触控按键式 Flash 单片机

BS83A02A-4/BS83A04A-3/BS83A04A-4

版本 : V1.72　日期 : 2021-11-10

www.holtek.com

目录

特性

CPU 特性

- 工作电压：
  - ♦ f_{SYS} = 8MHz：2.7V~5.5V (BS83A04A-3)
  - ♦ f_{SYS} = 8MHz：2.2V~5.5V (BS83A02A-4/BS83A04A-4)
- V_{DD}=5V，系统时钟为 8MHz 时，指令周期为 0.5μs
- 提供暂停和唤醒功能，以降低功耗
- 内部集成高 / 低速振荡器
  - ♦ 低速 -- 32kHz
  - ♦ 高速 -- 8MHz
- 多种工作模式：正常模式，低速模式，空闲模式和休眠模式
- 内部集成 8MHz 振荡器，无需外接元件
- 所有指令都可在 1 个或 2 个指令周期内完成
- 查表指令
- 61 条功能强大的指令系统
- 多达 4 层硬件堆栈
- 位操作指令

周边特性

- Flash 程序存储器：1K×16
- 数据存储器：96×8
- 看门狗定时器功能
- 多达 8 个双向 I/O 口
- 一个与 I/O 口复用的外部中断输入
- 一个 8 位可编程定时 / 计数器
- 一个时基功能，用于产生固定时间的中断信号
- 低电压复位功能
- 多达 4 个触控按键
- 封装类型：6-pin DFN，SOT23-6，8-pin SOP，10-pin MSOP

开发工具

为加快产品开发并简化单片机参数设置，Holtekt 提供相关开发工具，用户可通过以下链接下载：

https://www.holtek.com.cn/esk-bs-210 (仅 BS83A02A-4/A04A-3 支持)

概述

该系列单片机是一款 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。此系列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。该系列单片机除了 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。

该系列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。I/O 灵活、8-bit 定时器和其它特性增强了该系列单片机的功能和灵活性。

该系列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。

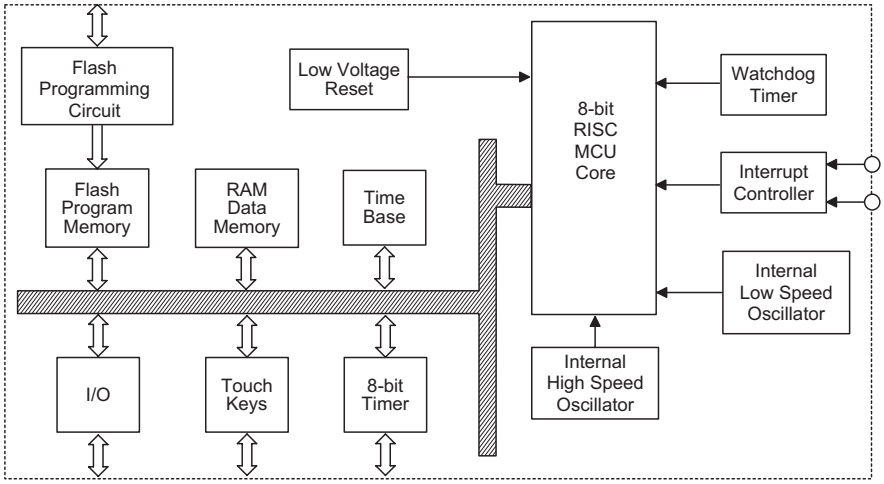
选型表

该系列单片机的大多特性都相同，他们的主要不同之处在于工作电压范围，输入 / 输出引脚个数，触控按键数，LVR 电压值和封装类型。以下表格概述了每款单片机的主要特性。

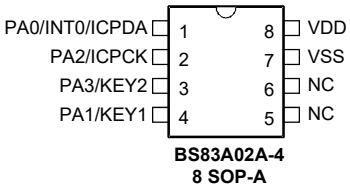
型号	内部时钟	V_{DD}	系统时钟	程序存储器	数据存储器	I/O	外部中断
BS83A02A-4	8MHz	2.2V~5.5V	8MHz	1K×16	96×8	4	1
BS83A04A-3		2.7V~5.5V				8	
BS83A04A-4		2.2V~5.5V					

型号	8-bit 定时器	按键个数	LVR	堆栈	封装形式	正印
BS83A02A-4	1	2	2.10V	4	6DFN SOT23-6 8SOP	BS83A02A-4 A2A4 (for 6DFN) 02A-4 (for SOT23-6) BS83A02A-4 (for 8SOP)
BS83A04A-3		4	2.55V		8SOP 10MSOP	BS83A04A-3 83A04A-3 (for 8SOP) 8304A-3 (for 10MSOP)
BS83A04A-4			2.10V			BS83A04A-4 83A04A-4 (for 8SOP) 8304A-4 (for 10MSOP)

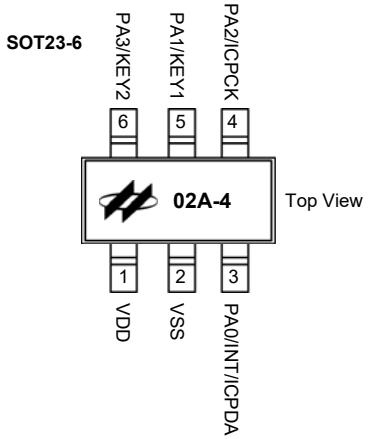
方框图

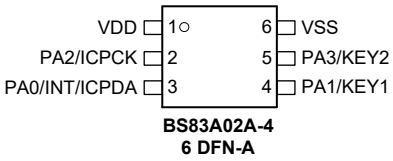


引脚图

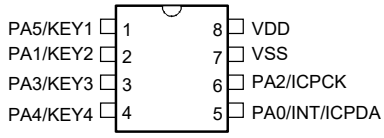


BS83A02A-4
8 SOP-A

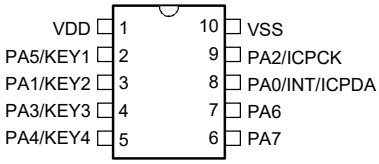




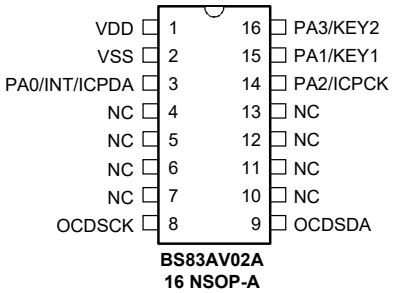
BS83A02A-4
6 DFN-A



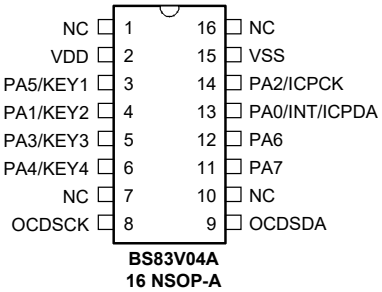
BS83A04A-3/BS83A04A-4
8 SOP-A



BS83A04A-3/BS83A04A-4
10 MSOP-A



BS83AV02A
16 NSOP-A



BS83V04A
16 NSOP-A

注：16NSOP 封装仅适用于 OCDS EV 芯片。

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。

该章节的引脚描述是针对最大封装的单片机，这些引脚并非都存在于小封装的单片机内。

BS83A02A-4

引脚名称	功能	OPT	I/T	O/T	说明
PA0/INT/ICPDA	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
	INT	INTEG	ST	—	外部中断
	ICPDA	—	ST	CMOS	ICP 数据 / 地址
PA1/KEY1	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
	KEY1	TKM0C1	NSI	—	触控按键输入口
PA2/ICPCK	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
	ICPCK	—	ST	—	ICP 时钟引脚
PA3/KEY2	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
	KEY2	TKM0C1	NSI	—	触控按键输入口
OCDSCK	OCDSCK	—	ST	—	片上调试时钟引脚，仅适应于 EV 芯片
OCDSDA	OCDSDA	—	ST	CMOS	片上调试数据 / 地址引脚，仅适应于 EV 芯片
VDD	VDD	—	PWR	—	电源电压
VSS	VSS	—	PWR	—	地

注：I/T：输入类型；　　O/T：输出类型
　　OPT：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
　　NSI：无标准输入

BS83A04A-3/BS83A04A-4

引脚名称	功能	OPT	I/T	O/T	说明
PA0/INT/ ICPDA	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
	INT	INTEG	ST	—	外部中断
	ICPDA	—	ST	CMOS	ICP 数据 / 地址
PA1/KEY2	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
	KEY2	TKM0C1	NSI	—	触控按键输入口
PA2/ICPCK	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
	ICPCK	—	ST	—	ICP 时钟引脚
PA3/KEY3	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
	KEY3	TKM0C1	NSI	—	触控按键输入口
PA4/KEY4	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
	KEY4	TKM0C1	NSI	—	触控按键输入口
PA5/KEY1	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
	KEY1	TKM0C1	NSI	—	触控按键输入口
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	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
OCDSCK	OCDSCK	—	ST	—	片上调试时钟引脚，仅适应于 EV 芯片
OCDSDA	OCDSDA	—	ST	CMOS	片上调试数据 / 地址引脚，仅适应于 EV 芯片
VDD	VDD	—	PWR	—	电源电压
VSS	VSS	—	PWR	—	地

注：I/T：输入类型；　　O/T：输出类型
OPT：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
NSI：无标准输入
对于 8-pin 封装，PA7 并未连接到外部，但内部与 PA4 连接，所以建议将 PA7 设置成输入模式并除能上拉功能。

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
储存温度 ..... -50°C~125°C
工作温度 ..... -40°C~85°C
I_{OL} 总电流..... 80mA
I_{OH} 总电流 ..... -80mA
总功耗 500mW

注：这里只强调额定功率，超过极限参数所规定的范围将对芯片造成损害，无法预期芯片在上述标示范围外的工作状态，而且若长期在标示范围外的条件下工作，可能影响芯片的可靠性。

直流电气特性

Ta=25°C

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) (BS83A04A-3)	—	f_{SYS}=8MHz	2.7	—	5.5	V
V_{DD}	工作电压 (HIRC) (BS83A02A-4/BS83A04A-4)	—	f_{SYS}=8MHz	2.2	—	5.5	V
I_{DD}	工作电流 (HIRC) (f_{SYS}=f_H, f_S=f_{SUB}= f_{LIRC})	3V	无负载，f_H=8MHz WDT 使能	—	0.8	1.5	mA
		5V		—	1.5	3.0	mA
	工作电流 (LIRC) (f_{SYS}=f_L= f_{LIRC}, f_S=f_{SUB}= f_{LIRC})	3V	无负载，WDT 使能，LVR 除能	—	10	20	μA
		5V		—	20	35	μA
I_{STB}	IDLE 模式静态电流 (HIRC) (f_{SYS}=off, f_S=f_{SUB}=f_{LIRC})	3V	无负载，WDT 使能，f_{SYS}=8MHz	—	1.5	3.0	μA
		5V		—	2.5	5.0	μA
	IDLE 模式静态电流 (LIRC) (f_{SYS}=off, f_S=f_{SUB}=f_{LIRC})	3V	无负载，WDT 使能，f_{SYS}=32KHz	—	1.5	3.0	μA
		5V		—	2.5	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 (BS83A04A-3)	—	LVR 使能，V_{LVR}=2.55V	-5%	2.55	+5%	V
	低电压复位电压 (BS83A02A-4/BS83A04A-4)	—	LVR 使能，V_{LVR}=2.10V	-5%	2.10	+5%	V
I_{LVR}	低电压复位电流	3V	LVR 除能 → 使能	—	15	25	μA
		5V		—	20	30	μA

