增强型触控式 Flash 单片机

BS83B08C/BS83B12C/BS83B16C

版本：V1.51　日期：2023-11-10

www.holtek.com

目录

特性

CPU 特性

- 工作电压：
  - ♦ f_{SYS}=8MHz：2.2V~5.5V
  - ♦ f_{SYS}=12MHz：2.7V~5.5V
  - ♦ f_{SYS}=16MHz：3.3V~5.5V
- V_{DD}=5V，系统时钟为 16MHz 时，指令周期为 0.25μs
- 提供暂停和唤醒功能，以降低功耗
- 振荡器类型：
  - ♦ 内部高速 RC – HIRC
  - ♦ 内部低速 32kHz RC – LIRC
- 多种工作模式：快速、低速、空闲和休眠
- 内部集成振荡器，无需外接元器件
- 所有指令都可在 1 或 2 个指令周期内完成
- 查表指令
- 61 条指令
- 6 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器：2K×16
- 数据存储器：288×8~512×8
- True EEPROM 存储器：64×8
- 看门狗定时器功能
- 多达 22 个双向 I/O 口
- 单个与 I/O 口共用的外部中断输入
- 单个 10-bit PTM 用于时间测量、捕捉输入、比较匹配输出、PWM 输出及单脉冲输出功能
- 单个时基功能用以产生固定的中断信号
- 串行接口模块包含 SPI 和 I^2C 通信
- 低电压复位功能
- 8/12/16 个触控按键功能
- 多种封装类型：16-pin NSOP/SSOP/QFN，20/24-pin SOP/SSOP/QFN

开发工具

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

https://www.holtek.com.cn/page/detail/dev_plat/washing_machine_workshop

https://www.holtek.com.cn/page/detail/dev_plat/Touch_Workshop

概述

该系列单片机是一款具有 8 位高性能精简指令集且完全集成触控按键功能的 Flash 单片机。该系列单片机具有内部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，无需外部元器件。除了 Flash 程序存储器，还包括 RAM 数据存储器和用于存储序列数据、校准数据等非易失性数据的 True EEPROM 存储器。内部看门狗定时器和低电压复位等内部保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地运行。

该系列单片机提供了内外部高低速振荡器功能选项，且内建完整的系统振荡器，无需外接元器件。其在不同工作模式之间动态切换的能力，为用户提供了一个优化单片机操作和减少功耗的手段。通过内部 SPI 和 I^2C 接口，可方便与外部设备之间的通讯，I/O 灵活、定时器模块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。

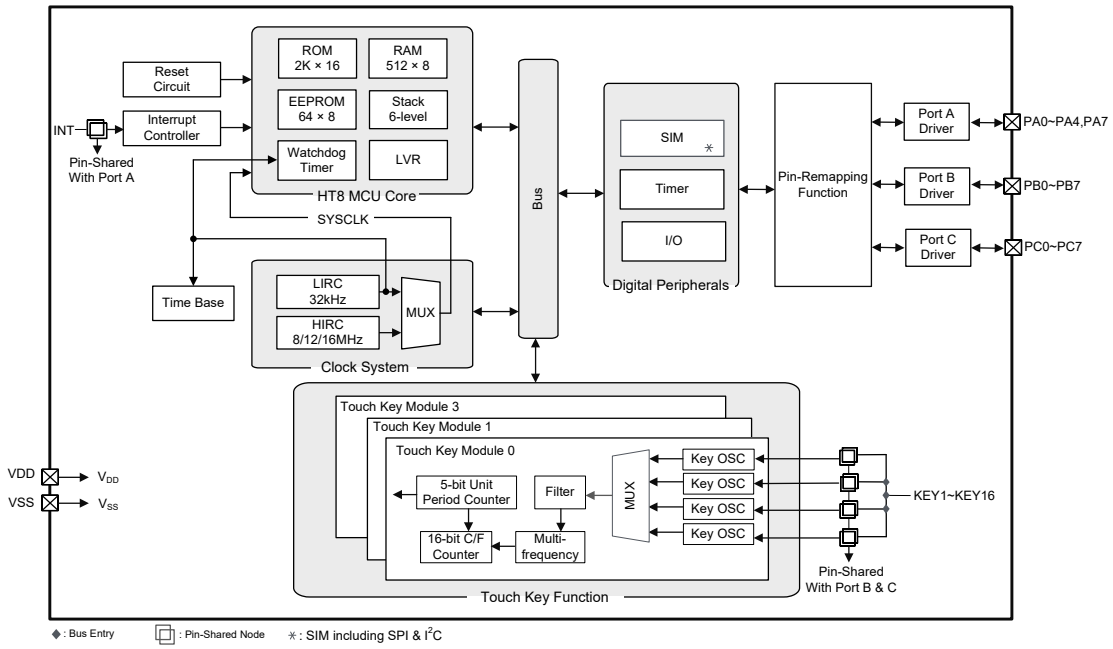
选型表

对此系列的芯片而言，大多数的特性参数都是一样的。主要差异在于数据存储器的容量，I/O 引脚数目，定时器模块数量和触控按键的数量。下表列出了各单片机的主要特性。

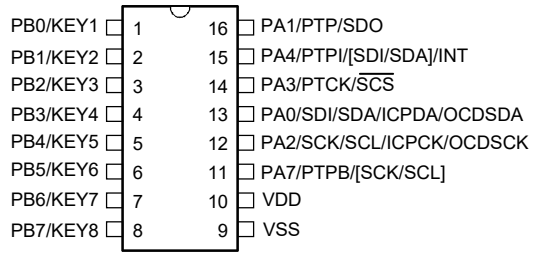
单片机型号	V_{DD}	程序存储器	数据存储器	EEPROM	I/O	外部中断
BS83B08C	2.2V~5.5V	2K×16	288×8	64×8	14	1
BS83B12C	2.2V~5.5V	2K×16	512×8	64×8	18	1
BS83B16C	2.2V~5.5V	2K×16	512×8	64×8	22	1

单片机型号	TM 模块	时基	堆栈	触控按键	SIM (SPI/I^2C)	封装
BS83B08C	10-bit PTM×1	1	6	8	1	16NSOP/SSOP/QFN
BS83B12C	10-bit PTM×1	1	6	12	1	20SOP/SSOP/QFN
BS83B16C	10-bit PTM×1	1	6	16	1	24SOP/SSOP/QFN

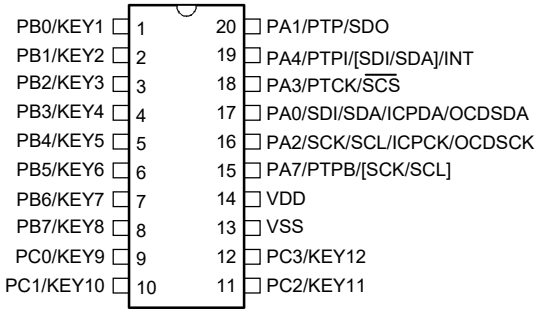
方框图



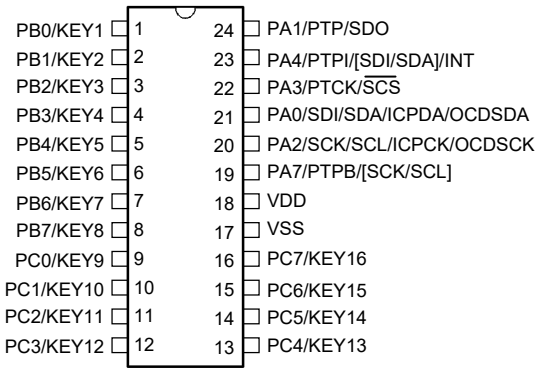
引脚图



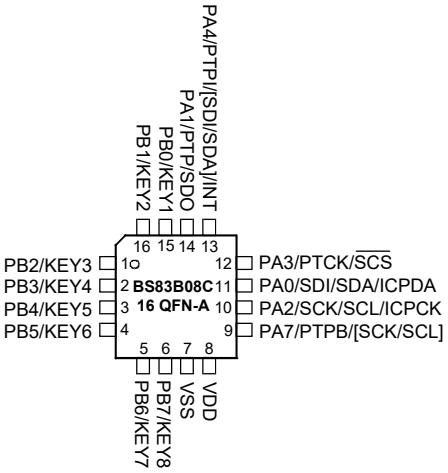
BS83B08C/BS83BV08C
16 NSOP-A/SSOP-A

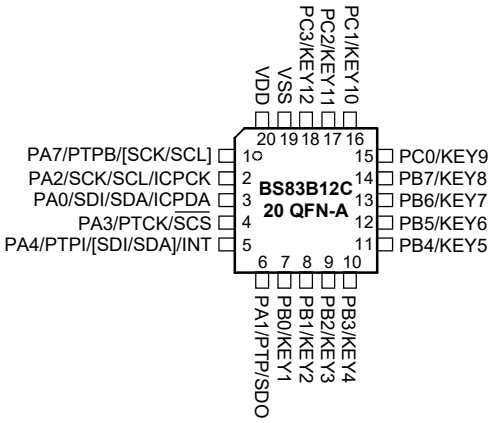


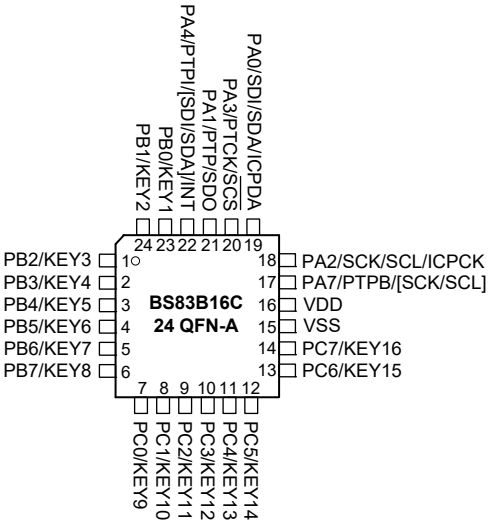
BS83B12C/BS83BV12C
20 SOP-A/SSOP-A



BS83B16C/BS83BV16C
24 SOP-A/SSOP-A







注：1. 括号内的引脚表示非默认引脚的位置。详细信息可以参考相关的章节。
2. 若共用脚同时有多种输出，“/”号右侧的引脚名具有更高的优先级。
3. OCDSCK 和 OCDSDA 引脚为 OCDS 专用引脚，仅存在于 BS83BVxxC 中，BS83BVxxC 是 BS83BxxC 的 OCDS EV 芯片。

引脚说明

除了电源引脚外，该系列单片机的所有引脚都以它们的端口名称进行标注，例如 PA0、PA1 等，用于描述这些引脚的数字输入 / 输出功能。然而，这些引脚也与其它功能共用，如触控按键功能，定时器模块引脚等。每个引脚的功能如下表所述，而引脚配置的详细内容见规格书其它章节。

BS83B08C

引脚名称	功能	OPT	I/T	O/T	描述
PA0/SDI/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
	SDI	SIMC0 PXRM	ST	—	SPI 串行数据输入
	SDA	SIMC0 PXRM	ST	NMOS	I^2C 数据引脚
	ICPDA	—	ST	CMOS	ICP 地址 / 数据引脚
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址引脚，仅用于 EV 芯片
PA1/PTP/SDO	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
	PTP	PTMC0 PTMC1 PXRM	—	CMOS	PTM 输出
	SDO	SIMC0	—	CMOS	SPI 串行数据输出
PA2/SCK/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
	SCK	SIMC0 PXRM	ST	CMOS	SPI 串行时钟
	SCL	SIMC0 PXRM	ST	NMOS	I^2C 时钟引脚
	ICPCK	—	ST	—	ICP 时钟引脚
	OCDSCK	—	ST	—	OCDS 时钟引脚，仅用于 EV 芯片
PA3/PTCK/$\overline{SCS}$	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
	PTCK	PTMC0	ST	—	PTM 时钟输入
	$\overline{SCS}$	SIMC0	ST	CMOS	SPI 从机选择引脚
PA4/PTPI/ [SDI/SDA]/INT	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
	PTPI	PTMC0 PTMC1	ST	—	PTM 捕捉输入
	SDI	SIMC0 PXRM	ST	—	SPI 串行数据输入
	SDA	SIMC0 PXRM	ST	NMOS	I^2C 数据引脚
	INT	INTC0 INTEG	ST	—	外部中断输入

引脚名称	功能	OPT	I/T	O/T	描述
PA7/PTPB/[SCK/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
	PTPB	PXRM	–	CMOS	PTM 反相输出
	SCK	SIMC0 PXRM	ST	CMOS	SPI 串行时钟
	SCL	SIMC0 PXRM	ST	NMOS	I^2C 时钟引脚
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
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
VDD	VDD	—	PWR	—	正电源电压
VSS	VSS	—	PWR	—	负电源电压，接地

注：I/T：输入类型；
OPT：通过寄存器选项来配置；
ST：施密特触发输入；
NMOS：NMOS 输出；
NSI：非标准输入。
O/T：输出类型；
PWR：电源；
CMOS：CMOS 输出；
AN：模拟信号；

BS83B12C

引脚名称	功能	OPT	I/T	O/T	说明
PA0/SDI/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
	SDI	SIMC0 PXRM	ST	—	SPI 串行数据输入
	SDA	SIMC0 PXRM	ST	NMOS	I^2C 数据引脚
	ICPDA	—	ST	CMOS	ICP 地址 / 数据引脚
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址引脚，仅用于 EV 芯片
PA1/PTP/SDO	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
	PTP	PTMC0 PTMC1 PXRM	—	CMOS	PTM 输出
	SDO	SIMC0	—	CMOS	SPI 串行数据输出

引脚名称	功能	OPT	I/T	O/T	说明
PA2/SCK/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
	SCK	SIMC0 PXRM	ST	CMOS	SPI 串行时钟
	SCL	SIMC0 PXRM	ST	NMOS	I^2C 时钟引脚
	ICPCK	—	ST	—	ICP 时钟引脚
	OCDSCK	—	ST	—	OCDS 时钟引脚，仅用于 EV 芯片
PA3/PTCK/$\overline{SCS}$	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
	PTCK	PTMC0	ST	—	PTM 时钟输入
	$\overline{SCS}$	SIMC0	ST	CMOS	SPI 从机选择引脚
PA4/PTPI/ [SDI/SDA]/INT	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
	PTPI	PTMC0 PTMC1	ST	—	PTM 捕捉输入
	SDI	SIMC0 PXRM	ST	—	SPI 串行数据输入
	SDA	SIMC0 PXRM	ST	NMOS	I^2C 数据引脚
	INT	INTC0 INTEG	ST	—	外部中断输入
PA7/PTPB/ [SCK/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
	PTPB	PXRM	–	CMOS	PTM 反相输出
	SCK	SIMC0 PXRM	ST	CMOS	SPI 串行时钟
	SCL	SIMC0 PXRM	ST	NMOS	I^2C 时钟引脚
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
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
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
VDD	VDD	—	PWR	—	正电源电压
VSS	VSS	—	PWR	—	负电源电压，接地

注：I/T：输入类型；　O/T：输出类型；　PWR：电源；
OPT：通过寄存器选项来配置；　NSI：非标准输入；
ST：施密特触发输入；　CMOS：CMOS 输出；
NMOS：NMOS 输出；　AN：模拟信号。

BS83B16C

引脚名称	功能	OPT	I/T	O/T	说明
PA0/SDI/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
	SDI	SIMC0 PXRM	ST	—	SPI 串行数据输入
	SDA	SIMC0 PXRM	ST	NMOS	I²C 数据引脚
	ICPDA	—	ST	CMOS	ICP 地址 / 数据引脚
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址引脚，仅用于 EV 芯片
PA1/PTP/SDO	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
	PTP	PTMC0 PTMC1 PXRM	—	CMOS	PTM 输出
	SDO	SIMC0	—	CMOS	SPI 串行数据输出
PA2/SCK/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
	SCK	SIMC0 PXRM	ST	CMOS	SPI 串行时钟
	SCL	SIMC0 PXRM	ST	NMOS	I²C 时钟引脚
	ICPCK	—	ST	—	ICP 时钟引脚
	OCDSCK	—	ST	—	OCDS 时钟引脚，仅用于 EV 芯片
PA3/PTCK/$\overline{SCS}$	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
	PTCK	PTMC0	ST	—	PTM 时钟输入
	$\overline{SCS}$	SIMC0	ST	CMOS	SPI 从机选择引脚
PA4/PTPI/[SDI/SDA]/INT	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
	PTPI	PTMC0 PTMC1	ST	—	PTM 捕捉输入
	SDI	SIMC0 PXRM	ST	—	SPI 串行数据输入
	SDA	SIMC0 PXRM	ST	NMOS	I²C 数据引脚
	INT	INTC0 INTEG	ST	—	外部中断输入
PA7/PTPB/[SCK/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
	PTPB	PXRM	–	CMOS	PTM 反相输出
	SCK	SIMC0 PXRM	ST	CMOS	SPI 串行时钟
	SCL	SIMC0 PXRM	ST	NMOS	I²C 时钟引脚

引脚名称	功能	OPT	I/T	O/T	说明
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
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
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
PC4/KEY13~ PC7/KEY16	PC4~PC7	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	KEY13~KEY16	TKM3C1	NSI	—	触控按键输入
VDD	VDD	—	PWR	—	正电源电压
VSS	VSS	—	PWR	—	负电源电压，接地

注：I/T：输入类型；
OPT：通过寄存器选项来配置；
ST：施密特触发输入；
NMOS：NMOS 输出；
NSI：非标准输入。
O/T：输出类型；
PWR：电源；
CMOS：CMOS 输出；
AN：模拟信号；

极限参数

电源供应电压 ..... V_{SS}-0.3V~V_{SS}+6.0V
输入电压 ..... V_{SS}-0.3V~V_{DD}+0.3V
储存温度 ..... -60°C~150°C
工作温度 ..... -40°C~85°C
I_{OH} 总电流 ..... -80mA
I_{OL} 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

以下表格中参数测量结果可能受多个因素影响，如振荡器类型、工作电压、工作频率、引脚负载状况、温度和程序指令等等。

工作电压特性

Ta=-40˚C~85˚C

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 – HIRC	f_{SYS}=8MHz	2.2	—	5.5	V
		f_{SYS}=12MHz	2.7	—	5.5	
		f_{SYS}=16MHz	3.3	—	5.5	
	工作电压 – LIRC	f_{SYS}=32kHz	2.2	—	5.5	V

待机电流特性

Ta=25˚C

符号	待机模式	测试条件		最小	典型	最大	最大 @85˚C	单位
		V_{DD}	条件					
I_{STB}	SLEEP 模式	2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3	3.6	
		5V		—	3	5	6	
	IDLE0 模式 – LIRC	2.2V	f_{SUB} on	—	2.4	4	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	IDLE1 模式 – HIRC	2.2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	432	600	720	μA
		3V		—	540	750	900	
		5V		—	800	1200	1440	
		3.3V	f_{SUB} on, f_{SYS}=16MHz	—	1.1	1.6	1.9	mA
		5V		—	1.4	2.0	2.4	

注：当使用该表格电气特性数据时，以下几点需注意：

1. 任何数字输入都设置为非浮空的状态。
2. 所有测量都在无负载且所有外围功能关闭的条件下进行。
3. 无直流电流路径。
4. 所有待机电流数值都是在 HALT 指令执行后测得，因此 HALT 后停止执行所有指令。

工作电流特性

Ta=25°C

符号	工作模式	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{DD}	低速模式 – LIRC	2.2V	f_{SYS}=32kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	快速模式 – HIRC	2.2V	f_{SYS}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	
		2.7V	f_{SYS}=12MHz	—	1.0	1.4	mA
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	
		3.3V	f_{SYS}=16MHz	—	3.0	4.5	mA
		5V		—	4.0	6.0	

注：当使用该表格电气特性数据时，以下几点需注意：

1. 任何数字输入都设置为非浮空的状态。
2. 所有测量都在无负载且所有外围功能关闭的条件下进行。
3. 无直流电流路径。
4. 所有工作电流数据都是通过执行连续 NOP 指令程序回路进行测量。

交流电气特性

以下表格中参数测量结果可能受多个因素影响，如振荡器类型、工作电压、工作频率和温度等等。

内部高速 RC 振荡器 HIRC 频率精准度

程序烧录时，烧录器会依据用户选择的 HIRC 频率和工作电压 (3V 或 5V) 对 HIRC 进行频率精准度调整。

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{HIRC}	通过烧录器调整后的 8MHz HIRC 频率	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	
	通过烧录器调整后的 12MHz HIRC 频率	3V/5V	25°C	-1%	12	+1%	MHz
			-40°C~85°C	-2%	12	+2%	
		2.7V~5.5V	25°C	-2.5%	12	+2.5%	
			-40°C~85°C	-3%	12	+3%	
f_{HIRC}	通过烧录器调整后的 16MHz HIRC 频率	5V	25°C	-1%	16	+1%	MHz
			-40°C~85°C	-2%	16	+2%	
		3.3V~5.5V	25°C	-2.5%	16	+2.5%	
			-40°C~85°C	-3%	16	+3%	

注：1. 烧录器可在 3V/5V 这两个可选的固定电压下对 HIRC 频率进行调整，在此提供 V_{DD}=3V/5V 时的参

数值。

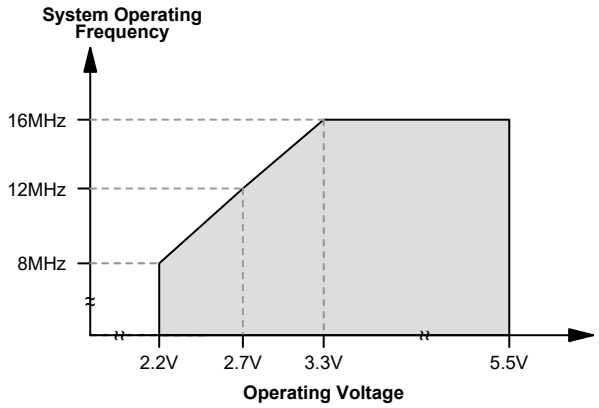
2. 3V/5V 表格列下面提供的是全压条件下的参数值。建议电压固定在 3V 微调应用电压范围从 2.2V 至 3.6V 和电压固定在 5V 微调应用电压范围从 3.3V 到 5.5V 范围中。
3. 表格中提供的最小和最大误差值仅在对应的烧录器调整频率下有效。当烧录器已对所选的频率进行调整，此后再通过程序中振荡器控制位将其频率改为其它值时，频率误差范围将增加到 ±20%。

内部低速振荡器电气特性 – LIRC

Ta=25°C, 除非另有规定

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{LIRC}	LIRC 振荡器频率	2.2V~5.5V	25°C	-10%	32	+10%	kHz
			-40°C~85°C	-50%	32	+60%	
t_{START}	LIR 启动时间	—	—	—	—	500	μs

工作频率电气特性曲线图



系统上电时间电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
t_{SST}	系统启动时间 (从 f_{SYS} off 的状态下唤醒)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	系统启动时间 (从 f_{SYS} on 的状态下唤醒)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SUB}
	系统速度切换时间 (快速模式 → 低速模式或 低速模式 → 快速模式)	—	f_{HIRC} off → on	—	16	—	t_{HIRC}
t_{RSTD}	系统复位延迟时间 (上电复位或 LVR 硬件复位)	—	RR_{POR}=5V/ms	42	48	54	ms
	系统复位延迟时间 (LVRC/WDTC 软件复位)	—	—				
	系统复位延迟时间 (WDT 溢出复位)	—	—	14	16	18	ms

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
t_{SRESET}	软件复位最小延迟脉宽	—	—	45	90	120	μs

注：1. 系统启动时间里提到的 f_{SYS} on/off 状态取决于工作模式类型以及所选的系统时钟振荡器。更多相关细节请参考系统工作模式章节。

2. t_{HIRC}、t_{SYS} 等符号所表示的时间单位，是对应频率值的倒数，相关频率值在前面表格有说明。例如，$t_{HIRC}=1/f_{HIRC}$，$t_{SYS}=1/f_{SYS}$ 等等。

3. LIRC 被选择作为系统时钟源且在休眠模式下 LIRC 关闭，则上面表格中对应 t_{SST} 数值还需加上 LIRC 频率表格里提供的 LIRC 启动时间 t_{START}。

4. 系统速度切换时间实际上是指新使能的振荡器的启动时间。

输入 / 输出口电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{IL}	I/O 口或输入引脚低电平输入电压	5V	—	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	
V_{IH}	I/O 口或输入引脚高电平输入电压	5V	—	3.5	—	5.0	V
		—		$0.8V_{DD}$	—	V_{DD}	
I_{OL}	I/O 口灌电流	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	I/O 口源电流	3V	$V_{OH}=0.9V_{DD}$	-4	-8	—	mA
		5V		-8	-16	—	
R_{PH}	I/O 口上拉电阻（注）	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I_{LEAK}	输入漏电流	5V	$V_{IN}=V_{DD}$ 或 $V_{IN}=V_{SS}$	—	—	±1	μA
t_{TCK}	TM 时钟输入引脚最小脉宽	—	—	0.3	—	—	μs
t_{TPI}	TM 捕捉输入引脚最小脉宽	—	—	0.3	—	—	μs

注：R_{PH} 内部上拉电阻值的计算方法是：将其接地并使能输入引脚的上拉电阻选项，然后在特定电源电压下测量输入灌电流，用电压值除以测得的电流值从而得到此上拉电阻值。

存储器电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{RW}	读 / 写工作电压	—	—	V_{DDmin}	—	V_{DDmax}	V
程序存储器 / EEPROM 存储器							
t_{DEW}	擦除 / 写周期时间 – Flash 程序存储器	—	—	—	2	3	ms
	写周期时间 – 数据 EEPROM 存储器	—	—	—	4	6	ms
I_{DDPGM}	V_{DD} 电压下烧录 / 擦除电流	—	—	—	—	5.0	mA

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
E_P	存储单元耐久性 – Flash 程序存储器	—	—	10K	—	—	E/W
	存储单元耐久性 – 数据 EEPROM 存储器	—	—	100K	—	—	E/W
t_{RETD}	ROM 数据保存时间	—	Ta = 25°C	—	40	—	Year
RAM 数据存储器							
V_{DR}	RAM 数据保存电压	—	—	1.0	—	—	V

