触控按键 Flash 单片机

BS83B24C/BS83C40C

版本 : V1.00　日期 : 2018-02-09

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 8/12/16MHz RC 振荡器 – HIRC
  - ♦ 内部 32kHz RC 振荡器 – LIRC
  - ♦ 外部 32.768kHz 晶振 – LXT
- 完全集成内部振荡器，无需外接元器件
- 多种工作模式：快速、低速、空闲和休眠
- 所有指令都可在 1~3 个指令周期内完成
- 查表指令
- 115 条功能强大的指令系统
- 多达 6 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器：多达 4K×16
- RAM 数据存储器：多达 768×8
- True EEPROM 存储器：128×8
- 多达 40 个触控按键功能 – 完全集成而无需外接元器件
- 看门狗定时器功能
- 多达 42 个双向 I/O 口
- 可编程 I/O 口源电流
- 单个引脚与外部中断口共用
- 多个定时器模块用于时间测量、捕捉输入、比较匹配输出、PWM 输出及单脉冲输出功能
- 双时基功能，可提供固定时间的中断信号
- 通用串行接口模块 – USIM，用于 SPI、I^2C 或 UART 通信
- 低电压复位功能
- 多种封装类型

概述

该系列单片机是 8 位具有高性能精简指令集且完全集成触控按键功能的 Flash 型单片机。该系列单片机含有内部触控按键功能和可多次编程的 Flash 存储器特性，为各种触控按键产品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、时基功能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。

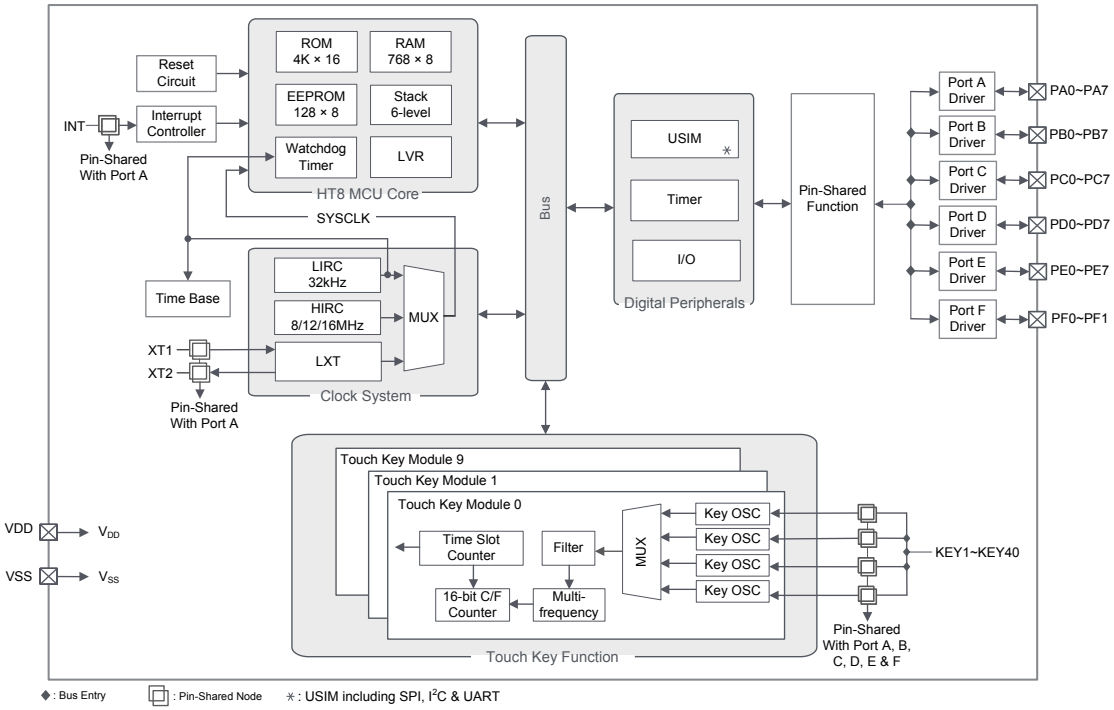
选型表

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。

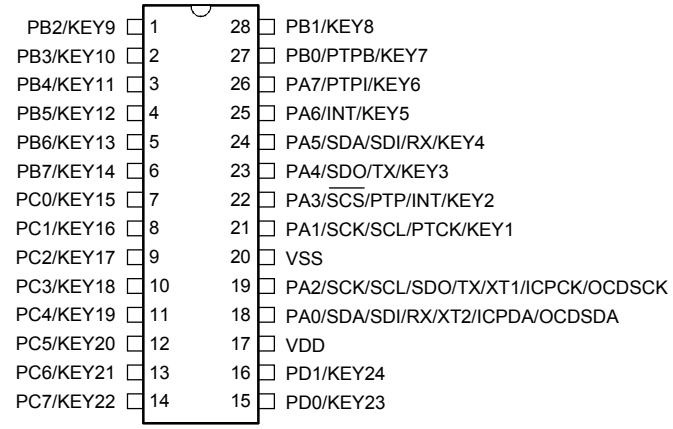
单片机型号	程序存储器	数据存储器	数据 EEPROM	I/O	外部中断
BS83B24C	3K×16	512×8	128×8	26	1
BS83C40C	4K×16	768×8	128×8	42	1

单片机型号	TM 模块	时基	触控按键	USIM	堆栈	封装
BS83B24C	10-bit PTM×1	2	24	√	6	28SOP 28SSOP
BS83C40C	10-bit CTM×1 10-bit PTM×1	2	40	√	6	44LQFP

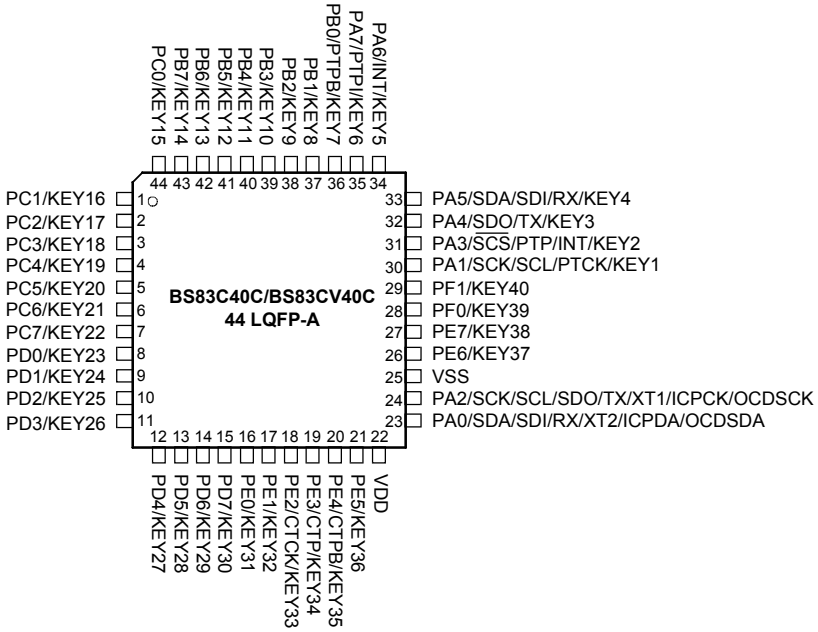
方框图



引脚图



BS83B24C/BS83BV24C
28 SOP-A/SSOP-A



注：1. 若共用脚同时有多种输出，所需引脚共用功能由相应软件控制位确定。
2. OCDSDA 和 OCDSCK 是 OCDS 专用引脚，仅存在于 BS83BV24C/BS83CV40C 中。BS83BV24C/BS83CV40C 分别是 BS83B24C/BS83C40C 的 OCDS EV 芯片。

引脚描述

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。

BS83B24C

引脚名称	功能	OPT	I/T	O/T	描述
PA0/SDA/SDI/RX/XT2/ICPDA/OCDSDA	PA0	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻和唤醒功能。
	SDA	PAS0 IFS	ST	NMOS	I^2C 数据
	SDI	PAS0 IFS	ST	—	SPI 串行数据输入
	RX	PAS0 IFS	ST	—	UART 串行数据输入
	XT2	PAS0	—	LXT	LXT 振荡器引脚
	ICPDA	—	ST	CMOS	ICP 数据 / 地址
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址，仅用于 EV 芯片

引脚名称	功能	OPT	I/T	O/T	描述
PA1/SCK/SCL/PTCK/KEY1	PA1	PAWU PAPU PAS0	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	PAS0 IFS	ST	CMOS	SPI 串行时钟
	SCL	PAS0 IFS	ST	NMOS	I^2C 时钟
	PTCK	PAS0	ST	—	PTM 捕捉输入
	KEY1	PAS0 TKM0C1	NSI	—	触控按键输入 1
PA2/SCK/SCL/SDO/TX/XT1/ICPCK/OCDSCK	PA2	PAWU PAPU PAS0	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	PAS0 IFS	ST	CMOS	SPI 串行时钟
	SCL	PAS0 IFS	ST	NMOS	I^2C 时钟
	SDO	PAS0	—	CMOS	SPI 串行数据输出
	TX	PAS0	—	CMOS	UART 串行数据输出
	XT1	PAS0	LXT	—	LXT 振荡器引脚
	ICPCK	—	ST	—	ICP 时钟输入
	OCDSCK	—	ST	—	OCDS 时钟输入，仅用于 EV 芯片
PA3/$\overline{SCS}$/PTP/INT/KEY2	PA3	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻和唤醒功能。
	$\overline{SCS}$	PAS0	ST	—	SPI 从机选择
	PTP	PAS0	—	CMOS	PTM 输出
	INT	PAS0 INTEG INTC0 IFS	ST	—	外部中断
	KEY2	PAS0 TKM0C1	NSI	—	触控按键输入 2
PA4/SDO/TX/KEY3	PA4	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻和唤醒功能。
	SDO	PAS1	—	CMOS	SPI 串行数据输出
	TX	PAS1	—	CMOS	UART 串行数据输出
	KEY3	PAS1 TKM0C1	NSI	—	触控按键输入 3

引脚名称	功能	OPT	I/T	O/T	描述
PA5/SDA/SDI/RX/KEY4	PA5	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻和唤醒功能。
	SDA	PAS1 IFS	ST	NMOS	I^2C 数据
	SDI	PAS1 IFS	ST	—	SPI 串行数据输入
	RX	PAS1 IFS	ST	—	UART 串行数据输入
	KEY4	PAS1 TKM0C1	NSI	—	触控按键输入 4
PA6/INT/KEY5	PA6	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻和唤醒功能。
	INT	PAS1 INTEG INTC0 IFS	ST	—	外部中断
	KEY5	PAS1 TKM1C1	NSI	—	触控按键输入 5
PA7/PTPI/KEY6	PA7	PAWU PAPU PAS1	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	PAS1	ST	—	PTM 捕捉输入
	KEY6	PAS1 TKM1C1	NSI	—	触控按键输入 6
PB0/PTPB/KEY7	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	PTPB	PBS0	—	CMOS	PTM 反相输出
	KEY7	PBS0 TKM1C1	NSI	—	触控按键输入 7
PB1/KEY8	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY8	PBS0 TKM1C1	NSI	—	触控按键输入 8
PB2/KEY9	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY9	PBS0 TKM2C1	NSI	—	触控按键输入 9
PB3/KEY10	PB3	PBPU PBS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY10	PBS0 TKM2C1	NSI	—	触控按键输入 10

引脚名称	功能	OPT	I/T	O/T	描述
PB4/KEY11	PB4	PBPU PBS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY11	PBS1 TKM2C1	NSI	—	触控按键输入 11
PB5/KEY12	PB5	PBPU PBS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY12	PBS1 TKM2C1	NSI	—	触控按键输入 12
PB6/KEY13	PB6	PBPU PBS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY13	PBS1 TKM3C1	NSI	—	触控按键输入 13
PB7/KEY14	PB7	PBPU PBS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY14	PBS1 TKM3C1	NSI	—	触控按键输入 14
PC0/KEY15	PC0	PCPU PCS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY15	PCS0 TKM3C1	NSI	—	触控按键输入 15
PC1/KEY16	PC1	PCPU PCS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY16	PCS0 TKM3C1	NSI	—	触控按键输入 16
PC2/KEY17	PC2	PCPU PCS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY17	PCS0 TKM4C1	NSI	—	触控按键输入 17
PC3/KEY18	PC3	PCPU PCS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY18	PCS0 TKM4C1	NSI	—	触控按键输入 18
PC4/KEY19	PC4	PCPU PCS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY19	PCS1 TKM4C1	NSI	—	触控按键输入 19
PC5/KEY20	PC5	PCPU PCS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY20	PCS1 TKM4C1	NSI	—	触控按键输入 20
PC6/KEY21	PC6	PCPU PCS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY21	PCS1 TKM5C1	NSI	—	触控按键输入 21

引脚名称	功能	OPT	I/T	O/T	描述
PC7/KEY22	PC7	PCPU PCS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY22	PCS1 TKM5C1	NSI	—	触控按键输入 22
PD0/KEY23	PD0	PDPU PDS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY23	PDS0 TKM5C1	NSI	—	触控按键输入 23
PD1/KEY24	PD1	PDPU PDS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY24	PDS0 TKM5C1	NSI	—	触控按键输入 24
VDD	VDD	—	PWR	—	正电源供电
VSS	VSS	—	PWR	—	负电源供电

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

BS83C40C

引脚名称	功能	OPT	I/T	O/T	描述
PA0/SDA/SDI/ RX/XT2/ICPDA/ OCDSDA	PA0	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻和唤醒功能。
	SDA	PAS0 IFS	ST	NMOS	I^2C 数据
	SDI	PAS0 IFS	ST	—	SPI 串行数据输入
	RX	PAS0 IFS	ST	—	UART 串行数据输入
	XT2	PAS0	—	LXT	LXT 振荡器引脚
	ICPDA	—	ST	CMOS	ICP 数据 / 地址
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址，仅用于 EV 芯片
PA1/SCK/SCL/ PTCK/KEY1	PA1	PAWU PAPU PAS0	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	PAS0 IFS	ST	CMOS	SPI 串行时钟
	SCL	PAS0 IFS	ST	NMOS	I^2C 时钟
	PTCK	PAS0	ST	—	PTM 捕捉输入
	KEY1	PAS0 TKM0C1	NSI	—	触控按键输入 1

引脚名称	功能	OPT	I/T	O/T	描述
PA2/SCK/SCL/SDO/TX/XT1/ICPCK/OCDSCK	PA2	PAWU PAPU PAS0	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	PAS0 IFS	ST	CMOS	SPI 串行时钟
	SCL	PAS0 IFS	ST	NMOS	I^2C 时钟
	SDO	PAS0	—	CMOS	SPI 串行数据输出
	TX	PAS0	—	CMOS	UART 串行数据输出
	XT1	PAS0	LXT	—	LXT 振荡器引脚
	ICPCK	—	ST	—	ICP 时钟输入
	OCDSCK	—	ST	—	OCDS 时钟输入，仅用于 EV 芯片
PA3/$\overline{SCS}$/PTP/INT/KEY2	PA3	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻和唤醒功能。
	$\overline{SCS}$	PAS0	ST	—	SPI 从机选择
	PTP	PAS0	—	CMOS	PTM 输出
	INT	PAS0 INTEG INTC0 IFS	ST	—	外部中断
	KEY2	PAS0 TKM0C1	NSI	—	触控按键输入 2
PA4/SDO/TX/KEY3	PA4	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻和唤醒功能。
	SDO	PAS1	—	CMOS	SPI 串行数据输出
	TX	PAS1	—	CMOS	UART 串行数据输出
	KEY3	PAS1 TKM0C1	NSI	—	触控按键输入 3
PA5/SDA/SDI/RX/KEY4	PA5	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻和唤醒功能。
	SDA	PAS1 IFS	ST	NMOS	I^2C 数据
	SDI	PAS1 IFS	ST	—	SPI 串行数据输入
	RX	PAS1 IFS	ST	—	UART 串行数据输入
	KEY4	PAS1 TKM0C1	NSI	—	触控按键输入 4

引脚名称	功能	OPT	I/T	O/T	描述
PA6/INT/KEY5	PA6	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻和唤醒功能。
	INT	PAS1 INTEG INTC0 IFS	ST	—	外部中断
	KEY5	PAS1 TKM1C1	NSI	—	触控按键输入 5
PA7/PTPI/KEY6	PA7	PAWU PAPU PAS1	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	PAS1	ST	—	PTM 捕捉输入
	KEY6	PAS1 TKM1C1	NSI	—	触控按键输入 6
PB0/PTPB/KEY7	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	PTPB	PBS0	—	CMOS	PTM 反相输出
	KEY7	PBS0 TKM1C1	NSI	—	触控按键输入 7
PB1/KEY8	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY8	PBS0 TKM1C1	NSI	—	触控按键输入 8
PB2/KEY9	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY9	PBS0 TKM2C1	NSI	—	触控按键输入 9
PB3/KEY10	PB3	PBPU PBS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY10	PBS0 TKM2C1	NSI	—	触控按键输入 10
PB4/KEY11	PB4	PBPU PBS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY11	PBS1 TKM2C1	NSI	—	触控按键输入 11
PB5/KEY12	PB5	PBPU PBS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY12	PBS1 TKM2C1	NSI	—	触控按键输入 12
PB6/KEY13	PB6	PBPU PBS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY13	PBS1 TKM3C1	NSI	—	触控按键输入 13

引脚名称	功能	OPT	I/T	O/T	描述
PB7/KEY14	PB7	PBPU PBS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY14	PBS1 TKM3C1	NSI	—	触控按键输入 14
PC0/KEY15	PC0	PCPU PCS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY15	PCS0 TKM3C1	NSI	—	触控按键输入 15
PC1/KEY16	PC1	PCPU PCS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY16	PCS0 TKM3C1	NSI	—	触控按键输入 16
PC2/KEY17	PC2	PCPU PCS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY17	PCS0 TKM4C1	NSI	—	触控按键输入 17
PC3/KEY18	PC3	PCPU PCS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY18	PCS0 TKM4C1	NSI	—	触控按键输入 18
PC4/KEY19	PC4	PCPU PCS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY19	PCS1 TKM4C1	NSI	—	触控按键输入 19
PC5/KEY20	PC5	PCPU PCS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY20	PCS1 TKM4C1	NSI	—	触控按键输入 20
PC6/KEY21	PC6	PCPU PCS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY21	PCS1 TKM5C1	NSI	—	触控按键输入 21
PC7/KEY22	PC7	PCPU PCS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY22	PCS1 TKM5C1	NSI	—	触控按键输入 22
PD0/KEY23	PD0	PDPU PDS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY23	PDS0 TKM5C1	NSI	—	触控按键输入 23
PD1/KEY24	PD1	PDPU PDS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY24	PDS0 TKM5C1	NSI	—	触控按键输入 24

引脚名称	功能	OPT	I/T	O/T	描述
PD2/KEY25	PD2	PDPU PDS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY25	PDS0 TKM6C1	NSI	—	触控按键输入 25
PD3/KEY26	PD3	PDPU PDS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY26	PDS0 TKM6C1	NSI	—	触控按键输入 26
PD4/KEY27	PD4	PDPU PDS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY27	PDS1 TKM6C1	NSI	—	触控按键输入 27
PD5/KEY28	PD5	PDPU PDS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY28	PDS1 TKM6C1	NSI	—	触控按键输入 28
PD6/KEY29	PD6	PDPU PDS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY29	PDS1 TKM7C1	NSI	—	触控按键输入 29
PD7/KEY30	PD7	PDPU PDS1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY30	PDS1 TKM7C1	NSI	—	触控按键输入 30
PE0/KEY31	PE0	PEPU PES0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY31	PES0 TKM7C1	NSI	—	触控按键输入 31
PE1/KEY32	PE1	PEPU PES0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY32	PES0 TKM7C1	NSI	—	触控按键输入 32
PE2/CTCK/KEY33	PE2	PEPU PES0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	CTCK	PES0	ST	—	CTM 时钟输入
	KEY33	PES0 TKM8C1	NSI	—	触控按键输入 33
PE3/CTP/KEY34	PE3	PEPU PES0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	CTP	PES0	—	CMOS	CTM 输出
	KEY34	PES0 TKM8C1	NSI	—	触控按键输入 34

引脚名称	功能	OPT	I/T	O/T	描述
PE4/CTPB/KEY35	PE4	PEPU PES1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	CTPB	PES1	—	CMOS	CTM 反相输出
	KEY35	PES1 TKM8C1	NSI	—	触控按键输入 35
PE5/KEY36	PE5	PEPU PES1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY36	PES1 TKM8C1	NSI	—	触控按键输入 36
PE6/KEY37	PE6	PEPU PES1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY37	PES1 TKM9C1	NSI	—	触控按键输入 37
PE7/KEY38	PE7	PEPU PES1	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY38	PES1 TKM9C1	NSI	—	触控按键输入 38
PF0/KEY39	PF0	PFPU PFS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY39	PFS0 TKM9C1	NSI	—	触控按键输入 39
PF1/KEY40	PF1	PFPU PFS0	ST	CMOS	通用 I/O 口。可通过寄存器设置上拉电阻。
	KEY40	PFS0 TKM9C1	NSI	—	触控按键输入 40
VDD	VDD	—	PWR	—	正电源供电
VSS	VSS	—	PWR	—	负电源供电

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

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

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

直流电气特性

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。

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	V
		f_{SYS}=16MHz	3.3	—	5.5	V
	工作电压 – LXT	f_{SYS}=32768Hz	2.2	—	5.5	V
	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	单位
		V_{DD}	条件				85°C	
I_{STB}	休眠模式	2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3.0	5.0	6.0	
	空闲模式 0 – LIRC	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3.0	5.0	6.0	
		5V		—	5.0	10	12	
	空闲模式 0 – LXT	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3.0	5.0	6.0	
		5V		—	5.0	10	12	
	空闲模式 1 – 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	参数	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	
	低速模式 – LXT	2.2V	f_{SYS}=32768Hz	—	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	
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	
		3.3V	f_{SYS}=16MHz	—	3.0	4.5	
		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. 所有的工作电流值都通过一个连续的 NOP 指令循环程序测得。

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。

内部高速振荡器 – HIRC – 频率精确度

程序烧录时，烧录器会调整 HIRC 振荡器使其工作在用户选择的 HIRC 频率和工作电压 (3V 或 5V) 条件下。

8/12/16MHz

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%	
	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。对于 2.2V~3.6V 的应用电压范围，建议调整电压固定为 3V，而对于 3.3V~5.5V 的应用电压范围，建议调整电压固定为 5V。
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。当烧录器已将 HIRC 调整为某一固定频率，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，除非另有指定

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}	LIRC 启动时间	—	—	—	—	500	μs

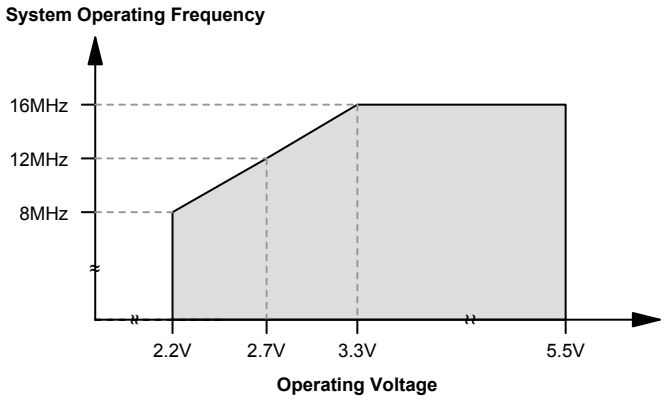
低速晶体振荡器电气特性 – LXT

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{LXT}	振荡器频率	2.2V~5.5V	-40°C~85°C	—	32768	—	Hz
t_{START}	启动时间	3V/5V	—	—	—	500	ms
Duty Cycle	占空比	—	—	45	50	55	%
R_{NEG}	负阻	2.2V	—	3×ESR	—	—	Ω

注：C1、C2 和 R_P 为外部元器件。C1=C2=10pF，R_P=R_U=10MΩ，C_L=7pF，ESR=30kΩ。

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}
		—	$f_{SYS}=f_{SUB}=f_{LXT}$	—	1024	—	t_{LXT}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	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
		—	$f_{SYS}=f_{SUB}=f_{LXT}$ or 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}
		—	f_{LXT} off → on	—	1024	—	t_{LXT}
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/RSTC 软件复位)	—	—				
	系统复位延迟时间 (WDT 溢出)	—	—	14	16	18	ms
t_{SRESET}	软件复位最小脉宽	—	—	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} 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 口低电平输入电压	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	I/O 口高电平输入电压	5V	—	3.5	—	5	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OH}	I/O 口源电流	3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=00B (n=0, 1 ...; m=0, 2, 4, 6)	-0.7	-1.5	—	mA
		5V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=00B (n=0, 1 ...; m=0, 2, 4, 6)	-1.5	-2.9	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=01B (n=0, 1 ...; m=0, 2, 4, 6)	-1.3	-2.5	—	
		5V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=01B (n=0, 1 ...; m=0, 2, 4, 6)	-2.5	-5.1	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=10B (n=0, 1 ...; m=0, 2, 4, 6)	-1.8	-3.6	—	
		5V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=10B (n=0, 1 ...; m=0, 2, 4, 6)	-3.6	-7.3	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=11B (n=0, 1 ...; m=0, 2, 4, 6)	-4	-8	—	
		5V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=11B (n=0, 1 ...; m=0, 2, 4, 6)	-8	-16	—	
I_{OL}	I/O 口灌电流	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
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_{TPI}	TM 捕捉输入最小脉宽	—	—	0.3	—	—	μs
t_{TCK}	TM 时钟输入最小脉宽	—	—	0.3	—	—	μs
t_{INT}	中断引脚最小输入脉宽	—	—	10	—	—	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，在此基础上测量上拉电阻上的分压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
Flash 程序存储器 / 数据 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	
I_{DDPGM}	V_{DD} 电压下烧录 / 擦除电流	—	—	—	—	5.0	mA
E_P	电容耐久性	—	—	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	—	单片机处于休眠模式	1.0	—	—	V

LVR 电气特性

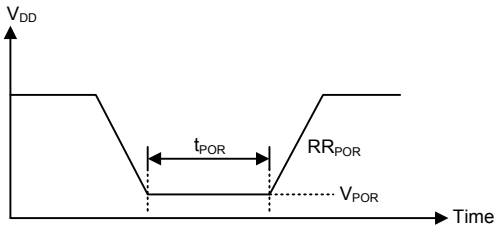
Ta=-40°C~85°C，除非另有指定

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{LVR}	低压复位电压	—	LVR 使能，电压选择 2.10V	-5%	2.10	+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 3.80V		3.80		
I_{LVRBG}	工作电流	3V	LVR 使能，VBGEN=0	—	—	18	μA
		5V		—	20	25	
		3V	LVR 使能，VBGEN=1	—	—	150	
		5V		—	180	200	
t_{LVR}	低电压复位最小低电压脉宽	—	—	120	240	480	μs
I_{LVR}	使能 LVR 增加的电流	—	VBGEN=0	—	—	24	μA

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



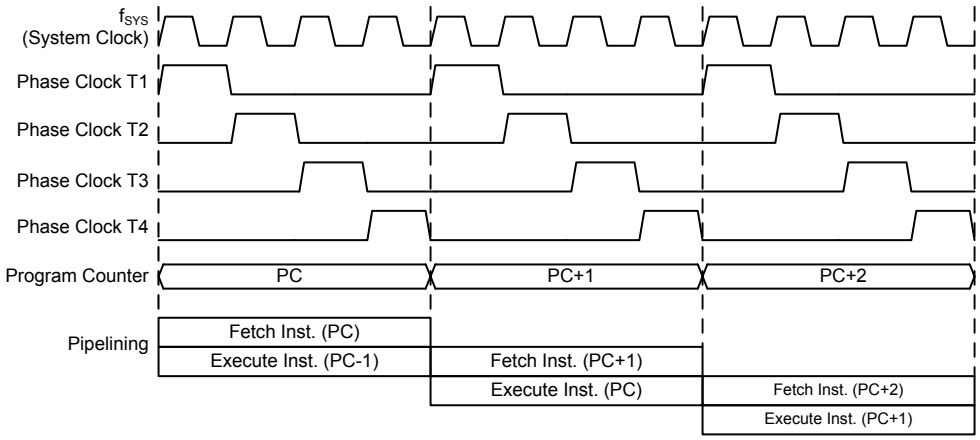
系统结构

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需要多一个指令周期外，大部分的标准指令或扩展指令分别能在一个指令周期或两个指令周期内完成。8-bit 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，确保了在提供具有最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。这些使得此系列单片机适用于低成本和批量生产的控制应用。

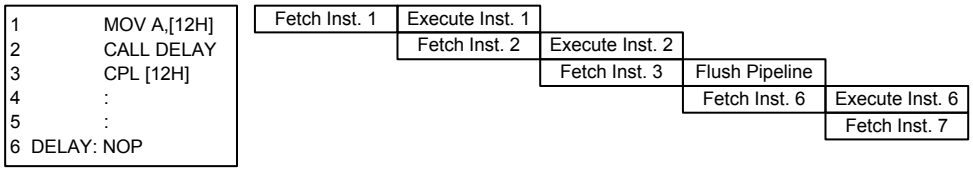
时序和流水线结构

主系统时钟由 LIRC、LXT 或 HIRC 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



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位址到程序寄存器来控制程序，对于条件跳转指令，一旦条件符合，在当前指令执行时取得的下一条指令将会被舍弃，而由一个空指令周期来取代。

单片机型号	程序计数器	
	高字节	低字节寄存器
BS83B24C	PC11~PC8	PCL7~PCL0
BS83C40C	PC11~PC8	PCL7~PCL0

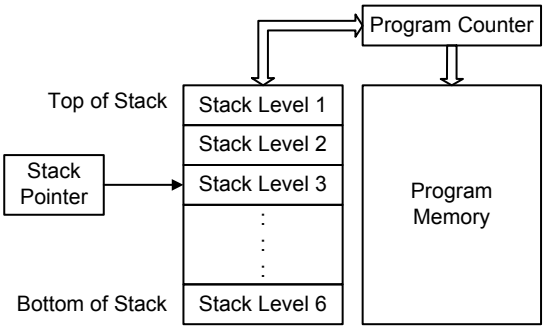
程序计数器

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。注意，当这样一个程序跳转要执行时，会插入一个空指令周期。PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

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。堆栈有 6 层，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，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以前的值。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，
  LADD，LADDM，LADC，LADCM，LSUB，LSUBM，LSBC，LSBCM，
  LDAA
- 逻辑运算：
  AND，OR，XOR，ANDM，ORM，XORM，CPL，CPLA，
  LAND，LOR，LXOR，LANDM，LORM，LXORM，LCPL，LCPLA
- 移位运算：
  RRA，RR，RRCA，RRC，RLA，RL，RLCA，RLC，
  LRRA，LRR，LRRCA，LRRC，LRLA，LRL，LRLCA，LRLC
- 递增和递减：
  INCA，INC，DECA，DEC，
  LINCA，LINC，LDECA，LDEC
- 分支判断：
  JMP，SZ，SZA，SNZ，SIZ，SDZ，SIZA，SDZA，CALL，RET，RETI，
  LSZ，LSZA，LSNZ，LSIZ，LSIZA，LSDZ，LSDZA

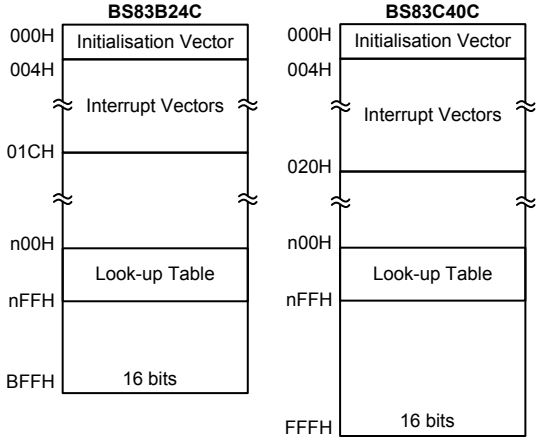
Flash 程序存储器

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。

单片机型号	容量
BS83B24C	3K×16
BS83C40C	4K×16

结构

程序存储器的容量为 3K×16~4K×16，程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口。数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。



程序存储器结构

特殊向量

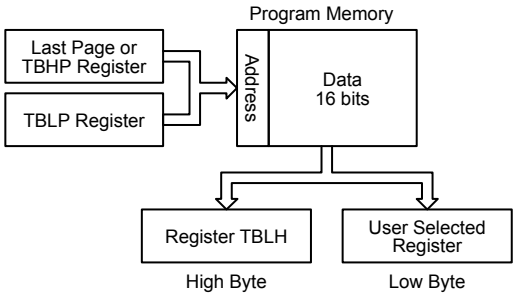
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

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，当数据存储器 [m] 位于 Sector 0，表格数据可以使用如“TABRD[m]”或“TABRDL[m]”等指令分别从程序存储器查表读取。如果存储器 [m] 位于其它 Sector，表格数据可以使用如“LTABRD[m]”或“LTABRDL[m]”等指令分别从程序存储器查表读取。当这些指令执行时，程序存储器中表格数据低字节，将被传送到使用者所指定的数据存储器 [m]，程序存储器中表格数据的高字节，则被传送到 TBLH 特殊寄存器。

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“F00H”指向的地址是 BS83C40C 单片机 4K 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“F06H”，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]”或“LTABRD[m]”指令被使用，则表格指针指向 TBLP 和 TBHP 寄存器所指定的当前页的第一个地址。在这个例子中，表格数据的高字节等于零，而当“TABRD[m]”或“LTABRD[m]”指令被执行时，此值将会自动的被传送到 TBLH 寄存器。

TBLH 寄存器为可读 / 写寄存器，且能重新储存，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
mov a,0Fh           ; initialise high table pointer
mov tbhp,a
:
tabrd tempreg1      ; transfers value in table referenced by table pointer
                    ; data at program memory address "F06H" transferred to
                    ; tempreg1 and TBLH
dec tblp            ; reduce value of table pointer by one
tabrd tempreg2      ; transfers value in table referenced by table pointer
                    ; data at program memory address "F05H" transferred to
                    ; tempreg2 and TBLH in this example the data "1AH" is
                    ; transferred to tempreg1 and data "0FH" to register
                    ; tempreg2
:
org F00h            ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

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。

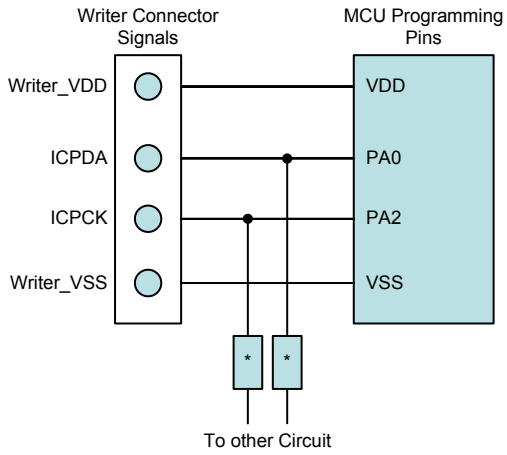
另外，Holtek 单片机提供 4 线接口的在线烧录方式。用户可将进行过烧录或未经过烧录的单片机芯片连同电路板一起制成，最后阶段进行程序的更新和程序的烧写，在无需去除或重新插入芯片的情况下方便地保持程序为最新版。

Flash 型单片机与烧录器引脚对应表如下：

烧录器引脚	MCU 在线烧录引脚	引脚描述
ICPDA	PA0	烧录串行数据 / 地址
ICPCK	PA2	烧录时钟
VDD	VDD	电源
VSS	VSS	地

单片机内部程序存储器和 EEPROM 存储器都可以通过 4 线的接口在线进行烧录。其中一条线用于数据串行下载或上传，一条用于串行时钟，剩余两条用于提供电源。芯片在线烧写的详细使用说明超出此文档的描述范围，将由专门的参考文献提供。

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



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

片上调试 – OCDS

EV 芯片 BS83BV24C 和 BS83CV40C 分别用于 BS83B24C 和 BS83C40C 单片机仿真。此 EV 芯片提供片上调试功能 (OCDS – On-Chip Debug Support) 用于开发过程中的单片机调试。除了片上调试功能方面，EV 芯片和实际单片机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 芯片对实际单片机的仿真。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，请参考相关文件。

e-Link 引脚	EV 芯片引脚	引脚描述
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
GND	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。

