



9V 高压感烟探测器 Flash 单片机

BA45F5450

版本 : V1.10 日期 : 2022-08-11

www.holtek.com

目录

特性	7
CPU 特性	7
周边特性	7
概述	8
方框图	8
引脚图	9
引脚说明	10
极限参数	14
直流电气特性	14
工作电压特性	14
工作电流特性	14
待机电流特性	15
交流电气特性	15
内部高速振荡器 HIRC 频率精准度	15
内部低速振荡器 LIRC 电气特性	15
系统上电时间电气特性	16
输入 / 输出电气特性	16
存储器电气特性	17
LVD & LVR 电气特性	18
A/D 转换器电气特性	18
内部参考电压特性	19
灌电流发生器电气特性	19
灌电流发生器电气特性曲线图	20
运算放大器电气特性 – 感烟探测器 AFE	21
16-bit 语音 D/A 转换器电气特性	22
LDO 电气特性	22
压电蜂鸣片驱动器电气特性	23
上电复位特性	23
系统结构	24
时序和流水线结构	24
程序计数器	25
堆栈	25
算术逻辑单元 – ALU	26
Flash 程序存储器	26
结构	26
特殊向量	27
查表	27
查表范例	27
在线烧录 – ICP	28

片上调试 – OCDS.....	29
在线应用编程 – IAP.....	29
数据存储器	44
结构	44
数据存储器寻址	44
通用数据存储器	45
特殊功能数据存储器	45
特殊功能寄存器	46
间接寻址寄存器 – IAR0, IAR1, IAR2.....	46
存储器指针 – MP0, MP1L, MP1H, MP2L, MP2H.....	46
累加器 – ACC	47
程序计数器低字节寄存器 – PCL.....	47
表格寄存器 – TBLP, TBHP, TBLH.....	48
状态寄存器 – STATUS.....	48
EEPROM 数据存储器.....	50
EEPROM 数据存储器结构	50
EEPROM 寄存器.....	50
从 EEPROM 中读取数据	51
写数据到 EEPROM.....	51
写保护	52
EEPROM 中断	52
编程注意事项	52
振荡器	53
振荡器概述	53
系统时钟配置	53
内部高速 RC 振荡器 – HIRC	54
内部 32kHz 振荡器 – LIRC	54
工作模式和系统时钟	54
系统时钟	54
系统工作模式	55
控制寄存器	56
工作模式切换	57
待机电流注意事项	60
唤醒	60
看门狗定时器	61
看门狗定时器时钟源	61
看门狗定时器控制寄存器	61
看门狗定时器操作	62
复位和初始化	63
复位功能	63
复位初始状态	65
输入 / 输出端口	69
上拉电阻	69
PA 口唤醒	70

输入 / 输出端口控制寄存器	70
源电流选择	70
引脚共用功能	72
输入 / 输出引脚结构	76
编程注意事项	76
定时器模块 – TM	77
简介	77
TM 操作	77
TM 时钟源	77
TM 中断	77
TM 外部引脚	77
编程注意事项	78
标准型 TM – STM	80
标准型 TM 操作	80
标准型 TM 寄存器介绍	80
标准型 TM 工作模式	84
周期型 TM – PTM	93
周期型 TM 操作	93
周期型 TM 寄存器介绍	93
周期型 TM 工作模式	98
感烟探测器 AFE	110
感烟探测器 AFE 寄存器	111
运算放大器操作	114
压电蜂鸣片驱动器	115
自激模式	115
他激模式	115
A/D 转换器	116
A/D 简介	116
A/D 转换寄存器介绍	116
A/D 转换器操作	119
A/D 转换器参考电压	119
A/D 转换器输入信号	120
A/D 转换率及时序图	120
A/D 转换步骤概述	121
编程注意事项	122
A/D 转换功能	122
A/D 转换应用范例	122
灌电流发生器	124
灌电流发生器寄存器	124
16-bit 语音 D/A 转换器	125
16-bit 语音 D/A 转换器寄存器	125
串行接口模块 – SIM	127
SPI 接口	127
I ² C 接口	134

UART 接口	144
UART 外部引脚.....	144
UART 数据传输方案.....	145
UART 状态和控制寄存器.....	145
波特率发生器	149
UART 模块的设置与控制.....	150
UART 发送器.....	151
UART 接收器.....	152
接收错误处理	153
UART 模块中断结构.....	154
UART 模块暂停和唤醒.....	155
低电压检测 – LVD	156
LVD 寄存器	156
LVD 操作	156
中断	157
中断寄存器	157
中断操作	161
外部中断	162
A/D 转换器中断	162
时基中断	163
多功能中断	164
串行接口模块中断	164
UART 中断.....	164
LVD 中断	164
EEPROM 中断	164
TM 中断	165
中断唤醒功能	165
编程注意事项	165
应用电路	166
他激式蜂鸣器应用电路	166
自激式蜂鸣器应用电路	167
指令集	168
简介	168
指令周期	168
数据的传送	168
算术运算	168
逻辑和移位运算	168
分支和控制转换	169
位运算	169
查表运算	169
其它运算	169
指令集概要	170
惯例	170
扩展指令集	173

指令定义	175
扩展指令定义	187
封装信息	197
20-pin SOP (300mil) 外形尺寸	198
24-pin SOP (300mil) 外形尺寸	199
28-pin SOP (300mil) 外形尺寸	200

特性

CPU 特性

- 工作电压：
 - ◆ $f_{SYS}=2\text{MHz}/4\text{MHz}/8\text{MHz}$: 3.3V
- $V_{DD}=3.3\text{V}$, 系统时钟为 8MHz 时, 指令周期为 $0.5\mu\text{s}$
- 提供暂停和唤醒功能, 以降低功耗
- 振荡器类型:
 - ◆ 内部高速 2/4/8MHz RC – HIRC
 - ◆ 内部低速 32kHz RC – LIRC
- 多种工作模式: 快速、低速、空闲和休眠
- 内部集成的振荡器无需外接元件
- 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
- 8 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器: $8\text{K}\times 16$
- RAM 数据存储器: 1024×8
- True EEPROM 存储器: 128×8
- 看门狗定时器功能
- 支持在线应用编程 – IAP
- 17 个双向 I/O 口
- 两个与 I/O 口共用的外部中断引脚
- 可编程 I/O 口源电流用于 LED 应用
- 用于恒定电流输出的灌电流发生器
- 感烟探测器 AFE, 带有 2 个运算放大器
- 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
- 串行接口模块 – SIM, 用于 SPI 或 I²C 通信
- 全双工通用异步收发器接口 – UART
- 7 个外部通道 12-bit 分辨精度的 A/D 转换器, 具有内部参考电压 V_{BREF}
- 16-bit 语音 D/A 转换器
- 低电压复位功能
- 低电压检测功能
- 12V/30mA TinyPower LDO

- 压电蜂鸣片驱动器，用于报警功能
- 封装类型：20/24/28-pin SOP

概述

BA45F5450 是一款 A/D 型具有 8 位高性能精简指令集的 Flash 单片机，专门为需要蜂鸣器报警功能的高压感烟探测器应用而设计。

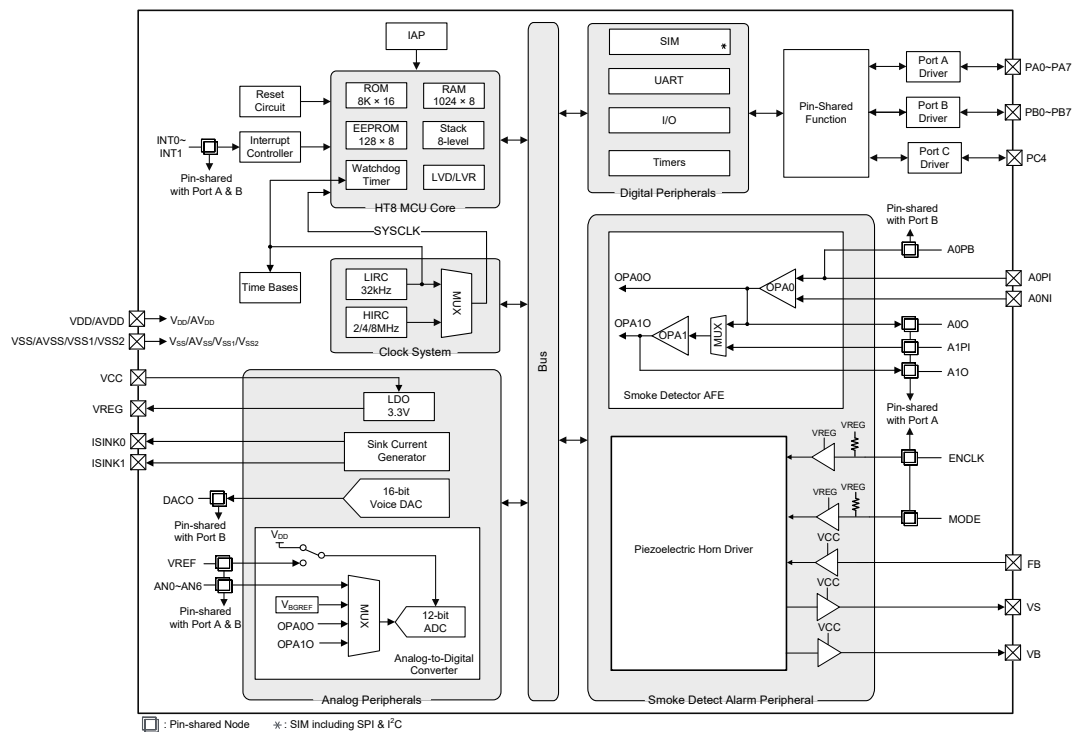
在存储器特性方面，Flash 存储器可多次编程的特性给用户提供了较大的方便。此外还包含了一个 RAM 数据存储器和一个可用于存储序号、校准数据等非易失性数据的 True EEPROM 存储器。

在模拟特性方面，该单片机包含一个多通道 A/D 转换器、两个运算放大器、一个 D/A 转换器和一个允许高达 12V 的电压输入并提供 3.3V 的固定电压的内部 TinyPower LDO。在内部定时器方面，带有多个使用灵活的定时器模块，可提供定时功能、脉冲产生功能及 PWM 产生功能。内建完整的 SPI、I²C 和 UART 接口，为设计者提供了一个易与外部硬件通信的方法。内部看门狗定时器、低电压复位和低电压检测等内部保护特性，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。

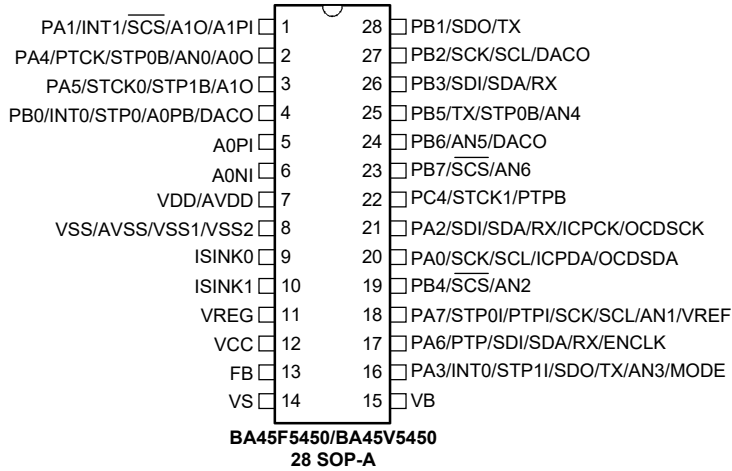
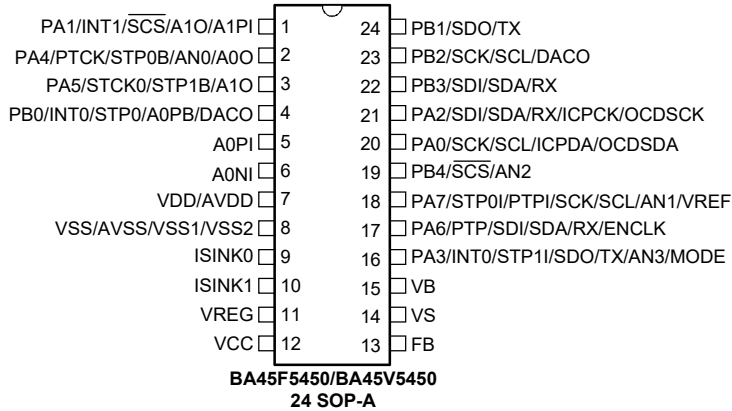
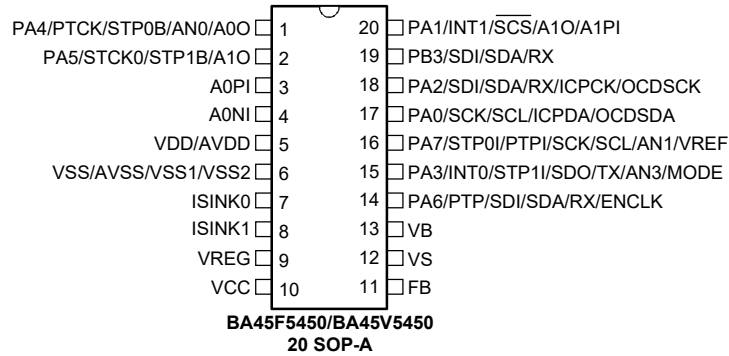
该单片机提供内部高速和低速振荡器功能选项，可灵活应用于不同程序。其不同工作模式之间动态切换的能力，为用户提供了一个优化单片机操作和减少功耗的手段。

该单片机包含了可编程 I/O 口源电流用于 LED 应用。外加 I/O 使用灵活和时基功能、压电蜂鸣片驱动器等其它特性，使该单片机可以广泛应用于感烟探测器报警应用产品中。

方框图



引脚图



- 注：1. 若共用引脚同时有多种输出，所需引脚共用功能通过相应的软件控制位决定。然而，MODE 和 ENCLK 功能分别与 PA3 和 PA6 引出到相同引脚上，这两个功能在上电后始终使能。
2. OCSDA 和 OCDSCK 引脚为片上调试功能专用引脚，仅存在于 BA45F5450 的 OCDS EV 芯片 BA45V5450。
3. 对于未引出的功能线，例如 20-pin SOP 封装中的 PB0~PB2、PB4~PB7、PC0~PC3、PC5、24-pin SOP 封装中的 PB5~PB7、PC0~PC3、PC5 和 28-pin SOP 封装中的 PC0~PC3、PC5，需合理设置其状态以避免输入浮空造成额外耗电，详见“待机电流注意事项”和“输入/输出端口”章节。
4. 在一般感烟探测器应用中，VREG 引脚需与 VDD 引脚外部相连，以保证正常操作。

引脚说明

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。由于该单片机存在不止一种封装，该表格反映的是较大封装类型的情况。

引脚名称	功能	OPT	I/T	O/T	说明
PA0/SCK/SCL/ICPDA/ OCDSDA	PA0	PAPU PAWU 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 IFS0	ST	CMOS	SPI 串行时钟
	SCL	PAS0 IFS0	ST	NMOS	I ² C 时钟线
	ICPDA	—	ST	CMOS	ICP 数据 / 地址
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址，仅用于 EV 芯片
PA1/INT1/ $\overline{\text{SCS}}$ /A1O/ A1PI	PA1	PAPU PAWU 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
	INT1	PAS0 INTC0 INTEG	ST	—	外部中断 1
	$\overline{\text{SCS}}$	PAS0 IFS0	ST	CMOS	SPI 从机选择
	A1O	PAS0	—	AN	运算放大器 1 输出
	A1PI	PAS0	AN	—	SD 运算放大器 1 同相输入端
PA2/SDI/SDA/RX/ ICPCK/OCDSCK	PA2	PAPU PAWU 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
	SDI	PAS0 IFS0	ST	—	SPI 串行数据输入
	SDA	PAS0 IFS0	ST	NMOS	I ² C 数据线
	RX	PAS0 IFS1	ST	—	UART 接收脚
	ICPCK	—	ST	—	ICP 时钟引脚
	OCDSCK	—	ST	—	OCDS 时钟引脚，仅用于 EV 芯片

引脚名称	功能	OPT	I/T	O/T	说明
PA3/INT0/STP1I/SDO/ TX/AN3/MODE	PA3	PAPU PAWU 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
	INT0	PAS0 INTC0 INTEG IFS1	ST	—	外部中断 0
	STP1I	PAS0	ST	—	STM1 捕捉输入
	SDO	PAS0	—	CMOS	SPI 串行数据输出
	TX	PAS0	—	CMOS	UART 发送脚
	AN3	PAS0	AN	—	A/D 转换器外部输入通道 3
	MODE	—	ST	—	压电蜂鸣片驱动器模式选择 – 反馈 / 非反馈
PA4/PTCK/STP0B/ AN0/A0O	PA4	PAPU PAWU 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
	PTCK	PAS1	ST	—	PTM 时钟输入或捕捉输入
	STP0B	PAS1	—	CMOS	STM0 反相输出
	AN0	PAS1	AN	—	A/D 转换器外部输入通道 0
	A0O	PAS1	—	AN	SD 运算放大器 0 输出
PA5/STCK0/STP1B/ A1O	PA5	PAPU PAWU 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
	STCK0	PAS1	ST	—	STM0 时钟输入
	STP1B	PAS1	—	CMOS	STM1 反相输出
	A1O	PAS1	—	AN	SD 运算放大器 1 输出
PA6/PTP/SDI/SDA/ RX/ENCLK	PA6	PAPU PAWU 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
	PTP	PAS1	—	CMOS	PTM 输出
	SDI	PAS1 IFS0	ST	—	SPI 串行数据输入
	SDA	PAS1 IFS0	ST	NMOS	I ² C 数据线
	RX	PAS1 IFS1	ST	—	UART 接收脚
	ENCLK	—	ST	—	压电蜂鸣片驱动器使能 / 时钟输入

引脚名称	功能	OPT	I/T	O/T	说明
PA7/STP0I/PTPI/SCK/ SCL/AN1/VREF	PA7	PAPU PAWU 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
	STP0I	PAS1	ST	—	STM0 捕捉输入
	PTPI	PAS1 IFS0	ST	—	PTM 捕捉输入
	SCK	PAS1 IFS0	ST	CMOS	SPI 串行时钟
	SCL	PAS1 IFS0	ST	NMOS	I ² C 时钟线
	AN1	PAS1	AN	—	A/D 转换器外部输入通道 1
	VREF	PAS1	AN	—	A/D 转换器外部参考电压
PB0/INT0/STP0/A0PB/ DACO	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	INT0	PBS0 INTC0 INTEG IFS1	ST	—	外部中断 0
	STP0	PBS0	—	CMOS	STM0 输出
	A0PB	PBS0	AN	—	SD 运算放大器 0 偏压输入
	DACO	PBS0	—	AN	16-bit D/A 转换器输出
PB1/SDO/TX	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SDO	PBS0	—	CMOS	SPI 串行数据输出
	TX	PBS0	—	CMOS	UART 发送脚
PB2/SCK/SCL/DACO	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SCK	PBS0 IFS0	ST	CMOS	SPI 串行时钟
	SCL	PBS0 IFS0	ST	NMOS	I ² C 时钟线
	DACO	PBS0	—	AN	16-bit 语音 D/A 转换器输出
PB3/SDI/SDA/RX	PB3	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SDI	PBS0 IFS0	ST	—	SPI 串行数据输入
	SDA	PBS0 IFS0	ST	NMOS	I ² C 数据线
	RX	PBS0 IFS1	ST	—	UART 接收脚
PB4/ $\overline{\text{SCS}}$ /AN2	PB4	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	$\overline{\text{SCS}}$	PBS1 IFS0	ST	CMOS	SPI 从机选择
	AN2	PBS1	AN	—	A/D 转换器外部输入通道 2

引脚名称	功能	OPT	I/T	O/T	说明
PB5/TX/STP0B/AN4	PB5	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TX	PBS1	—	CMOS	UART 发送脚
	STP0B	PBS1	—	CMOS	STM0 反相输出
	AN4	PBS1	AN	—	A/D 转换器外部输入通道 4
PB6/AN5/DACO	PB6	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AN5	PBS1	AN	—	A/D 转换器外部输入通道 5
	DACO	PBS1	—	AN	16-bit 语音 D/A 转换器输出
PB7/ $\overline{\text{SCS}}$ /AN6	PB7	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	$\overline{\text{SCS}}$	PBS1 IFS0	ST	CMOS	SPI 从机选择
	AN6	PBS1	AN	—	A/D 转换器外部输入通道 6
PC4/STCK1/PTPB	PC4	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	STCK1	PCS1	ST	CMOS	STM1 时钟输入
	PTPB	PCS1	—	CMOS	PTM 反相输出
ISINK0	ISINK0	—	—	AN	灌电流源 0
ISINK1	ISINK1	—	—	AN	灌电流源 1
A0NI	A0NI	—	AN	—	SD 运算放大器 0 反相输入
A0PI	A0PI	—	AN	—	SD 运算放大器 0 同相输入
FB	FB	—	ST	—	压电蜂鸣片驱动器反馈输入
VS	VS	—	—	CMOS	压电蜂鸣片驱动器输出
VB	VB	—	—	CMOS	压电蜂鸣片驱动器互补输出
VCC	VCC	—	PWR	—	LDO 和压电蜂鸣片驱动器电源电压
VREG	VREG	—	—	PWR	LDO 输出，在感烟探测器应用中，该引脚应与单片机正电源供电引脚 VDD 外部相连。
VDD/AVDD	VDD	—	PWR	—	数字正电源
	AVDD	—	PWR	—	A/D 转换器正电源
VSS/AVSS/VSS1/ VSS2	VSS	—	PWR	—	数字负电源，接地
	AVSS	—	PWR	—	A/D 转换器负电源，接地
	VSS1	—	PWR	—	灌电流发生器负电源，接地
	VSS2	—	PWR	—	灌电流发生器负电源，接地

注：I/T：输入类型；
OPT：通过寄存器选项来配置；
ST：施密特触发输入；
NMOS：NMOS 输出；

O/T：输出类型；
PWR：电源；
CMOS：CMOS 输出；
AN：模拟信号。

极限参数

电源供应电压 (V_{DD})	$V_{SS}-0.3V\sim6.0V$
电源供应电压 (V_{CC})	$-0.3V\sim16V$
端口输入电压	$V_{SS}-0.3V\sim V_{DD}+0.3V$
储存温度	$-60^{\circ}C\sim150^{\circ}C$
工作温度	$-40^{\circ}C\sim85^{\circ}C$
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、工作温度、工作频率、引脚负载状况、温度和程序指令等等。

工作电压特性

$T_a=-40^{\circ}C\sim85^{\circ}C$

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 – HIRC	$f_{SYS}=f_{HIRC}=2MHz$	—	3.3	—	V
		$f_{SYS}=f_{HIRC}=4MHz$	—	3.3	—	
		$f_{SYS}=f_{HIRC}=8MHz$	—	3.3	—	
	工作电压 – LIRC	$f_{SYS}=f_{LIRC}=32kHz$	—	3.3	—	V

工作电流特性

$T_a=-40^{\circ}C\sim85^{\circ}C$

符号	工作模式	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{DD}	低速模式 – LIRC	3.3V	$f_{SYS}=32kHz$	—	10	20	μA
	快速模式 – HIRC	3.3V	$f_{SYS}=2MHz$	—	0.2	0.3	mA
		3.3V	$f_{SYS}=4MHz$	—	0.4	0.6	mA
		3.3V	$f_{SYS}=8MHz$	—	0.8	1.2	mA

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. 无直流电流通路。
4. 所有工作电流数值都是通过连续的 NOP 指令循环测得。

待机电流特性

Ta=-40°C~85°C

符号	待机模式	测试条件		最小	典型	最大	最大 @85°C	单位
		V _{DD}	条件					
I _{STB}	休眠模式	3.3V	WDT on	—	1.5	3.0	3.6	μA
	空闲模式 0 – LIRC	3.3V	f _{SUB} on	—	3	5	6	μA
	空闲模式 1 – HIRC	3.3V	f _{SUB} on, f _{SYN} =2MHz	—	70	140	160	μA
		3.3V	f _{SUB} on, f _{SYN} =4MHz	—	110	220	240	μA
		3.3V	f _{SUB} on, f _{SYN} =8MHz	—	180	360	400	μA

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。

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 (3.3V) 条件下。

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 2MHz HIRC 频率	3.3V	25°C	-1%	2	+1%	MHz
			-20°C~60°C	-2%	2	+2%	
	通过烧录器调整后的 4MHz HIRC 频率	3.3V	25°C	-1%	4	+1%	MHz
	通过烧录器调整后的 8MHz HIRC 频率	3.3V	25°C	-1%	8	+1%	MHz

注：1. 烧录器可在 3.3V 的固定电压下对 HIRC 频率进行调整，在此提供 V_{DD}=3.3V 时的参数值。

2. 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 LIRC 电气特性

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	温度				
f _{LIRC}	LIRC 频率	3.3V	-40°C~85°C	-7%	32	+7%	kHz
t _{START}	LIRC 启动时间	—	-40°C~85°C	—	—	100	μs

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
t _{SST}	系统启动时间 (从 f _{sys} off 的状态下唤醒)	f _{sys} =f _H ~f _H /64, f _H =f _{HIRC}	—	16	—	t _{HIRC}
		f _{sys} =f _{SUB} =f _{LIRC}	—	2	—	t _{LIRC}
	系统启动时间 (从 f _{sys} on 的状态下唤醒)	f _{sys} =f _H ~f _H /64, f _H =f _{HIRC}	—	2	—	t _H
		f _{sys} =f _{SUB} =f _{LIRC}	—	2	—	t _{SUB}
	系统速度切换时间 (快速模式 → 低速模式或 低速模式 → 快速模式)	f _{HIRC} off → on	—	16	—	t _{HIRC}
t _{RSTD}	系统复位延迟时间 (上电复位或 LVR 硬件复位)	RR _{POR} =5V/ms	42	48	54	ms
	系统复位延迟时间 (WDTC 软件复位)	—				
	系统复位延迟时间 (WDT 溢出复位)	—	14	16	18	
t _{SRESET}	软件复位最小延迟脉宽	—	45	90	120	μs

- 注：1. 系统启动时间里提到的 f_{sys} on/off 状态取决于工作模式类型以及所选的系统时钟振荡器。更多相关细节请参考系统工作模式章节。
2. t_{HIRC} 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. 若 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=-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 _{IL}	I/O 口低电平输入电压	—	—	0	—	0.2V _{DD}	V
V _{IH}	I/O 口高电平输入电压	—	—	0.8V _{DD}	—	V _{DD}	V
I _{OL}	I/O 口灌电流	3.3V	V _{OL} =0.1V _{DD}	16	32	—	mA
I _{OH}	I/O 口源电流	3.3V	V _{OH} =0.9V _{DD} , SLEDCn[m+1:m]=00B (n=0, 1, m=0, 2, 4, 6)	-0.7	-1.5	—	mA
		3.3V	V _{OH} =0.9V _{DD} , SLEDCn[m+1:m]=01B (n=0, 1, m=0, 2, 4, 6)	-1.3	-2.5	—	
		3.3V	V _{OH} =0.9V _{DD} , SLEDCn[m+1:m]=10B (n=0, 1, m=0, 2, 4, 6)	-1.8	-3.6	—	
I _{OH}	I/O 口源电流	3.3V	V _{OH} =0.9V _{DD} , SLEDCn[m+1:m]=11B (n=0, 1, m=0, 2, 4, 6)	-4	-8	—	mA
R _{PH}	I/O 口上拉电阻 (注)	3.3V	—	20	60	100	kΩ
I _{LEAK}	输入漏电流	—	V _{IN} =V _{DD} 或 V _{IN} =V _{SS}	—	—	±1	μA

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
t _{TPI}	STM STPnI 输入引脚最小脉宽	—	—	0.3	—	—	μs
	PTM PTPI 输入引脚最小脉宽			0.1	—	—	
t _{TCK}	STM STCKn 输入引脚最小脉宽	—	—	0.3	—	—	μs
	PTM PTCK 输入引脚最小脉宽			0.3	—	—	
t _{INT}	中断引脚最小脉宽	—	—	10	—	—	μs
f _{TMCLK}	PTM 最大时钟源频率	—	—	—	—	1	f _{sys}
t _{CPW}	PTM 捕捉输入最小脉宽	—	—	2	—	—	t _{TMCLK}

注：1. R_{PH} 内部上拉电阻值的计算方法是：将引脚接地并设置为输入且使能上拉电阻功能，然后在特定电源电压下测量该引脚上的电流，最后电压除以测量的电流值从而得到此上拉电阻值。

2. t_{TMCLK}=1/f_{TMCLK}。

存储器电气特性

Ta=-40℃~85℃，除非另有说明。

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{RW}	读 / 写工作电压	—	—	V _{DDmin}	—	V _{DDmax}	V
程序存储器 / 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	ms
I _{DDPGM}	V _{DD} 电压下烧录 / 擦除电流	—	—	—	—	5.0	mA
E _P	电容耐久性 – Flash 程序存储器	—	—	10K	—	—	E/W
	电容耐久性 – 数据 EEPROM 存储器	—	—	100K	—	—	
t _{RETD}	程序存储器数据保存时间	—	Ta=25℃	—	40	—	Year
RAM 数据存储器							
V _{DR}	RAM 数据保存电压	—	单片机处于 SLEEP 模式	1.0	—	—	V

LVD & LVR 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 _{LVR}	低电压复位电压	—	LVR 使能	-5%	2.1	+5%	V
V _{LVD}	低电压检测电压	—	LVD 使能, 电压选择 2.0V	-5%	2.0	+5%	V
			LVD 使能, 电压选择 2.2V		2.2		
			LVD 使能, 电压选择 2.4V		2.4		
			LVD 使能, 电压选择 2.7V		2.7		
			LVD 使能, 电压选择 3.0V		3.0		
			LVD 使能, 电压选择 3.3V		3.3		
			LVD 使能, 电压选择 3.6V		3.6		
			LVD 使能, 电压选择 4.0V		4.0		
I _{LVRLVDBG}	工作电流	3.3V	LVD 使能, LVR 使能, VBGEN=0	—	—	20	μA
		3.3V	LVD 使能, LVR 使能, VBGEN=1	—	—	25	μA
t _{LVDS}	LVDO 稳定时间	—	LVR 使能时, VBGEN=0, LVD off → on	—	—	18	μs
			LVR 除能时, VBGEN=0, LVD off → on	—	—	150	
t _{LVR}	产生 LVR 复位的低电压最短保持时间	—	—	120	240	480	μs
t _{LVD}	产生 LVD 中断的低电压最短保持时间	—	—	60	120	240	μs
I _{LVR}	LVR 使能的额外电流	—	LVD 除能, VBGEN=0	—	—	24	μA
I _{LVD}	LVD 使能的额外电流	—	LVR 除能, VBGEN=0	—	—	24	μA

A/D 转换器电气特性

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 _{ADI}	输入电压	—	—	0	—	V _{REF}	V
V _{REF}	参考电压	—	—	2	—	V _{DD}	V
N _R	分辨精度	—	—	—	—	12	Bit
DNL	非线性微分误差	—	V _{REF} =V _{DD} , t _{ADCK} =0.5μs	-3	—	3	LSB
INL	非线性积分误差	—	V _{REF} =V _{DD} , t _{ADCK} =0.5μs	-4	—	4	LSB
I _{ADC}	A/D 转换器使能的额外电流	3.3V	无负载, t _{ADCK} =0.5μs	—	340	500	μA
t _{ADCK}	时钟周期	—	—	0.5	—	10.0	μs
t _{ON2ST}	A/D 转换器 On-to-Start 时间	—	—	4	—	—	μs
t _{ADC}	A/D 转换时间 (包括采样和保持时间)	—	—	—	16	—	t _{ADCK}

内部参考电压特性

Ta=-40°C~85°C，除非另有说明。

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{DD}	工作电压	—	—	2.2	—	5.5	V
V _{BGREF}	Bandgap 参考电压	—	—	-1.0%	1.2	+1.0%	V
I _{BGREF}	工作电流	3.3V	—	—	25	40	μA
PSRR	电源电压抑制比	—	Ta=25°C, V _{ripple} =1V _{P-P} , f _{ripple} =100Hz	75	—	—	dB
En	输出噪声	—	Ta=25°C, 无负载电流, f=0.1Hz~10Hz	—	300	—	μV _{RMS}
I _{DRV}	缓冲器驱动能力	—	ΔV _{BGREF} =-1%	1	—	—	mA
I _{SD}	关机电流	—	VBGREN=0	—	—	0.1	μA
t _{START}	启动时间	—	Ta=25°C	—	—	400	μs

注：1. 以上参数均应在无负载的条件下测得，除非有特别说明。

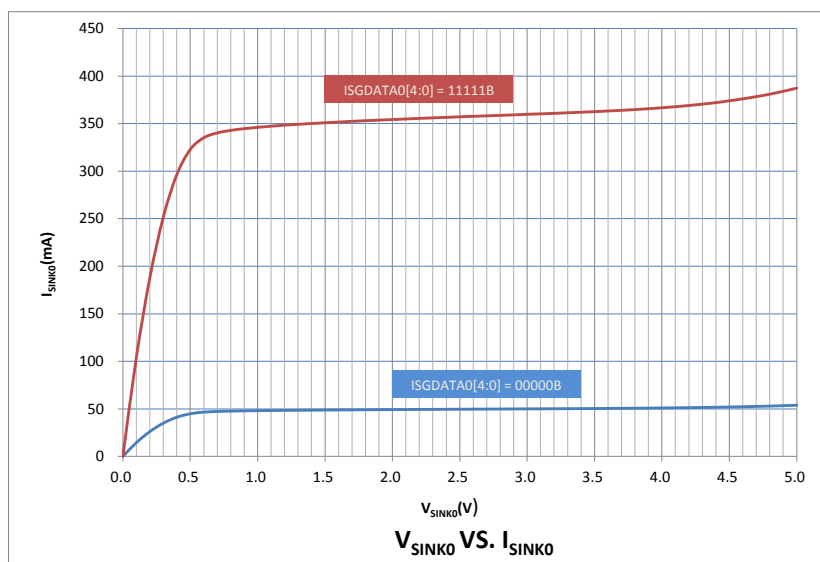
2. V_{DD} 引脚需连接一个 0.1μF 陶瓷电容到地。

3. V_{BGREF} 电压可以用作 A/D 转换器内部信号输入。

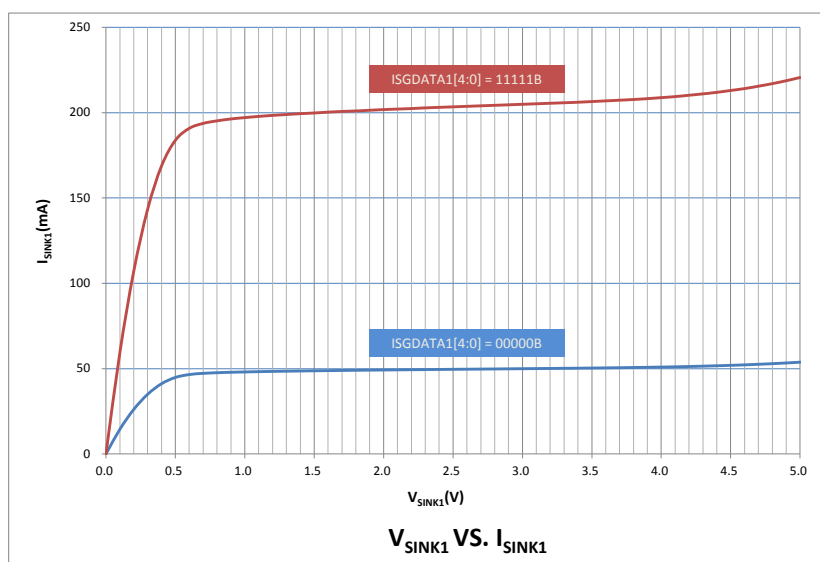
灌电流发生器电气特性

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
I _{SINK0}	ISINK0 引脚灌电流	—	Ta=-40°C~85°C, V _{ISINK0} =1.0V~3.3V, ISGDATA0[4:0]=00000B	41	50	59	mA
		—	Ta=-40°C~85°C, V _{ISINK0} =1.0V~3.3V, ISGDATA0[4:0]=11111B	295	360	425	
I _{SINK1}	ISINK1 引脚灌电流	—	Ta=-40°C~85°C, V _{ISINK1} =1.0V~3.3V, ISGDATA1[4:0]=00000B	41	50	59	mA
		—	Ta=-40°C~85°C, V _{ISINK1} =1.0V~3.3V, ISGDATA1[4:0]=11111B	168	205	242	