注：“E/W”表示擦 / 写次数。

LVR 电气特性

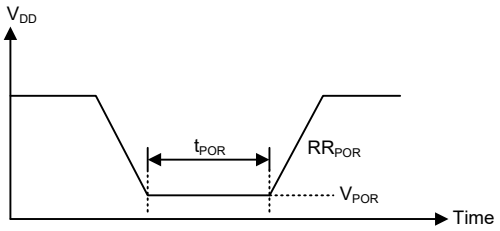
Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{LVR}	低电压复位电压	—	LVR 使能，电压选择 2.1V	-5%	2.1	+5%	V
		—	LVR 使能，电压选择 2.55V		2.55		
		—	LVR 使能，电压选择 3.15V		3.15		
		—	LVR 使能，电压选择 3.8V		3.8		
I_{LVR}	使用 LVR 的额外功耗	3V	LVR 除能 →LVR 使能	—	15	25	μA
		5V		—	20	30	
t_{LVR}	产生 LVR 复位的低电压最短保持时间	—	—	120	240	480	μs

上电复位特性

Ta = 25°C

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



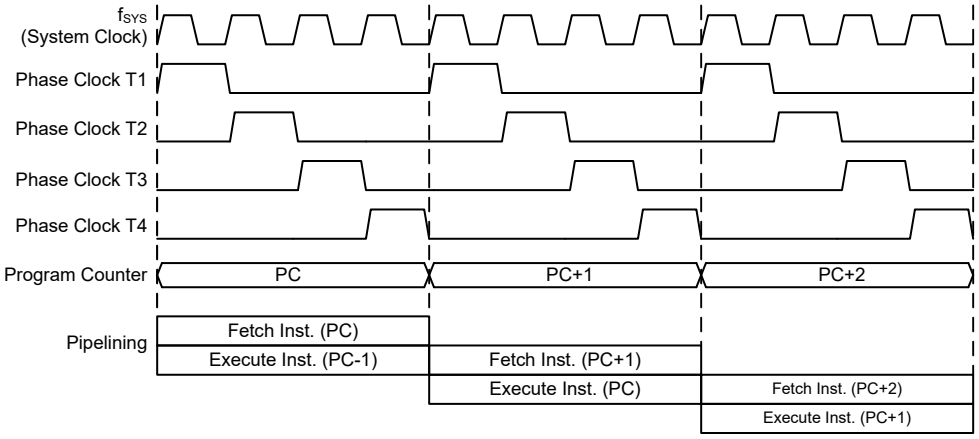
系统结构

内部系统结构是 Holtek 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和 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 I/O 控制系统时，仅需要少数的外部器件。使得该系列单片机适用于低成本和大量生产的控制应用。

时序和流水线结构

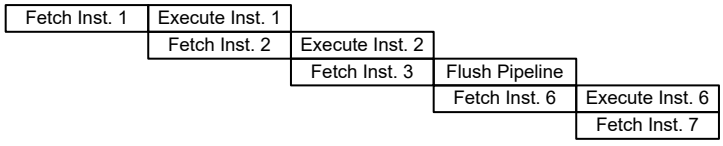
主系统时钟由 HIRC 或 LIRC 振荡器提供，它被细分为 T1~T4 四个内部产生的非重叠时序。在 T1 时间，程序计数器自动加一并抓取一条新的指令。剩下的时间 T2~T4 完成译码和执行功能，因此，一个 T1~T4 时钟周期构成一个指令周期。虽然指令的抓取和执行发生在连续的指令周期，但单片机流水线结构会保证指令在一个指令周期内被有效执行。除非程序计数器的内容被改变，如子程序的调用或跳转，在这种情况下指令将需要多一个指令周期的时间去执行。

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。



系统时序和流水线

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



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。只有较低的 8 位，即所谓的程序计数器低字节寄存器 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地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。

单片机型号	程序计数器	
	高字节	低字节 (PCL)
BS83B08C	PC10~PC8	PCL7~PCL0
BS83B12C	PC10~PC8	
BS83B16C	PC10~PC8	

程序计数器

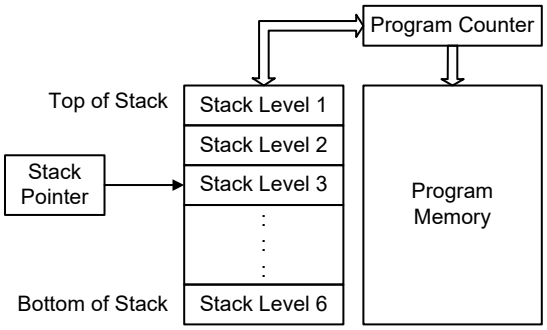
程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。程序计数器的低字节可由程序直接进行读取，PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

堆栈

堆栈是一个特殊的存储空间，用来存储程序计数器中的内容。该系列单片机有 6 层堆栈，堆栈既不是数据部分也不是程序空间部分，而且它既不是可读取也不是可写入的。当前层由堆栈指针 (SP) 加以指示，堆栈指针同样也是不可读写的。在子程序调用或中断响应服务时，程序计数器的内容被压入到堆栈中。当子程序或中断响应结束时，返回指令 (RET 或 RETI) 使程序计数器从堆栈中重新得到它保存的值。当一个芯片复位后，堆栈指针将指向堆栈顶部。

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

若堆栈溢出，则首个存入堆栈的程序计数器数据将会丢失。



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操作，并将结果存储在指定的寄存器，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：

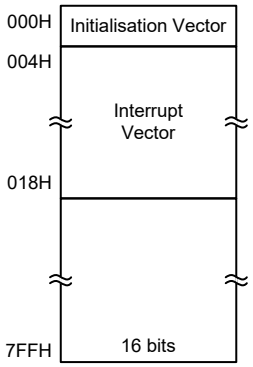
- 算术运算：ADD，ADDM，ADC，ADCM，SUB，SUBM，SBC，SBCM，DAA
- 逻辑运算：AND，OR，XOR，ANDM，ORM，XORM，CPL，CPLA
- 移位运算：RRA，RR，RRCA，RRC，RLA，RL，RLCA，RLC
- 递增和递减：INCA，INC，DECA，DEC
- 分支判断：JMP，SZ，SZA，SNZ，SIZ，SDZ，SIZA，SDZA，CALL，RET，RETI

Flash 程序存储器

程序存储器用来存放用户代码即储存程序。程序存储器为 Flash 类型意味着可以多次重复编程，方便用户使用同一芯片进行程序的修改。使用适当的单片机编程工具，此系列 Flash 单片机提供用户灵活便利的调试方法和项目开发规划及更新。

结构

该系列单片机的程序存储器的容量为 2K×16 位，程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口。数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。



程序存储器结构

特殊向量

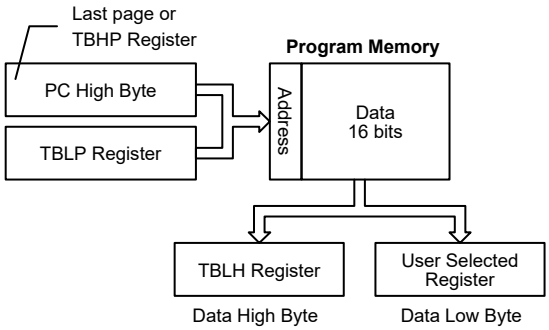
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

查表

程序存储器中的任何地址都可以定义成一个表格，以便储存固定的数据。使用表格时，表格指针必须先行设定，其方式是将表格的地址放在表格指针寄存器 TBLP 和 TBHP 中。这些寄存器定义表格总的地址。

在设定完表格指针后，表格数据可以使用“TABRD [m]”或“TABRDL [m]”指令分别从程序存储器查表读取。当这个指令执行时，程序存储器中表格数据低字节，将被传送到使用者所指定的数据存储器 [m]，程序存储器中表格数据的高字节，则被传送到 TBLH 特殊寄存器。

下图是查表中寻址 / 数据流程：



查表范例

以下范例说明表格指针和表格数据如何被定义和执行。这个例子使用的表格数据用 ORG 伪指令储存在存储器中。ORG 指令的值“700H”指向的地址是 2K 程序存储器中最后一页的起始地址。表格指针低字节寄存器的初始值设为 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 706H，即最后一页起始地址后的第六个地址。值得注意的是，假如“TABRD [m]”指令被使用，则表格指针指向 TBHP 指定的页。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。

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

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
mov tblp,a          ; to the last page or the page that tbhp pointed
mov a,07h           ; initialise high table pointer
mov tbhp,a
:
tabrd tempreg1      ; transfers value in table referenced by table pointer
                    ; data at program memory address "706H" 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 "705H" transferred to
                    ; tempreg2 and TBLH
                    ; in this example the data "1AH" is transferred to
```

```
                    ; tempreg1 and data "0FH" to register tempreg2
:
org 700h            ; sets initial address of program memory
dc  00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

在线烧录 – ICP

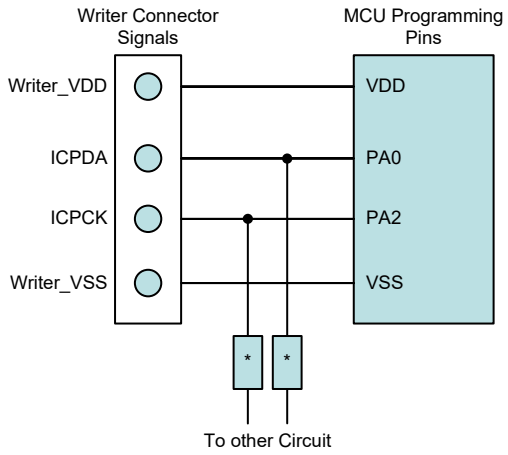
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。另外，Holtek 单片机提供 4 线接口的在线烧录方式。用户可将进行过烧录或未经过烧录的单片机芯片连同电路板一起制成，最后阶段进行程序的更新和程序的烧录，在无需去除或重新插入芯片的情况下方便地保持程序为最新版。

Holtek Flash MCU 与烧录器引脚对应表如下：

Holtek 烧录器引脚名称	MCU 在线烧录引脚名称	引脚说明
ICPDA	PA0	串行数据 / 地址输入 / 输出
ICPCK	PA2	串行时钟输入
VDD	VDD	电源
VSS	VSS	地

程序存储器可以通过 4 线的接口在线进行烧录。其中一个引脚用于数据串行下载或上传、另一个引脚用于串行时钟、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。

烧录过程中，烧录器会控制 PA0 和 PA2 引脚进行数据和时钟烧录，用户必须确保 ICPDA 和 ICPCK 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。

片上调试 – OCDS

EV 芯片 BS83BVxxC 用于 BS83BxxC 单片机仿真。此 EV 芯片提供片上调试功能 (OCDS) 用于开发过程中的单片机调试。除了片上调试功能方面，EV 芯片和实际 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 芯片对实际 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 芯片进行调试时，实际单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用

作 Flash 存储器烧录引脚。关于 OCDS 功能的详细描述，请参考“Holtek e-Link for 8-bit MCU OCDS 使用手册”文件。

Holtek e-Link 引脚名称	EV 芯片 OCDS 引脚名称	引脚说明
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更改的 8 位 RAM 内部存储器，用来储存临时数据。

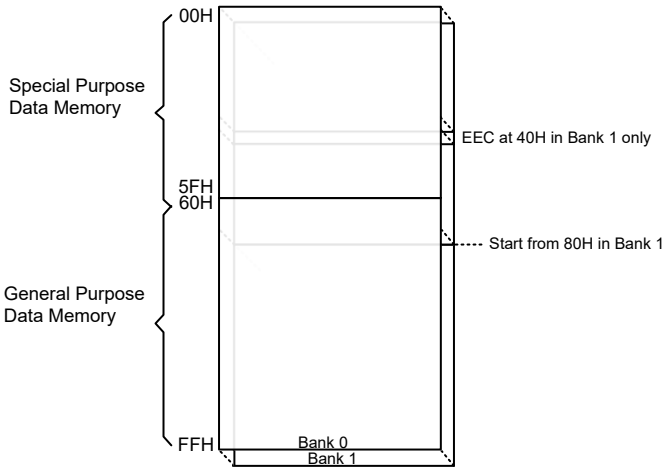
结构

数据存储器分为两个部分，第一部分是特殊功能数据存储器。这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二部分数据存储器是做一般用途使用，都可在程序控制下进行读取和写入。

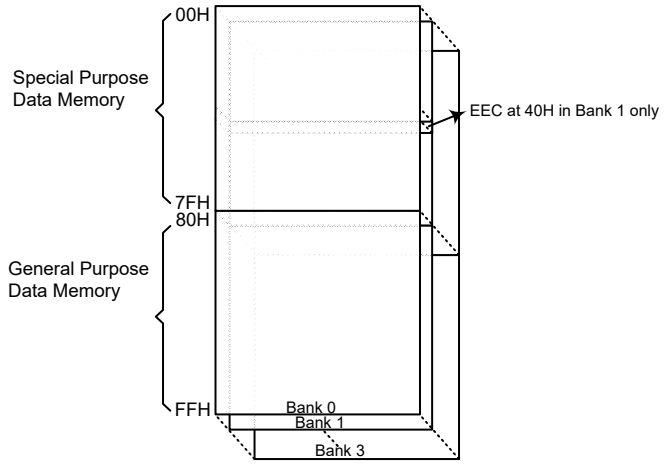
总的数据存储器被分为两个 Bank。大部分特殊功能数据寄存器均可在所有 Bank 被访问，除了 EEC 寄存器只位于 Bank 1 的“40H”地址。切换不同区域可通过设置区域指针实现。单片机的数据存储器的起始地址是“00H”。

单片机型号	特殊功能数据存储器		通用数据存储器	
	所在 Bank	Bank：地址	容量	Bank：地址
BS83B08C	0,1	0: 00H~5FH 1: 00H~7FH	288×8	0: 60H~FFH 1: 80H~FFH
BS83B12C	0~3	0: 00H~7FH 1: 00H~7FH 2: 00H~7FH 3: 00H~7FH	512×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH
BS83B16C	0~3	0: 00H~7FH 1: 00H~7FH 2: 00H~7FH 3: 00H~7FH	512×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH

数据存储器概要



数据存储器结构 – **BS83B08C**



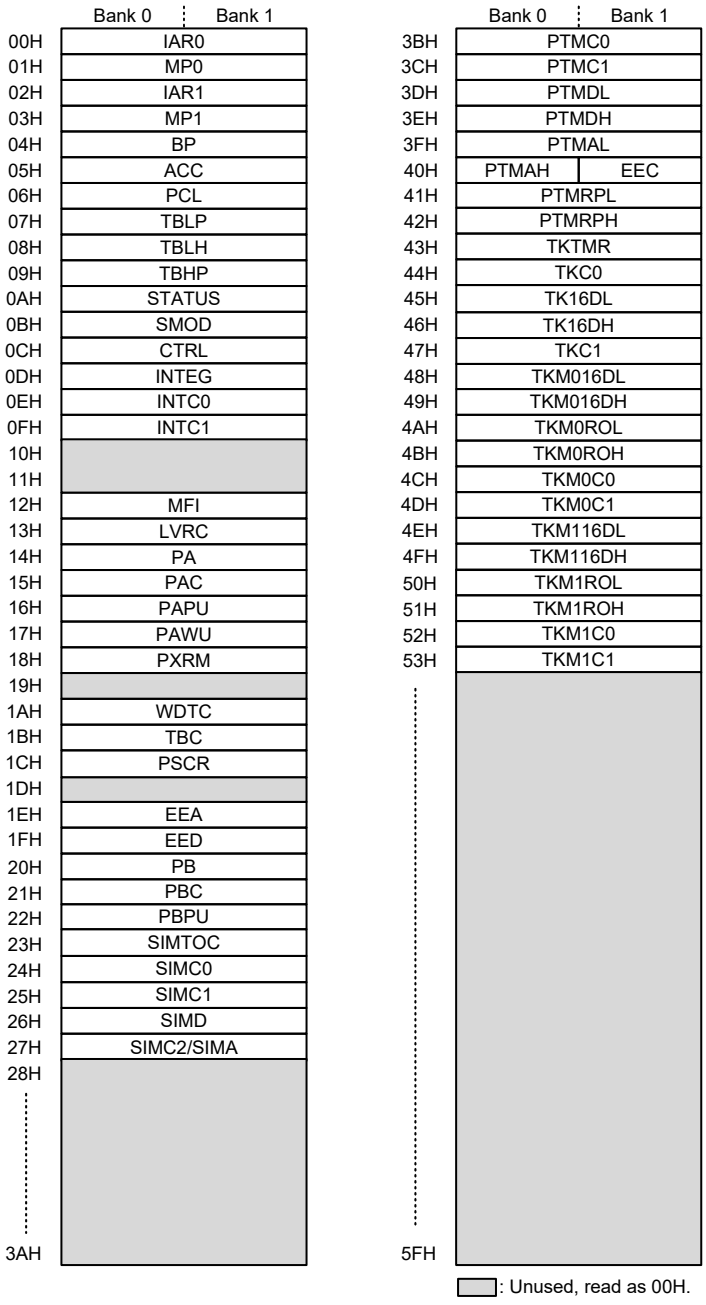
数据存储器结构 – **BS83B12C/BS83B16C**

通用数据存储器

所有的单片机程序需要一个读 / 写的存储区，让临时数据可以被储存和再使用，该 RAM 区域就是通用数据存储器。这个数据存储区可让使用者进行读取和写入的操作。使用位操作指令可对个别的位做置位或复位的操作，较大地方便了用户在数据存储器内进行位操作。

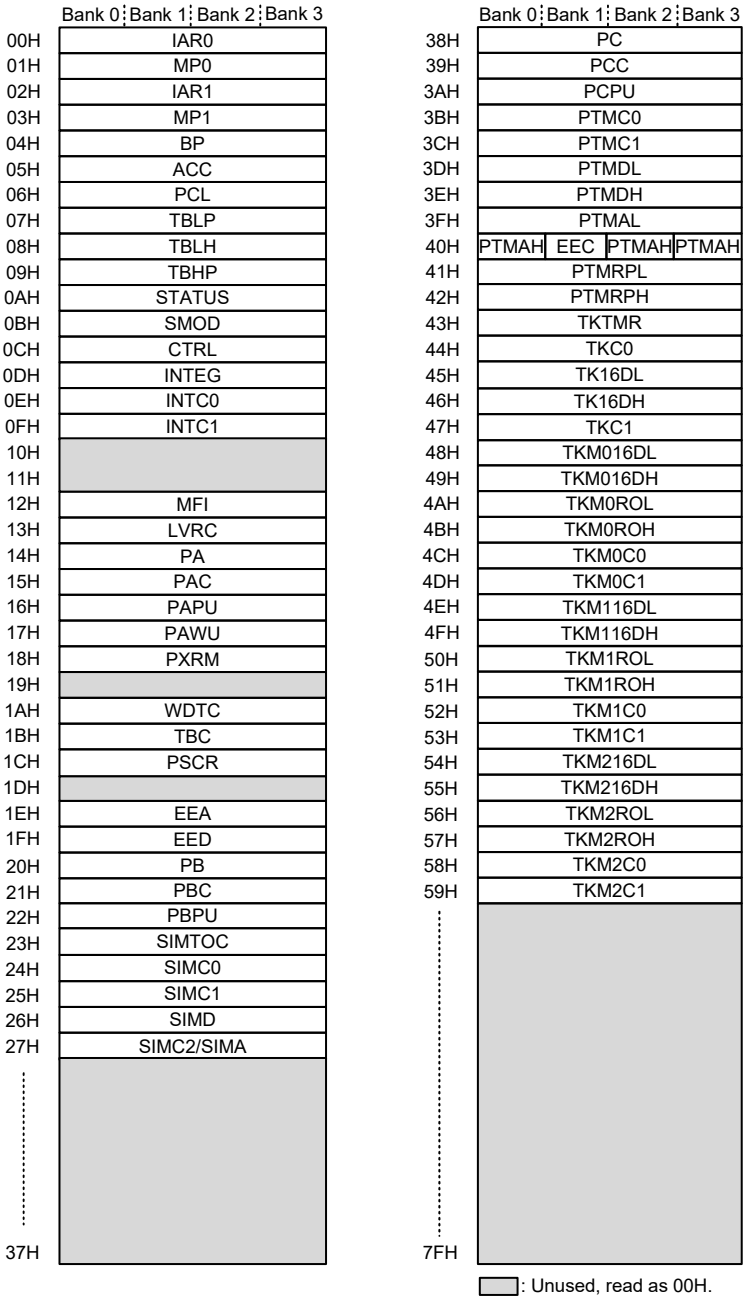
特殊功能数据存储器

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

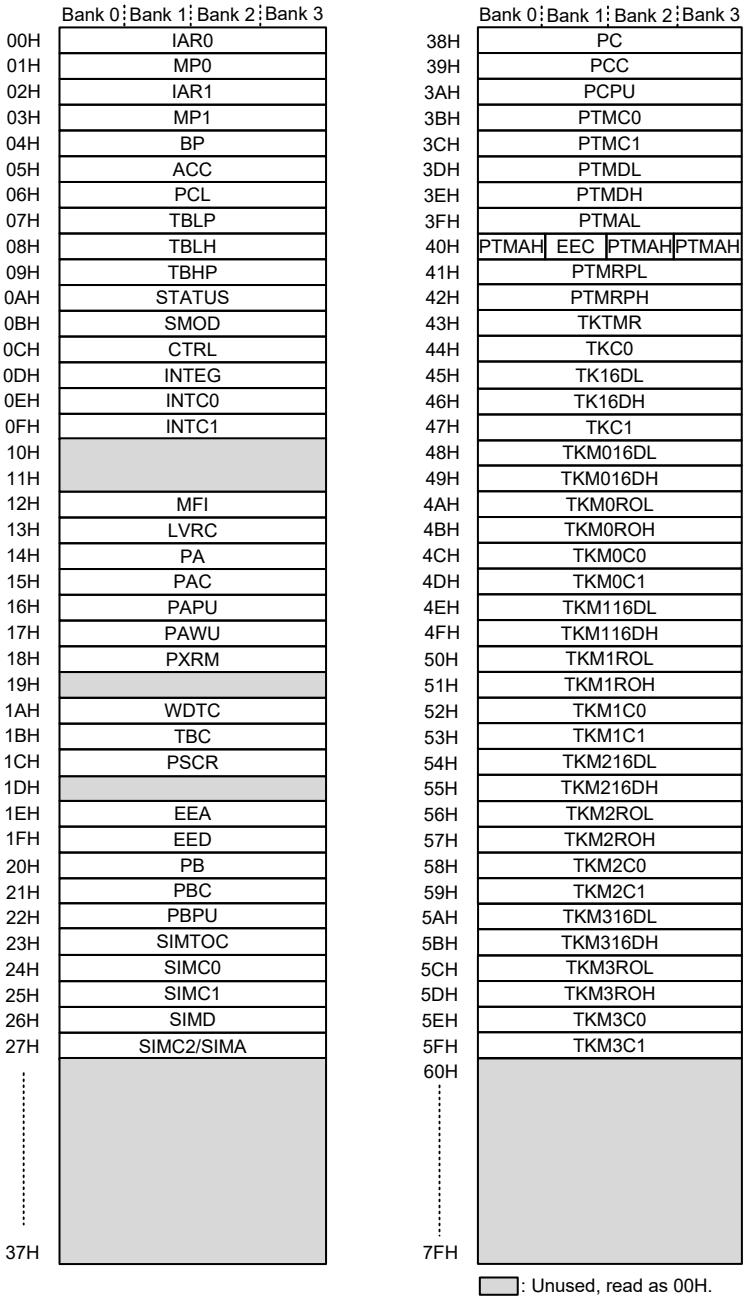


注：该单片机特殊功能数据存储器 Bank 1 的地址范围为 00H~7FH，60H~7FH 未使用，读为 00H。

特殊功能数据存储器结构 – **BS83B08C**



特殊功能数据存储器结构 – **BS83B16C**



特殊功能数据存储器结构 – **BS83B16C**

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章节描述，但有几个寄存器需在此章节单独描述。

间接寻址寄存器 – IAR0, IAR1

间接寻址寄存器IAR0和IAR1的地址虽位于数据存储区，但不同于普通寄存器，它们没有实际的物理地址。与定义实际存储器地址的直接存储器寻址不同，间接寻址是使用间接寻址寄存器和存储器指针来执行存储器数据操作。在间接寻址寄存器 IAR0 和 IAR1 上的任何动作，将对间接寻址寄存器 MP0 和 MP1 所指定的存储器地址产生对应的读 / 写操作。它们总是成对出现，IAR0 和 MP0 可以访问 Bank 0，而 IAR1 和 MP1 可以访问任何 Bank。因为这些间接寻址寄存器不是实际存在的，直接读取将返回“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般被操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由存储器指针所指定的地址。MP0、IAR0 用于访问 Bank 0，而 MP1 和 IAR1 可通过 BP 寄存器访问所有的 Bank。直接寻址仅可以用在 Bank 0 中，所有 Bank 都可使用 MP1 和 IAR1 进行间接寻址。

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

间接寻址程序举例

```
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:
```

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。

存储区指针 – BP

数据存储器被分为几个部分。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。BP 指针的 Bit 0~1 用于选择数据存储区的 Bank 0~3。

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，选择的数据存储器 Bank 不会改变。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 Bank 1~3，则必须要使用间接寻址方式。

- **BP 寄存器 – BS83B08C**

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**：数据存储区选择位
0：Bank 0
1：Bank 1

- **BP 寄存器 – BS83B12C/BS83B16C**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	DMBP1	DMBP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未定义，读为“0”

Bit 1~0　**DMBP0**：数据存储区选择位
00：Bank 0
01：Bank 1
10：Bank 2
11：Bank 3

累加器 – ACC

对任何单片机来说，累加器是相当重要的，且与 ALU 所完成的运算有密切关系，所有 ALU 得到的运算结果都会暂时存在 ACC 累加器里。若没有累加器，ALU 必须在每次进行如加法、减法和移位的运算时，将结果写入到数据存储器，这样会造成程序编写和时间的负担。另外数据传送也常常牵涉到累加器的临时储存功能，例如在使用者定义的一个寄存器和另一个寄存器之间传送数据时，由于两寄存器之间不能直接传送数据，因此必须通过累加器来传送数据。

程序计数器低字节寄存器 – PCL

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

表格寄存器 – TBLP, TBHP, TBLH

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

状态寄存器 – STATUS

这 8 位的状态寄存器由零标志位 (Z)、进位标志位 (C)、辅助进位标志位 (AC)、溢出标志位 (OV)、暂停标志位 (PDF) 和看门狗定时器溢出标志位 (TO) 组成。这些算术 / 逻辑操作和系统运行标志位是用来记录单片机的运行状态。

