至TBLH。
功能表示　[m] ← 程序代码(低字节)

TBLH ← 程序代码(高字节)
影响标志位 None

3.2.58 TABRDL [m] Read table (last page) to TBLH and Data Memory

指令说明 将表格指针TBLP所指的程序代码低字节(最后一页)移至指定数据存储器且将高字节移至TBLH。
功能表示 [m] ← 程序代码(低字节)
TBLH ← 程序代码(高字节)
影响标志位 None

3.2.59 XOR A, [m] Logical XOR Data Memory to ACC

指令说明 将存在累加器和指定数据存储器中的数据作XOR的运算，然后把结果储存回累加器。
功能表示 ACC ← ACC“XOR”[m]
影响标志位 Z

3.2.60 XORM A, [m] Logical XOR ACC to Data Memory

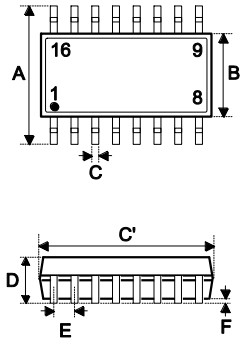
指令说明 将存在指定数据存储器和累加器中的数据作XOR的运算，然后把结果储存回数据存储器。
功能表示 [m] ← ACC“XOR”[m]
影响标志位 Z

3.2.61 XOR A, x Logical XOR immediate data to ACC

指令说明 将存在累加器中的数据和立即数作XOR的运算，然后把结果储存回累加器。
功能表示 ACC ← ACC“XOR”x
影响标志位 Z

第4章封装信息

4.1 16-pin NSOP (150mil)外形尺寸

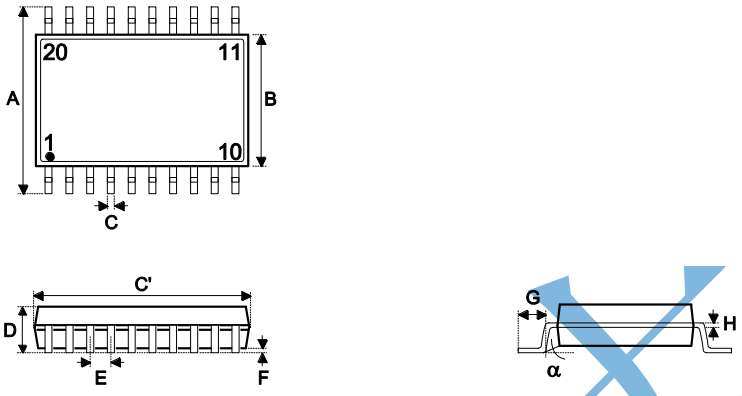


4.1.1 MS-012

符号	尺寸(单位：inch)		
	最小	正常	最大
A	0.228	—	0.244
B	0.150	—	0.157
C	0.012	—	0.020
C’	0.386	—	0.402
D	—	—	0.069
E	—	0.050	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.007	—	0.010
α	0°	—	8°

符号	尺寸(单位：mm)		
	最小	正常	最大
A	5.79	—	6.20
B	3.81	—	3.99
C	0.30	—	0.51
C’	9.80	—	10.21
D	—	—	1.75
E	—	1.27	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.18	—	0.25
α	0°	—	8°