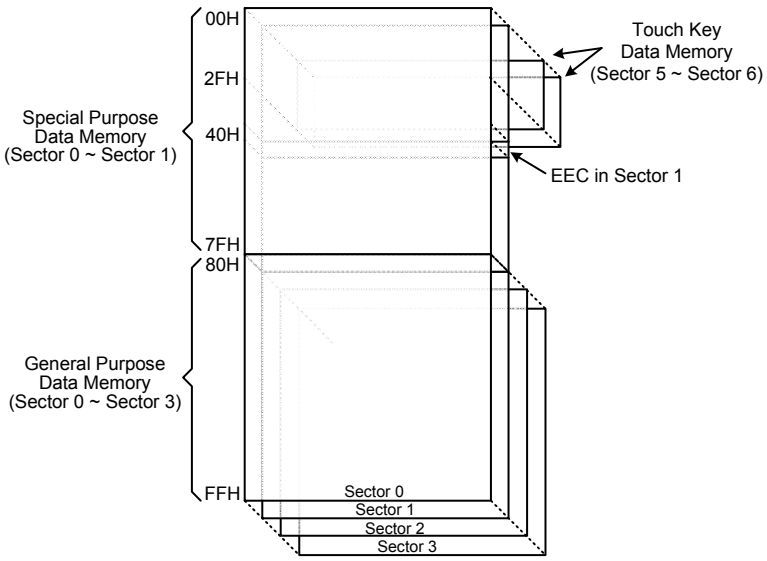
结构

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为通用数据存储器，做一般用途使用，可在程序控制下进行读取和写入。该系列单片机还具有触控按键数据存储器。

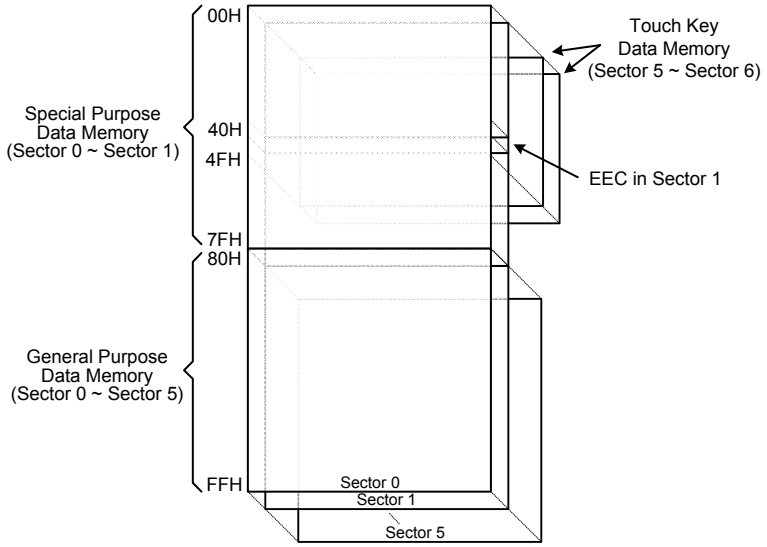
数据存储器被分为若干个 Sector，都可在 8-bit 的存储器中实现。每个数据存储器 sector 分为两种类型，即特殊功能数据存储器和通用数据存储器。特殊功能数据寄存器地址范围为 00H~7FH 而通用数据存储器地址范围为 80H~FFH。触控按键数据存储器位于 Sector 5 和 Sector 6。若使用间接寻址，切换不同的数据存储器 Sector 可通过设置正确的存储器指针值实现。所有单片机的数据存储器起始地址皆为 00H。

单片机型号	特殊功能数据存储器	通用数据存储器		触控按键数据存储器	
	可用 Sector	容量	Sector：地址	容量	Sector：地址
BS83B24C	0, 1	512×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH	96×8	5: 00H~2FH 6: 00H~2FH
BS83C40C	0, 1	768×8	0: 80H~FFH 1: 80H~FFH : 5: 80H~FFH	160×8	5: 00H~4FH 6: 00H~4FH

数据存储器概要



数据存储器结构 – **BS83B24C**



数据存储器结构 – **BS83C40C**

数据存储器寻址

此系列单片机支持扩展指令架构，它并没有可用于数据存储器的存储区指针。对于数据存储器所需的 Sector 是通过 MP1H 或 MP2H 寄存器指定，而所选 Sector 的某一数据存储器地址使用间接寻址访问方式时，通过 MP1L 或 MP2L 寄存器指定。

直接寻址可用于所有 Sector，通过相应的指令可以寻址所有可用的数据存储器空间。所访问的数据存储器位于除 Sector 0 外的任何数据存储器 Sector，扩展指令可代替间接寻址方式用来访问数据存储器。标准指令和扩展指令的主要区别在于扩展指令中的数据存储器地址“m”由多达 11 个有效位组成，高字节表示 Sector，低字节表示指定的地址。

通用数据存储器

所有的单片机程序需要一个读 / 写的存储区，让临时数据可以被储存和再使用，该 RAM 区域就是通用数据存储器。使用者可对这个数据存储区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，极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”。

	Sector 0	Sector 1
00H	IAR0	PAS0
01H	MP0	PAS1
02H	IAR1	PBS0
03H	MP1L	PBS1
04H	MP1H	PCS0
05H	ACC	PCS1
06H	PCL	PDS0
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH		
0CH	IAR2	TKTMR
0DH	MP2L	TKC0
0EH	MP2H	TK16DL
0FH	RSTFC	TK16DH
10H	INTEG	TKC1
11H	INTC0	TKM016DL
12H	INTC1	TKM016DH
13H		TKM0ROL
14H	PA	TKM0ROH
15H	PAC	TKM0C0
16H	PAPU	TKM0C1
17H	PAWU	TKM0C2
18H	LVRC	TKM116DL
19H	IFS	TKM116DH
1AH	WDTC	TKM1ROL
1BH	TB0C	TKM1ROH
1CH	TB1C	TKM1C0
1DH	PSCR	TKM1C1
1EH	MFI0	TKM1C2
1FH		TKM216DL
20H	PB	TKM216DH
21H	PBC	TKM2ROL
22H	PBPU	TKM2ROH
23H		TKM2C0
24H		TKM2C1
25H		TKM2C2
26H		TKM316DL
27H		TKM316DH
28H	PC	TKM3ROL
29H	PCC	TKM3ROH
2AH	PCPU	TKM3C0
2BH	SCC	TKM3C1
2CH	HIRCC	TKM3C2
2DH	LXTC	TKM416DL
2EH		TKM416DH
2FH	RSTC	TKM4ROL
30H	VBGC	TKM4ROH
31H		TKM4C0
32H		TKM4C1
33H		TKM4C2
34H		TKM516DL
35H	PD	TKM516DH
36H	PDC	TKM5ROL
37H	PDPU	TKM5ROH
38H	SLEDC0	TKM5C0
39H	SLEDC1	TKM5C1
3AH		TKM5C2
3BH		
3CH		
3DH		
3EH		
3FH		

	Sector 0	Sector 1
40H		EEC
41H	EEA	
42H	EED	
43H		
44H	PTMC0	
45H	PTMC1	
46H	PTMDL	
47H	PTMDH	
48H	PTMAL	
49H	PTMAH	
4AH	PTMRPL	
4BH	PTMRPH	
4CH		
4DH		
4EH		
4FH		
50H		
51H		
52H	SIMC0	
53H	SIMC1/UUCR1	
54H	SIMC2/SIMA/UUCR2	
55H	SIMD/UTXR_RXR	
56H	SIMTOC/UBRG	
57H	UUSR	
58H		
59H		
5AH		
5BH		
5CH		
5DH		
5EH		
5FH		
60H		
61H		
62H		
63H		
64H		
65H		
66H		
67H		
68H		
69H		
6AH		
6BH		
6CH		
6DH		
6EH		
6FH		
70H		
71H		
72H		
73H		
74H		
75H		
76H		
77H		
78H		
79H		
7AH		
7BH		
7CH		
7DH		
7EH		
7FH		

: Unused, read as 00H

特殊功能数据存储器 – **BS83B24C**

	Sector 0	Sector 1
00H	IAR0	PAS0
01H	MP0	PAS1
02H	IAR1	PBS0
03H	MP1L	PBS1
04H	MP1H	PCS0
05H	ACC	PCS1
06H	PCL	PDS0
07H	TBLP	PDS1
08H	TBLH	PES0
09H	TBHP	PES1
0AH	STATUS	PFS0
0BH		
0CH	IAR2	TKTMR
0DH	MP2L	TKC0
0EH	MP2H	TK16DL
0FH	RSTFC	TK16DH
10H	INTEG	TKC1
11H	INTC0	TKM016DL
12H	INTC1	TKM016DH
13H	INTC2	TKM0ROL
14H	PA	TKM0ROH
15H	PAC	TKM0C0
16H	PAPU	TKM0C1
17H	PAWU	TKM0C2
18H	LVRC	TKM116DL
19H	IFS	TKM116DH
1AH	WDTC	TKM1ROL
1BH	TB0C	TKM1ROH
1CH	TB1C	TKM1C0
1DH	PSCR	TKM1C1
1EH	MFI0	TKM1C2
1FH	MFI1	TKM216DL
20H	PB	TKM216DH
21H	PBC	TKM2ROL
22H	PBPU	TKM2ROH
23H		TKM2C0
24H		TKM2C1
25H		TKM2C2
26H		TKM316DL
27H		TKM316DH
28H	PC	TKM3ROL
29H	PCC	TKM3ROH
2AH	PCPU	TKM3C0
2BH	SCC	TKM3C1
2CH	HIRCC	TKM3C2
2DH	LXTC	TKM416DL
2EH		TKM416DH
2FH	RSTC	TKM4ROL
30H	VBGC	TKM4ROH
31H		TKM4C0
32H		TKM4C1
33H		TKM4C2
34H		TKM516DL
35H	PD	TKM516DH
36H	PDC	TKM5ROL
37H	PDPU	TKM5ROH
38H	SLEDC0	TKM5C0
39H	SLEDC1	TKM5C1
3AH	SLEDC2	TKM5C2
3BH	CTMC0	
3CH	CTMC1	
3DH	CTMDL	
3EH	CTMDH	
3FH	CTMAL	

	Sector 0	Sector 1
40H	CTMAH	EEC
41H	EEA	TKM616DL
42H	EED	TKM616DH
43H		TKM6ROL
44H	PTMC0	TKM6ROH
45H	PTMC1	TKM6C0
46H	PTMDL	TKM6C1
47H	PTMDH	TKM6C2
48H	PTMAL	TKM716DL
49H	PTMAH	TKM716DH
4AH	PTMRPL	TKM7ROL
4BH	PTMRPH	TKM7ROH
4CH	PE	TKM7C0
4DH	PEC	TKM7C1
4EH	PEPU	TKM7C2
4FH	PF	TKM816DL
50H	PFC	TKM816DL
51H	PFPU	TKM8ROL
52H	SIMC0	TKM8ROH
53H	SIMC1/UUCR1	TKM8C0
54H	SIMC2/SIMA/UUCR2	TKM8C1
55H	SIMD/UTXR_RXR	TKM8C2
56H	SIMTOC/UBRG	TKM916DL
57H	UUSR	TKM916DH
58H		TKM9ROL
59H		TKM9ROH
5AH		TKM9C0
5BH		TKM9C1
5CH		TKM9C2
5DH		
5EH		
5FH		
60H		
61H		
62H		
63H		
64H		
65H		
66H		
67H		
68H		
69H		
6AH		
6BH		
6CH		
6DH		
6EH		
6FH		
70H		
71H		
72H		
73H		
74H		
75H		
76H		
77H		
78H		
79H		
7AH		
7BH		
7CH		
7DH		
7EH		
7FH		

: Unused, read as 00H

特殊功能数据存储器 – **BS83C40C**

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, IAR2

间接寻址寄存器 IAR0、IAR1 和 IAR2 的地址虽位于数据存储区，但其并没有实际的物理地址。间接寻址的方法使用这些间接寻址寄存器和存储器指针做数据操作，以取代定义实际存储器地址的直接存储器寻址方法。在间接寻址寄存器 IAR0、IAR1 和 IAR2 上的任何动作，将对存储器指针 MP0、MP1L/MP1H 或是 MP2L/MP2H 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 Sector 0，而 IAR1 和 MP1L/MP1H、IAR2 和 MP2L/MP2H 可以访问任何 Sector。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, MP1L/MP1H, MP2L/MP2H

该系列单片机提供五个存储器指针，即 MP0、MP1L、MP1H、MP2L 和 MP2H。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。当对相关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 MP0 所指定的地址，此时间接寻址寄存器 IAR0 用于访问 Sector0 中的数据，而 MP1L/MP1H 和 IAR1、MP2L/MP2H 和 IAR2 可根据 MP1H 或 MP2H 寄存器访问所有的 Sector。直接寻址通过相关的数据存储器寻址指令来访问所有可用数据存储空间。

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

间接寻址程序范例

范例 1

```
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:
```

范例 2

```
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,01h              ; setup the memory sector
    mov mp1h,a
    mov a,offset adres1    ; Accumulator loaded with first RAM address
    mov mp1l,a             ; setup memory pointer with first RAM address
loop:
    clr IAR1               ; clear the data at address defined by MP1L
    inc mp1l               ; increment memory pointer MP1L
    sdz block              ; check if last memory location has been cleared
    jmp loop
continue:
    :
```

需注意，范例中并没有确定 RAM 地址。

使用扩展指令直接寻址程序举例

```
data .section 'data'
temp db ?
code .section at 0 'code'
org 00h
start:
    lmov a,[m]             ; move [m] data to acc
    lsub a, [m+1]          ; compare [m] and [m+1] data
    snz  c                 ; [m]>[m+1]?
    jmp  continue          ; no
    lmov a,[m]             ; yes, exchange [m] and [m+1] data
    mov  temp,a
    lmov a,[m+1]
    lmov [m],a
    mov  a,temp
    lmov m+1],a
continue:
    :
```

注：“m”是位于任何数据存储器 Sector 的某一地址。例如，m=1F0H 表示 Sector 1 中的地址 0F0H。

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-bit 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

该 8-bit 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)、SC 标志位、CZ 标志位、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 TO 和 PDF 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、C、SC 和 CZ 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。
- 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。
- SC：当 OV 与当前指令操作结果 MSB 执行“XOR”所得结果。
- CZ：不同指令不同标志位的操作结果。详细资料请参考寄存器定义部分。

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	SC	CZ	TO	PDF	OV	Z	AC	C
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
POR	x	x	0	0	x	x	x	x

"x"：未知

Bit 7 **SC**：当 OV 与当前指令操作结果 MSB 执行"XOR"所得结果

Bit 6 **CZ**：不同指令不同标志位的操作结果。
对于 SUB/SUBM/LSUB/LSUBM 指令，CZ 等于 Z 标志位。
对于 SBC/SBCM/LSBC/LSBCM 指令，CZ 等于上一个 CZ 标志位与当前零标志位执行"AND"所得结果。对于其它指令，CZ 标志位无影响。

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 也受循环移位指令的影响。

EEPROM 数据存储器

此系列每个单片机都内建 EEPROM 数据存储器，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。

EEPROM 数据存储器结构

该系列单片机的 EEPROM 数据存储器容量为 128×8。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 Sector 0 中的一个地址和数据寄存器以及 Sector 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 位于 Sector 0 中，它们能像其它特殊功能寄存器一样直接被访问。EEC 位于 Sector 1 中，不能被直接访问，仅能通过 MP1L/MP1H 和 IAR1 或 MP2L/MP2H 和 IAR2 进行间接读取或写入。由于 EEC 控制寄存器位于 Sector 1 中的“40H”，在 EEC 寄存器上的任何操作被执行前，MP1L 或 MP2L 必须先设为“40H”，MP1H 或 MP2H 被设为“01H”。

寄存器名称	位							
	7	6	5	4	3	2	1	0
EEA	—	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
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	—	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
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 6~0　　**EEA6~EEA0**：数据 EEPROM 地址 bit 6~bit 0

- **EED 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
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　　**EED7~EED0**：数据 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 未首先置高时，此位置高无效。

注：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”。

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，写入的数据需存入 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 EEC 寄存器中的 WR 位需立即置高以开始写操作，这两条指令必须连续执行。总中断位 EMI 在写周期开始前应当被清零，写周期开始后再将其使能。应注意若 WR 位已置为高而 WREN 位还未被设置则不能开始写操作。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 写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。上电后存储器指针高字节寄存器 MP1H 或 MP2H 将重置为“0”，这意味着数据存储区 Sector 0 被选中。由于 EEPROM 控制寄存器位于 Sector 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。更多细节详见中断章节。

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。存储器指针高字节寄存器也可以正常清零以阻止进入 EEPROM 控制寄存器存在的 Sector 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。WREN 位置位后，EEC 寄存器中的 WR 位需立即置位，以确保写周期正确地执行。写周期执行前总中断位 EMI 应先清零，写周期开始执行后再将此位重新使能。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。

程序范例

从 EEPROM 中读取数据 – 轮询法

```
MOV  A, EEPROM_ADRES        ; user defined address
MOV  EEA, A
MOV  A, 40H                 ; setup memory pointer low byte MP1L
MOV  MP1L, A                ; MP1L points to EEC register
MOV  A, 01H                 ; setup Memory Pointer high byte MP1H
MOV  MP1H, 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 write/read
CLR  MP1H
MOV  A, EED                 ; move read data to register
MOV  READ_DATA, A
```

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 low byte MP1L
MOV  MP1L, A                ; MP1L points to EEC register
MOV  A, 01H                 ; setup Memory Pointer high byte MP1H
MOV  MP1H, A
```