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

Z、OV、AC 和 C 标志位通常反映最近运算的状态。

- C：当加法运算的结果产生进位，或减法运算的结果没有产生借位时，则 C 被置位，否则 C 被清零，同时 C 也会被带进位的移位指令所影响。
- 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，状态寄存器不会自动压入到堆栈保存。假如状态寄存器的内容是重要的且子程序可能改变状态寄存器的话，则需谨慎的去做正确的储存。

- **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”指令后
1：看门狗溢出发生

Bit 4　**PDF**：暂停标志位
0：系统上电或执行“CLR WDT”指令后
1：执行“HALT”指令

Bit 3　**OV**：溢出标志位
0：无溢出
1：运算结果高两位的进位状态异或结果为 1

Bit 2　**Z**：零标志位
0：算术或逻辑运算结果不为 0
1：算术或逻辑运算结果为 0

Bit 1　**AC**：辅助进位标志位
0：无辅助进位
1：在加法运算中低四位产生了向高四位进位，或减法运算中低四位不发生从高四位借位

Bit 0　**C**：进位标志位
0：无进位
1：如果在加法运算中结果产生了进位，或在减法运算中结果不发生借位

C 标志位也受循环移位指令的影响。

EEPROM 数据存储器

该系列单片机的一个特性是内建 EEPROM 数据存储器。EEPROM 为电可擦可编程存储器，由于其非易失的存储结构，即使在电源掉电的情况下存储器内的数据仍然保存完好。这种存储区扩展了存储器空间，对设计者来说增加了许多新的应用机会。EEPROM 可以用来存储产品编号、校准值、用户特定数据、系统配置参数或其它产品信息等。EEPROM 的数据读取和写入过程也会变的更简单。

单片机型号	容量	地址
BS83B08C	64 × 8	00H ~ 3FH
BS83B12C		
BS83B16C		

EEPROM 数据存储器结构

该系列单片机的 EEPROM 数据存储器容量为 64×8 位。由于映射方式与程序存储器和数据存储器不同，因此不能像其它类型的存储器一样寻址。使用所有 Bank 中的地址寄存器和数据寄存器以及 Bank 1 中的一个控制寄存器，可以实现对 EEPROM 的单字节读写操作。

EEPROM 寄存器

有三个寄存器控制内部 EEPROM 数据存储器总的操作。地址寄存器 EEA、数据寄存器 EED 及控制寄存器 EEC。EEA 和 EED 可位于 Bank 0 中，它们能像其它位于 Bank0 的特殊功能寄存器一样直接被访问。EEC 位于 Bank 1 中，不能被直接访问，仅能通过 MP1 和 IAR1 进行间接读取或写入。由于 EEC 控制寄存器位于 Bank 1 中的“40H”，在 EEC 寄存器上的任何操作被执行前，MP1 必须先设为“40H”，BP 被设为“01H”。

寄存器名称	位							
	7	6	5	4	3	2	1	0
EEA	—	—	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM 寄存器列表

- **EEA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6　未定义，读为“0”

Bit 5~0　**EEA5~ EEA0：**数据 EEPROM 地址
数据 EEPROM 地址 bit 5~bit 0

- **EED 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 **D7~D0**：EEPROM 数据 bit 7~bit 0

- **EEC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	WREN	WR	RDEN	RD
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 **WREN**：数据 EEPROM 写使能位
0：除能
1：使能
此位为数据 EEPROM 写使能位，向数据 EEPROM 写操作之前需将此位置高。将此位清零时，则禁止向数据 EEPROM 写操作。

Bit 2 **WR**：EEPROM 写控制位
0：写周期结束
1：写周期有效
此位为数据 EEPROM 写控制位，由应用程序将此位置高将激活写周期。写周期结束后，硬件自动将此位清零。当 WREN 未先置高时，此位置高无效。

Bit 1 **RDEN**：数据 EEPROM 读使能位
0：除能
1：使能
此位为数据 EEPROM 读使能位，向数据 EEPROM 读操作之前需将此位置高。将此位清零时，则禁止向数据 EEPROM 读操作。

Bit 0 **RD**：EEPROM 读控制位
0：读周期结束
1：读周期有效
此位为数据 EEPROM 读控制位，由应用程序将此位置高将激活读周期。读周期结束后，硬件自动将此位清零。当 RDEN 未先置高时，此位置高无效。

注：1. 在同一条指令中 WREN、WR、RDEN 和 RD 不能同时置为“1”。WR 和 RD 不能同时置为“1”。

2. 确保 f_{SUB} 时钟在执行写动作前已稳定。

3. 确保写动作完成后才可改写 EEPROM 相关寄存器。

从 EEPROM 中读取数据

从 EEPROM 中读取数据，EEC 寄存器中的读使能位 RDEN 先置为高以使能读功能，EEPROM 中读取数据的地址要先放入 EEA 寄存器中。若 EEC 寄存器中的 RD 位被置高，一个读周期将开始。若 RD 位已置为高而 RDEN 位还未被设置则不能开始读操作。若读周期结束，RD 位将自动清除为“0”，数据可以从 EED 寄存器中读取。数据在其它读或写操作执行前将一直保留在 EED 寄存器中。应用程序将轮询 RD 位以确定数据可以有效地被读取。

写数据到 EEPROM

写数据至 EEPROM，EEPROM 中写入数据的地址要先放入 EEA 寄存器中，然后将要写入的数据放入 EED 寄存器中。EEC 寄存器中的写使能位 WREN 先置为高以使能写功能，然后立刻将 EEC 寄存器中 WR 位置为高，此时一个内部写周期才开始。这两条指令必须在两个指令周期内连续执行。在写操作执行之前需将总中断控制位 EMI 清零，在写周期开始后再置高。若 WR 位已置为高而 WREN 位还未被置高则不能开始写操作。由于控制 EEPROM 写周期是一个内部时钟，与单片机的系统时钟异步，所以数据写入 EEPROM 的时间将有所延迟。可通过轮询 EEC 寄存器中的 WR 位或判断 EEPROM 中断以侦测写周期是否完成。若写周期完成，WR 位将自动清除为“0”，通知用户数据已写入 EEPROM。因此，应用程序将轮询 WR 位以确定写周期是否结束。

写保护

防止误写入的写保护有以下几种。单片机上电后控制寄存器中的写使能位将被清除以杜绝任何写入操作。上电后 BP 将重置为“0”，这意味着数据存储区 Bank 0 被选中。由于 EEPROM 控制寄存器位于 Bank 1 中，这增加了对写操作的保护措施。在正常程序操作中确保控制寄存器中的写使能位被清除将能防止不正确的写操作。

EEPROM 中断

EEPROM 写周期结束后将产生 EEPROM 写中断，需先通过设置相关中断寄存器的 DEE 位使能 EEPROM 中断。当 EEPROM 写周期结束，DEF 请求标志位将被置位。若总中断和 EEPROM 中断使能且堆栈未满的情况下将跳转到相应的 EEPROM 中断向量中执行。当中断被响应，EEPROM 中断标志位 DEF 将自动复位。

编程注意事项

必须注意的是数据不会无意写入 EEPROM。在没有写动作时写使能位被正常清零可以增强保护功能。BP 指针也可以正常清零以阻止进入 EEPROM 控制寄存器存在的 Bank 1。尽管没有必要，写一个简单的读回程序以检查新写入的数据是否正确还是应该考虑的。

当 WREN 位被置为高以后，写数据位 WR 必须立刻置为高，以确保写循环正确执行。总中断位 EMI 在写循环开始前应当被清零，写循环开始后再将其使能。注意，单片机不应在 EEPROM 读或写操作完全完成之前进入空闲或休眠模式，否则 EEPROM 读或写操作将失败。

程序举例

从 EEPROM 中读取数据 – 轮询法

```
MOV  A, EEPROM_ADRES        ; user defined address
MOV  EEA, A
MOV  A, 040H                ; setup memory pointer MP1
MOV  MP1, A                 ; MP1 points to EEC register
MOV  A, 01H                 ; setup Bank Pointer
MOV  BP, A
SET  IAR1.1                 ; set RDEN bit, enable read operations
SET  IAR1.0                 ; start Read Cycle - set RD bit
BACK:
SZ   IAR1.0                 ; check for read cycle end
JMP  BACK
CLR  IAR1                   ; disable EEPROM read/if no more read operations
```

```
                          ; are required
CLR  BP
MOV  A, EED               ; move read data to register
MOV  READ_DATA, A
```

注：对于每一个读操作，即使地址是连续的，都必须重新设置地址寄存器，接着再将 RD 位置高开启一个读周期。

写数据到 EEPROM – 轮询法

```
MOV  A, EEPROM_ADRES      ; user defined address
MOV  EEA, A
MOV  A, EEPROM_DATA       ; user defined data
MOV  EED, A
MOV  A, 040H              ; setup memory pointer MP1
MOV  MP1, A               ; MP1 points to EEC register
MOV  A, 01H               ; setup Bank Pointer
MOV  BP, A
CLR  EMI
SET  IAR1.3               ; set WREN bit, enable write operations
SET  IAR1.2               ; start Write Cycle - set WR bit - executed
                          ; immediately after set WREN bit
SET  EMI
BACK:
SZ   IAR1.2               ; check for write cycle end
JMP  BACK
CLR  IAR1                 ; disable EEPROM read/write
CLR  BP
```

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方面可以达到较佳的优化。振荡器选择和操作是通过配置寄存器完成的。

振荡器概述

振荡器除了作为系统时钟源，还作为看门狗定时器和时基中断的时钟源。完全集成的两个内部振荡器不需要任何外围器件。它们提供的高速和低速系统振荡器具有较宽的频率范围。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该系列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。

类型	名称	频率
内部高速 RC	HIRC	8/12/16MHz
内部低速 RC	LIRC	32kHz

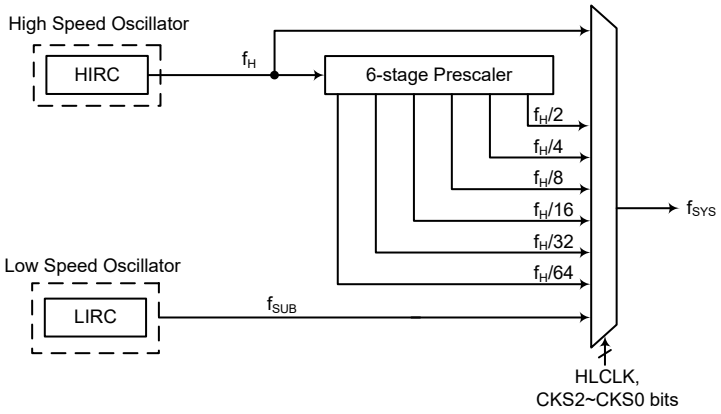
振荡器类型

系统时钟配置

该系列单片机有两个振荡器来产生系统时钟，包括一个高速振荡器和一个低速振荡器。高速振荡器为内部 8/12/16MHz RC 振荡器 HIRC，低速振荡器为内部 32kHz RC 振荡器 LIRC。使用高速或低速振荡器作为系统时钟的选择是通过设置 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位决定的，系统时钟可动态选择。

高速或低速振荡器的实际时钟源经由寄存器选择。低速或高速系统时钟频率由

SMOD 寄存器的 HLCLK 位及 CKS2~CKS0 位决定的。请注意，两个振荡器必须做出选择，即一个高速和一个低速振荡器。



系统时钟配置

内部高速 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，不需要其它外部器件。内部 RC 振荡器具有三种固定的频率：8MHz、12MHz 和 16MHz，可通过 CTRL 寄存器中的 HIRCS1 和 HIRCS0 位共同选择。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因电源电压、温度以及芯片制成工艺不同的影响较大程度地降低。

内部 32kHz 振荡器 – LIRC

内部 32kHz 系统振荡器是一个低频振荡器。它是一个完全集成 RC 振荡器，它在 5V 电压下运行的典型频率值为 32kHz 且无需外部元件。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响较大程度地降低。

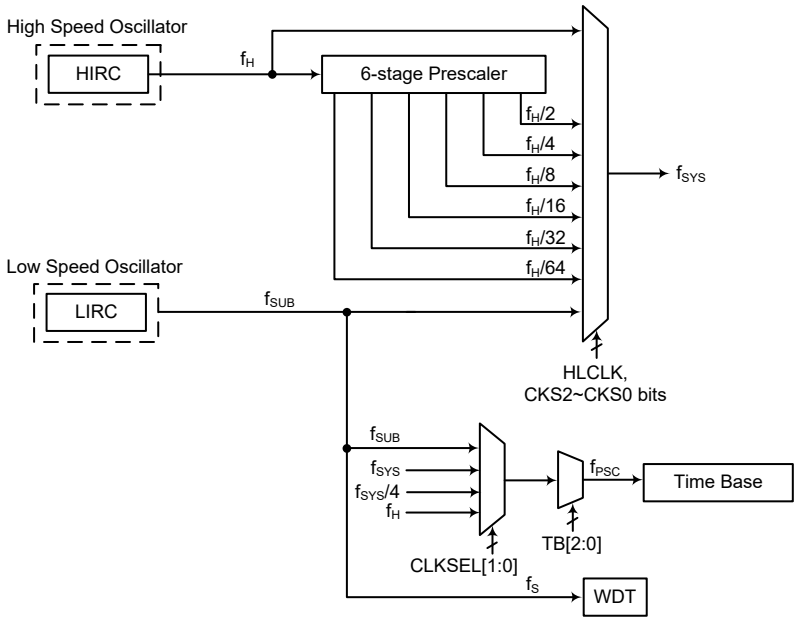
工作模式和系统时钟

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。Holtek 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

单片机为 CPU 和外围功能操作提供了多种不同的时钟源。用户使用寄存器编程可获取多种时钟，进而使系统时钟获取较大的应用性能。

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。高频时钟来自 HIRC 振荡器，低频系统时钟源来自内部时钟 f_{SUB}，若 f_{SUB} 被选择，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 $f_H/2$~$f_H/64$。



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。

单片机时钟配置

系统工作模式

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：快速模式和低速模式。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
快速模式	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

快速模式

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。该低速时钟源可来自 f_{SUB}。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，由于看门狗定时器功能始终使能，f_{SUB} 时钟将继续运行。

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，因此无法驱动 CPU。

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 WDT 和 TM。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。

控制寄存器

寄存器 SMOD 和 CTRL 用于控制单片机系统时钟和相关振荡器配置。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SMOD	CKS2	CKS1	CKS0	—	LTO	HTO	IDLEN	HLCLK
CTRL	FSYSON	—	HIRCS1	HIRCS0	—	LVRF	LRF	WRF

系统工作模式控制寄存器列表

- **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 未定义，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或经唤醒后何时稳定下来。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值总是为“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，若使用 HIRC 振荡器，则在上电复位或唤醒动作发生后的 15~16 个时钟周期后此位将转为高电平。

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。

- **CTRL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	HIRCS1	HIRCS0	—	LVRF	LRF	WRF
R/W	R/W	—	R/W	R/W	—	R/W	R/W	R/W
POR	0	—	0	0	—	x	0	0

“x”：未知

Bit 7 **FSYSON**：IDLE 模式下 f_{SYS} 控制位
0：除能
1：使能
该位用于控制系统时钟在空闲模式中是否开启，如果该位置为“0”，空闲模式中系统时钟关闭，如果该位置为“1”，空闲模式中系统时钟开启。

Bit 6 未定义，读为“0”

Bit 5~4 **HIRCS1~HIRCS0**：HIRC 时钟频率选择位
00：8MHz
01：16MHz
10：12MHz
11：8MHz

Bit 3 未定义，读为“0”

Bit 2 **LVRF**：LVR 复位标志位
详见其它章节。

Bit 1 **LRF**：LVR 控制寄存器软件复位标志位
详见其它章节。

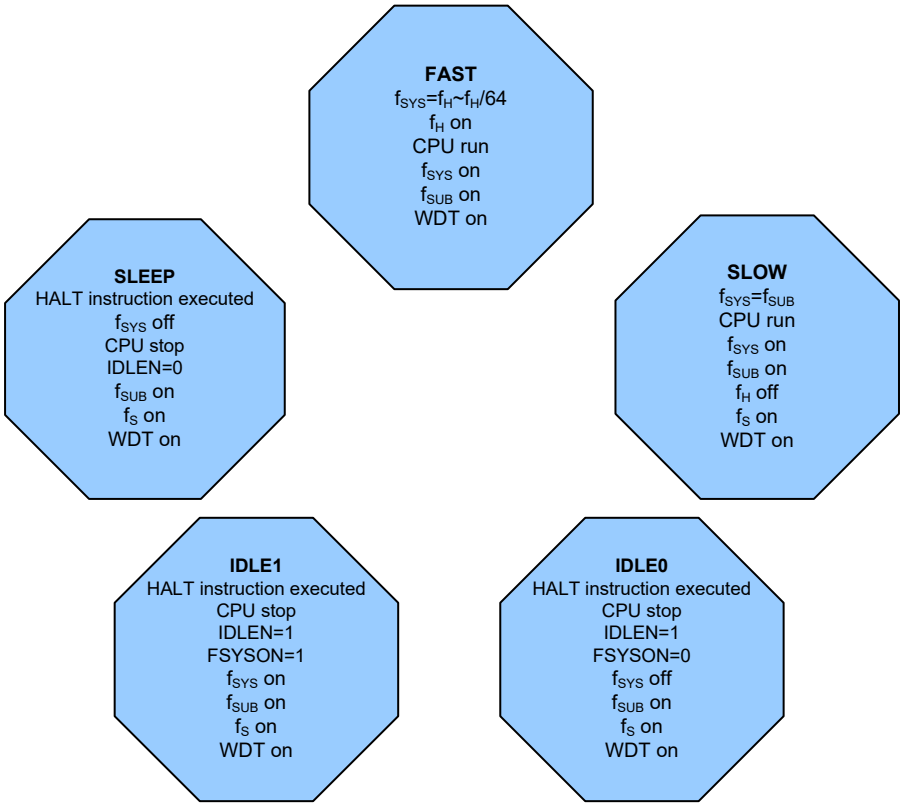
Bit 0 **WRF**：WDTC 控制寄存器软件复位标志位
详见其它章节。

工作模式转换

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，快速模式和低速模式间的切换仅需设置 SMOD 中的 HLCLK 位及 CKS2~CKS0 位即可实现，而快速模式 / 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。

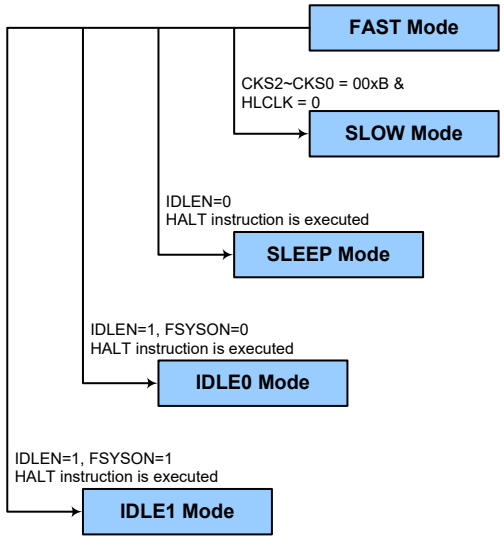
当 HLCLK 位变为低电平时，时钟源将由高速时钟源 f_H 转换成时钟源 $f_H/2$~$f_H/64$ 或 f_{SUB}。若时钟源来自 f_{SUB}，高速时钟源将停止运行以节省耗电。此时须注意，$f_H/16$ 和 $f_H/64$ 内部时钟源也将停止运行。所附流程图显示了单片机在不同工作模式间切换时的变化。



快速模式切换到低速模式

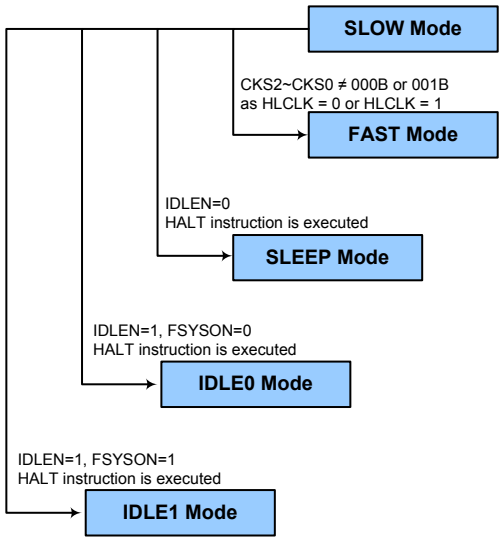
系统运行在快速模式时使用高速系统振荡器，因此较为耗电。可通过设置SMOD寄存器中的HLCLK位为“0”及CKS2~CKS0位为“000”或“001”使系统时钟切换至运行在低速模式下。此时将使用低速系统振荡器以节省耗电。用户可在对性能要求不高的操作中使用此方法以减少耗电。

低速模式的时钟源来自LIRC振荡器，因此要求这些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。



低速模式切换到快速模式

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快速模式需设置 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”。它需要一定的时间来使高频时钟稳定，可通过检测 HTO 标志位进行判断。



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:

- 系统时钟将停止运行，应用程序停止在“HALT”指令处，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。
- 由于 WDT 功能始终使能，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。
- 由于 WDT 功能始终使能，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：

- 系统时钟和低频 f_{SUB} 时钟将开启，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。
- 由于 WDT 功能始终使能，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。

待机电流注意事项

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。

另外应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 中系统振荡器开启，若外设功能时钟源来自高速系统振荡器，额外的待机电流也可能会有几百微安。

唤醒

单片机进入休眠模式或空闲模式后，系统时钟将停止以降低功耗。然而单片机再次唤醒，原来的系统时钟重新起振、稳定且恢复正常工作需要一定的时间。

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

当单片机执行 HALT 指令，PDF 将被置位。系统上电或执行清除看门狗的指令，会清零 PDF；若由 WDT 溢出唤醒，则会发生看门狗定时器复位。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。

编程注意事项

高速和低速振荡器使用相同的 SST 计数器。例如，若系统从休眠模式中唤醒，HIRC 振荡器需从关闭状态启动。

若单片机从休眠模式唤醒后进入快速模式，高速系统振荡器需要一个 SST 周期。在 HTO 位为“1”后，单片机开始执行首条指令。

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 f_{SUB} 由内部低速振荡器 LIRC 提供。电压为 5V 时内部振荡器 LIRC 的频率大约为 32kHz，这个特殊的内部时钟周期会随 V_{DD}、温度和制成的不同而变化。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。

看门狗定时器控制寄存器

WDTC 寄存器用于选择溢出周期、控制 WDT 功能的使能和 MCU 复位。除了 WDT 溢出复位硬件热复位外的其它复位发生后 WDTC 的值为 01010011B。

- **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 功能软件控制位
01010/10101：使能
其它：MCU 复位
若因外部环境噪声或软件设置使单片机复位，复位动作发生在一段延迟时间 t_{SRESET} 后，复位后 CTRL 寄存器中的 WRF 标志位会被置位。

Bit 2~0 **WS2~WS0**：WDT 溢出周期选择位
000：$2^8/f_{SUB}$
001：$2^{10}/f_{SUB}$
010：$2^{12}/f_{SUB}$
011：$2^{14}/f_{SUB}$
100：$2^{15}/f_{SUB}$
101：$2^{16}/f_{SUB}$
110：$2^{17}/f_{SUB}$
111：$2^{18}/f_{SUB}$
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	—	HIRCS1	HIRCS0	—	LVRF	LRF	WRF
R/W	R/W	—	R/W	R/W	—	R/W	R/W	R/W
POR	0	—	0	0	—	x	0	0

“x”：未知

Bit 7 **FSYSON**：IDLE 模式下 f_{SYS} 控制位
详见其它章节。

Bit 6 未定义，读为“0”

Bit 5~4 **HIRCS1~HIRCS0**：HIRC 频率时钟选择位
详见其它章节。

Bit 3 未定义，读为“0”

Bit 2 **LVRF**：LVR 复位标志位
详见其它章节。

Bit 1 **LRF**：LVR 控制寄存器软件复位标志位
详见其它章节。

Bit 0 **WRF**：WDT 控制寄存器软件复位标志位
0：未发生
1：发生
WDT 控制寄存器软件复位时，该位被置为“1”，且通过应用程序清零。注意，该位只能由应用程序清零。

看门狗定时器操作

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 位可提供使能控制以及看门狗定时器复位操作。如果 WE4~WE0 设置为“10101B”或“01010B”，则 WDT 使能；如果 WE4~WE0 设置为除“01010B”和“10101B”以外的其它任意值，则经过一段延迟时间 t_{SRESET} 后单片机复位。上电后这些位初始化为“01010B”。

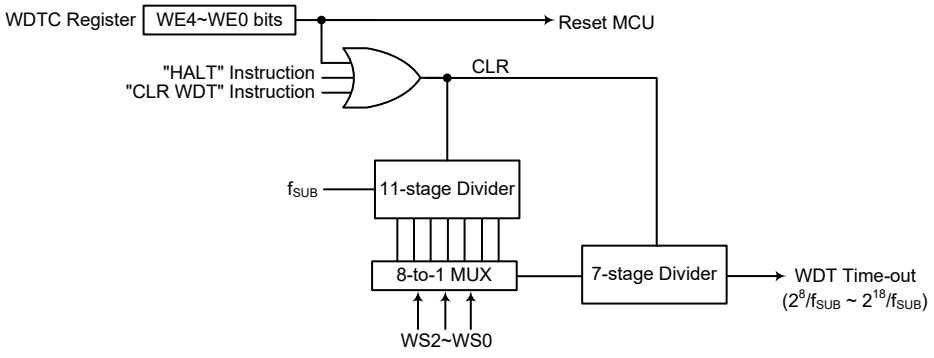
WE4~WE0 位	WDT 功能
01010B/10101B	使能
其它值	单片机复位

看门狗定时器使能 / 复位控制

程序正常运行时，WDT 溢出将导致单片机复位，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 PC 和堆栈指针复位。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 WDT 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软件指令清看门狗。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 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。

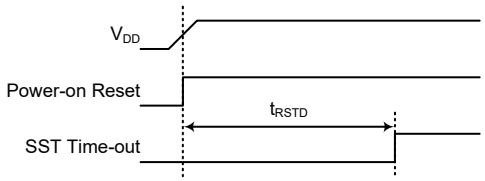
另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。另一种复位为看门狗溢出单片机复位，不同方式的复位操作会对寄存器产生不同的影响。