灌电流发生器电气特性曲线图



I_{SINK0} 电气特性曲线图



I_{SINK1} 电气特性曲线图

运算放大器电气特性 – 感烟探测器 AFE

$V_{DD}=3.3V$, $T_a=-40^{\circ}C\sim 85^{\circ}C$

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{OPA}	工作电流	—	SDAmBW[1:0]=00B (m=0, 1), 无负载	—	2.5	4.0	μA
			SDAmBW[1:0]=01B (m=0, 1), 无负载	—	10	16	
			SDAmBW[1:0]=10B (m=0, 1), 无负载	—	80	128	
			SDAmBW[1:0]=11B (m=0, 1), 无负载	—	200	320	
V_{OS}	输入失调电压	—	未校准 SDAmOF[5:0]=100000B (m=0, 1)	-15	—	+15	mV
			已校准	-2	—	+2	
I_{OS}	输入失调电流	—	$V_{IN}=1/2V_{CM}$	—	1	10	nA
V_{CM}	共模电压范围	—	SDAmBW[1:0]=00, 01, 10, 11 (m=0, 1)	V_{SS}	—	$V_{DD}-1.4$	V
PSRR	电源电压抑制比	—	SDAmBW[1:0]=00, 01, 10, 11 (m=0, 1)	50	70	—	dB
CMRR	共模抑制比	—	SDAmBW[1:0]=00, 01, 10, 11 (m=0, 1)	50	80	—	dB
A_{OL}	开环增益	—	SDAmBW[1:0]=00, 01, 10, 11 (m=0, 1)	60	80	—	dB
SR	转换速率	—	$R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, SDAmBW[1:0]=00 (m=0, 1)	0.5	1.5	—	V/ms
			$R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, SDAmBW[1:0]=01 (m=0, 1)	5	15	—	
			$R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, SDAmBW[1:0]=10 (m=0, 1)	180	500	—	
			$R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, SDAmBW[1:0]=11 (m=0, 1)	600	1800	—	
GBW	增益带宽	—	$R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, SDAmBW[1:0]=00 (m=0, 1)	2	5	—	kHz
		—	$R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, SDAmBW[1:0]=01 (m=0, 1)	15	40	—	
		—	$R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, SDAmBW[1:0]=10 (m=0, 1)	250	600	—	
		—	$R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, SDAmBW[1:0]=11 (m=0, 1)	800	2000	—	
V_{OR}	最大输出电压范围	—	SDAmBW[1:0]=00, 01 (m=0, 1) $R_{LOAD}=5k\Omega$ 接到 $V_{DD}/2$ 处	$V_{SS}+140$	—	$V_{DD}-160$	mV
			SDAmBW[1:0]=10, 11 (m=0, 1) $R_{LOAD}=5k\Omega$ 接到 $V_{DD}/2$ 处	$V_{SS}+120$	—	$V_{DD}-140$	

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
I _{SC}	输出短路电流	—	R _{LOAD} =5.1Ω, SDAmBW[1:0]=00, 01 (m=0, 1)	±1.2	±12.0	—	mA
			R _{LOAD} =5.1Ω, SDAmBW[1:0]=10, 11 (m=0, 1)	±2	±20	—	

注：表格中为特征值，未经实测。

16-bit 语音 D/A 转换器电气特性

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				
I _{DAC}	带缓冲器 D/A 转换器使能的额外电流	3.3V	—	—	—	3	mA
I _{STB(DAC)}	待机电流	—	DACEN=0	—	—	1	μA
THD+N	总谐波失真 + 噪音 (注)	3.3V	10kΩ 负载	—	-55	—	dB
V _{OUT}	输出电压范围	—	无负载	0.01	—	0.99	V _{DD}
t _{DACS}	D/A 转换器启动稳定时间	—	—	—	—	1	ms

注：正弦波输入 @ 1kHz, -6dBFS。

LDO 电气特性

V_{CC}=(V_{REG}+1V), Ta=25°C, C_{REG}=10μF(E-CAP), 除非另有说明

符号	参数	测试条件	最小	典型	最大	单位
V _{CC}	输入电压	V _{REG} =3.3V, 无负载	4.3	—	12	V
V _{REG}	输出电压	I _{REG} =1mA	3.201	3.300	3.399	V
I _{REG}	输出电流	ΔV _{REG} =-3%	30	—	—	mA
ΔV _{REG}	负载调整	1mA ≤ I _{REG} ≤ 10mA	—	15	45	mV
I _{SS}	静态电流	I _{REG} =0mA	—	3.0	5.0	μA
$\frac{\Delta V_{REG}}{\Delta V_{CC} \times V_{REG}}$	线性调整率	(V _{REG} +2V) ≤ V _{CC} ≤ 12V, I _{REG} =1mA	—	0.1	0.2	%/V
$\frac{\Delta V_{REG}}{\Delta T_a \times V_{REG}}$	温度系数	I _{REG} =1mA, Ta=-40°C ~ 85°C	—	±100	—	ppm/°C

注：在实际应用中 C_{REG} 为 10μF~100μF E-CAP。

压电蜂鸣片驱动器电气特性

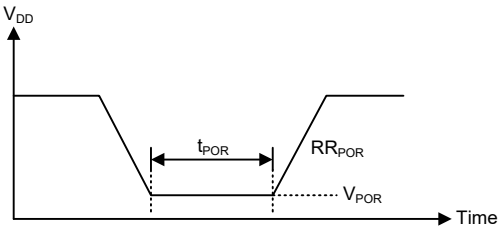
$V_{CC}=9V$, $V_{REG}=3.3V$, $T_a=25^{\circ}C$, 除非另有说明

符号	参数	测试条件		最小	典型	最大	单位
		测试引脚	条件				
V_{IH}	高电平输入电压	FB	—	$0.8V_{CC}$	—	—	V
		MODE, ENCLK	—	$0.7V_{REG}$	—	—	V
V_{IL}	低电平输入电压	FB	—	—	—	$0.2V_{CC}$	V
		MODE, ENCLK	—	—	—	$0.3V_{REG}$	V
I_{OH}	源电流	VB, VS	$V_{OH}=0.9V_{CC}$,	-40	-50	—	mA
I_{OL}	灌电流	VB, VS	$V_{OL}=0.1V_{CC}$	80	100	—	mA
I_{LEAK}	输入漏电流	FB	$V_{CC}=9V$ 或 $V_{CC}=V_{SS}$	—	—	± 0.1	μA
		MODE, ENCLK	$V_{CC}=3.3V$ 或 $V_{CC}=V_{SS}$	—	—	± 0.1	μA
R_{PU}	上拉电阻	MODE, ENCLK	—	0.7	1	1.3	$M\Omega$

上电复位特性

$T_a=-40^{\circ}C\sim 85^{\circ}C$

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



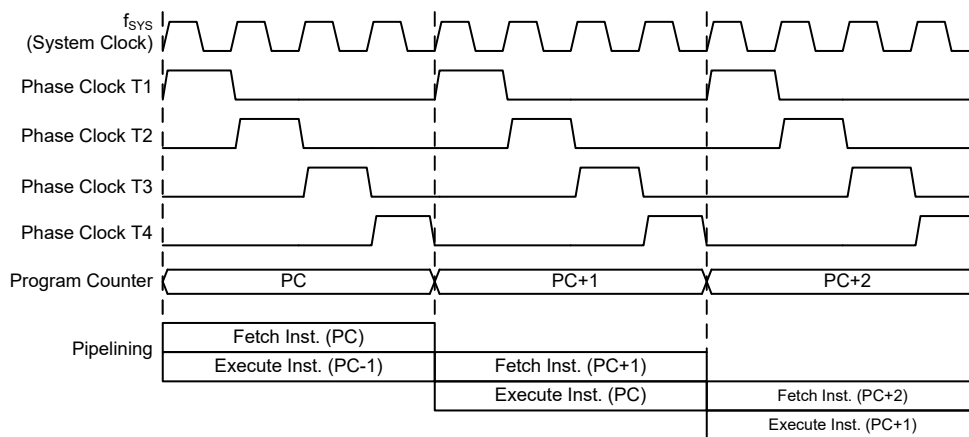
系统结构

内部系统结构是 Holttek 单片机具有良好性能的主要因素。由于采用 RISC 结构，此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一个以上指令周期外，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 位 ALU 参与指令集中所有的运算，它可完成算术运算、逻辑运算、移位、递增、递减和分支等功能，而内部的数据路径则是以通过累加器和 ALU 的方式加以简化。有些寄存器在数据存储器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供具有较大可靠度和灵活性的 I/O 和 A/D 控制系统时，仅需要少数的外部器件。使得这些单片机适用于低成本和批量生产的控制应用。

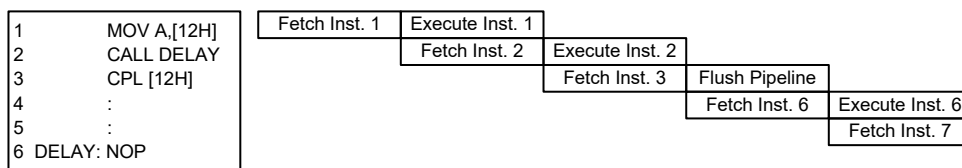
时序和流水线结构

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

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



系统时序和流水线



指令捕捉

程序计数器

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

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

程序计数器	
程序计数器高字节	PCL 寄存器
PC12~PC8	PCL7~PCL0

程序计数器

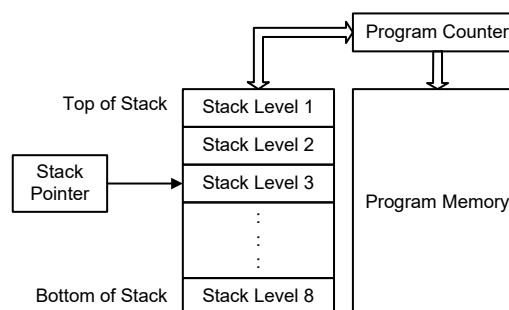
程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

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。该单片机有 8 层堆栈，堆栈既不是数据部分也不是程序空间部分，而且它既不是可读取也不是可写入的。当前层由堆栈指针 (SP) 加以指示，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：

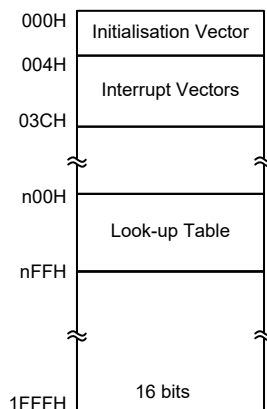
- 算术运算：
 - ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
 - LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- 逻辑运算：
 - AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
 - LAND, LOR, LXOR, LANDM, LORM, LXORM, LCPL, LCPLA
- 移位运算：
 - RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
 - LRRA, LRR, LRRCA, LRRC, LRLA, 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, LSDZ, LSIZA, 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，此单片机提供用户灵活便利的调试方法和项目开发规划及更新。

结构

程序存储器的容量为 8K×16 位，程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口。数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。



程序存储器结构

特殊向量

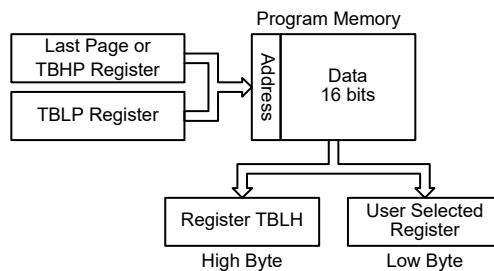
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 0000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

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“1F00H”指向的地址是 8K 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 1F06H，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”指令被使用，则表格指针指向 TBLP 和 TBHP 指定的地址。在这个例子中，表格数据的高字节等于零，而当“TABRD [m]”或“LTABRD”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 / 可写寄存器，且能重复储存，若主程序和中断服务程序都使用表格读取指令，应该注意它的保护。使用表格读取指令，中断服务程序可能会改变 TBLH 的值，若随后在主程序中再次使用这个值，则会发生错误，因此建议避免同时使用表格读取指令。然而在某些情况下，如果同时使用表格读取指令是不可避免的，则在执行任何主程序的表格读取指令前，中断应该先除能，另外要注意的是所有与表格相关的指令，都需要两个指令周期去完成操作。

表格读取程序范例

```

tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
:

```

```

mov a,06h          ; initialise low table pointer - note that this address
                   ; is referenced
mov tblp,a         ; to the last page or the page that tbhp pointed
mov a,1Fh          ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1     ; transfers value in table referenced by table pointer
                   ; data at program memory address "1F06H" 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 "1F05H" transferred to
                   ; tempreg2 and TBLH, in this example the data "1AH" is
                   ; transferred to tempreg1 and data "0FH" to register
                   ; tempreg2
:
:
org 1F00h          ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:

```

在线烧录 – ICP

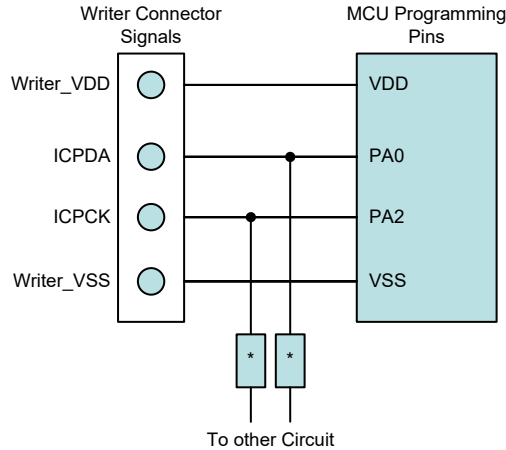
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。另外，Holtek 单片机提供 4 线接口的在线烧录方式。用户可将进行过烧录或未经过烧录的单片机芯片连同电路板一起制成，最后阶段进行程序的更新和程序的烧写，在无需去除或重新插入芯片的情况下方便地保持程序为最新版。

Holtek Flash MCU 与烧录器引脚对应表如下：

Holtek 烧录器引脚名称	MCU 在线烧录引脚名称	功能
ICPDA	PA0	串行数据 / 地址烧录
ICPCK	PA2	串行时钟
VDD	VDD	电源
VSS	VSS	地

程序存储器可以通过 4 线的接口在线进行烧录。其中一条线用于数据串行下载或上传、一条线用于串行时钟、剩下两条用于提供电源。芯片在线烧写的详细使用说明超出此文档的描述范围，将由专门的参考文献提供。

烧录过程中，用户必须确保 ICPDA 和 ICPCK 这两个引脚没有连接至其它输出脚。



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

片上调试 – OCDS

EV 芯片用于单片机仿真。此 EV 芯片提供片上调试功能 (On-Chip Debug) 用于开发过程中的实际单片机调试。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 OCSDSA 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。OCSDSA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户使用 EV 芯片进行调试时，实际单片机 OCSDSA 和 OCDSCK 引脚上的其它共用功能无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。关于 OCDS 功能的详细描述，请参考“Holtek e-Link for 8-bit MCU OCDS 使用手册”文件。

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

在线应用编程 – IAP

Flash 型程序存储器便于用户在同一芯片上对程序进行更新和修改。单片机提供的 IAP 功能使用户可以方便地对 Flash 程序存储器进行多次编程。IAP 功能可以通过内部固件进行程序的更新，而无需外接烧录器或 PC。此外，IAP 接口通过 I/O 引脚可以设置为任何类型的通信协议，例如 UART。关于内部固件，用户可以选择 Holtek 提供的版本或创建自己的内部固件。以下章节说明了如何执行 IAP 固件程序。

Flash 存储器读取 / 写入容量

Flash 存储器以页为单位进行擦 / 写操作，以字为单位进行读出操作。页的大小和写入缓冲器的大小都为 32 字。注意，在执行写入操作之前必须先执行擦除操作。

Flash 存储器擦 / 写功能成功使能时 CFWEN 位会被硬件置高，当该位被置高，便可写入数据到“写入缓冲器”。FWT 位用于启动写入程序，并指示写入操作的状态。当该位由应用程序置高时将开始一个写入程序，当写入操作结束后该位将由硬件清零。

读出操作是通过一个特定的读出程序来执行的。FRDEN 位用于使能读出功能，由应用程序设置 FRD 位来启动读出程序，并指示读出操作的状态。当读出操作结束后该位将由硬件清零。

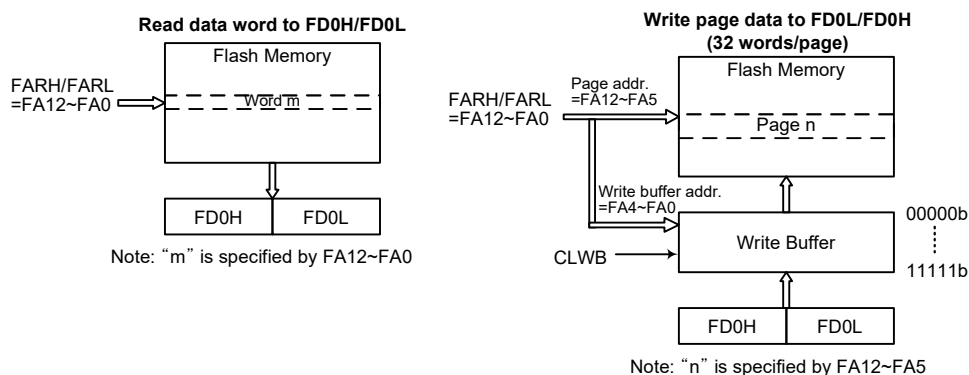
操作	格式
擦除	1 页 / 次
写入	32 字 / 次
读出	1 字 / 次
注：页大小 = 写入缓冲器大小 = 32 字	

IAP 操作格式

擦除页	FARH	FARL[7:5]	FARL[4:0]
0	0000 0000	000	x xxxx
1	0000 0000	001	x xxxx
2	0000 0000	010	x xxxx
3	0000 0000	011	x xxxx
4	0000 0000	100	x xxxx
:	:	:	:
:	:	:	:
254	0001 1111	110	x xxxx
255	0001 1111	111	x xxxx

“x”：无关

擦除页序号及选择



Flash 存储器 IAP 读 / 写结构

写入缓冲器

执行写入操作时写入缓冲器用于临时存储写入的数据。通过执行 Flash 存储器擦 / 写使能步骤成功使能 Flash 存储器擦 / 写功能后，才可将要写入的数据填入到写入缓冲器。通过配置 FC2 寄存器中的 CLWB 位可以清除写入缓冲器。置高 CLWB 位可以使能清除写入缓冲器程序，完成后该位会被硬件自动清零。建议第一次使用写入缓冲器或更新写入缓冲器内的数据时，应先置高 CLWB 位将写入缓冲器清零。

写入缓冲器的大小为每页 32 字。写入缓冲器的地址与存储器地址位 FA12~FA5 指定的 Flash 存储器页的地址相对应。写入到 FD0L 和 FD0H 寄存器的数据会被加载到写入缓冲器。当写入数据到高字节数据寄存器 FD0H 时，会将存储在

FD0L 和 FD0H 数据寄存器内的数据都加载到写入缓冲器，并使 Flash 存储器地址自动加一，之后新的地址会被加载到 FARH 和 FARL 地址寄存器。当 Flash 存储器地址到达当前页的最大地址，即 32 字的页为 11111b，地址将不再增加，并停在该页的最后一个地址，此时需要再设定一个新的页地址才可进行其它擦 / 写操作。

写入程序结束后，硬件会自动清除写入缓冲器。注意，如果在比对步骤时发现写入到 Flash 存储器的数据不正确，则需通过应用程序手动清除写入缓冲器，在写入缓冲器被清零之后再重新对其写入数据。

IAP Flash 程序存储器寄存器

与 IAP 相关的 Flash 存取寄存器有两个地址寄存器、四对 16-bit 数据寄存器和三个控制寄存器。使用地址、数据和控制寄存器可以对 Flash 存储器执行 16 位数据读 / 写操作。内部 Flash 程序存储器所有操作由一系列寄存器控制，即地址寄存器 FARL 和 FARH，数据寄存器 FDnL 和 FDnH，控制寄存器 FC0、FC1 和 FC2。

寄存器名称	位							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	—	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	—	—	—	FA12	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP 寄存器列表

• FARL 寄存器

Bit	7	6	5	4	3	2	1	0
Name	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
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 FA7~FA0: Flash 存储器地址 bit 7 ~ bit 0

● FARH 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	FA12	FA11	FA10	FA9	FA8
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 未定义，读为“0”

Bit 4~0 **FA12~FA8**: Flash 存储器地址 bit 12 ~ bit 8

● FD0L 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**: 第一个 Flash 存储器数据 bit 7 ~ bit 0

注意写入低字节数据寄存器 FD0L 的数据只能存储在 FD0L 寄存器，不会加载到低 8 位写入缓冲器。

● FD0H 寄存器

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8**: 第一个 Flash 存储器数据 bit 15 ~ bit 8

注意当写入 8 位数据到高字节数据寄存器 FD0H 时，存储在 FD0H 和 FD0L 寄存器内的 16 位数据将同时加载到 16 位写入缓冲器中，此时 Flash 存储器地址寄存器 FARH 和 FARL 的内容将自动加一。

● FD1L 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**: 第二个 Flash 存储器数据 bit 7 ~ bit 0

● FD1H 寄存器

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8**: 第二个 Flash 存储器数据 bit 15 ~ bit 8

● **FD2L 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:** 第三个 Flash 存储器数据 bit 7 ~ bit 0

● **FD2H 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8:** 第三个 Flash 存储器数据 bit 15 ~ bit 8

● **FD3L 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:** 第四个 Flash 存储器数据 bit 7 ~ bit 0

● **FD3H 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8:** 第四个 Flash 存储器数据 bit 15 ~ bit 8

● **FC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
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 **CFWEN:** Flash 存储器擦 / 写功能使能控制

- 0: Flash 存储器擦 / 写功能除能
- 1: Flash 存储器擦 / 写功能已成功使能

当此位由应用程序清零后, Flash 存储器擦 / 写功能除能。注意, 对此位直接写“1”不会使能擦 / 写功能。此位可用于指示 Flash 存储器擦 / 写功能状态。当此位由硬件置为“1”时, 表明 Flash 存储器擦 / 写功能已经成功使能, 若为“0”, 表明 Flash 存储器擦 / 写功能除能。

Bit 6~4 **FMOD2~FMOD0:** Flash 存储器模式选择

- 000: 写入模式
- 001: 页擦除模式
- 010: 保留
- 011: 读出模式
- 100: 保留

101: 保留
110: Flash 存储器擦 / 写使能模式
111: 保留

这几位用于选择 Flash 存储器的操作模式。注意在执行擦 / 写 Flash 存储器操作之前必须先成功使能“Flash 存储器擦 / 写使能模式”。

- Bit 3 **FWPEN**: Flash 存储器擦 / 写使能程序触发控制位
0: 擦 / 写使能程序未被触发或程序定时器发生溢出
1: 擦 / 写使能程序被触发且程序定时器开始计时
该位用于启动 Flash 存储器擦 / 写使能程序和内部定时器。此位由应用程序置高, 当内部定时器计时溢出后由硬件清零。需在 FWPEN 置高后尽快写入正确数据序列到 FD1L/FD1H、FD2L/FD2H 和 FD3L/FD3H 寄存器。
- Bit 2 **FWT**: Flash 存储器写入控制位
0: 未开始 Flash 存储器写入程序或 Flash 存储器写入程序已完成
1: 开始 Flash 存储器写入程序
此位由软件置“1”, 当 Flash 存储器写入程序完成后由硬件清零。
- Bit 1 **FRDEN**: Flash 存储器读出使能位
0: Flash 存储器读出除能
1: Flash 存储器读出使能
此位为 Flash 存储器读出使能位, 在执行 Flash 存储器读出操作之前需将此位置高。将此位清零则禁止 Flash 存储器读出操作。
- Bit 0 **FRD**: Flash 存储器读出控制位
0: 未开始 Flash 存储器读出程序或 Flash 存储器读出程序已完成
1: 开始 Flash 存储器读出程序
此位由软件置“1”, 当 Flash 存储器读出程序完成后由硬件清零。

注: 1. 在同一条指令中 FWT、FRDEN 和 FRD 位不可同时设置为“1”。
2. 确保 f_{SUB} 时钟在执行擦或写动作前已稳定。
3. 当读、擦或写动作成功启动后, CPU 相关操作将停止。
4. 确保读、擦或写动作成功完成后才可执行其它操作。

● FC1 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**: 整个芯片复位
当写入“55H”到该寄存器中, 将产生一个复位信号将整个单片机复位。

● FC2 寄存器

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

Bit 7~1 未定义, 读为“0”

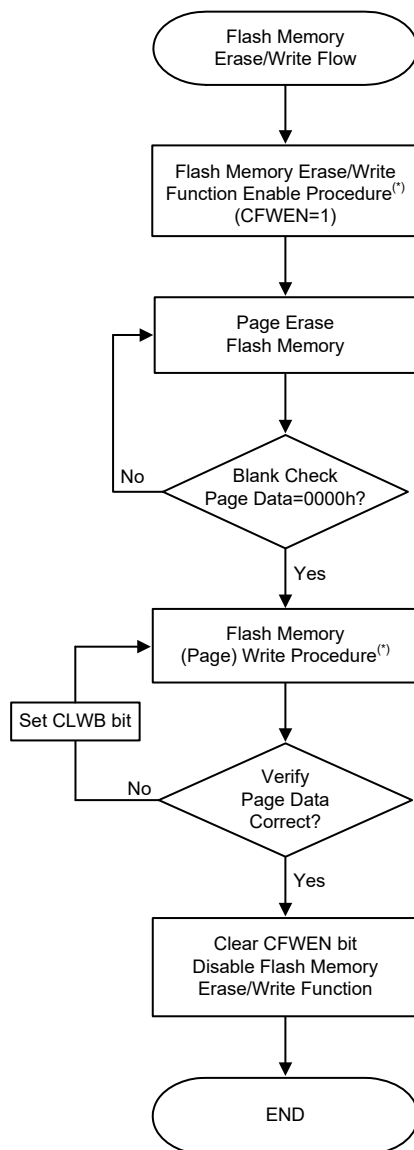
Bit 0 **CLWB**: Flash 存储器写入缓冲器清除控制位
0: 未开始写入缓冲器清除或写入缓冲器清除程序已完成
1: 开始写入缓冲器清除程序
此位由软件置“1”, 当写入缓冲器清除过程完成后由硬件清零。

Flash 存储器擦 / 写流程

在开始更新 Flash 存储器之前，先了解 Flash 存储器擦 / 写流程操作是很重要的，用户可参考下列步骤进行 IAP 程序开发，以确保 Flash 存储器内容更新正确。

Flash 存储器擦 / 写流程说明

1. 先启动“Flash 存储器擦 / 写使能程序”。当 Flash 存储器擦 / 写功能成功使能后，FC0 寄存器中的 CFWEN 位会由硬件自动置高，此时才可执行 Flash 存储器擦或写操作。详细内容请参考“Flash 存储器擦 / 写使能步骤”。
2. 配置 Flash 存储器地址以指定要擦除的页，然后擦除此页。
3. 查空确认是否擦除成功，可采用 TABRD 指令进行读取并比对是否为“0000h”，如果擦除不成功返回步骤 2 再执行页擦除。
4. 写入数据至该页，详细内容请参考“Flash 存储器写入步骤”。
5. 采用 TABRD 指令进行读取并比对写入数据是否正确，如果读出的数据与写入数据不符，即写入不成功，设置 CLWB 位为“1”清除“写入缓冲器”再返回步骤 4，再写入相同数据。
6. 完成当前页擦 / 写后，如果无需擦 / 写其它页，可清除 CFWEN 位来除能“Flash 存储器擦 / 写使能模式”。



Flash 存储器擦 / 写流程

注：* “Flash 存储器擦 / 写使能步骤” 和 “Flash 存储器写入步骤” 将在后面介绍。

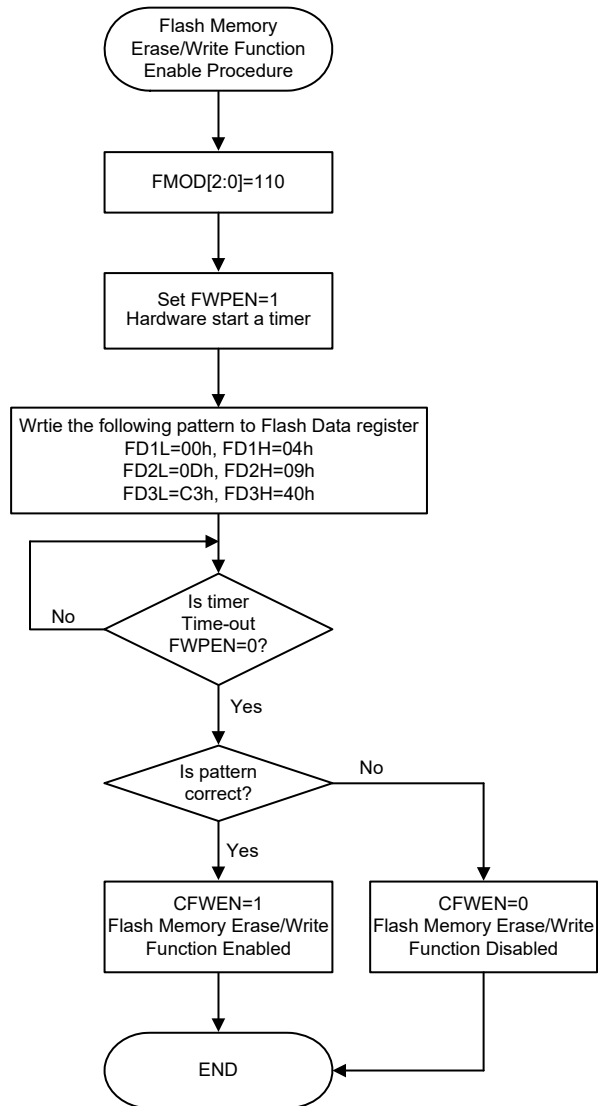
Flash 存储器擦 / 写使能步骤

Flash 存储器擦 / 写使能模式是专门为保护 Flash 存储器内容不被轻易地修改而设计的。用户必须先使能 Flash 存储器擦 / 写功能，才能通过 IAP 控制寄存器来更改 Flash 存储器数据。

Flash 存储器擦 / 写使能步骤说明

1. 写入数值“110”至 FC0 寄存器中的 FMOD[2:0] 位，选择 Flash 存储器擦 / 写使能模式。
2. 设置 FC0 寄存器中的 FWPEN 位为“1”，启动 Flash 存储器擦 / 写使能程序，此时内部硬件线路会启动一个内部定时器。
3. 使用者必须在 FWPEN 位置高后尽快填入以下数据序列至 FD1L~FD3L 和 FD1H~FD3H 寄存器中，数据序列依次为 FD1L=00h、FD1H=04h、FD2L=0Dh、FD2H=09h、FD3L=C3h、FD3H=40h。
4. 一旦定时器计时结束，无论写入的数据序列是否正确，FWPEN 位将由硬件自动清零。
5. 如果写入的数据序列不正确，表示 Flash 存储器擦 / 写功能没有成功使能，需重复以上步骤。如果写入的数据序列正确，表示 Flash 存储器擦 / 写功能成功使能。
6. 一旦 Flash 存储器擦 / 写功能成功使能，即可通过 IAP 控制寄存器进行页擦 / 写操作来更新 Flash 存储器内容。

将 FC0 寄存器中的 CFWEN 位清零，可除能 Flash 存储器擦 / 写功能，此时不必再执行以上步骤。



Flash 存储器擦 / 写使能步骤

Flash 存储器写入步骤

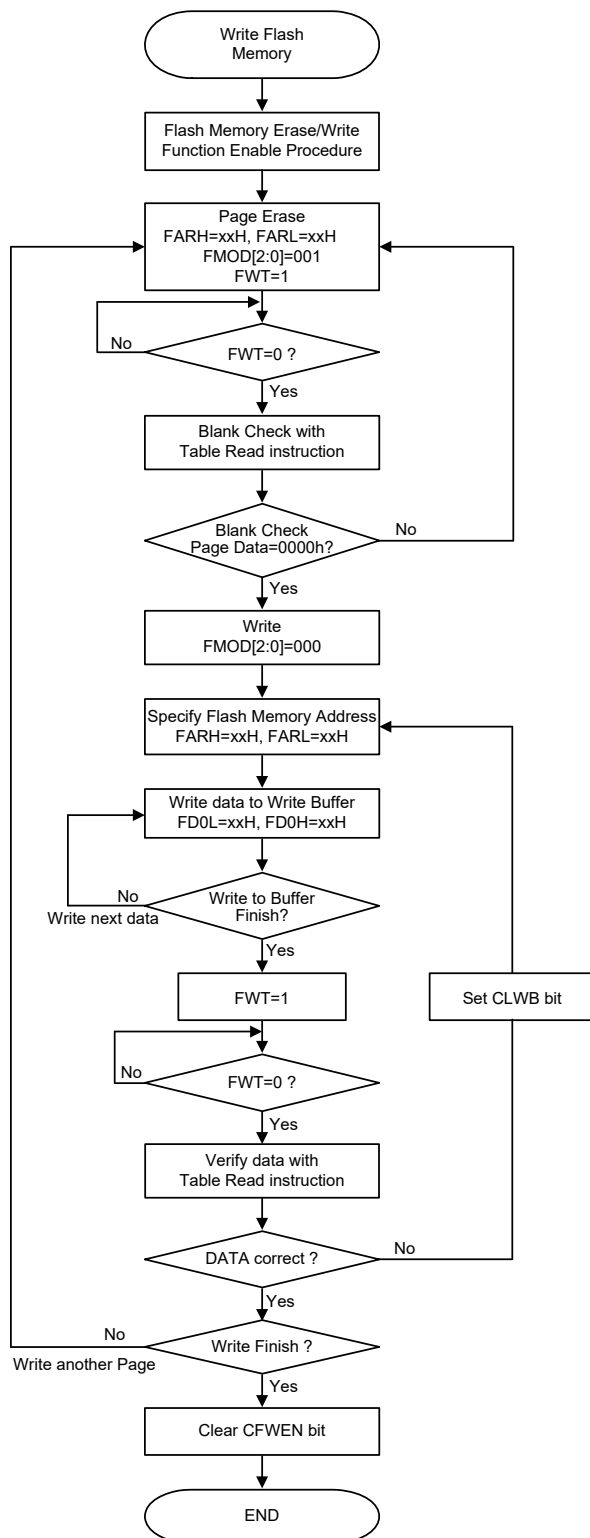
当 Flash 擦 / 写功能成功使能后，CFWEN 位会被硬件置高，此时要写入 Flash 存储器的数据才能加载到写入缓冲器。在开始写入程序之前，应先正确配置 IAP 控制寄存器，将所选的 Flash 存储器页的数据擦除。

写入缓冲器的大小为每页 32 个字，其地址与 FA12~FA5 指定的 Flash 存储器页的地址为相对应关系。注意，写入缓冲器的地址与对应存储器的地址必须在相同页。

Flash 存储器连续地址写入步骤说明

对于写入操作每次写入的数据最多为 32 字。多笔连续地址的数据写入时，写入缓冲器的地址将自动加 1。用户只需将第一笔数据的地址填入 FARL 和 FARH，并将第一笔数据依序填入 FD0L 和 FD0H 寄存器。先写 FD0L 再写 FD0H，才会将 FD0L 和 FD0H 数据一起填入写入缓冲器。写入缓冲器的地址将自动加 1，因此，要填入第二笔数据时，可不用修改 FARL 和 FARH 重新指定地址。当连续地址到达当前页的最后一个地址时，写入缓冲器的地址将不会再自动加“1”，保持在最后一个地址。

1. 启动 Flash 存储器擦 / 写使能程序，确认 CFWEN 的值，如果 CFWEN 被硬件置高，表示可进行 IAP 擦 / 写操作。详细内容请参考“Flash 存储器擦 / 写使能步骤”。
2. 设定 FMOD[2:0] 为“001”，选择擦除模式。设定 FWT 位为“1”，擦除 FARH 和 FARL 指定的目标页，直到 FWT 变为“0”。
3. 通过查表指令读出方式进行查空，以确保擦除操作已成功完成。
如果擦除操作不成功则返回步骤 2。
如果擦除操作成功则接着执行步骤 4。
4. 设定 FMOD[2:0] 为“000”，选择写入模式。
5. 先将目标起始地址写入 FARL 和 FARH 寄存器中，将要往连续地址所在页写入的数据依序写入 FD0L 和 FD0H 寄存器。最多可写入 32 个字。
6. 设定 FWT 位为“1”，将写入缓冲器的数据写入到对应的 Flash 存储器中，直到 FWT 变为“0”。
7. 通过查表指令读出方式进行数据比对，以确保写入操作已成功完成。
如果写入操作不成功，设置 CLWB 位为“1”清除写入缓冲器，再返回步骤 5。
如果写入操作成功则接着执行步骤 8。
8. 将 CFWEN 位清零以除能 Flash 存储器擦 / 写功能。