```
CLR  EMI
SET  IAR1.3                ; set WREN bit, enable write operations
SET  IAR1.2                ; start Write Cycle - set WR bit
SET  EMI
BACK:
SZ   IAR1.2                ; check for write cycle end
JMP  BACK
CLR  IAR1                  ; disable EEPROM write
CLR  MP1H
```

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最优化。振荡器选择是通过相关寄存器完成的。

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、时基功能和定时器模块的时钟源。外部振荡器需要一些外围器件，而集成的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。振荡器类型通过寄存器选择。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

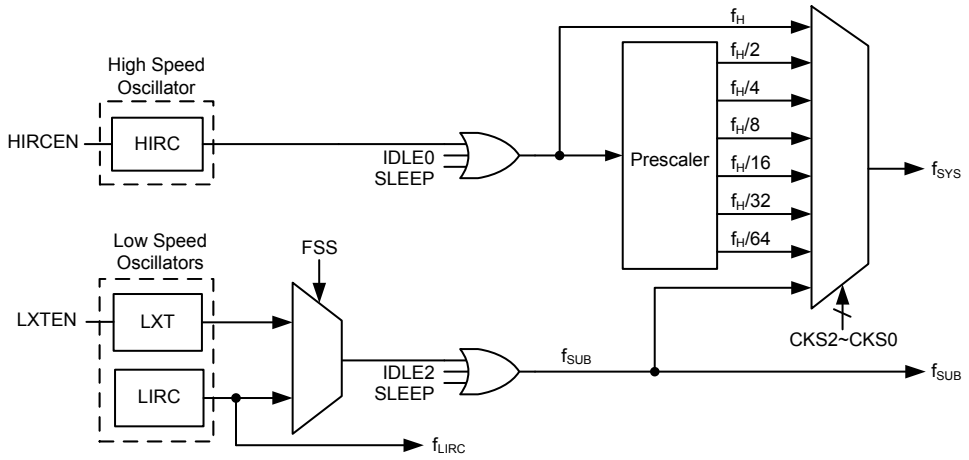
类型	名称	频率	引脚
内部高速 RC	HIRC	8/12/16MHz	—
内部低速 RC	LIRC	32kHz	—
外部低速晶振	LXT	32.768kHz	XT1/XT2

振荡器类型

系统时钟配置

此系列单片机有三个系统振荡器，包括一个高速振荡器和两个低速振荡器。高速振荡器有内部 8/12/16MHz RC 振荡器 HIRC，低速振荡器有外部 32.768kHz 晶振 LXT 和内部 32kHz 低速振荡器 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 SCC 寄存器中的 CKS2~CKS0 位决定的，系统时钟可动态选择。

低速振荡器的实际时钟源由寄存器选择，低速或高速系统时钟频率由 SCC 寄存器中的 CKS2~CKS0 位决定。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 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，不需其它外部器件。内部 RC 振荡器具有三种固定的频率：8MHz、12MHz 或 16MHz，可通过配置选项选择。此外 HIRCC 寄存器中的 HIRC1~HIRC0 位设置的频率必须与配置选项中选定的频率一致，以确保能够达到交流电气特性中标示的 HIRC 频率精度。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减至最低程度。当电源为 3V 或 5V 且温度为 25°C 时，所选中的调整后的振荡频率误差将在 1% 内。

外部 32.768kHz 晶体振荡器 – LXT

外部 32.768kHz 晶体振荡器是一个低频振荡器，经由软件控制为 FSS 选择。时钟频率固定为 32.768kHz，此时 XT1 和 XT2 间引脚必须连接 32.768kHz 的晶体振荡器，需要外部电阻和电容连接到 32.768kHz 晶振以帮助起振。对于那些要求精确频率的场合中，可能需要这些元件来对由制程产生的误差提供频率补偿。在系统上电期间，LXT 振荡器启动需要一定的延时。

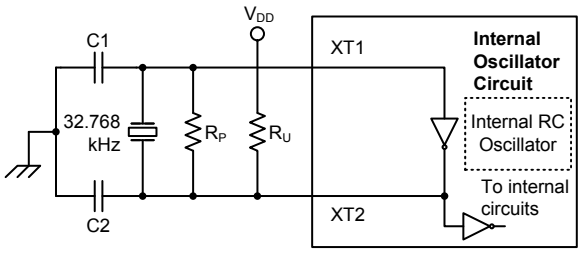
当系统进入空闲 / 休眠模式，系统时钟关闭以降低功耗。然而在某些应用，比如空闲 / 休眠模式下要保持内部定时器功能，必须提供额外的时钟，且与系统时钟无关。

然而，对于一些晶体，为了保证系统频率的启动与精度要求，需要外接两个小容量电容 C1 和 C2，具体数值与客户选择的晶体规格有关。外部并联的反馈电阻 R_P 和上拉电阻 R_U 是必需的。

引脚共用软件控制位决定 XT1/XT2 脚是用于 LXT 还是作为普通 I/O 口或者其它共用功能使用。

- 若 LXT 振荡器未被用于任何时钟源，XT1/XT2 脚能被用作一般 I/O 口或者其它共用功能使用。
- 若 LXT 振荡器被用于一些时钟源，32.768kHz 晶体应被连接至 XT1/XT2 脚。

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



Note: 1. R_P, R_U, C1 and C2 are required.
2. Although not shown XT1/XT2 pins have a parasitic capacitance of around 7pF.

外部 **LXT** 振荡器

LXT 振荡器 C1 和 C2 值		
晶体频率	**C1**	**C2**
32.768kHz	10pF	10pF
注：1、C1 和 C2 数值仅作参考用。 2、R_P 的建议值为 5MΩ~10MΩ。 3、R_U 的建议值为 10MΩ。		

32.768kHz 晶体电容推荐值

内部 32kHz 振荡器 – LIRC

内部 32kHz 系统振荡器是一个低频振荡器，通过软件控制位 FSS 选择。这种单片机有一个完全集成 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减至最低。在电源为 5V 且温度为 25°C 时，32kHz 的固定振荡频率误差将在 10% 内。

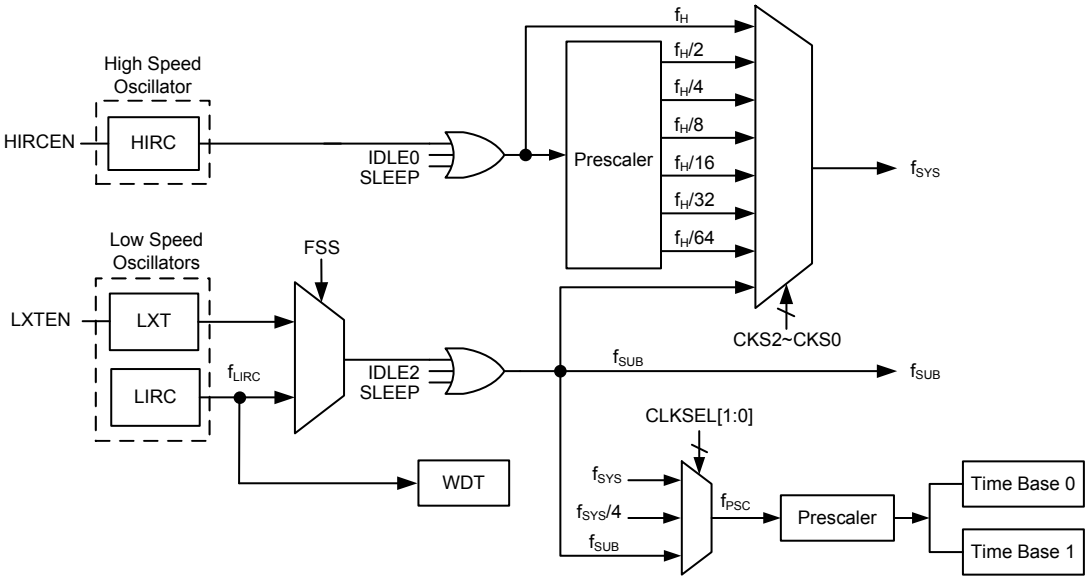
工作模式和系统时钟

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

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 SCC 寄存器中的 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，低频时钟可来自 LIRC 或 LXT 振荡器，由 SCC 寄存器中的 FSS 位选择。其它系统时钟还有高速系统振荡器的分频 $f_H/2$~$f_H/64$。



系统时钟配置

注：当系统时钟源 f_{SYS} 由 f_H 到 f_{SUB} 转换时，可以通过设置相应的高速振荡器使能控制位，选择停止以节省耗电，或者继续振荡，为外围电路提供 f_H~$f_H/64$ 的频率。

系统工作模式

单片机有 6 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 4 种工作模式：休眠模式、空闲模式 0、空闲模式 1 和空闲模式 2 用于单片机 CPU 关闭时以节省耗电。

工作模式	CPU	寄存器设置			f_{SYS}	f_H	f_{SUB}	f_{LIRC}
		FHIDEN	FSIDEN	CKS2~CKS0				
快速模式	On	x	x	000~110	f_H~f_H/64	On	On	On
低速模式	On	x	x	111	f_{SUB}	On/Off (1)	On	On
空闲模式 0	Off	0	1	000~110	Off	Off	On	On
				111	On			
空闲模式 1	Off	1	1	xxx	On	On	On	On
空闲模式 2	Off	1	0	000~110	On	On	Off	On
				111	Off			
休眠模式	Off	0	0	xxx	Off	Off	Off	On (2)

“x”：无关

注：1. 在低速模式中，f_H 开启或关闭由相应的振荡器使能位控制。
2. 由于 WDT 功能始终使能，f_{LIRC} 始终开启。

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高速振荡器提供。该模式下单片机正常工作的时钟源来自高速振荡器。高速振荡器频率可被分为 1~64 的不等比率，实际的比率由 SCC 寄存器中的 CKS2~CKS0 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。该低速时钟源来自 f_{SUB}。f_{SUB} 时钟可来源于 LIRC 或 LXT 振荡器，通过 SCC 寄存器中的软件控制位 FSS 选择。

休眠模式

在 HALT 指令执行后且 FHIDEN 和 FSIDEN 位为低时，系统进入休眠模式。在休眠模式中，CPU 停止运行，f_{SUB} 停止为外围功能提供时钟。由于看门狗定时器功能始终使能，f_{LIRC} 时钟将继续运行。

空闲模式 0

执行 HALT 指令后且 SCC 寄存器中 FHIDEN 位为低、FSIDEN 为高时，系统进入空闲模式 0。在空闲模式 0 中，系统振荡器停止，CPU 停止，但低速振荡器会开启以驱动一些外围功能。

空闲模式 1

执行 HALT 指令后且 SCC 寄存器中 FHIDEN 和 FSIDEN 位都为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，但高速和低速振荡器都会开启以确保一些外围功能继续工作。

空闲模式 2

执行 HALT 指令后且 SCC 寄存器中 FHIDEN 为高、FSIDEN 位都为低时，系统进入空闲模式 2。在空闲模式 2 中，CPU 停止，但高速振荡器会开启以确保一些外围功能继续工作。

控制寄存器

寄存器 SCC、HIRCC 和 LXTC 用于控制系统时钟和相应的振荡器配置。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SCC	CKS2	CKS1	CKS0	—	—	FSS	FHIDEN	FSIDEN
HIRCC	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
LXTC	—	—	—	—	—	—	LXTF	LXTEN

系统工作模式控制寄存器列表

- **SCC 寄存器**

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	CKS2	CKS1	CKS0	—	—	FSS	FHIDEN	FSIDEN
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7~5 **CKS2~CKS0**：系统时钟选择位
000：f_H
001：$f_H/2$
010：$f_H/4$
011：$f_H/8$
100：$f_H/16$
101：$f_H/32$
110：$f_H/64$
111：f_{SUB}
这三位用于选择系统时钟源。除了 f_H 或 f_{SUB} 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 4~3 未定义，读为“0”

Bit 2 **FSS**：低频时钟选择位
0：LIRC
1：LXT

Bit 1 **FHIDEN**：CPU 关闭时高频振荡器控制位
0：除能
1：使能
此位用来控制在 CPU 执行 HALT 指令关闭后高速振荡器是被激活还是停止。

Bit 0 **FSIDEN**：CPU 关闭时低频振荡器控制位
0：除能
1：使能
此位用来控制在 CPU 执行 HALT 指令关闭后低速振荡器是被激活还是停止。

- **HIRCC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	1

Bit 7~4　未定义，读为“0”

Bit 3~2　**HIRC1~HIRC0**：HIRC 频率选择位
00：8MHz
01：12MHz
10：16MHz
11：8MHz
当 HIRC 振荡器使能或通过应用程序改变 HIRC 频率选择位时，在 HIRCF 标志位置高后时钟频率会自动改变。
建议这里选择的频率与配置选项中选定的频率保持一致，以确保能够达到交流电气特性中标示的 HIRC 频率精度。

Bit 1　**HIRCF**：HIRC 振荡器稳定标志位
0：HIRC 未稳定
1：HIRC 稳定
此位用于表明 HIRC 振荡器是否稳定。HIRCEN 位置高使能 HIRC 振荡器，HIRCF 位会先被清零，在 HIRC 稳定后会被置高。

Bit 0　**HIRCEN**：HIRC 振荡器使能控制位
0：除能
1：使能

- **LXTC 寄存器**

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

Bit 7~2　未定义，读为“0”

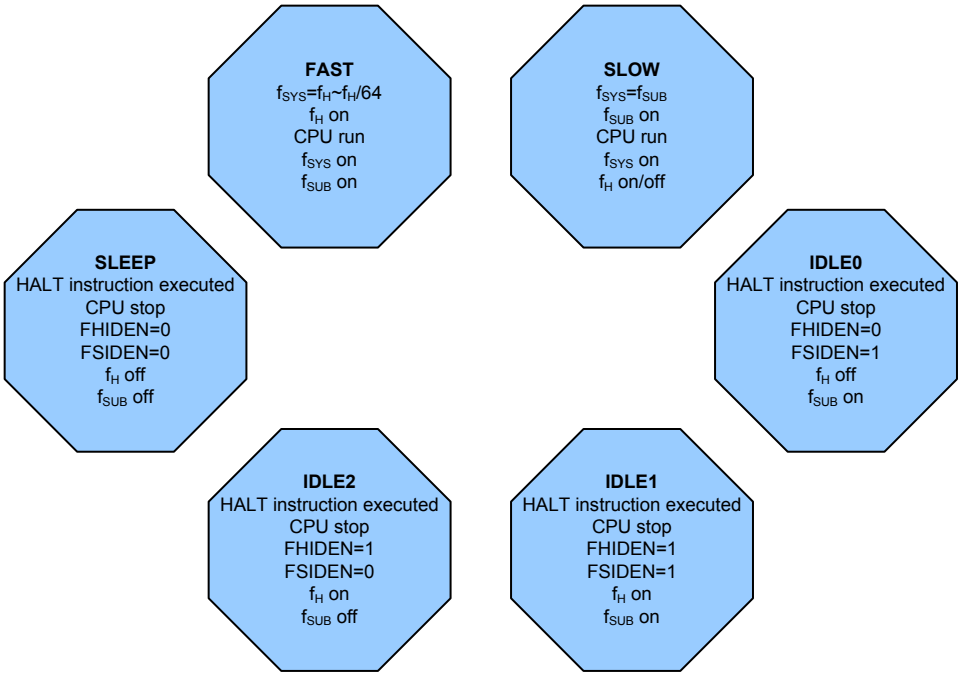
Bit 1　**LXTF**：LXT 振荡器稳定标志位
0：LXT 未稳定
1：LXT 稳定
此位用于表明 LXT 振荡器是否稳定。LXTEN 位置高使能 LXT 振荡器后，LXTF 位会先被清零，在 LXT 稳定后会被置高。

Bit 0　**LXTEN**：LXT 振荡器使能控制位
0：除能
1：使能

工作模式转换

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

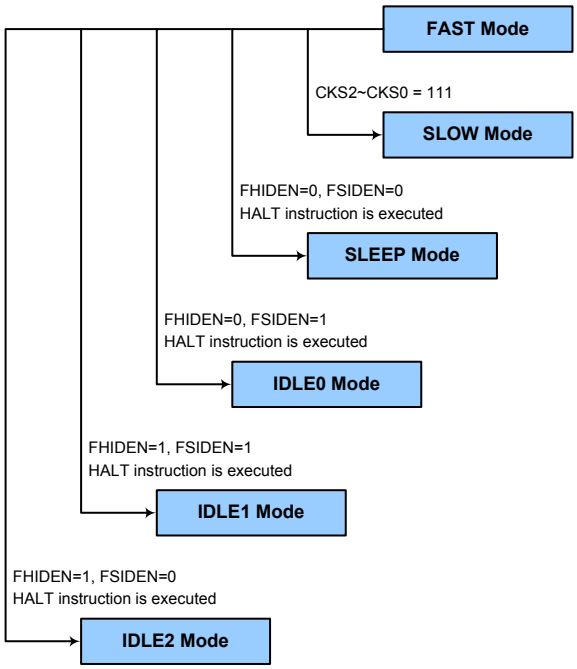
简单来说，快速模式和低速模式间的切换仅需设置 SCC 中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



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 SCC 寄存器中的 CKS2~CKS0 位为“111”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。

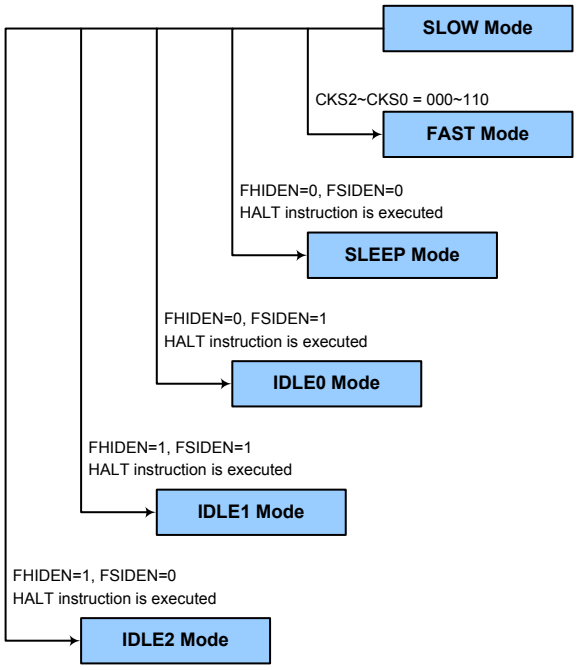
低速模式的时钟源来自 LXT 或 LIRC 振荡器，因此要求这些振荡器在所有模式切换动作发生前稳定下来。



低速模式切换到快速模式

在低速模式时系统时钟来自 f_{SUB}。切换回快速模式时，需设置 CKS2~CKS0 位为“000”~“110”使系统时钟从 f_{SUB} 切换到 f_H~f_H/64。

然而，如果在低速模式下 f_H 因未使用而关闭，那么从低速模式切换到快速模式时，它需要一定的时间来重新起振和稳定，可通过检测 HIRCC 寄存器中的 HIRCF 位进行判断，所需的高速系统振荡器稳定时间在相关电气特性中有说明。



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 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为“0”。在这种模式下，除了 WDT 以外的所有时钟和功能都将关闭。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。
- 数据存储器中的内容和寄存器将保持当前值。
- 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。
- 由于 WDT 功能始终使能，WDT 将被清零并重新开始计数。

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 SCC 寄存器中的 FHIDEN 位为“0”且 FSIDEN 位为“1”。在上述条件下执行该指令后，将发生的情况如下：

- f_H 时钟停止运行，应用程序停止在“HALT”指令处，但 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。
- 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。
- 由于 WDT 功能始终使能，WDT 将被清零并重新开始计数。

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 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为“1”。在上述条件下执行该指令后，将发生的情况如下：

- f_H 和 f_{SUB} 时钟开启，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 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。
- 由于 WDT 功能始终使能，WDT 将被清零并重新开始计数。

进入空闲模式 2

进入空闲模式 2 的方法仅有一种 —— 应用程序中执行“HALT”指令前需设置 SCC 寄存器中的 FHIDEN 位为“1”且 FSIDEN 位为“0”。在上述条件下执行该指令后，将发生的情况如下：

- f_H 时钟开启，f_{SUB} 时钟关闭，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。
- 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。
- 由于 WDT 功能始终使能，WDT 将被清零并重新开始计数。

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 和空闲模式 2 除外)，所以如果要将电路的电流降到最低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入 / 输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。

另外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。还需注意的是 LXT 或 LIRC 振荡器的使能也会消耗额外的待机电流。

在空闲模式 1 和空闲模式 2 中高速振荡器开启，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。

看门狗定时器

看门狗定时器的功能在于防止如电磁的干扰等外部不可控制事件，所造成的程序不正常动作或跳转到未知的地址。

看门狗定时器时钟源

WDT 定时器时钟源来自于内部时钟 f_{LIRC}。内部振荡器 LIRC 的频率大约为 32kHz，其内部时钟周期随 V_{DD}、温度和制成的不同而变化。看门狗定时器的时钟源可分频为 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 寄存器用于控制 WDT 功能的使能，单片机复位及溢出周期选择。

- **WDTC 寄存器**

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

Bit 7~3 **WE4~WE0**：WDT 软件控制位
01010 或 10101：使能
其它值：MCU 复位
如果由于不利的环境因素使这些位变为其它值，单片机将复位。复位动作发生在一段延迟时间 t_{SRESET} 后，且 RSTFC 寄存器的 WRF 位将置为“1”。

Bit 2~0 **WS2~WS0**：WDT 溢出周期选择位
000：$2^8/f_{LIRC}$
001：$2^{10}/f_{LIRC}$
010：$2^{12}/f_{LIRC}$
011：$2^{14}/f_{LIRC}$
100：$2^{15}/f_{LIRC}$
101：$2^{16}/f_{LIRC}$
110：$2^{17}/f_{LIRC}$
111：$2^{18}/f_{LIRC}$
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。

- **RSTFC 寄存器**

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

“x”：未知

Bit 7~4　未定义，读为“0”

Bit 3　**RSTF**：复位控制寄存器软件复位标志位
详见其它章节

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
当 WDT 控制寄存器软件复位发生时，此位被置为“1”，且只能通过应用程序清零。

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”。

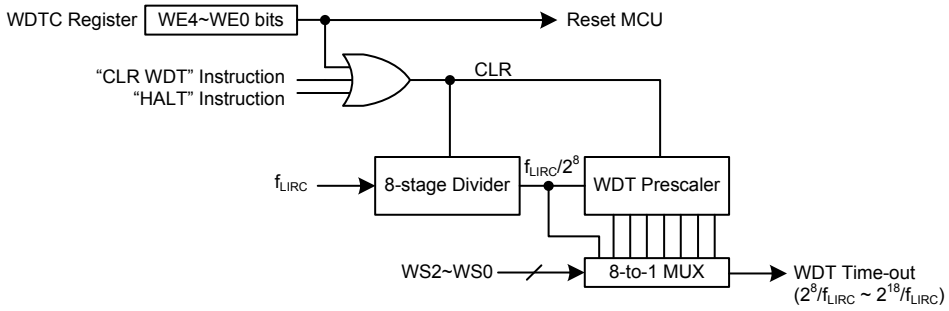
WE4~WE0	WDT 功能
10101B 或 01010B	使能
其它值	复位 MCU

看门狗定时器功能控制

程序正常运行时，WDT 溢出将导致芯片复位，并置位状态标志位 TO。若系统处于休眠或空闲模式，当 WDT 发生溢出时，状态寄存器中的 TO 标志位会被置位，且只有程序计数器 PC 和堆栈指针 SP 会被复位。有三种方法可以用来清除 WDT 的内容。第一种是 WDT 软件复位，即将 WE4~WE0 位设置成除了“01010B”和“10101B”外的任意值；第二种是通过看门狗定时器软件清除指令，而第三种是通过“HALT”指令。

该系列单片机只使用一条看门狗指令“CLR WDT”。因此只要执行“CLR WDT”便能清除 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，使得单片机可以设置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置为预先设置的状态。程序计数器就是其中之一，它会被清除为零，使得单片机从最低的程序存储器地址开始执行程序。

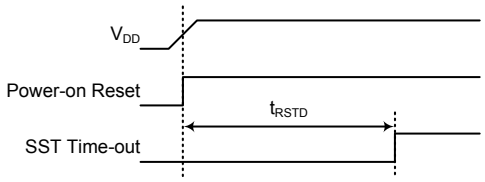
另一种复位为看门狗溢出单片机复位。此外还有一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设置值时，系统会产生 LVR 复位。不同方式的复位操作会对寄存器产生不同的影响。

复位功能

通过内部事件触发复位，单片机共有以下几种复位方式：

上电复位

这是最基本且不可避免的复位，发生在单片机上电后。除了保证程序存储器从开始地址执行，上电复位也使得其它寄存器被设置在预设条件。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置为输入状态。



上电复位时序图

内部复位控制

内部复位控制寄存器 RSTC 用于在单片机操作因环境噪声干扰不能正常运行时复位。如果 RSTC 寄存器设置为 01010101B 或 10101010B 外的其它任意值，则经过延迟时间 t_{SRESET} 后单片机复位。上电后这几位的值为 01010101B。

RSTC7~RSTC0	复位功能
01010101B	无操作
10101010B	无操作
其它任意值	复位 MCU

内部复位功能控制

- **RSTC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	RSTC7	RSTC6	RSTC5	RSTC4	RSTC3	RSTC2	RSTC1	RSTC0
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 **RSTC7~RSTC0**：复位功能控制位
01010101：无操作
10101010：无操作
其它：MCU 复位
如果由于不利的环境因素使这些位发生改变，单片机将复位。复位动作发生在一段延迟时间 t_{SRESET} 后，且 RSTFC 寄存器的 RSTF 位将置为“1”。

- **RSTFC 寄存器**

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

“x”：未知

Bit 7~4 未定义，读为“0”

Bit 3 **RSTF**：复位控制寄存器软件复位标志位
0：未发生
1：发生
当 RSTC 控制寄存器软件复位发生时，此位被置为“1”。注意此位只能通过应用程序清零。

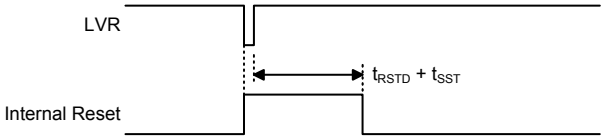
Bit 2 **LVRF**：LVR 复位标志位
详见其它章节

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
详见其它章节

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。LVR 始终使能，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在 0.9V~V_{LVR} 之间，这时 LVR 将会自动复位单片机且 RSTFC 寄存器中的 LVRF 标志位置位。LVR 包含以下的规格：有效的 LVR 信号，即在 0.9V~V_{LVR} 的低电压状态的时间，必须超过 LVR 电气特性中 t_{LVR} 参数的值。如果低电压存在不超过 t_{LVR} 参数的值，则 LVR 将会忽略它且不会执行复位功能。V_{LVR} 参数值通过 LVRC 寄存器中的 LVS 位段选择。若 LVS7~LVS0 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 RSTFC 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。



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。此时复位后的寄存器内容保持不变。
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。

- **RSTFC 寄存器**

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

“x”：未知

Bit 7~4 未定义，读为“0”

Bit 3 **RSTF**：复位控制寄存器软件复位标志位
详见其它章节

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 控制寄存器软件复位标志位
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

- **VBGC 寄存器**

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

“x”：未知

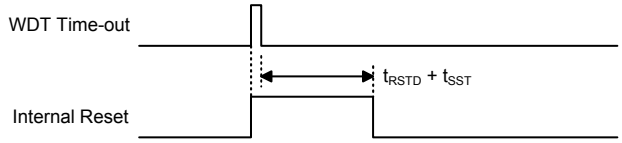
Bit 7~4　未定义，读为“0”

Bit 3　**VBGEN**：Bandgap 电压输出使能控制
0：除能
1：使能
应注意在 LVR 功能使能或当 VBGEN 置 1 的时候，Bandgap 电路使能。

Bit 2~0　未定义，读为“0”

正常运行时的看门狗溢出复位

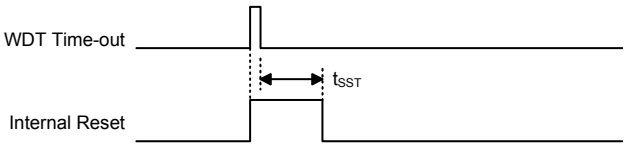
除了看门狗溢出标志位 TO 将被设为“1”之外，快速模式和低速模式下的看门狗溢出复位和 LVR 复位相同。



正常运行时看门狗溢出时序图

休眠或空闲时的看门狗溢出复位

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“0”及 TO 位被设为“1”外，绝大部分的条件保持不变。图中 t_{SST} 的详细说明请参考系统上电时间电气特性。



休眠或空闲时的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	快速模式或低速模式时的 LVR 复位
1	u	快速模式或低速模式时的 WDT 溢出复位
1	1	空闲或休眠模式时的 WDT 溢出复位

“u”：未改变

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
WDT，时基	复位后清零，WDT 开始计数
定时模块	所有定时模块停止
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状态是非常重要的。下表即为不同方式复位后内部寄存器的状况。

寄存器名称	BS83B24C	BS83C40C	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	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
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 uuuu	uuuu uuuu	uuuu uuuu
TBHP	●	●	---- xxxx	----uuuu	---- uuuu	---- uuuu
STATUS	●	●	xx00 xxxx	uuuu uuuu	uu1u uuuu	uu11 uuuu
IAR2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	●	●	---- 0x00	---- u1uu	---- uuuu	---- uuuu
INTEG	●	●	---- --00	---- --00	---- --00	---- --uu
INTC0	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2		●	---0 ---0	---0 ---0	---0 ---0	---u ---u
PA	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
LVRC	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
IFS	●	●	---- -000	---- -000	---- -000	---- -uuu
WDTC	●	●	0101 0011	0101 0011	0101 0011	uuuu uuuu
TB0C	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
TB1C	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu

寄存器名称	BS83B24C	BS83C40C	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)
PSCR	●	●	---- --00	---- --00	---- --00	---- --uu
MFI0	●	●	--00 --00	--00 --00	--00 --00	--uu --uu
MFI1		●	--00 --00	--00 --00	--00 --00	--uu --uu
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
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
SCC	●	●	000- -000	000- -000	000- -000	uuu- -uuu
HIRCC	●	●	---- 0001	---- 0001	---- 0001	---- uuuu
LXTC	●	●	---- --00	---- --00	---- --00	---- --uu
RSTC	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
VBGC	●	●	---- 0---	---- 0---	---- 0---	---- u---
PD	●		---- --11	---- --11	---- --11	---- --uu
		●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	●		---- --11	---- --11	---- --11	---- --uu
		●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDPU	●		---- --00	---- --00	---- --00	---- --uu
		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC1	●		--00 0000	--00 0000	--00 0000	--uu uuuu
		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC2		●	--00 0000	--00 0000	--00 0000	--uu uuuu
CTMC0		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMC1		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMDL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMDH		●	---- --00	---- --00	---- --00	---- --uu
CTMAL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMAH		●	---- --00	---- --00	---- --00	---- --uu
EEA	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
EED	●	●	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

寄存器名称	BS83B24C	BS83C40C	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)
PTMRPH	●	●	---- --00	---- --00	---- --00	---- --uu
PE		●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC		●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEPU		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PF		●	---- --11	---- --11	---- --11	---- --uu
PFC		●	---- --11	---- --11	---- --11	---- --uu
PFPU		●	---- --00	---- --00	---- --00	---- --uu
SIMC0	●	●	1110 0000	1110 0000	1110 0000	uuuu uuuu
UUCR1* (UMD=1)	●	●	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
SIMC1 (UMD=0)	●	●	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMA/SIMC2/UUCR2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMD/UTXR_RXR	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
UBRG*	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMTOC	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
UUSR	●	●	0000 1011	0000 1011	0000 1011	uuuu uuuu
PAS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS0	●		---- 0000	---- 0000	---- 0000	---- uuuu
		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS1		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PES0		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PES1		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PFS0		●	---- 0000	---- 0000	---- 0000	---- uuuu