复位功能

单片机的几种内部事件发生的复位方式：

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。

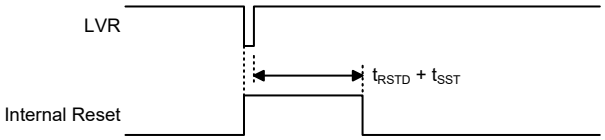


注：t_{RSTD} 为上电延迟时间，典型值为 48ms

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。低电压复位功能总是使能，并会设置一个电源复位低电压，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} 之间，这时 LVR 将会自动复位单片机，并且寄存器 CTRL 中的 LVRF 位将被自动置位为 1。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 LVR 电气特性中的 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。实际的 V_{LVR} 参数值可通过 LVRC 寄存器中的 LVS7~LVS0 位进行选择。若 LVS7~LCS0 位段的值由于不利的环境因素如噪声而发生改变，单片机将在一段延迟时间 t_{SRESET} 后复位，此时 CTRL 寄存器中的 LRF 位将被置为 1。上电后该寄存器的默认值为 01010101B。注意当单片机进入空闲或休眠模式，LVR 功能将自动关闭。



注：t_{RSTD} 为上电延迟时间，典型值为 48ms

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
01010101：2.1V
00110011：2.55V
10011001：3.15V
10101010：3.8V
其它值：单片机复位 – 寄存器复位为 POR 值
若低电压情况发生且满足以上定义的低电压复位值，则单片机复位。当低电压状况保持时间大于 t_{LVR} 后，响应复位。
除了以上定义的四个低电压复位值外，其它值也能导致单片机复位。需要经过一段延迟时间 t_{SRESET} 才响应复位。但此时寄存器内容将复位为 POR 值。

- **CTRL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	HIRCS1	HIRCS0	—	LVRF	LRF	WRF
R/W	R/W	—	R/W	R/W	—	R/W	R/W	R/W
POR	0	—	0	0	—	x	0	0

“x”：未知

Bit 7 **FSYSON**：IDLE 模式下 f_{SYS} 控制位
详见其它章节。

Bit 6 未定义，读为“0”

Bit 5~4 **HIRCS1~HIRCS0**：HIRC 频率时钟选择位
详见其它章节。

Bit 3 未定义，读为“0”

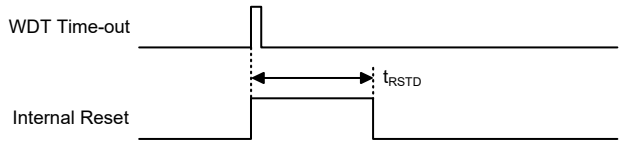
Bit 2 **LVRF**：LVR 复位标志位
0：未发生
1：发生
当特定的低电压复位条件发生时，该位被置为“1”。该位只能由应用程序清零。

Bit 1 **LRF**：LVR 控制寄存器软件复为标志
0：未发生
1：发生
如果 LVRC 寄存器包含任何非定义的 LVR 电压值，此位被置为“1”，这类似于软件复位功能，且只能通过应用程序清零。

Bit 0 **WRF**：WDT 控制寄存器软件复位标志位
详见其它章节。

正常运行时看门狗溢出复位

在快速或低速模式下正常运行时，看门狗溢出复位后 TO 将被设为“1”。

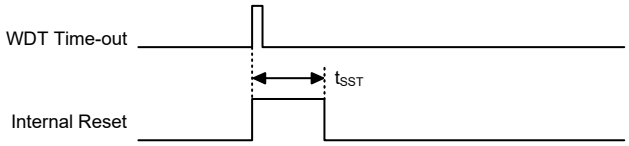


注：t_{RSTD} 为上电延迟时间，典型值为 16ms

正常运行时看门狗溢出复位时序图

休眠或空闲时看门狗溢出复位

休眠或空闲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零及 TO 与 PDF 位被设为 1 外，绝大部份的条件保持不变。图中 t_{SST} 的详细说明请参考系统上电时间电气特性。



休眠或空闲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，由休眠或空闲模式功能或看门狗计数器等几种控制器操作控制。复位标志位如下所示：

TO	PDF	复位条件
0	0	上电复位
u	u	快速或低速模式时的 LVR 复位
1	u	快速或低速模式时的 WDT 溢出复位
1	1	空闲或休眠模式时的 WDT 溢出复位

“u”：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，时基	都清除，且 WDT 重新计数
定时器模块	定时器模块停止
输入 / 输出口	I/O 口设为输入模式
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。

寄存器	BS83B08C	BS83B12C	BS83B16C	上电复位	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
IAR0	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP0	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
IAR1	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	●			---- ---0	---- ---0	---- ---0	---- ---u
		●	●	---- --00	---- --00	---- --00	---- --uu
ACC	●	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	●	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	●	●	●	xxxx xxxx	uuuu uuu	uuuu uuu	uuuu uuuu
TBHP	●	●	●	---- -xxx	---- -uuu	---- -uuu	---- -uuu
STATUS	●	●	●	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
SMOD	●	●	●	000- 0011	000- 0011	000- 0011	uuu- uuuu
CTRL	●	●	●	0-00 -x00	0-00 -100	0-00 -x00	u-uu --uu
LVRC	●	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
INTEG	●	●	●	---- --00	---- --00	---- --00	---- --uu
INTC0	●	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	●	●	●	-000 -000	-000 -000	-000 -000	-uuu -uuu
MFI	●	●	●	--00 --00	--00 --00	--00 --00	--uu --uu
PA	●	●	●	1--1 1111	1--1 1111	1--1 1111	u--u uuuu
PAC	●	●	●	1--1 1111	1--1 1111	1--1 1111	u--u uuuu
PAPU	●	●	●	0--0 0000	0--0 0000	0--0 0000	u--u uuuu
PAWU	●	●	●	0--0 0000	0--0 0000	0--0 0000	u--u uuuu
PXRM	●	●	●	00-- ---00	---- ---00	---- ---00	---- ---uu
WDTC	●	●	●	0101 0011	0101 0011	0101 0011	uuuu uuuu
TBC	●	●	●	---- 0000	---- 0000	---- 0000	---- uuuu
PSCR	●	●	●	---- –00	---- –00	---- –00	---- --uu
EEA	●	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
EED	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	●	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	●	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMTOC	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMC0	●	●	●	111- 0000	111- 0000	111- 0000	uuu- uuuu
SIMC1	●	●	●	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/SIMC2	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu

寄存器	BS83B08C	BS83B12C	BS83B16C	上电复位	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
PC		●		---- 1111	---- 1111	---- 1111	---- uuuu
PCC		●		---- 1111	---- 1111	---- 1111	---- uuuu
PCPU		●		---- 0000	---- 0000	---- 0000	---- uuuu
PC			●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC			●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU			●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMC0	●	●	●	0000 0---	0000 0---	0000 0---	uuuu u---
PTMC1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMDL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMDH	●	●	●	---- --00	---- --00	---- --00	---- --uu
PTMAL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMAH	●	●	●	---- --00	---- --00	---- --00	---- --uu
PTMRPL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMRPH	●	●	●	---- --00	---- --00	---- --00	---- --uu
TKTMR	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC0	●	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
TK16DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK16DH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC1	●	●	●	---- --11	---- --11	---- --11	---- --uu
TKM016DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM016DH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0ROL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0ROH	●	●	●	---- --00	---- --00	---- --00	---- --uu
TKM0C0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0C1	●	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM116DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM116DH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM1ROL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM1ROH	●	●	●	---- --00	---- --00	---- --00	---- --uu
TKM1C0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM1C1	●	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM216DL		●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM216DH		●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM2ROL		●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM2ROH		●	●	---- --00	---- --00	---- --00	---- --uu
TKM2C0		●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM2C1		●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu

寄存器	BS83B08C	BS83B12C	BS83B16C	上电复位	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
TKM316DL			●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM316DH			●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM3ROL			●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM3ROH			●	---- --00	---- --00	---- --00	---- --uu
TKM3C0			●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM3C1			●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
EEC	●	●	●	---- 0000	---- 0000	---- 0000	---- uuuu

注：“-”表示未定义
“x”表示未知
“u”表示不改变

输入 / 输出端口

Holtek 单片机的输入 / 输出口控制具有很大的灵活性。大部分引脚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，这些特性也使得此系列单片机在广泛应用上都能符合开发的需求。

该系列单片机提供 PA~PC 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。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PA	PA7	—	—	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	—	—	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	—	—	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	—	—	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

“—”：未定义，读为“0”

输入 / 输出逻辑功能寄存器列表 – **BS83B08C**

寄存器名称	位							
	7	6	5	4	3	2	1	0
PA	PA7	—	—	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	—	—	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	—	—	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	—	—	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	—	—	—	—	PC3	PC2	PC1	PC0
PCC	—	—	—	—	PCC3	PCC2	PCC1	PCC0
PCPU	—	—	—	—	PCPU3	PCPU2	PCPU1	PCPU0

“—”：未定义，读为“0”

输入 / 输出逻辑功能寄存器列表 – **BS83B12C**

寄存器名称	位							
	7	6	5	4	3	2	1	0
PA	PA7	—	—	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	—	—	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	—	—	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	—	—	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

“—”：未定义，读为“0”

输入 / 输出逻辑功能寄存器列表 – **BS83B16C**

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外部电阻来实现上拉的功能。为了免去外部电阻，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相关上拉电阻控制寄存器 PAPU~PCPU 来设置，它用一个 PMOS 晶体管来实现上拉电阻功能。

- **PxPU 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PxPU7	PxPU6	PxPU5	PxPU4	PxPU3	PxPU2	PxPU1	PxPU0
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

PxPUn：Px 口引脚上拉功能控制
0：除能
1：使能

PxPUn 位是输入 / 输出引脚上拉功能控制位。“x”可以是 A、B 或 C。每个实际输入 / 输出端口的位可以是不同的。

PA 口唤醒

当使用暂停指令“HALT”迫使单片机进入休眠或空闲模式，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 口的每个引脚可以通过设置 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	—	—	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7 **PAWU**：PA7 唤醒功能控制
0：除能
1：使能

Bit 6~5 未定义，读为“0”

Bit 4~0 **PAWU4~PAWU0**：PA4~PA0 唤醒功能控制
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，即 PAC~PCC，用来控制输入 / 输出状态。从而每个 I/O 引脚都可以通过软件控制，动态的设置为 CMOS 输出或输入。所有的 I/O 端口的引脚都各自对应于 I/O 端口控制的某一位。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。

- **PxC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PxC7	PxC5	PxC5	PxC4	PxC3	PxC2	PxC1	PxC0
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

PxCn：Px 口 I/O 引脚类型选择位
0：输出
1：输入
PxCn 位是引脚类型选择位“x”可以是 A、B 或 C。每个实际输入 / 输出端口的位可以是不同的。

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 PXRM 进行设定。

如果共用引脚功能同时有多个输出，“/”标记的右侧引脚名称拥有更高优先级。

- **PXRM 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TMPC1	TMPC0	—	—	—	—	PXRM1	PXRM0
R/W	R/W	R/W	—	—	—	—	R/W	R/W
POR	0	0	—	—	—	—	0	0

Bit 7 **TMPC1**：PTPB 引脚控制
详见其它章节。

Bit 6 **TMPC0**：PTP 引脚控制
详见其它章节。

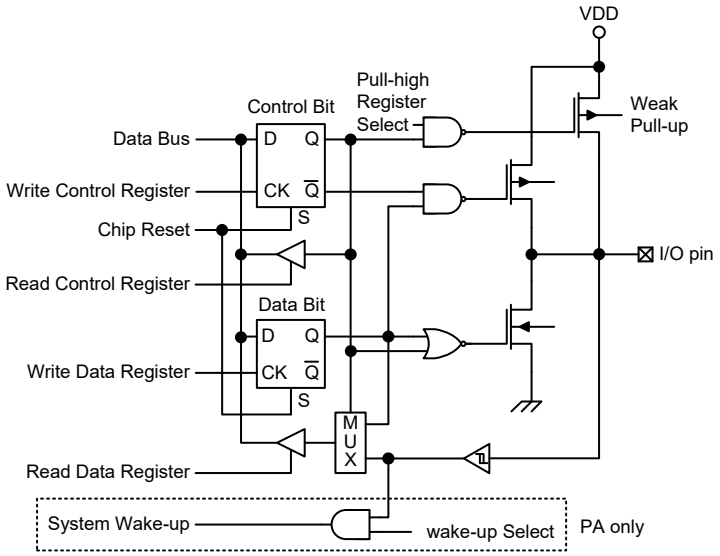
Bit 5~2 未定义，读为“0”

Bit 1 **PXRM1**：SIM 模块的 SCK/SCL 引脚重置选择位
0：PA2
1：PA7

Bit 0 **PXRM0**：SIM 模块的 SDI/SDA 引脚重置选择位
0：PA0
1：PA4

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 I/O 引脚功能的理解提供的一个参考。图中的引脚共用结构并非针对所有单片机。



逻辑功能输入 / 输出结构

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 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。

定时器模块 – TM

控制和测量时间在任何单片机中都是一个很重要的部分。该系列单片机提供单个定时器模块 (简称 TM)，来实现和时间有关的功能。定时器模块是包括多种操作的定时单元，提供的操作有：定时 / 计数器、捕捉输入、比较匹配输出、单脉冲输出以及 PWM 输出等功能。每个定时器模块有两个独立中断。每个 TM 外加的输入输出引脚，扩大了定时器的灵活性，便于用户使用。

简介

该系列单片机包含 1 个周期型 TM，命名为 PTM。PTM 的主要特性概要见下表。

TM 功能	PTM
定时 / 计数器	√
捕捉输入	√
比较匹配输出	√
PWM 输出	√
单脉冲输出	√
PWM 对齐方式	边沿对齐
PWM 调节周期 & 占空比	占空比或周期

TM 功能概要

TM 操作

PTM 提供从简单的定时操作到 PWM 信号产生等多种功能。理解 TM 操作的关键是比较 TM 内独立运行的计数器的值与内部比较器的预置值。当计数器的值与比较器的预置值相同时，则比较匹配，TM 中断信号产生，清零计数器并改变 TM 输出引脚的状态。用户选择内部时钟或外部时钟来驱动内部 TM 计数器。

TM 时钟源

驱动 TM 计数器的时钟源很多。通过设置 PTM 控制寄存器的 PTCK2~PTCK0 位，选择所需的时钟源。该时钟源来自系统时钟 f_{SYS} 或内部高速时钟 f_H 或 f_{SUB} 时钟源或外部 PTCK 引脚时钟的分频比。PTCK 引脚时钟源用于允许外部信号作为 TM 时钟源或用于事件计数。

TM 中断

周期型 TM 有两个内部中断，分别是内部比较器 A 或比较器 P，当比较匹配发生时产生 TM 中断。当 TM 中断产生时，计数器清零并改变 TM 输出引脚的状态。

TM 外部引脚

周期型 TM 有两个 TM 输入引脚 PTCK 和 PTPI。通过设置 PTMC0 寄存器中的 PTCK2~PTCK0 位进行选择，外部时钟源可通过该引脚来驱动内部 TM。TM 输入引脚可以选择上升沿或下降沿。PTCK 引脚还可用作单脉冲输出模式的外部触发输入引脚。

另一种 PTM 输入引脚 PTPI 作为捕捉输入脚，其有效边沿有上升沿、下降沿和双沿，通过设置 PTMC1 寄存器中的 PTIO1~PTIO0 位来选择有效边沿类型。

周期型 TM 有两个输出引脚，PTP 和 PTPB。PTPB 引脚输出为 PTP 的反相信号。当 TM 工作在比较匹配输出模式且比较匹配发生时，这些引脚会由 TM 控制切换到高电平或低电平或翻转。外部 PTP 和 PTPB 输出引脚也被 TM 用来产生 PWM 输出波形。当 TM 输出引脚与其它功能共用时，TM 输出功能需要通

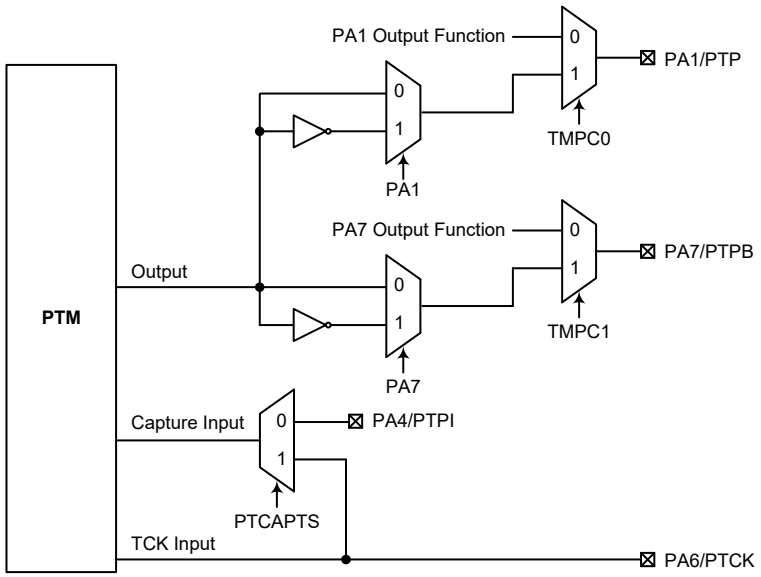
过相关寄存器先被设置。寄存器中的一个单独位用于决定其相关引脚用于外部 TM 输出还是用于其它功能。

PTM	
输入	输出
PTCK, PTPI	PTP, PTPB

TM 外部引脚

TM 输入 / 输出端口控制寄存器

选择作为 TM 输入 / 输出引脚还是其它共用引脚功能是通过设置相关 TM 引脚功能控制寄存器来实现的。每个 TM 输入 / 输出引脚都有对应的引脚控制位。设置选择位为高时，相关引脚用作 TM 输入 / 输出；若复位为低，引脚将恢复原本的功能。



PTM 功能引脚控制方框图

- **PXRM 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TMPC1	TMPC0	—	—	—	—	PXRM1	PXRM0
R/W	R/W	R/W	—	—	—	—	R/W	R/W
POR	0	0	—	—	—	—	0	0

Bit 7 **TMPC1**：PTPB 引脚控制
0：除能
1：使能

Bit 6 **TMPC0**：PTP 引脚控制
0：除能
1：使能

Bit 5~2 未定义，读为“0”

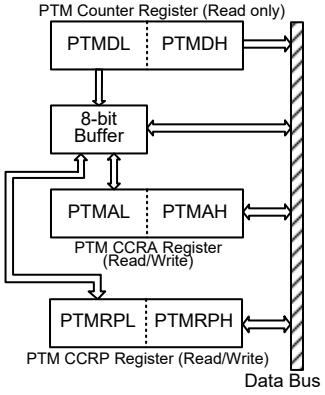
Bit 1 **PXRM1**：SIM 模块的 SCK/SCL 引脚重置选择位
详见其它章节。

Bit 0 **PXRM0**：SIM 模块的 SDI/SDA 引脚重置选择位
详见其它章节。

编程注意事项

TM 计数寄存器和捕捉 / 比较寄存器 CCRA、CCRP 为 10-bit 的寄存器，含有低字节和高字节结构。高字节可直接访问，低字节则仅能通过一个内部 8-bit 的缓存器进行访问。读写这些成对的寄存器需通过特殊的方式。值得注意的是 8-bit 缓存器的存取数据及相关低字节的读写操作仅在其相应的高字节读取操作执行时发生。

正如 CCRA 和 CCRP 寄存器按照下图方式执行且具体存取这些寄存器对的方式如上所述，建议使用“MOV”指令，通过以下步骤访问 CCRA 和 CCRP 低字节，命名为 PTMAL 和 PTMRPL。若不采用以下步骤访问 CCRA 和 CCRP 将导致不可预期的结果。

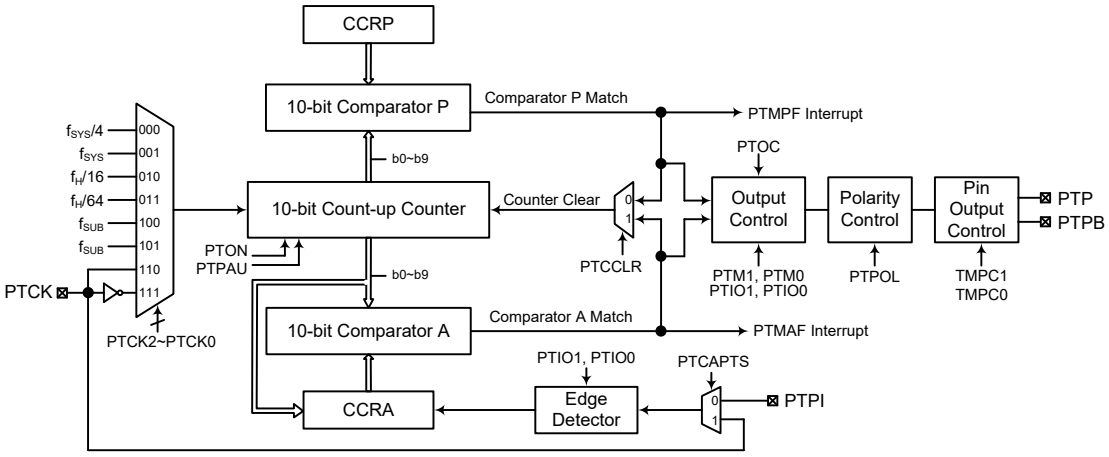


读写流程如下步骤所示：

- 写数据至 CCRA 或 CCRP
  - ♦ 步骤 1. 写数据至低字节寄存器 PTMAL 或 PTMRPL
    – 注意，此时数据仅写入 8-bit 缓存器。
  - ♦ 步骤 2. 写数据至高字节寄存器 PTMAH 或 PTMRPH
    – 注意，此时数据直接写入高字节寄存器，同时锁存在 8-bit 缓存器中的数据写入低字节寄存器。
- 由计数器寄存器和 CCRA 或 CCRP 中读取数据
  - ♦ 步骤 1. 由高字节寄存器 PTMDH、PTMAH 或 PTMRPH 读取数据
    – 注意，此时高字节寄存器中的数据直接读取，同时由低字节寄存器读取的数据锁存至 8-bit 缓存器中。
  - ♦ 步骤 2. 由低字节寄存器 PTMDL、PTMAL 或 PTMRPL 读取数据
    – 注意，此时读取 8-bit 缓存器中的数据。

周期型 TM – PTM

周期型 TM 包括 5 种工作模式，即比较匹配输出、定时 / 事件计数器、捕捉输入、单脉冲输出和 PWM 输出模式。周期型 TM 由两个外部输入脚控制并驱动两个外部输出脚。



注：PTPB 为 PTP 的反相输出。

周期型 TM 方框图

周期型 TM 操作

周期型 TM 核心是一个由用户选择的内部或外部时钟源驱动的 10 位向上计数器，它还包括两个内部比较器即比较器 A 和比较器 P。这两个比较器将计数器的值与 CCRA 和 CCRP 寄存器中的值进行比较。CCRP 比较器是 10 位宽度。

通过应用程序改变 10 位计数器值的唯一方法是使 PTON 位发生上升沿跳变清除计数器。此外，计数器溢出或比较匹配也会自动清除计数器。上述条件发生时，通常情况会产生 TM 中断信号。周期型 TM 可工作在不同的模式，可由包括来自输入脚的时钟源驱动，也可以控制输出脚。所有工作模式的设定都是通过设置相关寄存器来实现的。

周期型 TM 寄存器介绍

周期型 TM 的所有操作由一系列寄存器控制。一对只读寄存器用来存放 10 位计数器的值，两对读 / 写寄存器存放 10 位 CCRA 和 CCRP 的值。剩下两个控制寄存器用来设置不同的操作和控制模式。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	—	—	—	—	—	—	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	—	—	—	—	—	—	D9	D8
PTMRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMRPH	—	—	—	—	—	—	D9	D8

10-bit 周期型 TM 寄存器列表

- **PTMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **PTPAU**：PTM 计数器暂停控制位
0：运行
1：暂停
通过设置此位为高可使计数器暂停，清零此位恢复正常计数器操作。当处于暂停条件时，PTM 保持上电状态并继续耗电。当此位由低到高转变时，计数器将保留其剩余值，直到此位再次改变为低电平，并从此值开始继续计数。

Bit 6~4 **PTCK2~PTCK0**：选择 PTM 计数时钟位
000：$f_{SYS}/4$
001：f_{SYS}
010：$f_H/16$
011：$f_H/64$
100：f_{SUB}
101：f_{SUB}
110：PTCK 上升沿
111：PTCK 下降沿
此三位用于选择 PTM 的时钟源。外部引脚时钟源能被选择在上升沿或下降沿有效。f_{SYS} 是系统时钟，f_H 和 f_{SUB} 是其它的内部时钟源，细节方面请参考振荡器章节。

Bit 3 **PTON**：PTM 计数器 On/Off 控制位
0：Off
1：On
此位控制 PTM 的总开关功能。设置此位为高则使能计数器使其运行，清零此位则除能 PTM。清零此位将停止计数器并关闭 PTM 减少耗电。当此位经由低到高转变时，内部计数器将复位清零；当此位经由高到低转换时，内部计数器将保持其剩余值，直到此位再次改变为高电平。
若 PTM 处于比较匹配输出模式或 PWM 输出模式或单脉冲输出模式时，当 PTON 位经由低到高的转变时，PTM 输出脚将复位至 PTOC 位指定的初始值。

Bit 2~0 未定义，读为“0”

- **PTMC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
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 **PTM1~PTM0**：选择 PTM 工作模式位
00：比较匹配输出模式
01：捕捉输入模式
10：PWM 输出模式或单脉冲输出模式
11：定时 / 计数器模式
这两位设置 PTM 需要的工作模式。为了确保操作可靠，PTM 应在 PTM1 和 PTM0 位有任何改变前先关掉。在定时 / 计数器模式，PTM 输出脚状态为未知。

Bit 5~4 **PTIO1~PTIO0**：选择 PTM 外部引脚功能位
比较匹配输出模式
00：无变化
01：输出低
10：输出高
11：输出翻转

PWM 输出模式 / 单脉冲输出模式
00：强制无效状态
01：强制有效状态
10：PWM 输出
11：单脉冲输出

捕捉输入模式
00：在 PTPI 或 PTCK 上升沿输入捕捉
01：在 PTPI 或 PTCK 下降沿输入捕捉
10：在 PTPI 或 PTCK 双沿输入捕捉
11：输入捕捉除能