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{OL}	输入 / 输出口灌电流 (BS83A02A-4)	3V	V_{OL}=0.1V_{DD}	8	16	—	mA
		5V	V_{OL}=0.1V_{DD}	16	32	—	mA
	输入 / 输出口灌电流 (BS83A04A-3/BS83A04A-4)	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 (BS83A02A-4)	3V	V_{OH}=0.9V_{DD}	-3.75	-7.5	—	mA
		5V	V_{OH}=0.9V_{DD}	-7.5	-15	—	mA
	输入 / 输出口源电流 (BS83A04A-3/BS83A04A-4)	3V	V_{OH}=0.9V_{DD}	-2	-4	—	mA
		5V	V_{OH}=0.9V_{DD}	-5	-10	—	mA
R_{PH}	上拉电阻 (输入 / 输出口)	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

交流电气特性

Ta=25°C

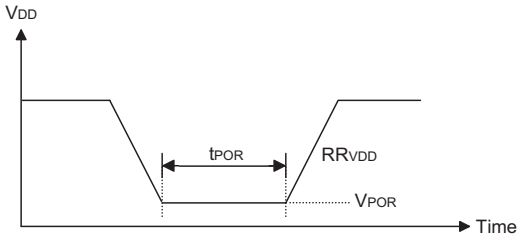
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{SYS}	系统时钟 (BS83A04A-3)	—	2.7V~5.5V	DC	—	8	MHz
	系统时钟 (BS83A02A-4/BS83A04A-4)	—	2.2V~5.5V	DC	—	8	MHz
f_{HIRC}	系统时钟 (HIRC)	3V/5V	Ta=25°C	-2%	8	+2%	MHz
f_{LIRC}	系统时钟 (LIRC)	5V	—	-10%	32	+10%	kHz
		2.2V~5.5V	Ta=-40°C~85°C	-50%	32	+60%	kHz
t_{INT}	中断脉冲宽度	—	—	10	—	—	t_{SYS}
t_{LVR}	低电压复位宽度	—	—	120	240	480	μs
t_{SST}	系统启动延时周期 (从 HALT 模式下唤醒)	—	f_{SYS}=HIRC	—	1024	1025	t_{SYS}
			f_{SYS}=LIRC	—	1~2	3	

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。

上电复位特性

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

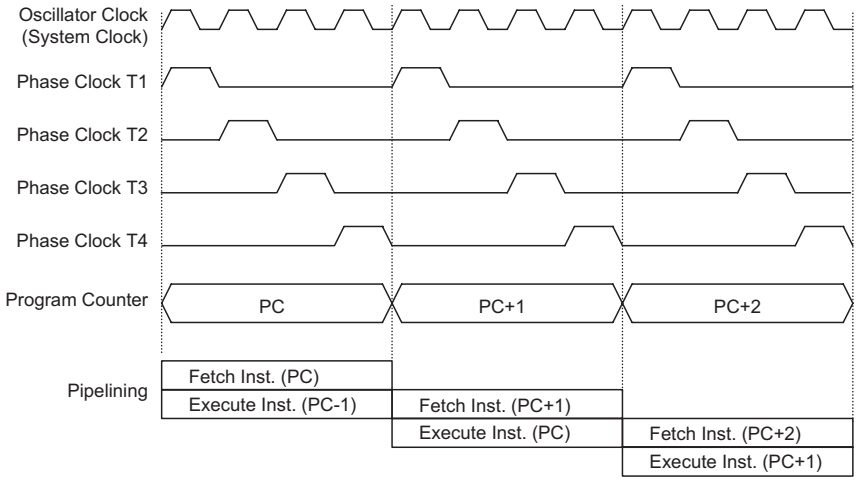


系统结构

内部系统结构是 Holek 单片机具有良好性能的主要因素。由于采用 RISC 结构，该系列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，确保了在提供具有较大可靠度和灵活性的实用性 I/O 时，仅需要少数的外部器件。这些特性使得该系列单片机适用于低成本，大批量生产的控制应用。

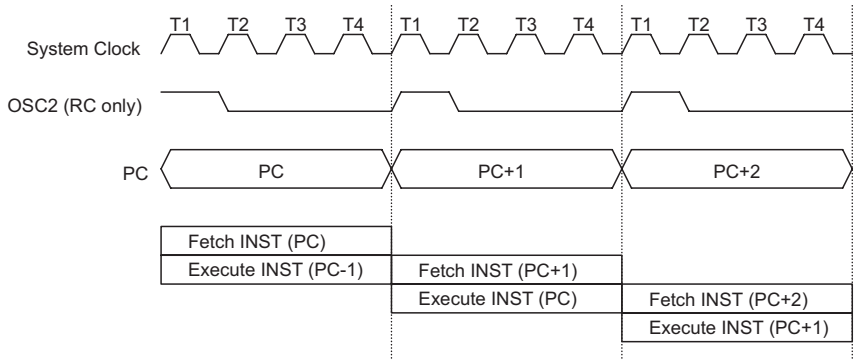
时序和流水线结构

主系统时钟由 HIRC 或 LIRC 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

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。然而只有较低的 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。

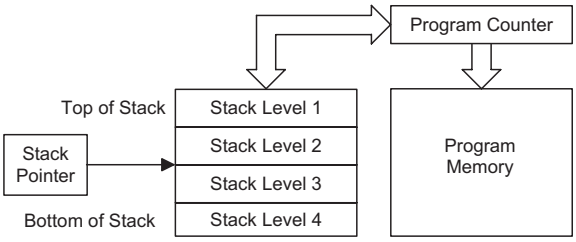
程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

程序计数器	
程序计数器高字节	PCL 寄存器
PC9~PC8	PCL7~PCL0

程序计数器

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。该系列单片机含有 4 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。

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

Flash 程序存储器

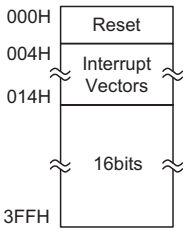
程序存储器用来存放用户代码即存储程序。该系列单片机提供可多次编程的 Flash 存储器，用户可以很方便的在同一个芯片中修改他们的应用代码。通过使用合适的编程工具，该 Flash 型单片机提供用户以灵活的方式自由开发他们的应用，这对于需要除错或需要经常升级和改变程序的产品是很有帮助的。

结构

程序存储器的容量为 1K×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。

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。



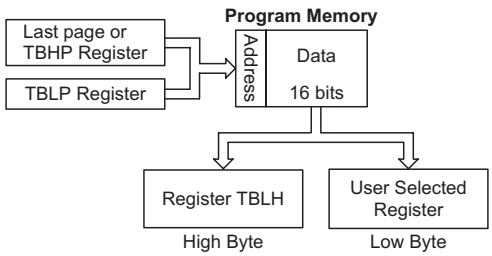
Flash 程序存储器结构

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。

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：



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“300H”，即 1K 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“306H”，即最后一页起始地址后的第 6 个地址。注意，假如“TABRDL [m]”指令被使用，则表格指针指向最后一页的起始地址。在这个例子中，表格数据的高字节等于零，而当“TABRDL [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。

指令	表格地址									
	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRD [m]	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	@7	@6	@5	@4	@3	@2	@1	@0

表格地址

注：PC9~PC8：当前程序计数器位
@7~@0：表格指针 TBLP 位

表格读取程序范例

```
tempreg1 db ?       ; temporary register #1
tempreg2 db ?       ; temporary register #2
:
mov a,06h           ; initialise low table pointer - note that this address
                    ; is referenced to the last page
mov tblp,a
:
:
tabrdl tempreg1     ; transfers value in table referenced by table pointer
                    ; to tempreg1 data at program memory
                    ; address “306H” transferred to tempreg1 and TBLH
dec tblp            ; reduce value of table pointer by one
tabrdl tempreg2     ; transfers value in table referenced by table pointer
                    ; to tempreg2 data at program memory address “305H”
                    ; transferred to tempreg2 and TBLH in this
                    ; example the data “1AH” is transferred to tempreg1 and
                    ; data “0FH” to register tempreg2
                    ; the value “00H” will be transferred to the high byte
                    ; register TBLH
:
:
org 300h            ; sets initial address of program memory
dc  00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
```

在线烧录

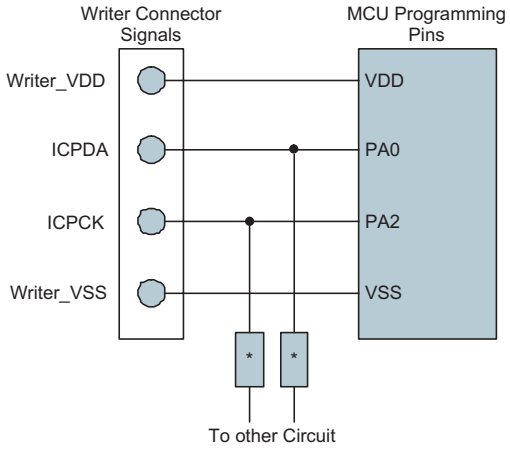
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。

另外，Holek 单片机提供 4 线接口的在线编程方式。用户可将进行过编程或未经过编程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。

Holtek 烧录引脚名称	引脚名称	功能
ICPDA	PA0	串行数据输入 / 输出 – 读 / 写
ICPCK	PA2	地址和数据串行时钟输入
VDD	VDD	电源 (5.0V)
VSS	VSS	地

芯片内部程序存储器可以通过 4 线的接口在线进行编程。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。

在编程过程中，编程器会控制 PA0 和 PA2 脚进行数据和时钟编程，用户必须确保这两个引脚没有连接至其它输出脚。



在线编程接口

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

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 Holtek 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“Holtek e-Link for 8-bit MCU OCDS User's Guide”文件。

Holtek e-Link 引脚名称	EV 芯片引脚名称	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

数据存储器

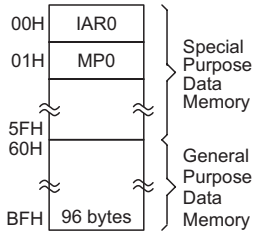
数据存储器是内容可以更改的 8 位 RAM 内部存储器，用来存储临时数据。

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。

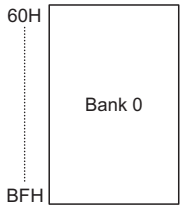
所有单片机的数据存储器的起始地址都是“00H”。



数据存储器

00H	IAR0	30H	Unused
01H	MP0	31H	Unused
02H	IAR1	32H	Unused
03H	MP1	33H	Unused
04H	Unused	34H	Unused
05H	ACC	35H	Unused
06H	PCL	36H	Unused
07H	TBLP	37H	Unused
08H	TBLH	38H	Unused
09H	TBHP	39H	Unused
0AH	STATUS	3AH	Unused
0BH	SMOD	3BH	Unused
0CH	CTRL	3CH	Unused
0DH	INTEG	3DH	Unused
0EH	INTC0	3EH	Unused
0FH	INTC1	3FH	Unused
10H	Unused	40H	Unused
11H	Unused	41H	Unused
12H	Unused	42H	Unused
13H	LVRC	43H	TKTMR
14H	PA	44H	TKC0
15H	PAC	45H	TK16DL
16H	PAPU	46H	TK16DH
17H	PAWU	47H	TKC1
18H	Unused	48H	TKM016DL
19H	Unused	49H	TKM016DH
1AH	WDTC	4AH	TKM0ROL
1BH	TBC	4BH	TKM0ROH
1CH	TMR	4CH	TKM0C0
1DH	TMRC	4DH	TKM0C1
1EH	Unused	4EH	Unused
1FH	Unused	4FH	Unused
20H	Unused	50H	Unused
21H	Unused	51H	Unused
22H	Unused	52H	Unused
23H	Unused	53H	Unused
24H	Unused	54H	Unused
25H	Unused	55H	Unused
26H	Unused	56H	Unused
27H	Unused	57H	Unused
28H	Unused	58H	Unused
29H	Unused	59H	Unused
2AH	Unused	5AH	Unused
2BH	Unused	5BH	Unused
2CH	Unused	5CH	Unused
2DH	Unused	5DH	Unused
2EH	Unused	5EH	Unused
2FH	Unused	5FH	Unused

特殊功能数据存储器



通用数据存储器

特殊功能寄存器描述

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。

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。

存储器指针 – MP0， MP1

该系列单片机提供两个存储器指针，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存储器指针所指定的地址。MP0 和 IAR0 用于访问 Bank 0，而 MP1 和 IAR1 可访问所有的 Bank。注意，对于该系列单片机，存储器指针 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。