TKTMR	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC0	●	●	0000 0-00	0000 0-00	0000 0-00	uuuu u-uu
TK16DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK16DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC1	●	●	0000 0011	0000 0011	0000 0011	uuuu uuuu
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	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM0C1	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM0C2	●	●	1110 0100	1110 0100	1110 0100	uuuu uuuu

寄存器名称	BS83B24C	BS83C40C	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)
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	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM1C1	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM1C2	●	●	1110 0100	1110 0100	1110 0100	uuuu 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	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM2C1	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM2C2	●	●	1110 0100	1110 0100	1110 0100	uuuu uuuu
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	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM3C1	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM3C2	●	●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM416DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM416DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM4ROL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM4ROH	●	●	---- --00	---- --00	---- --00	---- --uu
TKM4C0	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM4C1	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM4C2	●	●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM516DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM516DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM5ROL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM5ROH	●	●	---- --00	---- --00	---- --00	---- --uu
TKM5C0	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM5C1	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM5C2	●	●	1110 0100	1110 0100	1110 0100	uuuu uuuu
EEC	●	●	---- 0000	---- 0000	---- 0000	---- uuuu
TKM616DL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM616DH		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM6ROL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu

寄存器名称	BS83B24C	BS83C40C	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)
TKM6ROH		●	---- --00	---- --00	---- --00	---- --uu
TKM6C0		●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM6C1		●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM6C2		●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM716DL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM716DH		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM7ROL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM7ROH		●	---- --00	---- --00	---- --00	---- --uu
TKM7C0		●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM7C1		●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM816DL		●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM7C2		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM816DH		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM8ROL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM8ROH		●	---- --00	---- --00	---- --00	---- --uu
TKM8C0		●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM8C1		●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM8C2		●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM916DL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM916DH		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM9ROL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM9ROH		●	---- --00	---- --00	---- --00	---- --uu
TKM9C0		●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM9C1		●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM9C2		●	1110 0100	1110 0100	1110 0100	uuuu uuuu

注：“u”表示未改变
“x”表示未知
“-”表示未定义
“*”：UUCR1 和 SIMC1 寄存器共用同一个存储器地址，UBRG 和 SIMTOC 寄存器共用同一个存储器地址。复位发生后，通过应用程序手动设置 UMD 位为“1”后可获得 UUCR1 和 UBRG 寄存器的默认值。

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，这些特性也使得此类单片机在广泛应用上都能符合开发的需求。

此系列单片机提供了双向输入 / 输出口。这些寄存器在数据存储器有特定的地址，如特殊功能数据存储器表所示。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	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC5	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU4	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	—	—	—	—	—	—	PD1	PD0
PDC	—	—	—	—	—	—	PDC1	PDC0
PDPU	—	—	—	—	—	—	PDPU1	PDPU0

“—”：未定义

I/O 逻辑功能寄存器列表 – BS83B24C

寄存器名称	位							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC5	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU4	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PE	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEC	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PF	—	—	—	—	—	—	PF1	PF0
PFC	—	—	—	—	—	—	PFC1	PFC0
PFPU	—	—	—	—	—	—	PFPU1	PFPU0

“—”：未定义

I/O 逻辑功能寄存器列表 – BS83C40C

上拉电阻

许多产品应用在端口处于输入状态时需要外加一个上拉电阻来实现上拉的功能。为了免去外部上拉电阻，当引脚规划为输入时，可由内部连接到一个上拉电阻。这些上拉电阻可通过相关上拉控制寄存器来设置，它用一个 PMOS 晶体管来实现上拉电阻功能。

需要注意的是，当 I/O 引脚设为输入或 NMOS 输出时，上拉功能才会受相关上拉控制寄存器控制开启，其它状态下上拉功能不可用。

- **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：I/O 端口 x 引脚上拉功能控制

0：除能

1：使能

PxPUn 用于控制引脚上拉功能。这里的 x 可以是 A、B、C、D、E 或 F，这取决于选中的单片机型号。但是，每个 I/O 端口的实际有效位可能不同。

PA 口唤醒

当使用“HALT”指令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。

需要注意的是，只有当引脚功能为通用 I/O 功能且单片机处于空闲 / 休眠模式时，唤醒功能才会受 PAWU 控制开启，其它状态下此唤醒功能不可用。

- **PAWU 寄存器**

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

Bit 7~0 **PAWU7~PAWU0**：PA 端口引脚唤醒功能控制
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，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	PxC6	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：I/O 端口 x 引脚类型选择
0：输出
1：输入
PxCn 用于控制引脚类型选择。这里的 x 可以是 A、B、C、D、E 或 F，这取决于选中的单片机型号。但是，每个 I/O 端口的实际有效位可能不同。

输入 / 输出端口源电流选择

该系列单片机的每个 I/O 口都支持不同的源电流驱动能力。通过配置相应的选择寄存器 SLEDCn，特定的 I/O 端口可支持 4 个 Level 的源电流驱动能力。用户可参考输入 / 输出口电气特性章节为不同应用选择所需的源电流。

- **SLEDC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
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 **SLEDC07~SLEDC06**：PB7~PB4 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

Bit 5~4 **SLEDC05~SLEDC04**：PB3~PB0 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

Bit 3~2 **SLEDC03~SLEDC02**：PA7~PA4 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

Bit 1~0 **SLEDC01~SLEDC00**：PA3~PA0 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

- **SLEDC1 寄存器 – BS83B24C**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
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 5~4 **SLEDC15~SLEDC14**：PD1~PD0 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

Bit 3~2 **SLEDC13~SLEDC12**：PC7~PC4 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

Bit 1~0 **SLEDC11~SLEDC10**：PC3~PC0 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

- **SLEDC1 寄存器 – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
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 **SLEDC17~SLEDC16**：PD7~PD4 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

Bit 5~4 **SLEDC15~SLEDC14**：PD3~PD0 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

Bit 3~2 **SLEDC13~SLEDC12**：PC7~PC4 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

Bit 1~0 **SLEDC11~SLEDC10**：PC3~PC0 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

- **SLEDC2 寄存器 – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SLEDC25	SLEDC24	SLEDC23	SLEDC22	SLEDC21	SLEDC20
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 5~4 **SLEDC25~SLEDC24**：PF1~PF0 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

Bit 3~2 **SLEDC23~SLEDC22**：PE7~PE4 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

Bit 1~0 **SLEDC21~SLEDC20**：PE3~PE0 源电流选择
00：Level 0 (最小)
01：Level 1
10：Level 2
11：Level 3 (最大)

引脚共用功能

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。此外，这些引脚功能可以通过应用程序控制一系列寄存器进行设定。

引脚共用功能选择寄存器

封装中有限的引脚个数会对某些单片机功能造成影响。然而，引脚功能共用和引脚功能选择，使得小封装单片机具有更多不同的功能。单片机包含端口“x”输出功能选择寄存器“n”，记为 PxSn，和输入功能选择寄存器记为 IFS，这些寄存器可以用来选择共用引脚的特定功能。

当选择引脚共用输入功能，对应的输入和输出功能要进行合理设定。例如，I^2C SDA 引脚被使用，对应的引脚共用功能应通过配置 PASn 寄存器设置为 SDA/SDI/RX 功能，同时通过 IFS 寄存器选择 SDA 信号输入引脚。但是，如果选择

外部中断功能，相关的输出引脚共用功能应选择作为 I/O 功能引脚，且也要选择中断输入信号。

要注意的最重要一点是，确保所需的引脚共用功能被正确地选择和取消。对于大多数的引脚共用功能，要选择所需的引脚共用功能，首先应通过相应的引脚共用控制寄存器正确地选择该功能，然后再配置相应的外围功能设置以使能外围功能。但是，在设置相关引脚控制字段时，一些数字输入引脚如 INT、xTCK 等，与对应的通用 I/O 口共用同一个引脚共用设置选项。要选择这些引脚功能，除了上述的必要的引脚共用控制和外围功能设置外，还必须将其对应的端口控制寄存器位设置为输入。要正确地取消引脚共用功能，首先应除能外围功能，然后再修改相应的引脚共用控制寄存器以选择其它的共用功能。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	—	—	—	—	PDS03	PDS02	PDS01	PDS00
IFS	—	—	—	—	—	IFS2	IFS1	IFS0

引脚共用功能选择寄存器 – BS83B24C

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
PDS1	PDS17	PDS16	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
PES0	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
PES1	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
PFS0	—	—	—	—	PFS03	PFS02	PFS01	PFS00
IFS	—	—	—	—	—	IFS2	IFS1	IFS0

引脚共用功能选择寄存器 – BS83C40C

- **PAS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	PAS07	PAS06	PAS03	PAS02	PAS01	PAS00
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 **PAS07~PAS06**：PA3 引脚共用功能选择
00：PA3/$\overline{INT}$
01：$\overline{SCS}$
10：PTP
11：KEY2

Bit 5~4 **PAS05~PAS04**：PA2 引脚共用功能选择
00：PA2
01：SCK/SCL
10：SDO/TX
11：XT1

Bit 3~2 **PAS03~PAS02**：PA1 引脚共用功能选择
00/10：PA1/PTCK
01：SCK/SCL
11：KEY1

Bit 1~0 **PAS01~PAS00**：PA0 引脚共用功能选择
00/10：PA0
01：SDA/SDI/RX
11：XT2

- **PAS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
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 **PAS17~PAS16**：PA7 引脚共用功能选择
00/01/10：PA7/PTPI
11：KEY6

Bit 5~4 **PAS15~PAS14**：PA6 引脚共用功能选择
00/01/10：PA6/INT
11：KEY5

Bit 3~2 **PAS13~PAS12**：PA5 引脚共用功能选择
00/10：PA5
01：SDA/SDI/RX
11：KEY4

Bit 1~0 **PAS11~PAS10**：PA4 引脚共用功能选择
00/10：PA4
01：SDO/TX
11：KEY3

- **PBS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
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 **PBS07~PBS06**：PB3 引脚共用功能选择
00/01/10：PB3
11：KEY10

Bit 5~4 **PBS05~PBS04**：PB2 引脚共用功能选择
00/01/10：PB2
11：KEY9

Bit 3~2 **PBS03~PBS02**：PB1 引脚共用功能选择
00/01/10：PB1
11：KEY8

Bit 1~0 **PBS01~PBS00**：PB0 引脚共用功能选择
00/01：PB0
10：PTPB
11：KEY7

- **PBS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
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 **PBS17~PBS16**：PB7 引脚共用功能选择
00/01/10：PB7
11：KEY14

Bit 5~4 **PBS15~PBS14**：PB6 引脚共用功能选择
00/01/10：PB6
11：KEY13

Bit 3~2 **PBS13~PBS12**：PB5 引脚共用功能选择
00/01/10：PB5
11：KEY12

Bit 1~0 **PBS11~PBS10**：PB4 引脚共用功能选择
00/01/10：PB4
11：KEY11

- **PCS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
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 **PCS07~PCS06**：PC3 引脚共用功能选择
00/01/10：PC3
11：KEY18

Bit 5~4 **PCS05~PCS04**：PC2 引脚共用功能选择
00/01/10：PC2
11：KEY17

Bit 3~2 **PCS03~PCS02**：PC1 引脚共用功能选择
00/01/10：PC1
11：KEY16

Bit 1~0 **PCS01~PCS00**：PC0 引脚共用功能选择
00/01/10：PC0
11：KEY15

- **PCS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
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 **PCS17~PCS16**：PC7 引脚共用功能选择
00/01/10：PC7
11：KEY22

Bit 5~4 **PCS15~PCS14**：PC6 引脚共用功能选择
00/01/10：PC6
11：KEY21

Bit 3~2 **PCS13~PCS12**：PC5 引脚共用功能选择
00/01/10：PC5
11：KEY20

Bit 1~0 **PCS11~PCS10**：PC4 引脚共用功能选择
00/01/10：PC4
11：KEY19

- **PDS0 寄存器 – BS83B24C**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PDS03	PDS02	PDS01	PDS00
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”

Bit 3~2 **PDS03~PDS02**：PD1 引脚共用功能选择
00/01/10：PD1
11：KEY24

Bit 1~0 **PDS01~PDS00**：PD0 引脚共用功能选择
00/01/10：PD0
11：KEY23

- **PDS0 寄存器 – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
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 **PDS07~PDS06**：PD3 引脚共用功能选择
00/01/10：PD3
11：KEY26

Bit 5~4 **PDS05~PDS04**：PD2 引脚共用功能选择
00/01/10：PD2
11：KEY25

Bit 3~2 **PDS03~PDS02**：PD1 引脚共用功能选择
00/01/10：PD1
11：KEY24

Bit 1~0 **PDS01~PDS00**：PD0 引脚共用功能选择
00/01/10：PD0
11：KEY23

- **PDS1 寄存器 – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	PDS17	PDS16	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
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 **PDS17~PDS16**：PD7 引脚共用功能选择
00/01/10：PD7
11：KEY30

Bit 5~4 **PDS15~PDS14**：PD6 引脚共用功能选择
00/01/10：PD6
11：KEY29

Bit 3~2 **PDS13~PDS12**：PD5 引脚共用功能选择
00/01/10：PD5
11：KEY28

Bit 1~0 **PDS11~PDS10**：PD4 引脚共用功能选择
00/01/10：PD4
11：KEY27

- **PES0 寄存器 – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
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 **PES07~PES06**：PE3 引脚共用功能选择
00/01：PE3
10：CTP
11：KEY34

Bit 5~4 **PES05~PES04**：PE2 引脚共用功能选择
00/01/10：PE2/CTCK
11：KEY33

Bit 3~2 **PES03~PES02**：PE1 引脚共用功能选择
00/01/10：PE1
11：KEY32

Bit 1~0 **PES01~PES00**：PE0 引脚共用功能选择
00/01/10：PE0
11：KEY31

- **PES1 寄存器 – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
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 **PES17~PES16**：PE7 引脚共用功能选择
00/01/10：PE7
11：KEY38

Bit 5~4 **PES15~PES14**：PE6 引脚共用功能选择
00/01/10：PE6
11：KEY37

Bit 3~2 **PES13~PES12**：PE5 引脚共用功能选择
00/01/10：PE5
11：KEY36

Bit 1~0 **PES11~PES10**：PE4 引脚共用功能选择
00/01：PE4
10：CTPB
11：KEY35

- **PFS0 寄存器 – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PFS03	PFS02	PFS01	PFS00
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”

Bit 3~2 **PFS03~PFS02**：PF1 引脚共用功能选择
00/01/10：PF1
11：KEY40

Bit 1~0 **PFS01~PFS00**：PF0 引脚共用功能选择
00/01/10：PF0
11：KEY39

- **IFS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	IFS2	IFS1	IFS0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 未定义，读为“0”

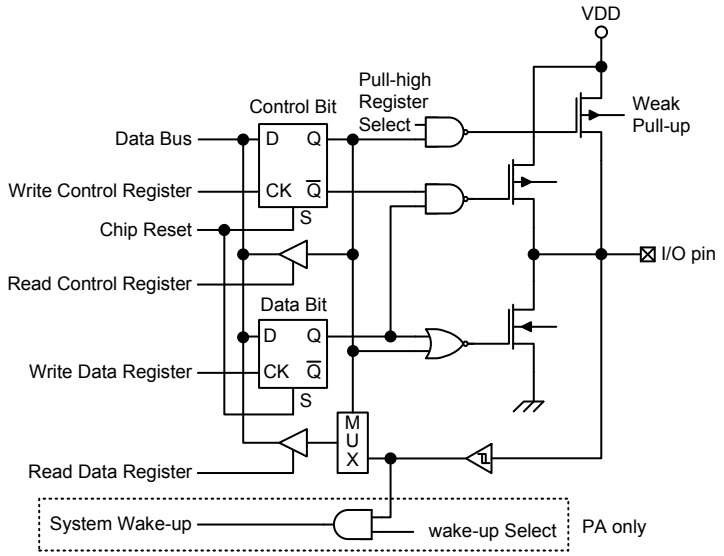
Bit 2 **IFS2**：INT 输入源引脚选择
0：PA6
1：PA3

Bit 1 **IFS1**：USIM SCK/SCL 输入源引脚选择
0：PA2
1：PA1

Bit 0 **IFS0**：USIM SDA/SDI/RX 输入源引脚选择
0：PA0
1：PA5

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 / 输出端口结构

编程注意事项

在编程中，最先要考虑的是端口的初始化。复位之后，所有的输入 / 输出数据及端口控制寄存器都将被设为逻辑高。所有输入 / 输出引脚默认为输入状态，而其电平则取决于其它相连接电路以及是否选择了上拉电阻。如果端口控制寄存器，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。该系列单片机提供几个定时器模块 (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 / 事件计数器，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。

这里只介绍各种 TM 的共性，更多详细资料请参考简易型和周期型定时器章节。

简介

该系列单片机包含了几个 TM 且每个 TM 可分为特定类型，即简易型 TM 或周期型 TM。虽然性质相似，但不同 TM 特性复杂度不同。本章介绍简易型和周期型 TM 的共性，更多详细资料分别见后面各章。两种类型 TM 的特性和区别见下表。

TM 功能	CTM	PTM
定时 / 计数器	√	√
捕捉输入	—	√
比较匹配输出	√	√
PWM 通道数	1	1
单脉冲输出	—	1
PWM 对齐方式	边沿对齐	边沿对齐
PWM 调节周期 & 占空比	占空比或周期	占空比或周期

TM 功能概要

单片机型号	CTM	PTM
BS83B24C	—	10-bit PTM
BS83C40C	10-bit CTM	10-bit PTM

TM 名称 / 类型参考

TM 操作

不同类型的 TM 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引脚来驱动内部 TM 计数器。

TM 时钟源

驱动 TM 计数器的时钟源很多。通过设置 xTM 控制寄存器的 xTCK2~xTCK0 位，选择所需的时钟源。其中“x”代表“C”或“P”型 TM。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 xTCK 引脚。xTCK 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

简易型和周期型 TM 都有两个内部中断，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

无论哪种类型的 TM，都有 TM 输入引脚，xTCK。TM 输入引脚 xTCK 作为 TM 时钟源输入脚，通过设置 xTMC0 寄存器中的 xTCK2~xTCK0 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 PTM 单脉冲输出模式的外部触发输入引脚。

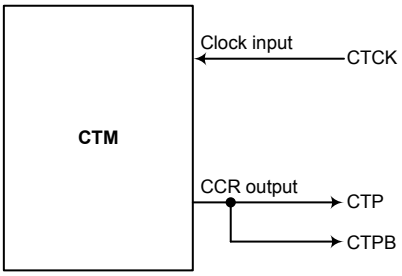
PTM 具有另一种 TM 输入引脚 PTPI 作为捕捉输入脚，其有效边沿有上升沿、下降沿和双沿，通过设置 PTMC1 寄存器中的 PTIO1~PTIO0 位来选择有效边沿类型。

每个 TM 各有两个输出引脚，xTP 和 xTPB。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 xTP 和 xTPB 输出引脚也被 TM 用来产生 PWM 输出波形。

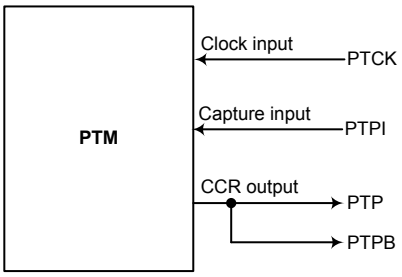
当 TM 输入 / 输出引脚与其它功能共用时，TM 输入 / 输出功能需要通过相关引脚共用功能选择寄存器先被设置，详见引脚共用功能章节。

单片机型号	CTM		PTM	
	输入	输出	输入	输出
BS83B24C	—	—	PTCK, PTPI	PTP, PTPB
BS83C40C	CTCK	CTP, CTPB	PTCK, PTPI	PTP, PTPB

TM 外部引脚



CTM 功能引脚框图

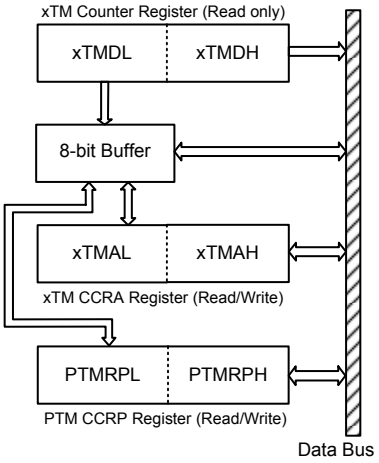


PTM 功能引脚框图

编程注意事项

TM 计数寄存器和捕捉 / 比较寄存器 CCRA、CCRP 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。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。

CCRA 和 CCRP 寄存器访问方式如下图所示，读写这些成对的寄存器需通过特殊的方式。建议使用“MOV”指令按照以下步骤访问 CCRA 或 CCRP 低字节寄存器，xTMAL 或 PTMRPL，否则可能导致无法预期的结果。

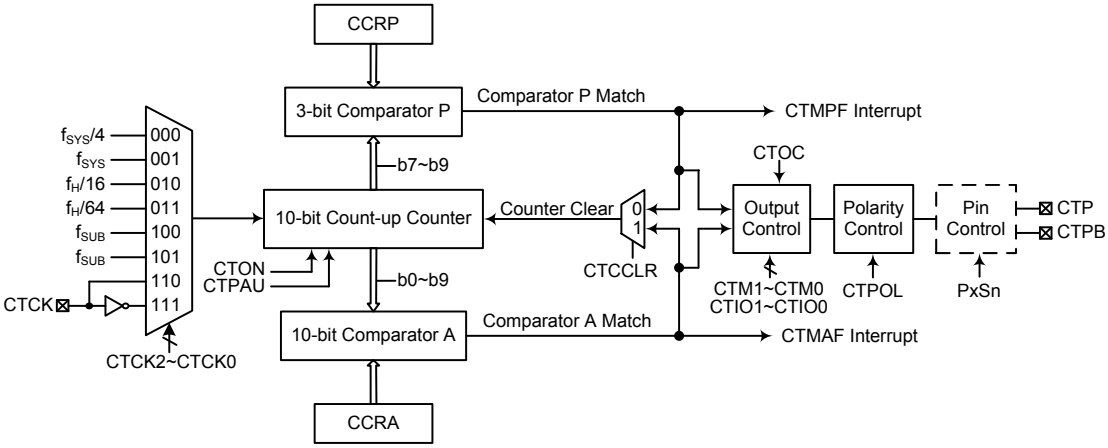


读写流程如下步骤所示：

- 写数据至 CCRA 或 CCRP
  - ♦ 步骤 1. 写数据至低字节寄存器 xTMAL 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 xTMAH 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。
- 从计数器寄存器和 CCRA 或 CCRP 中读取数据