定时 / 计数器模式
未使用

此两位用于决定在一定条件达到时 PTM 外部引脚如何改变状态。这两位值的选择取决于 PTM 运行在哪种模式下。

在比较匹配输出模式下，PTIO1 和 PTIO0 位决定当从比较器 A 比较匹配输出发生时 PTM 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 PTM 输出脚能设为切换高、切换低或翻转当前状态。若此两位同时为 0 时，这个输出将不会改变。PTM 输出脚的初始值通过 PTMC1 寄存器的 PTOC 位设置取得。注意，由 PTIO1 和 PTIO0 位得到的输出电平必须与通过 PTOC 位设置的初始值不同，否则当比较匹配发生时，PTM 输出脚将不会发生变化。在 PTM 输出脚改变状态后，通过 PTON 位由低到高电平的转换复位至初始值。

在 PWM 输出模式，PTIO1 和 PTIO0 用于决定比较匹配条件发生时怎样改变 PTM 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。必须在 PTM 关闭时改变 PTIO1 和 PTIO0 位的值。若在 PTM 运行时改变 PTIO1 和 PTIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **PTOC**：PTP 输出控制位

比较匹配输出模式
0：初始低
1：初始高

PWM 输出模式 / 单脉冲输出模式
0：低有效
1：高有效

这是 PTM 输出脚输出控制位。它取决于 PTM 此时正运行于比较匹配输出模式还是 PWM 输出模式 / 单脉冲输出模式。若 PTM 处于定时 / 计数器模式，则其不受影响。在比较匹配输出模式时，比较匹配发生前其决定 TM 输出脚的逻辑电平值。在 PWM 输出模式时，其决定 PWM 信号是高有效还是低有效。在单脉冲输出模式中，当 PTON 位从低到高的跳变时，其决定了 PTM 输出的逻辑电平值。

Bit 2 **PTPOL**：PTM PTP 输出极性控制位
0：同相
1：反相

此位控制 PTP 输出脚的极性。此位为高时 PTM 输出脚反相，为低时 PTM 输出脚同相。若 PTM 处于定时 / 计数器模式时其不受影响。

Bit 1 **PTCAPTS**：选择 PTM 捕捉触发源
0：来自 PTPI 引脚
1：来自 PTCK 引脚

Bit 0 **PTCCLR**：选择 PTM 计数器清零条件位
0：PTM 比较器 P 匹配
1：PTM 比较器 A 匹配

此位用于选择清除计数器的方法。周期型 TM 包括两个比较器 – 比较器 A 和比较器 P，两者都可以用作清除内部计数器。PTCCLR 位设为高，计数器在比较器 A 比较匹配发生时被清除；此位设为低，计数器在比较器 P 比较匹配发生或计数器溢出时被清除。计数器溢出清除的方法仅在 CCRP 被清除为 0 时才能生效。PTCCLR 位在 PWM 输出模式、单脉冲输出或输入捕捉模式时未使用。

- **PTMDL 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 **D7~D0**：PTM 计数器低字节寄存器 bit 7~bit 0
PTM 10-bit 计数器 bit 7~bit 0

- **PTMDH 寄存器**

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

Bit 7~2 未定义，读为“0”
Bit 1~0 **D9~D8**：PTM 计数器高字节寄存器 bit 1~bit 0
PTM 10-bit 计数器 bit 9~bit 8

- **PTMAL 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 **D7~D0**：PTM CCRA 低字节寄存器 bit 7~bit 0
PTM 10-bit CCRA bit 7~bit 0

- **PTMAH 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 未定义，读为“0”
Bit 1~0 **D9~D8**：PTM CCRA 高字节寄存器 bit 1~bit 0
PTM 10-bit CCRA bit 9~bit 8

- **PTMRPL 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 **D7~D0**：PTM CCRP 低字节寄存器 bit 7~bit 0
PTM 10-bit CCRP bit 7~bit 0

- **PTMRPH 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　未定义，读为“0”

Bit 1~0　**D9~D8**：PTM CCRP 高字节寄存器 bit 1~bit 0
PTM 10-bit CCRP bit 9~bit 8

周期型 TM 工作模式

周期型 TM 有五种工作模式，即比较匹配输出模式、PWM 输出模式、单脉冲输出模式、捕捉输入模式或定时 / 计数器模式。通过设置 PTMC1 寄存器的 PTM1 和 PTM0 位选择任意模式。

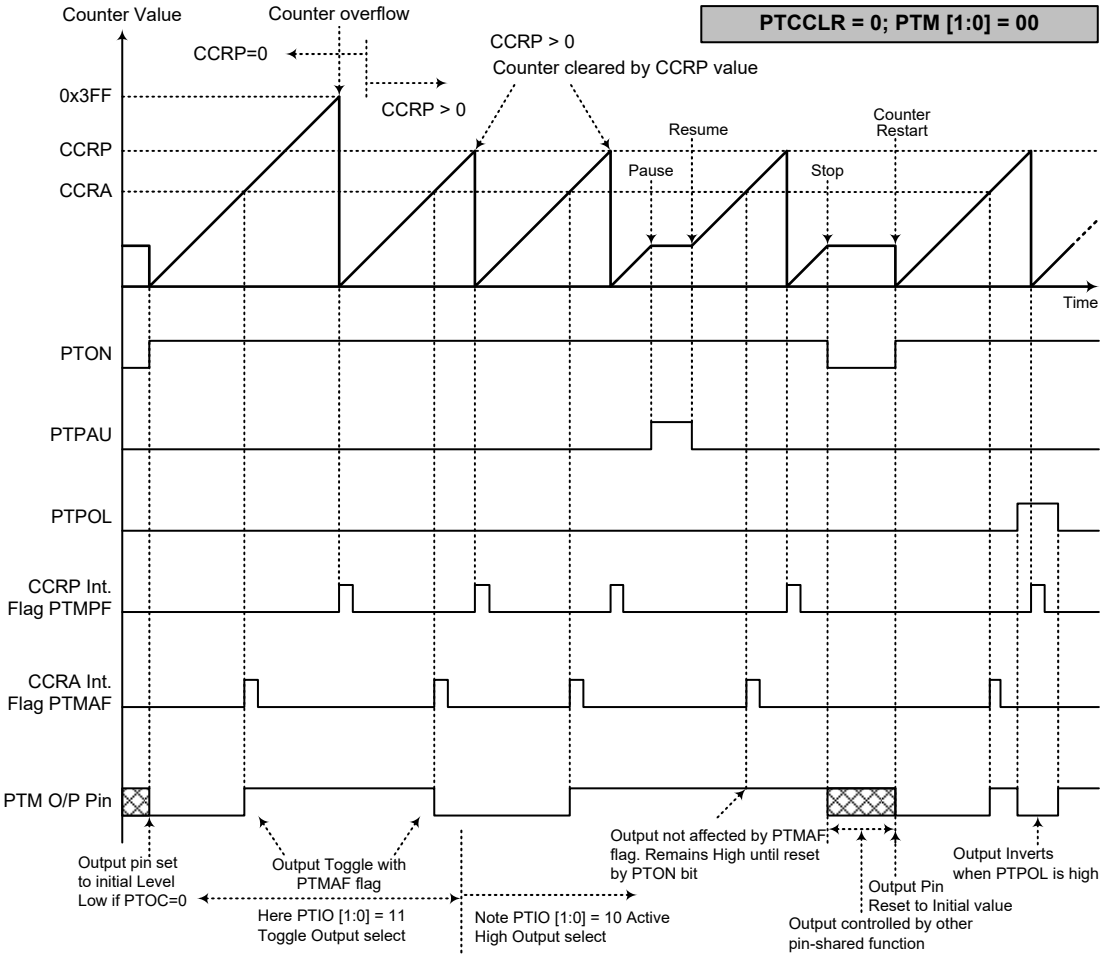
比较匹配输出模式

为使 TM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 位需要设置为“00”。当工作在该模式，一旦计数器使能并开始计数，有三种方法来清零，分别是：计数器溢出，比较器 A 比较匹配发生和比较器 P 比较匹配发生。当 PTCCLR 位为低，有两种方法清除计数器。一种是比较器 P 比较匹配发生，另一种是 CCRP 所有位设置为零并使得计数器溢出。此时，比较器 A 和比较器 P 的请求标志位 PTMAF 和 PTMPF 将分别置起。

如果 PTMC1 寄存器的 PTCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 PTMAF 中断请求标志产生。所以当 PTCCLR 为高时，不会产生 PTMPF 中断请求标志。在比较匹配输出模式中，CCRA 寄存器值不能设为“0”。

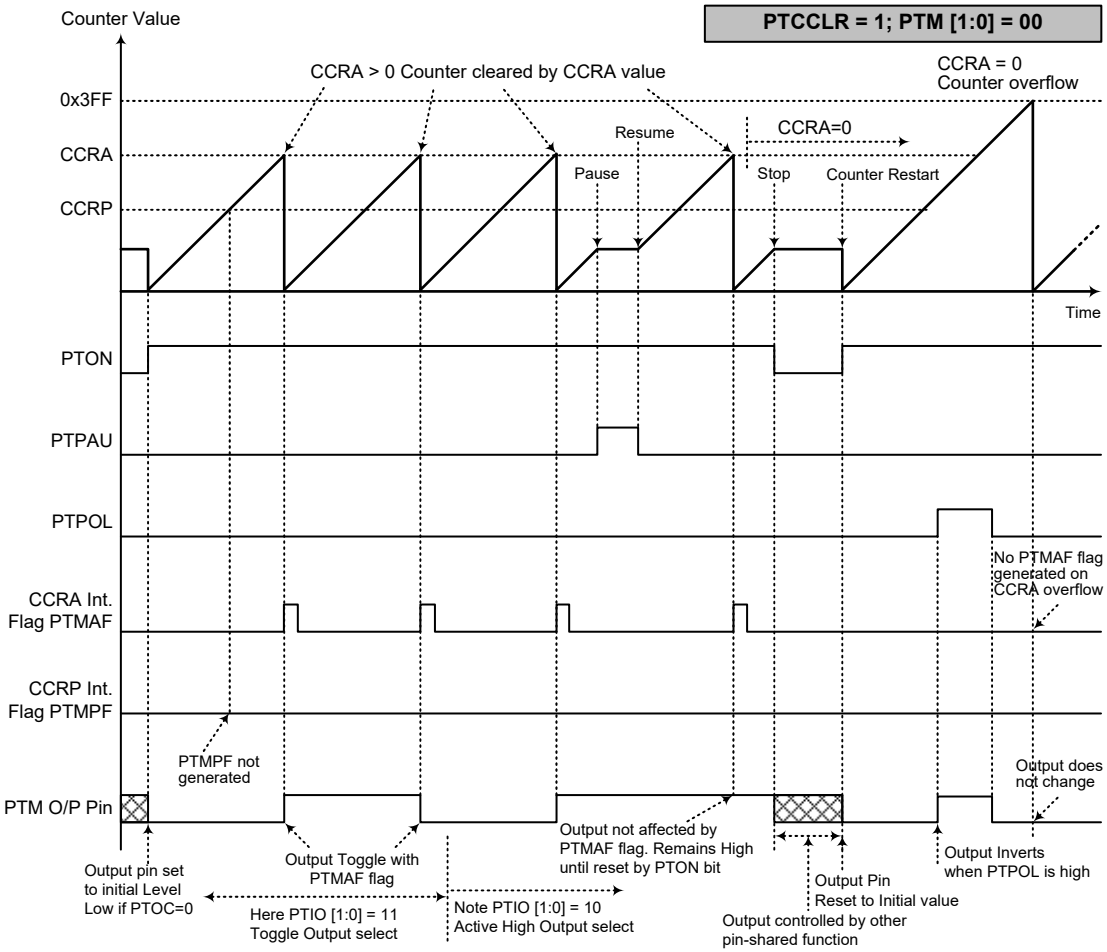
如果 CCRA 为“0”，当 CCRA 达到最大值 0x3FF 时，计数器将溢出，不会产生 PTMAF 中断请求标志。

正如该模式名所言，当比较匹配发生后，TM 输出脚状态改变。当比较器 A 比较匹配发生后 PTMAF 中断请求标志产生时，TM 输出脚状态改变。比较器 P 比较匹配发生时产生的 PTMPF 标志不影响 PTM 输出脚。PTM 输出脚状态改变方式由 PTMC1 寄存器中 PTIO1 和 PTIO0 位决定。当比较器 A 比较匹配发生时，PTIO1 和 PTIO0 位决定 PTM 输出脚输出高，低或翻转当前状态。PTM 输出脚初始值，在 PTON 位由低到高电平的变化后通过 PTOC 位设置。注意，若 PTIO1 和 PTIO0 位同时为 0 时，引脚输出不变。



比较器匹配输出模式 – PTCCLR = 0

注：1. PTCCLR=0，比较器 P 匹配将清除计数器
2. PTM 输出脚仅由 PTMAF 标志位控制
3. 在 PTON 上升沿 TM 输出脚复位至初始值



比较器匹配输出模式 – PTCCLR = 1

注：1. PTCCLR=1，比较器 A 匹配将清除计数器
2. PTM 输出脚仅由 PTMAF 标志位控制
3. 在 PTON 上升沿 TM 输出脚复位至初始值
4. 当 PTCCLR=1 时，不会产生 PTMPF 标志

定时 / 计数器模式

为使 TM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 位需要设置为“11”。定时 / 计数器模式与比较输出模式操作方式相同，并产生同样的中断请求标志。不同的是，在定时 / 计数器模式下 TM 输出脚未使用。因此，比较匹配输出模式中的描述和时序图可以适用于此功能。该模式中未使用的 TM 输出脚用作普通 I/O 脚或其它功能。因此，比较匹配输出模式中的描述和时序图可以适用于此功能。该模式中未使用的 PTM 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 TM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 位需要设置为“10”，且 PTIO1 和 PTIO0 位也需要设置为“10”。TM 的 PWM 功能在马达控制，加热控制，照明控制等方面十分有用。给 TM 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就较为灵活。在 PWM 输出模式中，PTCCLR 位对 PWM 周期无影响。CCRP 和 CCRA 寄存器都用于控制 PWM 方波。CCRP 寄存器通过清除内部计数从而控制 PWM 周期，CCRA 寄存器设置 PWM 的占空比。PWM 波形的周期和占空比由 CCRP 和 CCRA 寄存器的值控制。

当比较器 A 或比较器 P 比较匹配发生时，CCRA 和 CCRP 中断标志位分别产生。PTMC1 寄存器的 PTOC 位选择 PWM 波形的极性，PTIO1 和 PTIO0 位使能 PWM 输出或强制 PTM 输出脚为高电平或低电平。PTPOL 位用于 PWM 输出波形的极性反相控制。

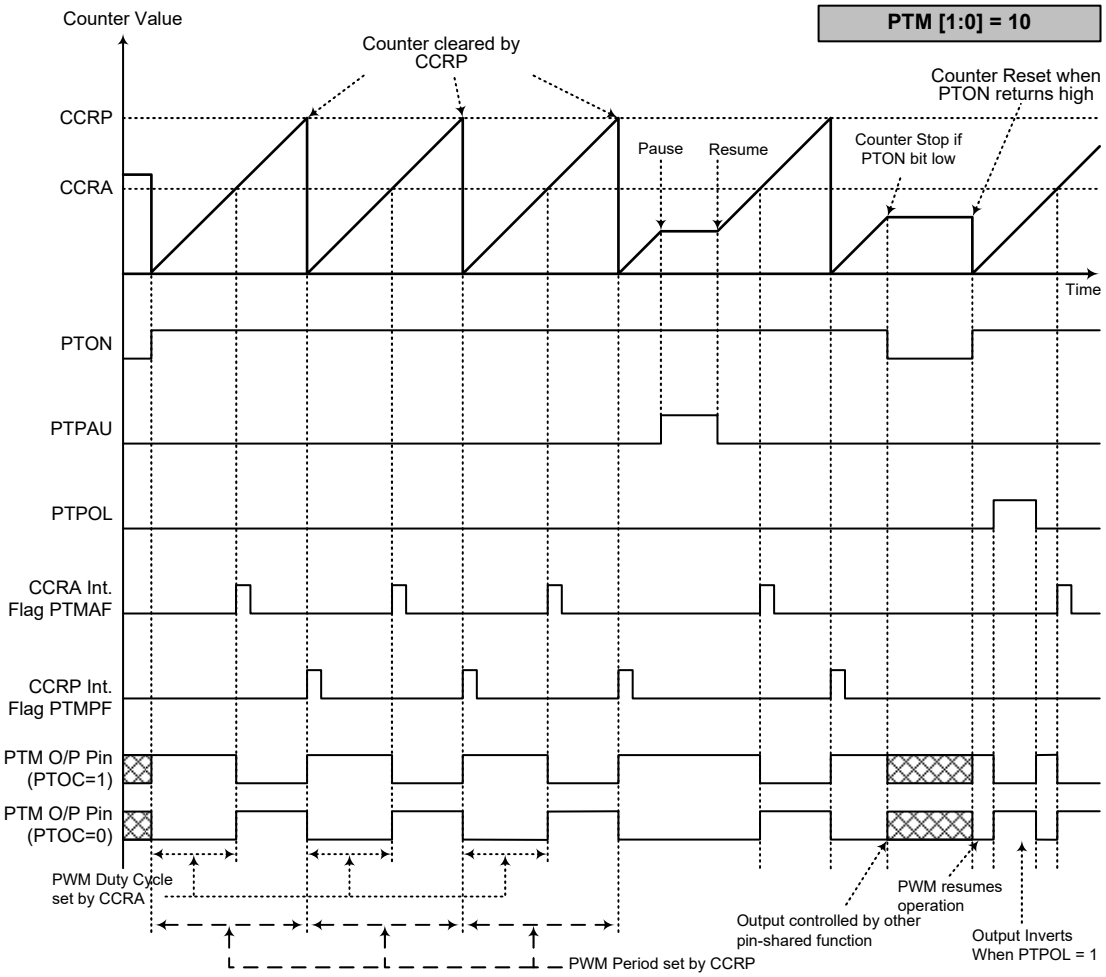
- **10-bit PTM，PWM 输出模式 , 边沿对齐模式**

CCRP	1~1023	0
Period	1~1023	1024
Duty	CCRA	

若 f_{SYS}=16MHz，PTM 时钟源选择 f_{SYS}/4，CCRP=512 且 CCRA=128，

PTM PWM 输出频率 = (f_{SYS}/4)/512=f_{SYS}/2048=7.8125kHz，duty=128/512=25%，

若由 CCRA 寄存器定义的 Duty 值等于或大于 Period 值，PWM 输出占空比为 100%。



PWM 输出模式

注：1. 这里计数器由 CCRP 清除

2. 计数器清除并决定 PWM 周期

3. 当 PTIO[1:0]=00 或 01，PWM 功能不变

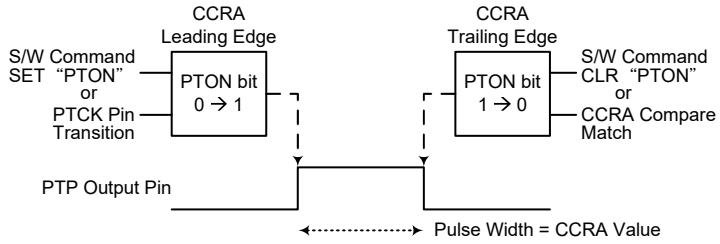
4. PTCCLR 位对 PWM 功能无影响

单脉冲输出模式

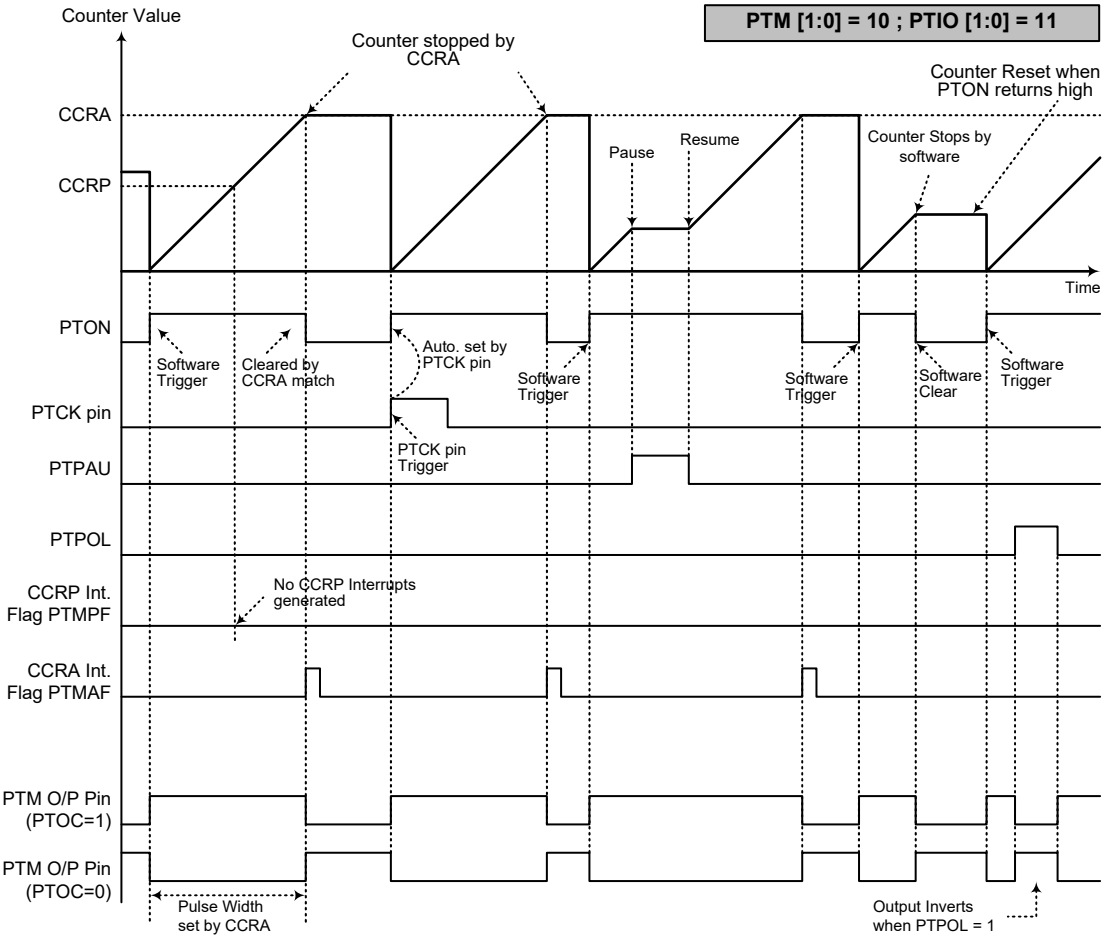
为使 TM 工作在此模式，PTM1 和 PTM0 位需要设置为“10”，并且相应的 PTIO1 和 PTIO0 需要设置为“11”。正如模式名所言，单脉冲输出模式，在 PTM 输出脚将产生一个脉冲输出。

通过应用程序控制 PTON 位由低到高的转变来触发脉冲前沿输出。而处于单脉冲输出模式时，PTON 位可由 PTCK 脚自动由低转变为高，进而依次初始化单脉冲输出。当 PTON 位转变为高电平时，计数器将开始运行，并产生脉冲前沿。通过应用程序使 PTON 位清零或比较器 A 比较匹配发生时，产生脉冲下降沿。

而比较器 A 比较匹配发生时，会自动清零 PTON 位并产生单脉冲输出边沿跳转。CCRA 的值通过这种方式控制脉冲宽度。比较器 A 比较匹配发生时，也会产生 PTM 中断。PTON 位在计数器重启时会发生由低到高的转变，此时计数器才复位至零。在单脉冲输出模式中，CCRP 寄存器和 PTCCLR 位未使用。



单脉冲产生示意图



单脉冲输出模式

注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 PTCK 脚或设置 PTON 位为高来触发脉冲
4. PTCK 脚有效沿会自动置位 PTON
5. 单脉冲输出模式中，PTIO[1:0] 需置位“11”，且不能更改

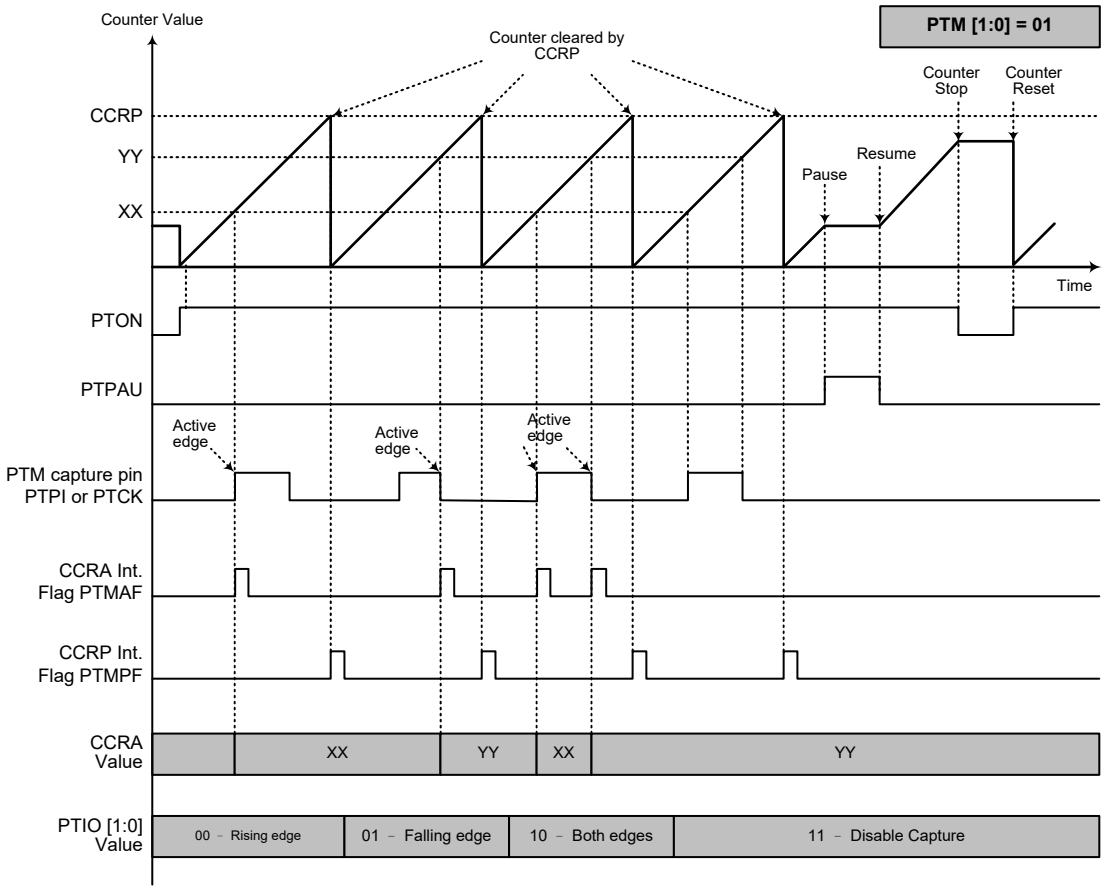
捕捉输入模式

为使 TM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 位需要设置为“01”。此模式使能外部信号捕捉并保存内部计数器当前值，因此被用于诸如脉冲宽度测量的应用中。PTPI 或 PTCK 引脚上的外部信号，通过设置 PTMC1 寄存器的 PTCAPTS 位选择。可通过设置 PTMC1 寄存器的 PTIO1 和 PTIO0 位选择有效边沿类型，即上升沿，下降沿或双沿有效。通过应用程序将 PTON 位由低置为高时，计数器启动。