Flash 存储器连续地址写入步骤

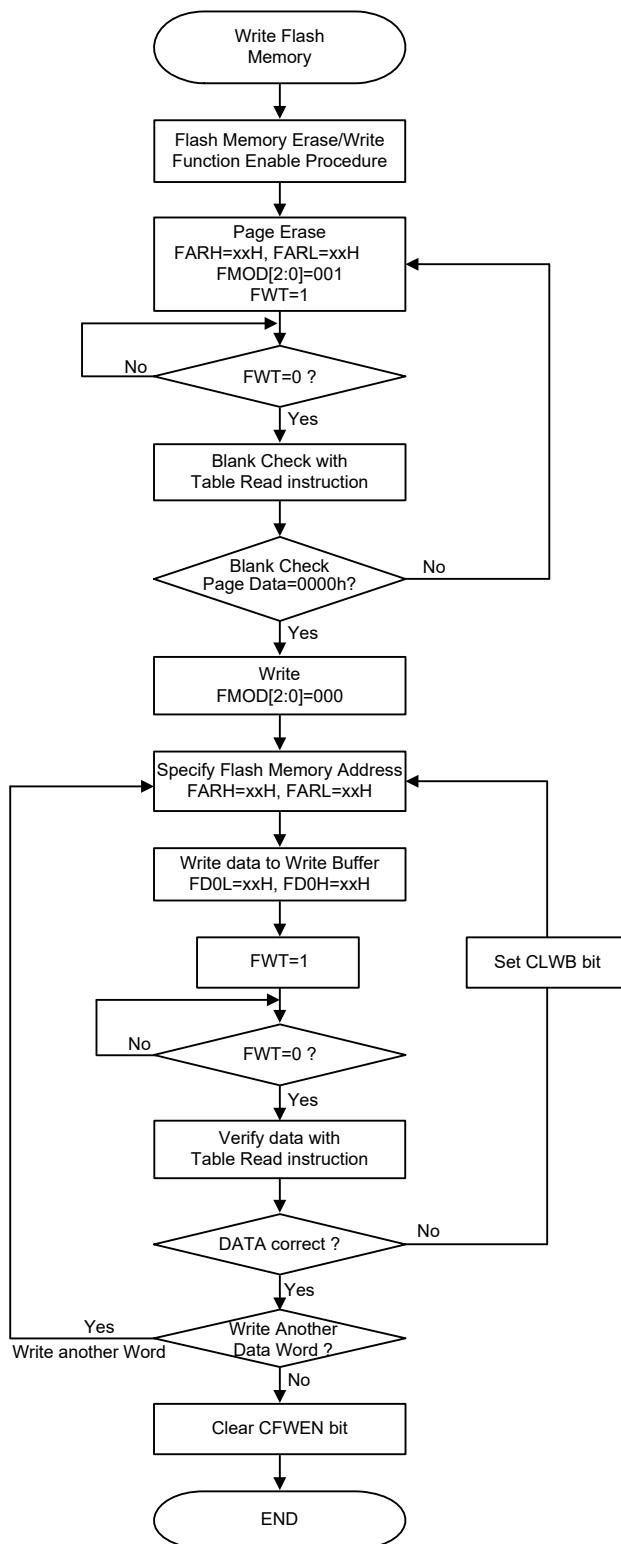
注：1. 当擦或写动作成功启动后，所有 CPU 相关操作将暂停。
2. FWT 位由高变低所需时间为 2.2ms (典型值)。

Flash 存储器非连续地址写入步骤说明

连续地址写入操作与非连续地址写入操作的主要差别在于要写入的数据是否位于连续地址。如果要写入的数据不是位于连续的地址，当一笔数据成功写入到 Flash 存储器后需重新配置另一个目标地址。

以两笔非连续的数据写入操作为例，说明如下：

1. 启动 Flash 存储器擦 / 写使能程序，确认 CFWEN 位的值，如果 CFWEN 被硬件置高，表示可进行 IAP 擦 / 写操作。详细内容请参考“Flash 存储器擦写使能步骤”。
2. 设定 FMOD[2:0] 为“001”，选择擦除模式。设定 FWT 位为“1”，擦除 FARH 和 FARL 指定的目标页，直到 FWT 变为“0”。
3. 通过查表指令读出方式进行查空，以确保擦除操作已成功完成。
如果擦除操作不成功则返回步骤 2。
如果擦除操作成功则接着执行步骤 4。
4. 设定 FMOD[2:0] 为“000”，选择写入模式。
5. 先将目标地址 ADDR1 写入 FARL 和 FARH 寄存器中，将要写入的数据 DATA1 先写入 FD0L 寄存器再写入 FD0H 寄存器。
6. 设定 FWT 位为“1”，将写入缓冲器的数据写入到对应的 Flash 存储器中，直到 FWT 变为“0”。
7. 通过查表指令读出方式进行数据比对，以确保写入操作已成功完成。
如果写入操作不成功，设置 CLWB 位为“1”清除写入缓冲器，再返回步骤 5。
如果写入操作成功则接着执行步骤 8。
8. 再将目标地址 ADDR2 写入 FARL 和 FARH 寄存器中，将要写入的数据 DATA2 先写入 FD0L 寄存器再写入 FD0H 寄存器。
9. 设定 FWT 位为“1”，将写入缓冲器的数据写入到对应的 Flash 存储器中，直到 FWT 变为“0”。
10. 通过查表指令读出方式进行数据比对，以确保写入操作已成功完成。
如果写入操作不成功，设置 CLWB 位为“1”清除写入缓冲器，再返回步骤 8。
如果写入操作成功则接着执行步骤 11。
11. 将 CFWEN 位清零以除能 Flash 存储器擦 / 写功能。



Flash 存储器非连续地址写入步骤

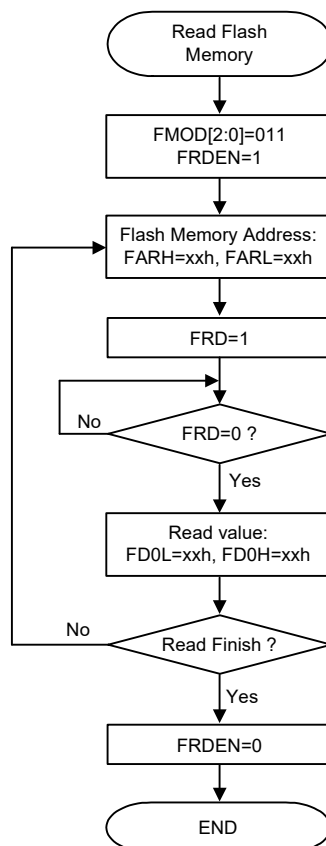
- 注：1. 当擦或写动作成功启动后，所有 CPU 相关操作将暂停。
2. FWT 位由高变低所需时间为 2.2ms (典型值)。

Flash 存储器写入操作注意事项

1. 要开始对 Flash 存储器进行 IAP 擦 / 写操作之前，必须先完成“Flash 存储器擦 / 写使能步骤”。
2. Flash 存储器擦除操作以页为单位进行擦除。
3. 写入缓冲器中的数据填入 Flash 存储器是以页为单位进行的，且写入时不可跨页填写。
4. 数据写入 Flash 存储器后，必须以查表指令“TABRD”读出方式比对所写数据是否正确，若比对发现写入数据不正确时，通过置高 CLWB 位将写入缓冲器清除，然后重新写入数据。无需清除对应的 Flash 存储器页，直接再写入，然后再比对，直到写入正确。
5. IAP 写入与数据比对时需与最高应用频率相同。

Flash 存储器读出步骤

要启动 Flash 存储器读出程序，需将 FMODE[2:0] 位设为“011”选择 Flash 存储器读出模式，将 FRDEN 位设为“1”使能读出功能。将要读出的地址填入 FARH 和 FARL 地址寄存器中，并将 FRD 位设为“1”，然后便可开始 Flash 存储器读出操作。当 FRD 被硬件清为“0”时，则可从 FD0H 和 FD0L 寄存器中取得 Flash 存储器中该地址数据。进行 Flash 存储器读出操作前，无需执行 Flash 存储器擦 / 写使能步骤。



Flash 存储器读出步骤

- 注：1. 当读动作成功启动后，所有 CPU 相关操作将暂停。
2. FRD 位由高变低所需时间为 3 个指令周期（典型值）。

数据存储器

数据存储器是内容可更改的 8-bit RAM 内部存储器，用来储存临时数据。

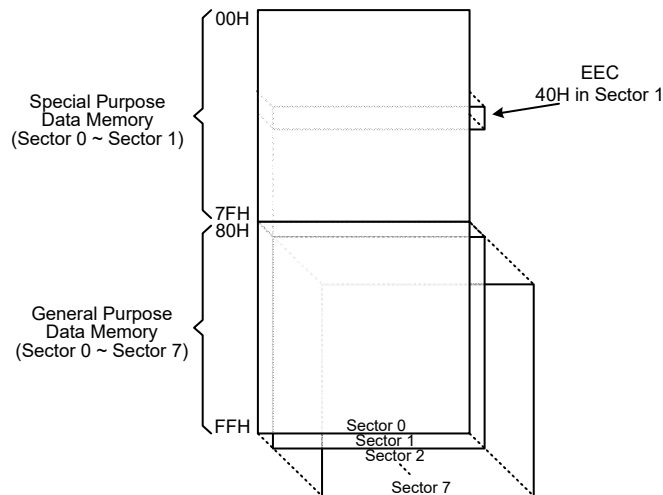
数据存储器分为两部分，第一部分是特殊功能数据存储器。这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二部分数据存储器是做一般用途使用，都可在程序控制下进行读取和写入。

若用间接寻址方式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。

结构

数据存储器被分为多个 Sector，都位于 8 位存储器中。每个数据存储器 Sector 分为两类，特殊功能数据存储器 and 通用数据存储器。特殊功能数据存储器地址范围为 00H~7FH，而通用数据存储器地址范围为 80H~FFH。

特殊功能数据存储器	通用数据存储器	
所在 Sector	容量	Sector: 地址
0: 00H~7FH 1: 40H (仅 EEC)	1024×8	0: 80H~FFH
		1: 80H~FFH
		:
		7: 80H~FFH



数据存储器结构

数据存储器寻址

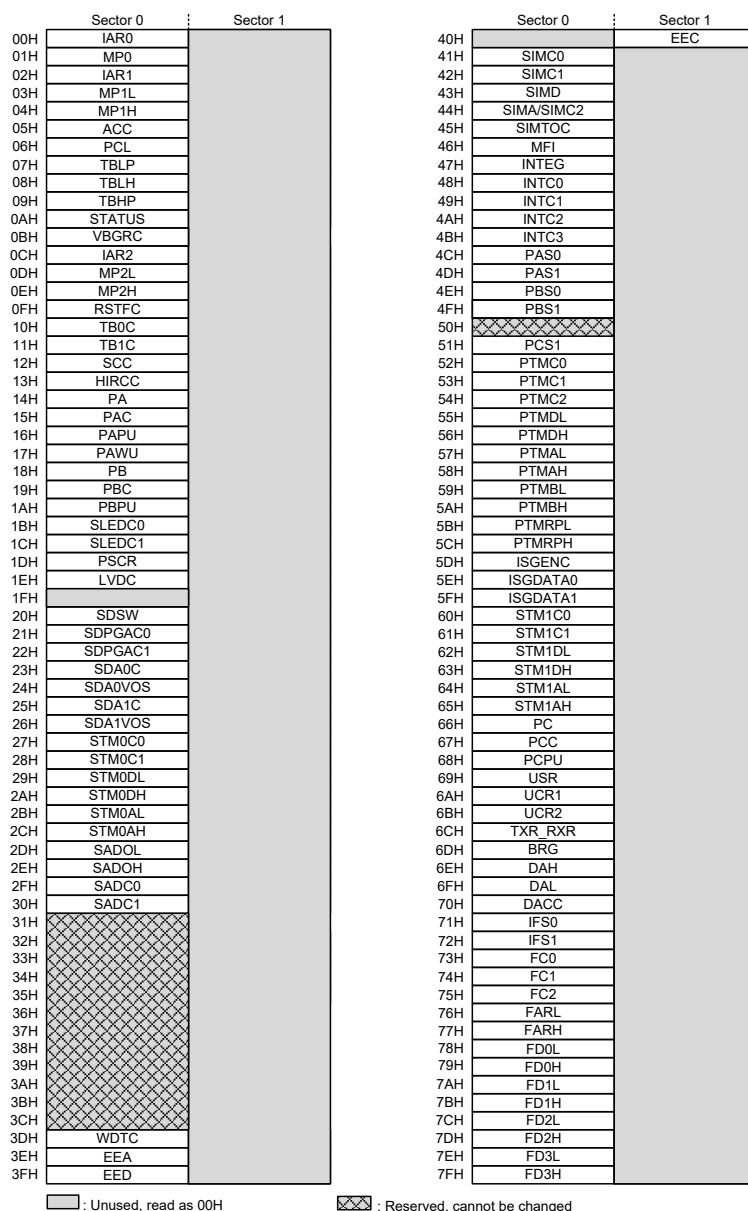
此单片机支持扩展指令架构，它并没有可用于数据存储器的存储区指针。对于数据存储器所需的 Sector 是通过 MP1H 或 MP2H 寄存器指定，而所选 Sector 的某一数据存储器地址是使用间接寻址访问方式通过 MP1L 或 MP2L 寄存器指定。直接寻址可用于所有 Sector，通过扩展指令可以寻址所有可用的数据存储器空间。所访问的数据存储器位于除 Sector 0 外的任何数据存储器 Sector，扩展指令可代替间接寻址方式用来访问数据存储器。对于该单片机而言，标准指令和扩展指令的主要区别在于扩展指令中的数据存储器地址“m”有 11 个有效位，高字节表示某一 Sector，低字节表示某一指定地址。

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。这个数据存储区可让使用者进行读取和写入的操作。使用位操作指令可对个别的位做置位或复位的操作，较大地方便了用户在数据存储器内进行位操作。

特殊功能数据存储器

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



特殊功能数据存储器结构

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。间接寻址的方法准许使用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 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区中能像普通的寄存器一般被操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由存储器指针所指定的地址。MP0、IAR0 用于访问 Sector 0，而 MP1L/MP1H 和 IAR1、MP2L/MP2H 和 IAR2 可根据 MP1H 或 MP2H 寄存器访问所有的 Sector。使用扩展指令可对所有的数据 Sector 进行直接寻址。

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

间接寻址程序范例 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:
```

间接寻址程序范例 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:
```

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。

使用扩展指令直接寻址程序范例

```
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 位长度，因此只允许在本页的程序存储器范围内进行跳转，而当使用这种运算时，要注意会插入一个空指令周期。

表格寄存器 – TBLP, TBHP, TBLH

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

状态寄存器 – STATUS

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

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

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

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 标志位也受循环移位指令的影响。

EEPROM 数据存储

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

EEPROM 数据存储结构

该单片机的 EEPROM 数据存储容量为 128×8 位。由于映射方式与程序存储器和数据存储器不同，因此不能像其它类型的存储器一样寻址。使用 Sector 0 中的一个地址寄存器和一个数据寄存器以及 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，只能通过 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：1. 在同一条指令中 WREN、WR、RDEN 和 RD 不能同时置为“1”。

2. 确保 f_{SUB} 时钟在执行写动作前已稳定。

3. 确保写动作完成后才可改写 EEPROM 相关寄存器。

从 EEPROM 中读取数据

从 EEPROM 中读取数据，EEPROM 中读取数据的地址要先放入 EEA 寄存器中。EEC 寄存器中的读使能位 RDEN 先置为高以使能读功能。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，这两条指令必须在两个指令周期内连续执行。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。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 及 MP2H 将重置为“0”，这意味着数据存储器 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 使能且堆栈未满的情况下将跳转到 EEPROM 中断向量中执行。当中断被响应，EEPROM 中断标志位将自动复位，更多内容请参考“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 MP1H 或 MP2H 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。

程序举例

从 EEPROM 中读取数据 — 轮询法

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, 040H               ; setup memory pointer 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 read if no more read operations
                           ; are required

CLR MP1H
MOV A, EED                 ; move read data to register
MOV READ_DATA, A
```

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

写数据到 EEPROM — 轮询法

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA        ; user defined data
MOV EED, A
MOV A, 040H               ; setup memory pointer 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 - executed
                      ; immediately after set WREN bit

SET EMI
BACK:
SZ IAR1.2            ; check for write cycle end
JMP BACK
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较佳的优化。振荡器选择是通过应用程序控制寄存器完成的。

振荡器概述

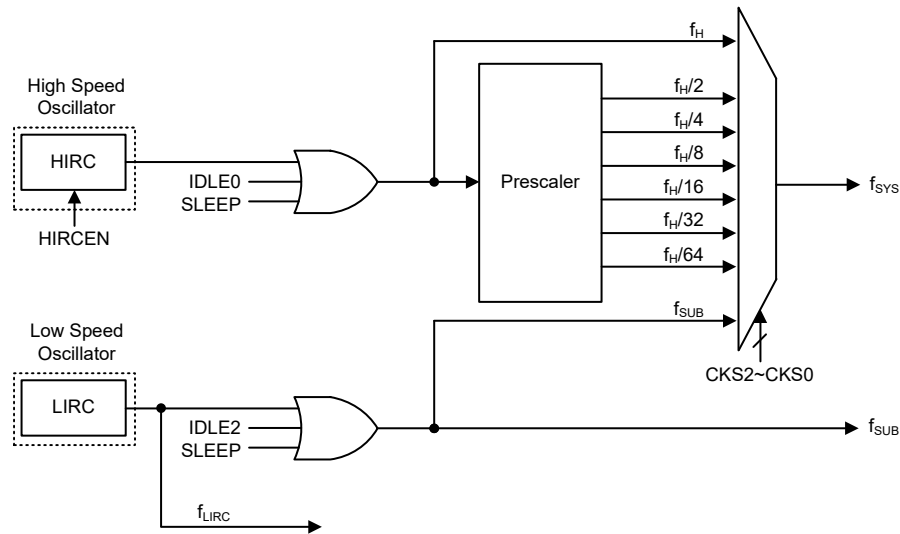
振荡器除了作为系统时钟源，还作为看门狗定时器和时基功能的时钟源。两个完全集成的内部振荡器不需要任何外围器件。它们提供的高速和低速系统振荡器具有较宽的频率范围。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

类型	名称	频率
内部高速 RC	HIRC	2/4/8MHz
内部低速 RC	LIRC	32kHz

振荡器类型

系统时钟配置

此单片机有两个系统振荡器，包括一个高速振荡器和一个低速振荡器。高速振荡器有内部 2/4/8MHz RC 振荡器 – HIRC，低速振荡器有内部 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。



系统时钟配置

内部高速 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，不需其它外部器件。内部 RC 振荡器具有三种固定的频率：2MHz，4MHz，8MHz。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因 V_{DD} 、温度以及芯片制成工艺不同的影响较大程度地降低。

内部 32kHz 振荡器 – LIRC

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

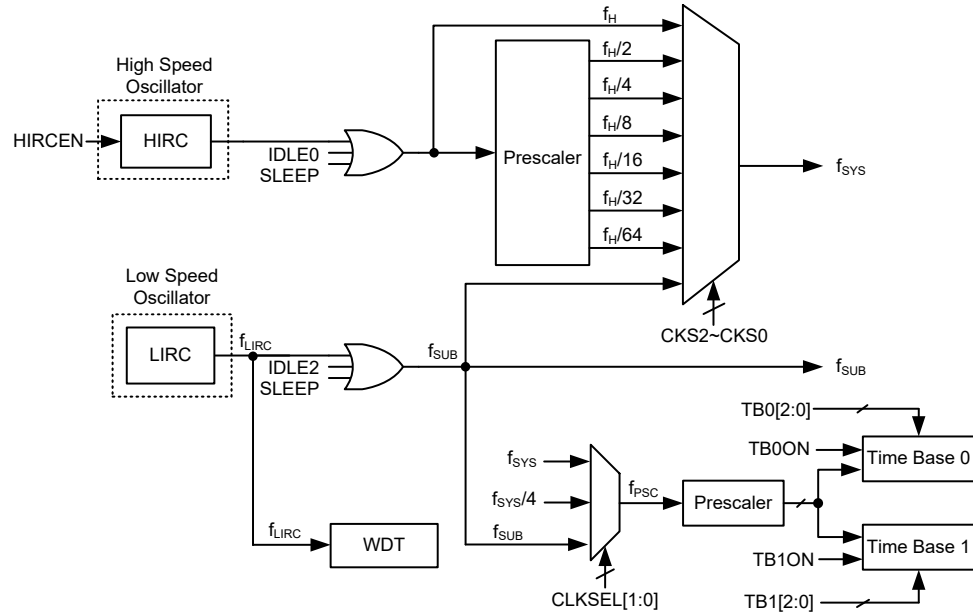
工作模式和系统时钟

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

系统时钟

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

主系统时钟可来自高频时钟源 f_H 或低频时钟源 f_{SUB} ，通过 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，低频时钟来自 LIRC 振荡器。其它系统时钟还有高速系统振荡器的分频 $f_H/2 \sim f_H/64$ 。



单片机时钟配置

注：当系统时钟源 f_{SYS} 由 f_H 切换为 f_{SUB} 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 $f_H \sim f_H/64$ 频率的时钟源。

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	$f_H \sim f_H/64$	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. 在休眠模式中， f_{LIRC} 时钟会开启因为 WDT 功能始终使能。

快速模式

这是主要的工作模式之一，单片机的所有功能均可在此模式中实现且系统时钟由一个高速振荡器提供。该模式下单片机正常工作的时钟源来自 HIRC 振荡器。高速振荡器频率可被分为 1~64 的不等比率，实际的比率由 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。该低速时钟源可来自 f_{SUB} ，而 f_{SUB} 来自 LIRC 振荡器。

休眠模式

在 HALT 指令执行后且 FHIDEN 位和 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使能， f_{LIRC} 继续运行。

空闲模式 0

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 位为低、FSIDEN 位为高时，系统进入空闲模式 0。在空闲模式 0 中，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 寄存器中的 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 寄存器中的 FHIDEN 位为高、FSIDEN 位为低时，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 和 HIRCC 用于控制系统时钟和 HIRC 振荡器配置。

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

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	—	—	—	FHIDEN	FSIDEN
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	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 4~2 未定义，读为“0”
- 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。
LIRC 振荡器是由该位与 WDT 功能使能控制位共同控制的。如果该位被清零，
但 WDT 功能使能，f_{LIRC} 振荡器也将使能。

● HIRCC 寄存器

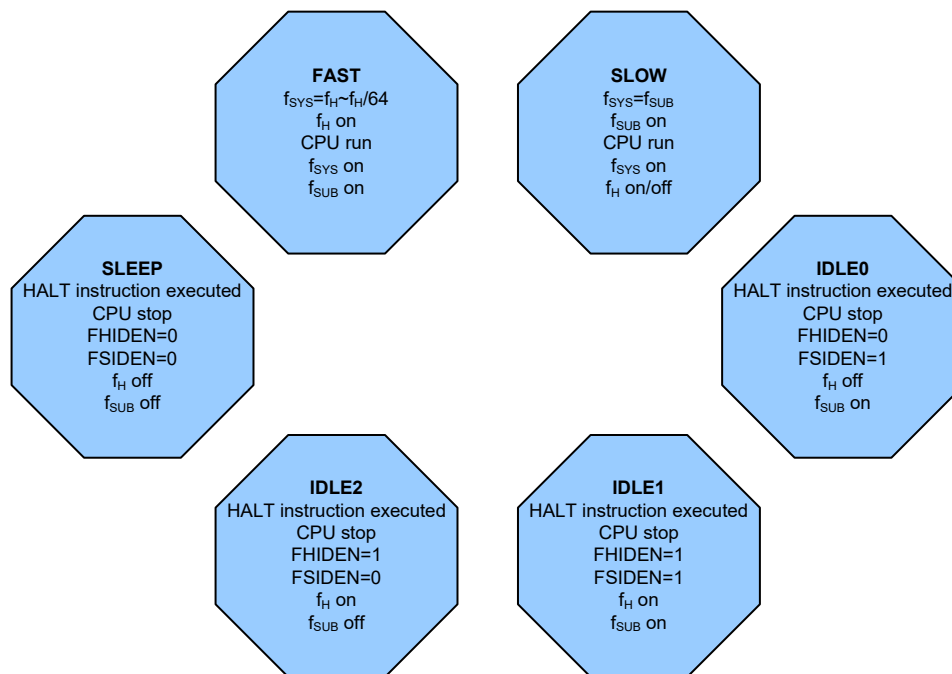
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	1

- Bit 7~4 未定义，读为“0”
- Bit 3~2 **HIRC1~HIRC0**: HIRC 频率选择位
00: 2MHz
01: 4MHz
10: 8MHz
11: 2MHz
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。
- 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 振荡器或通过应用程序改变 HIRC 振荡器频率选择位，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

工作模式切换

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

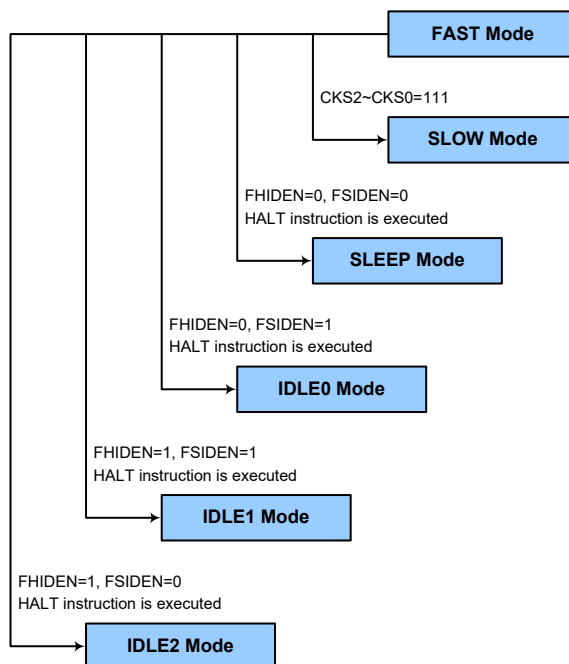
简单来说，快速模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



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。

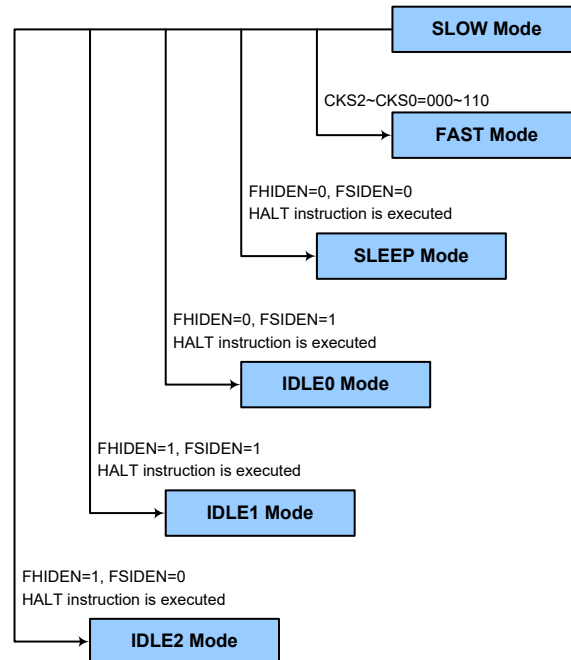
低速模式的时钟源来自 LIRC 振荡器，因此要求此振荡器在所有模式切换动作发生前稳定下来。



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 $CKS2 \sim CKS0$ 位为“000”~“110”使系统时钟从 f_{SUB} 切换到 $f_H \sim f_H/64$ 。

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系统上电时间电气特性中有说明。



进入休眠模式

进入休眠模式的方法仅有一种，即应用程序中执行“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 功能总是使能，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，即应用程序中执行“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 功能总是使能，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，即应用程序中执行“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 功能总是使能，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，即应用程序中执行“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 功能总是使能，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单片机的电流降低到尽可能低，空闲模式 0 和休眠模式可能到只有几个微安的级别，所以如果要将电路的电流进一步降低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入 / 输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。

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

在空闲模式 1 和空闲模式 2 中高速振荡器开启。若外围功能时钟源来自高速振荡器，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

单片机执行 HALT 指令，系统将进入空闲或休眠模式，PDF 将被置位；系统上

电或执行清除看门狗的指令，PDF 将被清零。

若系统由看门狗定时器溢出唤醒，则会发生看门狗定时器复位，TO 将被置位。看门狗计数器溢出将会置位 TO 标志并唤醒系统，这种复位会重置程序计数器和堆栈指针，其它标志保持原有状态。

PA 口中的每个引脚都可以通过 PAWU 寄存器使能下降沿唤醒功能。PA 端口唤醒后，程序将在“HALT”指令后继续执行。

如果系统是通过中断唤醒，则有两种可能发生。第一种情况是：相关中断除能或是中断使能且堆栈已满，则程序会在“HALT”指令之后继续执行。这种情况下，唤醒系统的中断会等到相关中断使能或有堆栈层可以使用之后才执行。第二种情况是：相关中断使能且堆栈未满，则中断可以马上执行。如果在进入休眠或空闲模式之前中断标志位已经被设置为“1”，则相关中断的唤醒功能将无效。

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 定时器时钟源由内部低速振荡器 LIRC 提供。内部振荡器 LIRC 的频率大约为 32kHz，这个特殊的内部时钟周期会随 V_{DD} 、温度和制成的不同而变化。看门狗定时器的时钟源可分频为 $2^8 \sim 2^{18}$ 以提供更大的溢出周期，分频比由 WDT 寄存器中的 WS2~WS0 位来决定。

看门狗定时器控制寄存器

WDT 寄存器用于选择溢出周期、控制 WDT 功能的使能和 MCU 复位操作。

• WDT 寄存器

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

Bit 7~3 **WE4~WE0**: WDT 功能软件控制位

10101 或 01010: 使能

其它: MCU 复位

若因外部环境噪声或软件设置使单片机复位，复位动作发生在一段延迟时间 t_{SRESET} 后，复位后 RSTFC 寄存器中的 WRF 标志位会被置位。

Bit 2~0 **WS2~WS0**: WDT 溢出周期选择位

000: $2^8/f_{LIRC}$

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	—	—	—	—	—	LVRF	—	WRF
R/W	—	—	—	—	—	R/W	—	R/W
POR	—	—	—	—	—	x	—	0

“x”：未知

Bit 7~3 未定义，读为“0”

Bit 2 **LVRF**: LVR 复位标志位
详见“低电压复位”章节

Bit 1 未定义，读为“0”

Bit 0 **WRF**: WDT 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，它产生一个单片机复位的动作。这也就意味着正常工作期间，用户需在应用程序中看门狗溢出前有策略地清看门狗定时器以防止其产生复位，可使用清除看门狗指令实现。无论什么原因，程序失常跳转到一个未知的地址或进入一个死循环，这个清除指令不能被正确执行，此种情况下，看门狗将溢出以使单片机复位。看门狗定时器控制寄存器 WDTC 中的 WE4~WE0 位可提供使能控制以及控制看门狗定时器复位单片机。如果 WE4~WE0 为 01010B 或 10101B 则 WDT 使能；而当设置为“01010B”和“10101B”以外的值时，单片机将在一段延迟时间 t_{SRESET} 后复位。上电后这些位初始化为“01010B”。

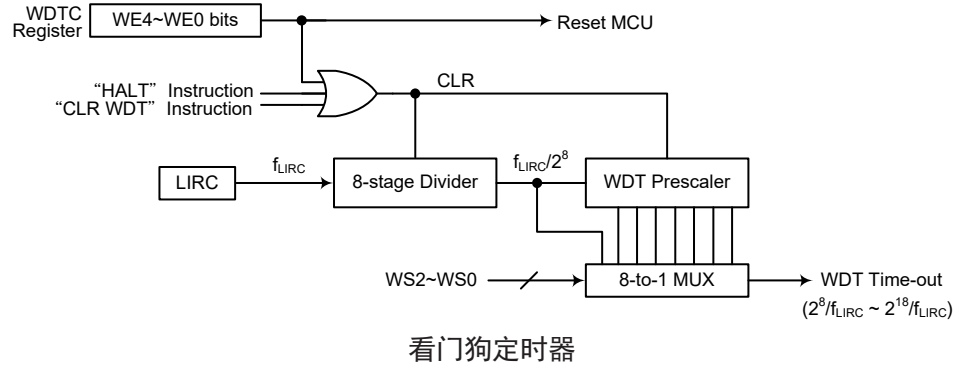
WE4~WE0 位	WDT 功能
01010B 或 10101B	使能
其它值	单片机复位

看门狗定时器使能 / 复位控制

程序正常运行时，WDT 溢出将导致单片机复位，并置位状态标志位 TO。若系统处于休眠或空闲模式，当 WDT 发生溢出时，状态寄存器中的 TO 应置位，仅 PC 和堆栈指针复位。有三种方法可以用来清除 WDT 的内容。第一种是 WDTC 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 WDT 软件清除指令，而第三种是通过“HALT”指令。

该单片机只使用一条软件指令清看门狗。只要执行“CLR WDT”便清除 WDT。

当设置分频比为 2^{18} 时，溢出周期最大。例如，时钟源为 32kHz LIRC 振荡器，分频比为 2^{18} 时最大溢出周期约 8s，分频比为 2^8 时最小溢出周期约 8ms。



复位和初始化

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

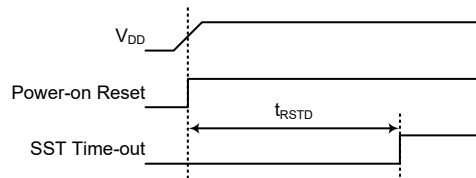
另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。另一种复位为看门狗溢出单片机复位，不同方式的复位操作会对寄存器产生不同的影响。

复位功能

单片机包含下面几种由内部事件触发的复位方式。

上电复位

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



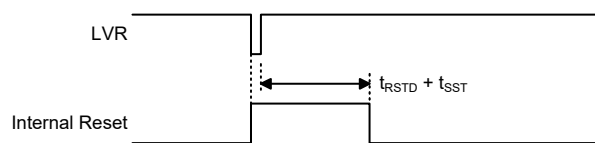
上电复位时序图

低电压复位 – LVR

单片机具有低电压复位电路，用来监测它的电源电压。当电源电压低于某一预定值时，它将复位单片机。

正常运行时 LVR 始终使能，并会设定一个电源复位低电压 V_{LVR} 。该单片机的 V_{LVR} 值固定为 2.1V。如果在更换电池的情况下，单片机供应的电压可能会在 $0.9V \sim V_{LVR}$ 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。有效的 LVR 信号，即在 $0.9V \sim V_{LVR}$ 的低电压状态的时间，必须超过 LVD & 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。

需要注意的是，当单片机进入空闲或休眠模式，LVR 功能将自动除能。



低电压复位时序图

• RSTFC 寄存器

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

“x”：未知

Bit 7~3 未定义，读为“0”

Bit 2 **LVRF**: LVR 功能复位标志位

0: 未发生

1: 发生

当特定的低电压复位条件发生时，此位被置为“1”，且只能通过应用程序清零。

Bit 1 未定义，读为“0”

Bit 0 **WRF**: WDT 控制寄存器软件复位标志位

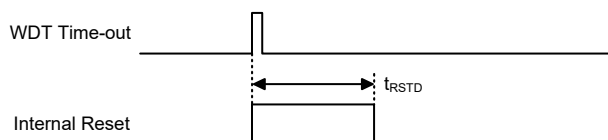
具体描述见看门狗定时器控制寄存器章节。

IAP 复位

当写值“55H”至 FC1 寄存器时，将产生一个复位信号将整个单片机复位。详见在线应用编程章节。

正常运行时看门狗溢出复位

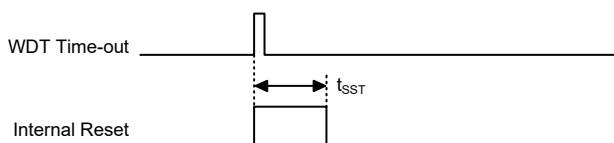
在正常运行即快速模式和低速模式下看门狗溢出复位后 TO 将被设为“1”。



正常运行时看门狗溢出复位时序图

休眠或空闲时看门狗溢出复位

休眠或空闲时看门狗溢出复位和其它种类的复位有些不同，除了程序计数器与堆栈指针将被清零及 TO 位被设为 1 外，绝大部份的条件保持不变。图中 t_{SST} 的详细说明请参考系统上电时间电气特性。



休眠或空闲时看门狗溢出复位时序图

复位初始状态

不同的复位形式以不同的途径影响复位标志位。这些标志位，即存放在状态寄存器中的 PDF 和 TO 位，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”代表不改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

项目	复位后情况
程序计数器	清除为零
中断	所有中断被除能
看门狗定时器，时基	都清除，且 WDT 重新计数
定时器模块	所有定时器模块停止
输入 / 输出口	I/O 口设为输入模式
堆栈指针	堆栈指针指向堆栈顶端

不同的复位形式对单片机内部寄存器的影响是不同的。为保证复位后程序能正常执行，了解寄存器在特定条件复位后的设置是非常重要的。下表即为不同方式复位后内部寄存器的状况。注意的是，此单片机支持多种封装类型，该表将反映较大封装类型的情况。