- 步骤 1. 由高字节寄存器 xTMDH、xTMAH 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 xTMDL、xTMAL 或 PTMRPL 读取数据
  – 注意，此时读取 8-bit 缓存器中的数据。

简易型 TM – CTM

简易型 TM 包括三种工作模式，即比较匹配输出、定时 / 事件计数器和 PWM 输出模式。简易型 TM 由一个外部输入脚控制并驱动两个外部输出脚。



注：CTPB 为 CTP 引脚的反相输出。

10-bit 简易型 TM 方框图

简易型 TM 操作

简易型 TM 核心是一个由用户选择的内部或外部时钟源驱动的 10-bit 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 CCRP 和 CCRA 寄存器中的值进行比较。CCRP 是 3 位的，与计数器的高 3 位比较；而 CCRA 是 10 位的，与计数器的所有位比较。

通过应用程序改变 10-bit 计数器值的唯一方法是使 CTON 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 CTM 中断信号。简易型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，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内部寄存器来实现的。

简易型 TM 寄存器介绍

简易型 TM 的所有操作由几个寄存器控制。其中包含一对只读寄存器用来存放 10 位计数器的值，一对读 / 写寄存器存放 10 位 CCRA 的值，剩下两个控制寄存器设置不同的操作和控制模式以及 CCRP 的 3 个位。

寄存器名称	位							
	7	6	5	4	3	2	1	0
CTMC0	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
CTMC1	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
CTMDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMDH	—	—	—	—	—	—	D9	D8
CTMAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMAH	—	—	—	—	—	—	D9	D8

10-bit 简易型 TM 寄存器列表

- **CTMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
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 **CTPAU**：CTM 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，CTM 保持上电状态并继续耗电。当此位由低到高转换时，计数器将保留其剩余值，直到此位再次改变为低电平，从此值开始继续计数。

Bit 6~4 **CTCK2~CTCK0**：选择 CTM 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：CTCK 上升沿时钟
111：CTCK 下降沿时钟
此三位用于选择 CTM 的时钟源。选择保留时钟输入将有效地除能内部计数器。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 **CTON**：CTM 计数器 On/Off 控制位
0：Off
1：On
此位控制 CTM 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 CTM。清零此位将停止计数器并关闭 CTM 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 CTM 处于比较匹配输出模式或 PWM 输出模式时，当 CTON 位经由低到高转换时，CTM 输出脚将复位至 CTOC 位指定的初始值。

Bit 2~0 **CTRP2~CTRP0**：CTM CCRP 3-bit 寄存器，与 CTM 计数器 bit 9~bit 7 比较
比较器 P 匹配周期
000：1024 个 CTM 时钟周期
001：128 个 CTM 时钟周期
010：256 个 CTM 时钟周期
011：384 个 CTM 时钟周期
100：512 个 CTM 时钟周期
101：640 个 CTM 时钟周期
110：768 个 CTM 时钟周期
111：896 个 CTM 时钟周期
此三位设定内部 CCRP 3-bit 寄存器的值，然后与内部计数器的高三位进行比较。如果 CTCCLR 位设定为 0 时，比较结果为 0 并清除内部计数器。CTCCLR 位设为低，内部计数器在比较器 P 比较匹配发生时被重置；由于 CCRP 只与计数器高三位比较，比较结果是 128 时钟周期的倍数。CCRP 被清零时，实际上会使得计数器在最大值溢出。

- **CTMC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
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 **CTM1~CTM0**：选择 CTM 工作模式位
00：比较匹配输出模式
01：未定义
10：PWM 输出模式
11：定时 / 计数器模式
这两位设置 CTM 需要的工作模式。为了确保操作可靠，CTM 应在 CTM1 和 CTM0 位有任何改变前先关掉。在定时 / 计数器模式，CTM 输出脚状态未定义。

Bit 5~4 **CTIO1~CTIO0**：选择 CTP 输出功能位
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
00：PWM 输出无效状态
01：PWM 输出有效状态
10：PWM 输出
11：未定义
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 CTM 输出脚如何改变状态。这两位值的选择取决于 CTM 运行在哪种模式下。
在比较匹配输出模式下，CTIO1 和 CTIO0 位决定当比较器 A 比较匹配输出发生时 CTM 输出脚如何改变状态。当比较器 A 比较匹配输出发生时 CTM 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。CTM 输出脚的初始值通过 CTMC1 寄存器的 CTOC 位设置取得。注意，由 CTIO1 和 CTIO0 位得到的输出电平必须与通过 CTOC 位设置的初始值不同，否则当比较匹配发生时，CTM 输出脚将不会发生变化。在 CTM 输出脚改变状态后，通过 CTON 位由低到高电平的转换复位至初始值。
在 PWM 输出模式，CTIO1 和 CTIO0 用于决定比较匹配条件发生时怎样改变 CTM 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。仅在 CTM 关闭时改变 CTIO1 和 CTIO0 位的值是很有必要的。若在 CTM 运行时改变 CTIO1 和 CTIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **CTOC**：CTP 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
0：低有效
1：高有效
这是 CTM 输出脚输出控制位。它取决于 CTM 此时正运行于比较匹配输出模式还是 PWM 输出模式。若 CTM 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 CTM 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。

Bit 2 **CTPOL**：CTP 输出极性控制位
0：同相
1：反相
此位控制 CTP 输出脚的极性。此位为高时 CTM 输出脚反相，为低时 CTM 输出脚同相。若 CTM 处于定时 / 计数器模式时其不受影响。

Bit 1 **CTDPX**：CTM PWM 周期 / 占空比控制位
0：CCRP – 周期；CCRA – 占空比
1：CCRP – 占空比；CCRA – 周期
此位决定 CCRA 与 CCRP 寄存器哪个被用于 PWM 波形的周期和占空比控制。

Bit 0 **CTCCLR**：选择 CTM 计数器清零条件位
0：CTM 比较器 P 匹配
1：CTM 比较器 A 匹配
此位用于选择清除计数器的方法。简易型 TM 包括两个比较器 - 比较器 A 和比较器 P。这两个比较器每个都可以用作清除内部计数器。CTCCLR 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。CTCCLR 位在 PWM 输出模式时未使用。

- **CTMDL 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 CTM 计数器低字节寄存器 bit 7~bit 0
CTM 10-bit 计数器 bit 7~bit 0

- **CTMDH 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 CTM 计数器高字节寄存器 bit 1~bit 0
CTM 10-bit 计数器 bit 9~bit 8

- **CTMAL 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 CTM CCRA 低字节寄存器 bit 7~bit 0
CTM 10-bit CCRA bit 7~bit 0

- **CTMAH 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　CTM CCRA 高字节寄存器 bit 1~bit 0
CTM 10-bit CCRA bit 9~bit 8

简易型 TM 工作模式

简易型 TM 有三种工作模式，即比较匹配输出模式，PWM 输出模式或定时 / 计数器模式。通过设置 CTMC1 寄存器的 CTM1 和 CTM0 位选择任意工作模式。

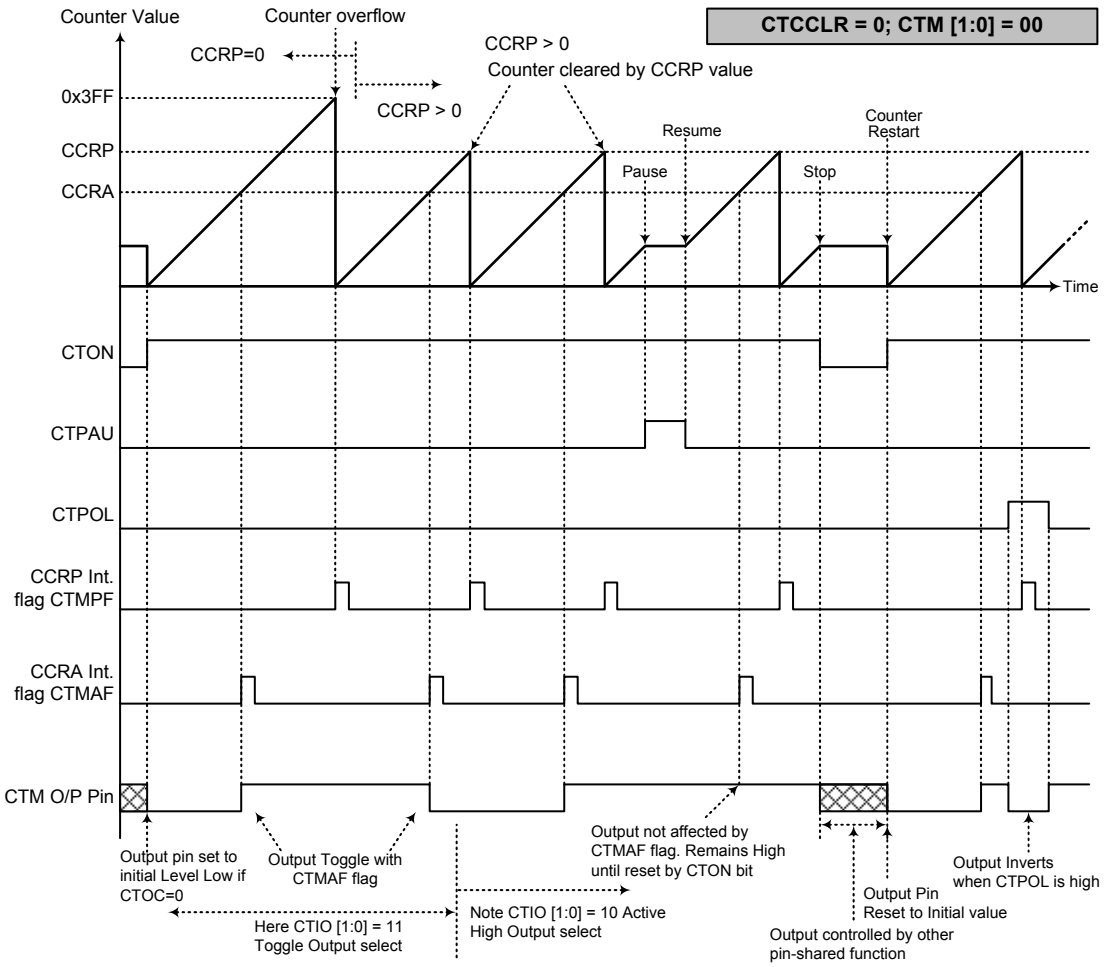
比较匹配输出模式

为使 CTM 工作在此模式，CTMC1 寄存器中的 CTM1 和 CTM0 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 CTCCLR 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 CTMAF 和 CTMPF 将分别置起。

如果 CTMC1 寄存器的 CTCCLR 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 CTMAF 中断请求标志产生。所以当 CTCCLR 为高时，不产生 CTMPF 中断请求标志。

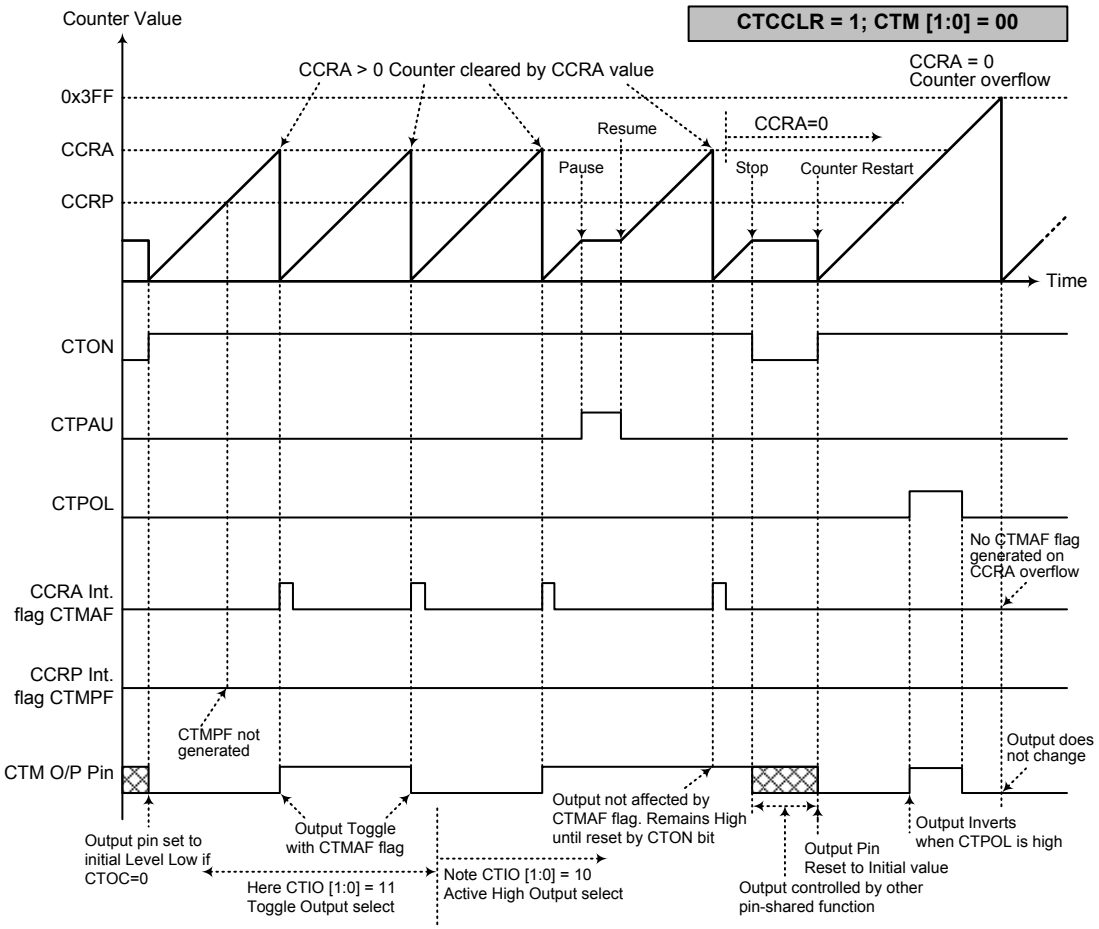
如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 CTMAF 请求标志。

正如该模式名所言，当比较匹配发生后，CTM 输出脚状态改变。当比较器 A 比较匹配发生后 CTMAF 标志产生时，CTM 输出脚状态改变。比较器 P 比较匹配发生时产生的 CTMPF 标志不影响 CTM 输出脚。CTM 输出脚状态改变方式由 CTMC1 寄存器中 CTIO1 和 CTIO0 位决定。当比较器 A 比较匹配发生时，CTIO1 和 CTIO0 位决定 TM 输出脚输出高，低或翻转当前状态。CTM 输出脚初始值，在 CTON 位由低到高电平的变化后通过 CTOC 位设置。注意，若 CTIO1 和 CTIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 – **CTCCLR=0**

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



比较匹配输出模式 – CTCCLR=1

注：1. CTCCLR=1，比较器 A 匹配将清除计数器
2. CTM 输出脚仅由 CTMAF 标志位控制
3. 在 CTON 上升沿 CTM 输出脚复位至初始值
4. CTCCLR=1 时不产生 CTMPF 标志位

定时 / 计数器模式

为使 CTM 工作在此模式，CTMC1 寄存器中的 CTM1 和 CTM0 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 CTM 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 CTM 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 CTM 工作在此模式，CTMC1 寄存器中的 CTM1 和 CTM0 位需要设置为“10”。CTM 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 CTM 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，其波形的选择就极其灵活。在 PWM 输出模式中，CTCCLR 位不影响 PWM 操作。CCRA 和 CCRP 寄存器决定 PWM 波形，一个用来清除内部计数器并控制 PWM 波形的频率，另一个用来控制占空比。哪个寄存器控制频率或占空比取决于 CTMC1 寄存器的 CTDPX 位。所以 PWM 波形频率和占空比由 CCRA 和 CCRP 寄存器共同决定。

当比较器 A 或比较器 P 比较匹配发生时，将产生 CCRA 或 CCRP 中断标志。CTMC1 寄存器中的 CTOC 位决定 PWM 波形的极性，CTIO1 和 CTIO0 位使能 PWM 输出或将 CTM 输出脚置为逻辑高或逻辑低。CTPOL 位对 PWM 输出波形的极性取反。

- **10-bit CTM，PWM 输出模式，边沿对齐模式，CTDPX=0**

CCRP	1~7	0
Period	CCRP×128	1024
Duty	CCRA	

若 f_{SYS}=8MHz，CTM 时钟源选择 f_{SYS}/4，CCRP=2，CCRA=128，

CTM PWM 输出频率 = (f_{SYS}/4)/(2×128)=f_{SYS}/1024=7.812kHz，

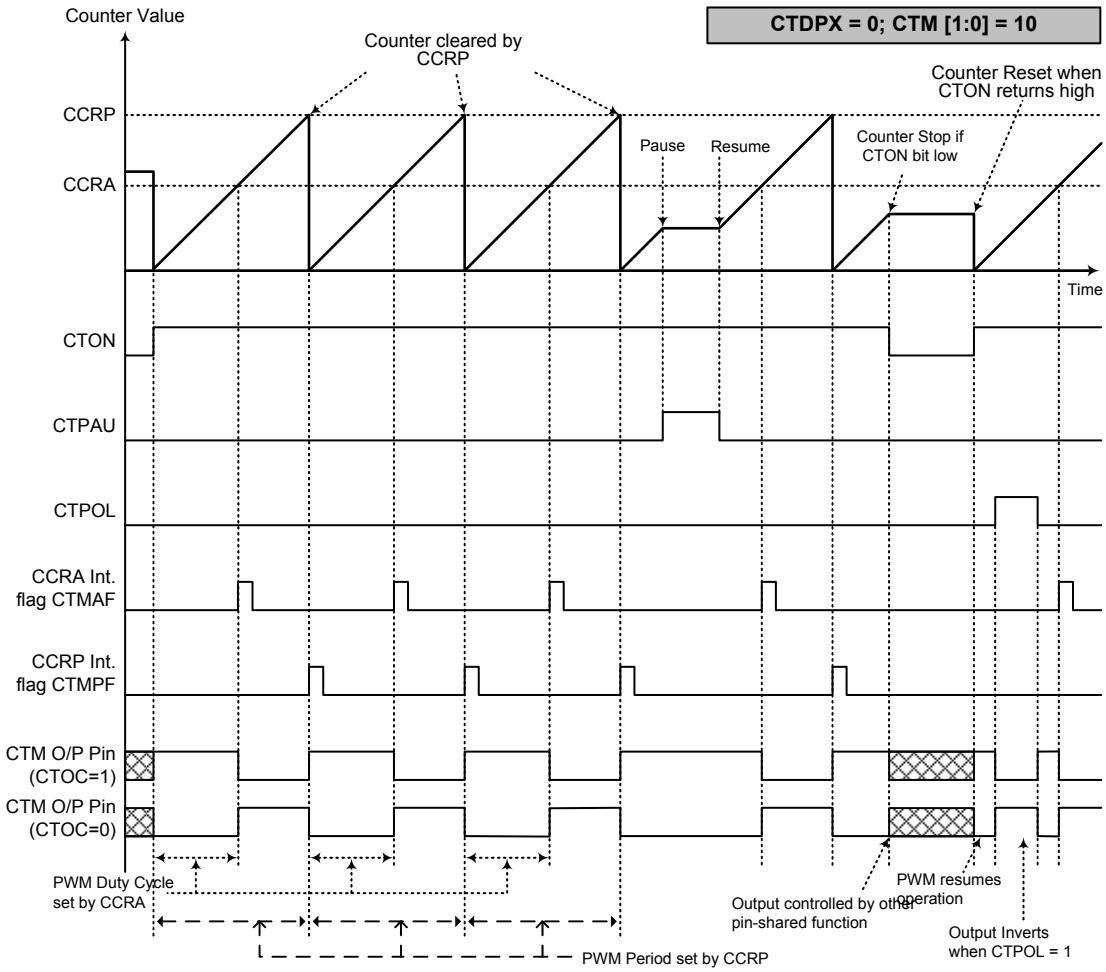
duty=128/(2×128)=50%。

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%。

- **10-bit CTM，PWM 输出模式，边沿对齐模式，CTDPX=1**

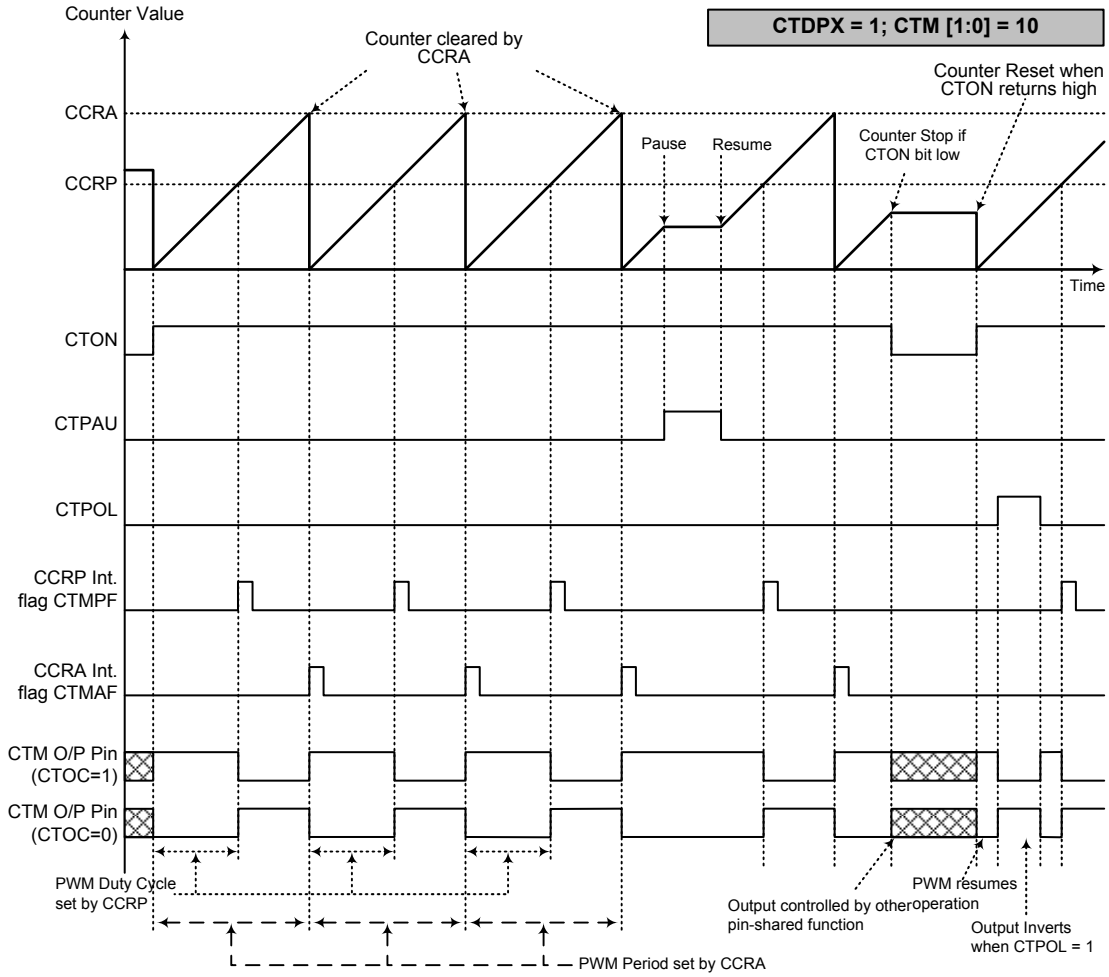
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

PWM 的输出周期由 CCRA 寄存器的值与 CTM 的时钟共同决定，PWM 的占空比由 CCRP 寄存器的值决定。



PWM 输出模式 – CTDPX=0

注：1. CTDPX=0，CCRP 匹配将清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 CTIO[1:0]=00 或 01，PWM 功能不变
4. CTCCLR 位不影响 PWM 操作

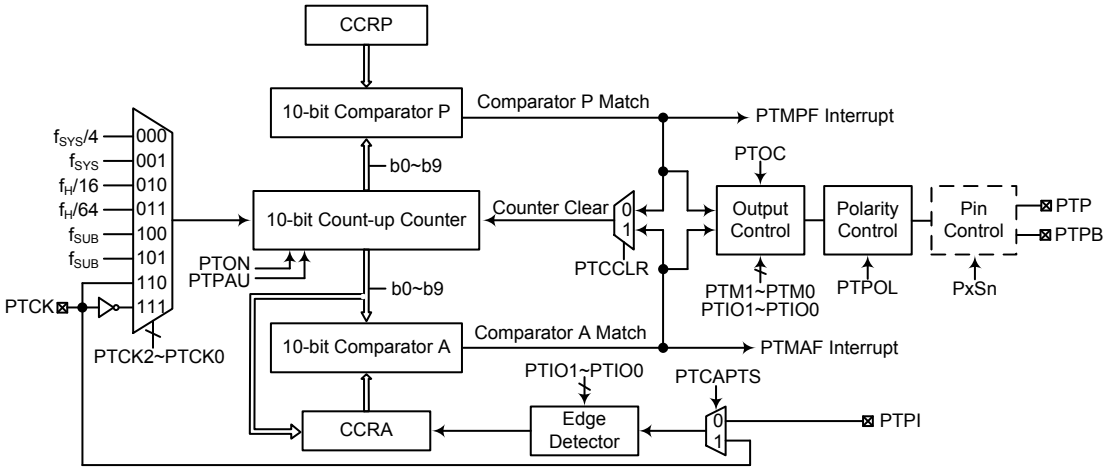


PWM 输出模式 – CTDPX=1

注：1. CTDPX=1，CCRA 匹配将清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 CTIO[1:0]=00 或 01，PWM 功能不变
4. CTCCLR 位不影响 PWM 操作

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 引脚的反相输出。

10-bit 周期型 TM 方框图

周期型 TM 操作

周期型 TM 的核心是一个由用户选择的内部或外部时钟源驱动的 10-bit 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 CCRP 和 CCRA 寄存器中的值进行比较。CCRP 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 PTM 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不同时钟源驱动，也可以控制输出脚。所有工作模式的设置都是通过设置相关寄存器来实现的。

周期型 TM 寄存器描述

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工作模式。

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

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

- **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~PTCK1**：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时，当 PTON 位经由低到高转换时，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 输出脚状态未定义。

Bit 5~4 **PTIO1~PTIO0**：选择 PTP 或 PTPI 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：PWM 输出无效状态
01：PWM 输出有效状态
10：PWM 输出
11：单脉冲输出
捕捉输入模式
00：在 PTCK 或 PTPI 上升沿输入捕捉
01：在 PTCK 或 PTPI 下降沿输入捕捉
10：在 PTCK 或 PTPI 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 输出脚如何改变状态。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。仅在 PTM 关闭时改变 PTIO1 和 PTIO0 位的值是很有必要的。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**：PTM 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 PTM 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。

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、单脉冲输出或输入捕捉模式时未使用。

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。

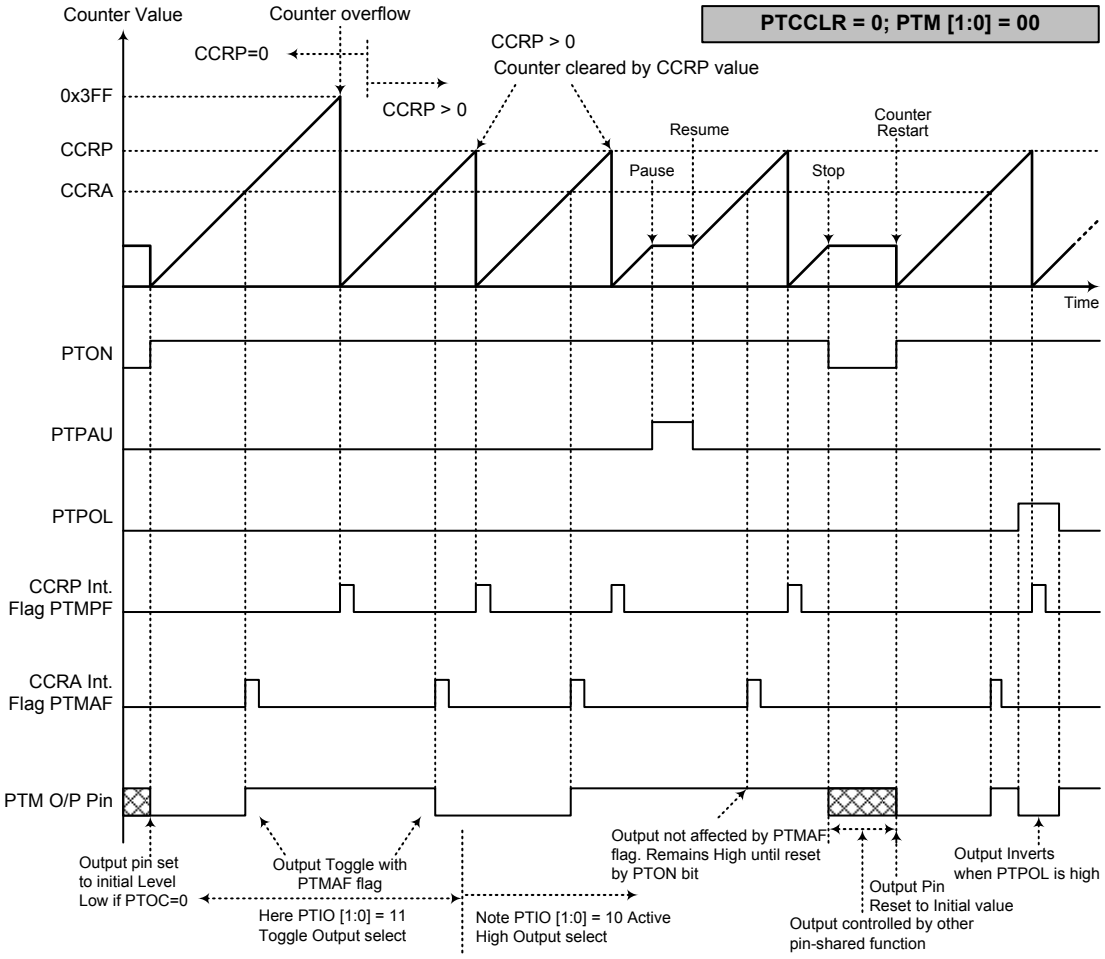
比较匹配输出模式

为使 PTM 工作在此模式，PTMC1 寄存器中的 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”。

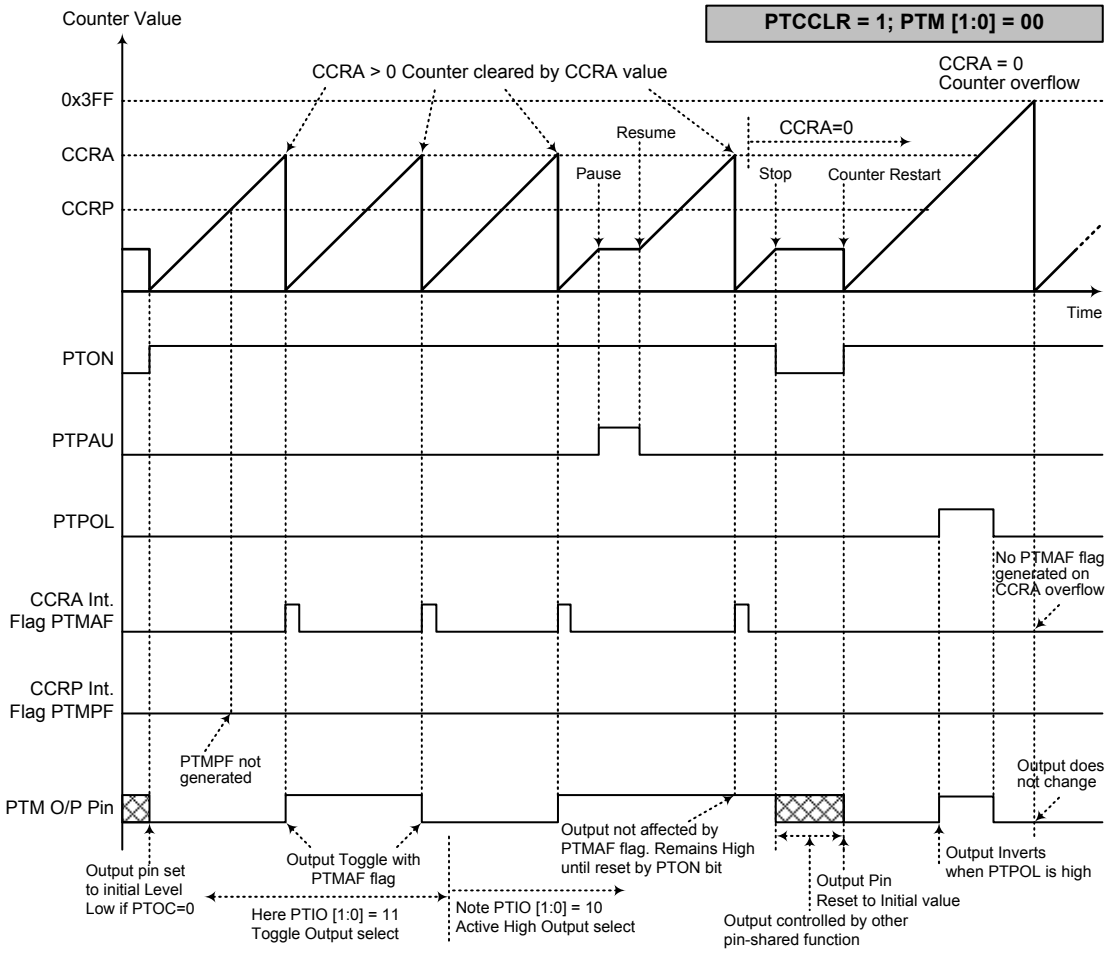
若 CCRA 全部清零，则计数器将在其最大值 3FFH 处溢出，但并不产生 PTMAF 中断请求标志位。

正如该模式名所言，当比较匹配发生后，PTM 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，PTM 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 PTM 输出脚复位至初始值
4. 10-bit PTM 最大计数器值为 0x3FF



比较匹配输出模式 – **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 PTM 输出脚复位至初始值
4. 当 PTCCLR=1 时，不会产生 PTMPF 标志
5. 10-bit PTM 最大计数器值为 0x3FF

定时 / 计数器模式

为使 PTM 工作在此模式，PTMC1 寄存器中的 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 PTM 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。

PWM 输出模式

为使 PTM 工作在此模式，PTMC1 寄存器中的 PTM1 和 PTM0 位需要设置为“10”。PTM 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 PTM 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，其波形的选择就极其灵活。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 TM 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。

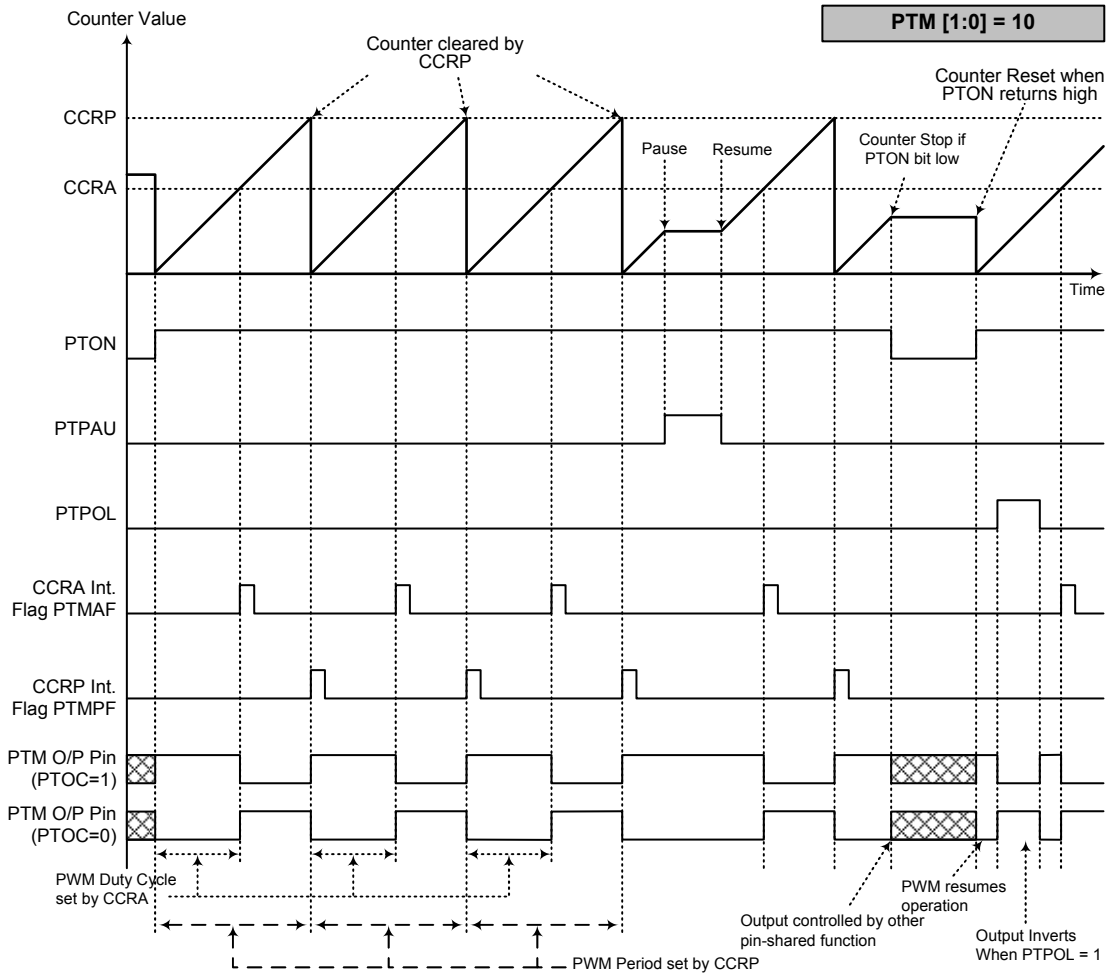
- **10-bit PTM，PWM 输出模式，边沿对齐模式**

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=8kHz，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

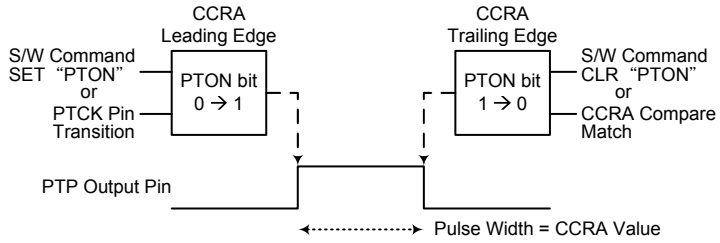
注：1. CCRP 清零计数器
2. 计数器清除并决定 PWM 周期
3. 当 PTIO[1:0]=00 或 01，PWM 功能不变
4. PTCCLR 位对 PWM 功能无影响

单脉冲输出模式

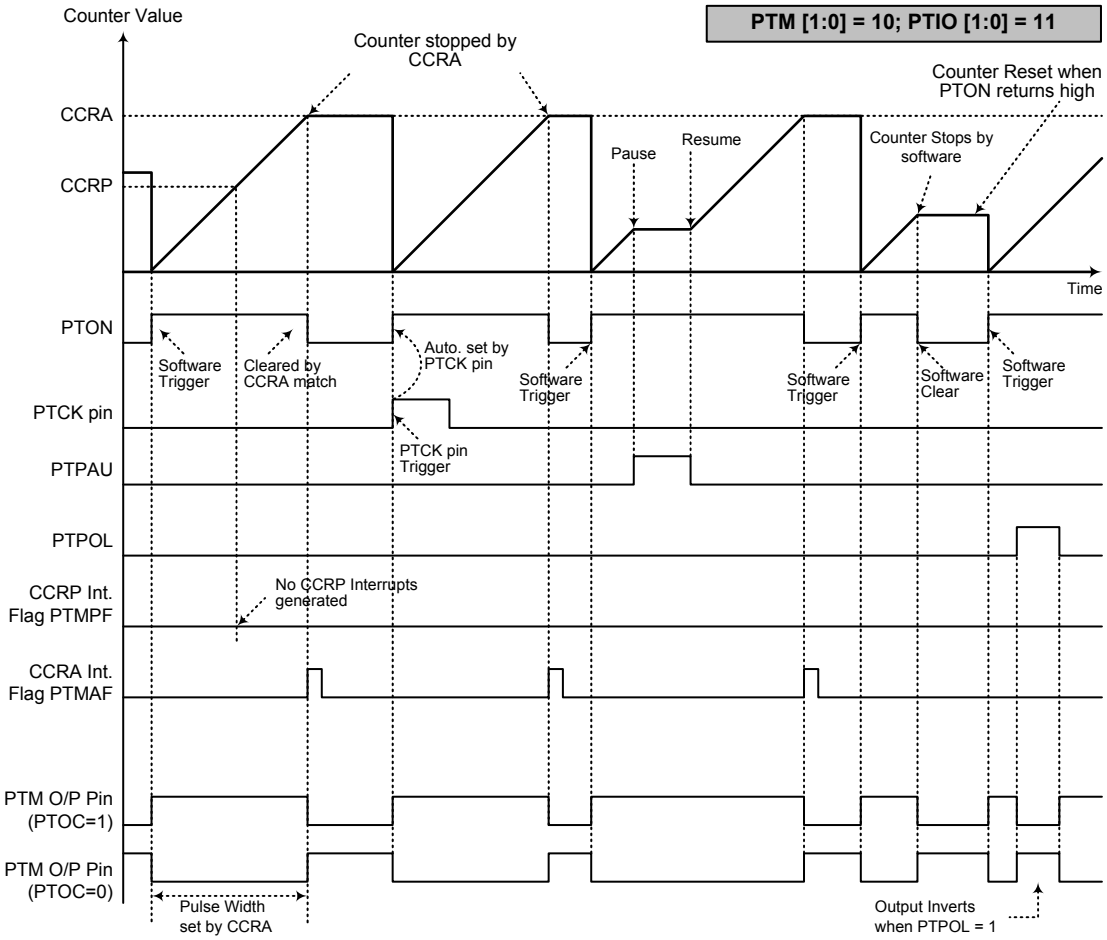
为使 PTM 工作在此模式，PTMC1 寄存器中的 PTM1 和 PTM0 位需要设置为“10”，并且相应的 PTIO1 和 PTIO0 需要设置为“11”。正如模式名所言，单脉冲输出模式，在 PTM 输出脚将产生一个脉冲输出。

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，会自动清除 PTON 位并产生单脉冲输出下降沿。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

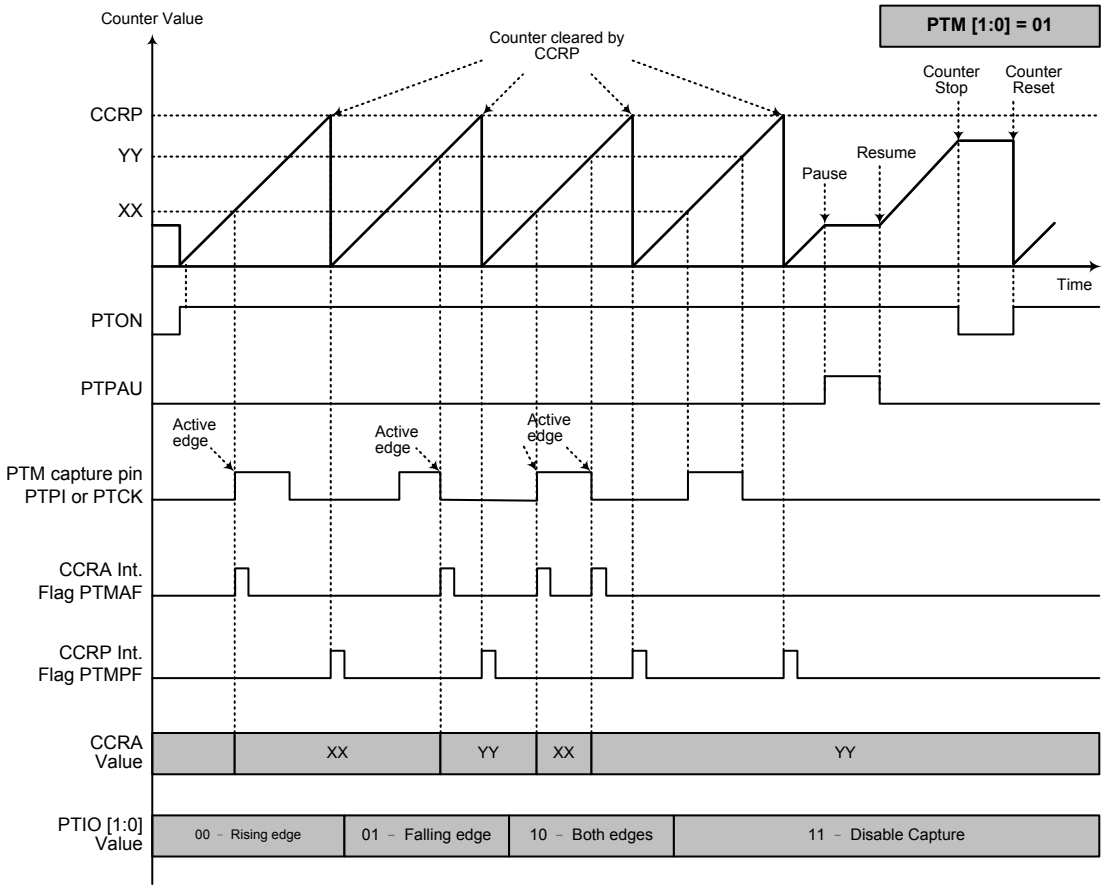
注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 PTCK 脚或设置 PTON 位为高来触发脉冲
4. PTCK 脚有效沿会自动置位 PTON
5. 单脉冲输出模式中，PTIO[1:0] 需置位“11”，且不能更改。

捕捉输入模式

为使 PTM 工作在此模式，PTMC1 寄存器中的 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 位由低到高转变时，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。无论 PTPI 或 PTCK 引脚发生哪种边沿转换，计数器将继续工作直到 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 PTM 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。如果 PTIO1 和 PTIO0 位都设置为高，无论 PTPI 或 PTCK 引脚发生哪种边沿转换都不会产生捕捉操作，但计数器仍会继续运行。

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出，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

通用串行接口模块 – USIM

此系列单片机内有一个通用串行接口模块，包括三种易与外部设备通信的串行接口：四线 SPI、两线 I^2C 或两线 UART 接口。这三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。因为 USIM 接口引脚是与其它 I/O 引脚共用，因此在使用 USIM 功能前，要先通过相应的引脚共用功能选择寄存器选定 USIM 引脚功能。因为这三种接口共用引脚和寄存器，所以要先通过 SIMC0 寄存器中的 UART 模式选择位 UMD 和 SPI/I^2C 工作模式控制位 SIM2~SIM0 选择哪一种通信接口。若 USIM 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 USIM 脚的上拉电阻。

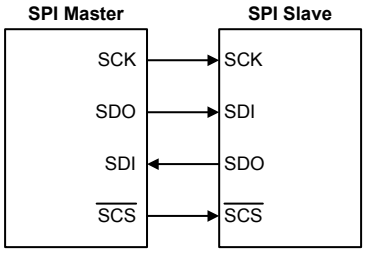
SPI 接口

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/UART 的功能脚共用。通过设定相关引脚共用选择位和 SIMC0/SIMC2 寄存器的对应位，来使能 SPI 接口。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。

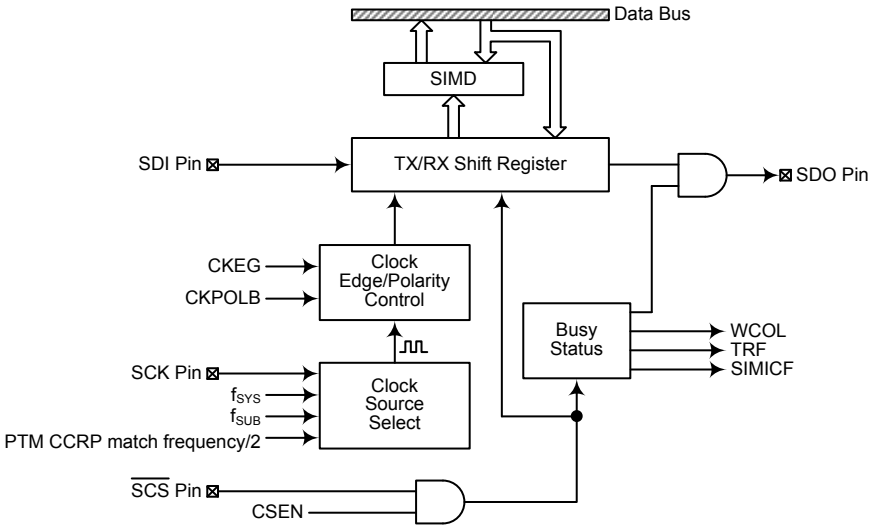


SPI 主 / 从机连接方式

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以及 CSEN、SIMEN 位的状态。



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，只有在合理设置 SIMC0 寄存器中的 UMD 位和 SIM2~SIM0 位选择 SPI 模式后，SIMC2 和 SIMD 寄存器以及它们的上电复位值才有效。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	UMD	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

SPI 数据寄存器

SIMD 用于存储发送和接收的数据。这个寄存器由 SPI 和 I^2C 功能所共用。在单片机将数据写入到 SPI 总线之前，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 实现。

- **SIMD 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	x	x	x	x	x	x	x	x

“x”：未知

Bit 7~0　**D7~D0**：USIM SPI/I^2C 数据寄存器位 bit 7~bit 0

SPI 控制寄存器

单片机中也有两个控制 SPI 接口功能的寄存器，SIMC0 和 SIMC2。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	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SIM2~SIM0**：USIM SPI/I^2C 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：I^2C 从机模式
111：未使用模式
当 UMD 位清零时，这几位用于设置 USIM SPI/I^2C 功能的工作模式，除了选择 USIM 模块的 I^2C 或 SPI 功能，还可选择 SPI 的主从模式和 SPI 的主机时钟频率。SPI 时钟源可来自于系统时钟和 f_{SUB} 也可以选择来自 PTM。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 **UMD**：UART 模式选择位
0：SPI 或 I^2C 模式
1：UART 模式
此位为 UART 模式选择位。当此位清零时，实际 SPI 或 I^2C 模式是通过 SIM2~SIM0 位选择。当选择 SPI 或 I^2C 模式时，此位必须清零。

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

Bit 1 **SIMEN**：USIM SPI/I^2C 控制位
0：除能
1：使能
此位为 USIM SPI/I^2C 接口的开 / 关控制位。此位为“0”时，USIM SPI/I^2C 接口除能，SDI、SDO、SCK 和 $\overline{SCS}$ 或 SDA 和 SCL 脚将失去 SPI 或 I^2C 功能，USIM 工作电流减小到最小值。此位为“1”时，USIM SPI/I^2C 接口使能。若 USIM 经由 UMD 位和 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 USIM 经由 UMD 位和 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**：USIM SPI 未完成标志位
0：未发生
1：发生
此位仅当 USIM 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 引脚为高电平
1：当时钟无效时，SCK 引脚为低电平
此位决定了时钟线的基础状态，若此位为高，当时钟无效时 SCK 为低电平，若此位为低，当时钟无效时 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。这两位必须在执行数据传输前先被设置好，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 $\overline{\text{SCS}}$ 引脚控制位
0：除能
1：使能
CSEN 位用于 $\overline{\text{SCS}}$ 引脚的使能 / 除能控制。此位为低时，$\overline{\text{SCS}}$ 除能并处于浮空状态。此位为高时，$\overline{\text{SCS}}$ 作为选择脚。

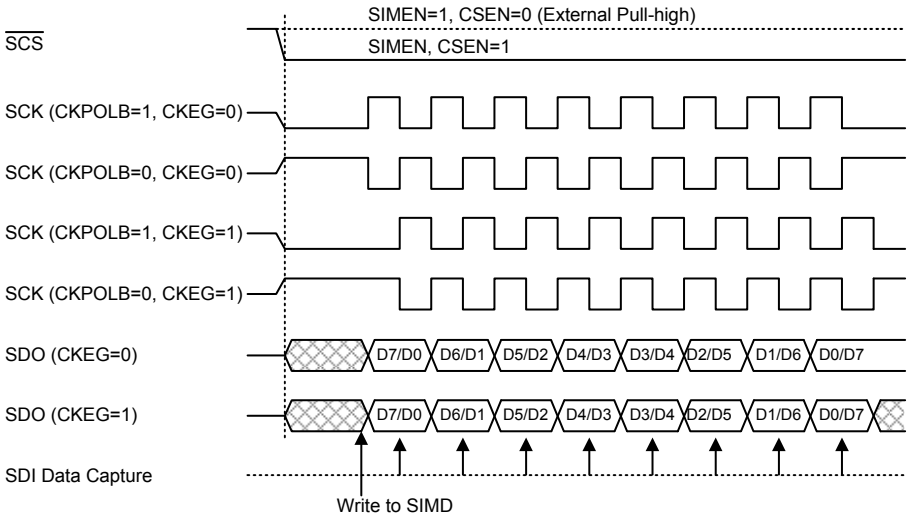
Bit 1 **WCOL**：SPI 写冲突标志位
0：无冲突
1：冲突
WCOL 标志位用于监测数据冲突的发生。此位为高时，表示在传输过程中有数据被写入 SIMD 寄存器。若数据正在被传输时，此写操作无效。此位可被应用程序清零。

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，但须通过应用程序设置为“0”。此位也可用于产生中断。

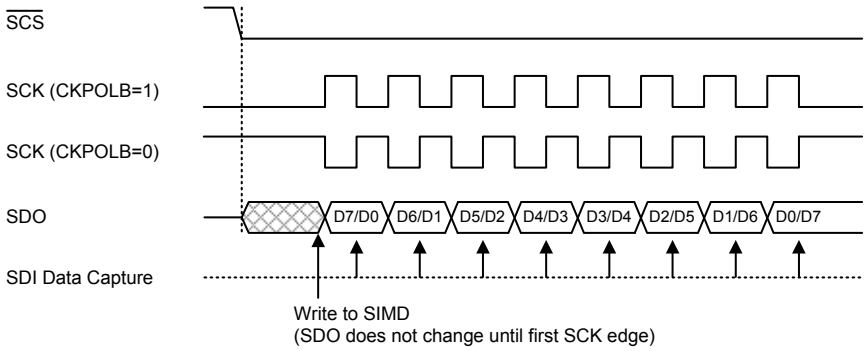
SPI 通信

将 SIMEN 设置为高，使能 SPI 功能之后，若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除只能通过应用程序完成。若单片机处于从机模式，收到主机发来的信号之后，会传输 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。

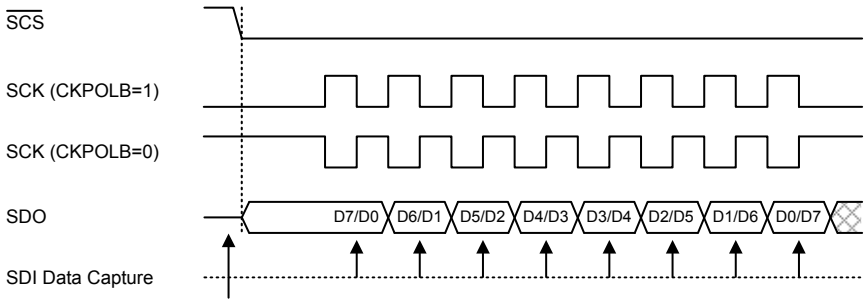
即使在单片机处于空闲模式时，若 SPI 接口使用的时钟源仍开启，SPI 功能仍将继续执行。



SPI 主机模式时序

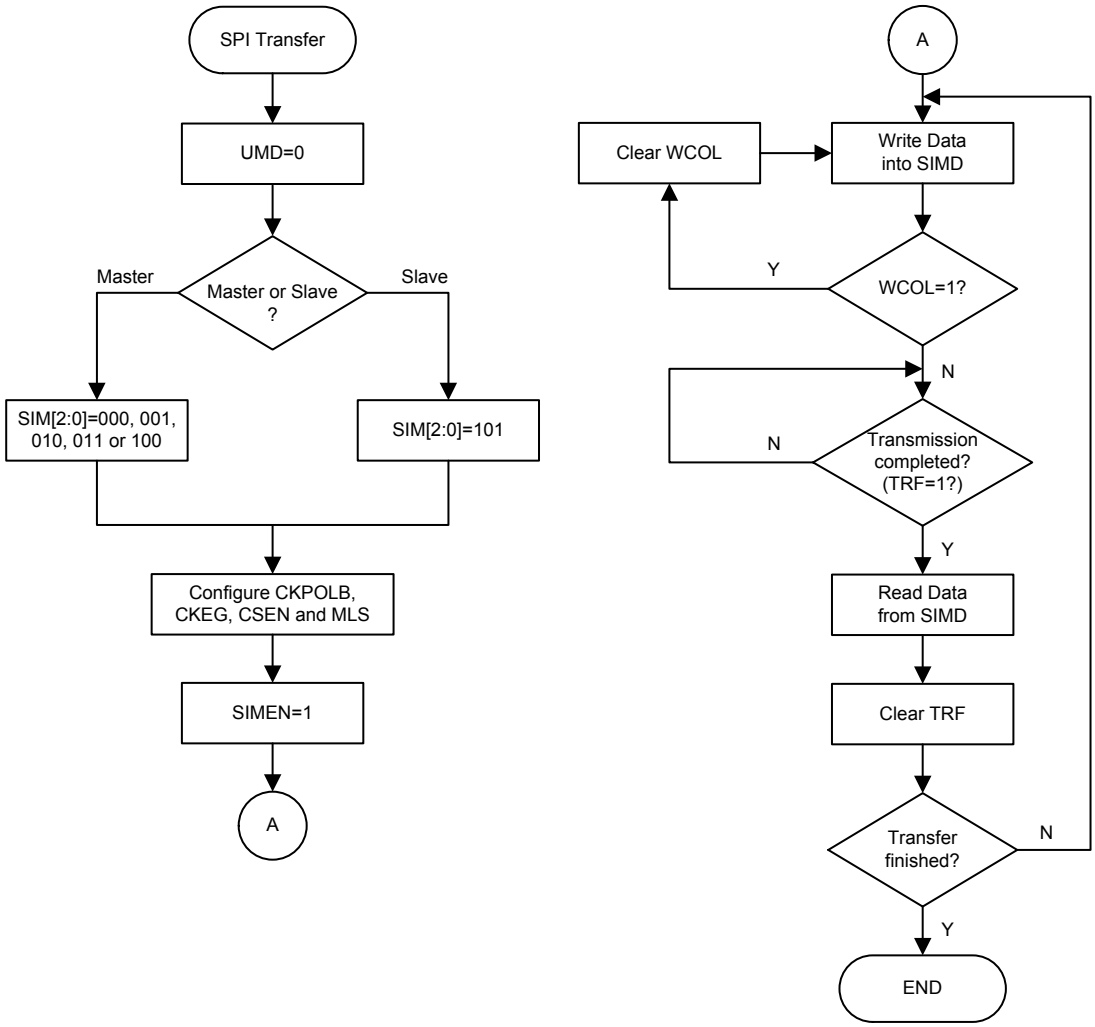


SPI 从机模式时序 – CKEG=0



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

SPI 从机模式时序 – CKEG=1



SPI 传输控制流程图

SPI 使能 / 除能

设置 CSEN=1、$\overline{SCS}$=0 将使能 SPI 总线，然后等待写数据到 SIMD 寄存器 (TXRX 缓存器)。单片机处于主机模式，数据写入 SIMD 寄存器后，自动开始数据传输或接收操作。数据传输完成时，TRF 位将自动被置位。单片机处于从机模式，SCK 引脚上收到脉冲信号之后，会传输 TXRX 中的数据，或将 SDI 引脚上的数据移入。

当 SPI 总线除能时，通过设置相应引脚共用控制位，SCK、SDI、SDO、$\overline{SCS}$ 可作为 I/O 口或其它功能引脚使用。

SPI 操作步骤

四线制 SPI 接口可完成所有主 / 从模式通信工作。

在 SIMC2 寄存器中，CSEN 位控制 SPI 接口的所有功能。设置此位为高，$\overline{SCS}$ 信号线有效将使能 SPI 接口。设置此位为低，SPI 接口将除能，$\overline{SCS}$ 信号线处于浮空状态因此不能控制 SPI 接口。CSEN 位和 SIMC0 寄存器中的 SIMEN 位设置为高，使得 SDI 信号线处于浮空状态且 SDO 信号线为高电平。主机模式中，如果 SCK 信号线为高还是低取决于 SIMC2 寄存器中的时钟极性选择位 CKPOLB。从机模式中，SCK 信号线处于浮空状态。如果 SIMEN 位设置为低，SPI 接口被除能，通过设置相应引脚共用控制位，$\overline{SCS}$、SDI、SDO 和 SCK 可作为 I/O 口或其它功能引脚使用。主机模式中，当数据被写入 SIMD 寄存器后，主机完成所有的数据传输初始化，并控制时钟信号。从机模式中，由外部主机发出数据传送 / 接收时钟信号。下面介绍主从模式中数据传输步骤。

主机模式：

- 步骤 1
  设置 SIMC0 控制寄存器中的 UMD 和 SIM2~SIM0 位，选择 SPI 主机模式和时钟源。
- 步骤 2
  设置 CSEN 和 MLS 位，选择高位或低位数据优先传送，这必须与从机设备一致。
- 步骤 3
  设置 SIMC0 控制寄存器中的 SIMEN 位，使能 SPI 接口功能。
- 步骤 4
  对于写操作：写数据到 SIMD 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。再使用 SCK 和 $\overline{SCS}$ 信号线将数据输出。跳至步骤 5。
  对于读操作：从 SDI 信号线移入的数据将被存储在 TXRX 缓存器中，直到所有数据接收完毕，此时数据全部锁存至 SIMD 寄存器。
- 步骤 5
  检测 WCOL 位，若此位为高，则发生数据冲突并跳回至步骤 4；若为低，则继续执行下面的步骤。
- 步骤 6
  检测 TRF 位或等待 USIM SPI 串行总线中断发生。
- 步骤 7
  从 SIMD 寄存器中读数据。
- 步骤 8
  清除 TRF。

- 步骤 9
  跳回至步骤 4。

从机模式：

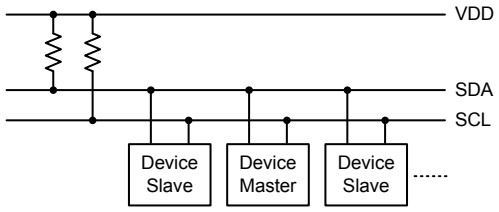
- 步骤 1
  设置 SIMC0 控制寄存器中的 UMD 和 SIM2~SIM0 位，选择 SPI 从机模式。
- 步骤 2
  设置 CSEN 和 MLS 位，选择高位或低位数据优先传送，这必须与主机设备一致。
- 步骤 3
  设置 SIMC0 控制寄存器中的 SIMEN 位，使能 SPI 接口功能。
- 步骤 4
  对于写操作：写数据到 SIMD 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。等待主机时钟 SCK 信号和 $\overline{SCS}$ 信号。跳至步骤 5。
  对于读操作：从 SDI 信号线移入的数据将被存储在 TXRX 缓存器中，直到所有数据接收完毕，此时数据全部锁存至 SIMD 寄存器。
- 步骤 5
  检测 WCOL 位，若此位为高，则发生数据冲突并跳回至步骤 4；若为低，则继续执行下面的步骤。