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。

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。

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。

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。

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，或低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”：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。

系统控制寄存器 – CTRL

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

“x”：未知

Bit 7　**FSYSON**：f_{SYS} 在空闲模式下的控制位
0：除能
1：使能

Bit 6~3　未使用，读为“0”

Bit 2　**LVRF**：LVR 复位标志位
0：无效
1：有效
该位可以被清为“0”，但不能设为“1”。

Bit 1　**LRF**：LVRC 控制的复位标志位
0：无效
1：有效
该位可以被清为“0”，但不能设为“1”。

Bit 0　**WRF**：由设置 WE[4:0] 引起的复位
0：无效
1：有效
该位可以被清为“0”，但不能设为“1”。

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较佳的优化。振荡器选项是通过寄存器来完成的。

振荡器概述

该系列单片机有两个内部振荡器，一个低速振荡器和一个高速振荡器。它们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。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
内部高速 RC	HIRC	8MHz
内部低速 RC	LIRC	32kHz

振荡器类型

系统时钟配置

该系列单片机有两个方式产生系统时钟，一个高速内部时钟源和一个低速内部时钟源。高速振荡器为内部 8MHz RC 振荡器，低速振荡器为内部 32kHz RC 振荡器。这两个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。

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 振荡器上电时的频率固定为 8 MHz。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较大程度地降低。

内部低速 RC 振荡器 – LIRC

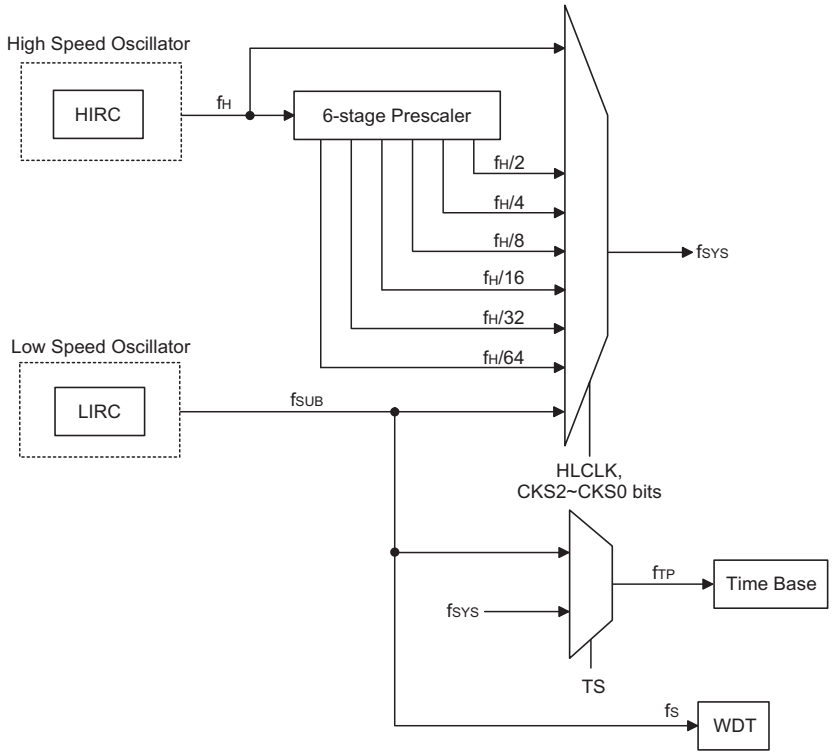
内部 32kHz 系统振荡器为低速振荡器。LIRC 是一个全集成的 RC 振荡器，无需外接器件，在常温 5V 条件下，振荡频率值为 32kHz。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因 V_{DD}、温度以及芯片制成工艺不同的影响较大程度地降低。系统上电，LIRC 振荡器就使能， 不存在将该振荡器除能的寄存器位。

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。此系列单片机提供高、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较佳性能 / 功耗比。

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_{SUB}，通过 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位进行选择。高频时钟和低频时钟都来自内部 RC 振荡器。



系统时钟配置

注：当系统时钟源 f_{SYS} 由 f_H 到 f_{SUB} 转换时，高速振荡器将停止以节省耗电。因此，没有为外围电路提供 f_H~f_H/64 的频率。

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	—	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	—	R	R	R/W	R/W
POR	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_{SUB} (f_{LIRC})
001：f_{SUB} (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 未使用，读为“0”

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。当系统处于 SLEEP 模式时，该标志为低。若系统时钟来自 LIRC 振荡器，该位转换为高需 1~2 个时钟周期。

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，1024 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_{SUB}
1：f_H
此位用于选择 f_H 或 $f_H/2$~ $f_H/64$ 还是 f_{SUB} 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_{SUB} 作为系统时钟。当系统时钟由 f_H 时钟向 f_{SUB} 时钟转换时，f_H 将自动关闭以降低功耗。

系统工作模式

该系列单片机有 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_{SUB}	f_S
正常模式	On	f_H~f_H/64	On	On
低速模式	On	f_{SUB}	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	On

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_{SUB} 和 f_S 时钟继续运行，因为看门狗定时器功能一直使能且其时钟来自于 f_{SUB}。

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，但一些外围功能如看门狗定时器和定时 / 计数器将继续工作。

空闲模式 1

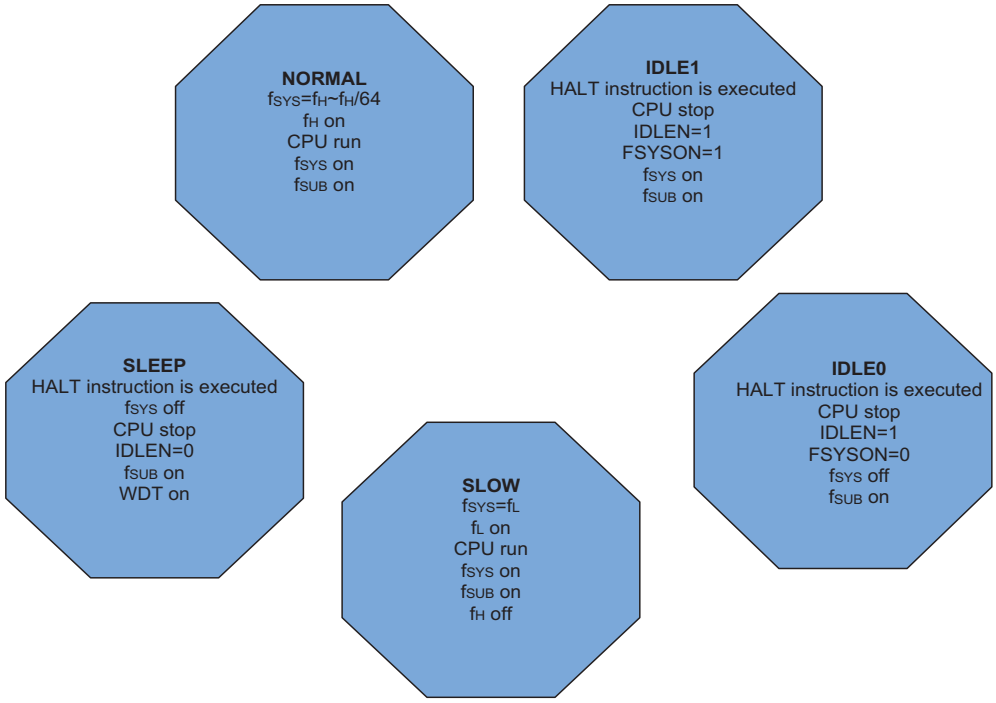
执行 HALT 指令后且 SMOD 寄存器中 IDLEN 位为高，CTRL 寄存器中 FSYSON 位为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，但会提供一个时钟源给一些外围功能如看门狗定时器和定时 / 计数器。在空闲模式 1 中，系统振荡器继续运行，该系统振荡器可以为高速或低速系统振荡器。在该模式中看门狗定时器时钟 f_S 开启。

工作模式切换

该系列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较佳的性能 / 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。

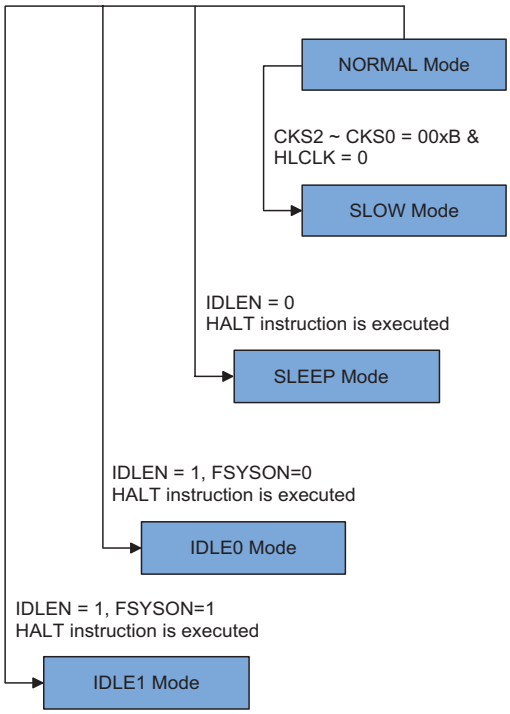
当 HLCLK 位变为低电平时，时钟源将由高速时钟源 f_H 转换成时钟源 $f_H/2$~$f_H/64$ 或 f_{SUB}。若时钟源来自 f_{SUB}，高速时钟源将停止运行以节省耗电。此时须注意，$f_H/16$ 和 $f_H/64$ 内部时钟源也将停止运行。所附流程图显示了该系列单片机在不同工作模式间切换时的变化。



正常模式切换到低速模式

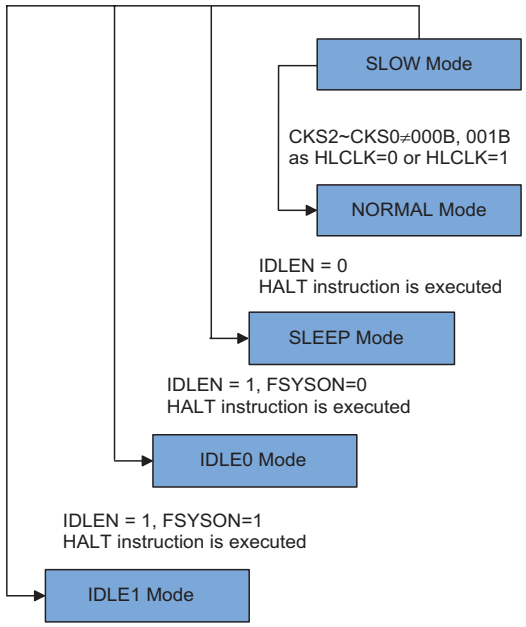
系统运行在正常模式时使用高速系统振荡器，因此较为耗电。可通过设置 SMOD 寄存器中的 HLCLK 位为“0”及 CKS2~CKS0 位为“000”或“001”使系统时钟切换至运行在低速模式下。此时将使用低速系统振荡器以节省耗电。用户可在对性能要求不高的操作中使用此方法以减少耗电。

低速模式的时钟源来自 LIRC 振荡器，因此要求该振荡器在所有模式切换动作发生前稳定下来。该动作由 SMOD 寄存器中 LTO 位控制。



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”且 WDT 功能使能。在上述条件下执行该指令后，将发生的情况如下：

- 系统时钟和时基时钟停止运行，应用程序停止在“HALT”指令处，WDT 继续运行，其时钟源来自 f_{SUB}。
- 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_{SUB} 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、时基时钟和 f_{SUB} 时钟开启，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 将被清除并重新开始计数。
- 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。

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进一步降低，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。注意，如果使用 LIRC 振荡器需要消耗一些而外的待机电流

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。

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：

- PA 口下降沿
- 系统中断
- WDT 溢出

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	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)
HIRC	1024 HIRC 周期		1024 HIRC 周期
LIRC	1~2 LIRC 周期		1~2 LIRC 周期