当 PTPI 或 PTCK 引脚出现有效边沿转换时，计数器当前值被锁存到 CCRA 寄存器，并产生 PTM 中断。不考虑 PTPI 或 PTCK 引脚事件，计数器继续工作直到 PTON 位发生下降沿跳变。当 CCRP 比较匹配发生时计数器复位至零；CCRP 的值通过这种方式控制计数器的最大值。当比较器 P CCRP 比较匹配发生时，也会产生 TM 中断。记录 CCRP 溢出中断信号的值可以测量长脉宽。通过设置 PTIO1 和 PTIO0 位选择 PTPI 或 PTCK 引脚为上升沿，下降沿或双沿有

效。不考虑 PTPI 或 PTCK 引脚事件，如果 PTIO1 和 PTIO0 位都设为高，不会产生捕捉操作，但计数器继续运行。

当 PTPI 或 PTCK 引脚与其它功能共用，PTM 工作在输入捕捉模式时需多加注意。这是因为如果引脚被设为输入，那么该引脚上的任何电平转变都可能执行输入捕捉操作。PTCCLR、PTOC 和 PTPOL 位在此模式中未使用。



捕捉输入模式

注：1. PTM[1:0]=01 并通过 PTIO[1:0] 位设置有效边沿

2. PTM 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中

3. PTCCLR 位未使用

4. 无输出功能，PTOC 和 PTPOL 位未使用

5. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大

触控按键功能

该系列单片机均提供多触控按键功能。触控按键功能完全内部集成不需外接元件，通过内部寄存器的简单操作即可实现此功能。

触控按键结构

触控按键与 PB~PC 逻辑引脚共用，通过寄存器的位来选择所需功能。按键被分成几个组，每组为一个模块，编号为 M0~Mn。每个模块为独立的一组，包含四个触控按键且每个按键有各自的振荡器。每个模块具有单独的控制逻辑电路和配套的寄存器系列。寄存器的名称和它相关的模块编号相对应。

单片机型号	触控按键数量	触控按键模块 (Mn)	触控按键	共用的 I/O 引脚
BS83B08C	8	M0	Key1~Key4	PB0~PB3
		M1	Key5~Key8	PB4~PB7
BS83B12C	12	M0	Key1~Key4	PB0~PB3
		M1	Key5~Key8	PB4~PB7
		M2	Key9~Key12	PC0~PC3
BS83B16C	16	M0	Key1~Key4	PB0~PB3
		M1	Key5~Key8	PB4~PB7
		M2	Key9~Key12	PC0~PC3
		M3	Key13~Key16	PC4~PC7

触控按键结构

触控按键寄存器定义

每个触控按键模块包含四个触控按键功能，且都有其配套的寄存器。以下表格显示了每个触控按键模块的寄存器系列。寄存器名称里的 Mn 对应触控按键模块的序号，BS83B08C 单片机具有模块 M0~M1，BS83B12C 单片机具有模块 M0~M2，BS83B16C 单片机具有模块 M0~M3。

寄存器名称	描述
TKTMR	触控按键时隙 8-bit 计数器预载寄存器
TKC0	触控按键功能控制寄存器 0
TKC1	触控按键功能控制寄存器 1
TK16DL	触控按键功能 16-bit 计数器低字节
TK16DH	触控按键功能 16-bit 计数器高字节
TKMn16DL	触控按键模块 n 16-bit C/F 计数器低字节
TKMn16DH	触控按键模块 n 16-bit C/F 计数器高字节
TKMnROL	触控按键模块 n 参考振荡器电容选择低字节
TKMnROH	触控按键模块 n 参考振荡器电容选择高字节
TKMnC0	触控按键模块 n 控制寄存器 0
TKMnC1	触控按键模块 n 控制寄存器 1

触控按键模块寄存器定义

寄存器名称	位							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	—	TKRCOV	TKST	TKCFOV	TK16OV	TSCS	TK16S1	TK16S0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKC1	—	—	—	—	—	—	TKFS1	TKFS0
TKMn16DL	D7	D6	D5	D4	D3	D2	D1	D0
TKMn16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKMnROL	D7	D6	D5	D4	D3	D2	D1	D0
TKMnROH	—	—	—	—	—	—	D9	D8
TKMnC0	MnMXS1	MnMXS0	MnDFEN	MnFILEN	MnSOFC	MnSOF2	MnSOF1	MnSOF0
TKMnC1	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN

触控按键功能寄存器列表

- **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 **D7~D0**：触控按键时隙 8-bit 计数器预载寄存器

触控按键时隙计数器预载寄存器用于确定触控按键时隙溢出时间。时隙单元周期为 32 个时隙时钟周期，通过一个 5-bit 计数器获得。因此，时隙计数器溢出时间可由下面的等式算出。

时隙计数器溢出时间 = (256-TKTMR[7:0])×32t_{TSC}，此处的 t_{TSC} 为时隙计数器时钟周期。

- **TKC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	TKRCOV	TKST	TKCFOV	TK16OV	TSCS	TK16S1	TK16S0
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

若通过 TSCS 位选择模块 0 或所有模块时隙计数器溢出，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将自动停止。此时触控按键模块 16-bit C/F 计数器、触控按键功能 16-bit 计数器、5-bit 时隙单元周期计数器和 8-bit 时隙计数器都会自动关闭。该标志位无法通过应用程序设置为“1”，但必须通过应用程序清零。

Bit 5 **TKST**：触控按键检测开启控制位

0：停止或无操作

0→1：开始检测

当该位为“0”时，所有模块的 16-bit C/F 计数器、16-bit 计数器和 5-bit 时隙计数器会自动清零。但 8-bit 可编程时隙计数器不清零。当该位由 0 到 1 转变时，触控按键功能的 16-bit C/F 计数器、16-bit 计数器、5-bit 时隙计数器和 8-bit 时隙计数器都会自动开启，并使能按键振荡器和参考振荡器以驱动相应的计数器。

Bit 4 **TKCFOV**：触控按键模块 16-bit C/F 计数器溢出标志位
0：无溢出
1：溢出
当任一触控按键模块 16-bit C/F 计数器溢出时，该位将被置高，但该位无法自动清零，必须通过应用程序清零。

Bit 3 **TK16OV**：触控按键模块 16-bit 计数器溢出标志位
0：无溢出
1：溢出
当任一触控按键模块 16-bit 计数器溢出时，该位将被置高，但该位无法自动清零，必须通过应用程序清零。

Bit 2 **TSCS**：触控按键时隙计数器选择位
0：每个模块使用自己的时隙计数器
1：所有触控按键模块使用模块 0 的时隙计数器

Bit 1~0 **TK16S1~TK16S0**：触控按键模块 16-bit 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**：触控振荡器和参考振荡器频率选择位
00：1MHz
01：3MHz
10：7MHz
11：11MHz

- **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 **D7~D0**：触控按键功能 16-bit 计数器低字节

- **TK16DH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8**：触控按键功能 16-bit 计数器高字节

TK16DH/TK16DL 寄存器对用于存储触控按键功能 16-bit 计数器值。该 16-bit 计数器可用于校准参考振荡器或按键振荡器频率。当触控按键时隙计数器溢出，此 16-bit 计数器将停止，计数器内容保持不变。当 TKST 位置低，该寄存器对将被清零。

- **TKMn16DL 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 **D7~D0**：触控按键模块 n 16-bit C/F 计数器低字节

- **TKMn16DH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8**：触控按键模块 n 16-bit C/F 计数器高字节

TKMn16DH/TKMn16DL 寄存器对用于存储触控按键模块 n 16-bit C/F 计数器值。当触控按键时隙计数器溢出，该 16-bit C/F 计数器将被停止，其内容保持不变。当 TKST 位置低，该寄存器对将被清零。

- **TKMnROL 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 **D7~D0**：触控按键模块 n 参考振荡器电容选择低字节

- **TKMnROH 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 未定义，读为“0”

Bit 1~0 **D9~D8**：触控按键模块 n 参考振荡器电容选择高字节

TKMnROH/TKMnROL 寄存器对用于存储触控按键模块 n 参考振荡器电容值。当选中自动扫描模式，该寄存器将在当前时隙结束时从专用的触控按键数据存储器中载入相应的下一个时隙电容值。

参考振荡器内部电容值 = (TKMnRO[9:0]×50pF)/1024

- **TKMnC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MnMXS1	MnMXS0	MnDFEN	MnFILEN	MnSOFC	MnSOF2	MnSOF1	MnSOF0
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 **MnMXS1~MnMXS0**：多路复用按键选择

位	触控模块序号			
MnMXS[1:0]	**M0**	**M1**	**M2**	**M3**
00	Key 1	Key 5	Key 9	Key 13
01	Key 2	Key 6	Key 10	Key 14
10	Key 3	Key 7	Key 11	Key 15
11	Key 4	Key 8	Key 12	Key 16
BS83B08C	√	√	—	—
BS83B12C	√	√	√	—
BS83B16C	√	√	√	√

Bit 5 **MnDFEN**：倍频功能控制位
0：除能
1：使能
此位用于控制触控按键振荡器频率的倍频功能。当此位置“1”，按键振荡器频率将是原来的倍频。

Bit 4 **MnFILEN**：滤波功能控制位
0：除能
1：使能

Bit 3 **MnSOFC**：C/F 振荡器跳频功能选择位
0：软件处理跳频功能，由 MnSOF2~MnSOF0 位决定
1：硬件处理跳频功能，MnSOF2~MnSOF0 位无作用
该位用来选择触控按键振荡器跳频功能控制方式，当此位置 1，按键振荡器跳频功能由硬件电路控制，而不受 MnSOF2~MnSOF0 位影响。

Bit 2~0 **MnSOF2~MnSOF0**：触控按键模块 n 参考和按键振荡器跳频选择位
000：1.125MHz
001：1.111MHz
010：1.099MHz
011：1.085MHz
100：1.074MHz
101：1.059MHz
110：1.040MHz
111：1.020MHz
当外部或内部电容值不同时，这里提到的频率将会发生变化。如果触控按键振荡器工作频率为 1MHz，在选择其它频率时用户可以调整频率。

- **TKMnC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN
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 **MnTSS**：触控按键模块 n 时隙计数器时钟选择位
0：触控按键模块 n 参考振荡器
1：$f_{SYS}/4$

Bit 6 未使用，读为“0”

Bit 5 **MnROEN**：触控按键模块 n 参考振荡器使能控制位
0：除能
1：使能

Bit 4 **MnKOEN**：触控按键模块 n 按键振荡器使能控制位
0：除能
1：使能

Bit 3 **MnK4EN**：触控按键模块 n Key 4 使能控制

MnK4EN	触控按键模块 n (Mn)			
	M0	M1	M2	M3
0：除能	I/O 或其它功能			
1：使能	KEY4	KEY8	KEY12	KEY16
BS83B08C	√	√	—	—
BS83B12C	√	√	√	—
BS83B16C	√	√	√	√

Bit 2 **MnK3EN**：触控按键模块 n Key 3 使能控制

MnK4EN	触控按键模块 n (Mn)			
	M0	M1	M2	M3
0：除能	I/O 或其它功能			
1：使能	KEY3	KEY7	KEY11	KEY15
BS83B08C	√	√	—	—
BS83B12C	√	√	√	—
BS83B16C	√	√	√	√

Bit 1 **MnK2EN**：触控按键模块 n Key 2 使能控制

MnK4EN	触控按键模块 n (Mn)			
	M0	M1	M2	M3
0：除能	I/O 或其它功能			
1：使能	KEY2	KEY6	KEY10	KEY14
BS83B08C	√	√	—	—
BS83B12C	√	√	√	—
BS83B16C	√	√	√	√

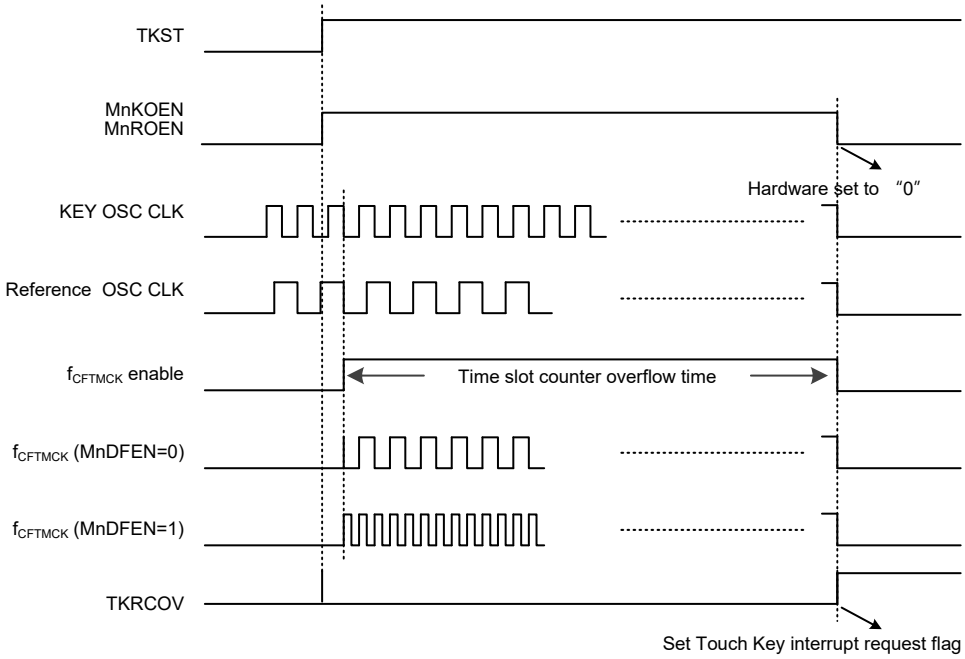
Bit 0 **MnK1EN**：触控按键模块 n Key 1 使能控制

MnK4EN	触控按键模块 n (Mn)			
	M0	M1	M2	M3
0：除能	I/O 或其它功能			
1：使能	KEY1	KEY5	KEY9	KEY13
BS83B08C	√	√	—	—
BS83B12C	√	√	√	—
BS83B16C	√	√	√	√

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。

触控按键感应振荡器和参考振荡器时序图如下图所示：



触控按键时序图

每个触控按键模块包含四个与逻辑 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。每个触控按键具有独立的感应振荡器，因此每个模块包含四个感应振荡器。

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。

通过设置 TKC0 寄存器中的 TSCS 位可以选择模块 0 的时隙计数器作为所有模块的时隙计数器。全部的触控按键模块共用一个起始信号，即 TKC0 寄存器中的 TKST。在此位清零时，所有模块的 16-bit C/F 计数器、16-bit 计数器和 5-bit

时隙计数器会自动清零，而 8-bit 可编程时隙计数器不清零，由用户设置溢出时间。在 TKST 上升沿时，16-bit C/F 计数器、16-bit 计数器、5-bit 时隙计数器和 8-bit 时隙计数器会自动开启。

当 5-bit 时隙计数器溢出，所有模块的按键振荡器和参考振荡器都会自动停止且 16-bit C/F 计数器、16-bit 计数器、5-bit 时隙计数器和 8-bit 时隙计数器会自动停止。时隙计数器和 8+5 位计数器时钟来自参考振荡器或 $f_{SYS}/4$。当 TKMnC1 寄存器中的 MnROEN 位为“1”时，选到的参考振荡器就使能。当 TKMnC1 寄存器中的 MnKOEN 为“1”时，选到的按键振荡器就使能。

当所有触控按键模块的时隙计数器都溢出时，或模块 0 时隙计数器溢出，才产生中断。这里所有的触控按键是指已使能的触控按键。

每 4 个按键为一个模块，所以 Key 1~Key 4 为模块 0，Key 5~Key 8 为模块 1，Key 9~Key 12 为模块 2，Key 13~Key 16 为模块 3，每个模块都是相同的架构。

触控按键中断

触控按键只有一个中断，所有触控按键模块的时隙计数器都溢出时，或模块 0 时隙计数器溢出时，才产生中断，这里所有的触控按键是指已使能的触控按键。此时所有模块的 16-bit C/F 计数器、16-bit 计数器，5-bit 时隙计数器和 8-bit 时隙计数器会自动清零。更多详细内容见规格书中断章节中的“触控按键中断”。

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，触控按键检测程序启动。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，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。

串行接口模块 – SIM

该系列单片机内有一个串行接口模块，包括两种易与外部设备通信的串行接口：四线 SPI 或两线 I^2C 接口。这两种接口具有相当简单的通信协议，单片机可以通过这些接口与传感器、闪存或 EEPROM 内存等硬件设备通信。SIM 接口的引脚与 I/O 引脚共用，所以要使用 SIM 功能时应先通过 SIMC0 寄存器的相关位选择 SIM 功能。因为这两种接口共用引脚和寄存器，所以要通过一个 SIMC0 寄存器中的 SIM2~SIM0 位来选择哪一种通信接口。若 SIM 功能使能，可通过上拉电阻控制寄存器选择与输入 / 输出口共用的 SIM 脚的上拉电阻。

SPI 接口

SPI 接口常用于与外部设备如传感器、闪存或 EEPROM 内存等通信。四线 SPI 接口最初是由摩托罗拉公司研制，是一个有相当简单的通信协议的串行数据接口，这个协议可以简化与外部硬件的编程要求。

SPI 通信模式为全双工模式，且能以主 / 从模式的工作方式进行通信，单片机既可以做为主机，也可以做为从机。虽然 SPI 接口理论上允许一个主机控制多个从机，但此处的 SPI 中只有一个片选信号引脚 $\overline{SCS}$。若主机需要控制多个从机，可使用输入 / 输出引脚选择从机。

SPI 接口操作

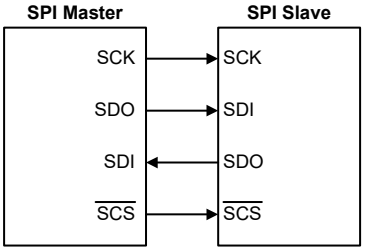
SPI 接口是一个全双工串行数据传输器。SPI 接口的四线为：SDI、SDO、SCK 和 $\overline{SCS}$。SDI 和 SDO 是数据的输入和输出线。SCK 是串行时钟线，$\overline{SCS}$ 是从机的选择线。SPI 的接口引脚与普通 I/O 口和 I^2C 的功能脚共用。通过设定

SIMC0/SIMC2 寄存器的对应位，来使能 SPI 接口。SPI 可以通过 SIMC0 寄存器中的 SIMEN 位来除能或使能。连接到 SPI 接口的单片机以从主 / 从模式进行通信，且主机完成所有的数据传输初始化，并控制时钟信号。由于单片机只有一个 $\overline{SCS}$ 引脚，所以只能拥有一个从机设备。可通过软件控制 $\overline{SCS}$ 引脚使能与除能，设置 CSEN 位为“1”使能 $\overline{SCS}$ 功能，设置 CSEN 位为“0”，$\overline{SCS}$ 引脚将处于浮空状态。

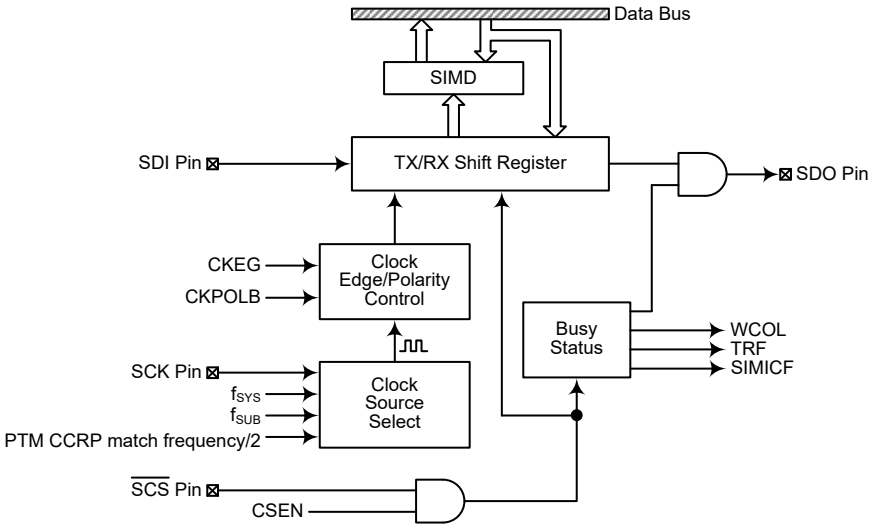
该系列单片机的 SPI 功能具有以下特点：

- 全双工同步数据传输
- 主从模式
- 最低有效位先传或最高有效位先传的数据传输模式
- 传输完成标志位
- 时钟源上升沿或下降沿有效

SPI 接口状态受很多因素的影响，如单片机处于主机或从机的工作模式和 CSEN，SIMEN 位的状态。



SPI 主 / 从机连接方式



SPI 方框图

SPI 寄存器

有三个内部寄存器用于控制 SPI 接口的所有操作，其中有一个数据寄存器 SIMD、两个控制寄存器 SIMC0 和 SIMC2。注意，SIMC1 寄存器仅用于 I^2C 接口。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
SIMD	D7	D6	D5	D4	D3	D2	D1	D0

SPI 寄存器列表

- **SIMD 寄存器**

SIMD 寄存器用于存储发送和接收的数据。这个寄存器由 SPI 和 I²C 功能所共用。在单片机尚未将数据写入到 SPI 总线中时，要传输的数据应先存在 SIMD 中。SPI 总线接收到数据之后，单片机就可以从 SIMD 数据寄存器中读取。所有通过 SPI 传输或接收的数据都必须通过 SIMD 寄存器实现。

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	x	x	x	x	x	x	x	x

“x”：未知

该系列单片机中也有两个控制 SPI 接口功能的寄存器，SIMC0 和 SIMC2。应注意的是 SIMC2 与 I²C 接口功能中的寄存器 SIMA 是同一个寄存器。SPI 功能不会用到寄存器 SIMC1，SIMC1 只适用于 I²C 中。寄存器 SIMC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 SIMC2 用于其它的控制功能如 LSB/MSB 选择，写冲突标志位等。

- **SIMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7~5 **SIM2~SIM0**：SIM 工作模式控制位
000：SPI 主机模式；SPI 时钟为 $f_{SYS}/4$
001：SPI 主机模式；SPI 时钟为 $f_{SYS}/16$
010：SPI 主机模式；SPI 时钟为 $f_{SYS}/64$
011：SPI 主机模式；SPI 时钟为 f_{SUB}
100：SPI 主机模式；SPI 时钟为 PTM CCRP 匹配频率 / 2
101：SPI 从机模式
110：I²C 从机模式
111：非 SIM 功能
这几位用于设置 SIM 功能的工作模式，用于选择 SPI 的主从模式和 SPI 的主机时钟频率及 I²C 或 SPI 功能。SPI 时钟源可来自于系统时钟也可以选择来自 PTM 和 f_{SUB}。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 未定义，读为“0”

Bit 3~2 **SIMDEB1~SIMDEB0**：I²C 去抖时间选择位
这些位只有在 SIM 设置成 I²C 接口模式时才有效。请参考 I²C 寄存器部分。

Bit 1 **SIMEN**：SIM 使能控制位
0：除能
1：使能
此位为 SIM 接口的开 / 关控制位。此位为“0”时，SIM 接口除能，SDI、SDO、

SCK 和 $\overline{SCS}$ 或 SDA 和 SCL 脚将失去 SPI 或 I^2C 功能，SIM 工作电流减小到最小值。此位为“1”时，SIM 接口使能。若 SIM 经由 SIM2~SIM0 位设置为工作在 SPI 接口，当 SIMEN 位由低到高转变时，SPI 控制寄存器中的设置不会发生变化，其首先应在应用程序中初始化。若 SIM 经由 SIM2~SIM0 位设置为工作在 I^2C 接口，当 SIMEN 位由低到高转变时，I^2C 控制寄存器中的设置，如 HTX 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I^2C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0 **SIMICF**：SIM 未完成标志位
0：未发生
1：发生
此位仅当 SIM 配置在 SPI 从机模式时有效。如果 SPI 工作在从机模式且 SIMEN 和 CSEN 位都为“1”，但在 SPI 数据传输完全结束前 $\overline{SCS}$ 线被外部主机拉高，SIMICF 和 TRF 位都会被置高。在这种情况下，如果相应的中断功能使能将产生一个中断。然而，如果 SIMICF 位是由软件应用程序设为 1，那么 TRF 位将不会置高。

- **SIMC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
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 **D7~D6**：未定义位
用户可通过软件程序对这两位进行读写。

Bit 5 **CKPOLB**：SPI 时钟线的基础状态位
0：当时钟无效时，SCK 口为高电平
1：当时钟无效时，SCK 口为低电平
此位决定了时钟线的基础状态，当时钟无效时，若此位为高，SCK 为低电平，若此位为低，SCK 为高电平。

Bit 4 **CKEG**：SPI 的 SCK 有效时钟边沿类型位
CKPOLB=0
0：SCK 为高电平且在 SCK 上升沿抓取数据
1：SCK 为高电平且在 SCK 下降沿抓取数据
CKPOLB=1
0：SCK 为低电平且在 SCK 下降沿抓取数据
1：SCK 为低电平且在 SCK 上升沿抓取数据
CKEG 和 CKPOLB 位用于设置 SPI 总线上时钟信号输入和输出方式。在执行数据传输前，这两位必须被设置，否则将产生错误的时钟边沿信号。CKPOLB 位决定时钟线的基本状态，若时钟无效且此位为高，则 SCK 为低电平，若时钟无效且此位为低，则 SCK 为高电平。CKEG 位决定有效时钟边沿类型，取决于 CKPOLB 的状态。