- 步骤 6
  检测 TRF 位或等待 USIM SPI 串行总线中断发生。
- 步骤 7
  从 SIMD 寄存器中读数据。
- 步骤 8
  清除 TRF。
- 步骤 9
  跳回至步骤 4。

错误侦测

SIMC2 寄存器中的 WCOL 位用于数据传输期间监测数据冲突的发生。此位由 SPI 串行接口设置为高，而由应用程序来清除为零。在数据传输期间如果写数据到 SIMD，此位被置高提示数据冲突发生，并阻止数据继续被写入。

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。

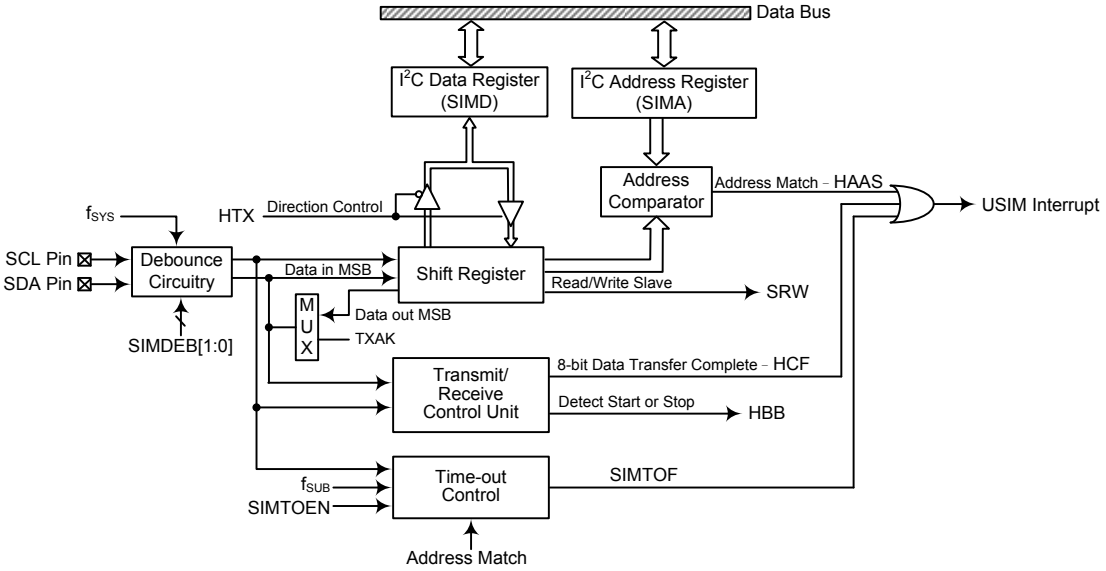


I²C 主从总线连接图

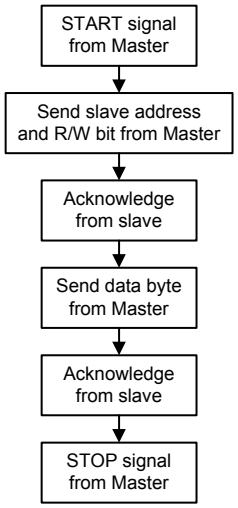
I²C 接口操作

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。即使 I²C 设备被激活，与 SCL/SDA 引脚共用的 I/O 口上拉电阻控制功能仍有效，其上拉电阻功能由相应的上拉电阻控制寄存器控制。



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，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，一个地址寄存器 SIMA 以及一个数据寄存器 SIMD。注意，只有在合理设置 SIMC0 寄存器中的 UMD 位和 SIM2~SIM0 位选择 I²C 模式后，SIMC1、SIMD、SIMA 和 SIMTOC 寄存器以及它们的上电复位值才有效。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	UMD	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

I²C 数据寄存器

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。在单片机将数据写入到 I²C 总线之前，要传输的数据应先存在 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 SIMD 实现。

- **SIMD 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	x	x	x	x	x	x	x	x

“x”：未知

Bit 7~0 **D7~D0**：USIM SPI/I²C 数据寄存器位 bit 7~bit 0

I²C 地址寄存器

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 的主机发送出的地址和寄存器 SIMA 中存储的地址相符，那么就选中了这个从机。

- **SIMA 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	0	0	0	0	0	0	0	0

Bit 7~1 **SIMA6~SIMA0**：I^2C 从机地址位
SIMA6~SIMA0 是从机地址 bit 6~bit 0。

Bit 0 **D0**：保留位，此位可通过软件程序进行读写。

I^2C 控制寄存器

单片机中有三个控制 I^2C 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 包括多个用于指示 I^2C 传输状态的相关标志位。SIMTOC 寄存器用于控制 I^2C 超时功能，此寄存器在 I^2C 超时一节介绍。

- **SIMC0 寄存器**

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

Bit 7~5 **SIM2~SIM0**：USIM SPI/I^2C 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：I^2C 从机模式
111：未使用模式
当 UMD 位清零时，这几位用于设置 USIM SPI/I^2C 功能的工作模式，除了选择 USIM 模块的 I^2C 或 SPI 功能，还可选择 SPI 的主从模式和 SPI 的主机时钟频率。SPI 时钟源可来自于系统时钟和 f_{SUB} 也可以选择来自 PTM。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 **UMD**：UART 模式选择位
0：SPI 或 I^2C 模式
1：UART 模式
此位为 UART 模式选择位。当此位清零时，实际 SPI 或 I^2C 模式是通过 SIM2~SIM0 位选择。当选择 SPI 或 I^2C 模式时，此位必须清零。

Bit 3~2 **SIMDEB1~SIMDEB0**：I^2C 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
当设置 UMD 位为“0”、SIM2~SIM0 位为“110”将 USIM 设置为 I^2C 接口功能时，这两个位用于选择 I^2C 去抖时间。

Bit 1 **SIMEN**：USIM SPI/I^2C 控制位
0：除能
1：使能
此位为 USIM SPI/I^2C 接口的开 / 关控制位。此位为“0”时，USIM SPI/I^2C 接口除能，SDI、SDO、SCK 和 $\overline{SCS}$ 或 SDA 和 SCL 脚将失去 SPI 或 I^2C 功能，USIM 工作电流减小到最小值。此位为“1”时，USIM SPI/I^2C 接口使能。若

USIM 经由 UMD 位和 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 USIM 经由 UMD 位和 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**：USIM SPI 未完成标志位
此位仅当 USIM 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	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 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的地址相同。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，此位变为高电平。当检测到 STOP 信号时 I²C 总线空闲，该位变为低电平。

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

Bit 3 **TXAK**：I²C 总线发送应答标志位
0：从机发送应答标志
1：从机没有发送应答标志
从机接收完 8 位数据之后，该位将在第九个从机时钟时被传到总线上。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 / 写位
0：从机应处于接收模式
1：从机应处于发送模式
SRW 位是从机读写位。决定主机是否希望传输数据或接收来自 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“0”时，主机往总线上写数据，从机处于接收模式以读取数据。

Bit 1 **IAMWU**：I²C 地址匹配唤醒控制位
0：除能
1：使能
此位设置为“1”则使能 I²C 地址匹配使系统从休眠或空闲模式中唤醒的功能。若进入休眠或空闲模式前 IAMWU 已经置高以使能 I²C 地址匹配唤醒功能，在系统唤醒后须软件清除此位以确保单片机正确地运行。

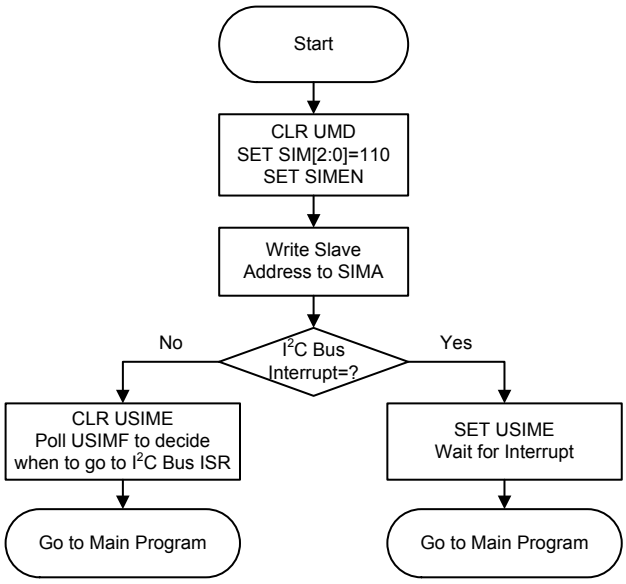
Bit 0 **RXAK**：I²C 总线接收应答标志位
0：从机接收到应答标志
1：从机没有接收到应答标志

RXAK 位是接收应答标志位。如果 RXAK 位为“0”，即表示 8 位数据传输之后，从机在第九个时钟有接受到一个应答信号。如果从机处于发送状态，从机作为发送方会检查 RXAK 位来判断主机接收方是否愿意继续接收下一个字节。因此发送方会一直发送数据，直到 RXAK 为“1”时才停止发送数据。这时，发送方将释放 SDA 线，主机方可发出停止信号从而释放 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 USIM 中断。进入中断服务程序后，系统要检测 HAAS 位和 SIMTOF 位，以判断中断源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 超时。在数据传递中，要注意的是，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 位以确定自己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 UMD 位为“0”、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 USIME 位以使能 USIM 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

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 USIM 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，从机随后发出一个低电平应答信号 (即第 9 位)。当从机地址匹配时，从机会将状态标志位 HAAS 置位。

USIM I²C 总线中断有三个中断源，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 USIM 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 超时。当是从机地址匹配发生中断时，则从机或是用于发送模式并将数据写进 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。

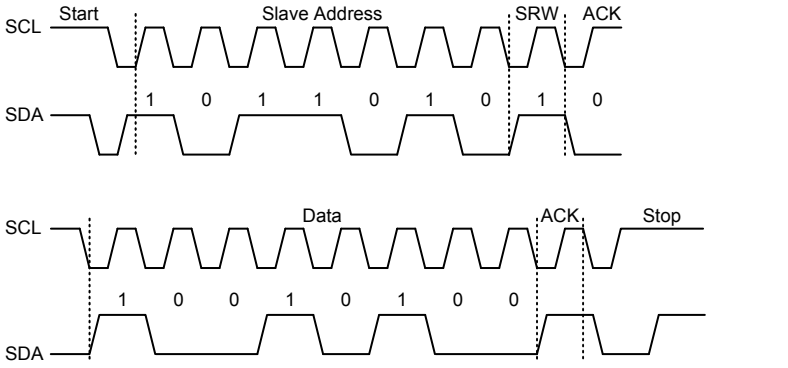
I²C 总线从机地址应答信号

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 总线数据和应答信号

在从机确认接收到从地址后，会进行 8 位宽度的数据传输。这个数据传输顺序是的高位在前，低位在后。接收方在接收到 8 位数据后必须发出一个应答信号 (“0”) 以继续接收下一个数据。如果从机发送方没接收到来自主机接收方的应答信号，发送方将释放 SDA 线，此时主机方可发出 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，如果从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。

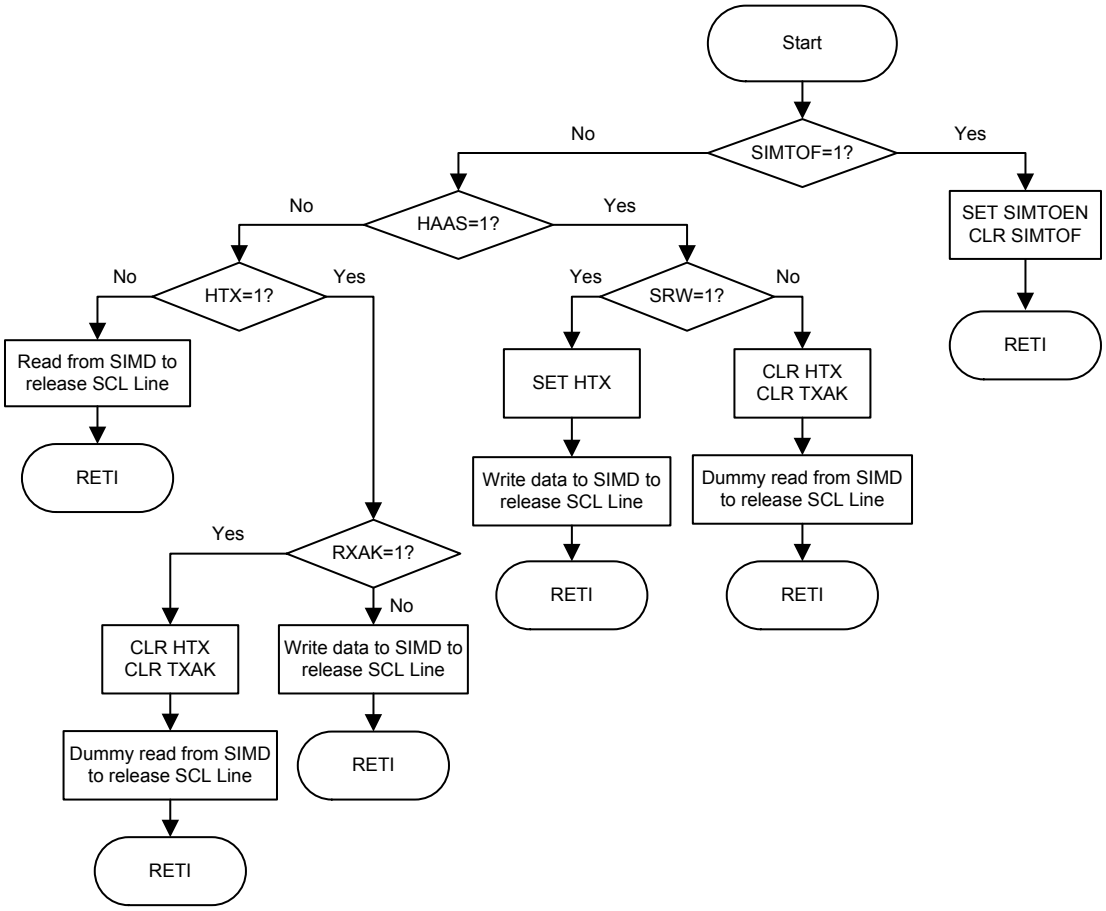


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

I^2C 通信时序图

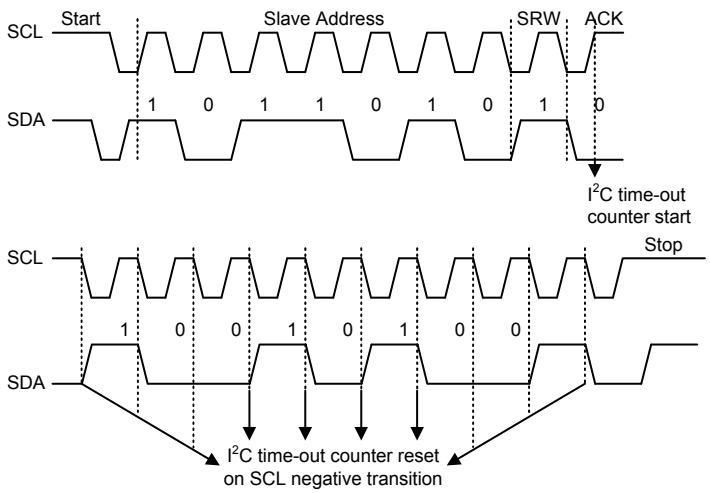
注：当从机地址匹配时，单片机必须选择设置为发送模式还是接收模式。若设置为发送模式，需写数据至 SIMD 寄存器；若设置为接收模式，需立即从 SIMD 寄存器中虚读数据以释放 SCL 线。



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 USIM 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 SIMTOSn 位进行选择。超时周期可通过公式计算：((1~64)×(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**：USIM I²C 超时控制位
0：除能
1：使能

Bit 6 **SIMTOF**：USIM I²C 超时标志位
0：超时未发生
1：超时发生
当发生超时，此位由硬件自动置位；此位必须通过应用程序清零。

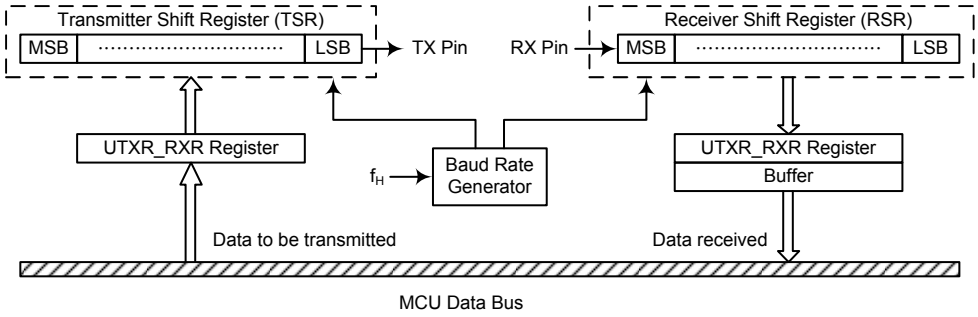
Bit 5~0 **SIMTOS5~SIMTOS0**：USIM I²C 超时时间选择位
I²C 超时时钟源是 f_{SUB}/32。
I²C 超时时间计算方法：([SIMTOS[5:0]+1)×(32/f_{SUB})。

UART 接口

该单片机具有一个全双工的异步串行通信接口 –UART，可以很方便的与其它具有串行口的芯片通信。UART 具有许多功能特性，发送或接收串行数据时，将数据组成一个 8 位或 9 位的数据块，连同数据特征位一并传输。具有检测数据覆盖或帧错误等功能。UART 功能与 SPI 和 I^2C 接口共用一个内部中断向量，当接收到数据或数据发送结束，触发中断。

内置的 UART 功能包含以下特性：

- 全双工通用异步接收器 / 发送器
- 8 位或 9 位传输格式
- 奇校验、偶校验或无校验
- 1 位或 2 位停止位
- 8 位预分频的波特率发生器
- 奇偶、帧、噪声和溢出检测
- 支持地址匹配中断 (最后一位 = 1)
- 独立的发送和接收使能
- 2-byte FIFO 接收缓冲器
- RX 引脚唤醒功能
- 发送和接收中断
- 中断可由下列条件初始化：
  - ♦ 发送器为空
  - ♦ 发送器空闲
  - ♦ 接收完成
  - ♦ 接收器溢出
  - ♦ 地址匹配



UART 数据传输方框图

UART 外部引脚

内部 UART 有两个外部引脚 TX 和 RX，可与外部串行接口进行通信。TX 和 RX 分别为 UART 发送脚和接收脚，与 I/O 口或其它功能共用引脚。在使用 UART 功能前，应先通过相应的引脚共用功能选择寄存器，选择 TX 和 RX 引脚功能。当 UMD、UREN、UTXEN 和 URXEN 位置高时，将自动设置这些 I/O 脚或其它共用功能脚作为 TX 输出和 RX 输入，并且除能 TX 和 RX 引脚上的上拉电阻功能。当 UMD、UREN、UTXEN 或 URXEN 位清零除能 TX 或 RX 引脚功能后，TX 或 RX 引脚将处于浮空状态。这时 TX 或 RX 引脚是否连接内部上拉电阻是由相应的 I/O 上拉电阻控制位决定的。

UART 数据传输方案

前面方框图显示了 UART 的整体结构。需要发送的数据首先写入 UTXR_RXR 寄存器，接着此数据被传输到发送移位寄存器 TSR 中，然后在波特率发生器的控制下将 TSR 寄存器中数据一位位地移到 TX 引脚上，低位在前。UTXR_RXR 寄存器被映射到单片机的数据存储器中，而发送移位寄存器没有实际地址，所以发送移位寄存器不可直接操作。

数据在波特率发生器的控制下，低位在前高位在后，从外部引脚 RX 进入接收移位寄存器 RSR。当数据接收完成，数据从接收移位寄存器移入可被用户程序操作的 UTXR_RXR 寄存器中。UTXR_RXR 寄存器被映射到单片机数据存储器中，而接收移位寄存器没有实际地址，所以接收移位寄存器不可直接操作。

需要注意的是，发送和接收都是共用同一个地址的数据寄存器，即 UTXR_RXR 寄存器。

UART 状态和控制寄存器

与 UART 功能相关的有六个寄存器，SIMC0 寄存器中的 UMD 位用于选择 UART 模式。其它包括控制 UART 模块整体功能的 UUSR、UUCR1 和 UUCR2 寄存器，控制波特率的 UBRG 寄存器，管理发送和接收数据的数据寄存器 UTXR_RXR。注意，只有在 SIMC0 寄存器中的 UMD 位设置为“1”后，UART 相关的寄存器以及它们的上电复位值才有效。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM2	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
UUSR	UPERR	UNF	UFERR	UOERR	URIDLE	URXIF	UTIDLE	UTXIF
UUCR1	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
UUCR2	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
UTXR_RXR	UTXRX7	UTXRX6	UTXRX5	UTXRX4	UTXRX3	UTXRX2	UTXRX1	UTXRX0
UBRG	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0

UART 寄存器列表

- **SIMC0 寄存器**

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

Bit 7~5 **SIM2~SIM0**：USIM SPI/I²C 工作模式控制位
当 UMD 位清零时，这几位用于设置 USIM SPI/I²C 功能的工作模式。更多细节详见 SPI 或 I²C 寄存器章节。

Bit 4 **UMD**：UART 模式选择位
0：SPI 或 I²C 模式
1：UART 模式
此位为 UART 模式选择位。当此位清零时，实际 SPI 或 I²C 模式是通过 SIM2~SIM0 位选择。当选择 SPI 或 I²C 模式时，此位必须清零。

Bit 3~2 **SIMDEB1~SIMDEB0**：I²C 去抖时间选择位
详见 I²C 寄存器章节。

Bit 1 **SIMEN**：USIM SPI/I²C 控制位
0：除能
1：使能
此位仅当 UMD 位设置为“1”选择 SPI 或 I²C 模式时才有效。详见 SPI 或 I²C 寄存器章节。

Bit 0 **SIMICF**：USIM SPI 未完成标志位
详见 SPI 寄存器章节。

- **UUSR 寄存器**

寄存器 UUSR 是 UART 的状态寄存器，可以通过程序读取以得知当前 UART 状态。所有 UUSR 位是只读的。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	UPERR	UNF	UFERR	UOERR	URIDLE	URXIF	UTIDLE	UTXIF
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

Bit 7 **UPERR**：奇偶校验出错标志位
0：奇偶校验正确
1：奇偶校验出错
UPERR 是奇偶校验出错标志位。若 UPERR=0，奇偶校验正确；若 UPERR=1，接收到的数据奇偶校验出错。只有使能了奇偶校验此位才有效。可使用软件清除该标志位，即先读取 UUSR 寄存器再读 UTXR_RXR 寄存器来清除此位。

Bit 6 **UNF**：噪声干扰标志位
0：没有受到噪声干扰
1：受到噪声干扰
UNF 是噪声干扰标志位。若 UNF=0，没有受到噪声干扰；若 UNF=1，UART 接收数据时受到噪声干扰。它与 URXIF 在同周期内置位，但不会与溢出标志位同时置位。可使用软件清除该标志位，即先读取 UUSR 寄存器再读 UTXR_RXR 寄存器将清除此标志位。

Bit 5 **UFERR**：帧错误标志位
0：无帧错误发生
1：有帧错误发生
UFREE 是帧错误标志位。若 UFREE=0，没有帧错误发生；若 UFREE=1，当前的数据发生了帧错误。可使用软件清除该标志位，即先读取 UUSR 寄存器再读 UTXR_RXR 寄存器来清除此位。

Bit 4 **UOERR**：溢出错误标志位
0：无溢出错误发生
1：有溢出错误发生
UOERR 是溢出错误标志位，表示接收缓冲器是否溢出。若 UOERR=0，没有溢出错误；若 UOERR=1，发生了溢出错误，它将禁止下一组数据的接收。可通过软件清除该标志位，即先读取 UUSR 寄存器再读 UTXR_RXR 寄存器将清除此标志位。

Bit 3 **URIDLE**：接收状态标志位
0：正在接收数据
1：接收器空闲
URIDLE 是接收状态标志位。若 URIDLE=0，正在接收数据；若 URIDLE=1，接收器空闲。在接收到停止位和下一个数据的起始位之间，URIDLE 被置位，表明 UART 空闲，RX 脚处于逻辑高状态。

Bit 2 **URXIF**：接收寄存器状态标志位
0：UTXR_RXR 寄存器为空
1：UTXR_RXR 寄存器含有有效数据
URXIF 是接收寄存器状态标志位。当 URXIF=0，UTXR_RXR 寄存器为空；当 URXIF=1，UTXR_RXR 寄存器接收到新数据。当数据从移位寄存器加载到

UTXR_RXR 寄存器中，如果 UUCR2 寄存器中的 URIE=1，则会触发中断。当接收数据时检测到一个或多个错误时，相应的标志位 UNF、UFERR 或 UPERR 会在同一周期内置位。读取 UUSR 寄存器再读 UTXR_RXR 寄存器，如果 UTXR_RXR 寄存器中没有新的数据，那么将清除 URXIF 标志。

Bit 1 **UTIDLE**：数据发送完成标志位
0：数据传输中
1：无数据传输

UTIDLE 是数据发送完成标志位。若 UTIDLE=0，数据传输中。当 UTXIF=1 且数据发送完毕或者暂停字被发送时，UTIDLE 置位。UTIDLE=1，TX 引脚空闲且处于逻辑高状态。读取 UUSR 寄存器再写 UTXR_RXR 寄存器将清除 UTIDLE 位。数据字符或暂停字就绪时，不会产生该标志位。

Bit 0 **UTXIF**：发送数据寄存器 UTXR_RXR 状态位
0：数据还没有从缓冲器加载到移位寄存器中
1：数据已从缓冲器加载到移位寄存器中 (UTXR_RXR 数据寄存器为空)

UTXIF 是发送数据寄存器为空标志位。若 UTXIF=0，数据还没有从缓冲器加载到移位寄存器中；若 UTXIF=1，数据已从缓冲器中加载到移位寄存器中。读取 UUSR 存器再写 UTXR_RXR 寄存器将清除 UTXIF。当 UTXEN 被置位，由于发送缓冲器未满，UTXIF 也会被置位。

- **UUCR1 寄存器**

UUCR1 和 UUCR2 是 UART 的两个控制寄存器，用来定义各种 UART 功能，例如 UART 的使能与除能、奇偶校验控制和传输数据的长度等等。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

“x”：未知

Bit 7 **UREN**：UART 功能使能位
0：UART 除能，TX 和 RX 脚处于浮空状态
1：UART 使能，TX 和 RX 脚作为 UART 功能引脚

此位为 UART 的使能位。UREN=0，UART 除能，RX 和 TX 处于浮空状态；UREN=1，若 UMD 位置高，UART 使能，TX 和 RX 将分别由 UTXEN 和 URXEN 控制。当 UART 被除能将清除缓冲器，所有缓冲器中的数据将被忽略，另外波特率计数器、错误和状态标志位被复位，UTXEN、URXEN、UTXBRK、URXIF、UOERR、UFERR、UPERR 和 UNF 清零，而 UTIDLE、UTXIF 和 URIDLE 置位，UUCR1、UUCR2 和 UBRG 寄存器中的其它位保持不变。若 UART 工作时 UREN 清零，所有发送和接收将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

Bit 6 **UBNO**：发送数据位数选择位
0：8-bit 传输数据
1：9-bit 传输数据

UBNO 是发送数据位数选择位。UBNO=1，传输数据为 9 位；UBNO=0，传输数据为 8 位。若选择了 9 位数据传输格式，URX8 和 UTX8 将分别存储接收和发送数据的第 9 位。

Bit 5 **UPREN**：奇偶校验使能位
0：奇偶校验除能
1：奇偶校验使能

此位为奇偶校验使能位。UPREN=1，使能奇偶校验；UPREN=0，除能奇偶校验。

Bit 4 **UPRT**：奇偶校验选择位
0：偶校验
1：奇校验
奇偶校验选择位。UPRT=1，奇校验；UPRT=0，偶校验。

Bit 3 **USTOPS**：停止位的长度选择位
0：有一位停止位
1：有两位停止位
此位用来设置停止位的长度。USTOP=1，有两位停止位；USTOP=0，只有一位停止位。

Bit 2 **UTXBRK**：暂停字发送控制位
0：没有暂停字要发送
1：发送暂停字
UTXBRK 是暂停字发送控制位。UTXBRK=0，没有暂停字要发送，TX 引脚正常操作；UTXBRK=1，将会发送暂停字，发送器将发送逻辑“0”。若 UTXBRK 为高，缓冲器中数据发送完毕后，发送器输出将至少保持 13 位宽的低电平直至 UTXBRK 复位。

Bit 1 **URX8**：接收 9-bit 数据传输格式中的第 9 位 (只读)
此位只有在传输数据为 9 位的格式中有效，用来存储接收数据的第 9 位。UBNO 是用来控制传输位数是 8 位还是 9 位。

Bit 0 **UTX8**：发送 9-bit 数据传输格式中的第 9 位 (只写)
此位只有在传输数据为 9 位的格式中有效，用来存储发送数据的第 9 位。UBNO 是用来控制传输位数是 8 位还是 9 位。

- **UUCR2 寄存器**

UUCR2 是 UART 的第二个控制寄存器，它的主要功能是控制发送器、接收器以及各种 USIM UART 模式中断源的使能或除能。它也可用来控制波特率，使能接收唤醒和地址侦测。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
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 **UTXEN**：UART 发送使能位
0：UART 发送除能
1：UART 发送使能
此位为发送使能位。UTXEN=0，发送将被除能，发送器立刻停止工作。另外发送缓冲器将被复位，此时 TX 引脚将处于浮空状态。若 UTXEN=1 且 UMD=1 及 UREN=1，则发送将被使能，TX 引脚将由 UART 来控制。在数据传输时清除 UTXEN 将中止数据发送且复位发送器，此时 TX 引脚将处于浮空状态。

Bit 6 **URXEN**：UART 接收使能位
0：UART 接收除能
1：UART 接收使能
此位为接收使能位。URXEN=0，接收将被除能，接收器立刻停止工作。另外接收缓冲器将被复位，此时 RX 引脚将处于浮空状态。若 URXEN=1 且 UMD=1 及 UREN=1，则接收将被使能，RX 引脚将由 UART 来控制。在数据传输时清除 URXEN 将中止数据接收且复位接收器，此时 RX 引脚将处于浮空状态。

Bit 5 **UBRGH**：波特率发生器高低速选择位
0：低速波特率
1：高速波特率
此位为波特率发生器高低速选择位，它和 UBRG 寄存器一起控制 UART 的波特率。UBRGH=1，为高速模式；UBRGH=0，为低速模式。

Bit 4 **UADDEN**：地址检测使能位
0：地址检测除能
1：地址检测使能
此位为地址检测使能和除能位。UADDEN=1，地址检测使能，此时数据的第 8 位 (UBNO=0) 或第 9 位 (UBNO=1) 为高，那么接到的是地址而非数据。若相应的中断使能且接收到的值最高位为 1，那么中断请求标志将会被置位，若地址检测功能使能且最高位为 0，那么将不会产生中断且收到的数据也会被忽略。

Bit 3 **UWAKE**：RX 脚下降沿唤醒 UART 功能使能位
0：RX 脚下降沿唤醒 UART 功能除能
1：RX 脚下降沿唤醒 UART 功能使能
此位用于控制 RX 引脚下降沿时是否唤醒 UART 功能。此位仅当 UART 时钟源 f_H 关闭时有效。若 UART 时钟源 f_H 还开启，则无 RX 引脚唤醒 UART 功能无效。若此位置高且 UART 时钟 f_H 关闭，当 RX 引脚发生下降沿时会产生 UART 唤醒请求。若相应的中断使能，将产生 RX 引脚唤醒 UART 的中断，以告知单片机使其通过应用程序开启 UART 时钟源 f_H，从而唤醒 UART 功能。否则，若此位为低，即使 RX 引脚发生下降沿也无法恢复 UART 功能。

Bit 2 **URIE**：接收中断使能位
0：接收中断除能
1：接收中断使能
此位为接收中断使能或除能位。若 URIE=1，当 UOERR 或 URXIF 置位时，USIM 的中断请求标志 USIMF 置位；若 URIE=0，USIM 中断请求标志 USIMF 不受 UOERR 和 URXIF 影响。

Bit 1 **UTIIE**：发送器空闲中断使能位
0：发送器空闲中断除能
1：发送器空闲中断使能
此位为发送器空闲中断的使能或除能位。若 UTIIE=1，当发送器空闲触发 UTIDLE 置位时，USIM 的中断请求标志 USIMF 置位；若 UTIIE=0，USIM 中断请求标志 USIMF 不受 UTIDLE 的影响。

Bit 0 **UTEIE**：发送寄存器为空中断使能位
0：发送寄存器为空中断除能
1：发送寄存器为空中断使能
此位为发送寄存器为空中断的使能或除能位。若 UTEIE=1，当发送器为空触发 UTXIF 置位时，USIM 的中断请求标志 USIMF 置位；若 UTEIE=0，USIM 中断请求标志 USIMF 不受 UTXIF 的影响。

- **UTXR_RXR 寄存器**

UTXR_RXR 是一个数据寄存器，用来存储 TX 引脚将要发送或 RX 引脚正在接收的数据。

Bit	7	6	5	4	3	2	1	0
Name	UTXRX7	UTXRX6	UTXRX5	UTXRX4	UTXRX3	UTXRX2	UTXRX1	UTXRX0
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 7~0 **UTXRX7~UTXRX0**：UART 发送 / 接收数据位 Bit 7~Bit 0

- **UBRG 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0
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 7~0 **UBRG7~UBRG0**：波特率值

软件设置 UUCR2 寄存器中的 UBRGH 位 (设置波特率发生器的速度) 和 UBRG 寄存器 (设置波特率的值)，一起控制 UART 的波特率。

注：若 UBRGH = 0，波特率 = $f_H/[64\times(N+1)]$；

若 UBRGH = 1，波特率 = $f_H/[16\times(N+1)]$。

波特率发生器

UART 自身具有一个波特率发生器，通过它可以设定数据传输速率。波特率是由一个独立的内部 8 位计数器产生，它由 UBRG 寄存器和 UUCR2 寄存器的 UBRGH 位来控制。UBRGH 是决定波特率发生器处于高速模式还是低速模式，从而决定计算公式的选用。UBRG 寄存器的值 N 可根据下表中的公式计算，N 的范围是 0 到 255。

UUCR2 的 UBRGH 位	0	1
波特率 (BR)	$f_H/[64\ (N+1)]$	$f_H/[16\ (N+1)]$

为得到相应的波特率，首先需要设置 UBRGH 来选择相应的计算公式从而算出 UBRG 的值。由于 UBRG 的值不连续，所以实际波特率和理论值之间有一个偏差。

下面举例怎样计算 UBRG 寄存器中的值 N 和误差。

波特率和误差的计算

若选用 4MHz 时钟频率且 UBRGH=0，若期望的波特率为 4800，计算它的 UBRG 寄存器的值 N，实际波特率和误差。

根据上表，波特率 $BR=f_H/[64\ (N+1)]$

转换后的公式 $N=[f_H/(BR\times64)] - 1$

带入参数 $N=[4000000/(4800\times64)] - 1=12.0208$

取最接近的值，十进制 12 写入 UBRG 寄存器，实际波特率如下

$BR=4000000/[64\times(12+1)]=4808$

因此，误差 = (4808 - 4800)/4800=0.16%

UART 模块的设置与控制

UART 采用标准的不归零码传输数据，这种方法通常被称为 NRZ 法。它由 1 位起始位，8 位或 9 位数据位和 1 位或者两位停止位组成。奇偶校验是由硬件自动完成的，可设置成奇校验、偶校验和无校验三种格式。常用的数据传输格式由 8 位数据位，1 位停止位，无校验组成，用 8、N、1 表示，它是系统上电的默认格式。数据位数、停止位数和奇偶校验由 UUCR1 寄存器的 UBNO、UPRT、UPREN 和 USTOPS 设定。用于数据发送和接收的波特率由一个内部的 8 位波特率发送器产生，数据传输时低位在前高位在后。尽管 UART 发送器和接收器在功能上相互独立，但它们使用相同的数据传输格式和波特率，在任何情况下，停止位是必须的。

UART 的使能和除能

UART 是由 UUCR1 寄存器的 UREN 位来使能和除能的。当 SIMC0 寄存器中的 UMD 位已设置为“1”选择 UART 模式，若 UREN、UTXEN 和 URXEN 都为高，则 TX 和 RX 分别为 UART 的发送端口和接收端口。若没有数据发送，TX 引脚默认状态为高电平。

UREN 清零将除能 TX 和 RX，通过设置相关引脚共用控制位，这两个引脚可用作普通 I/O 口或其它引脚共用功能。当 UART 被除能时将清空缓冲器，所有缓冲器中的数据将被忽略，另外一些使能控制、错误标志和状态标志将被复位，如 UTXEN、URXEN、UTXBRK、URXIF、UOERR、UFERR、UPERR 和 UNF 清零，而 UTIDLE、UTXIF 和 URIDLE 置位，UUCR1、UUCR2 和 UBRG 寄存器中的其它位保持不变。若 UART 工作时 UREN 清零，所有发送和接收将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

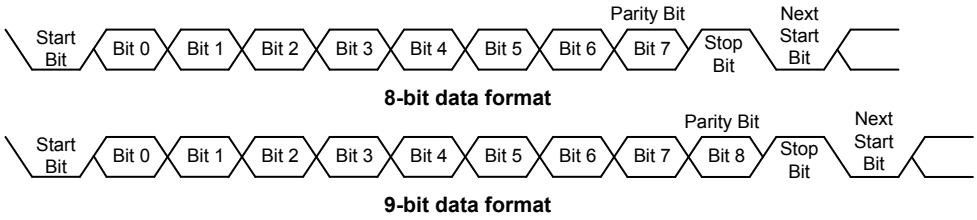
数据位、停止位位数以及奇偶校验的选择

数据传输格式由数据长度、是否校验、校验类型、地址位以及停止位长度组成。它们都是由 UUCR1 寄存器的各个位控制的。UBNO 决定数据传输是 8 位还是 9 位；UPRT 决定校验类型；UPREN 决定是否选择奇偶校验；而 USTOPS 决定选用 1 位还是 2 位停止位。下表列出了各种数据传输格式。若地址检测功能使能，地址位，即数据字节的最高位，用来确定此帧是地址还是数据。停止位的长度和数据位的长度无关，且只有发送器需设置停止位长度。接收器只接收一个停止位。

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

发送和接收数据格式

下图是传输 8 位和 9 位数据的波形。



UART 发送器

UUCR1 寄存器的 UBNO 位是控制数据传输的长度。UBNO=1 其长度为 9 位，第 9 位 MSB 存储在 UUCR1 寄存器的 UTX8 中。发送器的核心是发送移位寄存器 TSR，它的数据由发送寄存器 UTXR_RXR 提供，应用程序只须将发送数据写入 UTXR_RXR 寄存器。上组数据的停止位发出前，TSR 寄存器禁止写入。如果还有新的数据要发送，一旦停止位发出，待发数据将会从 UTXR_RXR 寄

存器加载到 TSR 寄存器。TSR 不像其它寄存器一样映射到数据存储器，所以应用程序不能对其进行读写操作。UTXEN=1，发送使能，但若 UTXR_RXR 寄存器没有数据或者波特率没有设置，发送器将不会工作。先写 UTXR_RXR 寄存器再置高 UTXEN 也会触发发送。当发送器使能，若 TSR 寄存器为空，数据写入 UTXR_RXR 寄存器将会直接加载到 TSR 寄存器中。发送器工作时，UTXEN 清零，发送器将立刻停止工作并且复位，此时通过设置相关引脚共用控制位，TX 引脚用作普通 I/O 口或其它引脚共用功能。

发送数据

当 UART 发送数据时，数据从移位寄存器中移到 TX 引脚上，其低位在前高位在后。在发送模式中，UTXR_RXR 寄存器在内部总线和发送移位寄存器间形成一个缓冲。如果选择 9 位数据传输格式，最高位 MSB 取自 UUCR1 寄存器的 UTX8。