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
IAR0	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	---x xxxx	---u uuuu	---u uuuu
STATUS	xx00 xxxx	xx1u uuuu	uull uuuu
VBGRC	---- --0	---- --0	---- --u
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- -x-0	---- -u-u	---- -u-u
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
SCC	000- --00	000- --00	uuu- --uu
HIRCC	---- 0001	---- 0001	---- uuuu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
SLEDC0	0000 0000	0000 0000	uuuu uuuu
SLEDC1	---- 0000	---- 0000	---- uuuu
PSCR	---- --00	---- --00	---- --uu
LVDC	--00 0000	--00 0000	--uu uuuu
SDSW	-000 0000	-000 0000	-uuu uuuu
SDPGAC0	--00 0000	--00 0000	--uu uuuu
SDPGAC1	0000 0000	0000 0000	uuuu uuuu
SDA0C	-00- --00	-00- --00	-uu- --uu
SDA0VOS	0010 0000	0010 0000	uuuu uuuu
SDA1C	-00- --00	-00- --00	-uu- --uu
SDA1VOS	0010 0000	0010 0000	uuuu uuuu
STM0C0	0000 0000	0000 0000	uuuu uuuu
STM0C1	0000 0000	0000 0000	uuuu uuuu
STM0DL	0000 0000	0000 0000	uuuu uuuu
STM0DH	---- --00	---- --00	---- --uu
STM0AL	0000 0000	0000 0000	uuuu uuuu
STM0AH	---- --00	---- --00	---- --uu
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFs=0)
			uuuu uuuu (ADRFs=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFs=0)
			---- uuuu (ADRFs=1)
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 0000	0000 0000	uuuu uuuu
WDTC	0101 0011	0101 0011	uuuu uuuu
EEA	-000 0000	-000 0000	-uuu uuuu
EED	0000 0000	0000 0000	uuuu uuuu
EEC	---- 0000	---- 0000	---- uuuu
SIMC0	111- 0000	111- 0000	uuu- uuuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
SIMC1	1000 0001	1000 0001	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA	0000 0000	0000 0000	uuuu uuuu
SIMC2	0000 0000	0000 0000	uuuu uuuu
SIMTOC	0000 0000	0000 0000	uuuu uuuu
MFI	-000 -000	-000 -000	-uuu -uuu
INTEG	---- 0000	---- 0000	---- uuuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	0000 0000	0000 0000	uuuu uuuu
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	0000 0000	0000 0000	uuuu uuuu
PBS1	0000 0000	0000 0000	uuuu uuuu
PCS1	---- 0000	---- 0000	---- uuuu
PTMC0	0000 0---	0000 0---	uuuu u---
PTMC1	0000 0000	0000 0000	uuuu uuuu
PTMC2	---- -000	---- -000	---- -uuu
PTMDL	0000 0000	0000 0000	uuuu uuuu
PTMDH	---- --00	---- --00	---- --uu
PTMAL	0000 0000	0000 0000	uuuu uuuu
PTMAH	---- --00	---- --00	---- --uu
PTMBL	0000 0000	0000 0000	uuuu uuuu
PTMBH	---- --00	---- --00	---- --uu
PTMRPL	0000 0000	0000 0000	uuuu uuuu
PTMRPH	---- --00	---- --00	---- --uu
ISGENC	0--- --00	0--- --00	u--- --uu
ISGDATA0	---0 0000	---0 0000	---u uuuu
ISGDATA1	---0 0000	---0 0000	---u uuuu
STM1C0	0000 0000	0000 0000	uuuu uuuu
STM1C1	0000 0000	0000 0000	uuuu uuuu
STM1DL	0000 0000	0000 0000	uuuu uuuu
STM1DH	---- --00	---- --00	---- --uu
STM1AL	0000 0000	0000 0000	uuuu uuuu
STM1AH	---- --00	---- --00	---- --uu
PC	--11 1111	--11 1111	--uu uuuu
PCC	--11 1111	--11 1111	--uu uuuu
PCPU	--00 0000	--00 0000	--uu uuuu
USR	0000 1011	0000 1011	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
UCR1	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	uuuu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	uuuu uuuu
DAH	0000 0000	0000 0000	uuuu uuuu
DAL	0000 0000	0000 0000	uuuu uuuu
DACC	---- --0	---- --0	---- --u
IFS0	0000 0000	0000 0000	uuuu uuuu
IFS1	---- 0000	---- 0000	---- uuuu
FC0	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	uuuu uuuu
FC2	---- --0	---- --0	---- --u
FARL	0000 0000	0000 0000	uuuu uuuu
FARH	---0 0000	---0 0000	---u uuuu
FD0L	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	uuuu uuuu

注：“u”表示不改变
“x”表示未知
“-”表示未定义

输入 / 输出端口

Holtek 单片机的输入 / 输出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。

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

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

“—”：未定义，读为“0”

输入 / 输出逻辑功能寄存器列表

注：未引出的功能线，例如 20-pin SOP 封装中的 PB0~PB2、PB4~PB7、PC0~PC3、PC5，24-pin SOP 封装中的 PB5~PB7、PC0~PC3、PC5 和 28-pin SOP 封装中的 PC0~PC3、PC5 也受寄存器中对应位的控制，需合理设置其状态以避免输入浮空造成额外耗电，详见“待机电流注意事项”章节。

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应的上拉控制寄存器 PAPU~PCPU 来设置，它用一个弱 PMOS 晶体管来实现上拉电阻功能。

需要注意的是当 I/O 引脚设为数字输入或 NMOS 输出时，上拉功能才会受 PxPU 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 Port x 引脚上拉功能控制位

0: 除能
1: 使能

PxPUn 位用于控制引脚上拉功能。此处的“x”可为 A、B 或 C。每个端口实际的有效位是不同的，具体信息可参考 I/O 逻辑功能寄存器列表。

PA 口唤醒

当使用暂停指令“HALT”迫使单片机进入休眠或空闲模式，单片机的系统时钟将会停止以降低功耗，此功能对于电池及低功耗应用很重要。唤醒单片机有很多种方法，其中之一就是使 PA 口的其中一个引脚从高电平转为低电平。这个功能特别适合于通过外部开关来唤醒的应用。PA 口的每个引脚可以通过设置 PAWU 寄存器来单独选择是否具有唤醒功能。

需要注意的是只有当引脚被设置为通用 I/O 功能输入类型且单片机处于空闲或休眠模式时，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: PA7~PA0 唤醒功能控制位

0: 除能

1: 使能

输入 / 输出端口控制寄存器

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

• PxC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PxC7	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 Port x 引脚输入 / 输出类型选择位

0: 输出

1: 输入

PxCn 位用于选择引脚输入 / 输出类型。此处的“x”可为 A、B 或 C。每个端口实际的有效位是不同的，具体信息可参考 I/O 逻辑功能寄存器列表。

源电流选择

该单片机的每个引脚都支持不同的源电流驱动能力，通过相应的源电流选择位控制。仅当对应的引脚被设为 CMOS 输出时，其源电流选择位才有效。否则，这些选择位无效。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。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SLEDC0	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
SLEDC1	—	—	—	—	SLEDC13	SLEDC12	SLEDC11	SLEDC10

源电流选择寄存器列表

● 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 源电流选择位
 00: 源电流 = Level 0 (最小)
 01: 源电流 = Level 1
 10: 源电流 = Level 2
 11: 源电流 = Level 3 (最大)

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

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

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

● SLEDC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SLEDC13	SLEDC12	SLEDC11	SLEDC10
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 **SLEDC13~SLEDC12**: PC4 源电流选择位
 00: 源电流 = Level 0 (最小)
 01: 源电流 = Level 1
 10: 源电流 = Level 2
 11: 源电流 = Level 3 (最大)

Bit 1~0 **SLEDC11~SLEDC10**: 该字段必须固定为“00”且不可更改

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。该单片机包含端口引脚输出功能选择寄存器 $PxSn$ ，和输入功能输入引脚选择寄存器 $IFSn$ ，这些寄存器可以用来对引脚上的功能进行配置。

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。对于大部分共用功能，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 $INTn$ 、 $xTCKn$ 、 $xTPnI$ 等，与对应的通用 I/O 口共用同一个引脚共用设置选项。要选择这个引脚功能，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
PCS1	—	—	—	—	PCS13	PCS12	PCS11	PCS10
IFS0	IFS07	IFS06	IFS05	IFS04	IFS03	IFS02	IFS01	IFS00
IFS1	—	—	—	—	IFS13	IFS12	IFS11	IFS10

引脚共用功能选择寄存器列表

● PAS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	PAS05	PAS04	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/INT0/STP1I

01: SDO

10: TX

11: AN3

注：由于 MODE 功能，即压电蜂鸣片驱动器其中一个输入信号，与此引脚一同引出且该功能始终使能，该引脚上的任何变化都可能引起 MODE 输入的改变。因此在使用此引脚和设定相关寄存器时应格外注意。

Bit 5~4 **PAS05~PAS04:** PA2 引脚共用功能选择

00: PA2

01: SDI/SDA

10: RX

11: PA2

- Bit 3~2 **PAS03~PAS02:** PA1 引脚共用功能选择
 00: PA1/INT1
 01: $\overline{\text{SCS}}$
 10: A1O
 11: A1PI
- Bit 1~0 **PAS01~PAS00:** PA0 引脚共用功能选择
 00: PA0
 01: SCL/SCK
 10: PA0
 11: PA0

● **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: PA7/STP0I/PTPI
 01: SCK/SCL
 10: AN1
 11: VREF

- Bit 5~4 **PAS15~PAS14:** PA6 引脚共用功能选择
 00: PA6
 01: PTP
 10: SDI/SDA
 11: RX

注：由于 ENCLK 功能，即压电蜂鸣片驱动器其中一个输入信号，与此引脚一同引出且该功能始终使能，该引脚上的任何变化都可能引起 ENCLK 输入的改变。因此，在使用此引脚和设定相关寄存器时应格外注意。建议在选中 2PIN Buzzer 功能的时候该字段值应设置为“01”以选择 PTP 功能进行 ENCLK 状态控制，在选中 3PIN Buzzer 功能的时候该字段值应设置为“00”选择 PA6 功能。

- Bit 3~2 **PAS13~PAS12:** PA5 引脚共用功能选择
 00: PA5/STCK0
 01: STP1B
 10: A1O
 11: PA5/STCK0

- Bit 1~0 **PAS11~PAS10:** PA4 引脚共用功能选择
 00: PA4/PTCK
 01: STP0B
 10: AN0
 11: A0O

● **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: PB3
 01: 保留，不可用
 10: SDI/SDA
 11: RX

- Bit 5~4 **PBS05~PBS04:** PB2 引脚共用功能选择
 00: PB2
 01: 保留, 不可用
 10: SCK/SCL
 11: DACO
- Bit 3~2 **PBS03~PBS02:** PB1 引脚共用功能选择
 00: PB1
 01: 保留, 不可用
 10: SDO
 11: TX
- Bit 1~0 **PBS01~PBS00:** PB0 引脚共用功能选择
 00: PB0/INT0
 01: STP0
 10: A0PB
 11: DACO

● **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: PB7
 01: $\overline{SCS}$
 10: AN6
 11: PB7
- Bit 5~4 **PBS15~PBS14:** PB6 引脚共用功能选择
 00: PB6
 01: AN5
 10: DACO
 11: PB6
- Bit 3~2 **PBS13~PBS12:** PB5 引脚共用功能选择
 00: PB5
 01: TX
 10: STP0B
 11: AN4
- Bit 1~0 **PBS11~PBS10:** PB4 引脚共用功能选择
 00: PB4
 01: $\overline{SCS}$
 10: AN2
 11: PB4

● **PCS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PCS13	PCS12	PCS11	PCS10
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 **PCS13~PCS12:** 该字段值必须固定为 “00” 且不可更改

Bit 1~0 **PCS11~PCS10:** PC4 引脚共用功能选择
 00: PC4/STCK1
 01: PTPB
 10: PC4/STCK1
 11: PC4/STCK1

● **IFS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	IFS07	IFS06	IFS05	IFS04	IFS03	IFS02	IFS01	IFS00
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 **IFS07~IFS06:** PTPI 输入源选择
 01: PA7
 其它值: 保留, 不可用
 注: 若选择了 PTPI 功能, 该字段需设置为 “01”。

Bit 5~4 **IFS05~IFS04:** SCS 输入源引脚选择
 00: PB4
 01: PB7
 10: PA1
 11: PB4

Bit 3~2 **IFS03~IFS02:** SCK/SCL 输入源引脚选择
 00: PB2
 01: PA0
 10: PA7
 11: 保留, 不可用
 注: 如果选择 SPI 主机模式, 当 SIMEN 位设置为高电平时, PA0、PA7、PB2 引脚都可以用作 SCK 引脚功能, 忽略 IFS0[3:2] 位的设置。

Bit 1~0 **IFS01~IFS00:** SDI/SDA 输入源引脚选择
 00: PB3
 01: PA2
 10: PA6
 11: 保留, 不可用

● **IFS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IFS13	IFS12	IFS11	IFS10
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

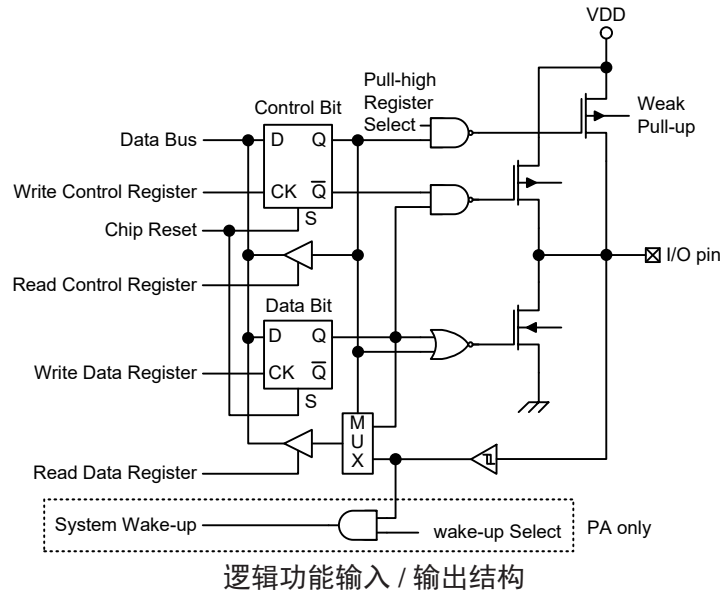
Bit 7~4 未定义, 读为 “0”

Bit 3~2 **IFS13~IFS12:** INT0 输入源引脚选择
 00: PB0
 01: PA3
 10: PB0
 11: PB0

Bit 1~0 **IFS11~IFS10:** RX 输入源引脚选择
 00: PB3
 01: PA2
 10: PA6
 11: 保留, 不可用

输入 / 输出引脚结构

下图为输入 / 输出引脚逻辑功能的内部结构图。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 引脚逻辑功能的理解提供一个参考。由于存在诸多的引脚共用结构，在此不方便提供所有类型引脚功能结构图。



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将某些引脚设置为输出状态，这些输出引脚会有初始高电平输出，除非端口数据寄存器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标准型和周期型定时器章节。

简介

该单片机包含 3 个 TM。每个 TM 可被划分为一个特定的类型，即标准型 TM 和周期型 TM。虽然性质相似，但不同 TM 特性复杂度不同。本章介绍标准型和周期型 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。

功能	STM	PTM
定时 / 计数器	√	√
捕捉输入	√	√
比较匹配输出	√	√
PWM 输出	√	√
单脉冲输出	√	√
PWM 对齐方式	边沿对齐	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

TM 操作

两种不同类型的 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时钟来驱动内部 TM 计数器。

TM 时钟源

驱动 TM 计数器的时钟源很多。通过设置 xTMn 控制寄存器的 xTnCK2~xTnCK0 位，选择所需的时钟源，其中 x 代表 S 或 P 类型，n 代表同一类型 TM 中每个 TM 的编号。由于该单片机只包含一个 PTM，因此这类型的 TM 相关的引脚和控制位不带编号。该时钟源来自系统时钟 f_{sys} 或内部高速时钟 f_H 的分频比或 f_{sub} 时钟源或外部 xTCKn 引脚。xTCKn 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

标准型和周期型 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，都有两个 TM 输入引脚 xTCKn 和 xTPnI。xTMn 输入引脚 xTCKn 作为 xTMn 时钟源输入脚，通过设置 xTMnC0 寄存器中的

xTnCK2~xTnCK0 位进行选择。外部时钟源可通过该引脚来驱动内部 TM。
xTCKn 引脚可选择上升沿有效或下降沿有效。xTCKn 引脚还可分别用作 xTMn
单脉冲模式的外部触发引脚。

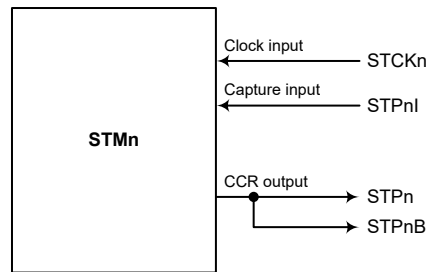
另一种 TM 输入引脚 xTPnI 作为捕捉输入脚，其有效边沿有上升沿、下降沿和
双沿，通过设置 xTMnC1 寄存器中的 xTnIO1~xTnIO0 位来选择有效边沿类型。
除了 PTPI 引脚外，PTCK 引脚也可用作 PTM 捕捉输入模式的外部触发引脚。

每个 TM 都有一个或两个输出引脚 xTPn 和 xTPnB。xTPnB 信号为 xTPn 输出的
反相信号。当 TM 工作在比较匹配输出模式且比较匹配发生时，这些引脚会由
TM 控制切换到高电平或低电平或翻转。外部输出引脚也被 TM 用来产生 PWM
输出波形。

因 TM 输入和输出引脚与其它功能共用，TM 输入和输出功能需要事先通过相关
引脚共用功能选择位进行设置。更多引脚共用功能选择详见引脚共用功能章节。

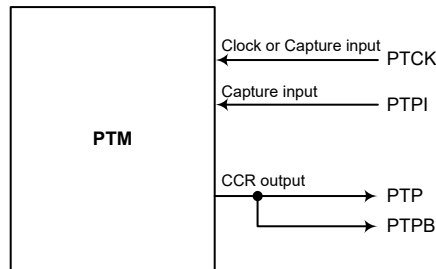
STMn		PTM	
输入	输出	输入	输出
STCKn, STPnI	STPn, STPnB	PTCK, PTPI	PTP, PTPB

TM 外部引脚



注：STP1 功能不可用。

STMn 功能引脚方框图 (n=0~1)



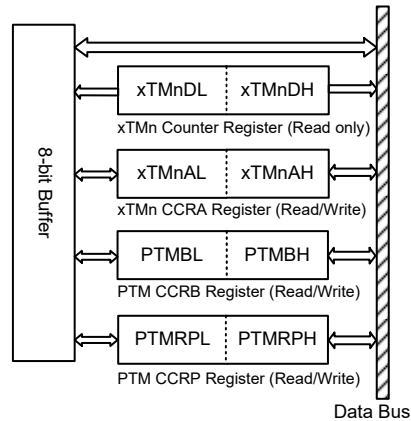
PTM 功能引脚方框图

编程注意事项

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

CCRA、CCRP 和 CCRB 寄存器访问方式如下图所示，读写这些成对的寄存器
需通过上述的特殊方式。建议使用“MOV”指令按照以下步骤访问 CCRA、

CCRP 和 CCRB 低字节寄存器，即 xTMnAL、PTMRPL 和 PTMBL，否则可能导致无法预期的结果。

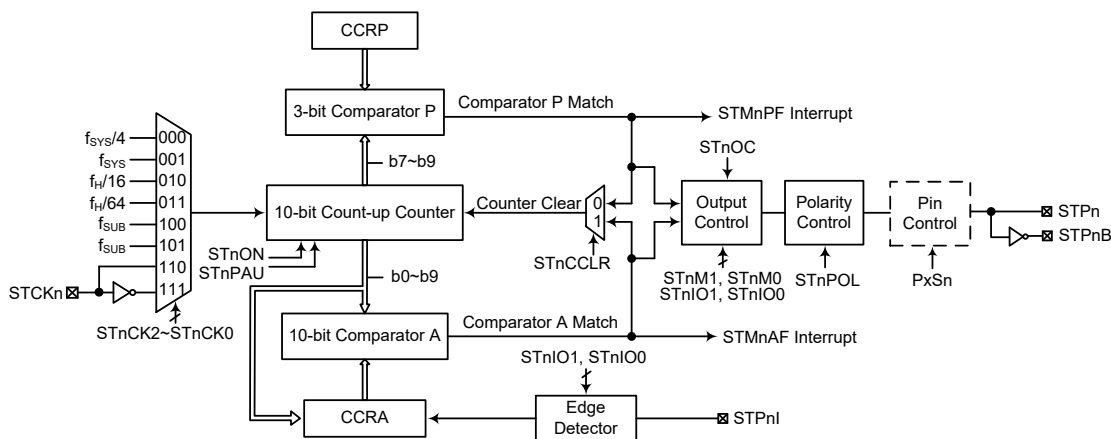


读写流程如下步骤所示：

- 写数据至 CCRA、CCRB 或 CCRP
 - ◆ 步骤 1. 写数据至低字节寄存器 xTMnAL、PTMBL 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 xTMnAH、PTMBH 或 PTMRPH
 - 注意，此时数据直接写入高字节寄存器，同时锁存在 8-bit 缓存器中的数据写入低字节寄存器。
- 由计数器寄存器、CCRA、CCRB 或 CCRP 中读取数据
 - ◆ 步骤 1. 由高字节寄存器 xTMnDH、xTMnAH、PTMBH 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 xTMnDL、xTMnAL、PTMBL 或 PTMRPL 读取数据
 - 注意，此时读取 8-bit 缓存器中的数据。

标准型 TM – STM

标准型 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。标准型 TM 由两个外部输入脚控制并驱动两个外部输出脚。



- 注：1. STPnB 信号为 STPn 输出的反相信号。需注意 STP1 功能不可用。
2. STMn 外部引脚与其它功能共用，所以在使用 STMn 功能前引脚共用功能寄存器必须正确设置。若使用 STCK 和 STPI 引脚，还需通过设定端口控制寄存器中的对应位将对应引脚设置为输入。

标准型 TM 方框图 (n=0~1)

标准型 TM 操作

标准型 STMn 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 位宽度，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 STnON 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 STMn 中断信号。标准型 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寄存器来实现的。

标准型 TM 寄存器介绍

标准型 TM 的所有工作模式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，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 3 位 CCRP 的值。

寄存器名称	位							
	7	6	5	4	3	2	1	0
STMnC0	STnPAU	STnCK2	STnCK1	STnCK0	STnON	STnRP2	STnRP1	STnRP0
STMnC1	STnM1	STnM0	STnIO1	STnIO0	STnOC	STnPOL	STnDPX	STnCCLR
STMnDL	D7	D6	D5	D4	D3	D2	D1	D0
STMnDH	—	—	—	—	—	—	D9	D8
STMnAL	D7	D6	D5	D4	D3	D2	D1	D0
STMnAH	—	—	—	—	—	—	D9	D8

10-bit 标准型 TM 寄存器列表 (n=0~1)

● STMnC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	STnPAU	STnCK2	STnCK1	STnCK0	STnON	STnRP2	STnRP1	STnRP0
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 **STnPAU**: STMn 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, STMn 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 **STnCK2~STnCK0**: 选择 STMn 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: STCKn 上升沿时钟
111: STCKn 下降沿时钟

此三位用于选择 STMn 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 **STnON**: STMn 计数器 On/Off 控制位

0: Off
1: On

此位控制 STMn 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 STMn。清零此位将停止计数器并关闭 STMn 减少耗电。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 STMn 处于比较匹配输出模式时, 当 STnON 位经由低到高的转换时, STMn 输出脚将复位至 STnOC 位指定的初始值。

Bit 2~0 **STnRP2~STnRP0**: STMn CCRP 3-bit 寄存器, 与 STMn 计数器 bit 9~bit 7 比较器 P 匹配周期 =

000: 1024 个 STMn 时钟周期
001: 128 个 STMn 时钟周期
010: 256 个 STMn 时钟周期
011: 384 个 STMn 时钟周期
100: 512 个 STMn 时钟周期
101: 640 个 STMn 时钟周期
110: 768 个 STMn 时钟周期
111: 896 个 STMn 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 STnCCLR 位设为 0 时, 选择 CCRP 比较匹配后清除内部计数器。STnCCLR 位设为低, CCRP 比较匹配结果将重置内部计数器。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。

● STMnC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	STnM1	STnM0	STnIO1	STnIO0	STnOC	STnPOL	STnDPX	STnCCLR
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 **STnM1~STnM0**: 选择 STMn 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 STMn 需要的工作模式。为了确保操作可靠，STMn 应在 STnM1 和 STnM0 位有任何改变前先关掉。在定时 / 计数器模式，STMn 输出引脚状态未定义。

Bit 5~4 **STnIO1~STnIO0**: 选择 STMn 外部引脚功能

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 STPnI 上升沿输入捕捉
- 01: 在 STPnI 下降沿输入捕捉
- 10: 在 STPnI 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 STMn 外部引脚如何改变状态。这两位值的选择取决于 STMn 运行在何种模式下。

在比较匹配输出模式下，STnIO1 和 STnIO0 位决定当从比较器 A 比较匹配输出发生时 STMn 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 STMn 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。STMn 输出脚的初始值通过 STMnC1 寄存器的 STnOC 位设置取得。注意，由 STnIO1 和 STnIO0 位得到的输出电平必须与通过 STnOC 位设置的初始值不同，否则当比较匹配发生时，STMn 输出脚将不会发生变化。在 STMn 输出脚改变状态后，通过 STnON 位由低到高电平的转换复位至初始值。

在 PWM 输出模式，STnIO1 和 STnIO0 决定比较匹配条件发生时怎样改变 STPn 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 STMn 关闭时改变 STnIO1 和 STnIO0 位的值是很有必要的。若在 STMn 运行时改变 STnIO1 和 STnIO0 的值，PWM 输出的值将无法预料。

Bit 3 **STnOC**: STMn 输出脚 STPn 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 STMn 输出脚输出控制位。它取决于 STMn 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 STMn 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 STMn 输出脚 STPn 的逻辑电平值。在 PWM 输出模式 / 单脉冲输出模式时，其决定 PWM 信号是高有效还是低有效。在单脉冲输出模式，其决定 STnON 位由低变高时 STMn 输出脚的逻辑电平。

- Bit 2 **STnPOL:** STPn 输出极性控制位
0: 同相
1: 反相
此位控制 STPn 输出脚的极性。此位为高时 STPn 输出脚反相，为低时 STPn 输出脚同相。若 STMn 处于定时 / 计数器模式时其不受影响。
- Bit 1 **STnDPX:** STMn 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 **STnCCLR:** 选择 STMn 计数器清零条件位
0: STMn 比较器 P 匹配
1: STMn 比较器 A 匹配
此位用于选择清除计数器的方法。标准型 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。STnCCLR 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。STnCCLR 位在 PWM 输出，单脉冲输出或输入捕捉模式时未使用。

• STMnDL 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 STMn 计数器低字节寄存器 bit 7~bit 0
STMn 10-bit 计数器 bit 7~bit 0

• STMnDH 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 STMn 计数器高字节寄存器 bit 1 ~ bit 0
STMn 10-bit 计数器 bit 9 ~ bit 8

• STMnAL 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 STMn CCRA 低字节寄存器 bit 7~bit 0
STMn 10-bit CCRA bit 7~bit 0

• STMnAH 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 STMn CCRA 高字节寄存器 bit 1 ~ bit 0

STMn 10-bit CCRA bit 9 ~ bit 8

标准型 TM 工作模式

标准型 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 STMnC1 寄存器的 STnM1 和 STnM0 位选择任意模式。

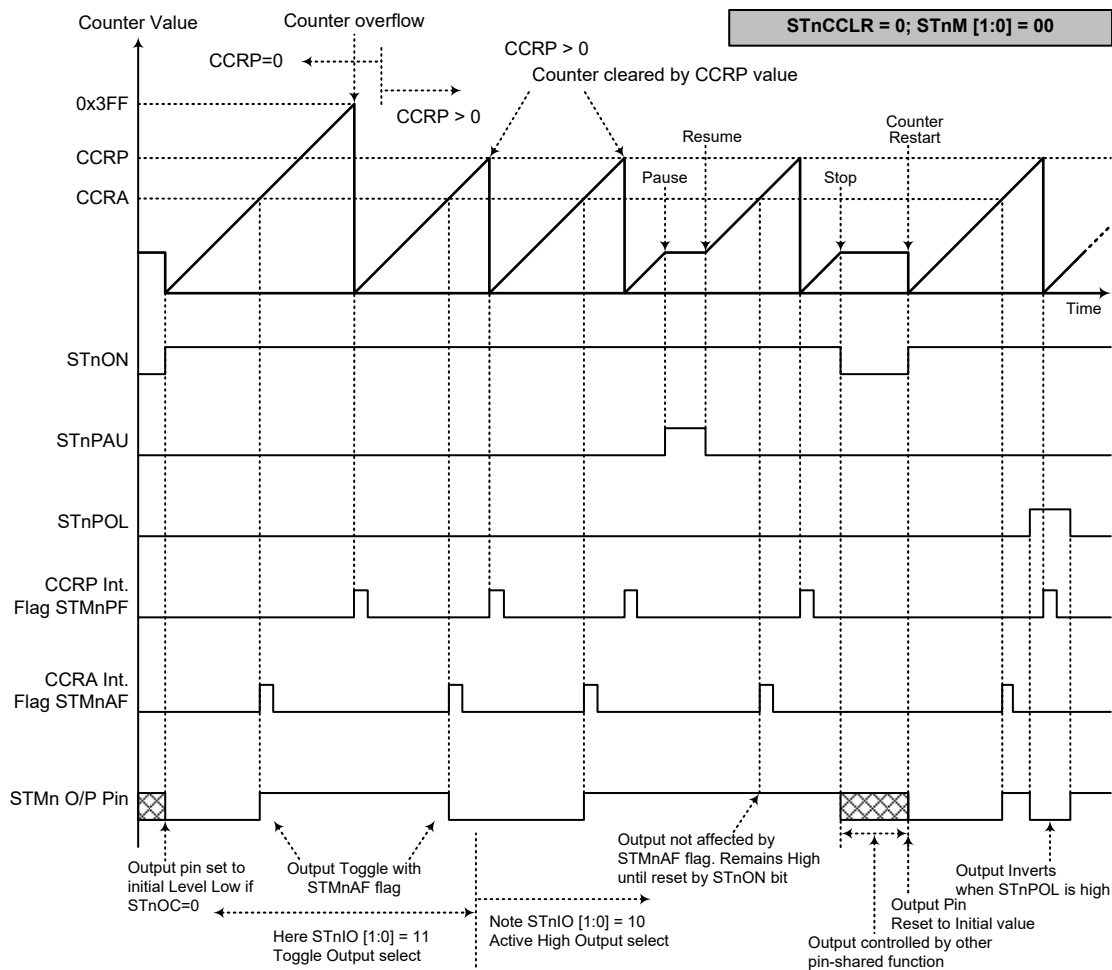
比较匹配输出模式

为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 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 STnCCLR 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 STMnAF 和 STMnPF 将分别置位。

如果 STMnC1 寄存器的 STnCCLR 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，仅产生 STMnAF 中断请求标志。所以当 STnCCLR 为高时 nbvc，不会产生 STMnPF 中断请求标志。在比较匹配输出模式下，CCRA 不能设为“0”。

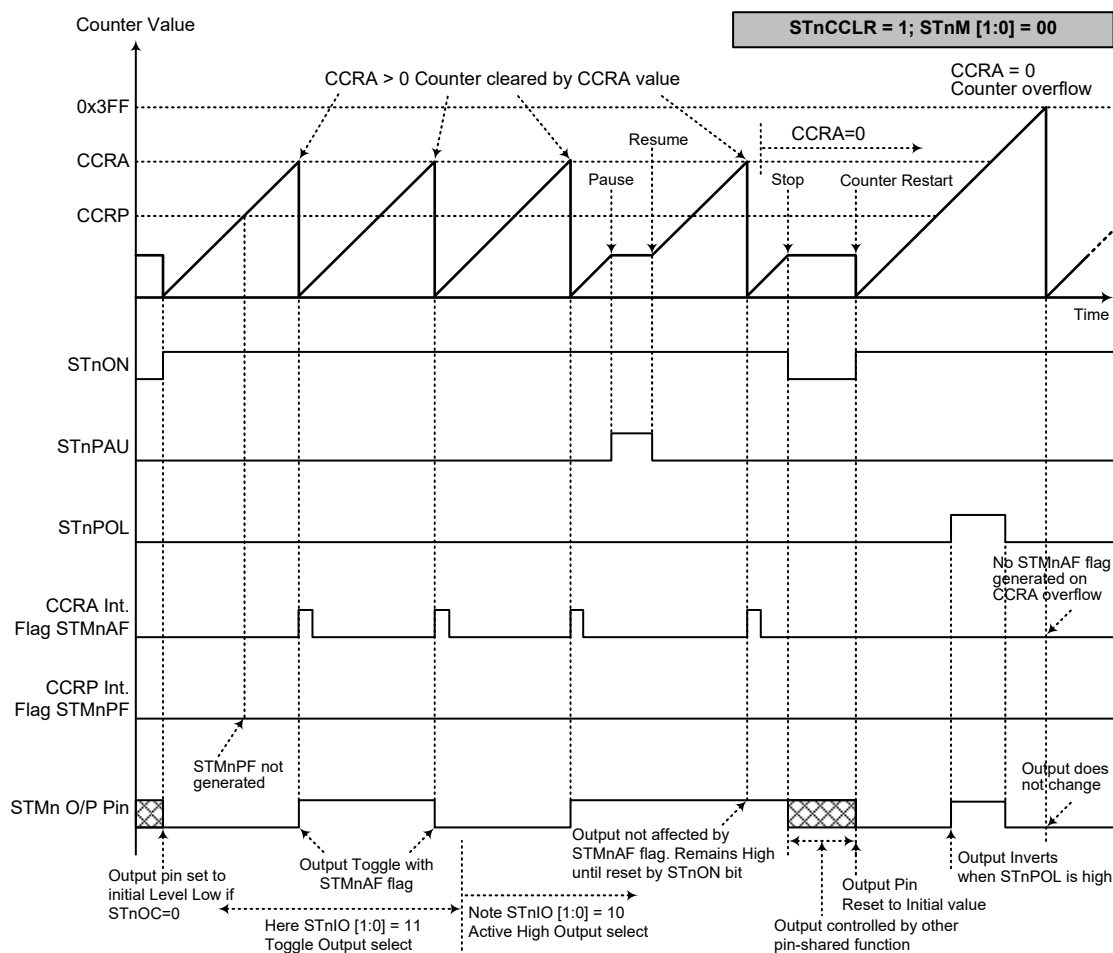
如果 CCRA 位都清为零，当计数器的值达到 10 位最大值 3FFH 时将溢出，但此时不会产生 STMnAF 中断请求标志。

正如该模式名所言，当比较匹配发生后，STMn 输出脚状态改变。当比较器 A 比较匹配发生后 STMnAF 标志产生时，STMn 输出脚状态改变。比较器 P 比较匹配发生时产生的 STMnPF 标志不影响 STMn 输出脚。STMn 输出脚状态改变方式由 STMnC1 寄存器中 STnIO1 和 STnIO0 位决定。当比较器 A 比较匹配发生时，STnIO1 和 STnIO0 位决定 STMn 输出脚输出高，低或翻转当前状态。STMn 输出脚初始值，在 STnON 位由低到高电平的变化后通过 STnOC 位设置。注意，若 STnIO1 和 STnIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 – STnCCR=0 (n=0~1)

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



比较匹配输出模式 – STnCCR=1 (n=0~1)

- 注：1. STnCCR=1，比较器 A 匹配将清除计数器
2. STMn 输出脚仅由 STMnAF 标志位控制
3. 在 STnON 上升沿 STMn 输出脚复位至初始值
4. 当 STnCCR=1 时，不会产生 STMnPF 标志位

定时 / 计数器模式

为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 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 STMn 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 STMn 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 位需要设置为“10”，且 STnIO1 和 STnIO0 位也需要设置为“10”。STMn 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 STMn 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就较为灵活。在 PWM 输出模式中，STnCCLR 位不影响 PWM 周期。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 STMnC1 寄存器的 STnDPX 位。所以 PWM 波形由 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。STMnC1 寄存器中的 STnOC 位决定 PWM 波形的极性，STnIO1 和 STnIO0 位使能 PWM 输出或将 TM 输出脚置为逻辑高或逻辑低。STnPOL 位对 PWM 输出波形的极性取反。