Bit 3 **MLS**：SPI 数据移位命令位
0：LSB 优先
1：MSB 优先
数据移位选择位，用于选择数据传输时高位优先传输还是低位优先传输。此位设置为高时高位优先传输，为低时低位优先传输。

Bit 2 **CSEN**：SPI $\overline{SCS}$ 引脚控制位
0：除能
1：使能
CSEN 位用于 $\overline{SCS}$ 引脚的使能 / 除能控制。此位为低时，$\overline{SCS}$ 除能并作为 I/O 引脚或其它共用功能。此位为高时，$\overline{SCS}$ 使能并作为选择脚。

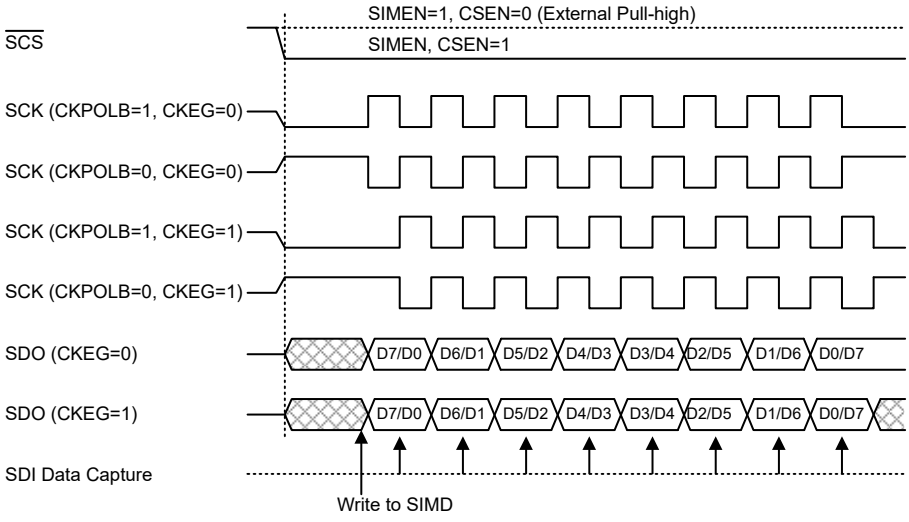
Bit 1 **WCOL**：SPI 写冲突标志位
0：无冲突
1：冲突
WCOL 标志位用于监测数据冲突的发生。此位为高时，数据在传输时被写入 SIMD 寄存器。若数据正在被传输时，此操作无效。此位可被应用程序清零。

Bit 0 **TRF**：SPI 发送 / 接收结束标志位
0：数据正在发送
1：数据发送结束
TRF 位为发送 / 接收结束标志位，当 SPI 数据传输结束时，此位自动置为高，但需通过应用程序设置为“0”。此位也可用于产生中断。

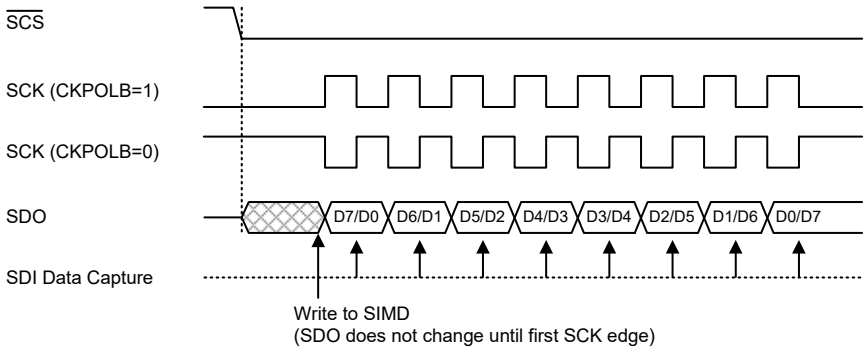
SPI 通信

将 SIMEN 设置为高，使能 SPI 功能之后，单片机处于主机模式，当数据写入到寄存器 SIMD 的同时传输 / 接收开始进行。数据传输完成时，TRF 位将自动被置位，但清零只能通过应用程序完成。单片机处于从机模式时，收到主机发来的信号之后，会传输到 SIMD 中的数据，而且在 SDI 引脚上的数据也会被移位到 SIMD 寄存器中。主机应在输出时钟信号之前先输出一个 $\overline{SCS}$ 信号以使能从机，从机的数据传输功能也应在与 SCK 信号相关的适当时候准备就绪，这由 CKPOLB 和 CKEG 位决定。所附时序图表明了在 CKPOLB 和 CKEG 位各种设置情况下从机数据与 SCK 信号的关系。

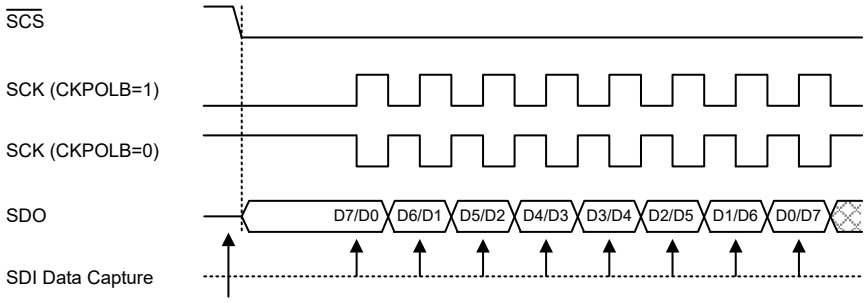
即使在单片机处于空闲模式，如果所选的 SPI 时钟源有效，SPI 主机功能仍将继续执行。



SPI 主机模式时序

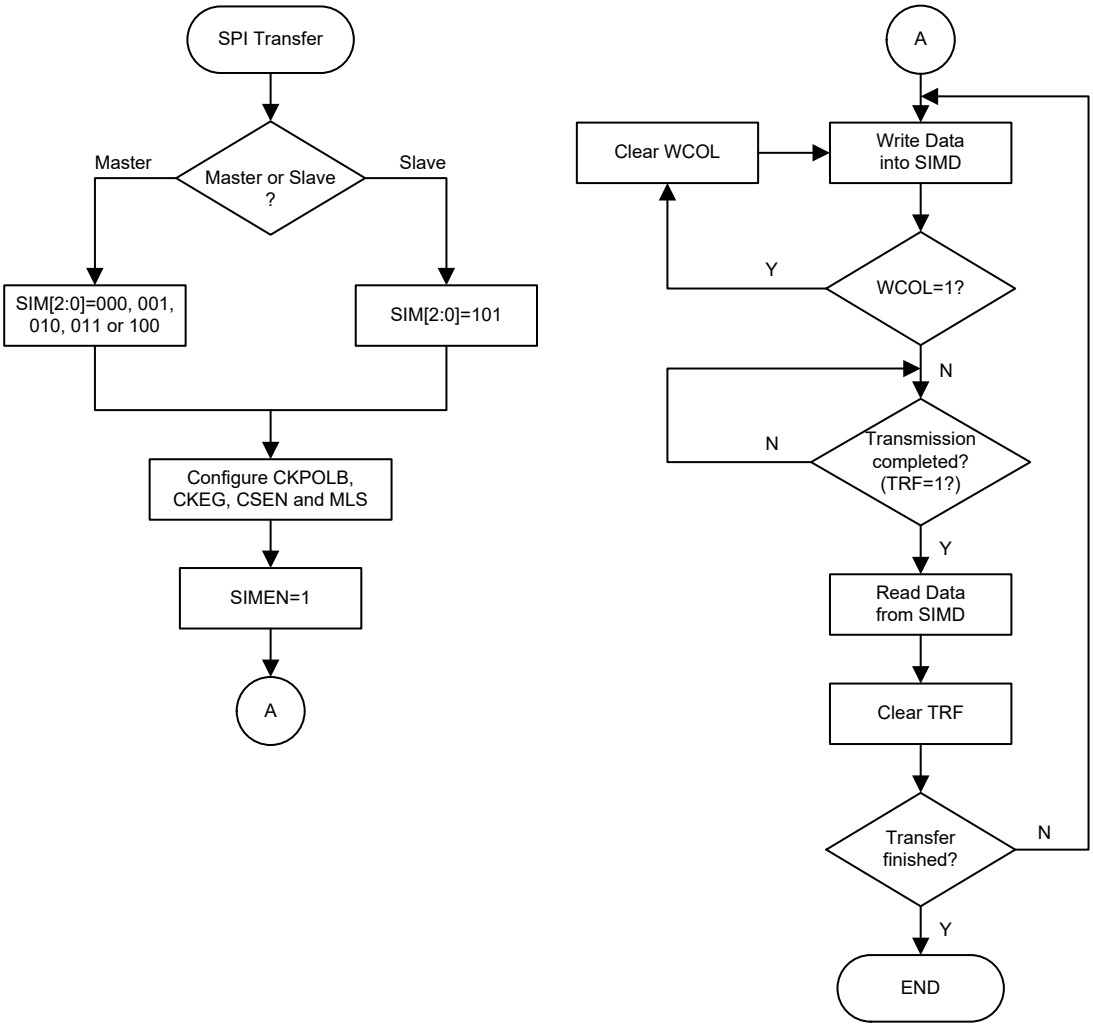


SPI 从机模式时序 – CKEG=0



Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always enabled and ignores the $\overline{SCS}$ level.

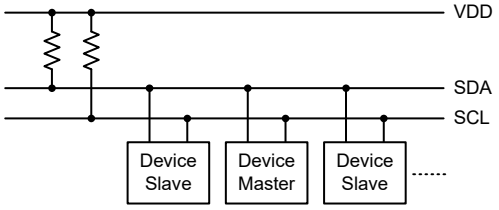
SPI 从机模式时序 – CKEG=1


SPI Transfer
Master or Slave ?
Master
Slave
SIM[2:0]=000, 001, 010, 011 or 100
SIM[2:0]=101
Configure CKPOLB, CKEG, CSEN and MLS
SIMEN=1
A
A
Clear WCOL
Write Data into SIMD
WCOL=1?
Y
N
Transmission completed? (TRF=1?)
N
Y
Read Data from SIMD
Clear TRF
Transfer finished?
N
Y
END

SPI 传输控制流程图

I²C 接口

I²C 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。I²C 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。

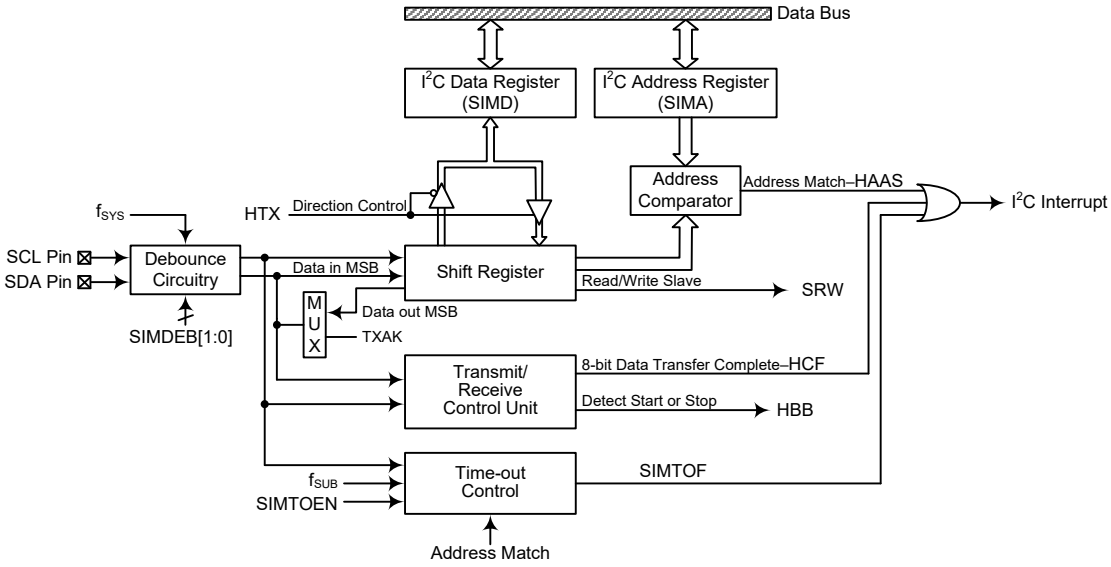


I²C 主从总线连接图

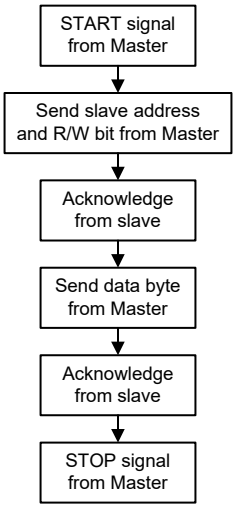
I²C 接口操作

I²C 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，I²C 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 I²C 通信。

如果有两个设备通过双向的 I²C 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 I²C 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。



I²C 方框图



I²C 接口操作

SIMDEB1 和 SIMDEB0 位决定 I²C 接口的去抖时间。这个功能可以使用内部时钟在外部时钟上增加一个去抖间隔，会减小时钟线上毛刺发生的可能性，以避免单片机发生误动作。如果选择了这个功能，去抖时间可以选择 2 个或 4 个系统时钟。为了达到需要的 I²C 数据传输速度，系统时钟 f_{SYS} 和 I²C 去抖时间之间存在一定的关系。I²C 标准模式或者快速模式下，用户需注意所选的系统时钟频率与标准匹配去抖时间的设置，其具体关系如下表所示。

I²C 去抖时间选择	I²C 标准模式 (100kHz)	I²C 快速模式 (400kHz)
无去抖时间	f_{SYS} > 2MHz	f_{SYS} > 5MHz
2 个系统时钟去抖时间	f_{SYS} > 4MHz	f_{SYS} > 10MHz
4 个系统时钟去抖时间	f_{SYS} > 8MHz	f_{SYS} > 20MHz

I²C 最小 f_{SYS} 频率要求

I²C 寄存器

I²C 总线有三个控制寄存器 SIMC0、SIMC1 和 SIMTOC，及一个从机地址寄存器 SIMA 和一个数据寄存器 SIMD。SIMD 寄存器，SPI 章节中已有介绍，用于存储正在 I²C 总线上传输和接收的数据，当单片机将数据写入 I²C 总线之前，实际将被传输的数据存放在寄存器 SIMD 中。从 I²C 总线接收到数据之后，单片机就可以从寄存器 SIMD 中读取这个数据。I²C 总线上的所有传输或接收到的数据都必须通过寄存器 SIMD。

应注意的是 SIMA 寄存器也有另外一个名字，SIMC2，使用 SPI 功能时会用到。I²C 接口会用到寄存器 SIMC0 中的 SIMEN 位和 SIM2~SIM0 位。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMA	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C 寄存器列表

- **SIMD 寄存器**

SIMD 用于存储发送和接收的数据。这个寄存器由 SPI 和 I²C 功能所共用。在单片机尚未将数据写入到 I²C 总线中时，要传输的数据应存在 SIMD 中。I²C 总线接收到数据之后，单片机就可以从 SIMD 数据寄存器中读取。所有通过 I²C 传输或接收的数据都必须通过寄存器 SIMD 实现。

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	x	x	x	x	x	x	x	x

“x”：未知

- **SIMA 寄存器**

SIMA 寄存器也在 SPI 接口功能中使用，但其名称改为 SIMC2。SIMA 寄存器用于存放 7 位从机地址，寄存器 SIMA 中的 bit 7~bit 1 是单片机的从机地址，Bit 0 未定义。

如果接至 I²C 的主机发送处的地址和寄存器 SIMA 中存储的地址相符，那么就选中了这个从机。应注意的是寄存器 SIMA 和 SPI 接口使用的寄存器 SIMC2 是同一个寄存器。

Bit	7	6	5	4	3	2	1	0
Name	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”：未知

Bit 7~1 **SIMA6~SIMA0**：I²C 从机地址位
SIMA6~SIMA0 是 I²C 从机地址 bit 6~bit 0

Bit 0 **D0**：未定义位
此位可通过软件程序进行读写。

单片机中有三个控制 I²C 接口功能的寄存器，SIMC0、SIMC1 和 SIMTOC。寄存器 SIMC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 SIMC1 包括多个用于表明 I²C 传输状态的相关标志位。SIMTOC 寄存器用于控制 I²C 总线超时功能，在 I²C 超时控制章节有描述。

- **SIMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7~5 **SIM2~SIM0**：SIM 工作模式控制位
000：SPI 主机模式；SPI 时钟为 $f_{SYS}/4$
001：SPI 主机模式；SPI 时钟为 $f_{SYS}/16$
010：SPI 主机模式；SPI 时钟为 $f_{SYS}/64$
011：SPI 主机模式；SPI 时钟为 f_{SUB}
100：SPI 主机模式；SPI 时钟为 PTM CCRP 匹配频率 / 2
101：SPI 从机模式
110：I²C 从机模式
111：非 SIM 功能

这几位用于设置 SIM 功能的工作模式，用于选择 SPI 的主从模式和 SPI 的主机时钟频率及 I²C 或 SPI 功能。SPI 时钟源可来自于系统时钟也可以选择来自 PTM 或 f_{SUB}。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4　未定义，读为“0”

Bit 3~2　**SIMDEB1~SIMDEB0**：I²C 去抖时间选择位
00：无去抖时间
01：2 个系统时钟去抖时间
1x：4 个系统时钟去抖时间

Bit 1　**SIMEN**：SIM 使能控制位
0：除能
1：使能
此位为 SIM 接口的开 / 关控制位。此位为“0”时，SIM 接口除能，SDI、SDO、SCK 和 $\overline{SCS}$ 或 SDA 和 SCL 脚将失去 SPI 或 I²C 功能，SIM 工作电流减小到最小值。此位为“1”时，SIM 接口使能。若 SIM 经由 SIM2~SIM0 位设置为工作在 SPI 接口，当 SIMEN 位由低到高转变时，SPI 控制寄存器中的设置不会发生变化，其首先应在应用程序中初始化。若 SIM 经由 SIM2~SIM0 位设置为工作在 I²C 接口，当 SIMEN 位由低到高转变时，I²C 控制寄存器中的设置，如 HTX 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I²C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0　**SIMICF**：SIM 未完成标志位
0：未发生
1：发生
此位仅当 SIM 配置在 SPI 从机模式时有效。请参考 SPI 寄存器部分。

- **SIMC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
R/W	R	R	R	R/W	R/W	R/W	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7　**HCF**：I²C 总线数据传输结束标志位
0：数据正在被传输
1：8 位数据传输完成
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　**HAAS**：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　**HBB**：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。当检测到 STOP 信号时 I²C 总线停止，该位变为低。

Bit 4　**HTX**：I²C 从机处于发送或接收模式选择位
0：从机处于接收模式
1：从机处于发送模式

Bit 3　**TXAK**：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
从机接收 8 位数据之后会将该位在第九个时钟传到总线上。如果从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 **SRW**：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
SRW 位是 I²C 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，HAAS 位会被设置为高，从机将检测 SRW 位来决定进入发送模式还是接收模式。如果 SRW 位为高时，主机会请求从总线上读数据，此时从机处于传输模式。当 SRW 位为低时，主机往总线上写数据，从机处于接收模式以读取该数据。

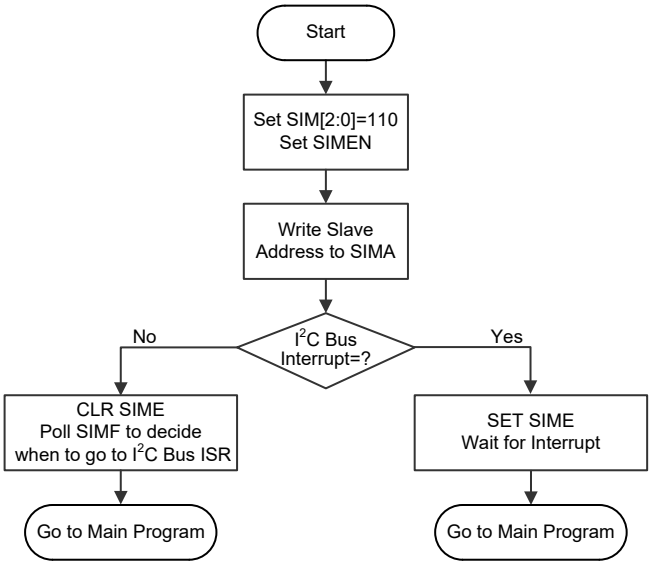
Bit 1 **IAMWU**：I²C 地址匹配唤醒控制位
0：除能
1：使能 – 唤醒后必须由应用程序清零
此位应设置为“1”使能 I²C 地址匹配以使系统从休眠或空闲模式中唤醒。若进入休眠或空闲模式前 IAMWU 已经设置以使能 I²C 地址匹配唤醒功能，在系统唤醒后须应用程序清零此位以确保单片机正确地运行。

Bit 0 **RXAK**：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
RXAK 位是接收确认标志位。如果 RXAK 位为“0”即 8 位数据传输之后，从机在第九个时钟有接收到一个正确的确认位。如果从机处于发送状态，从机会检查 RXAK 位来判断主机是否愿意继续接收下一个字节。因此直到 RXAK 为“1”时，从机传输方停止发送数据。这时，从机传输方将释放 SDA 线，主机发出停止信号释放 I²C 总线。

I²C 总线通信

I²C 总线上的通信需要四步完成，一个起始信号，一个从机地址发送，一个数据传输，还有一个停止信号。当起始信号被写入 I²C 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，SIMC1 寄存器的 HAAS 位会被置位，同时产生 I²C 中断。进入中断服务程序后，系统要检测 HAAS 位和 SIMTOF 位，以判断 I²C 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，或是来自 I²C 总线超时。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 SRW 位中。从机通过检测 SRW 位以确定主控制器是要进入发送模式还是接收模式。在 I²C 总线开始传送数据前，需要先初始化 I²C 总线，初始化 I²C 总线步骤如下：

- 步骤 1
  设置 SIMC0 寄存器中 SIM2~SIM0 位为“110”和 SIMEN 位为“1”，以使能 I²C 总线。
- 步骤 2
  向 I²C 总线地址寄存器 SIMA 写入从机地址。
- 步骤 3
  设置中断控制寄存器中的 SIME 中断使能位，以使能 SIM 中断。



I²C 总线初始化流程图

I²C 总线起始信号

起始信号只能由连接 I²C 总线的主机产生，而不是由从机产生。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 HBB。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。

I²C 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 SIMC1 寄存器的 SRW 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 HAAS 置位。

I²C 总线有三个中断源，当程序运行至中断服务子程序时，通过检测 HAAS 位和 SIMTOF 位，以判断 I²C 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，或是来自 I²C 总线超时。当是从机地址匹配发生中断时，则从机是用于发送模式并将数据写进 SIMD 寄存器，或是用于接收模式并从 SIMD 寄存器中读取空值以释放 SCL 线。

I²C 总线读 / 写信号

SIMC1 寄存器的 SRW 位用来表示主机是要从 I²C 总线上读取数据还是要将数据写到 I²C 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 SRW 置“1”，表示主机要从 I²C 总线上读取数据，从机则作为发送方，将数据写到 I²C 总线；当 SRW 清“0”，表示主机要写数据到 I²C 总线上，从机则做为接收方，从 I²C 总线上读取数据。

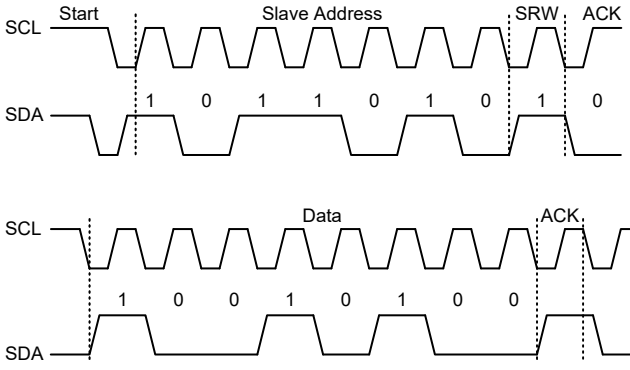
I²C 总线从机地址确认信号

主机发送呼叫地址后，当 I²C 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 HAAS 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 SRW 位，以确定自己是作为发送方还是作为接收方。如果 SRW 位为高，从机须设置成发送方，这样会置位 SIMC1 寄存器的 HTX 位。如果 SRW 位为低，从机须设置成接收方，这样会清零 SIMC1 寄存器的 HTX 位。

I²C 总线数据和确认信号

在从机确认接收到从机地址后，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 I²C 总线。所传送的数据存储在 SIMD 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 SIMD 寄存器中；如果设置成接收方，从机必须从 SIMD 寄存器读取数据。

当从机接收器想要继续接收下一个数据时，必须在第 9 个时钟发出应答信号 (TXAK)。被设为发送方的从机将检测寄存器 SIMC1 中的 RXAK 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。