发送器初始化可由如下步骤完成：

- 正确地设置 UBNO、UPRT、UPREN 和 USTOPS 位以确定数据长度、校验类型和停止位长度。
- 设置 UBRG 寄存器，选择期望的波特率。
- 置高 UTXEN，使能 UART 发送器且使 TX 作为 UART 的发送端。
- 读取 UUSR 寄存器，然后将待发数据写入 UTXR_RXR 寄存器。注意，此步骤会清除 UTXIF 标志位。

如果要发送多个数据只需重复上一步骤。

当 UTXIF=0 时，数据将禁止写入 UTXR_RXR 寄存器。可以通过以下步骤来清除 UTXIF：

1. 读取 UUSR 寄存器

2. 写 UTXR_RXR 寄存器

只读标志位 UTXIF 由 UART 硬件置位。若 UTXIF=1，UTXR_RXR 寄存器为空，其它数据可以写入而不会覆盖之前的数据。若 UTEIE=1，UTXIF 标志位会产生中断。在数据传输时，写 UTXR_RXR 指令会将待发数据暂存在 UTXR_RXR 寄存器中，当前数据发送完毕后，待发数据被加载到发送移位寄存器中。当发送器空闲时，写 UTXR_RXR 指令会将数据直接加载到 TSR 寄存器中，数据传输立刻开始且 UTXIF 置位。当发送完停止位或暂停帧后，表示一帧数据已发送完毕，此时 UTIDLE 位将被置位。

可以通过以下步骤来清除 UTIDLE：

1. 读取 UUSR 寄存器

2. 写 UTXR_RXR 寄存器

清除 UTXIF 和 UTIDLE 软件执行次序相同。

发送暂停字

若 UTXBRK=1，下一帧将会发送暂停字。它是由一个起始位、13×N (N=1，2……) 位逻辑 0 组成。置位 UTXBRK 将会发送暂停字，而清除 UTXBRK 将产生停止位，传输暂停字不会产生中断。需要注意的是，暂停字至少 13 位宽。若 UTXBRK 持续为高，那么发送器会一直发送暂停字；当应用程序将 UTXBRK 清零后，发送器结束最后一帧暂停字的发送后接着发送一位或两位停止位。最后一帧暂停字的结尾自动为高电平，以确保下一帧数据起始位的检测。

UART 接收器

UART 接收器支持 8 位或者 9 位数据接收。若 UBNO=1，数据长度为 9 位，而最高位 MSB 存放在 UUCR1 寄存器的 URX8 中。接收器的核心是串行移位寄存器 RSR。RX 引脚上的数据送入数据恢复器中，它在 16 倍波特率的频率下工作，而串行移位器工作在正常波特率下。当在 RX 引脚上检测到停止位，若 UTXR_RXR 寄存器为空，数据从 RSR 寄存器中加载到 UTXR_RXR 寄存器。RX 引脚上的每一位数据会被采样三次以判断其逻辑状态。RSR 不像其它寄存器一样映射在数据存储器，所以应用程序不能对其进行读写操作。

接收数据

当 UART 接收数据时，数据低位在前高位在后，连续地从 RX 引脚进入移位寄存器。UTXR_RXR 寄存器在内部总线和接收移位寄存器间形成一个缓冲。UTXR_RXR 寄存器是一个两层的 FIFO 缓冲器，它能保存两帧数据的同时接收第三帧数据，应用程序必须保证在接收完第三帧前读取 UTXR_RXR 寄存器，否则忽略第三帧数据并且发生溢出错误。

接收器的初始化可由如下步骤完成：

- 正确地设置 UBNO、UPRT 和 UPREN 位以确定数据长度和校验类型。
- 设置 UBRG 寄存器，选择期望的波特率。
- 置高 URXEN，使能 UART 发送器且使 RX 作为 UART 的接收端。

此时接收器被使能并检测起始位。

接收数据将会发生如下事件：

- 当 UTXR_RXR 寄存器中包含有效数据时，UUSR 寄存器中的 URXIF 位将会置位，溢出错误发生之前至多还有一帧数据可读。
- 若 URIE=1，数据从 RSR 寄存器加载到 UTXR_RXR 寄存器中将产生中断。
- 若接收器检测到帧错误、噪声干扰错误、奇偶出错或溢出错误，那么相应的错误标志位置位。

可以通过如下步骤来清除 URXIF：

1. 读取 UUSR 寄存器
2. 读取 UTXR_RXR 寄存器

接收暂停字

UART 接收任何暂停字都会当作帧错误处理。接收器只根据 UBNO 位的设置外加一个停止位来确定一帧数据的长度。若暂停字位数大于 UBNO 位指定的长度外加一个停止位，接收器认为接收已完毕，URXIF 和 UFERR 置位，UTXR_RXR 寄存器清 0，若相应的中断允许且 URIDLE 为高将会产生中断。暂停字只会被认为包含信息 0 且会置位 UFERR 标志位。如果检测到较长的暂停信号，接收器会将此信号视为包含一个起始位、数据位和无效的停止位的数据帧并且置位 UFERR 标志位。在下个开始位到来之前，接收器必须等待一个有效的停止位。接收器不会假定线上的暂停信号是下一个开始位。暂停字将会加载到缓冲器中，在接收到停止位前不会再接收数据，没有检测到停止位也会置位只读标志位 URIDLE。

UART 接收到暂停字会产生以下事件：

- 帧错误标志位 UFERR 置位。
- UTXR_RXR 寄存器清零。
- UOERR、UNF、UPERR、URIDLE 或 URXIF 可能会置位。

空闲状态

当 UART 接收数据时，即在起初位和停止位之间，UUSR 寄存器的接收状态标志位 URIDLE 清零。在停止位和下一帧数据的起始位之间，URIDLE 被置位，表示接收器空闲。

接收中断

UUSR 寄存器的只读标志位 URXIF 由接收器的边沿触发置位。若 URIE=1，数据从移位寄存器 RSR 加载到 UTXR_RXR 寄存器时产生中断，同样地，溢出也会产生中断。

接收错误处理

UART 会产生几种接收错误，下面部分将描述各错误以及怎样处理。

溢出 – UOERR 标志

UTXR_RXR 寄存器是一个两层的 FIFO 缓冲器，它能保存两帧数据的同时接收第三帧数据，应用程序必须保证在接收完第三帧前读取 UTXR_RXR 寄存器，否则发生溢出错误。

产生溢出错误时将会发生以下事件：

- UUSR 寄存器中 UOERR 被置位。
- UTXR_RXR 寄存器中数据不会丢失。
- RSR 寄存器数据将会被覆盖。
- 若 URIE=1，将会产生中断。

先读取 UUSR 寄存器再读取 UTXR_RXR 寄存器可将 UOERR 清零。

噪声干扰 – UNF 标志

数据恢复时多次采样可以有效的鉴别出噪声干扰。当检测到数据受到噪声干扰时将会发生以下事件：

- 在 URXIF 上升沿，UUSR 寄存器中只读标志位 UNF 置位。
- 数据从 RSR 寄存器加载到 UTXR_RXR 寄存器中。
- 不产生中断，但此位置位发生在 URXIF 置位产生中断的同周期内。

先读取 UUSR 寄存器再读取 UTXR_RXR 寄存器可将 UNF 清零。

帧错误 – UFERR 标志

若在停止位上检测到 0，UUSR 寄存器中只读标志 UFERR 置位。若选择两位停止位，此两位都必须为高，否则将置位 UFERR。此标志位同接收的数据分别记录在 UUSR 寄存器和 UTXR_RXR 寄存器中，此标志位可被任何复位清零。

奇偶校验错误 – UPERR 标志

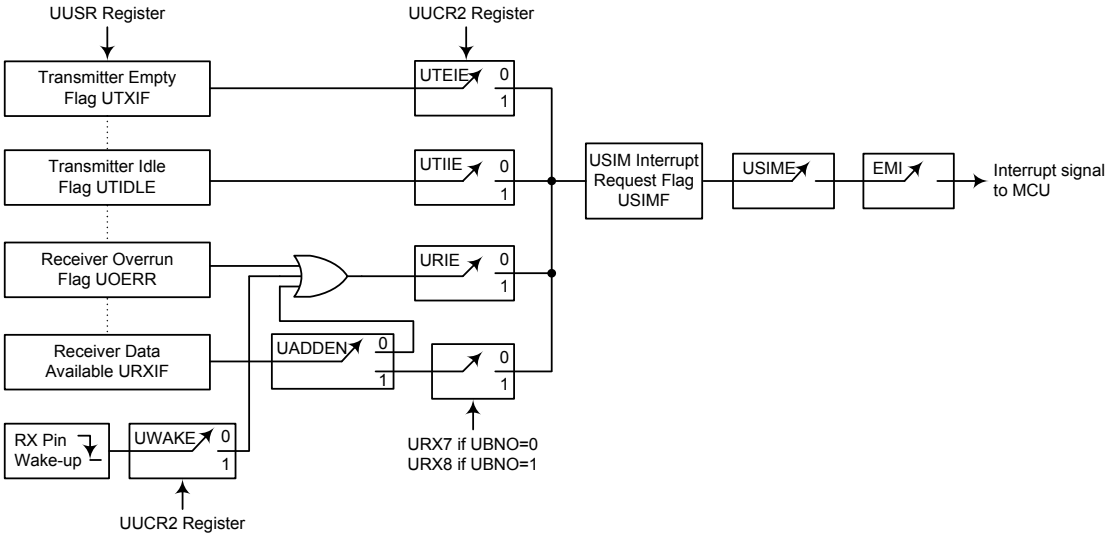
若接收到数据出现奇偶校验错误，UUSR 寄存器中只读标志 UPERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。此标志位同接收的数据分别记录在 UUSR 寄存器和 UTXR_RXR 寄存器中，此标志位可被任何复位清零。注意，在读取相应的数据之前必须先访问 UUSR 寄存器中的 UFERR 和 UPERR 错误标志位。

UART 模块中断结构

几个独立的 UART 条件可以产生一个 USIM 中断。当条件满足时，会产生一个低脉冲信号。发送寄存器为空、发送器空闲、接收器数据有效、溢出和地址检测和 RX 引脚唤醒都会产生中断。若总中断使能、USIM 中断允许且堆栈未满，程序将会跳转到相应的中断向量执行中断服务程序，而后再返回主程序。其中四种情况，若其 UUCR2 寄存器中相应中断允许位被置位，则 UUSR 寄存器中对应中断标志位将产生 USIM 中断。发送器相关的两个中断情况有各自对应的中断允许位，而接收器相关的两个中断情况共用一个中断允许位。这些允许位可用于禁止个别的 USIM UART 模式中断源。

地址检测也是 USIM UART 模式的中断源，它没有相应的标志位，若 UUCR2 寄存器中 UADDEN=1，当检测到地址将会产生 USIM 中断。RX 引脚唤醒也可以产生 USIM 中断，它没有相应的标志位，当 UART 时钟源 f_H 关闭且 UUCR2 中的 UWAKE 和 URIE 位被置位，RX 引脚上有下降沿时会产生 USIM 中断。

注意，UUSR 寄存器标志位为只读状态，软件不能对其进行设置，和其它一些中断一样，在进入相应中断服务程序时也不能清除这些标志位。这些标志位仅在 UART 特定动作发生时才会自动被清除，详细解释见 UART 寄存器章节。整体 UART 中断的使能或除能可由 USIM 中断控制寄存器中的相关中断使能控制位控制，其中断请求由 UART 模块决定。



UART 中断框图

地址检测模式

置位 UUCR2 寄存器中的 UADDEN 将启动地址检测模式。若此位为“1”，可产生接收数据有效中断，其请求标志位为 URXIF。若 UADDEN 有效，只有在接收到数据最高位为 1 才会产生中断，中断允许位 USIME 和 EMI 也要使能才会产生中断。地址的最高位为第 9 位 (UBNO=1) 或第 8 位 (UBNO=0)，若此位为高，则接收到的是地址而非数据。只有接收的数据的最后一位为高才会产生中断。若 UADDEN 除能，每接收到一个有效数据便会置位 URXIF，而不用考虑数据的最后一位。地址检测和奇偶校验在功能上相互排斥，若地址检测模式使能，为了确保操作正确，必须将奇偶校验使能位清零以除能奇偶校验。

UADDEN	Bit 9 (UBNO=1) Bit 8 (UBNO=0)	产生 USIM 中断
0	0	√
	1	√
1	0	×
	1	√

UADDEN 位功能

UART 模块暂停和唤醒

UART 时钟 f_H 关闭后 UART 模块将停止运行。当传送数据时 UART 时钟 f_H 关闭，发送将停止直到 UART 模块时钟再次使能。同样地，当接收数据时单片机进入空闲或休眠模式，数据接收也会停止。当单片机进入空闲或休眠模式，UUSR、UUCR1、UUCR2、接收 / 发送寄存器以及 UBRG 寄存器都不会受到影响。建议在单片机进入空闲或休眠模式前先确保数据发送或接收已完成。

UART 功能中包括了 RX 引脚的唤醒功能，由 UUCR2 寄存器中 UWAKE 位控制。当 UART 时钟 f_H 关闭时，若 UWAKE 位与 UART 模式选择位 UMD、UART 允许位 UREN、接收器允许位 URXEN 和接收器中断允许位 URIE 都被置位，则 RX 引脚的下降沿可触发产生 RX 引脚唤醒 UART 的中断。唤醒后系统需延时一段时间才能正常工作，在此期间，RX 引脚上的任何数据将被忽略。

若要产生唤醒 UART 的中断，除了唤醒使能控制位和接收中断使能控制位需置位外，全局中断允许位 EMI 和 USIM 中断使能控制位 USIME 也必须置位；若这两控制位没有被置位，那么，可发生唤醒事件但不会产生中断。唤醒后系统需一定的延时才能正常工作，然后才会产生 USIM 中断。

触控按键功能

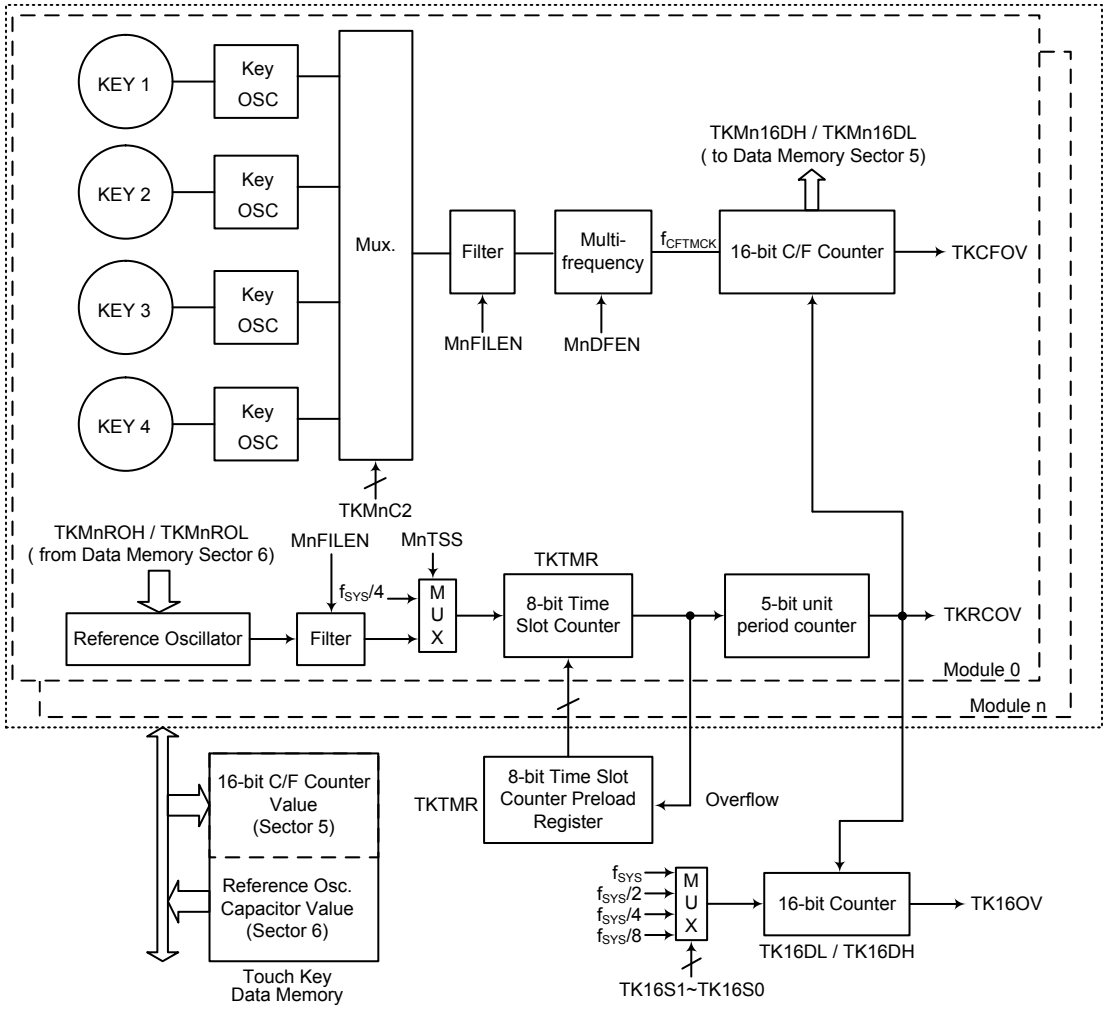
该系列单片机均提供多触控按键功能。触控按键功能完全内部集成不需外接元件，通过内部寄存器的简单操作即可实现此功能。

触控按键结构

触控按键与 I/O 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		触控按键
BS83B24C	24	Mn (n=0~5)	M0	KEY1~KEY4
			M1	KEY5~KEY8
			M2	KEY9~KEY12
			M3	KEY13~KEY16
			M4	KEY17~KEY20
			M5	KEY21~KEY24
BS83C40C	40	Mn (n=0~9)	M0	KEY1~KEY4
			M1	KEY5~KEY8
			M2	KEY9~KEY12
			M3	KEY13~KEY16
			M4	KEY17~KEY20
			M5	KEY21~KEY24
			M6	KEY25~KEY28
			M7	KEY29~KEY32
			M8	KEY33~KEY36
			M9	KEY37~KEY40

触控按键结构



注：1. 虚线中的结构适用于具有四个触控按键的触控按键模块。
2. 当 MnTSS=0 且 MnROEN=1 或当 MnTSS=1 时，触控按键功能 16-bit 计数器可正常工作。

触控按键功能方框图

触控按键寄存器

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，该系列单片机具有多达 10 个触控按键模块，取决于所选中的单片机型号。

寄存器名称	作用
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
TKMnC2	触控按键模块 n 控制寄存器 2

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	TKRAMC	TKRCOV	TKST	TKCFOV	TK16OV	—	TKMOD	TKBUSY
TKC1	D7	D6	D5	TSCS	TK16S1	TK16S0	TKFS1	TKFS0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
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	—	—	MnDFEN	MnFILEN	MnSOFC	MnSOF2	MnSOF1	MnSOF0
TKMnC1	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN
TKMnC2	MnSK31	MnSK30	MnSK21	MnSK20	MnSK11	MnSK10	MnSK01	MnSK00

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} 位时隙计数器时钟周期。

- **TKC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TKRAMC	TKRCOV	TKST	TKCFOV	TK16OV	—	TKMOD	TKBUSY
R/W	R/W	R/W	R/W	R/W	R/W	—	R/W	R
POR	0	0	0	0	0	—	0	0

Bit 7 **TKRAMC**：触控按键 RAM 存取控制位
0：由 MCU 读 / 写
1：由触控按键模块读 / 写
该位用于控制触控按键模块 RAM 是由 MCU 还是触控按键模块使用。但若触控按键模块工作在自动扫描模式 (当 TKMOD 位为 0 时，TKST 位由 0 转为 1 可进入自动扫描模式)，由触控按键模块可优先存取 RAM 内容。当结束自动扫描后 (例如 TKBUSY 位状态由 1 转为 0)，触控按键模块 RAM 存取才可通过 TKRAMC 位控制。因此建议当触控按键模块工作在自动扫描模式时，将 TKRAMC 位置为“1”。否则，若选择 MCU 存取 RAM，在自动扫描阶段，触控按键的 RAM 内容可能会被改动。

Bit 6 **TKRCOV**：触控按键时隙计数器溢出标志位
0：无溢出
1：溢出
此位可通过应用程序读 / 写。应注意该位不能通过应用程序置位，但必须通过应用程序清零。
在自动扫描模式时，如果由 TSCS 位选择的模块 0 或所有模块的时隙计数器溢出，但自动扫描还未完成，TKRCOV 位将不会被置位，同时所有触控按键模块 16 位 C/F 计数器、触控按键功能 16 位计数器、5 位时隙单位周期计数器将会自动清零，但 8 位时隙定时计数器将会从 8 位时隙定时计数器预置寄存器 (TKTMR 寄存器) 重新加载数据。当自动扫描工作结束，TKRCOV 位及触控按键中断请求标志位 TKMF 将被置位同时所有模块按键振荡器和参考振荡器自动停止。所有触控按键模块的 16 位 C/F 计数器、触控按键功能 16 位计数器、5 位时隙单位周期计数器和 8 位时隙定时计数器都会自动关闭。
在手动扫描模式时，如果由 TSCS 位选择的模块 0 或所有模块的时隙计数器溢出，TKRCOV 位及触控按键中断请求标志位 TKMF 将被置位同时所有模块按键振荡器和参考振荡器自动停止。所有触控按键模块的 16 位 C/F 计数器、触控按键功能 16 位计数器、5 位时隙单位周期计数器和 8 位时隙定时计数器也都会自动关闭。

Bit 5 **TKST**：触控按键检测开启控制位
0：检测停止或无操作
0 → 1：启动检测
当该位为“0”时，所有触控按键模块的 16 位 C/F 计数器、触控按键功能 16 位计数器和 5 位时隙单位周期计数器会自动清零但 8 位可编程时隙定时计数器不会被清零。当该位由 0 → 1 时，16 位 C/F 计数器、触控按键功能 16 位计数器、5 位时隙单位周期计数器和 8 位时隙定时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 位 C/F 计数器溢出标志位
0：无溢出
1：溢出
该位由触控按键模块 16 位 C/F 计数器溢出置位，必须通过应用程序清零。

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

Bit 2 未定义，读为“0”

Bit 1 **TKMOD**：触控按键扫描模式选择位
0：自动扫描模式
1：手动扫描模式
在手动扫描模式时，应在扫描开始之前，合理的设置参考振荡器电容值，并在扫描结束后通过应用程序读取触控按键模块 16 位 C/F 计数器值。
在自动扫描模式时，上面所说的数据的读取或写入是通过硬件完成。有一个专用的触控按键模块数据存储区可存放所有被扫描按键的参考振荡器电容值及 16 位 C/F 计数器值。直至扫描完所有计划扫描的按键后，自动扫描工作才会停止。

Bit 0 **TKBUSY**：触控按键扫描忙碌标志位
0：空闲 – 没有在执行按键扫描或按键扫描已完成
1：忙碌 – 正在扫描中
该位用于指示触控按键扫描工作是否在执行中。当 TKST 位置高启动扫描工作，该位被置“1”。
在自动扫描模式中，当触控按键扫描工作完成时，该位会自动清零。在手动扫描模式中，当触控时隙计数器溢出时，该位会自动清零。

- **TKC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	TSCS	TK16S1	TK16S0	TKFS1	TKFS0
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	1	1

Bit 7~5 **D7~D5**：测试数据位
这些位仅供内部测试使用，正常工作时，要保持 D7~D5 为“000”。

Bit 4 **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 3~2 **TK16S1~TK16S0**：触控按键功能 16 位计数器时钟选择位
00：f_{SYS}
01：$f_{SYS}/2$
10：$f_{SYS}/4$
11：$f_{SYS}/8$

Bit 1~0 **TKFS1~TKFS0**：触控按键振荡器和参考振荡器频率选择位
00：1MHz
01：3MHz
10：7MHz
11：11MHz

- **TK16DH/TK16DL – 触控按键功能 16-bit 计数器寄存器对**

寄存器	TK16DH								TK16DL							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

该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。在手动扫描模式中，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。在自动扫描模式中，该 16-bit 计数器内容将在时隙 0~2 结束时被清零但在时隙 3 结束时保持不变。当 TKST 位置低，该寄存器对将被清零。

- **TKMn16DH/TKMn16DL – 触控按键模块 n 16-bit C/F 计数器寄存器对**

寄存器	TKMn16DH								TKMn16DL							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

该寄存器对用于存储触控按键模块 n 16-bit C/F 计数器值。在手动扫描模式中，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。若选择自动扫描模式，此 16-bit C/F 计数器内容在写入到触控按键存储器后，将在时隙 0~2 结束时被清零，在时隙 3 结束时保持不变。当 TKST 位置低，该寄存器对将被清零。

- **TKMnROH/TKMnROL – 触控按键模块 n 参考振荡器电容选择寄存器对**

寄存器	TKMnROH								TKMnROL							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	—	—	—	—	—	—	D9	D8	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	R/W	R/W
POR	—	—	—	—	—	—	0	0	0	0	0	0	0	0	0	0

该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	—	—	MnDFEN	MnFILEN	MnSOFC	MnSOF2	MnSOF1	MnSOF0
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　**MnDFEN**：触控按键模块 n 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”，按键振荡器频率将为原来的倍频。

Bit 4　**MnFILEN**：触控按键模块 n 滤波器功能控制位
0：除能
1：使能

Bit 3　**MnSOFC**：触控按键模块 n 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
000：1.020MHz
001：1.040MHz
010：1.059MHz
011：1.074MHz

100：1.085MHz
101：1.099MHz
110：1.111MHz
111：1.125MHz

上述频率会随着外部或内部电容值的不同而变化。若触控按键振荡器频率选择 1MHz，用户选择其它频率时，可依此比例调整。

- **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
该位用于使能触控按键模块参考振荡器。在自动扫描模式时，若选择使用参考振荡器作为时隙时钟源，当 TKST 位由低到高变化时，必须通过将 MnROEN 位置高使能参考振荡器。MnTSS 位和 MnK4EN~MnK1EN 位共同决定了参考振荡器是否被用。当 TKBUSY 位从高变为低时，MnROEN 位将被自动清零来除能参考振荡器。
在手动扫描时，若选择使用参考振荡器，在将 TKST 位设置为低到高前，必须先使能参考振荡器，当 TKBUSY 位从高变为低时，参考振荡器将除能。

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
该位用于使能触控按键模块按键振荡器。在自动扫描模式时，当 TKST 位由低到高变化时，必须通过将 MnKOEN 位置高使能按键振荡器。当 TKBUSY 位由高到低变化时，MKOEN 位将被自动清零来除能按键振荡器。
在手动扫描模式时，若使能相应的按键进行扫描时，在将 TKST 位设置为低到高前，必须先使能按键振荡器。当 TKBUSY 位从高变为低时，按键振荡器将除能。

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	M4	M5	M6	M7	M8	M9
0：除能	I/O 或其它功能									
1：使能	KEY4	KEY8	KEY12	KEY16	KEY20	KEY24	KEY28	KEY32	KEY36	KEY40
BS83B24C	√	√	√	√	√	√	—	—	—	—
BS83C40C	√	√	√	√	√	√	√	√	√	√

Bit 2 **MnK3EN**：触控按键模块 n KEY 3 使能控制

MnK3EN	触控按键模块 n (Mn)									
	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
0：除能	I/O 或其它功能									
1：使能	KEY3	KEY7	KEY11	KEY15	KEY19	KEY23	KEY27	KEY31	KEY35	KEY39
BS83B24C	√	√	√	√	√	√	—	—	—	—
BS83C40C	√	√	√	√	√	√	√	√	√	√

Bit 1 **MnK2EN**：触控按键模块 n KEY 2 使能控制

MnK2EN	触控按键模块 n (Mn)									
	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
0：除能	I/O 或其它功能									
1：使能	KEY2	KEY6	KEY10	KEY14	KEY18	KEY22	KEY26	KEY30	KEY34	KEY38
BS83B24C	√	√	√	√	√	√	—	—	—	—
BS83C40C	√	√	√	√	√	√	√	√	√	√

Bit 0 **MnK1EN**：触控按键模块 n KEY 1 使能控制

MnK1EN	触控按键模块 n (Mn)									
	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
0：除能	I/O 或其它功能									
1：使能	KEY1	KEY5	KEY9	KEY13	KEY17	KEY21	KEY25	KEY29	KEY33	KEY37
BS83B24C	√	√	√	√	√	√	—	—	—	—
BS83C40C	√	√	√	√	√	√	√	√	√	√

- **TKMnC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MnSK31	MnSK30	MnSK21	MnSK20	MnSK11	MnSK10	MnSK01	MnSK00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	1	0	0

Bit 7~6 **MnSK31~MnSK30**：触控按键模块 n 时隙 3 扫描按键选择位
00：KEY1
01：KEY2
10：KEY3
11：KEY4
这两位用于选择在时隙 3 时要被扫描的按键，仅在自动扫描模式时有效。

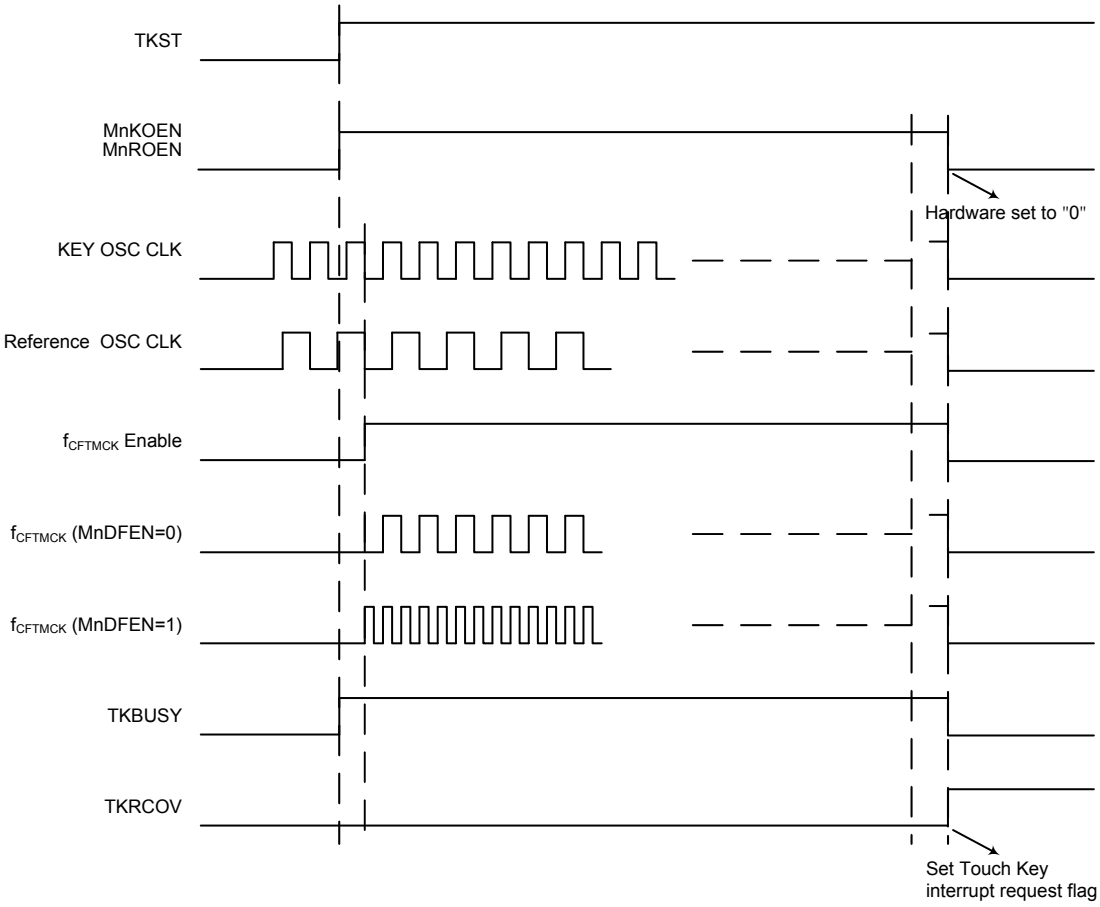
Bit 5~4 **MnSK21~MnSK20**：触控按键模块 n 时隙 2 扫描按键选择位
00：KEY1
01：KEY2
10：KEY3
11：KEY4
这两位用于选择在时隙 2 时要被扫描的按键，仅在自动扫描模式时有效。

Bit 3~2 **MnSK11~MnSK10**：触控按键模块 n 时隙 1 扫描按键选择位
00：KEY1
01：KEY2
10：KEY3
11：KEY4
这两位用于选择在时隙 1 时要被扫描的按键，仅在自动扫描模式时有效。

Bit 1~0 **MnSK01~MnSK00**：触控按键模块 n 时隙 0 扫描按键选择位
00：KEY1
01：KEY2
10：KEY3
11：KEY4
这两位用于选择在自动扫描模式下时隙 0 时要被扫描的按键或在手动模式时选择要扫描的按键。

触控按键操作

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



触控按键扫描模式时序图

每个触控按键模块包含四个与 I/O 引脚共用的触控按键，通过寄存器可设置相应引脚功能。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 TKC1 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。

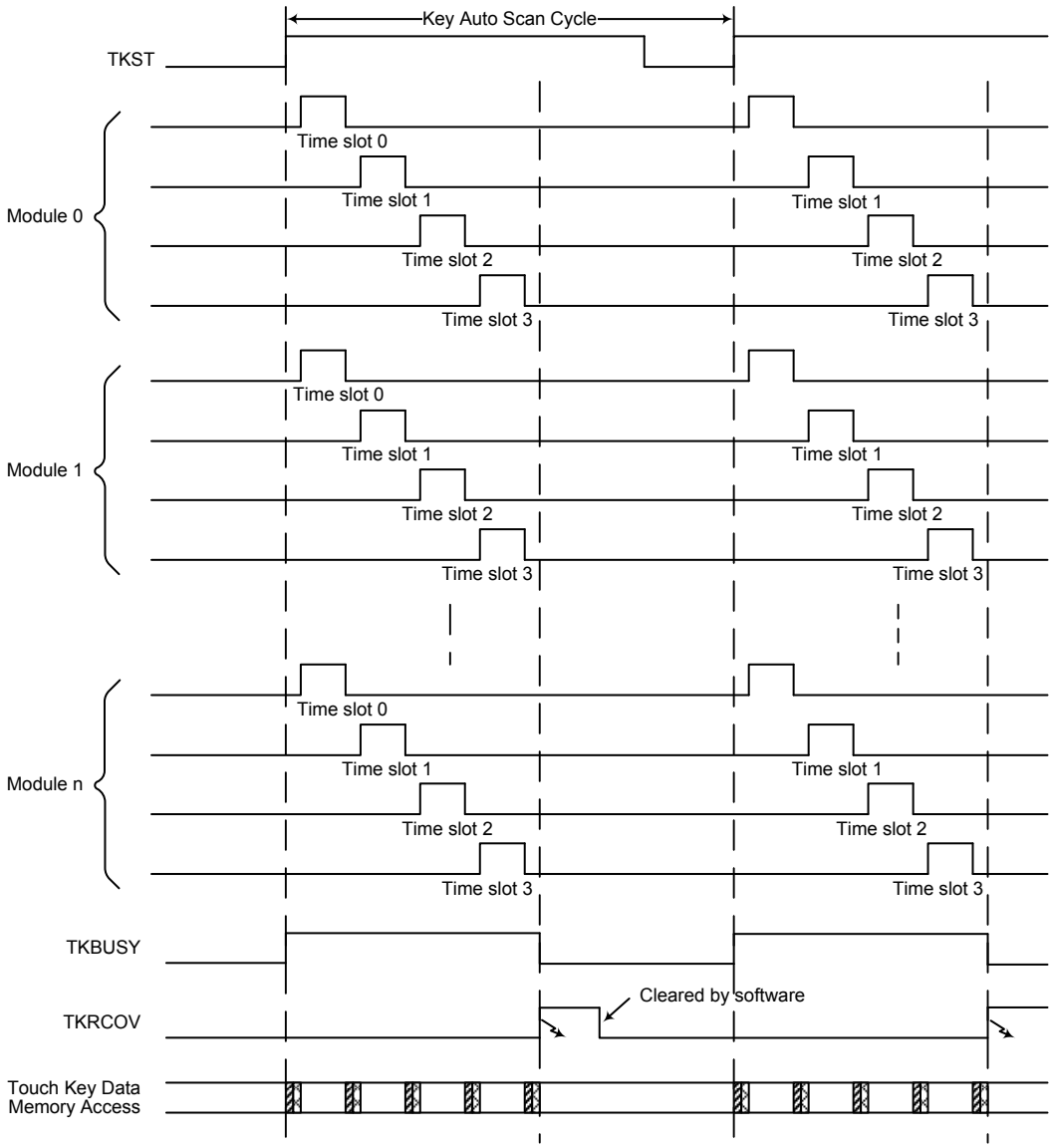
当时隙计数器溢出，所有模块的按键振荡器和参考振荡器都会自动停止且 16-bit C/F 计数器、16-bit 计数器、5-bit 时隙计数器和 8-bit 时隙计数器会自动停止。时隙计数器和 8+5 位计数器时钟来自参考振荡器或 $f_{SYS}/4$。当 TKMnC1 寄存器中的 MnROEN 位为“1”时，选到的参考振荡器就使能。当 TKMnC1 寄存器中的 MnKOEN 为“1”时，选到的按键振荡器就使能。

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 等等，每个模块都是相同的架构。

自动扫描模式

触控按键功能包含两种按键扫描模式，即自动扫描模式和手动扫描模式，可通过寄存器 TKC0 的 TKMOD 位选择。自动扫描模式可以最大程度的减少程序负担并能提高按键扫描执行效率。设置 TKMOD 位为 0 可选择自动扫描模式，在此模式下以指定顺序自动扫描模块按键。扫描顺序及按键由 TKMnC2 寄存器的 MnSK3[1:0]~MnSK0[1:0] 位决定。



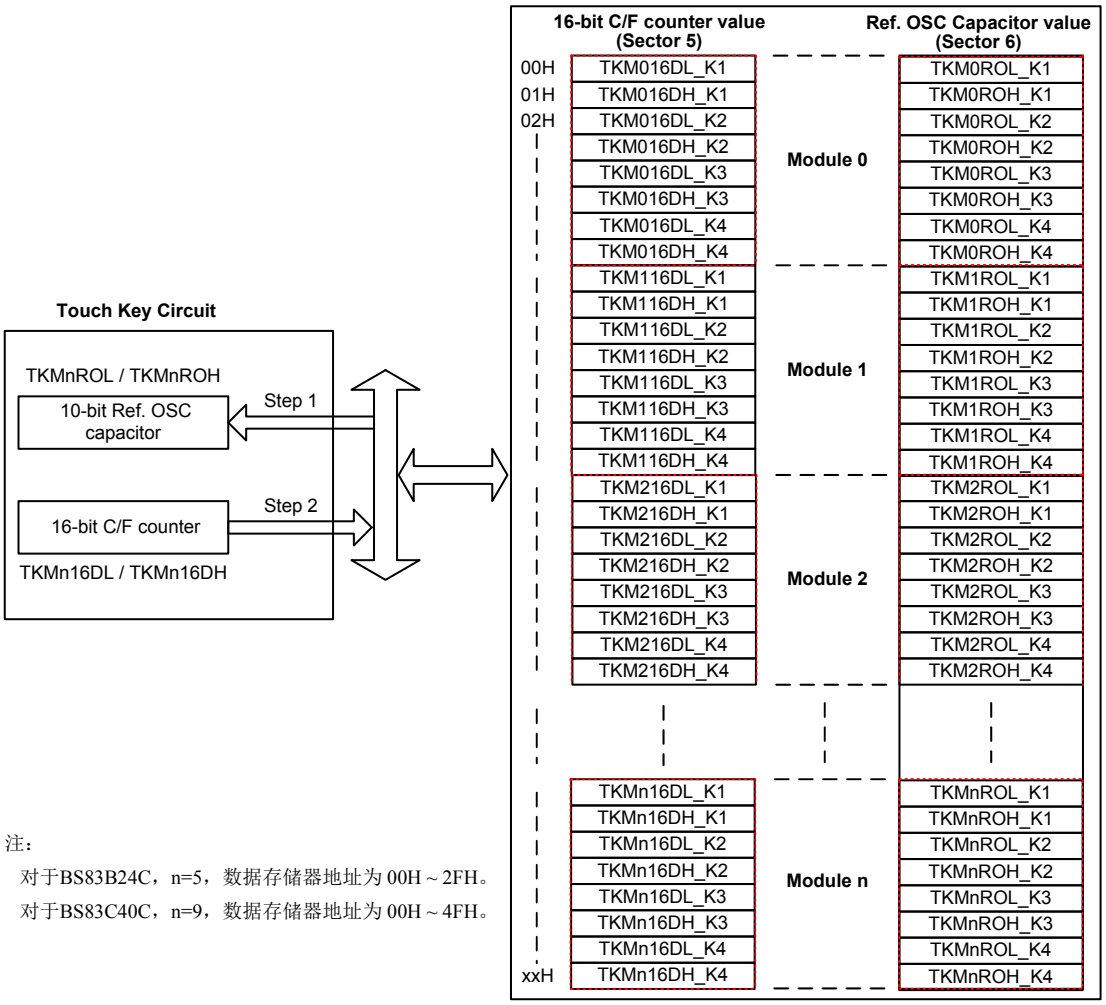
触控按键自动扫描模式时序图

在自动扫描模式时，按键振荡器和参考振荡器在 TKST 位由低变为高时，自动使能，在 TKBUSY 位由高到低时，自动除能。在自动扫描模式时，若已设置 TKST 位由低到高，硬件会自动从专用触控按键数据存储区指定位置读取时隙 0 选择扫描的按键对应的参考振荡器的电容值，并写入相应的 TKMnROH/TKMnROL 寄存器对中。并将 16-bit C/F 计数器的值写回到专用触控按键数据存储区时隙 3 扫描的按键对应的位置。之后开始扫描选择的时隙 0 的按键。时隙 0 按键扫描结束时，硬件会自动从专用触控按键数据存储区读取下一个要扫描按键对应的参考振荡器电容值并写入相应的 TKMnROH/TKMnROL 寄存器

对中。并将当前 16-bit C/F 计数器的值写回到专用触控按键数据存储区对应位置。整个自动扫描操作会按上述指定的方式从时隙 0 到时隙 3 有序的执行。时隙 3 按键扫描结束时，硬件会再次从专用触控按键数据存储区读取时隙 0 选择扫描的按键对应的参考振荡器的电容值，并写入相应的 TKMnROH/TKMnROL 寄存器对中。16 位 C/F 计数器值也会被再次写回到专用触控按键数据存储区时隙 3 扫描的按键对应的位置。四个按键全部扫描后，TKRCOV 位将被置高同时 TKBUSY 位拉低，意味着自动扫描模式下的扫描工作已完成。

触控按键模块数据存储器

该系列单片机为触控按键模块自动扫描模式提供 2 个专用的数据存储器区域。一个位于数据存储区 5 用于存储触控按键模块 16 位 C/F 计数器值，另一个位于数据存储区 6 用于存储触控按键模块参考振荡器内部电容值。



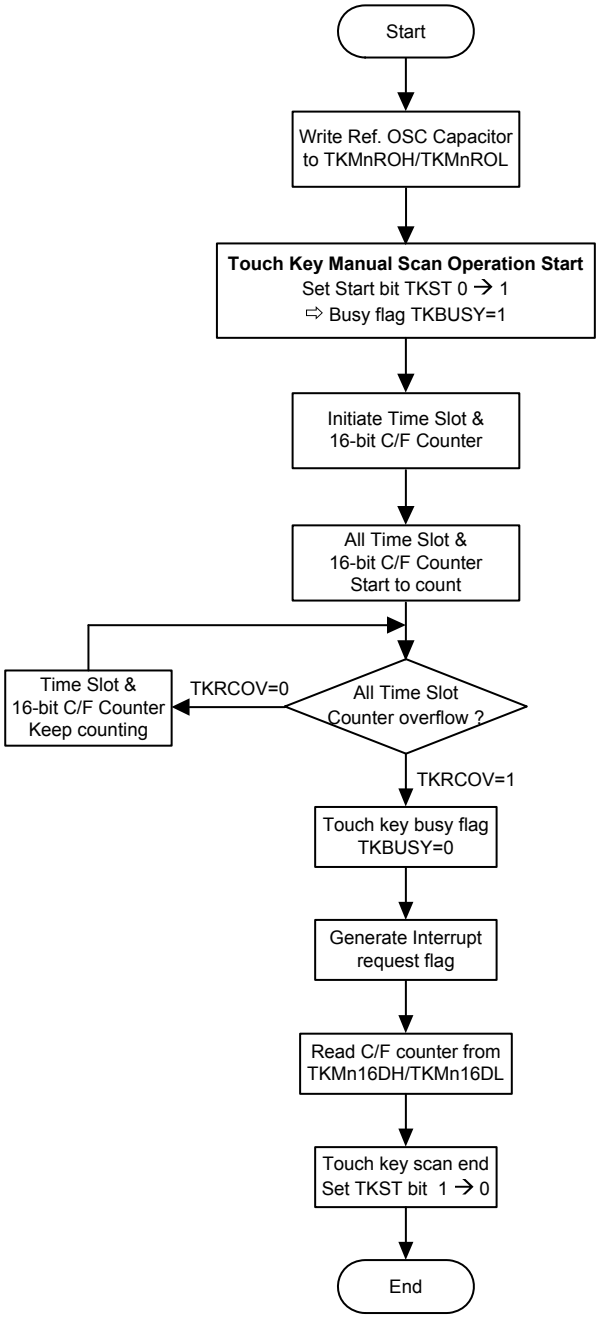
注：

对于BS83B24C，n=5，数据存储器地址为 00H ~ 2FH。

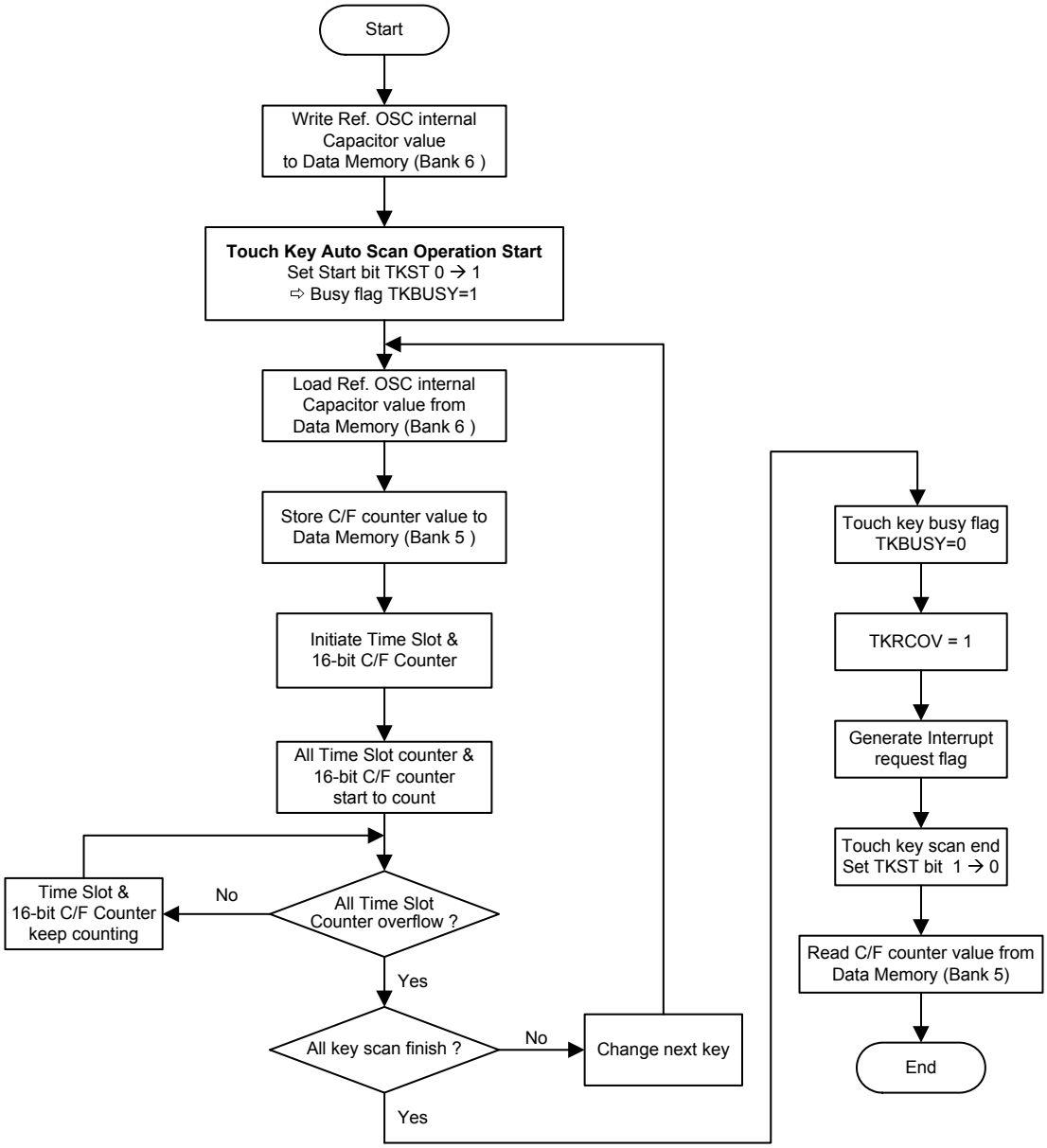
对于BS83C40C，n=9，数据存储器地址为 00H ~ 4FH。

触控按键模块数据存储器分布

触控按键扫描流程图



触控按键手动扫描模式流程图



触控按键自动扫描模式流程图

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。

当任意触控按键模块 16-bit C/F 计数器溢出时，16-bit C/F 计数器溢出标志位 TKCFOV 将置高。由于此标志位无法被自动清零，需通过应用程序将此位清零。

当 16-bit 计数器溢出时，其溢出标志位 TK16OV 将置高。由于此标志位无法被自动清零，需通过应用程序将此位清零。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。

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发生触摸动作或定时 / 计数器溢出等，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。此系列单片机提供单个外部中断和多个内部中断功能，外部中断由 INT 引脚，而内部中断由各种内部功能，如定时器模块、时基、EEPROM、触控按键模块和 USIM 模块等产生。

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~INTC2 寄存器，用于设置基本的中断；第二类是 MFI0~MFI1 寄存器，用于设置多功能中断；第三类是 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，紧接着是中断号 (可选)，最后的字母“E”代表使能 / 除能位，“F”代表请求标志位。

功能	使能位	请求标志位	注释
总中断	EMI	—	—
INT 脚	INTE	INTF	—
触控按键模块	TKME	TKMF	—
多功能	MFnE	MFnF	对于 BS83B24C，n=0 对于 BS83C40C，n=0~1
USIM	USIME	USIMF	—
时基	TBnE	TBnF	n=0~1
EEPROM	DEE	DEF	—
CTM	CTMPE	CTMPF	仅适用于 BS83C40C
	CTMAE	CTMAF	
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	—	MF0F	TKMF	INTF	MF0E	TKME	INTE	EMI
INTC1	DEF	TB1F	TB0F	USIMF	DEE	TB1E	TB0E	USIME
MFI0	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE

中断寄存器列表 – BS83B24C

寄存器名称	位							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	MF0F	TKMF	INTF	MF0E	TKME	INTE	EMI
INTC1	DEF	TB1F	TB0F	USIMF	DEE	TB1E	TB0E	USIME
INTC2	—	—	—	MF1F	—	—	—	MF1E
MFI0	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE
MFI1	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE

中断寄存器列表 – BS83C40C

- **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：除能中断
01：上升沿中断
10：下降沿中断
11：双沿中断

- **INTC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	MF0F	TKMF	INTF	MF0E	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　**MF0F**：多功能中断 0 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**：外部中断请求标志位
0：无请求
1：中断请求

Bit 3 **MF0E**：多功能中断 0 控制
0：除能
1：使能

Bit 2 **TKME**：触控按键模块中断控制
0：除能
1：使能

Bit 1 **INTE**：外部中断控制
0：除能
1：使能

Bit 0 **EMI**：总中断控制
0：除能
1：使能