唤醒时间

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中唤醒，HIRC 振荡器起振需要一定的延迟时间。

如果该系列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。

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。

看门狗定时器时钟源

WDT 定时器时钟源来自于内部时钟 f_{SUB}，而 f_{SUB} 的时钟源由 LIRC 提供。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。

看门狗定时器控制寄存器

WDTC 寄存器用于控制 WDT 功能的使能及溢出周期选择。

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 功能选择
01010B：使能
10101B：使能 (默认)
其它值：MCU 复位 (需要 2~3 个 LIRC 周期响应复位)
如果单片机复位且由 WDTC 寄存器中的 WE[4:0] 引起，则在复位后 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/fs
001：2^{10}/fs
010：2^{12}/fs
011：2^{14}/fs
100：2^{15}/fs
101：2^{16}/fs
110：2^{17}/fs
111：2^{18}/fs
这三位控制 WDT 时钟源的分频比，从而实现对 WDT 溢出周期的控制。

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

“x”：未知

Bit 7　详见其它章节

Bit 6~3　未使用，读为“0”

Bit 2　**LVRF**：LVR 复位标志位
详见其它章节

Bit 1　**LRF**：LVRC 控制的复位标志位
详见其它章节

Bit 0　**WRF**：由设置 WE[4:0] 引起的复位
0：无效
1：有效
该位可以被清为“0”，但不能设为“1”。

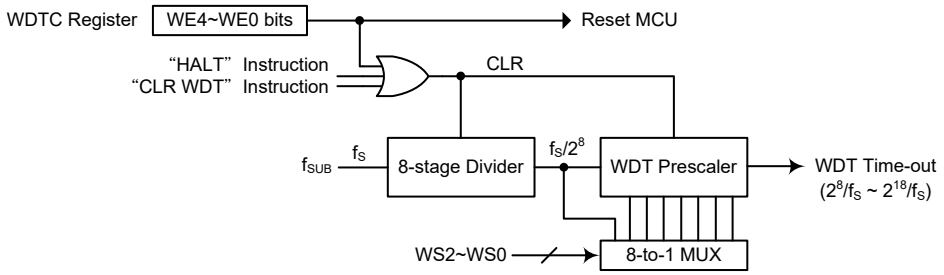
看门狗定时器操作

当 WDT 溢出时，看门狗定时器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溢出前有策略地清看门狗定时器以防止其产生复位，可使用清除看门狗指令实现。

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 应置位，仅程序计数器 PC 和堆栈指针 SP 复位。有三种方法可以用来清除 WDT 的内容。第一种是 WDT 软件复位，即将 WE4~WE0 位设置成除了“01010B”和“10101B”外的任意值；第二种是通过看门狗定时器软件清除指令，而第三种是通过“HALT”指令。

该系列单片机只使用一条清看门狗指令“CLR WDT”。因此只要执行“CLR WDT”便清除 WDT。

当设置分频比为 2^{18} 时，溢出周期最大。例如，时钟源为 32kHz 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

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。

另一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。

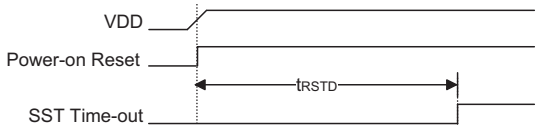
复位功能

单片机共有 4 种复位方式，包括内部和外部事件触发复位：

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。

由于上电条件不稳定，单片机具有内部复位功能。RC 电路产生一段延迟时间，能确保电源电压稳定时使得 POR 较长时间处于低电平。在此期间，单片机无法正常工作。当 POR 达到一定的电压值后，再经过一段复位延迟时间 t_{RSTD} 后，单片机才开始正常工作。

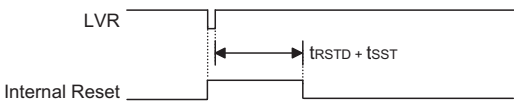


上电复位时序图

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落在 0.9V~V_{LVR} 的范围内，这时 LVR 将会自动复位单片机，CTRL 寄存器的 LVRF 标志位会被置位。

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 LVS7~LVS0 位设置。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 LVRC 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 – BS83A04A-3

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	R	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7~0 **LVS7 ~ LVS0**：LVR 电压选择
01010101：2.55V
00110011：2.55V
10011001：2.55V
10101010：2.55V
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。

● LVRC 寄存器 – BS83A02A-4/BS83A04A-4

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	R	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7~0 **LVS7 ~ LVS0**：LVR 电压选择
01010101：2.10V
00110011：2.10V
10011001：2.10V
10101010：2.10V
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。

● CTRL 寄存器

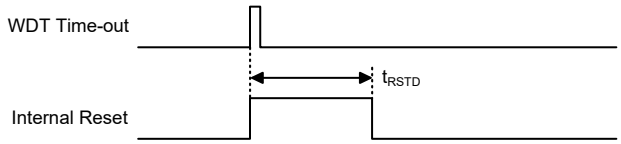
Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	—	LVRF	LRF	WRF
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	x	0	0

“x”：未知

Bit 7 详见其它章节
Bit 6~3 未使用，读为“0”
Bit 2 **LVRF**：LVR 复位标志位
0：无效
1：有效
此位只能被清零，不能被置为“1”。
Bit 1 **LRF**：LVRC 控制的复位标志位
0：无效
1：有效
此位只能被清零，不能被置为“1”。
Bit 0 详见其它章节

正常运行时看门狗溢出复位

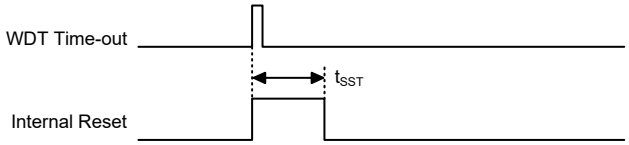
除了看门狗溢出标志位 TO 将被设为“1”之外，正常运行时看门狗溢出复位和外部硬件上电复位相同。



正常运行时看门狗溢出时序图

空闲或休眠时看门狗溢出复位

空闲或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。



注：如果系统时钟为 HIRC 时，则 t_{SST} 为 1024~1025 个时钟周期。
如果为 LIRC，则 t_{SST} 为 1~2 个时钟周期。

空闲或休眠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		LVR & 上电复位	WDT 溢出 (正常模式)	WDT 溢出 (空闲或休眠模式)
MP0		xxxx xxxx	uuuu uuuu	uuuu uuuu
MP1		xxxx xxxx	uuuu uuuu	uuuu uuuu
ACC		xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL		0000 0000	0000 0000	0000 0000
TBLP		xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH		xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP		---- --xx	---- --uu	---- --uu
STATUS		--00 xxxx	--1u uuuu	--11 uuuu
SMOD		000- 0011	000- 0011	uuu- uuuu
CTRL		0--- -x00	0--- -x00	u--- -uuu
INTEG		---- --00	---- --00	---- --uu
INTC0		-000 0000	-000 0000	-uuu uuuu
INTC1		--0- --0-	--0- --0-	--u- --u-
LVRC		0101 0101	0101 0101	uuuu uuuu
PA	BS83A02A-4	---- 1111	---- 1111	---- uuuu
	BS83A04A-3 BS83A04A-4	1111 1111	1111 1111	uuuu uuuu
PAC	BS83A02A-4	---- 1111	---- 1111	---- uuuu
	BS83A04A-3 BS83A04A-4	1111 1111	1111 1111	uuuu uuuu
PAPU	BS83A02A-4	---- 0000	---- 0000	---- uuuu
	BS83A04A-3 BS83A04A-4	0000 0000	0000 0000	uuuu uuuu
PAWU	BS83A02A-4	---- 0000	---- 0000	---- uuuu
	BS83A04A-3 BS83A04A-4	0000 0000	0000 0000	uuuu uuuu
WDTC		0101 0011	0101 0011	uuuu uuuu
TBC		--00 ----	--00 ----	--uu ----
TMR		0000 0000	0000 0000	uuuu uuuu
TMRC		--00 -000	--00 -000	--uu -uuu
TKTMR		0000 0000	0000 0000	uuuu uuuu
TKC0		-000 0-00	-000 0-00	-uuu u-uu
TK16DL		0000 0000	0000 0000	uuuu uuuu
TK16DH		0000 0000	0000 0000	uuuu uuuu
TKC1		---- --11	---- --11	---- --uu
TKM016DL		0000 0000	0000 0000	uuuu uuuu
TKM016DH		0000 0000	0000 0000	uuuu uuuu
TKM0ROL		0000 0000	0000 0000	uuuu uuuu
TKM0ROH		---- --00	---- --00	---- --uu

寄存器		LVR & 上电复位	WDT 溢出 (正常模式)	WDT 溢出 (空闲或休眠模式)
TKM0C0	BS83A02A-4	-00- 0000	-00- 0000	-uu- uuuu
	BS83A04A-3 BS83A04A-4	0000 0000	0000 0000	uuuu uuuu
TKM0C1	BS83A02A-4	0-00 --00	0-00 --00	u-uu --uu
	BS83A04A-3 BS83A04A-4	0-00 0000	0-00 0000	u-uu uuuu

注：“-”表示未定义
“x”表示未知
“u”表示不改变

输入 / 输出端口

Holek 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。

此系列单片机提供 PA 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。

单片机	寄存器名称	位							
		7	6	5	4	3	2	1	0
BS83A02A-4	PAWU	—	—	—	—	PAWU3	PAWU2	PAWU1	PAWU0
	PAPU	—	—	—	—	PAPU3	PAPU2	PAPU1	PAPU0
	PAC	—	—	—	—	PA3	PA2	PA1	PA0
	PA	—	—	—	—	PAC3	PAC2	PAC1	PAC0
BS83A04A-3 BS83A04A-4	PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
	PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
	PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
	PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0

输入 / 输出口寄存器列表

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 来设置，它用一个 PMOS 晶体管来实现上拉电阻功能。

PAPU 寄存器 – BS83A02A-4

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PAPU3	PAPU2	PAPU1	PAPU0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未使用，读为“0”

Bit 3~0　**PAPU3~PAPU0**：PA 口 bit 3~bit 0 上拉电阻控制
0：除能
1：使能

PAPU 寄存器 – BS83A04A-3/BS83A04A-4

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　**PAPU7~PAPU0**：PA 口 bit 7~bit 0 上拉电阻控制
0：除能
1：使能

PA 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 – BS83A02A-4

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PAWU3	PAWU2	PAWU1	PAWU0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未使用，读为“0”

Bit 3~0　**PAWU3~PAWU0**：PA 口 bit 3~bit 0 唤醒控制
0：除能
1：使能

PAWU 寄存器 – BS83A04A-3/BS83A04A-4

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　**PAWU7~PAWU0**：PA 口 bit 7~bit 0 唤醒控制
0：除能
1：使能

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 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 CMOS 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 – BS83A02A-4

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PAC3	PAC2	PAC1	PAC0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	1

Bit 7~4　未使用，读为“0”

Bit 3~0　**PAC3~PAC0**：PA 口 bit 3~bit 0 输入 / 输出控制
0：输出
1：输入

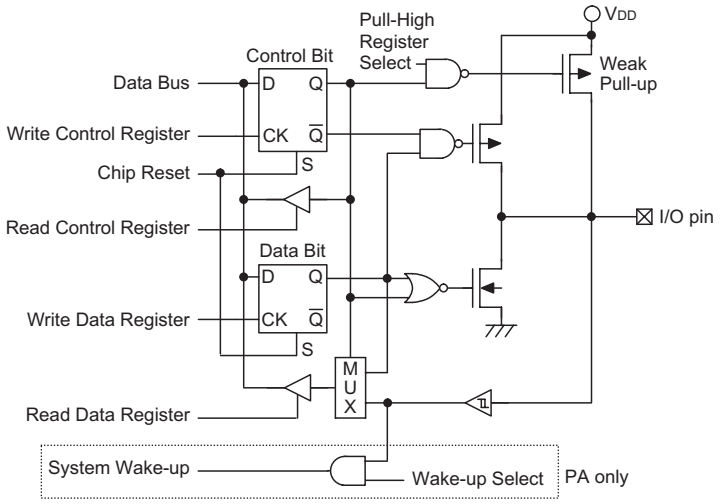
PAC 寄存器 – BS83A04A-3/BS83A04A-4

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　**PAC7~PAC0**：PA 口 bit 7~bit 0 输入 / 输出控制
0：输出
1：输入