• 10-bit STMn, PWM 输出模式，边沿对齐模式，STnDPX=0

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

若 $f_{SYS}=4\text{MHz}$ ，STMn 时钟源为 $f_{SYS}/4$ ，CCRP=4，CCRA=128，

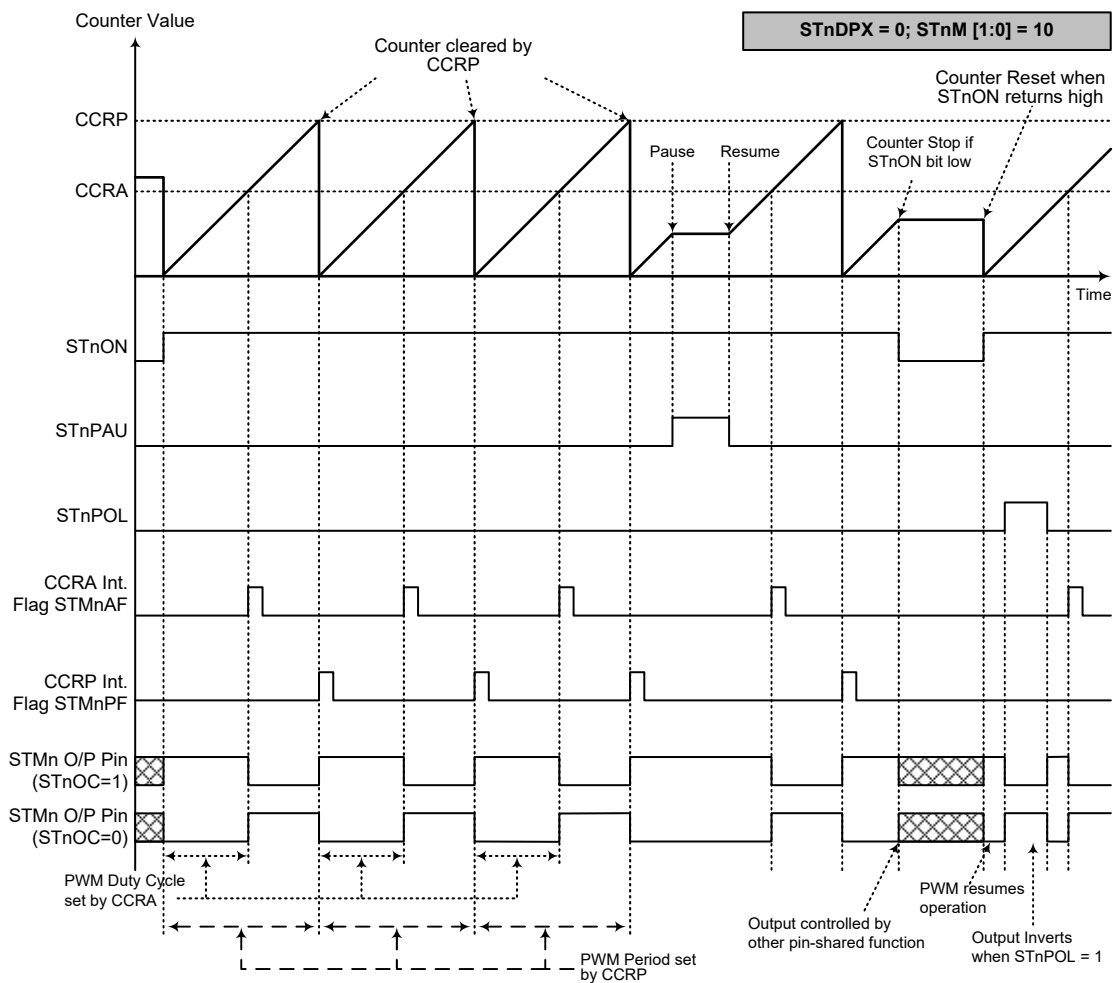
STMn PWM 输出频率 = $(f_{SYS}/4)/(4 \times 128) = f_{SYS}/2048 = 2\text{kHz}$ ，duty = $128/(4 \times 128) = 25\%$ ，

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%。

• 10-bit STMn, PWM 输出模式，边沿对齐模式，STnDPX=1

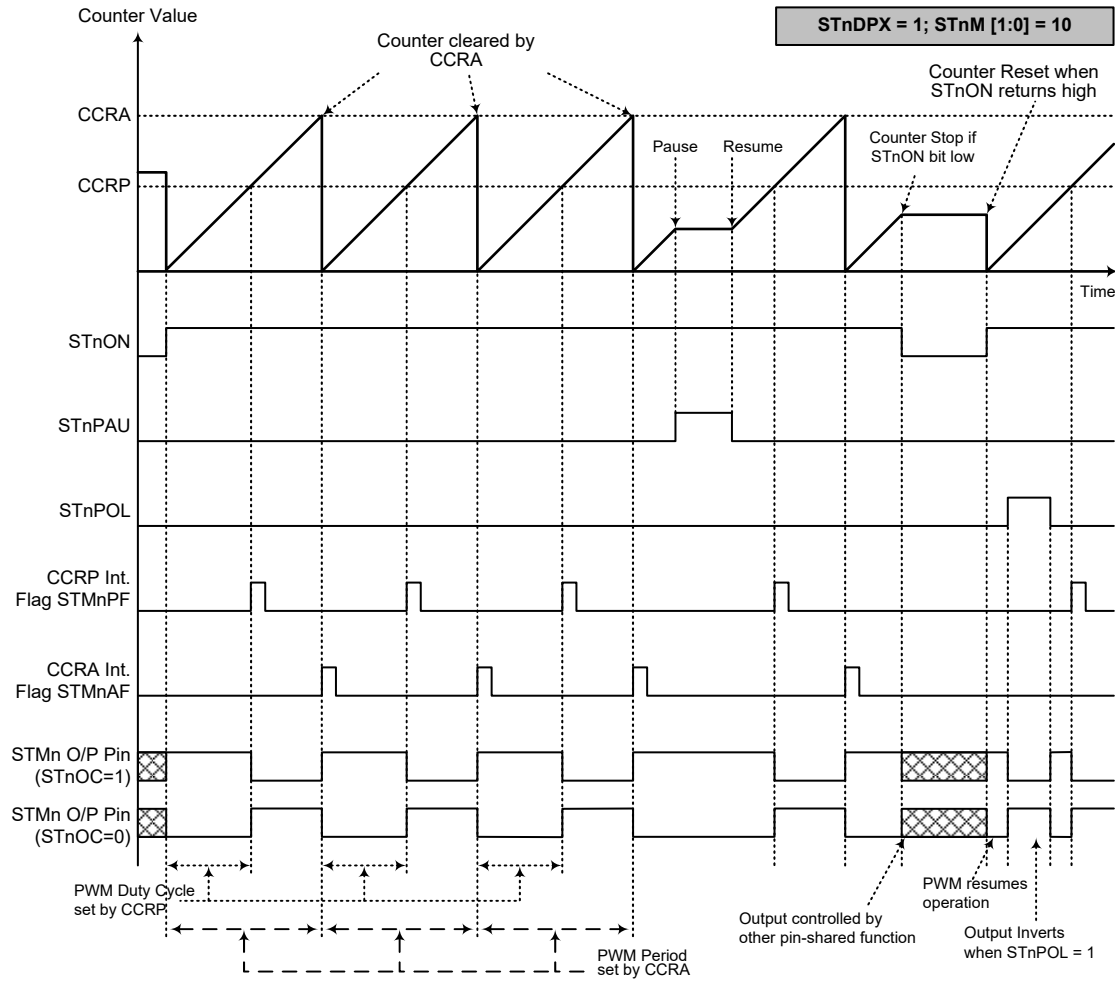
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

PWM 的输出周期由 CCRA 寄存器的值与 STMn 的时钟共同决定，PWM 的占空比由 CCRP×256 (除了 CCRP 为“0”外) 的值决定。



PWM 输出模式 – STnDPX=0 (n=0~1)

- 注：1. STnDPX=0, CCRP 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 STnIO[1:0]=00 或 01, PWM 功能不变
4. STnCCLR 位不影响 PWM 操作



PWM 输出模式 – STnDPX=1 (n=0~1)

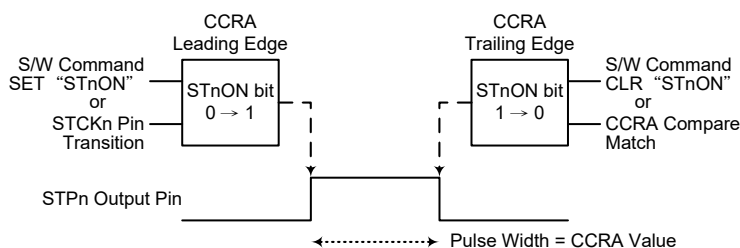
- 注：1. STnDPX=1, CCRA 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 STnIO[1:0]=00 或 01, PWM 功能不变
4. STnCCLR 位不影响 PWM 操作

单脉冲输出模式

为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 位需要设置为“10”，同时 STnIO1 和 STnIO0 位需要设置为“11”。正如模式名所言，单脉冲输出模式，在 STMn 输出脚将产生一个脉冲输出。

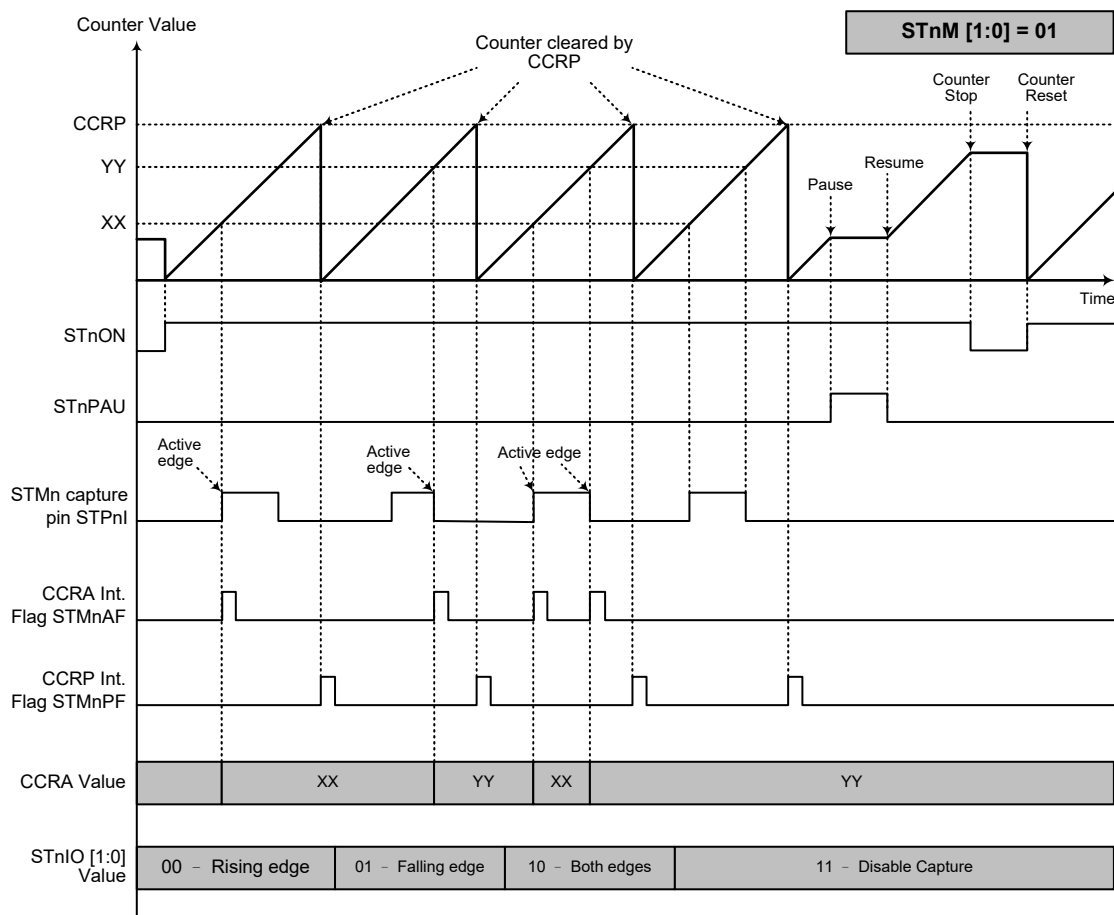
脉冲输出可以通过应用程序控制 STnON 位由低到高的转变来触发脉冲前沿输出。而处于单脉冲输出模式时，STnON 位可由 STnCK 脚自动由低转变为高，进而开始单脉冲输出。当 STnON 位转变为高电平时，计数器将开始运行，并产生脉冲前沿。当脉冲有效时 STnON 位保持高电平。通过应用程序使 STnON 位清零或比较器 A 比较匹配发生时，产生脉冲下降沿。

然而，比较器 A 比较匹配发生时，会自动清除 STnON 位并产生单脉冲输出边沿跳变。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 STMn 中断。STnON 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，STnCCLR 和 STnDPX 位未使用。



单脉冲产生示意图

捕捉操作，但计数器仍会继续运行。当 STPnI 引脚与其它功能共用，STMn 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。STnCCLR 和 STnDPX 位在此模式中未使用。

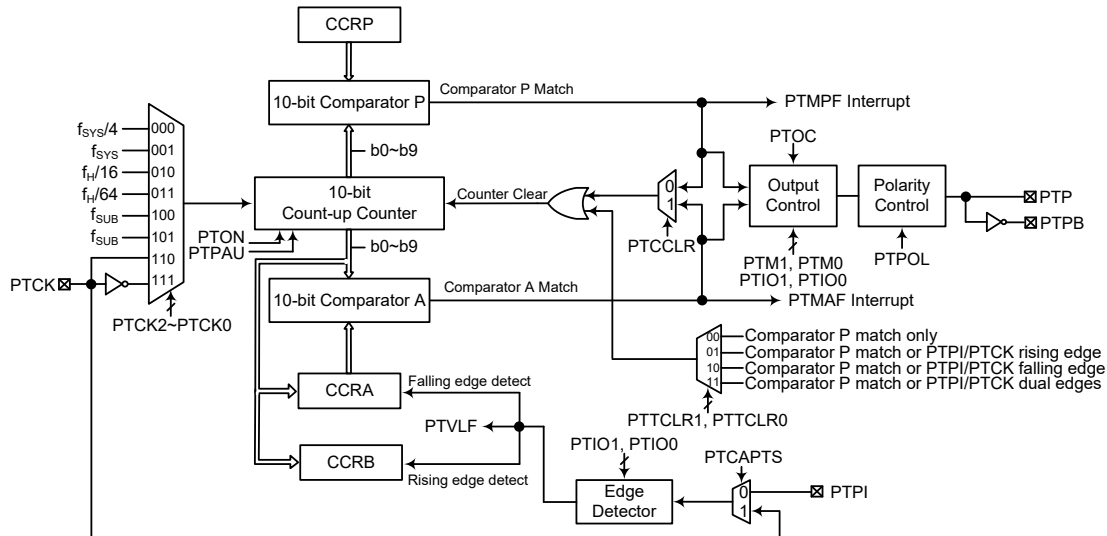


捕捉输入模式 (n=0~1)

- 注：1. STnM[1:0]=01 并通过 STnIO1 和 STnIO0 位设置有效边沿
2. STMn 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中
3. STnCCLR 位未使用
4. 无输出功能 - STnOC 和 STnPOL 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

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：1. PTPB 为 PTP 的反相输出。
2. PTM 外部引脚与其它功能共用，所以在使用 PTM 功能前引脚共用功能寄存器必须正确设置以使用 PTM 引脚功能。若使用 PTCK 和 PTPI 引脚，还需通过设定端口控制寄存器中的对应位设置对应引脚为输入。
3. 当使用 PTPI 引脚时，需确保 IFS0[7:6] 字段值保持为“01”。

周期型 TM 方框图

周期型 TM 操作

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

通过应用程序改变 10-bit 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或在捕捉输入模式下通过设置 PTTCLR[1:0] 位选择有效触发沿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，可由包括来自两个输入脚的不同时钟源驱动，也可以控制多个输出脚。所有工作模式的设定都是通过设置相关寄存器来实现的。

周期型 TM 寄存器介绍

周期型 TM 的所有操作由一系列寄存器控制。一对只读寄存器用来存放 10-bit 计数器的值，三对读 / 写寄存器存放 10-bit CCRA 值、CCRP 值和 CCRB 值。剩下三个控制寄存器用来设置不同的操作和控制模式。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
PTMC2	—	—	—	—	—	PTTCLR1	PTTCLR0	PTVLF
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	—	—	—	—	—	—	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	—	—	—	—	—	—	D9	D8
PTMBL	D7	D6	D5	D4	D3	D2	D1	D0
PTMBH	—	—	—	—	—	—	D9	D8
PTMRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMRPH	—	—	—	—	—	—	D9	D8

10-bit 周期型 TM 寄存器列表

● PTMC0 寄存器

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

- Bit 7 **PTPAU**: PTM 计数器暂停控制位
 0: 运行
 1: 暂停
 通过设置此位为高可使计数器暂停，清零此位恢复正常计数器操作。当处于暂停条件时，PTM 保持上电状态并继续耗电。当此位由低到高转变时，计数器将保留其剩余值，直到此位再次改变为低电平，并从此值开始继续计数。
- Bit 6~4 **PTCK2~PTCK0**: PTM 计数器时钟选择位
 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 整体 On/Off 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、PWM 输出模式或单脉冲输出模式时，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**: PTM 工作模式选择位

- 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 输出引脚状态未知。

Bit 5~4 **PTIO1~PTIO0**: PTM 外部引脚功能选择位

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

PTTCLR[1:0]=00B:

- 00: 在 PTPI 或 PTCK 上升沿输入捕捉, 计数器值将锁存至 CCRA
- 01: 在 PTPI 或 PTCK 下降沿输入捕捉, 计数器值将锁存至 CCRA
- 10: 在 PTPI 或 PTCK 双沿输入捕捉, 计数器值将锁存至 CCRA
- 11: 输入捕捉除能

PTTCLR[1:0]=01B、10B 或 11B:

- 00: 在 PTPI 或 PTCK 上升沿输入捕捉, 计数器值将锁存至 CCRB
- 01: 在 PTPI 或 PTCK 下降沿输入捕捉, 计数器值将锁存至 CCRA
- 10: 在 PTPI 或 PTCK 双沿输入捕捉, 下降沿计数器值将锁存至 CCRA, 上升沿计数器值将锁存至 CCRB
- 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 功能如何改变状态。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, 其决定比较匹配发生前 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。在单脉冲输出模式时, 其决定 PTON 位由低变高时 PTM 输出脚的逻辑电平。
- 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**: PTM 捕捉触发源控制位
0: 来自 PTPI 输入信号
1: 来自 PTCK 引脚
- Bit 0 **PTCCLR**: PTM 计数器清零条件选择位
0: PTM 比较器 P 匹配
1: PTM 比较器 A 匹配
此位用于选择清除计数器的方法。周期型 TM 包括两个比较器 – 比较器 A 和比较器 P, 两者都可以用作清除内部计数器。PTCCLR 位设为高, 计数器在比较器 A 比较匹配发生时被清除; 此位设为低, 计数器在比较器 P 比较匹配发生或计数器溢出时被清除。计数器溢出清除的方法仅在 CCRP 被清除为 0 时才能生效。PTCCLR 位在 PWM 输出模式、单脉冲输出模式或输入捕捉模式时未使用。

● PTMC2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PTTCLR1	PTTCLR0	PTVLF
R/W	—	—	—	—	—	R/W	R/W	R
POR	—	—	—	—	—	0	0	0

- Bit 7~3 未定义, 读为 “0”
- Bit 2~1 **PTTCLR1~PTTCLR0**: 捕捉输入模式时 PTM 计数器清零条件选择位
00: 比较器 P 比较匹配
01: 比较器 P 比较匹配或 PTCK/PTPI 上升沿
10: 比较器 P 比较匹配或 PTCK/PTPI 下降沿
11: 比较器 P 比较匹配或 PTCK/PTPI 双沿
注意, PTTCLR1~PTTCLR0 位仅在 PTM 处于捕捉输入模式时可用。
- Bit 0 **PTVLF**: PTM 计数器值锁存边沿标志位
0: 下降沿触发计数器值锁存
1: 上升沿触发计数器值锁存
当 PTTCLR1~PTTCLR0 位为 00B 时, 忽略该标志位状态。

● PTMDL 寄存器

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

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

● PTMDH 寄存器

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

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

● PTMAL 寄存器

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

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

● PTMAH 寄存器

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

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

● PTMBL 寄存器

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

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

● PTMBH 寄存器

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

Bit 7~2 未定义，读为“0”

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

● PTMRPL 寄存器

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

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

● PTMRPH 寄存器

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

Bit 7~2 未定义，读为“0”

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

周期型 TM 工作模式

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

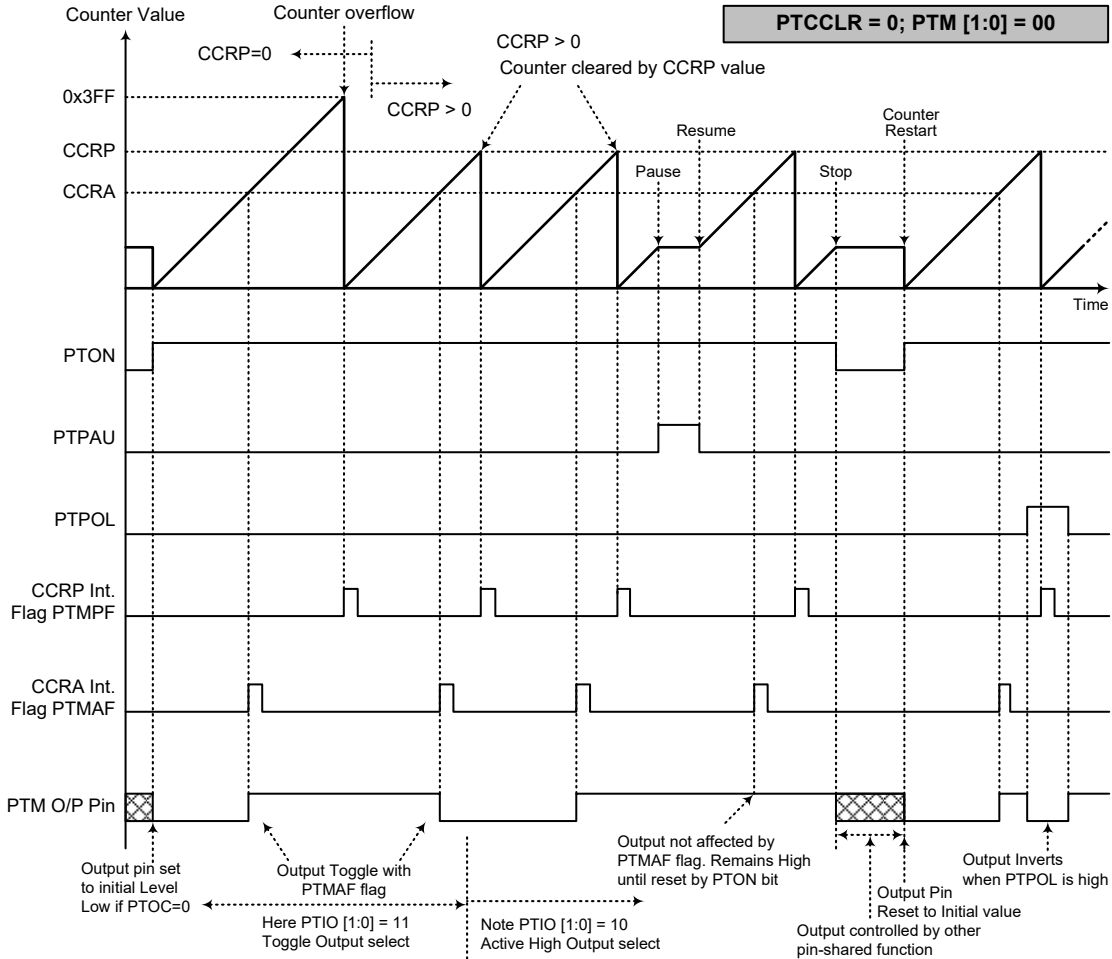
比较匹配输出模式

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

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

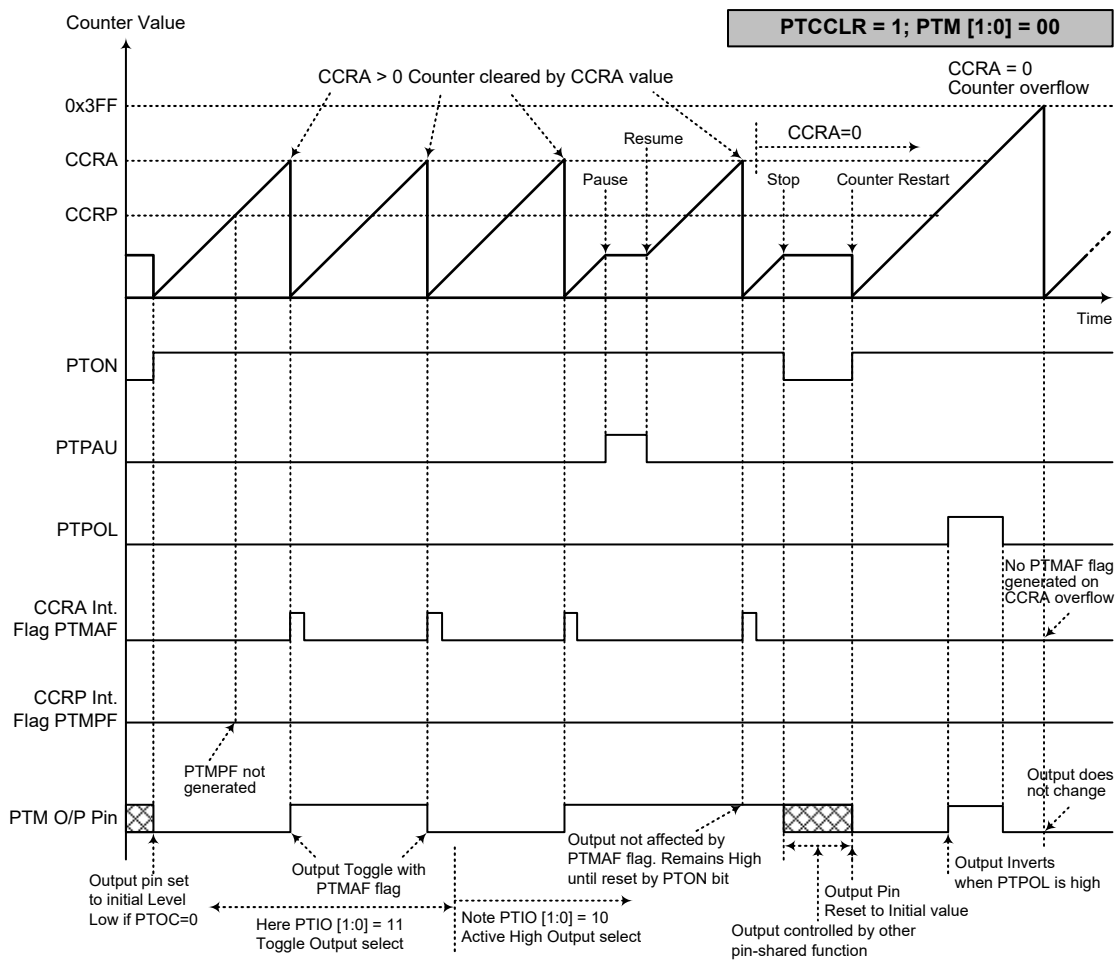
如果 CCRA 位都清除为零，当计数器的值达到 10 位最大值 3FFH 时将溢出，但此时不会产生 PTMAF 中断请求标志。

正如该模式名所言，当比较匹配发生后，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。



比较器匹配输出模式 – PTCCLR=0

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



比较器匹配输出模式 – PTCCLR=1

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

定时 / 计数器模式

为使 PTM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 位需要设置为“11”。定时 / 计数器模式与比较输出模式操作方式相同，并产生同样的中断请求标志。不同的是，在定时 / 计数器模式下 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。该模式中未使用的 PTM 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 PTM 工作在此模式，PTMC1 寄存器的 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，其波形的选择就较为灵活。在 PWM 输出模式中，PTCCLR 位对 PWM 周期无影响。CCRP 和 CCRA 寄存器都用于控制 PWM 方波。CCRP 寄存器通过清除内部计数从而控制 PWM 周期，CCRA 寄存器设置 PWM 的占空比。PWM 波形的周期和占空比由 CCRP 和 CCRA 寄存器的值控制。

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

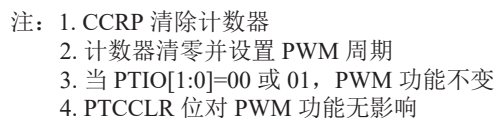
● 10-bit PTM，PWM 输出模式，边沿对齐模式

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

若 $f_{SYS}=8\text{MHz}$ ，PTM 时钟源选择 $f_{SYS}/4$ ，CCRP=512 且 CCRA=128，

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

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%。

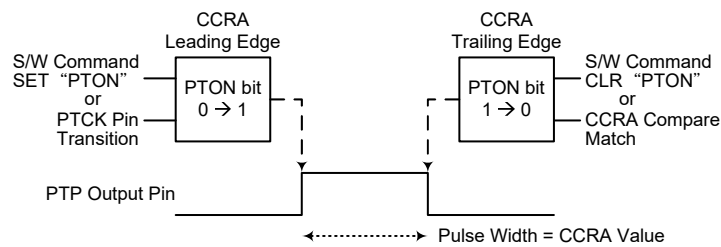


单脉冲输出模式

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 位可在 PTCK 脚发生有效边沿跳转时自动由低转变为高，进而开始单脉冲输出。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，产生脉冲后沿。

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



单脉冲产生示意图



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. 在单脉冲输出模式，PTIO[1:0] 需设置为“11”且不可改变

捕捉输入模式

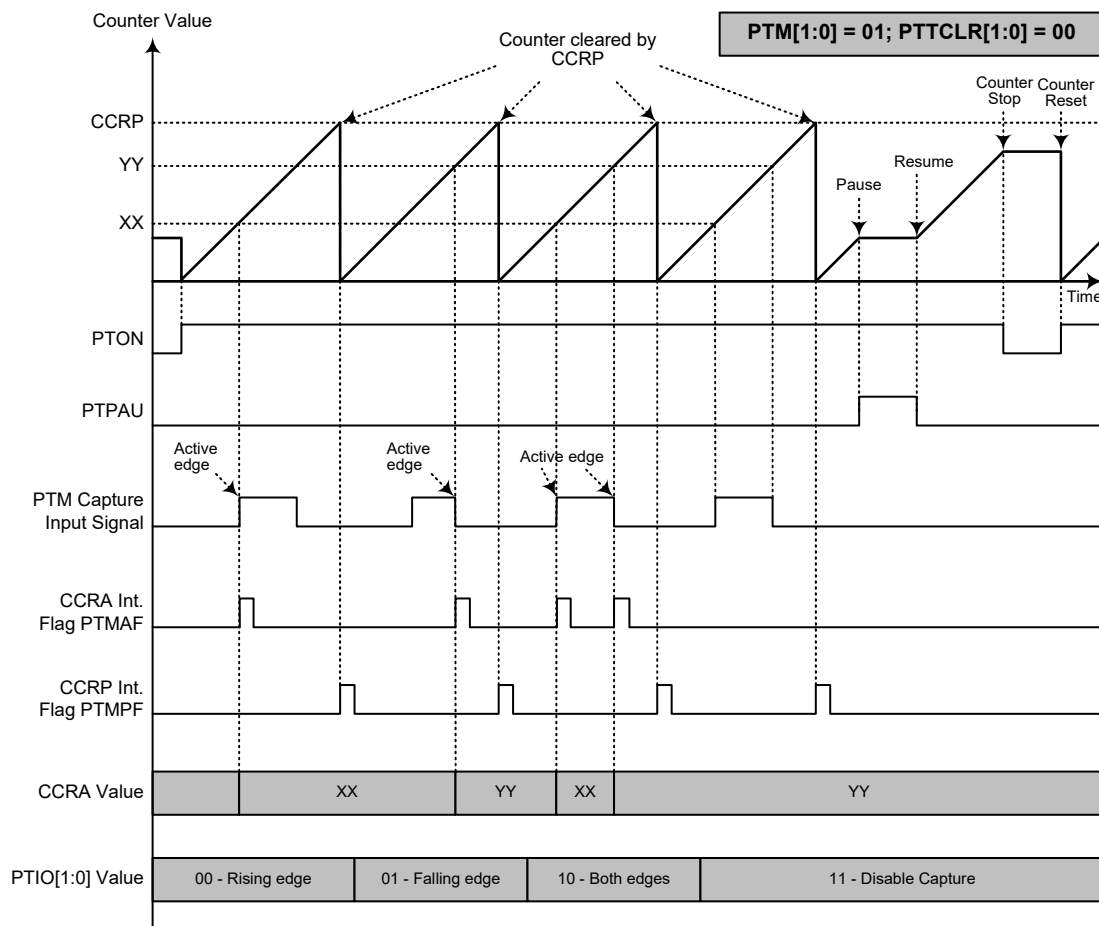
为使 PTM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 位需要设置为“01”。此模式使能外部或内部信号捕捉并保存内部计数器当前值，因此被用于诸如脉冲宽度测量的应用中。通过设置 PTMC1 寄存器的 PTCAPTS 位选择捕捉输入信号来自 PTPI 信号还是 PTCK 引脚，其中 PTPI 信号可由 IFS0 寄存器的 IFS07~IFS06 位选择来自外部引脚输入。可通过设置 PTMC1 寄存器的 PTIO1 和 PTIO0 位选择有效边沿类型，即上升沿，下降沿或双沿有效。通过应用程序将 PTON 位由低变为高时，计数器启动。

PTIO1 和 PTIO0 位决定触发产生中断且被锁存的有效边沿。PTTCLR1 和 PTTCLR0 位决定计数器复位至零的条件。当前计数器值是锁存至 CCRA 还是 CCRB 取决于 PTIO1~PTIO0 位和 PTTCLR1~PTTCLR0 位的设置。PTIO1~PTIO0 位和 PTTCLR1~PTTCLR0 位的设置是相互独立且不相互影响的。

当选择的输入信号出现有效边沿转换时，计数器当前值被锁存到 CCRA 或 CCRB 寄存器，并产生 PTM 中断。无论选择的输入信号发生哪种边沿转换，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；通过这种方式 CCRP 的值可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 位选择输入信号为上升沿，下降沿或双沿有效。如果 PTIO1 和 PTIO0 位都设置为高，无论选择的输入信号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。

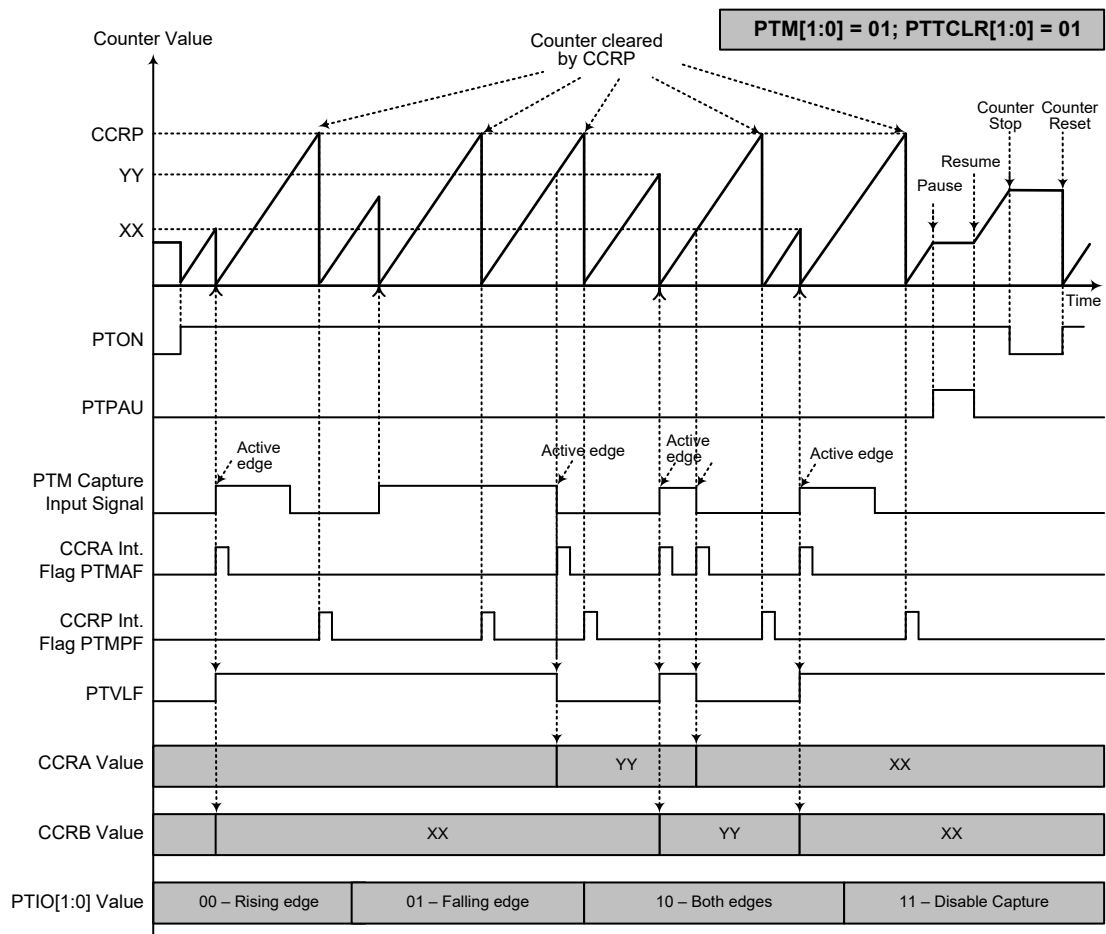
若捕捉脉宽小于两个 PTM 时钟周期，硬件可能会忽略该脉冲。PTM 时钟必须小于或等于 50MHz，否则计数器可能计数失败。

当 PTCK 或 PTPI 引脚与其它功能共用，PTM 工作在捕捉输入模式时需多加注意。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。