- **INTC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	DEF	TB1F	TB0F	USIMF	DEE	TB1E	TB0E	USIME
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 **DEF**：数据 EEPROM 中断请求标志位
0：无请求
1：中断请求

Bit 6 **TB1F**：时基 1 中断请求标志位
0：无请求
1：中断请求

Bit 5 **TB0F**：时基 0 中断请求标志位
0：无请求
1：中断请求

Bit 4 **USIMF**：USIM 模块中断请求标志位
0：无请求
1：中断请求

Bit 3 **DEE**：数据 EEPROM 中断控制
0：除能
1：使能

Bit 2 **TB1E**：时基 1 中断控制
0：除能
1：使能

Bit 1 **TB0E**：时基 0 中断控制
0：除能
1：使能

Bit 0 **USIME**：USIM 模块中断控制
0：除能
1：使能

- **INTC2 寄存器 – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	MF1F	—	—	—	MF1E
R/W	—	—	—	R/W	—	—	—	R/W
POR	—	—	—	0	—	—	—	0

Bit 7~5　未定义，读为“0”

Bit 4　**MF1F**：多功能中断 1 请求标志位
0：无请求
1：中断请求

Bit 3~1　未定义，读为“0”

Bit 0　**MF1E**：多功能中断 1 控制
0：除能
1：使能

- **MFI0 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 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4　**PTMPF**：PTM 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求

Bit 3~2　未定义，读为“0”

Bit 1　**PTMAE**：PTM 比较器 A 匹配中断控制
0：除能
1：使能

Bit 0　**PTMPE**：PTM 比较器 P 匹配中断控制
0：除能
1：使能

- **MFI1 寄存器 – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE
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　**CTMAF**：CTM 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4　**CTMPF**：CTM 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求

Bit 3~2　未定义，读为“0”

Bit 1　**CTMAE**：CTM 比较器 A 匹配中断控制
0：除能
1：使能

Bit 0　**CTMPE**：CTM 比较器 P 匹配中断控制
0：除能
1：使能

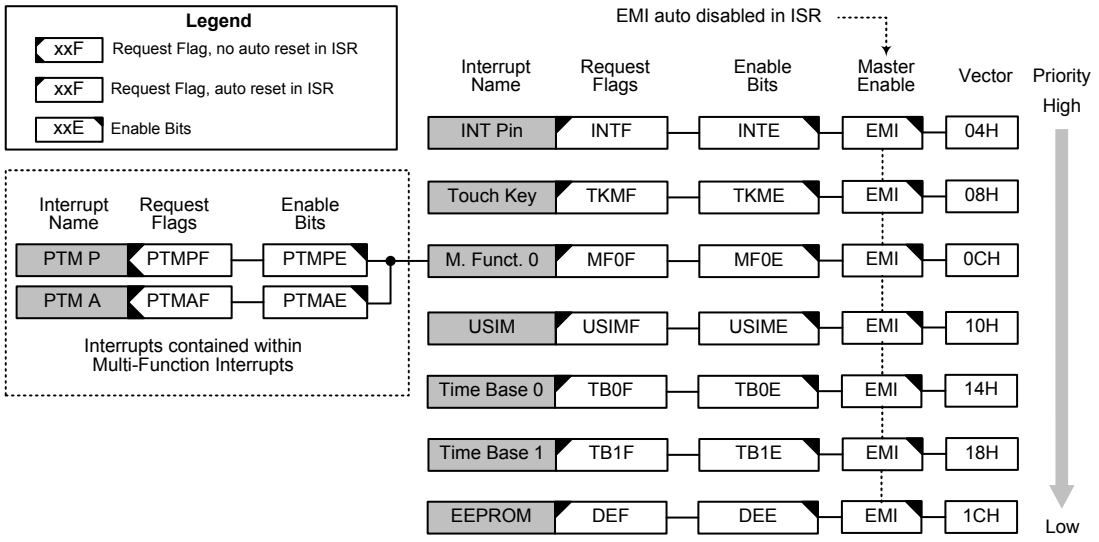
中断操作

若中断事件条件产生，如一个 TM 比较器 P、比较器 A 匹配或 EEPROM 写周期结束等等，相关中断请求标志将置起。中断标志产生后程序是否会跳转至相关中断向量执行是由中断使能位的条件决定的。若使能位为“1”，程序将跳至相关中断向量中执行；若使能位为“0”，即使中断请求标志置起中断也不会发生，程序也不会跳转至相关中断向量执行。若总中断使能位为“0”，所有中断都将除能。

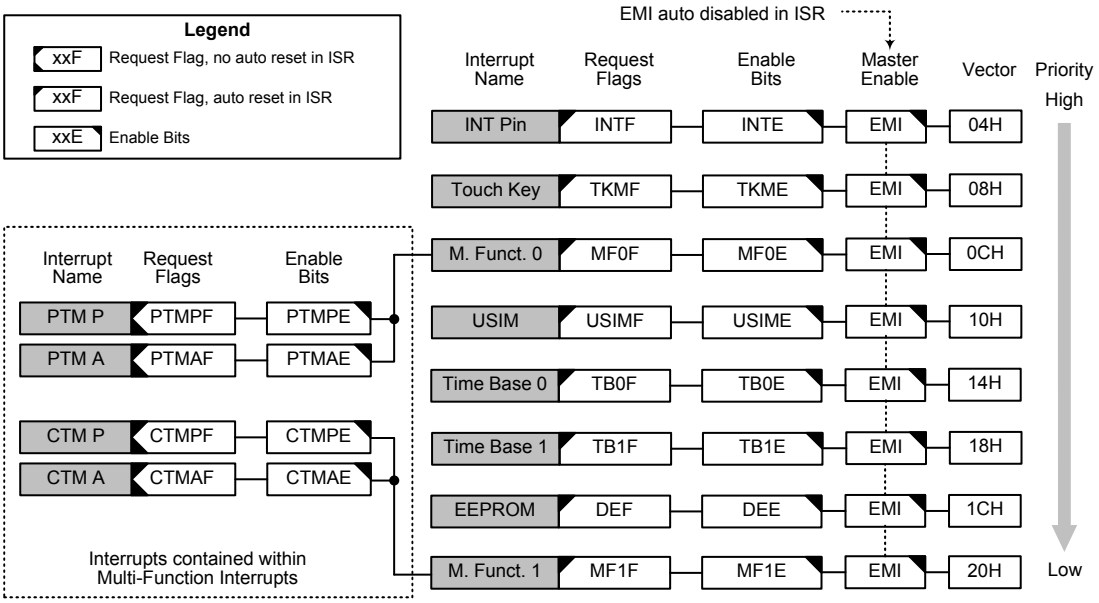
当中断发生时，下条指令的地址将被压入堆栈。相应的中断向量地址加载至 PC 中。系统将从此向量取下条指令。中断向量处通常为跳转指令，以跳转到相应的中断服务程序。中断服务程序必须以“RETI”指令返回至主程序，以继续执行原来的程序。

各个中断使能位以及相应的请求标志位，以优先级的次序显示在下图。一些中断源有自己的向量，但是有些中断却共用多功能中断向量。一旦中断子程序被响应，系统将自动清除 EMI 位，所有其它的中断将被屏蔽，这个方式可以防止任何进一步的中断嵌套。其它中断请求可能发生在此期间，虽然中断不会立即响应，但是中断请求标志位会被记录。

如果某个中断服务子程序正在执行时，有另一个中断要求立即响应，那么 EMI 位应在程序进入中断子程序后置位，以允许此中断嵌套。如果堆栈已满，即使此中断使能，中断请求也不会被响应，直到 SP 减少为止。如果要求立刻动作，则堆栈必须避免成为储满状态。请求同时发生时，执行优先级如下流程图所示。所有被置起的中断请求标志都可把单片机从休眠或空闲模式中唤醒，若要防止唤醒动作发生，在单片机进入休眠或空闲模式前应将相应的标志置起。



中断结构 – BS83B24C



中断结构 – **BS83C40C**

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。

多功能中断

该系列单片机具有多功能中断。与其它中断不同，这些没有独立源，但由其它现有的中断源构成，即 TM 中断。

当多功能中断中任何一种中断请求标志 MFnF 被置位，多功能中断请求产生。当所包含的任一功能产生中断请求标志，多功能中断标志将置位。若要跳转到相应的中断向量地址，当多功能中断使能，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必须由应用程序清零。

TM 中断

简易型和周期型 TM 各有两个内部中断，即内部比较器 A 或内部比较器 P，当发生比较匹配时将产生 TM 中断。不同的 TM 都有两个中断请求标志位和两个使能位。当 TM 比较器 P、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、相应 TM 中断使能位和相关多功能中断使能位 MFnE 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。但是只有相关的 MFnF 标志位会被自动清除。由于 TM 中断请求标志位不会自动复位，其必须通过应用程序手动清除。

EEPROM 中断

当写周期结束，EEPROM 中断请求标志 DEF 被置位，EEPROM 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 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。

USIM 中断

通用串行接口模块中断，即 USIM 中断。当 USIM 接口中断标志位 USIMF 置位时，产生中断请求。由于 USIM 接口可工作在三个模式下：SPI 模式、I^2C 模式和 UART 模式，USIMF 标志位置位可由不同情况触发，取决于所选择的接口模式。

若选择 SPI 或 I^2C 模式，当一个字节数据已由 SPI/I^2C 接口接收或发送完，或 I^2C 从机地址匹配，或 I^2C 超时，中断请求标志 USIMF 被置位，USIM 中断请求产生。若选择 UART 模式，USIM 中断由几种 UART 传输条件控制。当发送器为空、发送器空闲、接收器数据有效、接收器溢出、地址检测和RX引脚唤醒，USIM 中断请求标志 USIMF 被置位，USIM 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 和通用串行接口中断使能位 USIME 需先被置位。当中断使能，堆栈未满且以上任一种情况发生时，将调用相应的 USIM 中断向量子程序。当响应中断服务子程序时，通用串行接口中断标志位 USIMF 会自动复位且 EMI 将被自动清零以除能其它中断。

注意，当 USIM 中断是由 UART 接口触发产生的，当中断响应后，UUSR 寄存器里的标志位只有在对 UART 执行特定动作时才会被清零，详细参考 UART 章节。

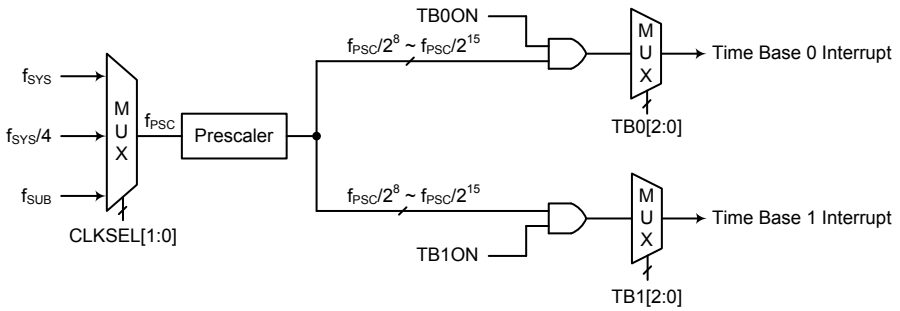
触控按键模块中断

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。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。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。当出现这种情况时其各自的中断请求标志 TBnF 被置位，中断请求发生。若要跳转到其相应的中断向量地址，总中断使能位 EMI 和时基使能位 TBnE 需先被置位。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 TBnF 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} 来源于内部时钟源 f_{SYS}、$f_{SYS}/4$ 或 f_{SUB}，经过一个分频器，其分频比可通过配置 TBnC 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　**CLKSEL01~CLKSEL00**：预分频器时钟源选择
00：f_{SYS}
01：$f_{SYS}/4$
1x：f_{SUB}

- **TB0C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	—	—	—	—	TB02	TB01	TB00
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7　**TB0ON**：时基 0 使能 / 除能控制位
0：除能
1：使能

Bit 6~3　未定义，读为“0”

Bit 2~0　**TB02~TB00**：时基 0 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：$2^{15}/f_{psc}$

- **TB1C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TB1ON	—	—	—	—	TB12	TB11	TB10
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB1ON**：时基 1 使能 / 除能控制位
0：除能
1：使能

Bit 6~3 未定义，读为“0”

Bit 2~0 **TB12~TB10**：时基 1 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：$2^{15}/f_{psc}$

中断唤醒功能

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

编程注意事项

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

有的中断为多功能中断，当响应中断服务时，只有多功能中断请求标志 MFnF 会自动清零，其独立中断请求标志需通过应用程序手动清零。

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，清除 EMI 位，除能进一步中断。

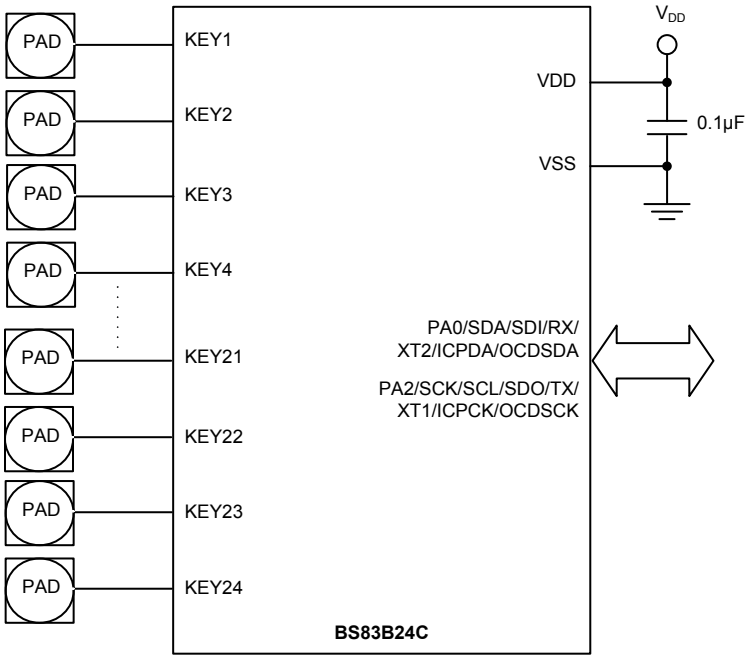
配置选项

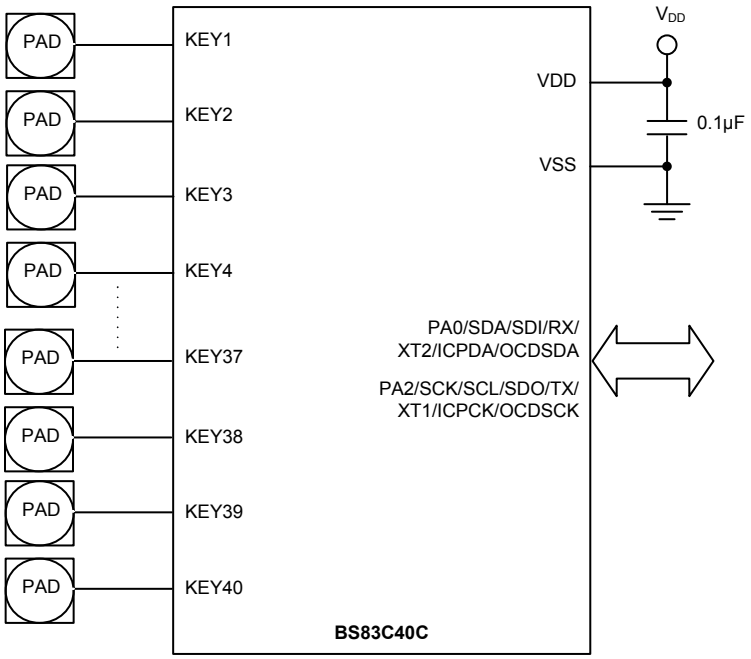
配置选项在烧写程序时写入芯片。通过 HT-IDE 的软件开发环境，使用者在开发过程中可以选择配置选项。由于使用的是硬件工具将配置选项烧入单片机，之后无法再通过应用程序修改。所有的选项必须按系统的需要定义，具体内容可参考下表：

序号	选项
振荡器选项	
1	HIRC 频率选择： 1. 8MHz 2. 12MHz 3. 16MHz

注：当 HIRC 配置选项已选定上表中的一个频率，HIRC1 和 HIRC0 位选择的频率应与其保持一致，以确保能够达到交流电气特性中标示的 HIRC 频率精度。

应用电路




PAD
KEY1
PAD
KEY2
PAD
KEY3
PAD
KEY4
PAD
KEY37
PAD
KEY38
PAD
KEY39
PAD
KEY40
BS83C40C
VDD
VSS
V_{DD}
0.1μF
PA0/SDA/SDI/RX/
XT2/ICPDA/OCDSDA
PA2/SCK/SCL/SDO/TX/
XT1/ICPCK/OCDSCK

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

当要操作的数据存储器位于数据存储器 Sector 0 时，下表说明了与数据存储器存取有关的指令。

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

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

助记符	说明	指令周期	影响标志位
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多达 3 个周期，如果没有发生跳转，则只需一个周期。
2. 任何指令若要改变 PCL 的内容将需要 2 个周期来执行。
3. 对于“CLR WDT”指令而言，TO 和 PDF 标志位也许会受执行结果影响，“CLR WDT”被执行后，TO 和 PDF 标志位会被清除，否则 TO 和 PDF 标志位保持不变。

扩展指令集

扩展指令用来提供更大范围的数据存储器寻址。当被存取的数据存储器位于 Sector 0 之外的任何数据存储器 Sector，扩展指令可直接存取数据存储器而无需使用间接寻址，此举也提高了 CPU 韧体性能。

助记符		说明	指令周期	影响标志位
算术运算				
LADD	A,[m]	ACC 与数据存储器相加，结果放入 ACC	2	Z, C, AC, OV, SC
LADDM	A,[m]	ACC 与数据存储器相加，结果放入数据存储器	2注	Z, C, AC, OV, SC
LADC	A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	2	Z, C, AC, OV, SC
LADCM	A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	2注	Z, C, AC, OV, SC
LSUB	A,[m]	ACC 与数据存储器相减，结果放入 ACC	2	Z, C, AC, OV, SC, CZ
LSUBM	A,[m]	ACC 与数据存储器相减，结果放入数据存储器	2注	Z, C, AC, OV, SC, CZ
LSBC	A,[m]	ACC 与数据存储器、进位标志的反相减，结果放入 ACC	2	Z, C, AC, OV, SC, CZ
LSBCM	A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	2注	Z, C, AC, OV, SC, CZ
LDAA	[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	2注	C
逻辑运算				
LAND	A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	2	Z
LOR	A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	2	Z
LXOR	A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	2	Z
LANDM	A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	2注	Z
LORM	A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	2注	Z
LXORM	A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	2注	Z
LCPL	[m]	对数据存储器取反，结果放入数据存储器	2注	Z
LCPLA	[m]	对数据存储器取反，结果放入 ACC	2	Z
递增和递减				
LINCA	[m]	递增数据存储器，结果放入 ACC	2	Z
LINC	[m]	递增数据存储器，结果放入数据存储器	2注	Z
LDECA	[m]	递减数据存储器，结果放入 ACC	2	Z
LDEC	[m]	递减数据存储器，结果放入数据存储器	2注	Z
移位				
LRRA	[m]	数据存储器右移一位，结果放入 ACC	2	无
LRR	[m]	数据存储器右移一位，结果放入数据存储器	2注	无
LRRCA	[m]	带进位将数据存储器右移一位，结果放入 ACC	2	C
LRRC	[m]	带进位将数据存储器右移一位，结果放入数据存储器	2注	C
LRLA	[m]	数据存储器左移一位，结果放入 ACC	2	无
LRL	[m]	数据存储器左移一位，结果放入数据存储器	2注	无
LRLCA	[m]	带进位将数据存储器左移一位，结果放入 ACC	2	C
LRLC	[m]	带进位将数据存储器左移一位，结果放入数据存储器	2注	C
数据传送				
LMOV	A,[m]	将数据存储器送至 ACC	2	无
LMOV	[m],A	将 ACC 送至数据存储器	2注	无
位运算				
LCLR	[m].i	清除数据存储器的位	2注	无
LSET	[m].i	置位数据存储器的位	2注	无

助记符		说明	指令周期	影响标志位
转移				
LSZ	[m]	如果数据存储器为零，则跳过下一条指令	2 注	无
LSZA	[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
LSNZ	[m]	如果数据存储器不为零，则跳过下一条指令	2 注	无
LSZ	[m].i	如果数据存储器的第 i 位为零，则跳过下一条指令	2 注	无
LSNZ	[m].i	如果数据存储器的第 i 位不为零，则跳过下一条指令	2 注	无
LSIZ	[m]	递增数据存储器，如果结果为零，则跳过下一条指令	2 注	无
LSDZ	[m]	递减数据存储器，如果结果为零，则跳过下一条指令	2 注	无
LSIZA	[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	2 注	无
LSDZA	[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	2 注	无
查表				
LTABRD	[m]	读取当前页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
LTABRDL	[m]	读取最后页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
LITABRD	[m]	读表指针 TBLP 自加，读取当前页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
LITABRDL	[m]	读表指针 TBLP 自加，读取最后页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
其它指令				
LCLR	[m]	清除数据存储器	2 注	无
LSET	[m]	置位数据存储器	2 注	无
LSWAP	[m]	交换数据存储器的高低字节，结果放入数据存储器	2 注	无
LSWAPA	[m]	交换数据存储器的高低字节，结果放入 ACC	2	无

注：1. 对扩展跳转指令而言，如果比较的结果牵涉到跳转即需多达 4 个周期，如果没有发生跳转，则只需两个周期。
2. 任何扩展指令若要改变 PCL 的内容将需要 3 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、SC

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、SC

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、SC

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、SC

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、SC

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 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 PC ← PC + 1
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相应，则这个中断将在返回主程序之前被相应。
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、SC、CZ

SBC A, x Subtract immediate data from ACC with Carry
指令说明 将累加器减去立即数以及进位标志，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、SC、CZ

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、SC、CZ

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] Decrement data memory and place result in ACC, skip if 0
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

SNZ [m] Skip if Data Memory is not 0
指令说明 判断指定存储器，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]≠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、SC、CZ

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、SC、CZ

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、SC、CZ

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 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) to TBLH and Data Memory

指令说明 将表格指针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 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

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

ITABRD [m] Increment table pointer low byte first and read table to TBLH and data memory

指令说明 将自加表格指针 TBLP 所指的程序代码低字节 (当前页) 移至指定的数据存储器且将高字节移至 TBLH。

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

ITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and data memory

指令说明 将自加表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定的数据存储器且将高字节移至 TBLH。

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

扩展指令定义

扩展指令被用来直接存取存储在任何数据存储器 Sector 中的数据。

LADC 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、SC

LADCM 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、SC

LADD A, [m] Add Data Memory to ACC
指令说明 将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、SC

LADDM 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、SC

LAND 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

LANDM 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

LCLR [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

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

LCPL [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

LCPLA [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，结果被存放回累加器且数据寄存器的内容保持不变。
功能表示 ACC ← $[\overline{m}]$
影响标志位 Z

LDAA [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 调整就执行对低四位加“6”，否则低四位保持不变；如果高四位的值大于“9”或C=1，那么BCD调整就执行对高四位加“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

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

LDECA [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

LINC [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

LINCA [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

LMOV A, [m] Move Data Memory to ACC
指令说明 将指定数据存储器的内容复制到累加器中。
功能表示 ACC ← [m]
影响标志位 无

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

LOR 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

LORM 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

LRL [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

LRLA [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

LRLC [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

LRLC 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

LRR [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

LRRA [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

LRRC [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

LRRCA [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

LSBC 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、SC、CZ

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

指令说明 将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。

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

影响标志位 OV、Z、AC、C、SC、CZ

LSDZ [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 $[m] \leftarrow [m] - 1$，如果 [m]=0 跳过下一条指令执行

影响标志位 无

LSDZA [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 $ACC \leftarrow [m] - 1$，如果 ACC=0 跳过下一条指令执行

影响标志位 无

LSET [m] Set Data Memory

指令说明 将指定数据存储器的每一个位置位为 1。

功能表示 $[m] \leftarrow FFH$

影响标志位 无

LSET [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 $[m].i \leftarrow 1$

影响标志位 无

LSIZ [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

LSIZA [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

LSNZ [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

LSNZ [m] Skip if Data Memory is not 0
指令说明 判断指定数据存储器，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]≠0，跳过下一条指令执行
影响标志位 无

LSUB 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、SC、CZ

LSUBM 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、SC、CZ

LSWAP [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

LSWAPA [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 位和高 4 位互相交换，再将结果存放到累加器且指定数据寄存器的数据保持不变。
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

LSZ [m] Skip if Data Memory is 0
指令说明 判断指定数据存储器的内容是否为 0，若为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 如果 [m]=0，跳过下一条指令执行
影响标志位 无

LSZA [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

LSZ [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

LTABRD [m] Move the ROM code to TBLH and data memory

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

功能表示 [m] ← 程序代码 (低字节)
TBLH ← 程序代码 (高字节)

影响标志位 无

LTABRDL [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

LITABRD [m] Increment table pointer low byte first and read table to TBLH and data memory

指令说明 将自加表格指针 TBHP 和 TBLP 所指的程序代码低字节移至指定的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

LITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and data memory

指令说明 将自加表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定的数据存储器且将高字节移至 TBLH。

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

LXOR 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

LXORM 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

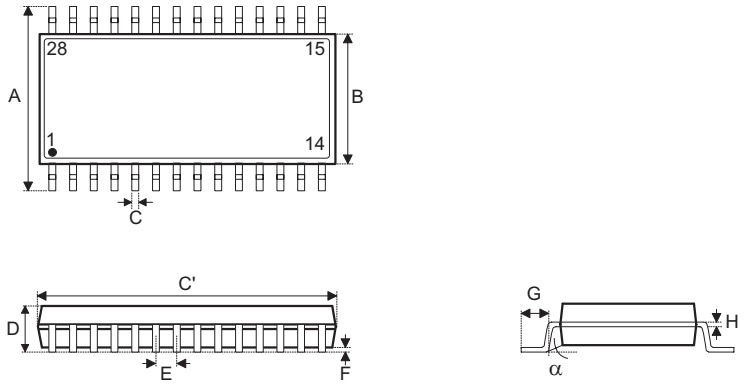
封装信息

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

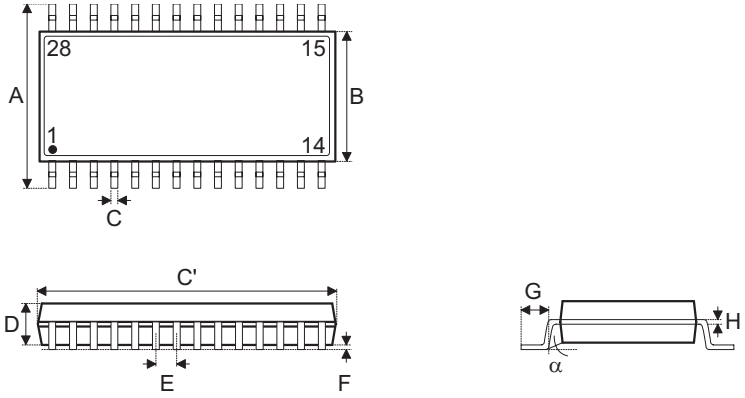
28-pin SOP (300mil) 外形尺寸



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’	—	0.705 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’	—	17.90 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°

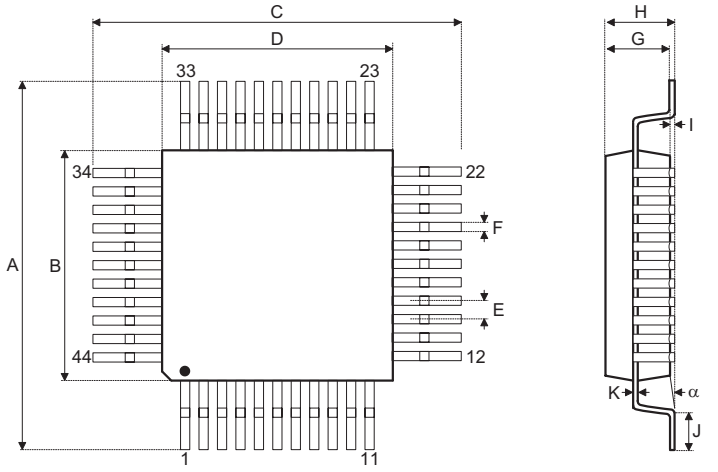
28-pin SSOP (150mil) 外形尺寸



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’	—	0.390 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’	—	9.90 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°

44-pin LQFP (10mm×10mm) (FP2.0mm) 外形尺寸



符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	—	0.472 BSC	—
B	—	0.394 BSC	—
C	—	0.472 BSC	—
D	—	0.394 BSC	—
E	—	0.032 BSC	—
F	0.012	0.015	0.018
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	—	12.00 BSC	—
B	—	10.00 BSC	—
C	—	12.00 BSC	—
D	—	10.00 BSC	—
E	—	0.80 BSC	—
F	0.30	0.37	0.45
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°