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。



通用输入 / 输出端口结构

编程注意事项

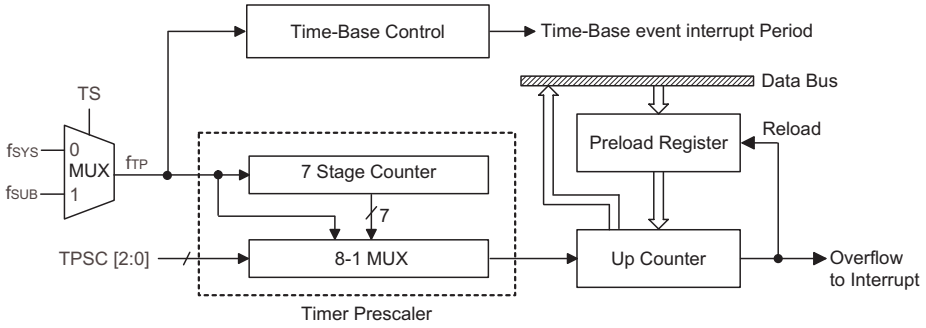
在编程中，最先要考虑的是端口的初始化。复位之后，所有的输入 / 输出数据及端口控制寄存器都将被设为逻辑高。所有输入 / 输出引脚默认为输入状态，而其电平则取决于其它相连接电路以及是否选择了上拉电阻。如果端口控制寄存器 PAC，某些引脚位被设定输出状态，这些输出引脚会有初始高电平输出，除非数据寄存器端口 PA 在程序中被预先设定。设置哪些引脚是输入及哪些引脚是输出，可通过设置正确的值到适当的端口控制寄存器，或使用指令“SET [m].i”及“CLR [m].i”来设定端口控制寄存器中个别的位。注意，当使用这些位控制指令时，系统即将产生一个读 - 修改 - 写的操作。单片机需要先读入整个端口上的数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

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。该系列单片机具有 1 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

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_{SUB} 振荡器，由 TMRC 寄存器的 TS 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 TPSC0~TPSC2 来确定。

定时 / 计数寄存器 – TMR

定时 / 计数寄存器 TMR，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，此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。

定时 / 计数控制寄存器 – TMRC

定时 / 计数控制寄存器为 TMRC，配合相应的 TMR 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 / 计数控制寄存器的第 4 位即 TON，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 / 计数控制寄存器的第 0~2 位用来控制输入时钟预分频器。TS 位用来选择内部时钟源。

TMRC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	TS	TON	—	TPSC2	TPSC1	TPSC0
R/W	—	—	R/W	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7 ~ 6　未使用，读为“0”

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

Bit 4　**TON**：定时 / 计数器控制位
0：除能
1：使能

Bit 3　未使用，读为“0”

Bit 2 ~ 0　**TPSC2 ~ TPSC0**：选择定时器预分频比选择位
定时器内部时钟 =
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：$f_{TP}/128$

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_{SUB} 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的 TPSC2~TPSC0 位来确定。定时器使能位，即 TON 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 INTC0 中的定时器中断使能位 TE 为 0，可以禁止计数器中断。

预分频器

TMRC 寄存器的 TPSC0~TPSC2 位用来确定定时 / 计数器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。

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。

触控按键功能

该系列单片机提供了多个触控按键功能。该按键功能内建于单片机内，不需外接元件，通过内部寄存器对其进行简单的操作。

触控按键结构

触控按键与 PA 的 I/O 引脚共用。通过寄存器的位来选择此功能。触控按键模块，具有自己的控制逻辑电路和设置寄存器。

单片机	Keys - n	触控按键	共用 I/O 口
BS83A02A-4	2	KEY1~KEY2	PA1, PA3
BS83A04A-3 BS83A04A-4	4	KEY1~KEY4	PA5, PA1, PA3, PA4

触控按键寄存器描述

触控按键模块包含 2 个或 4 个触控按键功能由所选的单片机而定，且都有自己相匹配的 8 个寄存器。以下表格显示了触控按键模块的寄存器设置。

名称	作用
TKTMR	触控按键 8 位定时 / 计数器寄存器
TKC0	计数器开关和清零控制 / 参考时钟控制 / 开始位
TK16DL	触控按键模块 16 位 C/F 低字节计数器
TK16DH	触控按键模块 16 位 C/F 高字节计数器
TKC1	触控按键振荡器频率选择
TKM016DH	16 位 C/F 高字节计数器
TKM016DL	16 位 C/F 低字节计数器
TKM0ROL	参考振荡器内建电容选择
TKM0ROH	参考振荡器内建电容选择
TKM0C0	控制寄存器 0，按键选择
TKM0C1	控制寄存器 1，按键选择或 I/O 引脚

触控按键寄存器

单片机	寄存器名称	位							
		7	6	5	4	3	2	1	0
BS83A02A-4 BS83A04A-3 BS83A04A-4	TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
	TKC0	—	TKRCOV	TKST	TKCFOV	TK16OV	—	TK16S1	TK16S0
	TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
	TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
	TKC1	—	—	—	—	—	—	TKFS1	TKFS0
	TKM016DL	D7	D6	D5	D4	D3	D2	D1	D0
	TKM016DH	D15	D14	D13	D12	D11	D10	D9	D8
	TKM0ROL	D7	D6	D5	D4	D3	D2	D1	D0
	TKM0ROH	—	—	—	—	—	—	D9	D8
BS83A02A-4	TKM0C0	—	M0MXS0	M0DFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
	TKM0C1	M0TSS	—	M0ROEN	M0KOEN	—	—	M0K2IO	M0K1IO
BS83A04A-3 BS83A04A-4	TKM0C0	M0MXS1	M0MXS0	M0DFEN	M0FILEN	M0SOFC	M0SOF2	M0SOF1	M0SOF0
	TKM0C1	M0TSS	—	M0ROEN	M0KOEN	M0K4IO	M0K3IO	M0K2IO	M0K1IO

触控按键寄存器列表

TKTMR 寄存器

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
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　触控按键 8 位定时 / 计数器寄存器
时隙计数器溢出设定的时间为 (256-TKTMR[7:0]) × 32

TKC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	TKRCOV	TKST	TKCFOV	TK16OV	—	TK16S1	TK16S0
R/W	—	R/W	R/W	R/W	R/W	—	R/W	R/W
POR	—	0	0	0	0	—	0	0

Bit 7　未使用，读为“0”

Bit 6　**TKRCOV**：时隙计数器溢出标志位
0：无溢出
1：溢出
如果模块 0 时隙计数器溢出，则触控按键中断请求标志位 TKMF 将会被置位且所有模块按键振荡器和参考振荡器自动停止。模块 0 的 16 位 C/F 计数器、16 位计数器、5 位时隙计数器和 8 位时隙时钟计数器都会自动关闭。

Bit 5　**TKST**：开启触控按键检测控制位
0：停止
0→1：开启
当该位为“0”时，模块 0 的 16 位 C/F 计数器、16 位计数器和 5 位时隙计数器会自动清零（8 位可编程时隙计数器不清，由用户设定溢出时间）。
当该位由 0→1 时，16 位 C/F 计数器、16 位计数器、5 位时隙计数器和 8 位时隙时钟计数器都会自动开启，并使能按键振荡器和参考振荡器输出时钟输入到这些计数器。

Bit 4　**TKCFOV**：触控按键模块 16 位 C/F 计数器溢出标志位
0：无溢出
1：溢出
该位由触控按键模块 16 位 C/F 计数器溢出置位，必须通过应用程序清零。

Bit 3 **TK16OV**：触控按键模块 16 位计数器溢出标志位
0：无溢出
1：溢出
该位由触控按键功能 16 位计数器溢出置位，必须通过应用程序清零。

Bit 2 未使用，读为“0”

Bit 1~0 **TK16S1~TK16S0**：触控按键模块 16 位计数器时钟选择位
00：f_{SYS}
01：$f_{SYS}/2$
10：$f_{SYS}/4$
11：$f_{SYS}/8$

TKC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	TKFS1	TKFS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	1	1

Bit 7~2 未使用，读为“0”

Bit 1~0 **TKFS1~TKFS0**：触控按键振荡器频率选择位
00：500kHz
01：1000kHz
10：1500kHz
11：2000kHz

TK16DL 寄存器

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 触控按键模块 16 位计数器低字节内容

TK16DH 寄存器

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 触控按键模块 16 位计数器高字节内容

TKM016DL 寄存器

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 模块 0 16 位计数器低字节内容

TKM016DH 寄存器

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0　模块 0 16 位计数器高字节内容

TKM0ROL 寄存器

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
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　参考振荡器内建电容选择
振荡器内建电容选择为 (TKM0RO[9:0] × 50pF)/1024

TKM0ROH 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未使用，读为“0”
Bit 1~0　振荡器内建电容选择为 (TKM0RO[9:0] × 50pF)/1024

TKM0C0 寄存器 – BS83A02A-4

Bit	7	6	5	4	3	2	1	0
Name	—	M0MXS0	M0DFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
R/W	—	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	—	0	0	—	0	0	0	0

Bit 7　未使用，读为“0”
Bit 6　**M0MXS0**：复用按键选择
0：KEY1
1：KEY2
Bit 5　**M0DFEN**：倍频功能控制
0：除能
1：使能
Bit 4　未使用，读为“0”
Bit 3　**M0SOFC**：C/F 振荡器跳频功能控制选择位
0：由 M0SOF2~M0SOF0 位控制
1：由硬件电路控制
该位用于选择 C/F 振荡器跳频功能控制方式。当此位置 1 时，按键振荡器跳频功能由硬件电路控制，M0SOF2~M0SOF0 位的设置无效。

Bit 2~0 **M0SOF2~M0SOF0**：选择按键振荡器和参考振荡器作为选择 C/F 振荡器时频率选择位 (M0SOFC=0)
000：1380kHz
001：1500kHz
010：1670kHz
011：1830kHz
100：2000kHz
101：2230kHz
110：2460kHz
111：2740kHz
这些频率会随着外部或内部电容值的变化而变化。如果触控按键工作于 2MHz 频率下，则当选择其它频率时，用户可以在一定范围内调整频率。

TKM0C0 寄存器 – BS83A04A-3/BA83A04A-4

Bit	7	6	5	4	3	2	1	0
Name	M0MXS1	M0MXS0	M0DFEN	M0FILEN	M0SOFC	M0SOF2	M0SOF1	M0SOF0
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 7~6 **M0MXS1~M0MXS0**：复用按键选择
00：KEY1
01：KEY2
10：KEY3
11：KEY4

Bit 5 **M0DFEN**：倍频功能控制
0：除能
1：使能

Bit 4 **M0FILEN**：滤波器功能控制
0：除能
1：使能

Bit 3 **M0SOFC**：C/F 振荡器跳频功能控制选择位
0：由 M0SOF2~M0SOF0 位控制
1：由硬件电路控制
该位用于选择 C/F 振荡器跳频功能控制方式。当此位置 1 时，按键振荡器跳频功能由硬件电路控制，M0SOF2~M0SOF0 位的设置无效。

Bit 2~0 **M0SOF2~M0SOF0**：选择按键振荡器和参考振荡器作为选择 C/F 振荡器时频率选择位 (M0SOFC=0)
000：1380kHz
001：1500kHz
010：1670kHz
011：1830kHz
100：2000kHz
101：2230kHz
110：2460kHz
111：2740kHz
上述频率会随着外部或内部电容值的不同而变化。按键振荡器频率选择 2MHz 时，用户选择其它频率时，可依此比例调整。

TKM0C1 寄存器 – BS83A02A-4

Bit	7	6	5	4	3	2	1	0
Name	M0TSS	—	M0ROEN	M0KOEN	—	—	M0K2IO	M0K1IO
R/W	R/W	—	R/W	R/W	—	—	R/W	R/W
POR	0	—	0	0	—	—	0	0

Bit 7 **M0TSS**：时隙计数器选择
0：参考振荡器
1：$f_{SYS}/4$

Bit 6 未使用，读为“0”