4.2 20-pin SOP(300mil)外形尺寸

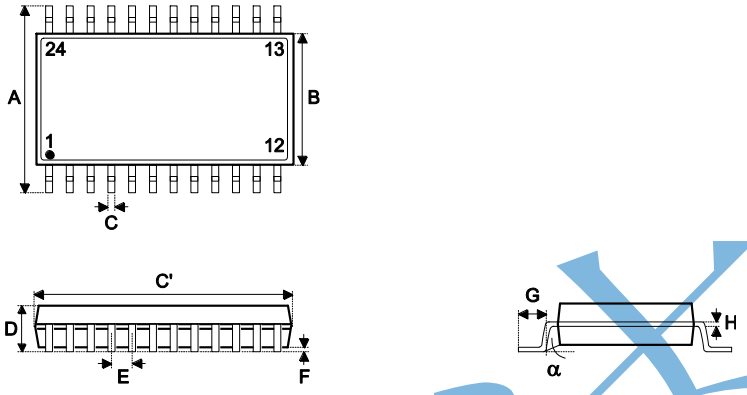


4.2.1 MS-013

符号	尺寸(单位：inch)		
	最小	正常	最大
A	0.393	—	0.419
B	0.256	—	0.300
C	0.012	—	0.020
C’	0.496	—	0.512
D	—	—	0.104
E	—	0.050	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0 °	—	8 °

符号	尺寸(单位：mm)		
	最小	正常	最大
A	9.98	—	10.64
B	6.50	—	7.62
C	0.30	—	0.51
C’	12.60	—	13.00
D	—	—	2.64
E	—	1.27	—
F	0.10	—	0.30
G	0.41	—	1.27
H	0.20	—	0.33
α	0 °	—	8 °

4.3 24-pin SOP(300mil)外形尺寸

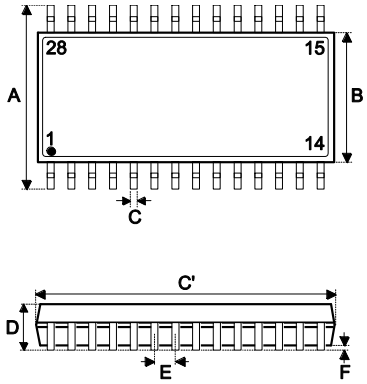


4.3.1 MS-013

符号	尺寸(单位：inch)		
	最小	正常	最大
A	0.393	—	0.419
B	0.256	—	0.300
C	0.012	—	0.020
C’	0.598	—	0.613
D	—	—	0.104
E	—	0.050	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

符号	尺寸(单位：mm)		
	最小	正常	最大
A	9.98	—	10.64
B	6.50	—	7.62
C	0.30	—	0.51
C’	15.19	—	15.57
D	—	—	2.64
E	—	1.27	—
F	0.10	—	0.30
G	0.41	—	1.27
H	0.20	—	0.33
α	0°	—	8°

4.4 28-pin SOP(300mil)外形尺寸



4.4.1 MS-013

符号	尺寸(单位：inch)		
	最小	正常	最大
A	0.393	—	0.419
B	0.256	—	0.300
C	0.012	—	0.020
C’	0.697	—	0.713
D	—	—	0.104
E	—	0.050	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0 °	—	8 °

符号	尺寸(单位：mm)		
	最小	正常	最大
A	9.98	—	10.64
B	6.50	—	7.62
C	0.30	—	0.51
C’	17.70	—	18.11
D	—	—	2.64
E	—	1.27	—
F	0.10	—	0.30
G	0.41	—	1.27
H	0.20	—	0.33
α	0 °	—	8 °

第5章　订购

5.1　订购信息

下单规格	功能简述	芯片型号	封装
BS45F1BNS16	8-Bit 触摸按键式 Flash 单片机	BS45F1B	NSOP16
BS45F1BS20	8-Bit 触摸按键式 Flash 单片机	BS45F1B	SOP20
BS45F1BS24	8-Bit 触摸按键式 Flash 单片机	BS45F1B	SOP24
BS45F1BS28	8-Bit 触摸按键式 Flash 单片机	BS45F1B	SOP28

深圳市格瑞达实业有限公司（总公司）
SHENZHEN GREENMCU TECHNOLOGY CO.,LTD.
地址：深圳市福田区彩田南路海鹰大厦 20B
电话：（86）755-83051793　　82913392
　　（86）755-82914749　　82913502
传真：（86）755-82971356
网址：www.greenmcu.com

深圳市格瑞达实业有限公司（顺德办事处）
地址：顺德区容桂镇文海西路保利百合花园 10 栋 B 单元 1901
电话：（86）757-28302691　　22909432
传真：（86）757-28302691

最新信息请登陆我们的网址：www.greenmcu.com