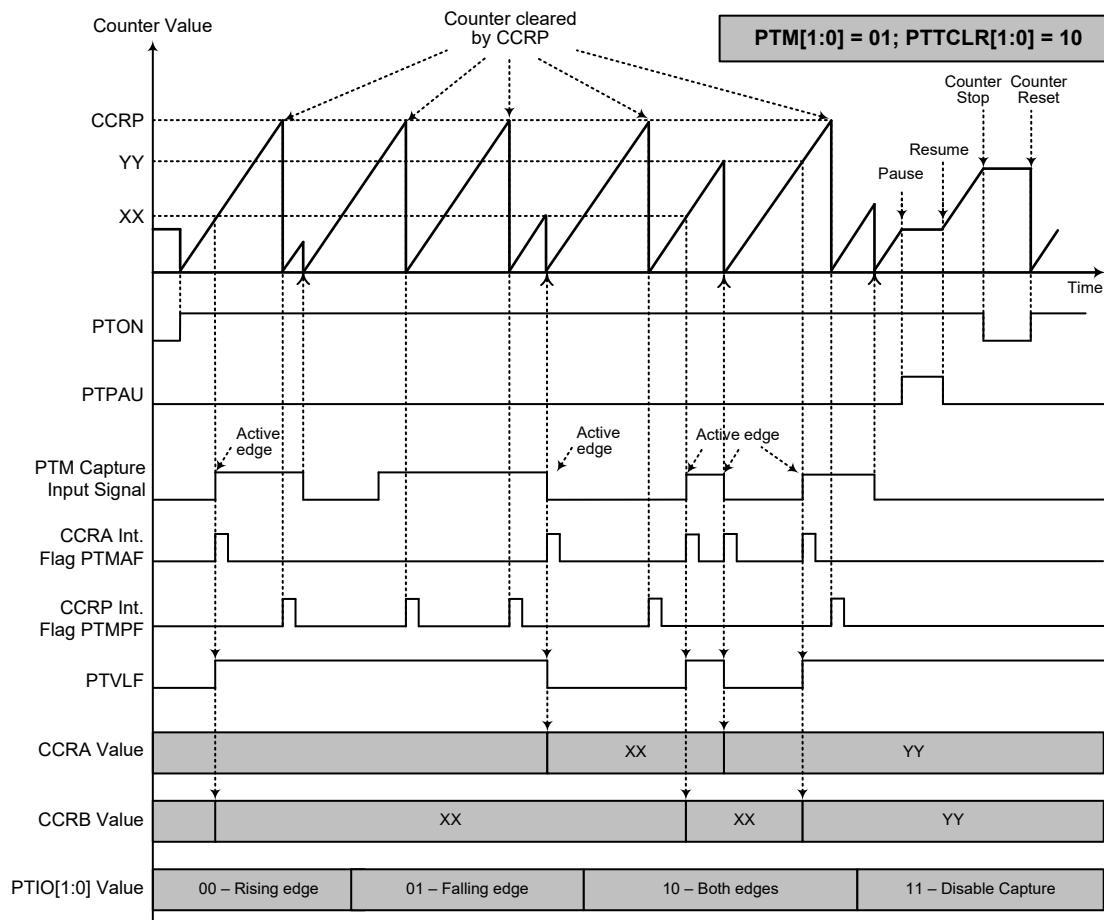
捕捉输入模式 – PTTCLR[1:0]=00

- 注：1. PTM[1:0]=01，PTTCLR[1:0]=00 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. 比较器 P 比较匹配，计数器清零。
4. PTTCLR 位未使用
5. 无输出功能 – PTOC 和 PTPOL 位未使用
6. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大
7. 当 PTTCLR[1:0]=00 时忽略 PTVLF 位的状态



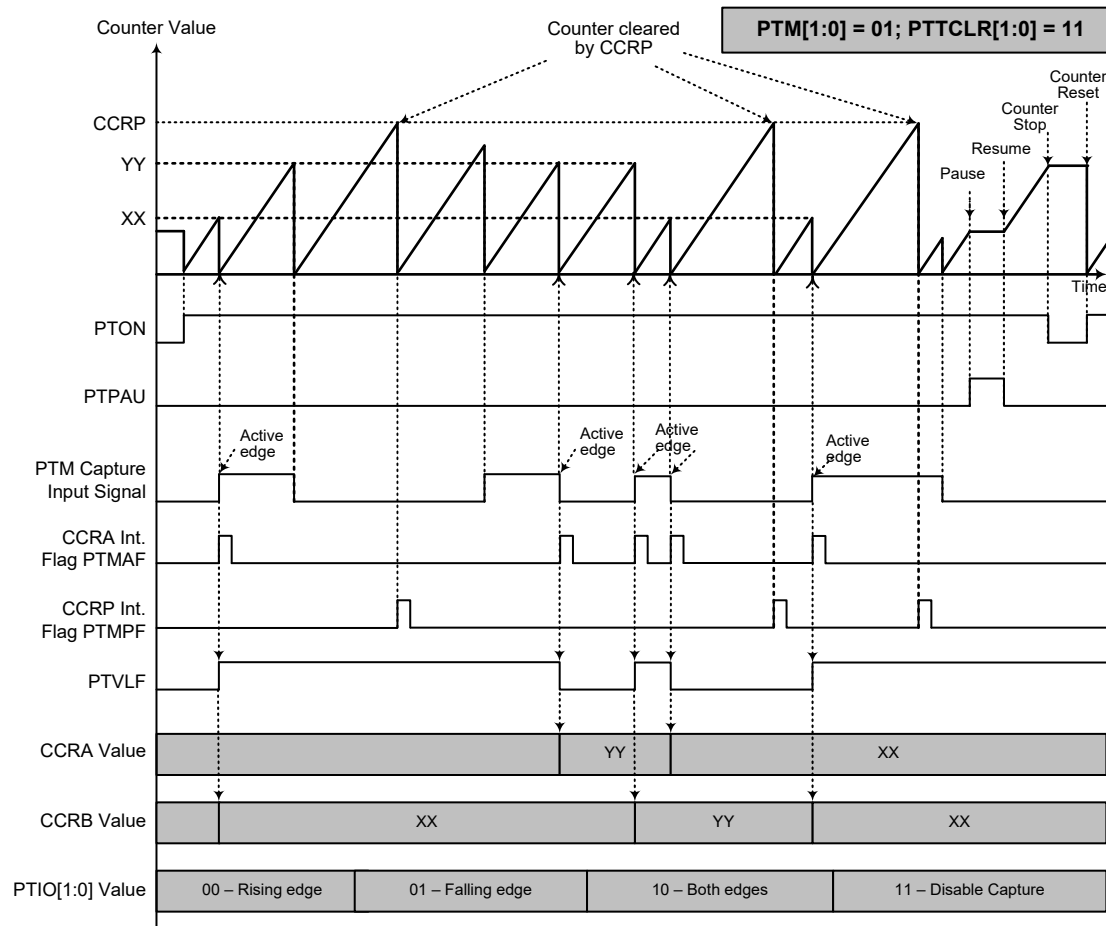
捕捉输入模式 – PTTCLR[1:0]=01

- 注：1. PTM[1:0]=01，PTTCLR[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 或 CCRB 中
3. 比较器 P 比较匹配或 PTM 捕捉输入上升沿时，计数器清零。
4. PTCCLR 位未使用
5. 无输出功能 – PTOC 和 PTPOL 位未使用
6. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大



捕捉输入模式 – PTTCLR[1:0]=10

- 注：1. PTM[1:0]=01，PTTCLR[1:0]=10 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 或 CCRB 中
3. 比较器 P 比较匹配或 PTM 捕捉输入下降沿时，计数器清零。
4. PTCCLR 位未使用
5. 无输出功能 – PTOC 和 PTPOL 位未使用
6. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大

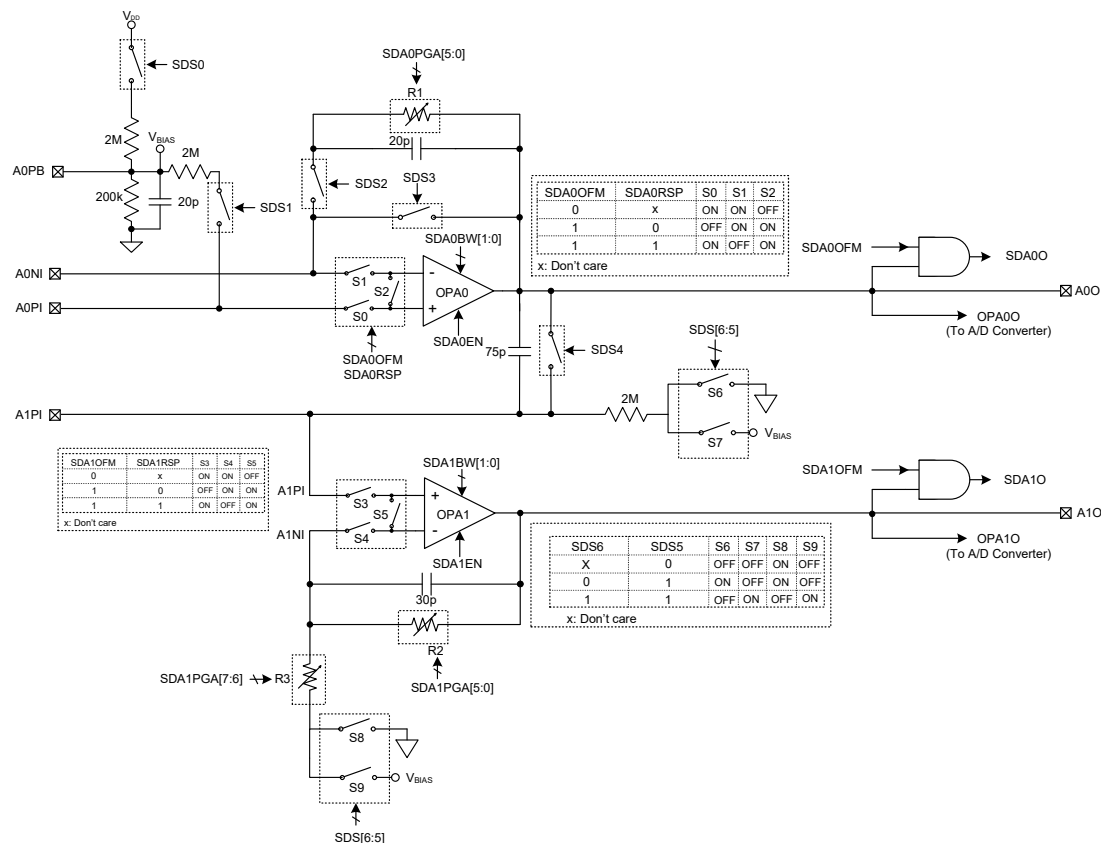


捕捉输入模式 – PTTCLR[1:0]=11

- 注：1. PTM[1:0]=01，PTTCLR[1:0]=11 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 或 CCRB 中
3. 比较器 P 比较匹配或 PTM 捕捉输入上升沿或下降沿时，计数器清零。
4. PTCCLR 位未使用
5. 无输出功能 – PTOC 和 PTPOL 位未使用
6. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大

感烟探测器 AFE

该单片机提供一个感烟探测器 AFE 电路，可用于感烟探测器应用的光信号检测。该电路包含两个完全集成的运算放大器。光信号可通过这两个运算放大器检测和处理。



感烟探测器 AFE 方框图

应注意，虽然 SD OPAm 带宽由 SDAmBW1~SDAmBW0 位决定，但 OPAm 搭配 A/D 转换器使用时则有一些限制。由于 OPAm 带宽导致输出电流较小，所以选择 SD OPAm 带宽就必须小心。使用者可以参考下面的表格，被标上 √ 的数值是可使用的。应确保 12-bit A/D 转换器读取到的数值已小于 1 LSB。

SD OPAm 带宽选择	A/D 转换器时钟频率 (kHz)							
	15.625	31.25	62.5	125	250	500	1000	2000
SDAmBW[1:0]=00	√	—	—	—	—	—	—	—
SDAmBW[1:0]=01	√	√	√	√	—	—	—	—
SDAmBW[1:0]=10	√	√	√	√	√	√	√	√
SDAmBW[1:0]=11	√	√	√	√	√	√	√	√

感烟探测器 AFE SD OPAm 带宽范例 (m=0~1)

感烟探测器 AFE 寄存器

感烟探测器 AFE 电路的所有操作由一系列寄存器控制。SDSW 寄存器用于控制开关 On/Off，进而控制运算放大器的工作模式。SDPGAC0 和 SDPGAC1 寄存器用于选择 R1、R2 和 R3 的阻值。SDAnC 寄存器用于控制 SD OPAn 使能/除能、带宽以及储存输出状态，其中 n=0~1。SDAnVOS 寄存器用于控制 SD OPAn 输入失调电压校准功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SDSW	—	SDS6	SDS5	SDS4	SDS3	SDS2	SDS1	SDS0
SDPGAC0	—	—	SDA0PGA5	SDA0PGA4	SDA0PGA3	SDA0PGA2	SDA0PGA1	SDA0PGA0
SDPGAC1	SDA1PGA7	SDA1PGA6	SDA1PGA5	SDA1PGA4	SDA1PGA3	SDA1PGA2	SDA1PGA1	SDA1PGA0
SDA0C	—	SDA0EN	SDA0O	—	—	—	SDA0BW1	SDA0BW0
SDA1C	—	SDA1EN	SDA1O	—	—	—	SDA1BW1	SDA1BW0
SDA0VOS	SDA0OFM	SDA0RSP	SDA0OF5	SDA0OF4	SDA0OF3	SDA0OF2	SDA0OF1	SDA0OF0
SDA1VOS	SDA1OFM	SDA1RSP	SDA1OF5	SDA1OF4	SDA1OF3	SDA1OF2	SDA1OF1	SDA1OF0

感烟探测器 AFE 寄存器列表

• SDSW 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	SDS6	SDS5	SDS4	SDS3	SDS2	SDS1	SDS0
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 6~5 **SDS6~SDS5**: 模式控制位
 00: 外部模式
 01: AC 耦合模式
 10: 外部模式
 11: DC 耦合模式 (SDS1 开关不能同时 On)

Bit 4 **SDS4**: SDS4 On/Off 控制位
 0: Off
 1: On

Bit 3 **SDS3**: SDS3 On/Off 控制位
 0: Off
 1: On

Bit 2 **SDS2**: SDS2 On/Off 控制位
 0: Off
 1: On

Bit 1 **SDS1**: SDS1 On/Off 控制位
 0: Off
 1: On

Bit 0 **SDS0**: SDS0 On/Off 控制位
 0: Off
 1: On

● SDPGAC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	SDA0PGA5	SDA0PGA4	SDA0PGA3	SDA0PGA2	SDA0PGA1	SDA0PGA0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 未定义，读为“0”

Bit 5~0 **SDA0PGA5~SDA0PGA0**: R1 阻值控制位

$R1 = SDA0PGA[5:0] \times 100k\Omega$

这些位用于选择 R1 阻值，应注意当这些位设置为“000000”时 $R1 \neq 0\Omega$ 。

● SDPGAC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SDA1PGA7	SDA1PGA6	SDA1PGA5	SDA1PGA4	SDA1PGA3	SDA1PGA2	SDA1PGA1	SDA1PGA0
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 **SDA1PGA7~SDA1PGA6**: R3 阻值控制位

00: 10k Ω

01: 20k Ω

10: 30k Ω

11: 40k Ω

Bit 5~0 **SDA1PGA5~SDA1PGA0**: R2 阻值控制位

$R2 = SDA1PGA[5:0] \times 100k\Omega$

这些位用于选择 R2 阻值，应注意当这些位设置为“000000”时 $R2 \neq 0\Omega$ 。

● SDA0C 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	SDA0EN	SDA0O	—	—	—	SDA0BW1	SDA0BW0
R/W	—	R/W	R	—	—	—	R/W	R/W
POR	—	0	0	—	—	—	0	0

Bit 7 未定义，读为“0”

Bit 6 **SDA0EN**: SD OPA0 使能 / 除能控制位

0: 除能

1: 使能

Bit 5 **SDA0O**: SD OPA0 输出状态 (正逻辑电平)

该位是只读位。

当 SDA0OFM=1, SDA0O 定义 SD OPA0 输出状态，详细内容参考“运算放大器输入失调校准”章节。

当 SDA0OFM=0, 该位固定为低电平。

Bit 4~2 未定义，读为“0”

Bit 1~0 **SDA0BW1~SDA0BW0**: SD OPA0 带宽控制位

00: 5kHz

01: 40kHz

10: 600kHz

11: 2MHz

详细内容参考“运算放大器电气特性”。

• SDA1C 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	SDA1EN	SDA1O	—	—	—	SDA1BW1	SDA1BW0
R/W	—	R/W	R	—	—	—	R/W	R/W
POR	—	0	0	—	—	—	0	0

Bit 7 未定义，读为“0”

Bit 6 **SDA1EN**: SD OPA1 使能 / 除能控制位
0: 除能
1: 使能

Bit 5 **SDA1O**: SD OPA1 输出状态 (正逻辑电平)
该位是只读位。
当 SDA1OFM=1, SDA1O 定义 SD OPA1 输出状态，详细内容参考“运算放大器输入失调校准”章节。
当 SDA1OFM=0, 该位固定为低电平。

Bit 4~2 未定义，读为“0”

Bit 1~0 **SDA1BW1~SDA1BW0**: SD OPA1 带宽控制位
00: 5kHz
01: 40kHz
10: 600kHz
11: 2MHz
详细内容参考“运算放大器电气特性”。

• SDA0VOS 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SDA0OFM	SDA0RSP	SDA0OF5	SDA0OF4	SDA0OF3	SDA0OF2	SDA0OF1	SDA0OF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **SDA0OFM**: SD OPA0 正常操作或输入失调电压校准模式选择位
0: 正常操作
1: 失调校准模式

Bit 6 **SDA0RSP**: SD OPA0 输入失调电压校准参考选择位
0: 输入参考电压来自 A0NI
1: 输入参考电压来自 A0PI

Bit 5~0 **SDA0OF5~SDA0OF0**: SD OPA0 输入失调电压校准控制位
这 6 位用于执行运算放大器的输入失调校准操作，并重新储存 SD OPA0 的输入失调校准值。更多详细资料请参考“运算放大器输入失调校准”章节。

• SDA1VOS 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SDA1OFM	SDA1RSP	SDA1OF5	SDA1OF4	SDA1OF3	SDA1OF2	SDA1OF1	SDA1OF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **SDA1OFM**: SD OPA1 正常操作或输入失调电压校准模式选择位
0: 正常操作
1: 失调校准模式

Bit 6 **SDA1RSP**: SD OPA1 输入失调电压校准参考选择位
0: 输入参考电压来自 A1NI
1: 输入参考电压来自 A1PI

Bit 5~0 **SDA1OF5~SDA1OF0:** SD OPA1 输入失调电压校准控制位
这 6 位用于执行运算放大器的输入失调校准操作，并重新储存 SD OPA1 的输入失调校准值。更多详细资料请参考“运算放大器输入失调校准”章节。

运算放大器操作

单片机内部集成两个运算放大器，OPA0 和 OPA1，可根据用户的特定需求对信号进行放大。它们的使能或除能通过软件设置内部寄存器来完成。通过控制特殊寄存器，OPA 相关的应用可以很容易的实现，例如单位增益缓冲器、同相放大器、反相放大器和各种各样的滤波器等。

运算放大器输入失调校准

应注意，由于 SD 运算放大器输入引脚与 I/O 引脚共用，在输入失调校准前应先将引脚配置为 SD 运算放大器输入引脚。

步骤 1: 设置 SDA_nOFM=1 和 SDA_nRSP=1，使 SD 运算放大器 n 工作于失调校准模式，S0 和 S2 都 On。为了确保校准后的 V_{AnOS} 尽可能小，校准模式下的输入参考电压应该跟正常模式下的输入直流工作电压相同。

步骤 2: 设置 SDA_nOF[5:0]=000000，读取 SDA_nO 位。

步骤 3: 使 SDA_nOF[5:0]=SDA_nOF[5:0]+1，读取 SDA_nO 位。

如果 SDA_nO 位状态不变，重复步骤 3 直到 SDA_nO 位状态改变。

如果 SDA_nO 位状态改变，记录此时的 SDA_nOF[5:0] 值为 V_{AnOS1} 然后转到步骤 4。

步骤 4: 设置 SDA_nOF[5:0]=111111，读取 SDA_nO 位。

步骤 5: 使 SDA_nOF[5:0]=SDA_nOF[5:0]-1，读取 SDA_nO 位。

如果 SDA_nO 位状态不变，重复步骤 5 直到 SDA_nO 位状态改变。

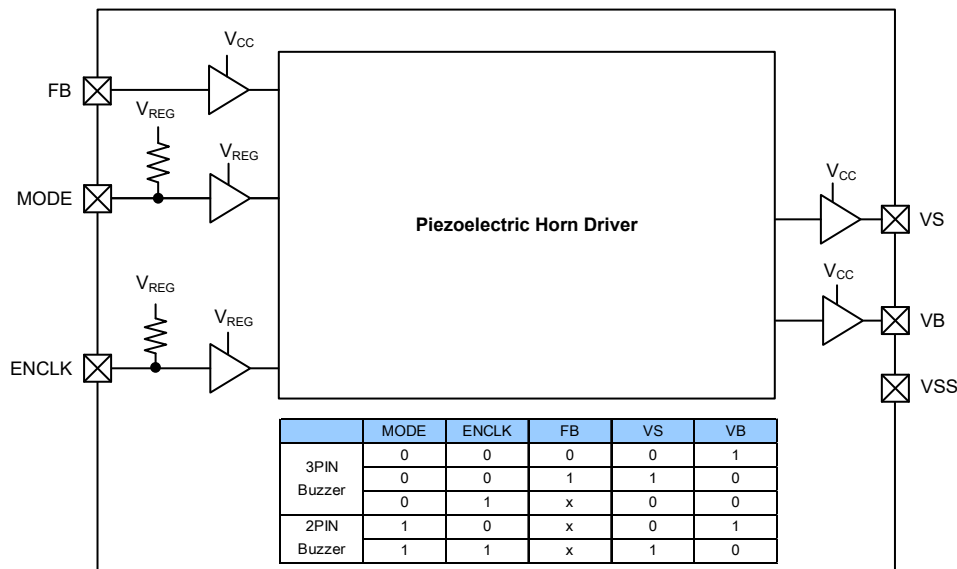
如果 SDA_nO 位状态改变，记录此时的 SDA_nOF[5:0] 值为 V_{AnOS2} 然后转到步骤 6。

步骤 6: 将 SD 运算放大器 n 输入失调校准值 V_{AnOS} 存入 SDA_nOF[5:0] 位中，校准结束。

其中 $V_{AnOS} = (V_{AnOS1} + V_{AnOS2}) / 2$ 。如果 $(V_{AnOS1} + V_{AnOS2}) / 2$ 不是整数，舍弃小数。

压电蜂鸣片驱动器

内置的压电蜂鸣片驱动器可用于自激和他激式蜂鸣器驱动。VS 和 VB 为驱动器互补式输出。当输出为逻辑高时，输出的电压大小为 V_{CC} 。通过控制 MODE 和 ENCLK 输入电平，可使能 / 除能该驱动器或选择模式。蜂鸣器关闭时，建议设置 MODE 和 ENCLK 输入为逻辑 1，以达到省电效果。



自激模式

自激式蜂鸣器连接该驱动器 FB、VS 和 VB 引脚，另外需接外部元件（电阻和电容），具体参考应用电路章节。

- MODE 和 ENCLK 输入为 0，此时为自激模式。
- 通过 FB 输入控制 VS 和 VB 的输出状态。
- ENCLK 为 1 时，VS 和 VB 输出为 0，可避免自激式所需的外围元件（电阻和电容）耗电。
- 在自激模式中，PAS1 寄存器中的 PAS1[5:4] 字段建议设置为“00”选择 PA6 功能。

他激模式

他激式蜂鸣器连接该驱动器 VS 和 VB 引脚。

- MODE 输入为 1，此时为他激模式。
- 通过 ENCLK 输入控制 VS 和 VB 的输出状态。
- 在他激模式中，PAS1 寄存器中的 PAS1[5:4] 字段建议设置为“01”选择 PTP 功能。

A/D 转换器

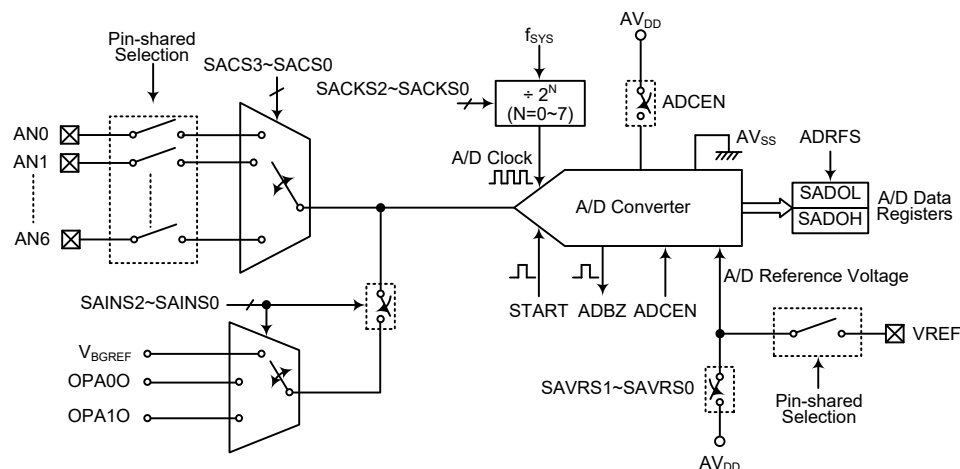
对于大多数电子系统而言，处理现实世界的模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先需要通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

A/D 简介

此单片机包含一个多通道 12-bit A/D 转换器，它可以直接接入外部模拟信号（来自传感器或其它控制信号）或内部模拟信号（高性能 Bandgap 参考电压 V_{BGRF} 、SD 运算放大器 0 输出信号 OPA0O、SD 运算放大器 1 输出信号 OPA1O 并直接将这些信号转换成 12-bit 的数字量。选择转换外部或内部模拟信号由 SAINS2~SAINS0 位和 SACS3~SACS0 位共同控制。关于 A/D 输入信号的详细描述请参考“A/D 转换器控制寄存器”和“A/D 转换器输入信号”两节内容。

外部输入通道	内部信号	通道选择位
7: AN0~AN6	3: V_{BGRF} , OPA0O, OPA1O	SAINS2~SAINS0, SACS3~SACS0

图显示了 A/D 转换器整体的内部结构和相关的寄存器。



A/D 转换器结构

A/D 转换寄存器介绍

A/D 转换器的所有工作由一系列寄存器控制。一对只读寄存器来存放 12-bit A/D 转换数据的值。剩下两个控制寄存器设置 A/D 转换器的操作和控制功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

A/D 转换寄存器列表

A/D 转换器数据寄存器 – SADOH, SADOH

对于具有 12 位 A/D 转换器的芯片，需要两个数据寄存器存放转换结果，一个高字节寄存器 SADOH 和一个低字节寄存器 SADOL。在 A/D 转换完毕后，单片机可以直接读取这些寄存器以获得转换结果。由于寄存器只使用了 16 位中的 12 位，其数据存储格式由 SADC0 寄存器的 ADRFS 位控制，如下表所示。D0~D11 是 A/D 转换数据结果位。未使用的位读为“0”。应注意，当 A/D 转换器除能时，数据寄存器的内容不变。

ADRFS	SADOH								SADOL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D 数据寄存器

A/D 转换器控制寄存器 – SADC0, SADC1

寄存器 SADC0 和 SADC1 用来控制 A/D 转换器的功能和操作。这些 8-bit 的寄存器定义包括选择连接至内部 A/D 转换器的模拟通道，数字化数据格式，A/D 时钟源，并控制和监视 A/D 转换器的忙碌状态。由于单片机只包含一个实际的模数转换电路，因此这些外部和内部模拟信号中的每一个都需要分别被发送到转换器。SADC0 寄存器中的 SACS3~SACS0 位用于选择哪个外部模拟输入通道被连接到内部 A/D 转换器。SADC1 寄存器中的 SAINS2~SAINS0 位用于选择外部模拟输入通道或内部模拟信号被连接到内部 A/D 转换器。

引脚共用功能选择寄存器的相关位用来定义 I/O 端口中的哪些引脚为 A/D 转换器的模拟输入，哪些引脚不作为 A/D 转换输入。当引脚作为 A/D 输入时，其原来的 I/O 或其它引脚共用功能消失，此外，其内部上拉电阻也将自动断开。

• SADC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7 START:** 启动 A/D 转换位
0 → 1 → 0: 启动
此位用于启动 A/D 转换过程。通常此位为低，但如果设为高再被清零，将初始化 A/D 转换过程。
- Bit 6 ADBZ:** A/D 转换忙碌标志位
0: A/D 转换结束或未开始转换
1: A/D 转换中
此位只读标志位用于表明 A/D 转换过程是否完成。当 START 位由低变为高再变为低时，ADBZ 位为高，表明 A/D 转换已初始化。A/D 转换结束后，此位被清零。
- Bit 5 ADCEN:** A/D 转换器使能 / 除能控制位
0: 除能
1: 使能
此位控制 A/D 内部功能。该位被置高将使能 A/D 转换器。如果该位清零将关闭 A/D 转换器以降低功耗。当 A/D 转换器除能时，A/D 数据寄存器 SADOH 和 SADOL 的内容将保持不变。

- Bit 4 **ADRF5**: A/D 转换数据格式选择位
 0: A/D 转换数据格式 → SADOH=D[11:4]; SADOL=D[3:0]
 1: A/D 转换数据格式 → SADOH=D[11:8]; SADOL=D[7:0]
 此位控制存放在两个 A/D 数据寄存器中的 12 位 A/D 转换结果的格式。细节方面请参考 A/D 数据寄存器章节。
- Bit 3~0 **SACS3~SACS0**: A/D 外部模拟通道输入选择位
 0000: AN0
 0001: AN1
 0010: AN2
 0011: AN3
 0100: AN4
 0101: AN5
 0110: AN6
 0111: 保留, 不可用
 1000~1111: 未定义, 输入浮空

• SADC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
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 7~5 **SAINS2~SAINS0**: A/D 输入信号选择位
 000: 外部输入 – 外部模拟通道输入 ANn
 001: 内部输入 – 内部高性能 Bandgap 参考电压 V_{BGREF}
 010: 内部输入 – 内部 SD 运算放大器 0 输出信号 OPA0O
 011: 内部输入 – 内部 SD 运算放大器 1 输出信号 OPA1O
 100: 保留, 不可用
 101~111: 外部输入 – 外部模拟通道输入 ANn
 当 SAINS2~SAINS0 被设为“001”~“011”选择转换内部模拟信号时, 需特别注意。当选择内部模拟信号时, 应正确设置 SACS3~SACS0 位, 避免外部通道输入作为 A/D 输入信号。否则, 外部输入通道会和内部模拟信号一起连接至内部 A/D 转换器, 这将导致无法预期的损害。
- Bit 4~3 **SAVRS1~SAVRS0**: A/D 转换器参考电压选择位
 00: 外部 VREF 引脚
 01: 内部 A/D 转换器电源 AV_{DD}
 1x: 外部 VREF 引脚
 这两位用于选择 A/D 转换器参考电压。当 SAVRS1~SAVRS0 为被设为“01”选择内部参考电压时需特别注意。当选择 A/D 转换器电源作为 A/D 转换器参考电压时, 外部 VREF 引脚需通过引脚共用控制位选择作为其它共用的引脚功能。否则, 外部 VREF 输入电压会和内部参考电压一起连接至内部 A/D 转换器, 可能会导致内部电路的损坏。
- Bit 2~0 **SACKS2~SACKS0**: A/D 时钟源选择位
 000: f_{SYS}
 001: $f_{SYS}/2$
 010: $f_{SYS}/4$
 011: $f_{SYS}/8$
 100: $f_{SYS}/16$
 101: $f_{SYS}/32$
 110: $f_{SYS}/64$
 111: $f_{SYS}/128$
 这三位用于选择 A/D 转换器的时钟源。

A/D 转换器操作

SADC0 寄存器中的 START 位，用于开启 A/D 转换器。当单片机设置此位从逻辑低到逻辑高，然后再到逻辑低，就会开始一个模数转换周期。

SADC0 寄存器中的 ADBZ 位用于表明模数转换过程是否正在进行。A/D 转换成功启动后，ADBZ 位会被单片机自动置为“1”。在转换周期结束后，ADBZ 位会自动置为“0”。此外，也会置位中断控制寄存器内相应的 A/D 中断请求标志位，如果中断使能，就会产生对应的内部中断信号。A/D 内部中断信号将引导程序跳转到相应的 A/D 内部中断地址。如果 A/D 内部中断除能，可以让单片机轮询 SADC0 寄存器中的 ADBZ 位，检查此位是否被清除，作为另一种侦测 A/D 转换周期结束的方法。

A/D 转换器的时钟源为系统时钟 f_{SYS} 或其分频，而分频系数由 SADC1 寄存器中的 SACKS2~SACKS0 位决定。虽然 A/D 时钟源是由系统时钟 f_{SYS} 和 SACKS2~SACKS0 位决定，但可选择的最大 A/D 时钟源则有一些限制。由于允许的 A/D 时钟周期 t_{ADCK} 的范围为 $0.5\mu s \sim 10\mu s$ ，所以选择系统时钟速度时就必须小心。例如，如果系统时钟速度为 8MHz 时，SACKS2~SACKS0 位不能设为“000”、“001”或“111”。必须保证设置的 A/D 转换时钟周期不小于时钟周期的最小值或不大于时钟周期的最大值。使用者可以参考下面的表格，被标上星号 * 的数值是不允许的，因为它们超出了 A/D 转换时钟周期规定的范围。

f_{SYS}	A/D 时钟周期 (t_{ADCK})							
	SACKS [2:0]=000 (f_{SYS})	SACKS [2:0]=001 ($f_{SYS}/2$)	SACKS [2:0]=010 ($f_{SYS}/4$)	SACKS [2:0]=011 ($f_{SYS}/8$)	SACKS [2:0]=100 ($f_{SYS}/16$)	SACKS [2:0]=101 ($f_{SYS}/32$)	SACKS [2:0]=110 ($f_{SYS}/64$)	SACKS [2:0]=111 ($f_{SYS}/128$)
1MHz	1 μs	2 μs	4 μs	8 μs	16 μs *	32 μs *	64 μs *	128 μs *
2MHz	500ns	1 μs	2 μs	4 μs	8 μs	16 μs *	32 μs *	64 μs *
4MHz	250ns *	500ns	1 μs	2 μs	4 μs	8 μs	16 μs *	32 μs *
8MHz	125ns *	250ns *	500ns	1 μs	2 μs	4 μs	8 μs	16 μs *

A/D 时钟周期范例

SADC0 寄存器中的 ADCEN 位用于控制 A/D 转换电路电源的开启和关闭。该位必须置高以开启 A/D 转换器电源。当设置 ADCEN 位为高开启 A/D 转换器内部电路时，在 A/D 转换成功开启前需一段延时，如时序图中所示。即使通过相关引脚共用控制位选择无引脚作为 A/D 输入，如果 ADCEN 设为“1”，那么仍然会产生功耗。因此在功耗敏感的应用中，当未使用 A/D 转换器功能时，建议清零 ADCEN 位以减少功耗。

A/D 转换器参考电压

A/D 转换器参考电压来自正电源电压 AV_{DD} 或外部参考源引脚 VREF，通过 SAVRS1 和 SAVRS0 位选择。当 SAVRS1~SAVRS0 位为“01”时，A/D 转换器参考电压来自 AV_{DD} 。当 SAVRS1~SAVRS0 位为“01”以外的任意值时，A/D 转换器参考电压来自 VREF 引脚。由于 VREF 引脚与其它功能共用，当选择 VREF 引脚作为参考电压源时，需先正确设置引脚共用选择位将 VREF 引脚配置为参考电压输入功能。然而，当内部 A/D 转换器电源被选作参考电压源时，相关的引脚共用控制位不可选择 VREF 参考电压输入功能，避免 VREF 引脚电压跟内部参考电压 AV_{DD} 一起接入 A/D 转换器。模拟输入值一定不能超过所选的参考电压值。

A/D 转换器输入信号

所有的 A/D 模拟输入引脚都与 I/O 口及其它功能共用。使用 PxS0 和 PxS1 寄存器中的相应位，可以将它们设置为 A/D 转换器模拟输入脚或具有其它功能。如果对应的引脚作为 A/D 转换输入，那么它原来的引脚功能将除能。通过这种方式，引脚的功能可由程序来控制，灵活地切换引脚功能。如果将引脚设为 A/D 输入，则通过寄存器编程设置的所有上拉电阻会自动断开。请注意，端口控制寄存器不需要为使能 A/D 输入而先设定为输入模式，当 A/D 输入功能选择位使能 A/D 输入时，端口控制寄存器的状态将被重置。

另外还有几个内部模拟信号可作为 A/D 转换器的模拟输入信号，分别来自高性能 Bandgap 参考电压 V_{BGREF} 、SD 运算放大器 0 输出信号 OPA0O、SD 运算放大器 1 输出信号 OPA1O，通过设置 SAINS2~SAINS0 位来选择。若 SAINS2~SAINS0 位为“000”或“101~111”，则选择转换外部模拟输入信号，具体通道编号由 SACS3~SACS0 位决定。若选择内部模拟信号时，应将 SACS3~SACS0 位设置为一个适当的值，以关闭外部模拟通道输入。否则，外部通道输入会与内部模拟信号一起接入从而导致错误。

V_{BGREF} 是具有驱动能力的高性能 Bandgap 的参考电压。当输入电源电压 AV_{DD} 改变或温度变化时，Bandgap 会输出精准的参考电压。而且该 Bandgap 在低压下便可启动。因此，该参考电压对于存在低压降线性稳压器 LDO 的应用具有较高的电源电压抑制比 PSRR。