Bit 5 **M0ROEN**：参考振荡器控制
0：除能
1：使能

Bit 4 **M0KOEN**：按键振荡器控制
0：除能
1：使能

Bit 3~2 未使用，读为“0”

Bit 1 **M0K2IO**：I/O 引脚和触控按键 2 功能选择
0：I/O 引脚
1：触控按键

Bit 0 **M0K1IO**：I/O 引脚和触控按键 1 功能选择
0：I/O 引脚
1：触控按键

TKM0C1 寄存器 – BS83A04A-3/BA83A04A-4

Bit	7	6	5	4	3	2	1	0
Name	M0TSS	—	M0ROEN	M0KOEN	M0K4IO	M0K3IO	M0K2IO	M0K1IO
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 **M0TSS**：时隙计数器选择
0：参考振荡器
1：$f_{SYS}/4$

Bit 6 未使用，读为“0”

Bit 5 **M0ROEN**：参考振荡器控制
0：除能
1：使能

Bit 4 **M0KOEN**：按键振荡器控制
0：除能
1：使能

Bit 3 **M0K4IO**：I/O 引脚和触控按键 4 功能选择
0：I/O 引脚
1：触控按键

Bit 2 **M0K3IO**：I/O 引脚和触控按键 3 功能选择
0：I/O 引脚
1：触控按键

Bit 1 **M0K2IO**：I/O 引脚和触控按键 2 功能选择
0：I/O 引脚
1：触控按键

Bit 0 **M0K1IO**：I/O 引脚和触控按键 1 功能选择
0：I/O 引脚
1：触控按键

触控按键操作

手指接近或接触到触控面板时，面板的电容量会增大，电容量的变化会轻微改变内部感应振荡器的频率，通过测量频率的变化可以感知触控动作。参考时钟通过内部可编程分频器能够产生一个固定的时间周期。在这个时间周期内，通过在此固定时间周期内对感应振荡器产生的时钟周期计数，可确定触控按键的动作。

每个触控按键模块包含 2 个或 4 个与 I/O 引脚共用的触控按键。通过寄存器可设置相应引脚功能。触控按键模块有自己的独立中断向量和中断标志位。

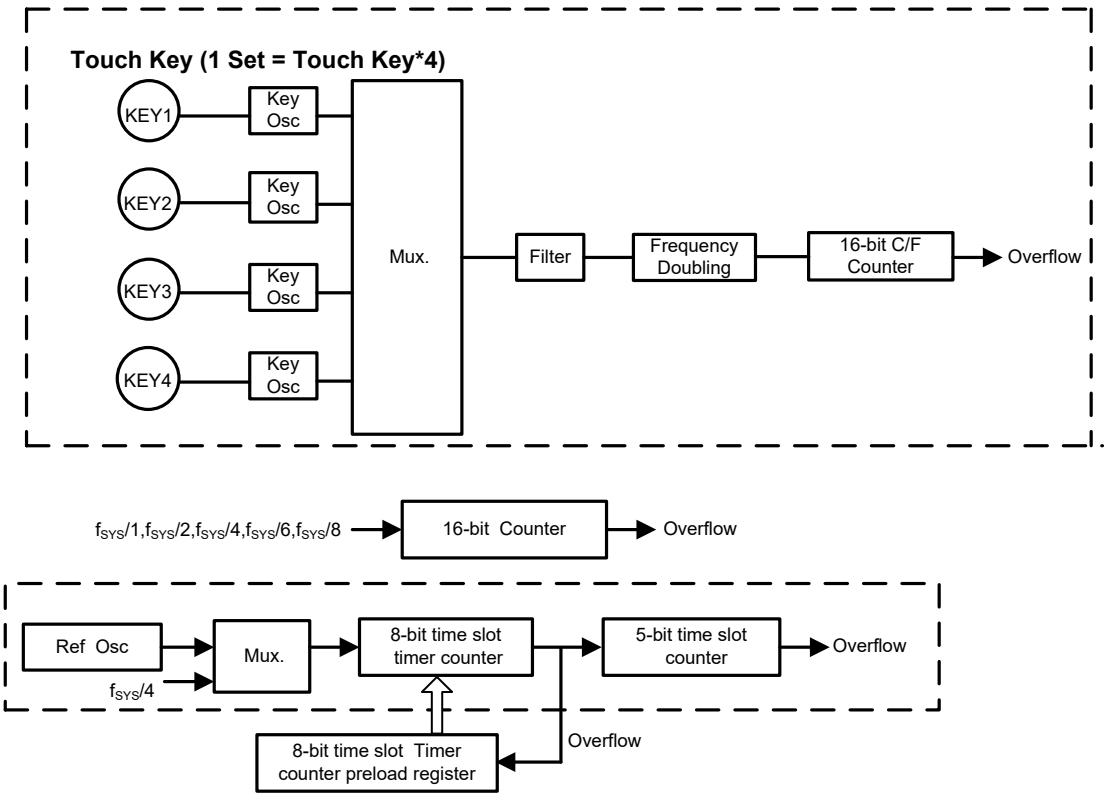
在参考时钟固定的时间间隔内，感应振荡器产生的时钟周期数是可以测量的。测到的周期数可以用于判断触控动作是否有效发生。在最后一个时间间隔后，会产生一个触控按键中断信号。

在 TKST 上升沿时，16 位 C/F 计数器、16 位计数器、5 位时隙计数器和 8 位时隙定时 / 计数器会自动开启；在 TKST 下降沿时，16 位 C/F 计数器、16 位计数器和 5 位时隙计数器会自动清除 (8 位可编程时隙计数器不清，由用户设定溢出时间)。

当 5 位时隙计数器溢出时，模块 0 的按键振荡器和参考振荡器会自动停止且 16 位 C/F 计数器、16 位计数器、5 位时隙计数器和 8 位时隙定时 / 计数器会自动停止。5 位时隙计数器和 8 位时隙定时 / 计数器时钟来自参考振荡器或 $f_{SYS}/4$。

当 TKM0C1 寄存器中的 M0ROEN 位为“1”时，参考振荡器使能；当 TKM0C1 寄存器中的 M0KOEN 为“1”时，按键振荡器使能。

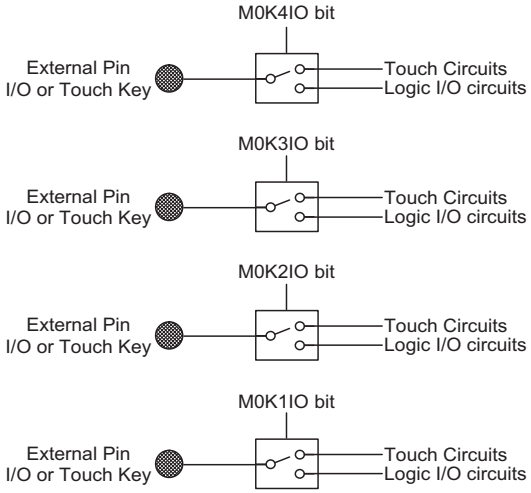
当时隙计数器溢出时，才产生中断。



注：(1) 虚线部分是每个触控按键都单独有的部分。

(2) 对于 BS83A02A-4，一个模块内仅包含 2 个触控按键且无滤波器功能。

触控按键模块方框图



触控按键或输入 / 输出功能选择

触控按键中断

触控按键模块包含 2 个或 4 个触控按键，有一个独立的中断。触控按键模块的时隙计数器溢出时，才产生中断，此时 16 位 C/F 计数器、16 位计数器、5 位时隙计数器和 8 位时隙计数器会自动清零。只有使能的按键都溢出时才产生中断。更多细节请参考触控按键中断章节。

编程注意事项

相关寄存器设置后，TKST 位由低电平变为高电平，触控按键检测程序初始化。此时所有相关的振荡器将使能并同步。时隙计数器标志位 TKRCOV 将变为高电平直到计数器溢出。计数器溢出发生时，将会产生一个中断信号。

当外部触控按键的大小和布局确定时，其相关的电容将决定感应振荡器的频率。

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。

该系列单片机提供一个外部中断和多个内部中断。外部中断由 INT 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。

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系列寄存器控制的。寄存器的数量由所选单片机的型号决定，但总的分为两类。第一类是 INTC0~INTC1 寄存器，用于设置基本的中断；第二类由 INTEG 寄存器，用于设置外部中断边沿触发类型。

寄存器中含有中断控制位和中断请求标志位。中断控制位用于使能或除能各种中断，中断请求标志位用于存放当前中断请求的状态。它们都按照特定的模式命名，前面表示中断类型的缩写，紧接着的字母“E”代表使能 / 除能位，“F”代表请求标志位。

功能	使能位	请求标志
总中断	EMI	—
外部中断	INTE	INTF
触控按键模块中断	TKME	TKMF
定时 / 计数器中断	TE	TF
时基中断	TBE	TBF

中断寄存器位命名模式

寄存器名称	位							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	TF	TKMF	INTF	TE	TKME	INTE	EMI
INTC1	—	—	TBF	—	—	—	TBE	—

中断寄存器列表

INTEG 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INTS1	INTS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未定义，读为“0”

Bit 1~0　**INTS1, INTS0**：INT 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	—	TF	TKMF	INTF	TE	TKME	INTE	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　**TF**：定时 / 计数器中断请求标志位
0：无请求
1：中断请求

Bit 5　**TKMF**：触控按键模块中断请求标志位
0：无请求
1：中断请求

Bit 4　**INTF**：INT 中断请求标志位
0：无请求
1：中断请求

Bit 3　**TE**：定时 / 计数器中断控制位
0：除能
1：使能

Bit 2　**TKME**：触控按键模块中断控制位
0：除能
1：使能

Bit 1　**INTE**：INT 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	—	—	TBF	—	—	—	TBE	—
R/W	—	—	R/W	—	—	—	R/W	—
POR	—	—	0	—	—	—	0	—

Bit 7~6　未定义，读为“0”

Bit 5　**TBF**：时基中断请求标志位
0：无请求
1：中断请求

Bit 4~2　未定义，读为“0”

Bit 1　**TBE**：时基中断控制位
0：除能
1：使能

Bit 0　未定义，读为“0

中断操作

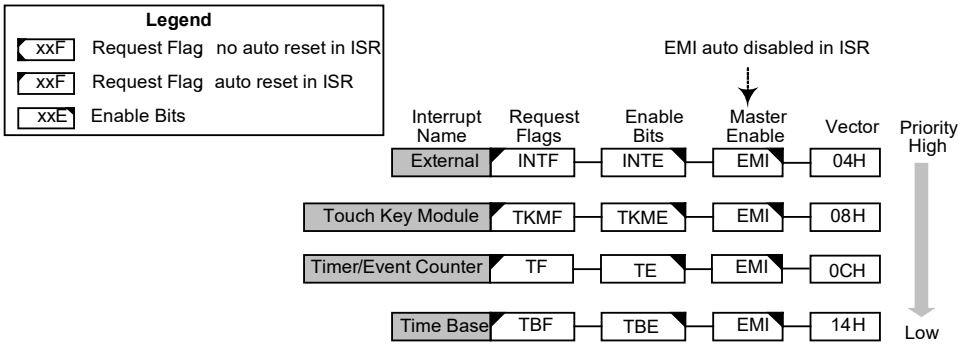
若中断事件条件产生，如触控按键计数器溢出、定时 / 计数器溢出等等，相关中断请求标志将置起。中断标志产生后程序是否会跳转至相关中断向量执行是由中断使能位的条件决定的。若使能位为“1”，程序将跳至相关中断向量中执行；若使能位为“0”，即使中断请求标志置起中断也不会发生，程序也不会跳转至相关中断向量执行。若总中断使能位为“0”，所有中断都将除能。

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。一些中断源有自己的向量，但是有些中断却共用多功能中断向量。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。