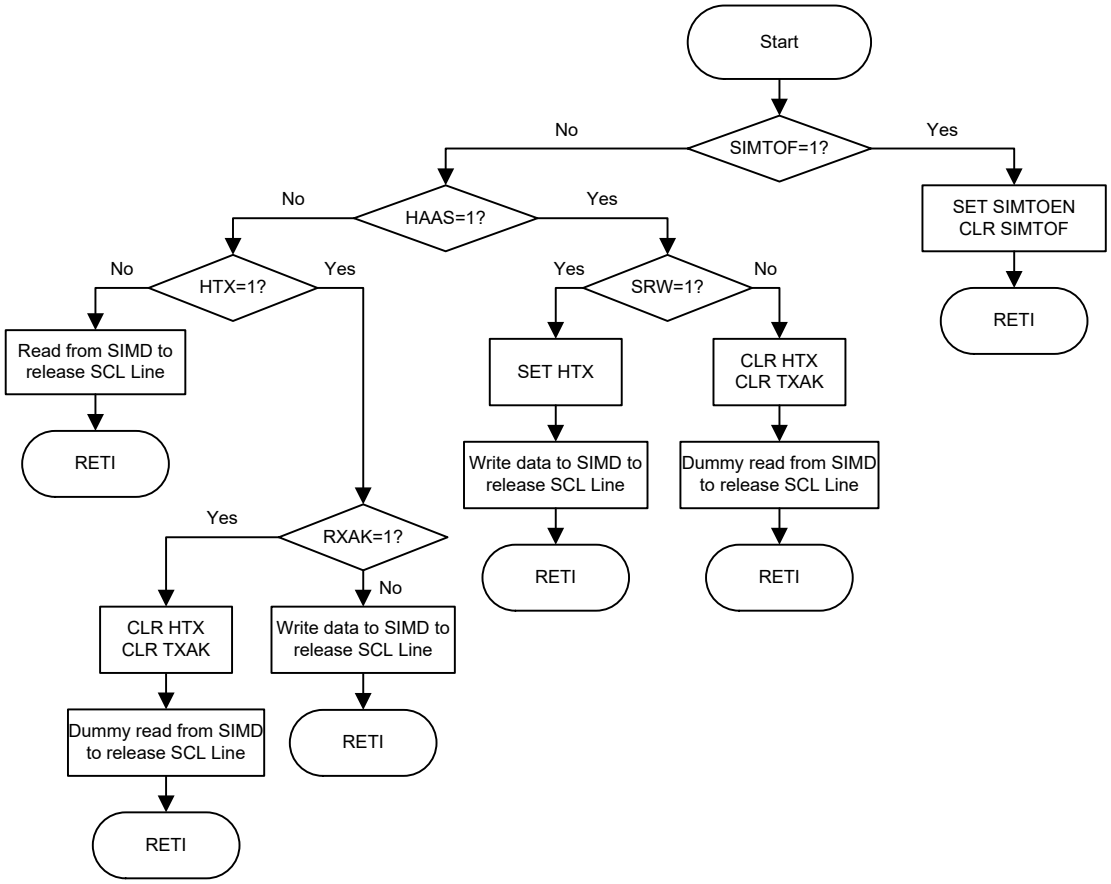


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	·····	S	SA	SR	M	D	A	D	A	·····	P

I²C 通信时序图

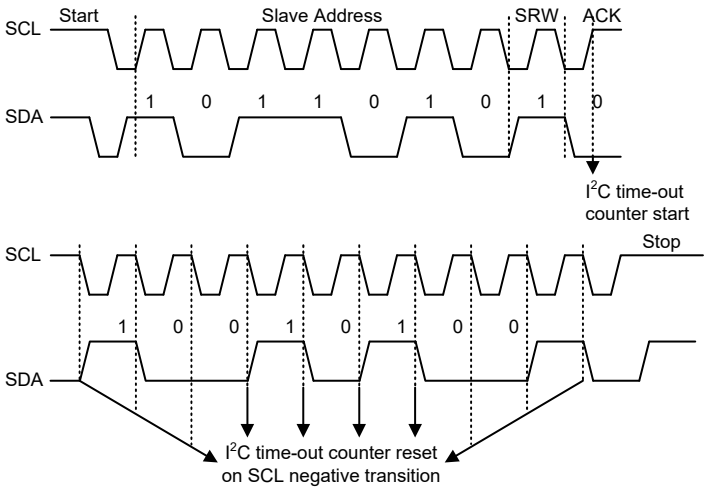
注：当从机地址匹配时，从机必须选择设置为发送模式还是接收模式。若设置为发送模式，需写数据至 SIMD 寄存器；若设置为接收模式，需立即从 SIMD 寄存器中虚读数据以释放 SCL 线。



I²C 总线 ISR 流程图

I²C 超时控制

超时功能可减少 I²C 接收错误的时钟源而引起的锁死问题。如果连接到 I²C 总线的时钟源经过一段时间还未接收到，则在一定的超时周期后，I²C 电路和寄存器将复位。超时计数器在 I²C 总线“START”和“地址匹配”条件下开始计数，且在 SCL 下降沿清零。在下一个 SCL 下降沿到来之前，如果超时时间大于 SIMTOC 寄存器指定的超时周期，则超时发生。I²C“STOP”条件发生时超时功能终止。



I²C 超时时序图

当 I²C 超时计数器溢出时，计数器将停止计数，SIMTOEN 位被清零，且 SIMTOF 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	I²C 超时发生后
SIMD，SIMA，SIMC0	保持不变
SIMC1	复位至 POR

超时发生后的 I²C 寄存器

SIMTOF 标志位由应用程序清零。共有 64 个超时周期，可通过 SIMTOC 寄存器的 SIMTOS 位段进行选择。超时周期可通过公式计算：$((1\sim64)\times(32/f_{SUB}))$。由此可得超时周期范围为 1ms~64ms。

- **SIMTOC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0
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 **SIMTOEN**：SIM I²C 超时功能控制位
0：除能
1：使能

Bit 6 **SIMTOF**：SIM I²C 超时标志位
0：未发生
1：发生

Bit 5~0 **SIMTOS5~SIMTOS0**：SIM I²C 超时时间选择位
I²C 超时时钟源是 $f_{SUB}/32$
I²C 超时时间计算方法：$(SIMTOS[5:0]+1)\times(32/f_{SUB})$

中断

中断是单片机一个重要功能。当外部事件或内部功能如触摸动作或定时器模块有效，并且产生中断时，系统会暂时中止当前的程序而转到执行相对应的中断服务程序。此系列单片机提供多个外部中断和内部中断功能，外部中断由 INT 引脚动作产生，而内部中断由各种内部功能，如触控按键、定时器模块、时基和 SIM 等产生。

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特殊功能数据存储器中的一系列寄存器控制的。寄存器总的分为三类。第一类是 INTC0~INTC1 寄存器，用于设置基本的中断；第二类是 MFI 寄存器，用于设置多功能中断；最后一种有 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	—
INT 引脚	INTE	INTF
触控按键模块	TKME	TKMF
时基	TBE	TBF
多功能	MFE	MFF
EEPROM	DEE	DEF
SIM	SIME	SIMF
PTM	PTMPE	PTMPF
	PTMAE	PTMAF

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	—	MFF	TKMF	INTF	MFE	TKME	INTE	EMI
INTC1	—	DEF	TBF	SIMF	—	DEE	TBE	SIME
MFI	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE

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	—	MFF	TKMF	INTF	MFE	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 **MFF**：多功能中断请求标志位
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 **MFE**：多功能中断控制位
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	—	DEF	TBF	SIMF	—	DEE	TBE	SIME
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 **DEF**：数据 EEPROM 中断请求标志位
0：无请求
1：中断请求

Bit 5 **TBF**：时基中断请求标志位
0：无请求
1：中断请求

Bit 4 **SIMF**：SIM 中断请求标志位
0：无请求
1：中断请求

Bit 3 未定义，读为“0”

Bit 2 **DEE**：数据 EEPROM 中断控制位
0：除能
1：使能

Bit 1 **TBE**：时基中断控制位
0：除能
1：使能

Bit 0 **SIME**：SIM 中断控制位
0：除能
1：使能

- **MFI 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 未定义 , 读为“0”

Bit 5 **PTMAF**：PTM CCRA 比较器中断请求标志位
0：无请求
1：中断请求

Bit 4 **PTMPF**：PTM CCRP 比较器中断请求标志位
0：无请求
1：中断请求

Bit 3~2 未定义，读为“0”

Bit 1 **PTMAE**：PTM CCRA 比较器中断控制位
0：除能
1：使能

Bit 0 **PTMPE**：PTM CCRP 比较器中断控制位
0：除能
1：使能

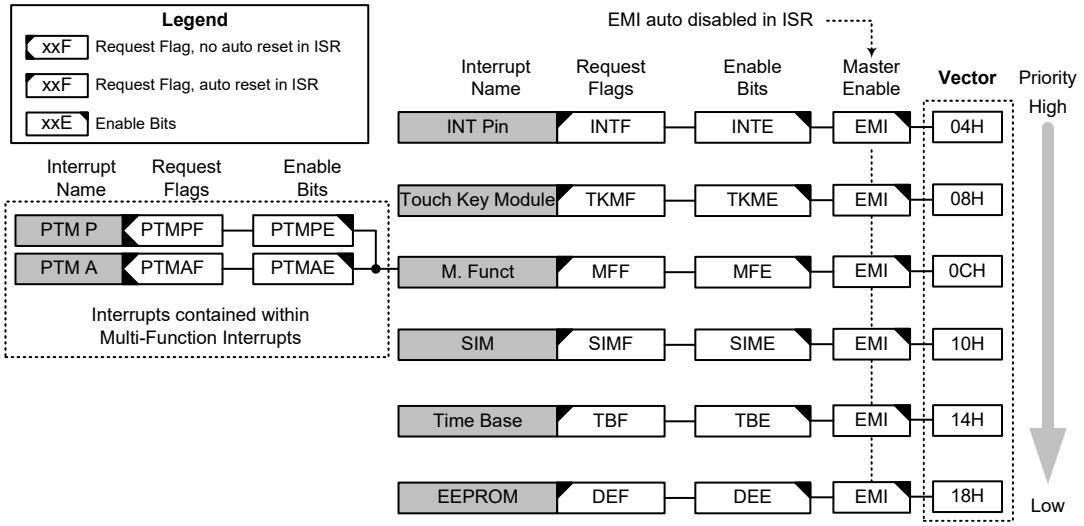
中断操作

若中断事件条件产生，如一个触控按键计数器溢出、TM 比较器 P 或比较器 A 匹配等，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“JMP”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，系统将自动清零 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

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，其上拉电阻选择仍保持有效。

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。

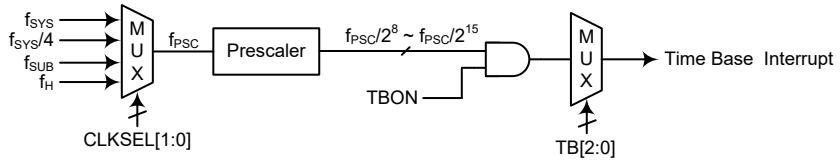
触控按键中断

要使触控按键中断发生，总中断控制位 EMI 和触摸按键中断使能位 TKME 必须先被置位。当触控按键模块中的时隙计数器溢出时，触摸按键中断请求标志 TKMF 将被置位，触控按键中断产生。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，将调用相应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。

时基中断

时基中断提供一个固定周期的中断信号，由各自的定时器功能产生溢出信号控制。当各自的中断请求标志 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。其时钟源 f_{PSC} 来自内部时钟源 f_{SYS}、$f_{SYS}/4$、f_{SUB} 或 f_H，经过一个分频器，其分频比可通过配置 TBC 寄存器中的位选择以获得更长的中断周期。时钟源控制时基中断周期，分别通过 PSCR 寄存器中的 CLKSEL[1:0] 位进行选择。



时基中断

- **PSCR 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL1	CLKSEL0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未定义，读为“0”

Bit 1~0　**CLKSEL1~CLKSEL0**：时基预分频器时钟源 f_{PSC} 选择
00：f_{SYS}
01：$f_{SYS}/4$
10：f_{SUB}
11：f_H

- **TBC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	TBON	TB2	TB1	TB0
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　**TBON**：时基功能使能控制位
0：除能
1：使能

Bit 2~0　**TB2~TB0**：时基溢出周期选择位
000：$2^8/f_{PSC}$
001：$2^9/f_{PSC}$
010：$2^{10}/f_{PSC}$
011：$2^{11}/f_{PSC}$
100：$2^{12}/f_{PSC}$
101：$2^{13}/f_{PSC}$
110：$2^{14}/f_{PSC}$
111：$2^{15}/f_{PSC}$

多功能中断

此系列单片机中有一个多功能中断，与其它中断不同，它没有独立源，但由其它现有的中断源构成，即 PTM 中断。

当多功能中断请求标志 MFF 被置位，多功能中断请求产生。当中断使能，堆栈未满，包括在多功能中断中的任意一个中断发生时，将调用多功能中断向量中的一个子程序。当响应中断服务子程序时，相关的多功能请求标志位会自动复位且 EMI 位会自动清零以除能其它中断。

但必须注意的是，在中断响应时，虽然多功能中断标志会自动复位，但多功能中断源的请求标志位，即 PTM 中断的请求标志位不会自动复位，必须由应用程序清零。

串行接口模块中断

串行接口模块中断，即 SIM 中断，由 SPI 或 I^2C 数据传输控制。当一个字节数据已由 SIM 接口接收或发送完，或 I^2C 从机地址匹配，或 I^2C 总线发生超时，中断请求标志 SIMF 被置位，SIM 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 SIME 需先被置位。当中断使能，堆栈未满且以上任何一种情况发生时，可跳转至相关 SIM 中断向量子程序中执行。当串行接口中断响应，相应的中断请求标志位 SIMF 会自动复位且 EMI 位会被清零以除能其它中断。

EEPROM 中断

当写周期结束，EEPROM 写中断请求标志 DEF 被置位，EEPROM 写中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和 EEPROM 写中断使能位 DEE 需先被置位。当中断使能，堆栈未满且 EEPROM 写周期结束时，可跳转至相关中断向量子程序中执行。当 EEPROM 中断响应，DEF 标志会自动复位且 EMI 将被自动清零以除能其它中断。

TM 中断

周期型 TM 有两个中断，分别来自比较器 P 和比较器 A 匹配，都属于多功能中断。TM 有两个中断请求标志位及两个使能位。当 TM 比较器 P 和比较器 A 匹配情况发生时，相应 TM 中断请求标志被置位，TM 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 和相应 TM 中断使能位和相关多功能中断使能位 MFE 需先被置位。当中断使能，堆栈未满且 TM 比较器匹配情况发生时，可跳转至相关多功能中断向量子程序中执行。当 TM 中断响应，EMI 将被自动清零以除能其它中断，相关 MFF 标志也可自动清零，但 TM 中断请求标志需在应用程序中手动清零。

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或低电压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应用程序清零。

多功能中断中所含中断相应程序执行时，多功能中断请求标志 MFF 可以自动清零，但各自的请求标志需在应用程序中手动清零。

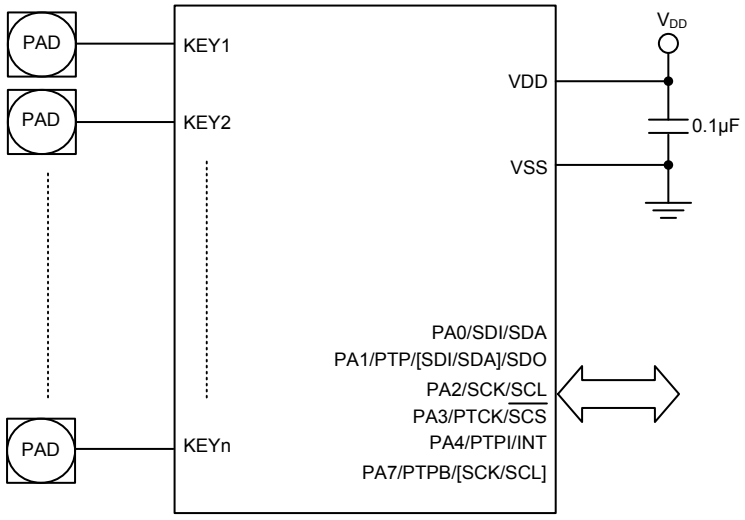
建议在中断服务子程序中不要使用“CALL 子程序”指令。中断通常发生在不可预料的情况或是需要立刻执行的某些应用。假如只剩下一层堆栈且没有控制好中断，当“CALL 子程序”在中断服务子程序中执行时，将破坏原来的控制序列。

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，清零 EMI 位，除能进一步中断。

应用电路


PAD
PAD
PAD
KEY1
KEY2
KEYn
VDD
VSS
V_{DD}
0.1μF
PA0/SDI/SDA
PA1/PTP/[SDI/SDA]/SDO
PA2/SCK/SCL
PA3/PTCK/SCS
PA4/PTPI/INT
PA7/PTPB/[SCK/SCL]

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 Holtek 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 Holtek 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 Holtek 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 Holtek 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，Holtek 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] ← 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

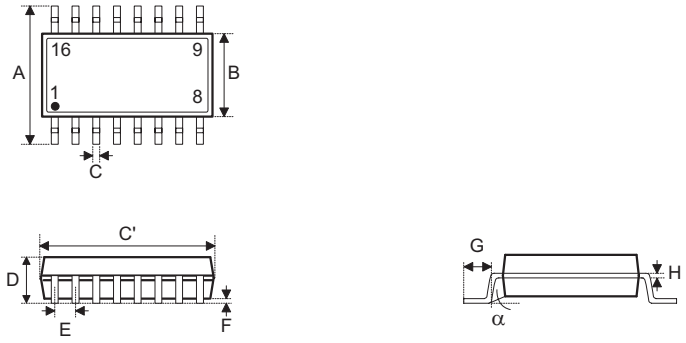
封装信息

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

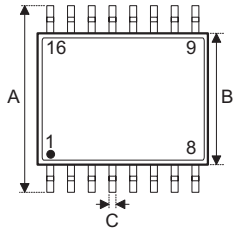
16-pin NSOP (150mil) 外形尺寸

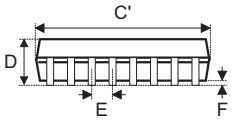


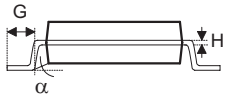
符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A		0.236 BSC	
B		0.154 BSC	
C	0.012	—	0.020
C’		0.390 BSC	
D	—	—	0.069
E		0.050 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A		6.00 BSC	
B		3.90 BSC	
C	0.31	—	0.51
C’		9.90 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°

16-pin SSOP (150mil) 外形尺寸



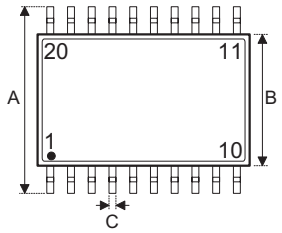


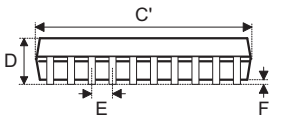


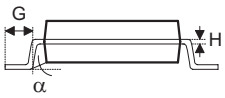
符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A		0.236 BSC	
B		0.154 BSC	
C	0.008	—	0.012
C’		0.193 BSC	
D	—	—	0.069
E		0.025 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A		6.00 BSC	
B		3.90 BSC	
C	0.20	—	0.30
C’		4.90 BSC	
D	—	—	1.75
E		0.635 BSC	
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

20-pin SOP (300mil) 外形尺寸



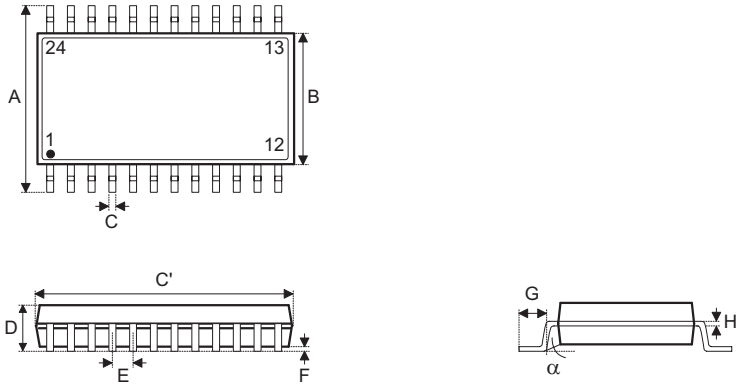




符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A		0.406 BSC	
B		0.295 BSC	
C	0.012	—	0.020
C’		0.504 BSC	
D	—	—	0.104
E		0.050 BSC	
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A		10.30 BSC	
B		7.50 BSC	
C	0.31	—	0.51
C’		12.80 BSC	
D	—	—	2.65
E		1.27 BSC	
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

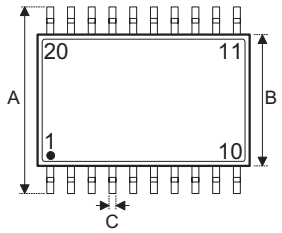
24-pin SOP (300mil) 外形尺寸

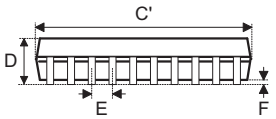


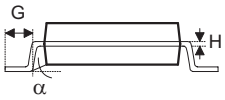
符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A		0.406 BSC	
B		0.295 BSC	
C	0.012	—	0.020
C’		0.606 BSC	
D	—	—	0.104
E		0.050 BSC	
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A		10.30 BSC	
B		7.50 BSC	
C	0.31	—	0.51
C’		15.40 BSC	
D	—	—	2.65
E		1.27 BSC	
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

20-pin SSOP (150mil) 外形尺寸



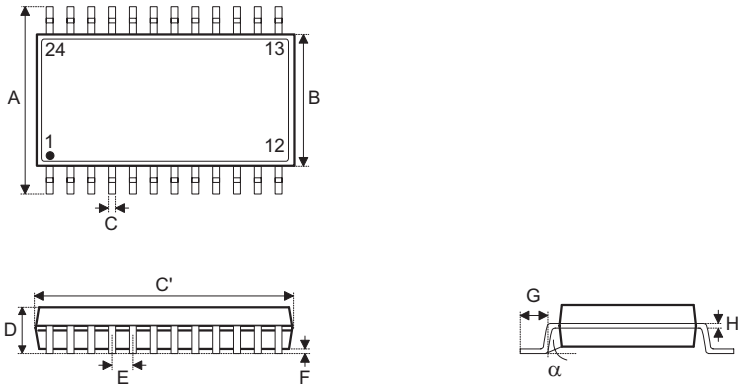




符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A		0.236 BSC	
B		0.154 BSC	
C	0.008	—	0.012
C’		0.341 BSC	
D	—	—	0.069
E		0.025 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A		6.00 BSC	
B		3.90 BSC	
C	0.20	—	0.30
C’		8.66 BSC	
D	—	—	1.75
E		0.635 BSC	
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

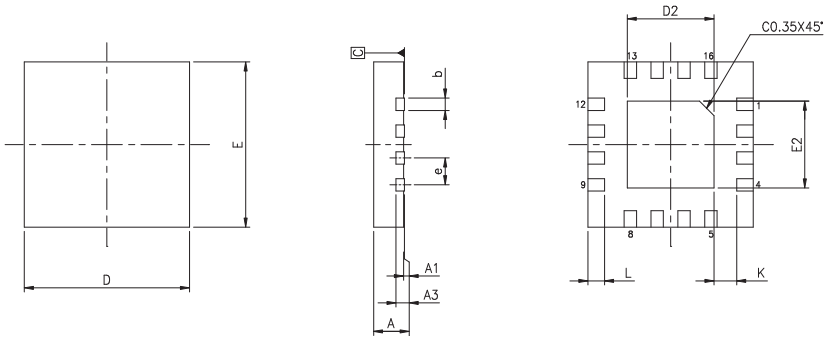
24-pin SSOP (150mil) 外形尺寸



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	0.008	—	0.012
C’		0.341 BSC	
D	—	—	0.069
E		0.025 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		6.00 BSC	
B		3.90 BSC	
C	0.20	—	0.30
C’		8.66 BSC	
D	—	—	1.75
E		0.635 BSC	
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

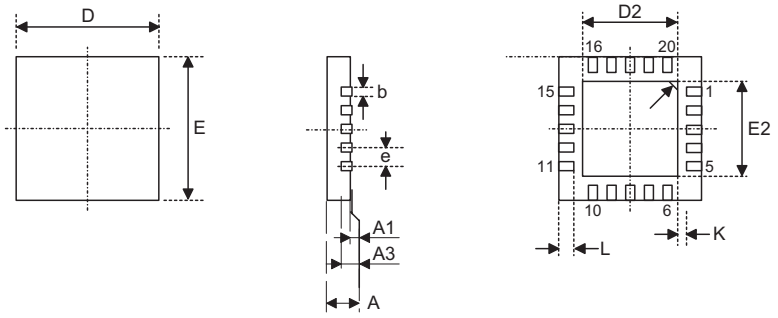
SAW Type 16-pin QFN (4mm×4mm×0.75mm) 外形尺寸



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		0.080 REF	
b	0.010	0.012	0.014
D		0.157 BSC	
E		0.157 BSC	
e		0.026 BSC	
D2	0.079	—	0.106
E2	0.079	—	0.106
L	0.014	0.016	0.018
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		0.203 REF	
b	0.25	0.30	0.35
D		4.00 BSC	
E		4.00 BSC	
e		0.65 BSC	
D2	2.00	—	2.70
E2	2.00	—	2.70
L	0.35	0.40	0.45
K	0.20	—	—

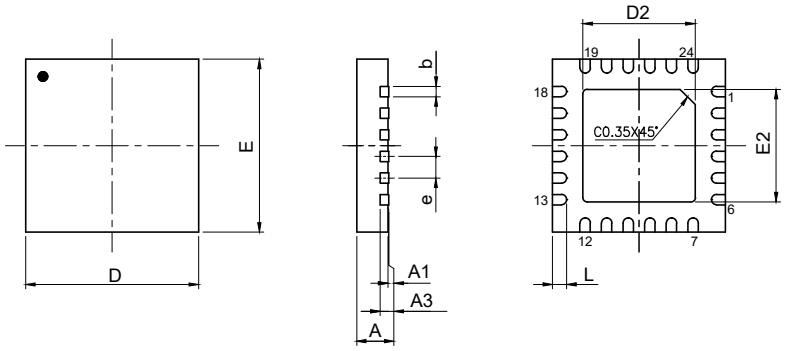
SAW Type 20-pin QFN (4mm×4mm×0.75mm) 外形尺寸



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 REF	
b	0.008	0.010	0.012
D		0.157 BSC	
E		0.157 BSC	
e		0.020 BSC	
D2	0.075	—	0.083
E2	0.075	—	0.083
L	0.012	0.016	0.020
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		0.203 REF	
b	0.20	0.25	0.30
D		4.00 BSC	
E		4.00 BSC	
e		0.50 BSC	
D2	1.90	—	2.10
E2	1.90	—	2.10
L	0.30	0.40	0.50
K	0.20	—	—

SAW Type 24-pin QFN (4mm×4mm, lead: 0.325mm) 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 REF	
b	0.007	0.010	0.012
D		0.157 BSC	
E		0.157 BSC	
e		0.020 BSC	
D2	0.100	—	0.110
E2	0.100	—	0.110
L	0.011	0.013	0.015

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		0.203 REF	
b	0.18	0.25	0.30
D		4.00 BSC	
E		4.00 BSC	
e		0.50 BSC	
D2	2.55	—	2.80
E2	2.55	—	2.80
L	0.28	0.33	0.38