SAINS[2:0]	SACS[3:0]	输入信号	说明
000, 101~111	0000~0110	AN0~AN6	外部模拟通道输入
	1000~1111	—	未选择外部通道，输入浮空
001	1000~1111	V_{BGREF}	内部高性能 Bandgap 参考电压
010	1000~1111	OPA0O	内部 SD 运算放大器 0 输出信号
011	1000~1111	OPA1O	内部 SD 运算放大器 1 输出信号

A/D 转换器输入信号选择

• VBGRC 寄存器

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

Bit 7~1 未定义，读为“0”

Bit 0 **VBGREN**: Bandgap 使能 / 除能控制位

0: 除能

1: 使能

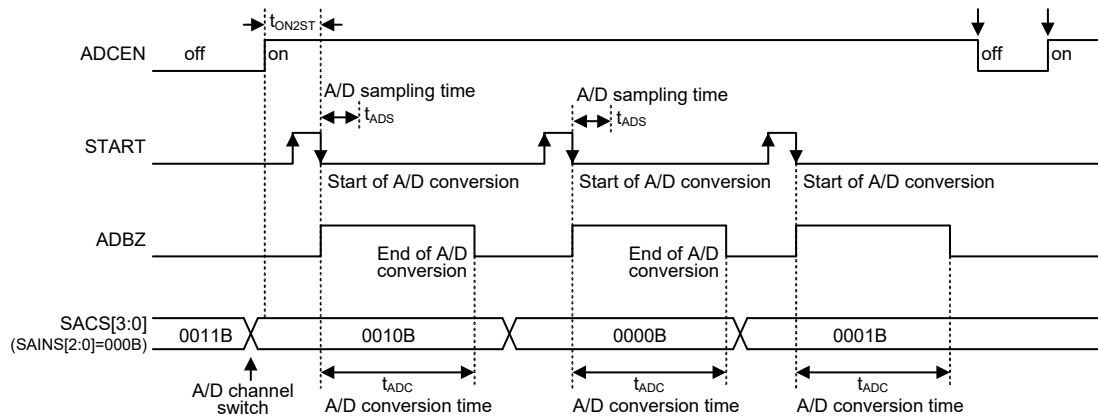
当 VBGREN 位被清零时，Bandgap 输出处于高阻抗状态。

A/D 转换率及时序图

一个完整的 A/D 转换包含两部分，数据采样和数据转换。数据采样时间定义为 t_{ADS} ，需要 4 个 A/D 时钟周期，而数据转换需要 12 个 A/D 时钟周期。所以一个完整的 A/D 转换时间， t_{ADC} ，一共需要 16 个 A/D 时钟周期。

最大 A/D 转换率 = A/D 时钟周期 ÷ 16

下列时序图表示模数转换过程中不同阶段的图形与时序。由应用程序控制开始 A/D 转换过程后，单片机的内部硬件就会开始进行转换，在这个过程中，程序可以继续其它功能。A/D 转换时间为 $16t_{ADCK}$ ， t_{ADCK} 为 A/D 时钟周期。



A/D 转换时序图 – 外部通道输入

A/D 转换步骤概述

下面概述实现 A/D 转换过程的各个步骤。

- 步骤 1
通过 SADC1 寄存器中的 SACKS2~SACKS0 位，选择所需的 A/D 转换时钟。
- 步骤 2
将 SADC0 寄存器中的 ADCEN 位置高使能 A/D 转换器。
- 步骤 3
通过配置 SAINS2~SAINS0 位，选择连接至内部 A/D 转换器的信号。
若选择外部通道输入，接着执行步骤 4。
若选择内部模拟信号，接着执行步骤 5。
- 步骤 4
若已通过 SAINS2~SAINS0 位选择 A/D 输入信号来自外部通道输入，接着应设置相关的引脚共用控制位将该引脚规划为 A/D 输入引脚。通过设置 SACS3~SACS0 位选择哪个外部通道接至 A/D 转换器。接着执行步骤 6。
- 步骤 5
选择内部模拟信号前，应正确设置 SACS3~SACS0 位，将外部通道输入切换到无通道输入。然后再设置 SAINS2~SAINS0 位选择所需的内部模拟信号。接着执行步骤 6。
- 步骤 6
通过 SADC1 寄存器中的 SAVRS1~SAVRS0 位选择参考电压。
- 步骤 7
设置 SADC0 寄存器中的 ADRFS 位选择 A/D 转换器输出数据格式。
- 步骤 8
如果要使用中断，则中断控制寄存器需要正确地设置，以确保 A/D 中断功能是激活的。总中断控制位 EMI 需要置位为“1”，以及 A/D 转换器中断位 ADE 也需要置位为“1”。
- 步骤 9
现在可以通过设置 SADC0 寄存器中的 START 位从“0”到“1”再回到“0”，开始模数转换的过程。

● 步骤 10

如果 A/D 转换正在进行中，ADBZ 位会被置为逻辑高。A/D 转换完成后，ADBZ 位会被置为逻辑低，并可从 SADOH 和 SADOL 寄存器中读取输出数据。

注：若使用轮询 SADC0 寄存器中 ADBZ 位的状态的方法来检查转换过程是否结束时，则中断使能的步骤可以省略。

编程注意事项

在编程时，如果 A/D 转换器未使用，通过设置 SADC0 寄存器中的 ADCEN 为低，关闭 A/D 内部电路以减少电源功耗。此时，不考虑输入脚的模拟电压，内部 A/D 转换器电路不产生功耗。如果 A/D 转换器输入脚用作普通 I/O 脚，必须特别注意，输入电压为无效逻辑电平也可能增加功耗。

A/D 转换功能

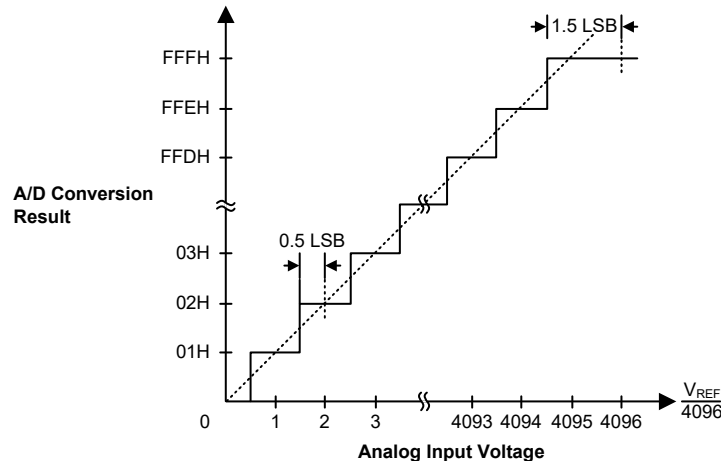
单片机含有一组 12 位的 A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于实际 A/D 转换器参考电压值 V_{REF} ，因此每一位可表示 $V_{REF}/4096$ 的模拟输入值。

$$1 \text{ LSB} = V_{REF} \div 4096$$

通过下面的等式可估算 A/D 转换器输入电压值：

$$\text{A/D 输入电压} = \text{A/D 数字输出值} \times (V_{REF} \div 4096)$$

下图显示 A/D 转换器模拟输入值和数字输出值之间理想的转换功能。除了数字化数值 0，其后的数字化数值会在精确点之前的 0.5 LSB 处改变，而数字化数值的最大值将在 V_{REF} 之前的 1.5 LSB 处改变。注意，这里的 V_{REF} 电压指的是通过 SAVRS[1:0] 位选择的实际输入 A/D 转换器的参考电压。



理想的 A/D 转换功能

A/D 转换应用范例

下面两个范例程序用来说明怎样使用 A/D 转换。第一个范例是轮询 SADC0 寄存器中的 ADBZ 位来判断 A/D 转换是否完成；第二个范例则使用中断的方式判断。

范例：使用查询 ADBZ 的方式来检测转换结束

```
clr ADE                ; disable ADC interrupt
mov a,03H
mov SADC1,a            ; select fsys/8 as A/D clock
set ADCEN
```

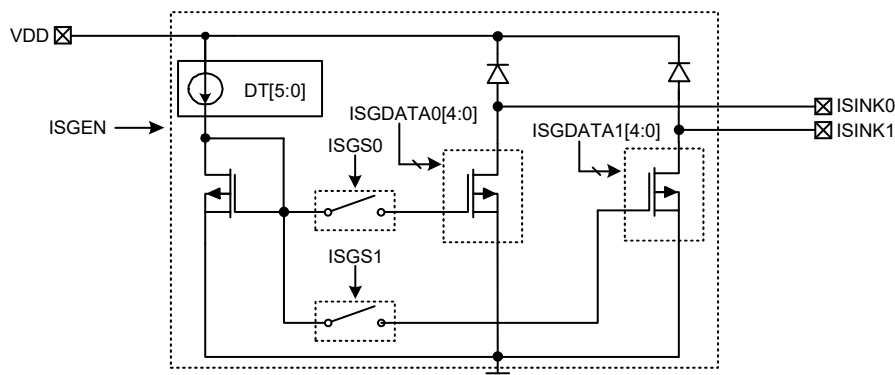
```
mov a,02h                ; setup PAS1 register to configure pin AN0
mov PAS1,a
mov a,20h
mov SADC0,a              ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr START                ; high pulse on start bit to initiate conversion
set START                ; reset A/D
clr START                ; start A/D
polling_EOC:
sz ADBZ                  ; poll the SADC0 register ADBZ bit to detect end
                        ; of A/D conversion
jmp polling_EOC          ; continue polling
mov a,SADOL               ; read low byte conversion result value
mov SADOL_buffer,a       ; save result to user defined register
mov a,SADOH               ; read high byte conversion result value
mov SADOH_buffer,a       ; save result to user defined register
:
:
jmp start_conversion      ; start next A/D conversion
```

范例：使用中断的方式来检测转换结束

```
clr ADE                  ; disable ADC interrupt
mov a,03H
mov SADC1,a              ; select fsys/8 as A/D clock
set ADCEN
mov a, 02h               ; setup PAS1 register to configure pin AN0
mov PAS1,a
mov a,20h
mov SADC0,a              ; enable and connect AN0 channel to A/D converter
Start_conversion:
clr START                ; high pulse on START bit to initiate conversion
set START                ; reset A/D
clr START                ; start A/D
clr ADF                  ; clear ADC interrupt request flag
set ADE                  ; enable ADC interrupt
set EMI                  ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov acc_stack,a          ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a       ; save STATUS to user defined memory
:
:
mov a,SADOL               ; read low byte conversion result value
mov SADOL_buffer,a       ; save result to user defined register
mov a,SADOH               ; read high byte conversion result value
mov SADOH_buffer,a       ; save result to user defined register
:
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a             ; restore STATUS from user defined memory
mov a,acc_stack          ; restore ACC from user defined memory
reti
```

灌电流发生器

无论 V_{ISINK} 电压为 1.0V~3.3V 内何值，灌电流发生器都可以提供恒定的电流。恒定电流值通过 ISGDATA0/ISGDATA1 寄存器控制，灌电流范围为 50mA~360mA。



灌电流发生器寄存器

灌电流发生器的所有工作由一系列寄存器控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
ISGENC	ISGEN	—	—	—	—	—	ISGS1	ISGS0
ISGDATA0	—	—	—	D4	D3	D2	D1	D0
ISGDATA1	—	—	—	D4	D3	D2	D1	D0

灌电流发生器寄存器列表

• ISGENC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ISGEN	—	—	—	—	—	ISGS1	ISGS0
R/W	R/W	—	—	—	—	—	R/W	R/W
POR	0	—	—	—	—	—	0	0

Bit 7 **ISGEN**: 灌电流发生器使能控制

0: 除能
1: 使能

当 ISGEN 位为 0 时将除能灌电流发生器，此时 ISINK0 和 ISINK1 引脚状态为：
 $V_{ISINK0} \& V_{ISINK1} = \text{浮空}$ ， $I_{ISINK0} \& I_{ISINK1} = 0$ 。

Bit 6~2 未定义，读为“0”

Bit 1 **ISGS1**: ISINK1 引脚灌电流使能 / 除能控制

0: 除能
1: 使能

Bit 0 **ISGS0**: ISINK0 引脚灌电流使能 / 除能控制

0: 除能
1: 使能

● ISGDATA0 寄存器

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

Bit 7~5 未定义，读为“0”

Bit 4~0 **D4~D0**: ISINK0 引脚灌电流发生器控制
灌电流值 (mA)=50+10×(ISGDATA0[4:0])
更多内容请参考“灌电流发生器电气特性”。

● ISGDATA1 寄存器

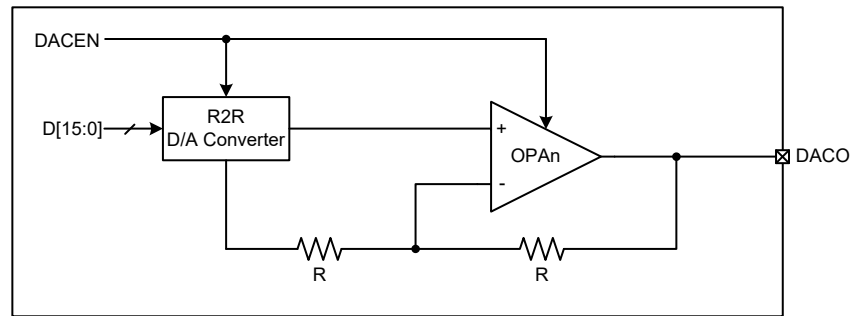
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	D4	D3	D2	D1	D0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 未定义，读为“0”

Bit 4~0 **D4~D0**: ISINK1 引脚灌电流发生器控制
灌电流值 (mA)=50+5×(ISGDATA1[4:0])
更多内容请参考“灌电流发生器电气特性”。

16-bit 语音 D/A 转换器

此单片机内置一个 10-bit D/A 转换器，该电路是用于音频应用的 16-bit R2R D/A 转换器。16-bit 语音 D/A 转换器的参考电压源仅来自模拟电源并能够降压以降低功耗。16-bit 语音 D/A 转换器适用于语音或音频应用。虽然该 D/A 转换器不是将通常的数据一对一地转换为模拟信号，但它提供了良好及等同的音频质量，无论声音音量的大小。应注意，16-bit 语音 D/A 转换器的电压通过 OPAn 放大并缓存后输出。



16-bit 语音 D/A 转换器结构

16-bit 语音 D/A 转换器寄存器

16-bit 语音 D/A 转换器所有的操作由三个寄存器控制。一个 16-bit 语音 D/A 转换器数据高字节寄存器 DAH，一个 16-bit 语音 D/A 转换器数据低字节寄存器 DAL 和一个用于控制 16-bit 语音 D/A 转换器使能或除能的控制寄存器 DACC。

寄存器名称	位							
	7	6	5	4	3	2	1	0
DAH	D15	D14	D13	D12	D11	D10	D9	D8
DAL	D7	D6	D5	D4	D3	D2	D1	D0
DACC	—	—	—	—	—	—	—	DACEN

16-bit 语音 D/A 转换器寄存器列表

• DAH 寄存器

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8:** 16-bit 语音 D/A 转换器数据高字节
 如果需要更新 16-bit 语音 D/A 转换器数据应先更新 16-bit 语音 D/A 转换器数据低字节寄存器 DAL 再进行 DAH 寄存器的更新。每次写入 DAH 寄存器时，16-bit 的数据都将加载到 D/A 转换器中并启动转换循环。应注意，在更新 D/A 转换器数据之前需先使能 D/A 转换器。

• DAL 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~D:** 16-bit 语音 D/A 转换器数据低字节
 写数据至该寄存器时，数据仅写入影子寄存器；写数据至 DAH 寄存器时，数据直接写入高字节寄存器，同时锁存在影子寄存器中的数据写入到 DAL 寄存器。应注意，在更新 D/A 转换器数据之前需先使能 D/A 转换器。

• DACC 寄存器

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

Bit 7~1 未定义，读为“0”
 Bit 0 **DACEN:** 16-bit 语音 D/A 转换器使能 / 除能控制位
 0: 除能
 1: 使能
 16-bit 语音 D/A 转换器使能后，电路稳定需要一定的延时 t_{DACS} ，需确保 16-bit 语音 D/A 转换器电路在 16-bit 数据寄存器更新前已稳定。

串行接口模块 – SIM

此单片机内有一个串行接口模块，包括两种易与外部设备通信的串行接口：四线 SPI 或两线 I²C 接口。这两种接口具有相当简单的通信协议，单片机可以通过这些接口与传感器、闪存或 EEPROM 内存等硬件设备通信。因 SIM 接口引脚是与其它 I/O 引脚共用，因此在使用 SIM 功能前，要先通过相应的引脚共用功能选择寄存器选定 SIM 引脚功能。因为这两种接口 SPI 和 I²C 共用引脚和寄存器，所以要通过一个 SIMC0 寄存器中的 SIM2~SIM0 位来选择哪一种通信接口。若 SIM 功能使能，可通过上拉电阻控制寄存器选择与输入 / 输出共用的 SIM 输入引脚的上拉电阻。

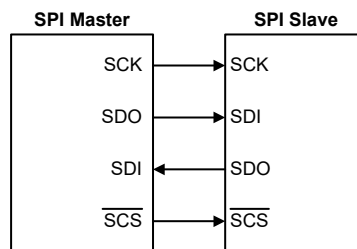
SPI 接口

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 $\overline{\text{SCS}}$ 。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 $\overline{\text{SCS}}$ 。SDI 和 SDO 是数据的输入和输出线。SCK 是串行时钟线， $\overline{\text{SCS}}$ 是从机的选择线。SPI 的接口引脚与普通 I/O 口和 I²C 的功能脚共用。通过设定 SIMC0/SIMC2 寄存器的对应位，来使能 SPI 接口。SPI 可以通过 SIMC0 寄存器中的 SIMEN 位来除能或使能。连接到 SPI 接口的单片机以主机 / 从机模式进行通信。由 SPI 通信中的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 $\overline{\text{SCS}}$ 引脚，所以只能拥有一个从机设备。可通过软件控制 $\overline{\text{SCS}}$ 引脚使能与除能，设置 CSEN 位为“1”使能 $\overline{\text{SCS}}$ 功能，设置 CSEN 位为“0”， $\overline{\text{SCS}}$ 引脚将处于浮空状态。

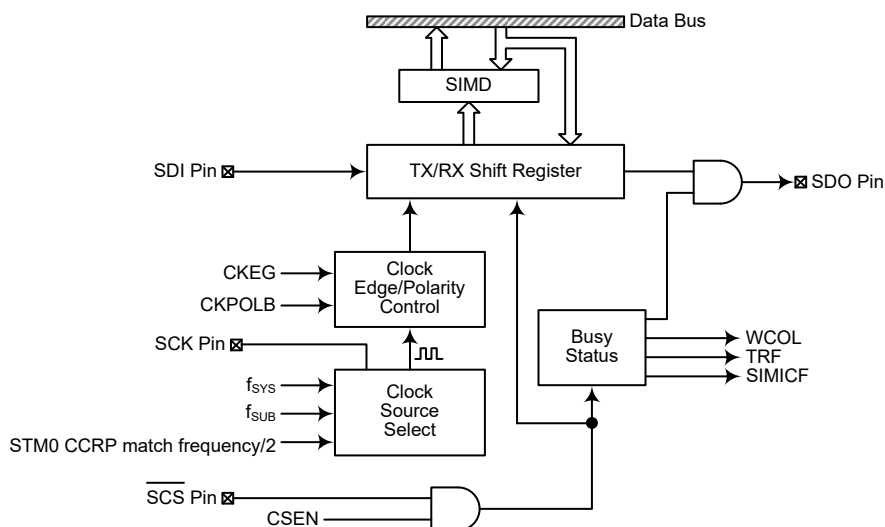


SPI 主 / 从机连接方式

该单片机的 SPI 功能具有以下特点：

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

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



SPI 方框图

SPI 寄存器

有三个内部寄存器用于控制 SPI 接口的所有操作，其中有一个数据寄存器 SIMD、两个控制寄存器 SIMC0 和 SIMC2。注意，SIMC1 寄存器仅用于 I²C 接口控制。

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

SPI 寄存器列表

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 I²C 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: SIM 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。应注意的是 SIMC2 与 I²C 接口功能中的寄存器 SIMA 是同一个寄存器。SPI 功能不会用到寄存器 SIMC1，SIMC1 寄存器仅在工作于 I²C 接口时才被使用。寄存器 SIMC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 SIMC2 用于其它的控制功能如 LSB/MSB 选择，写冲突标志位等。

• SIMC0 寄存器

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

Bit 7~5 **SIM2~SIM0: SIM 工作模式控制位**

000: SPI 主机模式; SPI 时钟为 $f_{sys}/4$
 001: SPI 主机模式; SPI 时钟为 $f_{sys}/16$
 010: SPI 主机模式; SPI 时钟为 $f_{sys}/64$
 011: SPI 主机模式; SPI 时钟为 f_{sub}
 100: SPI 主机模式; SPI 时钟为 STM0 CCRP 匹配频率 / 2
 101: SPI 从机模式
 110: I²C 从机模式
 111: 未使用模式

这几位用于设置 SIM 功能的工作模式，除了选择 I²C 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 STM0。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 未定义，读为“0”

Bit 3~2 **SIMDEB1~SIMDEB0: I²C 去抖时间选择位**

这些位只有在 SIM 设置成 I²C 接口模式时才有效。请参考 I²C 寄存器部分。

Bit 1 **SIMEN: SIM 使能控制位**

0: 除能
 1: 使能

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

Bit 0 **SIMICF: SIM SPI 未完成标志位**

0: SPI 传输未完成状况未发生
 1: SPI 传输未完成状况发生

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

● SIMC2 寄存器

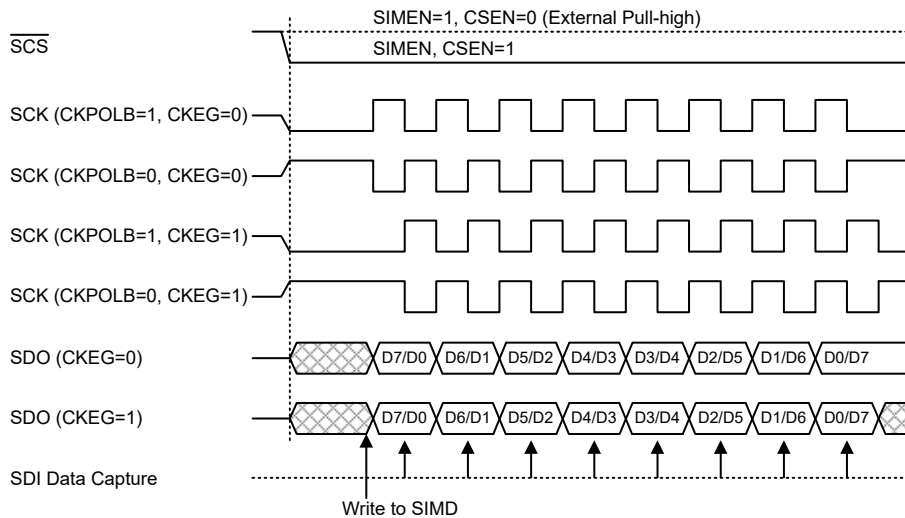
Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7~6 **D7~D6:** 未定义位
用户可通过软件程序对这两位进行读写。
- Bit 5 **CKPOLB:** SPI 时钟线的基础状态位
0: 当时钟无效时, SCK 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 数据移位顺序控制位
0: LSB 优先
1: MSB 优先
数据移位顺序选择位, 用于选择数据传输时高位优先传输还是低位优先传输。此位设置为高时高位优先传输, 为低时低位优先传输。
- Bit 2 **CSEN:** SPI $\overline{SCS}$ 引脚控制位
0: 除能
1: 使能
CSEN 位用于 $\overline{SCS}$ 引脚的使能 / 除能控制。此位为低时, $\overline{SCS}$ 除能并处于浮空状态。此位为高时, $\overline{SCS}$ 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。

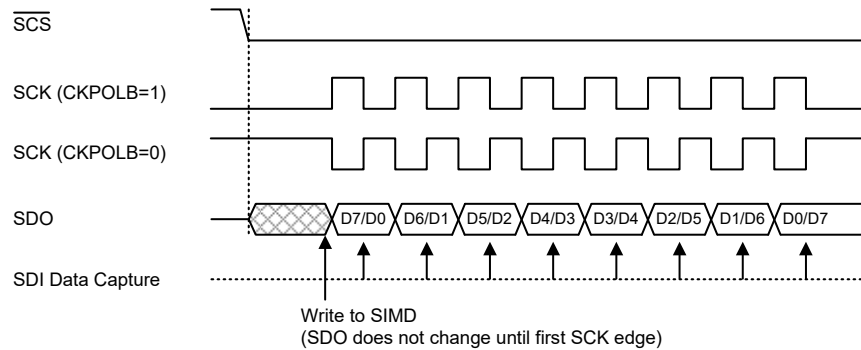
SPI 通信

将 SIMEN 设置为高，使能 SPI 功能之后，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 位将自动被置位但只能通过应用程序清零。单片机处于从机模式时，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 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 CKPOLB 和 CKEG 位各种设置情况下从机数据与 SCK 信号的关系。

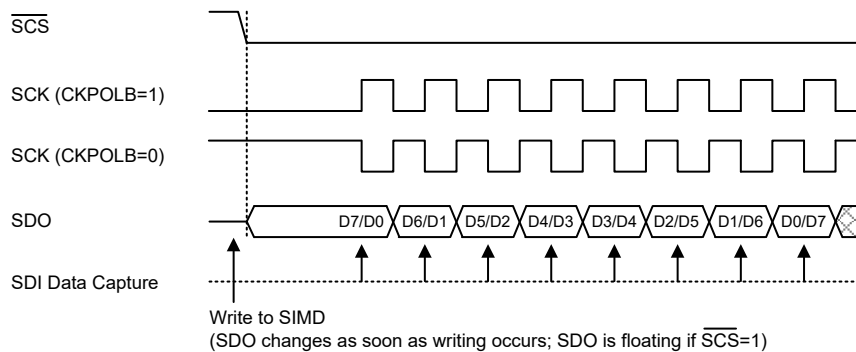
即使在单片机处于空闲模式时，若 SPI 接口使用的时钟源仍开启，SPI 功能仍将继续执行。



SPI 主机模式时序

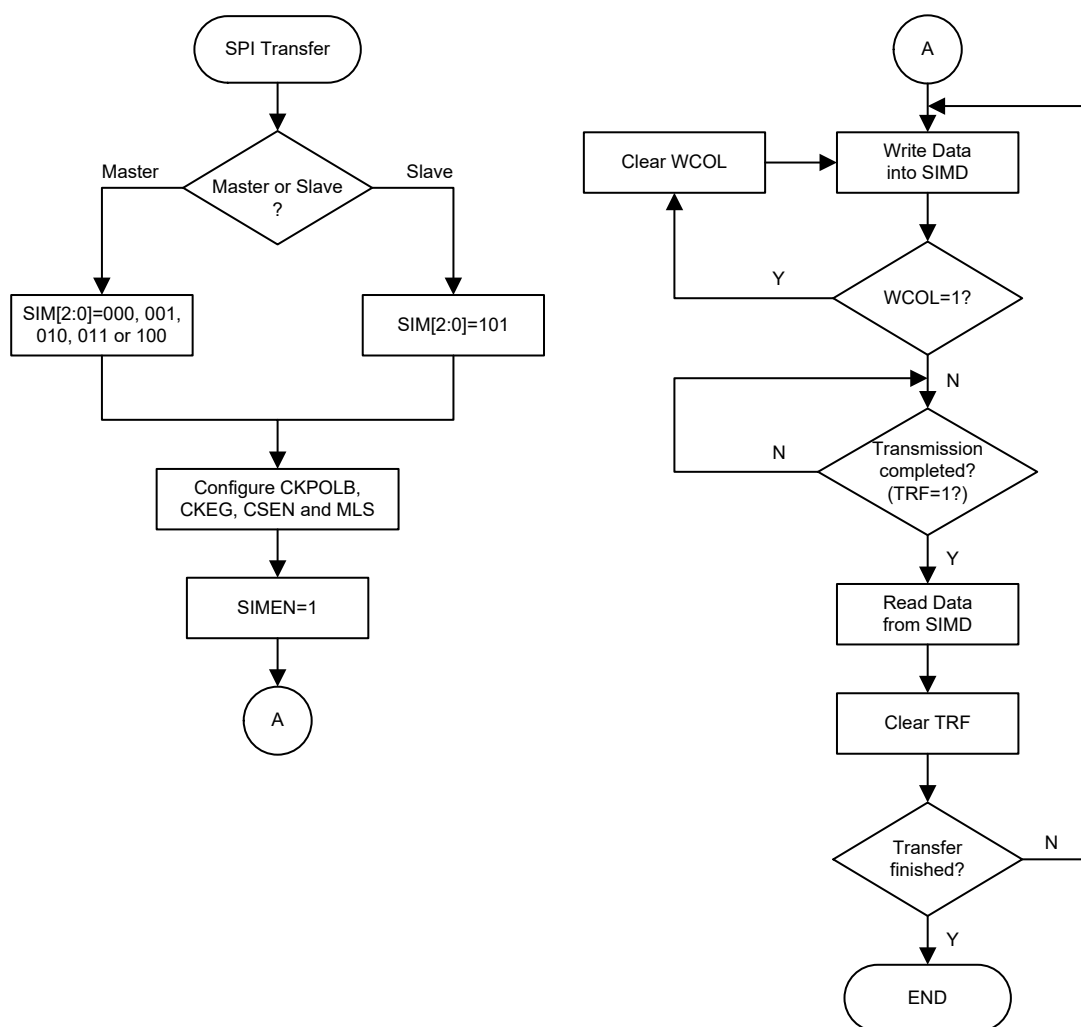


SPI 从机模式时序 - CKEG=0



Note: For SPI slave mode, if $SIMEN=1$ and $CSEN=0$, SPI is always enabled and ignores the $\overline{SCS}$ 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 $\overline{\text{CS}}\text{EN}=1$ 、 $\overline{\text{SCS}}=0$ 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，通过设置相应共用功能选择控制位，SCK、SDI、SDO、 $\overline{\text{SCS}}$ 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， $\overline{\text{SCS}}$ 信号线有效将使能 SPI 接口。设置此位为低，SPI 接口将除能， $\overline{\text{SCS}}$ 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， $\overline{\text{SCS}}$ 、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 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 $\overline{\text{SCS}}$ 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 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

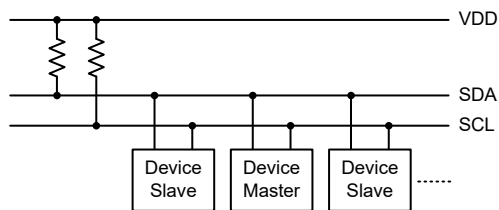
- 步骤 1
设置 SIMC0 控制寄存器中的 SIM2~SIM0 位，选择 SPI 从机模式。
- 步骤 2
设置 CSEN 和 MLS 位，选择高位或低位数据优先传送，这必须与主机设备一致。
- 步骤 3
设置 SIMC0 控制寄存器中的 SIMEN 位，使能 SPI 接口功能。
- 步骤 4
对于写操作：写数据到 SIMD 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。等待主机时钟 SCK 信号和 SCS 信号。跳至步骤 5。
对于读操作：从 SDI 信号线移入的数据将被存储在 TXRX 缓存器中，直到所有数据接收完毕，此时数据全部锁存至 SIMD 寄存器。
- 步骤 5
检测 WCOL 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 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。

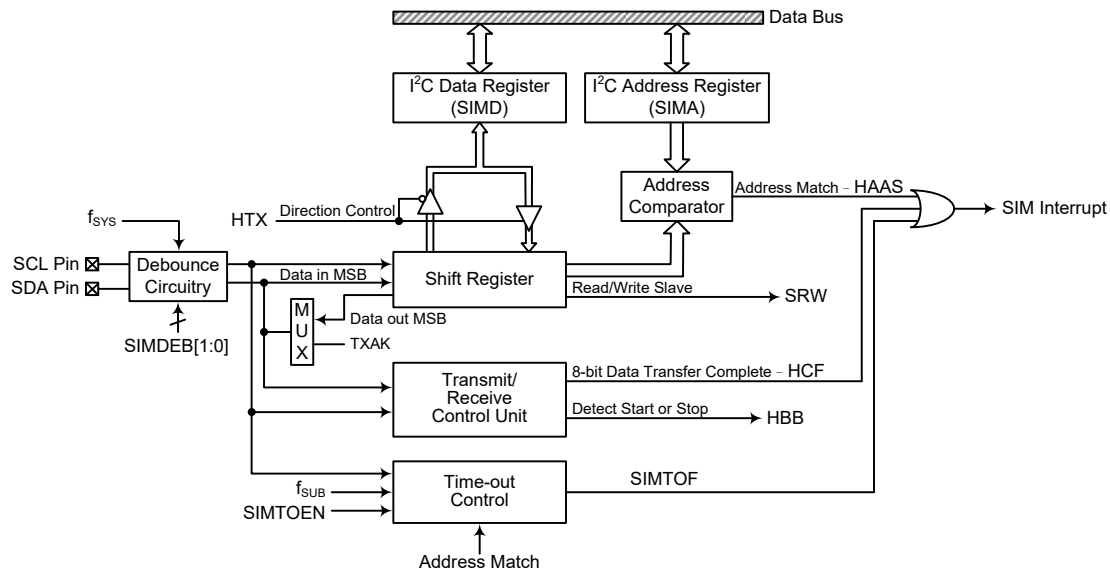


I²C 主从总线连接图

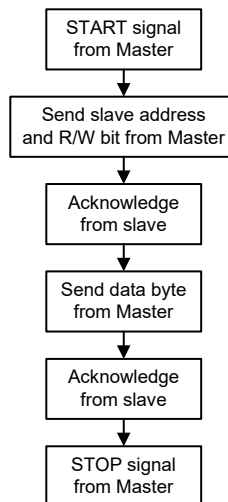
I²C 接口操作

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。因此应在这些输出上都加上拉电阻。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端口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。

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

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。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	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: SIM 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。应注意的是寄存器 SIMA 和 SPI 接口使用的寄存器 SIMC2 共用同一个寄存器地址。

● 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**: I²C 从机地址位
SIMA6~SIMA0 是 7 位从机地址 bit 6 ~ bit 0。

Bit 0 **D0**: 保留位, 此位可通过软件程序进行读或写

I²C 控制寄存器

单片机中有三个控制 I²C 接口功能的寄存器, SIMC0、SIMC1 和 SIMTOC。
寄存器 SIMC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 SIMC1 包括多个用于指示 I²C 传输状态的相关标志位。SIMTOC 寄存器用于控制 I²C 超时功能, 此寄存器在 I²C 超时章节介绍。

● SIMC0 寄存器

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

Bit 7~5 **SIM2~SIM0**: SIM 工作模式控制位
000: SPI 主机模式; SPI 时钟为 $f_{\text{SYS}}/4$
001: SPI 主机模式; SPI 时钟为 $f_{\text{SYS}}/16$
010: SPI 主机模式; SPI 时钟为 $f_{\text{SYS}}/64$
011: SPI 主机模式; SPI 时钟为 f_{SUB}
100: SPI 主机模式; SPI 时钟为 STM0 CCRP 匹配频率 / 2
101: SPI 从机模式
110: I²C 从机模式
111: 未使用模式

这几位用于设置 SIM 功能的工作模式, 用于选择 SPI 的主从模式和 SPI 的主机时钟频率及 I²C 或 SPI 功能。SPI 时钟源可来自于系统时钟和 f_{SUB} 也可以选择来自 STM0。若选择的是作为 SPI 从机, 则其时钟源从外部主机而得。

Bit 4 未定义, 读为 “0”

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

当设置 SIM2~SIM0 位为 “110” 将 SIM 设置为 I²C 接口功能时, 这两个位用于选择 I²C 去抖时间。

Bit 1 **SIMEN**: SIM 使能控制位
0: 除能
1: 使能

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

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

● SIMC1 寄存器

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

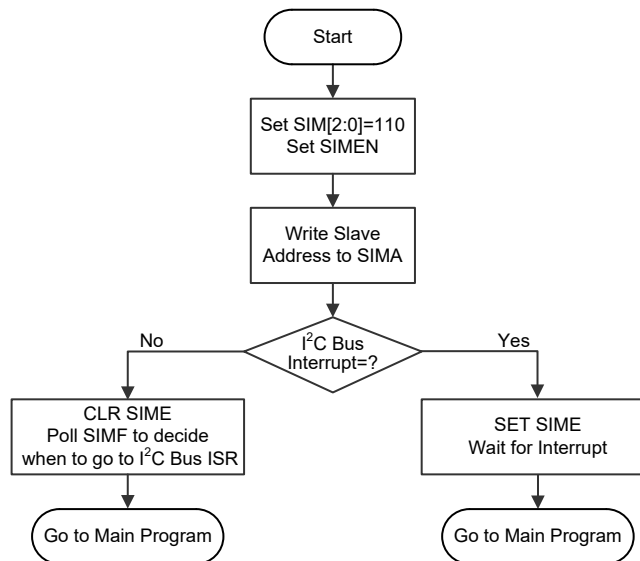
Bit 1 **IAMWU**: I²C 地址匹配唤醒控制位
0: 除能
1: 使能
此位设置为“1”则使能 I²C 地址匹配使系统从休眠或空闲模式中唤醒的功能。若进入休眠或空闲模式前 IAMWU 已经置高以使能 I²C 地址匹配唤醒功能，在系统唤醒后须软件清除此位以确保单片机正确地运行。