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。



中断结构

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。

外部中断

通过 INT 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，INT 引脚的状态发生变化，外部中断请求标志 INTF 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 INTE 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 INTF 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。

触控按键中断

要使触控按键中断发生，总中断控制位 EMI 和相应的内部中断使能位 TKME 必须先被置位。当触控按键中的时隙计数器溢出，相应的中断请求标志位 TKMF 将置位并触发触控按键中断。中断使能，堆栈未满，当触控按键时隙计数器溢出发生中断时，将调用位于触控按键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 TKMF 会被自动复位且 EMI 位会被清零以除能其它中断。

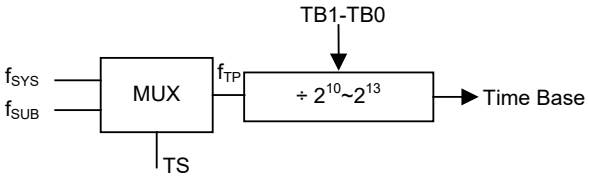
定时 / 计数器中断

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 TE 必须先被置位。当定时 / 计数器溢出，相应的中断请求标志位 TF 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 TF 会被自动复位且 EMI 位会被清零以除能其它中断。

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_{SUB} 由 TMRC 寄存器中的 TS 位选择。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。f_{TP} 时钟源控制着时基中断周期，而 f_{TP} 可来自于不同的时钟源，可从工作模式章节获得。



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”

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。

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。

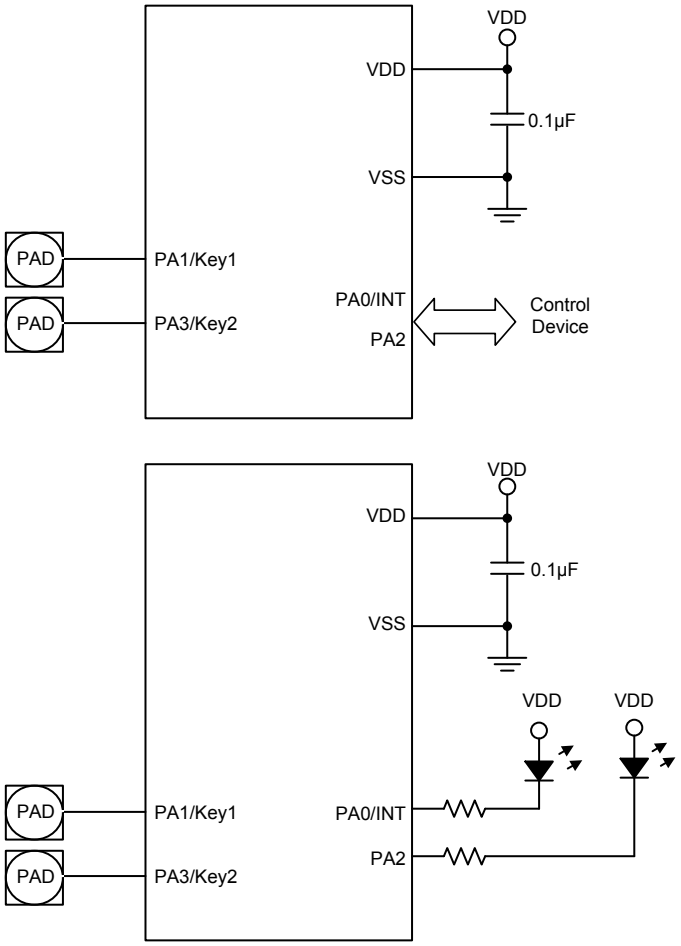
所有中断在休眠或空闲模式下都具有唤醒功能，当中断请求标志发生由低到高的转变时都可产生唤醒功能。若要避免相应中断产生唤醒动作，在单片机进入休眠或空闲模式前需先将相应请求标志置为高。

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。

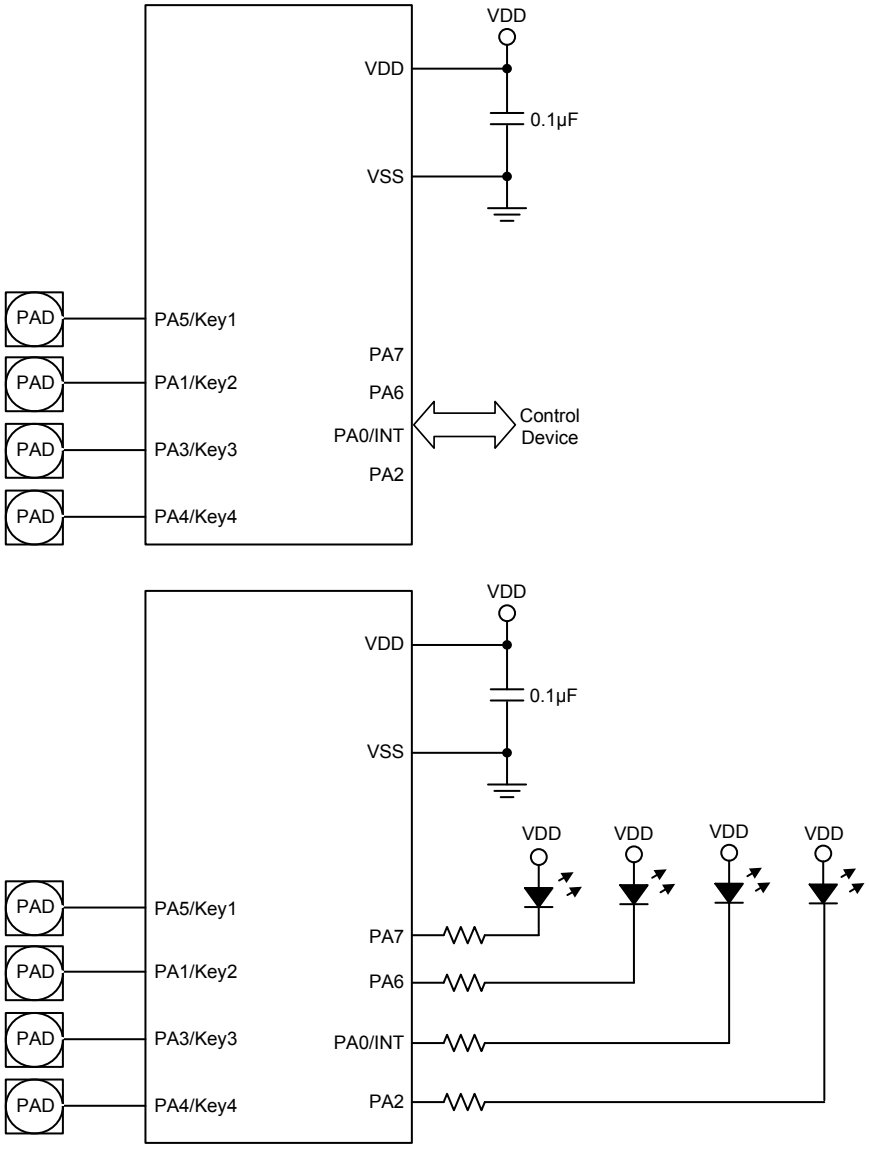
若从中断子程序中返回可执行 RET 或 RETI 指令。除了能返回至主程序外，RETI 指令还能自动设置 EMI 位为高，允许进一步中断。RET 指令只能返回至主程序，清除 EMI 位，除能进一步中断。

应用电路

BS83A02A-4



BS83A04A-3/BA83A04A-4



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 Holek 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它们的应用。

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 Holek 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 Holek 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 Holek 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，Holek 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

下表中说明了按功能分类的指令集，用户可以将该表作为基本的指令参考。

惯例

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

助记符	说明	指令周期	影响标志位
算术运算			
ADD A,[m]	ACC 与数据存储器相加，结果放入 ACC	1	Z, C, AC, OV
ADDM A,[m]	ACC 与数据存储器相加，结果放入数据存储器	1 注	Z, C, AC, OV
ADD A, x	ACC 与立即数相加，结果放入 ACC	1	Z, C, AC, OV
ADC A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	1	Z, C, AC, OV
ADCM A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	1 注	Z, C, AC, OV
SUB A, x	ACC 与立即数相减，结果放入 ACC	1	Z, C, AC, OV
SUB A,[m]	ACC 与数据存储器相减，结果放入 ACC	1	Z, C, AC, OV
SUBM A,[m]	ACC 与数据存储器相减，结果放入数据存储器	1 注	Z, C, AC, OV
SBC A,[m]	ACC 与数据存储器、进位标志相减，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

助记符	说明	指令周期	影响标志位
RL [m]	数据存储器左移一位，结果放入数据存储器	1 注	无
RLCA [m]	带进位将数据存储器左移一位，结果放入 ACC	1	C
RLC [m]	带进位将数据存储器左移一位，结果放入数据存储器	1 注	C
数据传送			
MOV A,[m]	将数据存储器送至 ACC	1	无
MOV [m],A	将 ACC 送至数据存储器	1 注	无
MOV A, x	将立即数送至 ACC	1	无
位运算			
CLR [m].i	清除数据存储器的位	1 注	无
SET [m].i	置位数据存储器的位	1 注	无
转移			
JMP addr	无条件跳转	2	无
SZ [m]	如果数据存储器为零，则跳过下一条指令	1 注	无
SZA [m]	数据存储器送至 ACC，如果内容为零，则跳过下一条指令	1 注	无
SZ [m].i	如果数据存储器的第 i 位为零，则跳过下一条指令	1 注	无
SNZ [m].i	如果数据存储器的第 i 位不为零，则跳过下一条指令	1 注	无
SIZ [m]	递增数据存储器，如果结果为零，则跳过下一条指令	1 注	无
SDZ [m]	递减数据存储器，如果结果为零，则跳过下一条指令	1 注	无
SIZA [m]	递增数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	1 注	无
SDZA [m]	递减数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	1 注	无
CALL addr	子程序调用	2	无
RET	从子程序返回	2	无
RET A, x	从子程序返回，并将立即数放入 ACC	2	无
RETI	从中断返回	2	无
查表			
TABRD [m]	读取特定页或当前页的 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
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。

指令定义

ADC A, [m] Add Data Memory to ACC with Carry
指令说明 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。
功能表示 ACC ← ACC + [m] + C
影响标志位 OV、Z、AC、C

ADCM A, [m] Add ACC to Data Memory with Carry
指令说明 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。
功能表示 [m] ←ACC + [m] + C
影响标志位 OV、Z、AC、C

ADD A, [m] Add Data Memory to ACC
指令说明 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。
功能表示 ACC ← ACC + [m]
影响标志位 OV、Z、AC、C

ADD A, x Add immediate data to ACC
指令说明 将累加器和立即数相加，结果存放到累加器。
功能表示 ACC ← ACC + x
影响标志位 OV、Z、AC、C

ADDM A, [m] Add ACC to Data Memory
指令说明 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。
功能表示 [m] ←ACC + [m]
影响标志位 OV、Z、AC、C

AND A, [m] Logical AND Data Memory to ACC
指令说明 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。
功能表示 ACC ← ACC “AND” [m]
影响标志位 Z

AND A, x	Logical AND immediate data to ACC
指令说明	将累加器中的数据和立即数做逻辑与，结果存放到累加器。
功能表示	ACC ← ACC“AND”x
影响标志位	Z

ANDM A, [m]	Logical AND ACC to Data Memory
指令说明	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。
功能表示	[m] ← ACC“AND”[m]
影响标志位	Z

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	无

CLR [m]	Clear Data Memory
指令说明	将指定数据存储器的内容清零。
功能表示	[m] ← 00H
影响标志位	无

CLR [m].i	Clear bit of Data Memory
指令说明	将指定数据存储器的第 i 位内容清零。
功能表示	[m].i ← 0
影响标志位	无

CLR WDT	Clear Watchdog Timer
指令说明	WDT 计数器、暂停标志位 PDF 和看门狗溢出标志位 TO 清零。
功能表示	WDT cleared TO & PDF ← 0
影响标志位	TO、PDF

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

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

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory
指令说明 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。
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

DEC [m] Decrement Data Memory
指令说明 将指定数据存储器内容减 1。
功能表示 [m] ← [m] – 1
影响标志位 Z

DECA [m] Decrement Data Memory with result in ACC
指令说明 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。
功能表示 ACC ← [m] – 1
影响标志位 Z

HALT Enter power down mode
指令说明 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。
功能表示 TO ← 0
PDF ← 1
影响标志位 TO、PDF

INC [m] Increment Data Memory
指令说明 将指定数据存储器的内容加 1。
功能表示 [m] ← [m] + 1
影响标志位 Z

INCA [m] Increment Data Memory with result in ACC
指令说明 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。
功能表示 ACC ← [m] + 1
影响标志位 Z

JMP addr Jump unconditionally
指令说明 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。
功能表示 Program Counter ← addr
影响标志位 无

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。
功能表示 ACC← [m]
影响标志位 无

MOV A, x Move immediate data to ACC
指令说明 将 8 位立即数载入累加器。
功能表示 ACC ← x
影响标志位 无

MOV [m], A Move ACC to Data Memory
指令说明 将累加器的内容复制到指定的数据存储器。
功能表示 [m] ← ACC
影响标志位 无

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	无操作
影响标志位	无

OR A, [m]	Logical OR Data Memory to ACC
指令说明	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。
功能表示	ACC ← ACC“OR”[m]
影响标志位	Z

OR A, x	Logical OR immediate data to ACC
指令说明	将累加器中的数据和立即数逻辑或，结果存放到累加器。
功能表示	ACC ← ACC“OR”x
影响标志位	Z

ORM A, [m]	Logical OR ACC to Data Memory
指令说明	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。
功能表示	[m] ← ACC“OR”[m]
影响标志位	Z

RET	Return from subroutine
指令说明	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。
功能表示	Program Counter←Stack
影响标志位	无

RET A, x	Return from subroutine and load immediate data to ACC
指令说明	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。
功能表示	Program Counter ← Stack ACC←x
影响标志位	无

RETI Return from interrupt

指令说明 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 无

RL [m] Rotate Data Memory left

指令说明 将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。

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

影响标志位 无

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。

功能表示 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

指令说明 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。

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

影响标志位 C

RLC A [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。

功能表示 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
指令说明 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。
功能表示 [m].i ← [m].(i+1) (i=0~6)
[m].7 ← [m].0
影响标志位 无

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。
功能表示 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
指令说明 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。
功能表示 [m].i ← [m].(i+1) (i=0~6)
[m].7← C
C ← [m].0
影响标志位 C

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。
功能表示 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 from ACC with Carry
指令说明 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。
功能表示 ACC ← ACC – [m] – $\overline{C}$
影响标志位 OV、Z、AC、C

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

指令说明 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。

功能表示 $[m] \leftarrow ACC - [m] - \overline{C}$

影响标志位 OV、Z、AC、C

SDZ [m] Skip if Decrement Data Memory is 0

指令说明 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。

功能表示 [m] ← [m] – 1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

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。

功能表示 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

指令说明 将指定数据存储器的每一位设置为 1。

功能表示 [m] ← FFH

影响标志位 无

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 无

SIZ [m]	Skip if increment Data Memory is 0
指令说明	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。
功能表示	[m] ←[m] + 1，如果 [m]=0 跳过下一条指令执行
影响标志位	无

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。
功能表示	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 Data Memory is not 0
指令说明	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。
功能表示	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 ACC
指令说明	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。
功能表示	ACC ← ACC – [m]
影响标志位	OV、Z、AC、C

SUBM A, [m]	Subtract Data Memory from ACC with result in Data Memory
指令说明	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。
功能表示	[m] ← ACC – [m]
影响标志位	OV、Z、AC、C

SUB A, x Subtract immediate Data from ACC
指令说明 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。
功能表示 ACC ← ACC – x
影响标志位 OV、Z、AC、C

SWAP [m] Swap nibbles of Data Memory
指令说明 将指定数据存储器的低 4 位和高 4 位互相交换。
功能表示 [m].3~[m].0 ↔ [m].7~[m].4
影响标志位 无

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，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。
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 无

SZ [m] Skip if Data Memory is 0
指令说明 指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。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。
功能表示 如果 [m]=0，跳过下一条指令执行
影响标志位 无

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。
功能表示 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 Data Memory is 0
指令说明	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。
功能表示	如果 [m].i=0，跳过下一条指令执行
影响标志位	无

TABRD [m]	Read table (specific page or current page) to TBLH and Data Memory
指令说明	将表格指针 (TBHP 和 TBLP，若无 TBHP 则仅 TBLP) 所指的程序代码低字节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	无

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	无

XOR A, [m]	Logical XOR Data Memory to ACC
指令说明	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。
功能表示	ACC ← ACC“XOR”[m]
影响标志位	Z

XORM A, [m]	Logical XOR ACC to Data Memory
指令说明	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。
功能表示	[m] ← ACC“XOR”[m]
影响标志位	Z

XOR A, x	Logical XOR immediate data to ACC
指令说明	将累加器的数据与立即数逻辑异或，结果存放到累加器。
功能表示	ACC ← ACC“XOR”x
影响标志位	Z

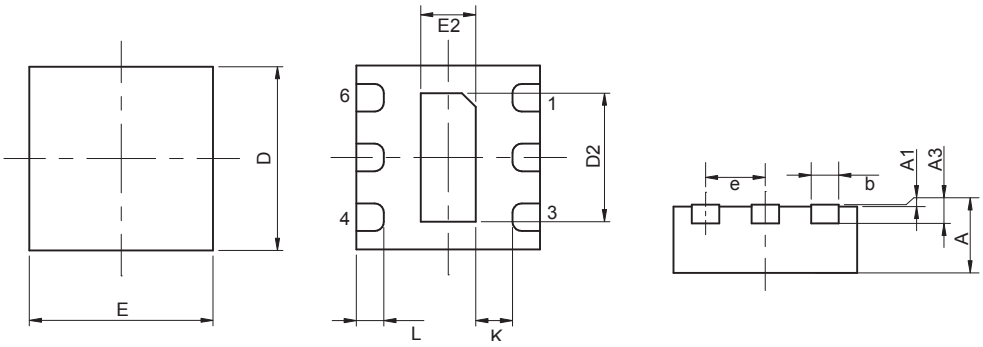
封装信息

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

封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

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

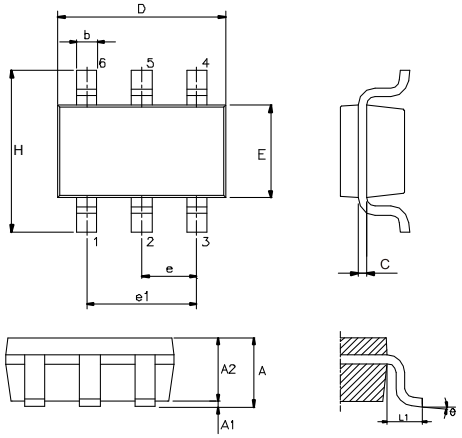
6-pin DFN (2mm×2mm) 外形尺寸



符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3	—	0.008 BSC	—
b	0.010	0.012	0.014
D	—	0.079 BSC	—
E	—	0.079 BSC	—
e	—	0.026 BSC	—
D2	0.053	0.055	0.057
E2	0.022	0.024	0.026
L	0.010	0.012	0.014
K	0.008	—	—

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	—	0.20 BSC	—
b	0.25	0.30	0.35
D	—	2.0 BSC	—
E	—	2.0 BSC	—
e	—	0.65 BSC	—
D2	1.35	1.40	1.45
E2	0.55	0.60	0.65
L	0.25	0.30	0.35
K	0.20	—	—

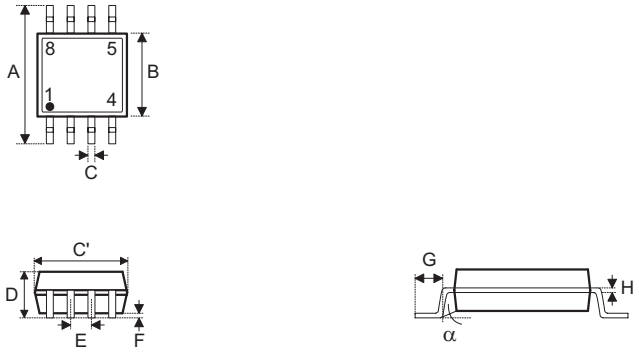
SOT23-6 外形尺寸



符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	—	—	0.057
A1	—	—	0.006
A2	0.035	0.045	0.051
b	0.012	—	0.020
C	0.003	—	0.009
D	—	0.114 BSC	—
E	—	0.063 BSC	—
e	—	0.037 BSC	—
e1	—	0.075 BSC	—
H	—	0.110 BSC	—
L1	—	0.024 BSC	—
θ	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	—	—	1.45
A1	—	—	0.15
A2	0.90	1.15	1.30
b	0.30	—	0.50
C	0.08	—	0.22
D	—	2.90 BSC	—
E	—	1.60 BSC	—
e	—	0.95 BSC	—
e1	—	1.90 BSC	—
H	—	2.80 BSC	—
L1	—	0.60 BSC	—
θ	0°	—	8°

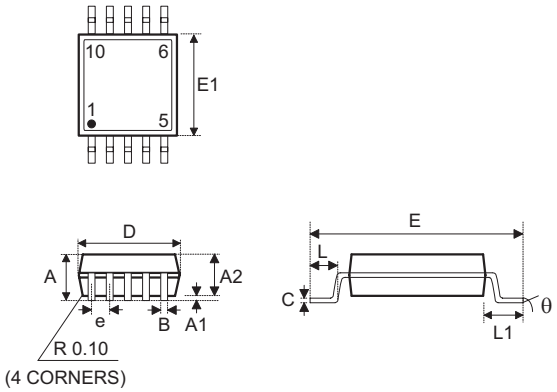
8-pin SOP (150mil) 外形尺寸



符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C’	—	0.193 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	—	6.0 BSC	—
B	—	3.9 BSC	—
C	0.31	—	0.51
C’	—	4.9 BSC	—
D	—	—	1.75
E	—	1.27 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

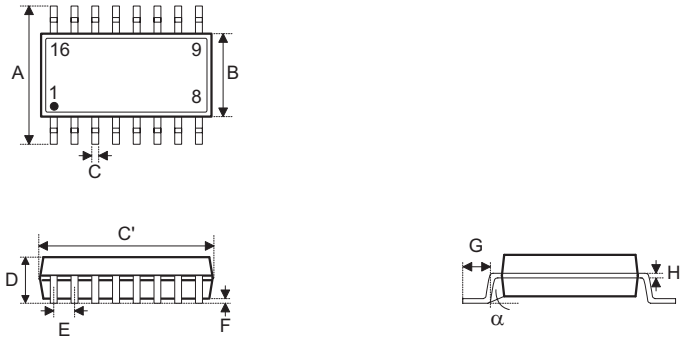
10-pin MSOP (300mil) 外形尺寸



符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	—	—	0.043
A1	0.000	—	0.006
A2	0.030	0.033	0.037
b	0.007	—	0.013
c	0.003	—	0.009
D	—	0.118 BSC	—
E	—	0.193 BSC	—
E1	—	0.118 BSC	—
e	—	0.020 BSC	—
L	0.016	0.024	0.031
L1	—	0.037 BSC	—
y	—	0.004	—
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	—	—	1.10
A1	0.00	—	0.15
A2	0.75	0.85	0.95
b	0.17	—	0.33
c	0.08	—	0.23
D	—	3.0 BSC	—
E	—	4.9 BSC	—
E1	—	3.0 BSC	—
e	—	0.5 BSC	—
L	0.40	0.60	0.80
L1	—	0.95 BSC	—
y	—	0.1	—
α	0°	—	8°

16-pin NSOP (150mil) 外形尺寸



符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C’	—	0.390 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	—	6.0 BSC	—
B	—	3.9 BSC	—
C	0.31	—	0.51
C’	—	9.9 BSC	—
D	—	—	1.75
E	—	1.27 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°