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收到一个应答信号。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 I²C 中断。进入中断服务程序后，系统要检测 HAAS 位和 SIMTOF 位，以判断 I²C 总线中断是来自从机地址匹配，还是来自 8 位数据传输完毕，或是来自 I²C 超时。在数据传输中，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 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
设置相关中断使能控制位使能 SIM 中断。



I²C 总线初始化流程图

I²C 总线起始信号

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

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，紧接着主机将发送从机地址以选择要进行数据传输的从机。所有在 I²C 总线上的从机

接收到 7 位地址数据后，都会将其与各自内部的地址进行比较。如果从机从主机上接收到的地址与自身内部的地址相匹配，则会产生一个 I²C 总线中断信号。地址位接下来的一位为读 / 写状态位（即第 8 位），将被保存到 SIMC1 寄存器的 SRW 位，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，从机将状态标志位 HAAS 置位。

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 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。

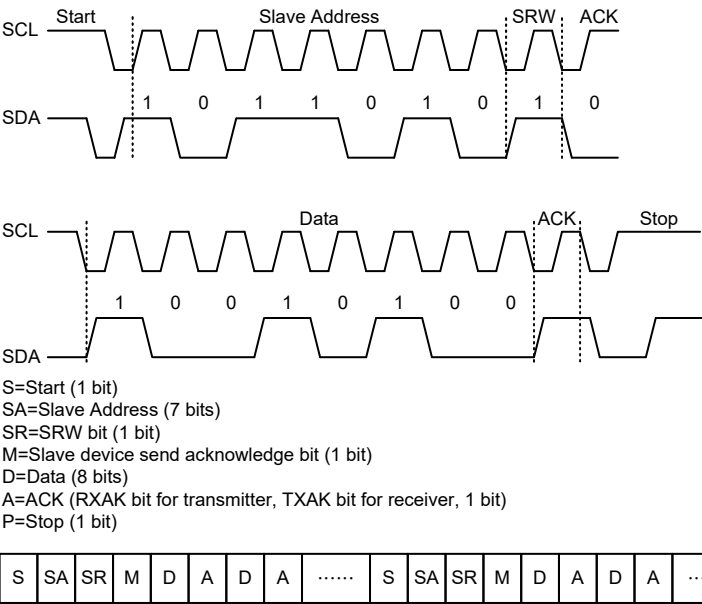
I²C 总线从机地址应答信号

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

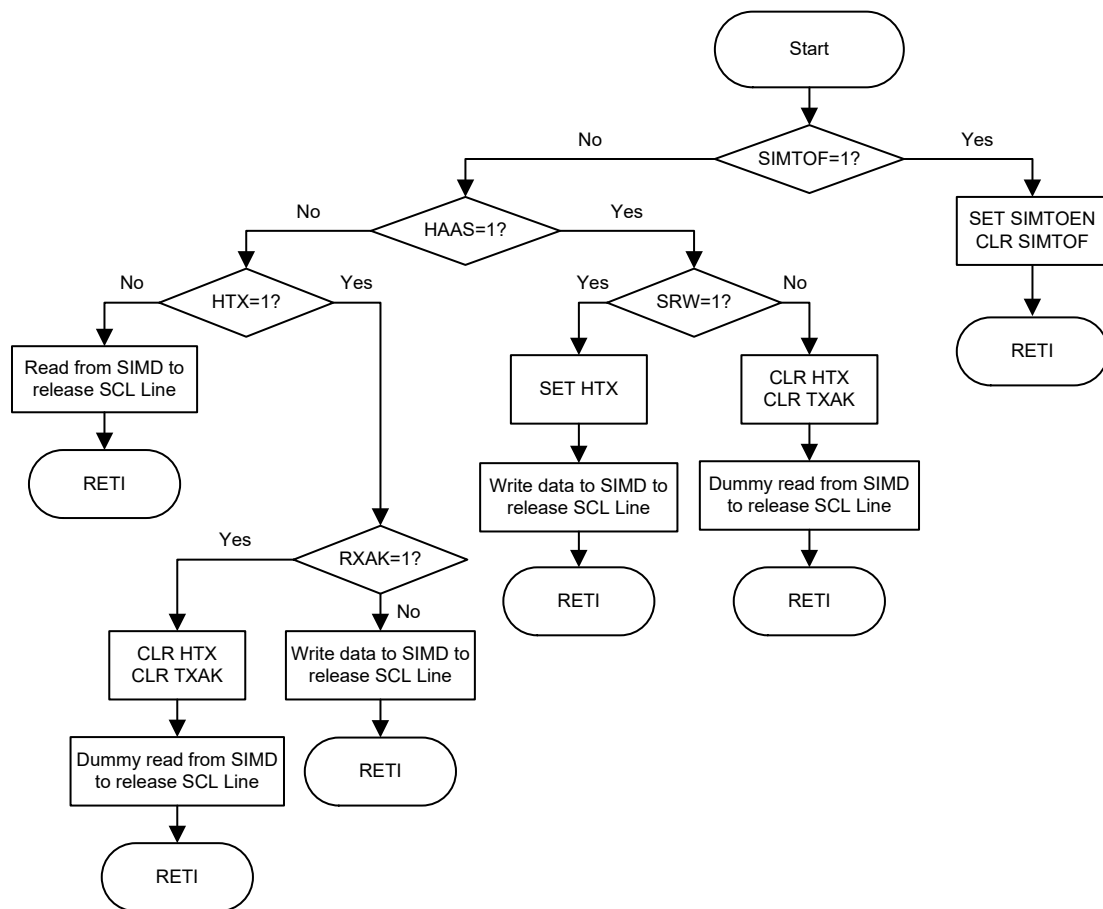
在从机确认接收到从地址后，会进行 8 位宽度的数据传输。这个数据传输顺序是的高位在前，低位在后。接收方在接收到 8 位数据后必须发出一个应答信号（“0”）以继续接收下一个数据。如果从机发送方没接收到来自主机接收方的应答信号，发送方将释放 SDA 线，此时主机方可发出 STOP 信号以释放 I²C 总线。所传送的数据存储在 SIMD 寄存器中。如果设置成发送方，从机必须先将欲传输的数据写到 SIMD 寄存器中；如果设置成接收方，从机必须从 SIMD 寄存器读取数据。

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。



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。

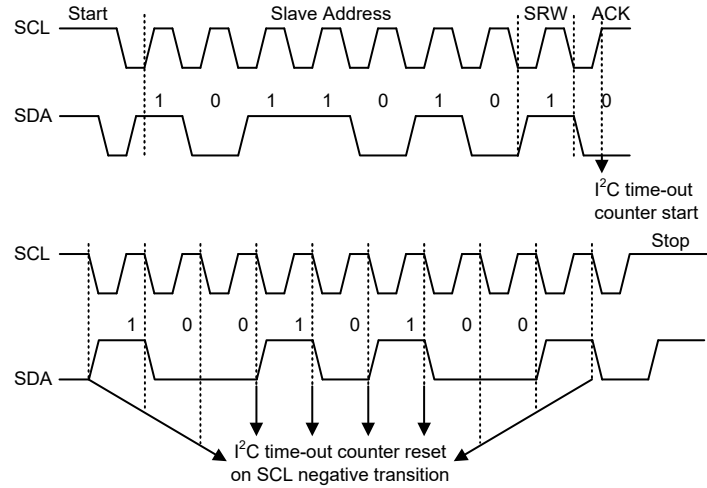
I²C 通信时序图



I²C 总线 ISR 流程图

I²C 超时控制

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



I²C 超时时序图

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

寄存器	I²C 超时发生后
SIMD, SIMA, SIMC0	保持不变
SIMC1	复位至 POR

超时发生后的 I²C 寄存器

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

● SIMTOC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **SIMTOEN**: SIM I²C 超时控制位

0: 除能
1: 使能

Bit 6 **SIMTOF**: SIM I²C 超时标志位

0: 超时未发生
1: 超时发生

Bit 5~0 **SIMTOS5~SIMTOS0**: SIM I²C 超时时间选择位

I²C 超时时钟源是 $f_{SUB}/32$

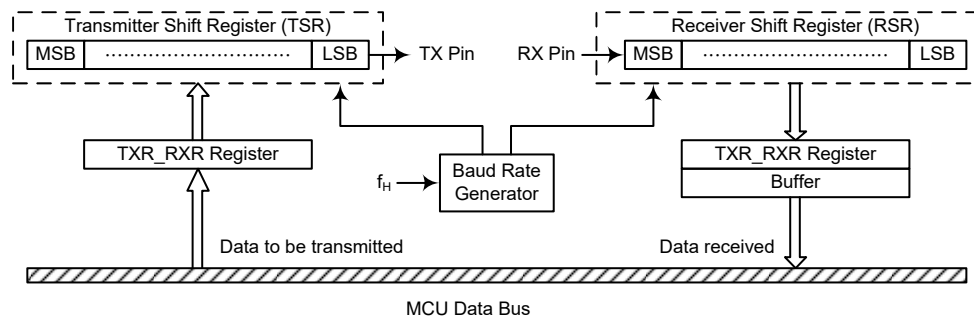
I²C 超时时间计算方法: $(SIMTOS[5:0]+1) \times (32/f_{SUB})$

UART 接口

该单片机具有一个全双工的异步串行通信接口，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 功能占用一个内部中断向量，当接收到数据或数据发送结束，触发 UART 中断。

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 接收数据缓冲器
- RX 引脚唤醒功能
- 发送和接收中断
- 中断可由下列条件触发：
 - ◆ 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 UARTEN 和 TXEN/RXEN 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 UARTEN、TXEN 或 RXEN 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 TXR_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。TXR_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 TXR_RXR 寄存器中。TXR_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数据存储器地址的数据寄存器，即 TXR_RXR 寄存器。

UART 状态和控制寄存器

与 UART 功能相关的有五个寄存器，包括控制 UART 模块整体功能的 USR、UCR1 和 UCR2 寄存器，控制波特率的 BRG 寄存器，管理发送和接收数据的数据寄存器 TXR_RXR。

寄存器名称	位							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0

UART 寄存器列表

● USR 寄存器

寄存器 USR 是 UART 的状态寄存器，可以通过程序读取。所有 USR 位是只读的。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

Bit 7 **PERR**: 奇偶校验出错标志位

0: 奇偶校验正确

1: 奇偶校验出错

PERR 是奇偶校验出错标志位。若 PERR=0，奇偶校验正确；若 PERR=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 USR 寄存器再读 TXR_RXR 寄存器来清除此位。

Bit 6 **NF**: 噪声干扰标志位

0: 未检测到噪声

1: 检测到噪声

NF 是噪声干扰标志位。若 NF=0，没有受到噪声干扰；若 NF=1，UART 接收数据时受到噪声干扰。它与 RXIF 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 USR 寄存器再读 TXR_RXR 寄存器将清除此标志位。

Bit 5	FERR: 帧错误标志位 0: 无帧错误发生 1: 有帧错误发生 FERR 是帧错误标志位。若 FERR=0, 没有帧错误发生; 若 FERR=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 USR 寄存器再读 TXR_RXR 寄存器来清除此位。
Bit 4	OERR: 溢出错误标志位 0: 无溢出错误发生 1: 有溢出错误发生 OERR 是溢出错误标志位, 表示接收缓冲器是否溢出。若 OERR=0, 没有溢出错误; 若 OERR=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 USR 寄存器再读 TXR_RXR 寄存器将清除此标志位。
Bit 3	RIDLE: 接收状态标志位 0: 正在接收数据 1: 接收器空闲 RIDLE 是接收状态标志位。若 RIDLE=0, 正在接收数据; 若 RIDLE=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, RIDLE 被置位, 表明 UART 空闲, RX 脚处于逻辑高状态。
Bit 2	RXIF: 接收寄存器状态标志位 0: TXR_RXR 寄存器为空 1: TXR_RXR 寄存器含有有效数据 RXIF 是接收寄存器状态标志位。当 RXIF=0, TXR_RXR 寄存器为空; 当 RXIF=1, TXR_RXR 寄存器接收到新数据。当数据从移位寄存器加载到 TXR_RXR 寄存器中, 如果 UCR2 寄存器中的 RIE=1, 则会触发中断。当接收数据时检测到一个或多个错误时, 相应的标志位 NF、FERR 或 PERR 会在同一周期内置位。读取 USR 寄存器再读 TXR_RXR 寄存器, 如果 TXR_RXR 寄存器中没有新的数据, 那么将清除 RXIF 标志。
Bit 1	TIDLE: 数据发送完成标志位 0: 数据传输中 1: 无数据传输 TIDLE 是数据发送完成标志位。若 TIDLE=0, 数据传输中。当 TXIF=1 且数据发送完毕或者暂停字被发送时, TIDLE 置位。TIDLE=1, TX 引脚空闲且处于逻辑高状态。读取 USR 寄存器再写 TXR_RXR 寄存器将清除 TIDLE 位。数据字符或暂停字就绪时, 不会产生该标志位。
Bit 0	TXIF: 发送数据寄存器 TXR_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 (TXR_RXR 数据寄存器为空) TXIF 是发送数据寄存器为空标志位。若 TXIF=0, 数据还没有从缓冲器加载到移位寄存器中; 若 TXIF=1, 数据已从缓冲器中加载到移位寄存器中。读取 USR 寄存器再写 TXR_RXR 寄存器将清除 TXIF。当 TXEN 被置位, 由于发送缓冲器未满, TXIF 也会被置位。

● UCR1 寄存器

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

Bit	7	6	5	4	3	2	1	0
Name	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
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 **UARTEN**: 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。UARTEN=0, UART 除能, RX 和 TX 处于浮空状态; UARTEN=1, UART 使能, TX 和 RX 将分别由 TXEN 和 RXEN 控制。当 UART 被除能清除缓冲器, 所有缓冲器中的数据将被忽略, 另外波特率计数器、错误和状态标志位被复位, TXEN、RXEN、TXBRK、RXIF、OERR、FERR、PERR 和 NF 清零, 而 TIDLE、TXIF 和 RIDLE 置位, UCR1、UCR2 和 BRG 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零, 所有发送和接收将停止, 模块也将复位成上述状态。当 UART 再次使能时, 它将在上次配置下重新工作。
- Bit 6 **BNO**: 发送数据位数选择位
 0: 8-bit 传输数据
 1: 9-bit 传输数据
 BNO 是发送数据位数选择位。BNO=1, 传输数据为 9 位; BNO=0, 传输数据为 8 位。若选择了 9 位数据传输格式, RX8 和 TX8 将分别存储接收和发送数据的第 9 位。
- Bit 5 **PREN**: 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。PREN=1, 使能奇偶校验; PREN=0, 除能奇偶校验。
- Bit 4 **PRT**: 奇偶校验选择位
 0: 偶校验
 1: 奇校验
 奇偶校验选择位。PRT=1, 奇校验; PRT=0, 偶校验。
- Bit 3 **STOPS**: 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。STOP=1, 有两位停止位; STOP=0, 只有一位停止位。
- Bit 2 **TXBRK**: 暂停字发送控制位
 0: 没有暂停字要发送
 1: 发送暂停字
 TXBRK 是暂停字发送控制位。TXBRK=0, 没有暂停字要发送, TX 引脚正常操作; TXBRK=1, 将会发送暂停字, 发送器将发送逻辑“0”。若 TXBRK 为高, 缓冲器中数据发送完毕后, 发送器输出将至少保持 13 位宽的低电平直至 TXBRK 复位。
- Bit 1 **RX8**: 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。BNO 是用来控制传输位数是 8 位还是 9 位。
- Bit 0 **TX8**: 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。BNO 是用来控制传输位数是 8 位还是 9 位。

● UCR2 寄存器

UCR2 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 UART 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	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
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 TXEN: 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。TXEN=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 TXEN=1 且 UARTE=1，则发送将被使能，TX 引脚将由 UART 来控制。在数据传输时清除 TXEN 将中止数据发送且复位发送器，此时 TX 引脚将处于浮空状态。
- Bit 6 RXEN: 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。RXEN=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 RXEN=1 且 UARTE=1，则接收将被使能，RX 引脚将由 UART 来控制。在数据传输时清除 RXEN 将中止数据接收且复位接收器，此时 RX 引脚将处于浮空状态。
- Bit 5 BRGH: 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 BRG 寄存器一起控制 UART 的波特率。BRGH=1，为高速模式；BRGH=0，为低速模式。
- Bit 4 ADDEN: 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。ADDEN=1，地址检测使能，此时数据的第 8 位 (BNO=0) 或第 9 位 (BNO=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 WAKE: 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_{cl} 关闭时有效。若 UART 时钟源 f_{cl} 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_{cl} 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_{cl}，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 RIE: 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 RIE=1，当 OERR 或 RXIF 置位时，UART 的中断请求标志置位；若 RIE=0，UART 中断请求标志不受 OERR 和 RXIF 影响。
- Bit 1 TIIE: 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 TIIE=1，当发送器空闲触发 TIDLE

置位时，UART 的中断请求标志置位；若 TIIE=0，UART 中断请求标志不受 TIDLE 的影响。

Bit 0 **TEIE**: 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 TEIE=1，当发送器为空触发 TXIF 置位时，UART 的中断请求标志置位；若 TEIE=0，UART 中断请求标志不受 TXIF 的影响。

• TXR_RXR 寄存器

TXR_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	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
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 **TXRX7~TXRX0**: UART 发送 / 接收数据位 Bit 7~Bit 0

• BRG 寄存器

Bit	7	6	5	4	3	2	1	0
Name	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
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 **BRG7~BRG0**: 波特率值

软件设置 BRGH 位 (设置波特率发生器的速度) 和 BRG 寄存器 (设置波特率的值)，一起控制 UART 的波特率。

注：若 BRGH=0，波特率 = $f_{H}/[64 \times (N+1)]$ ；

若 BRGH=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 BRG 寄存器和 UCR2 寄存器的 BRGH 位来控制。BRGH 是决定波特率发生器处于高速模式还是低速模式，从而决定计算公式的选用。BRG 寄存器的值 N 可根据下表中的公式计算，N 的范围是 0 到 255。

UCR2 的 BRGH 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 BRGH 来选择相应的计算公式从而算出 BRG 的值。由于 BRG 的值不连续，所以实际波特率和理论值之间有一个偏差。下面举例怎样计算 BRG 寄存器中的值 N 和误差。

波特率和误差的计算

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

根据上表，波特率 $BR = f_{H}/[64(N+1)]$

转换后的公式 $N = [f_H / (BR \times 64)] - 1$

带入参数 $N = [4000000 / (4800 \times 64)] - 1 = 12.0208$

取最接近的值，十进制 12 写入 BRG 寄存器，实际波特率如下

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

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 UCR1 寄存器的 BNO、PRT、PREN 和 STOPS 设定。用于数据发送和接收的波特率由一个内部的 8 位波特率发送器产生，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 UCR1 寄存器的 UARTEN 位来使能和除能的。若 UARTEN、TXEN 和 RXEN 都为高，则 TX 和 RX 分别为 UART 的发送端口和接收端口。若没有数据发送，TX 引脚默认状态为高电平。

UARTEN 清零将除能 TX 和 RX，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 TXEN、RXEN、TXBRK、RXIF、OERR、FERR、PERR 和 NF 清零，而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2 和 BRG 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零，所有发送和接收将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

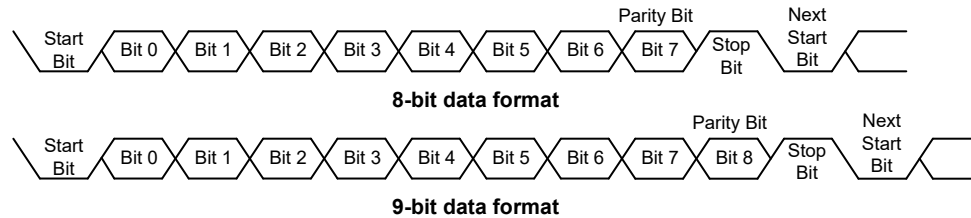
数据位、停止位位数以及奇偶校验的选择

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 UCR1 寄存器的各个位控制的。BNO 决定数据传输是 8 位还是 9 位；PRT 决定校验类型；PREN 决定是否选择奇偶校验；而 STOPS 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

UCR1 寄存器的 BNO 位是控制数据传输的长度。BNO=1 其长度为 9 位，第 9 位 MSB 存储在 UCR1 寄存器的 TX8 中。发送器的核心是发送移位寄存器 TSR，它的数据由发送寄存器 TXR_RXR 提供，应用程序只须将发送数据写入 TXR_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 TXR_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。TXEN=1，发送使能，但若 TXR_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 TXR_RXR 寄存器再置高 TXEN 也会触发发送。当发送器使能，若 TSR 寄存器为空，数据写入 TXR_RXR 寄存器将会直接加载到 TSR 寄存器中。发送器工作时，TXEN 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，TXR_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 UCR1 寄存器的 TX8。

发送器初始化可由如下步骤完成：

- 正确地设置 BNO、PRT、PREN 和 STOPS 位以确定数据长度、校验类型和停止位长度。
- 设置 BRG 寄存器，选择期望的波特率。
- 置高 TXEN，使能 UART 发送器且使 TX 作为 UART 的发送端。
- 读取 USR 寄存器，然后将待发数据写入 TXR_RXR 寄存器。注意，此步骤会清除 TXIF 标志位。

如果要发送多个数据只需重复上一步骤。

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

1. 读取 USR 寄存器
2. 写 TXR_RXR 寄存器

只读标志位 TXIF 由 UART 硬件置位。若 TXIF=1，TXR_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 TEIE=1，TXIF 标志位会产生中断。在数据传输时，写 TXR_RXR 指令会将待发数据暂存在 TXR_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 TXR_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 TXIF 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 TIDLE 位将被置位。

可以通过以下步骤来清除 TIDLE:

1. 读取 USR 寄存器
2. 写 TXR_RXR 寄存器

清除 TXIF 和 TIDLE 软件执行次序相同。

发送暂停字

若 TXBRK=1 且此状态保持时间超过 $[(BRG+1) \times t_{th}]$, 下一帧将会发送暂停字。它是由一个起始位、 $13 \times N$ ($N=1, 2, \dots$) 位逻辑 0 组成。置位 TXBRK 将会发送暂停字, 而清除 TXBRK 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 TXBRK 持续为高, 那么发送器会一直发送暂停字; 当应用程序将 TXBRK 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 BNO=1, 数据长度为 9 位, 而最高位 MSB 存放在 UCR1 寄存器的 RX8 中。接收器的核心是串行移位寄存器 RSR。RX 引脚上的数据送入数据恢复器中, 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 TXR_RXR 寄存器为空, 数据从 RSR 寄存器中加载到 TXR_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。TXR_RXR 寄存器在内部总线和接收移位寄存器间形成一个缓冲。TXR_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 TXR_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 BNO、PRT 和 PREN 位以确定数据长度和校验类型。
- 设置 BRG 寄存器, 选择期望的波特率。
- 置高 RXEN, 使能 UART 接收器且使 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 TXR_RXR 寄存器中包含有效数据时, USR 寄存器中的 RXIF 位将会置位, 溢出错误发生之前至多还有一帧数据可读。
- 若 RIE=1, 数据从 RSR 寄存器加载到 TXR_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 RXIF:

1. 读取 USR 寄存器
2. 读取 TXR_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 BNO 位的设置外加一个停止位来确定一帧数据的长度。若暂停字数大于 BNO 位指定的长度外加一个停止位，接收器认为接收已完毕，RXIF 和 FERR 置位，TXR_RXR 寄存器清 0，若相应的中断允许且 RIDLE 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 FERR 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 FERR 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 RIDLE。

UART 接收到暂停字会产生以下事件：

- 帧错误标志位 FERR 置位。
- TXR_RXR 寄存器清零。
- OERR、NF、PERR、RIDLE 或 RXIF 可能会置位。

空闲状态

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

接收中断

USR 寄存器的只读标志位 RXIF 由接收器的边沿触发置位。若 RIE=1，数据从移位寄存器 RSR 加载到 TXR_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 – OERR 标志

TXR_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 TXR_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：

- USR 寄存器中 OERR 被置位。
- TXR_RXR 寄存器中数据不会丢失。
- RSR 寄存器数据将会被覆盖。
- 若 RIE=1，将会产生中断。

先读取 USR 寄存器再读取 TXR_RXR 寄存器可将 OERR 清零。

噪声干扰 – NF 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 RXIF 上升沿，USR 寄存器中只读标志位 NF 置位。
- 数据从 RSR 寄存器加载到 TXR_RXR 寄存器中。
- 不产生中断，但此位置位发生在 RXIF 置位产生中断的同周期内。

先读取 USR 寄存器再读取 TXR_RXR 寄存器可将 NF 清零。

帧错误 – FERR 标志

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

奇偶校验错误 – PERR 标志

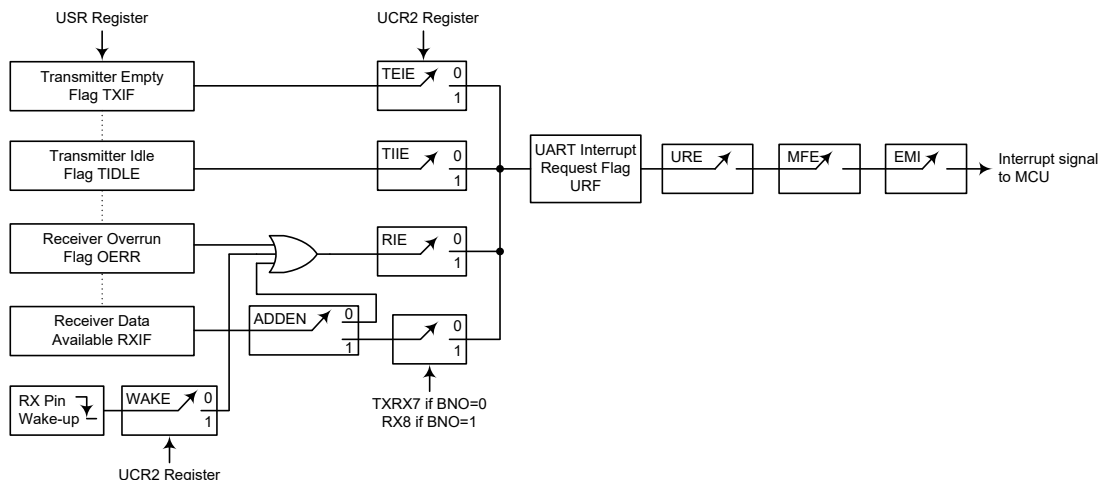
若接收到数据出现奇偶校验错误，USR 寄存器中只读标志 PERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。此标志位同接收的数据分别记录在 USR 寄存器和 TXR_RXR 寄存器中，此标志位可被任何复位清零。注意，在读取相应的数据之前必须先访问 USR 寄存器中的 FERR 和 PERR 错误标志位。

UART 模块中断结构

几个独立的 UART 条件可以产生一个 UART 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。若总中断使能位及相应的中断控制位使能且堆栈未满，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 UCR2 寄存器中相应中断允许位被置位，则 USR 寄存器中对应中断标志位将产生 UART 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 UART 中断源。

地址检测也是 UART 的中断源，它没有相应的标志位，若 UCR2 寄存器中 ADDEN=1，当检测到地址将会产生 UART 中断。RX 引脚唤醒也可以产生 UART 中断，它没有相应的标志位，当 UART 时钟源 f_{clk} 关闭且 UCR2 中的 WAKE 和 RIE 位被置位，RX 引脚上有下降沿时会产生 UART 中断。

注意，USR 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 中断结构图

地址检测模式

置位 UCR2 寄存器中的 ADDEN 将启动地址检测模式。若此位为“1”，可产生接收数据有效中断，其请求标志位为 RXIF。若 ADDEN 有效，只有在接收到数据最高位为 1 才会产生中断，注意 URE 和 EMI 中断使能位也要使能才会产生中断。地址的最高位为第 9 位 (BNO=1) 或第 8 位 (BNO=0)，若此位为高，则接收到的是地址而非数据。只有接收的数据的最后一位为高才会产生中断。若 ADDEN 除能，每接收到一个有效数据便会置位 RXIF，而不用考虑数据的最后一位。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。

ADDEN	9th Bit (BNO=1) 8th Bit (BNO=0)	产生 UART 中断
0	0	√
	1	√
1	0	×
	1	√

ADDEN 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，USR、UCR1、UCR2、接收 / 发送寄存器以及 BRG 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 UCR2 寄存器中 WAKE 位控制。当单片机进入空闲或休眠模式且 UART 时钟 f_h 关闭时，若 WAKE 位与 UART 允许位 UARTEN、接收器允许位 RXEN 和接收器中断允许位 RIE 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。

若要唤醒并产生 UART 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，总中断使能位 EMI 和 UART 中断使能控制位 URE 也必须置位；若这三个控制位没有被置位，那么单片机将可以被唤醒但不会产生中断。同样唤醒后系统需一定的延时才能正常工作，然后才会产生 UART 中断。

低电压检测 – LVD

此单片机具有低电压检测功能，即 LVD。该功能使能用于监测电源电压 V_{DD} ，若电源电压低于一定值可提供一个警告信号。此功能在电池类产品中非常有用，在电池电压较低时产生警告信号。低电压检测也可产生中断信号。

LVD 寄存器

低电压检测功能由 LVDC 寄存器控制。VLVD2~VLVD0 位用于选择 8 个固定电压中的一个参考点。LVDO 位被置位时低电压情况发生，若 LVDO 位为低表明 V_{DD} 电压工作在当前所设置低电压水平值之上。LVDEN 位用于控制低电压检测功能的开启/关闭，设置此位为高使能此功能，反之，关闭内部低电压检测电路。低电压检测会有一定的功耗，在不使用时可考虑关闭此功能，此举在功耗要求严格的电池供电应用中值得考虑。

• LVDC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	VBGEN	VLVD2	VLVD1	VLVD0
R/W	—	—	R	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 **LVDO**: LVD 输出标志位
0: 未检测到低电压
1: 检测到低电压

Bit 4 **LVDEN**: 低电压检测控制位
0: 除能
1: 使能

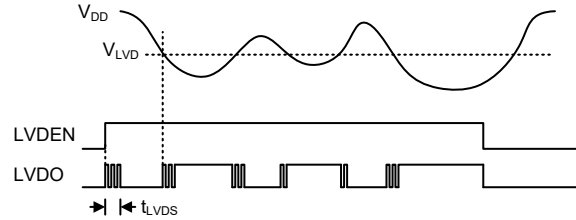
Bit 3 **VBGEN**: Bandgap 缓冲器控制位
0: 除能
1: 使能

应注意，当 LVD 或 LVR 功能使能或此位置位时，Bandgap 电路使能。

Bit 2~0 **VLVD2~VLVD0**: 选择 LVD 电压位
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

LVD 操作

通过比较电源电压 V_{DD} 与存储在 LVDC 寄存器中的预置电压值的结果，低电压检测功能工作。其设置的范围为 2.0V~4.0V。当电源电压 V_{DD} 低于预置电压值时，LVDO 位被置为高，表明低电压产生。低电压检测功能由一个自动使能的参考电压提供。当单片机处于休眠模式时，即使 LVDEN 位为高，低电压检测器除能。低电压检测器使能后，读取 LVDO 位前，电路稳定需要一定的延时 t_{LVDS} 。注意， V_{DD} 电压可能上升或下降比较缓慢，在 V_{LVD} 电压值附近时，LVDO 位可能有多种变化。



LVD 操作

低电压检测器也有自己的中断功能，它是除了轮询 LVDO 位之外的另一种检测低电压的方法。中断条件产生置位 LVDO 并延时 t_{LVD} 后，中断产生。此种情况下，若 V_{DD} 降至小于 LVD 预置电压值时，中断请求标志位 LVF 将被置位，中断产生，单片机将被从空闲模式中唤醒。若不要求低电压检测的唤醒功能使能，在单片机进入空闲模式前应将 LVF 标志置为高。

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定时器模块或 A/D 转换器有效，并且产生中断时，系统会暂时中止当前的程序而转到执行相对应的中断服务程序。此单片机提供多个外部中断和内部中断功能，外部中断由 INT0~INT1 引脚动作产生，而内部中断由各种内部功能，如定时器模块、时基、LVD、EEPROM 和 A/D 转换器等产生。

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。寄存器总的分为两类。第一类是 INTC0~INTC3 寄存器，用于设置基本的中断；第二类是 MFI 寄存器，用于设置多功能中断；最后一种是 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	注释
总中断	EMI	—	—
INTn 引脚	INTnE	INTnF	n=0~1
A/D 转换器	ADE	ADF	—
时基	TBnE	TBnF	n=0~1
SIM	SIME	SIMF	—
UART	URE	URF	—
多功能中断	MFE	MFF	—
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
STMn	STMnPE	STMnPF	n=0~1
	STMnAE	STMnAF	
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	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	INT1F	INT0F	D4	INT1E	INT0E	D1	EMI
INTC1	DEF	ADF	LVF	SIMF	DEE	ADE	LVE	SIME
INTC2	STM0AF	STM0PF	PTMAF	PTMPF	STM0AE	STM0PE	PTMAE	PTMPE
INTC3	MFF	D6	TB1F	TB0F	MFE	D2	TB1E	TB0E
MFI	—	URF	STM1AF	STM1PF	—	URE	STM1AE	STM1PE

中断寄存器列表

• INTEG 寄存器

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

Bit 7~4 未定义，读为“0”

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

00: 除能
01: 上升沿
10: 下降沿
11: 双沿

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

00: 除能
01: 上升沿
10: 下降沿
11: 双沿

• INTC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	INT1F	INT0F	D4	INT1E	INT0E	D1	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 **INT1F**: INT1 中断请求标志位

0: 无请求
1: 中断请求

Bit 5 **INT0F**: INT0 中断请求标志位

0: 无请求
1: 中断请求

Bit 4 **D4**: 保留位，必须固定为“0”

Bit 3 **INT1E**: INT1 中断控制位

0: 除能
1: 使能

Bit 2 **INT0E**: INT0 中断控制位

0: 除能
1: 使能

Bit 1 **D1**: 保留位，必须固定为“0”

Bit 0 **EMI**: 总中断控制位
0: 除能
1: 使能

• **INTC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	DEF	ADF	LVF	SIMF	DEE	ADE	LVE	SIME
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 **ADF**: A/D 转换器中断请求标志位
0: 无请求
1: 中断请求

Bit 5 **LVF**: LVD 中断请求标志位
0: 无请求
1: 中断请求

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

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

Bit 2 **ADE**: A/D 转换器中断控制位
0: 除能
1: 使能

Bit 1 **LVE**: LVD 中断控制位
0: 除能
1: 使能

Bit 0 **SIME**: SIM 中断控制位
0: 除能
1: 使能

• **INTC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	STM0AF	STM0PF	PTMAF	PTMPF	STM0AE	STM0PE	PTMAE	PTMPE
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 **STM0AF**: STM0 比较器 A 匹配中断请求标志位
0: 无请求
1: 中断请求

Bit 6 **STM0PF**: STM0 比较器 P 匹配中断请求标志位
0: 无请求
1: 中断请求

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 **STM0AE**: STM0 比较器 A 匹配中断控制位
0: 除能
1: 使能
- Bit 2 **STM0PE**: STM0 比较器 P 匹配中断控制位
0: 除能
1: 使能
- Bit 1 **PTMAE**: PTM 比较器 A 匹配中断控制位
0: 除能
1: 使能
- Bit 0 **PTMPE**: PTM 比较器 P 匹配中断控制位
0: 除能
1: 使能

• INTC3 寄存器

Bit	7	6	5	4	3	2	1	0
Name	MFF	D6	TB1F	TB0F	MFE	D2	TB1E	TB0E
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 **MFF**: 多功能中断请求标志位
0: 无请求
1: 中断请求
- Bit 6 **D6**: 保留位, 必须固定为 “0”
- Bit 5 **TB1F**: 时基 1 中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **TB0F**: 时基 0 中断请求标志位
0: 无请求
1: 中断请求
- Bit 3 **MFE**: 多功能中断控制位
0: 除能
1: 使能
- Bit 2 **D2**: 保留位, 必须固定为 “0”
- Bit 1 **TB1E**: 时基 1 中断控制位
0: 除能
1: 使能
- Bit 0 **TB0E**: 时基 0 中断控制位
0: 除能
1: 使能

• MFI 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	URF	STM1AF	STM1PF	—	URE	STM1AE	STM1PE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

- Bit 7 未定义, 读为 “0”
- Bit 6 **URF**: UART 中断请求标志位
0: 无请求
1: 中断请求

Bit 5	STM1AF: STM1 比较器 A 匹配中断请求标志位 0: 无请求 1: 中断请求
Bit 4	STM1PF: STM1 比较器 P 匹配中断请求标志位 0: 无请求 1: 中断请求
Bit 3	未定义, 读为 “0”
Bit 2	URE: UART 中断控制位 0: 除能 1: 使能
Bit 1	STM1AE: STM1 比较器 A 匹配中断控制位 0: 除能 1: 使能
Bit 0	STM1PE: STM1 比较器 P 匹配中断控制位 0: 除能 1: 使能

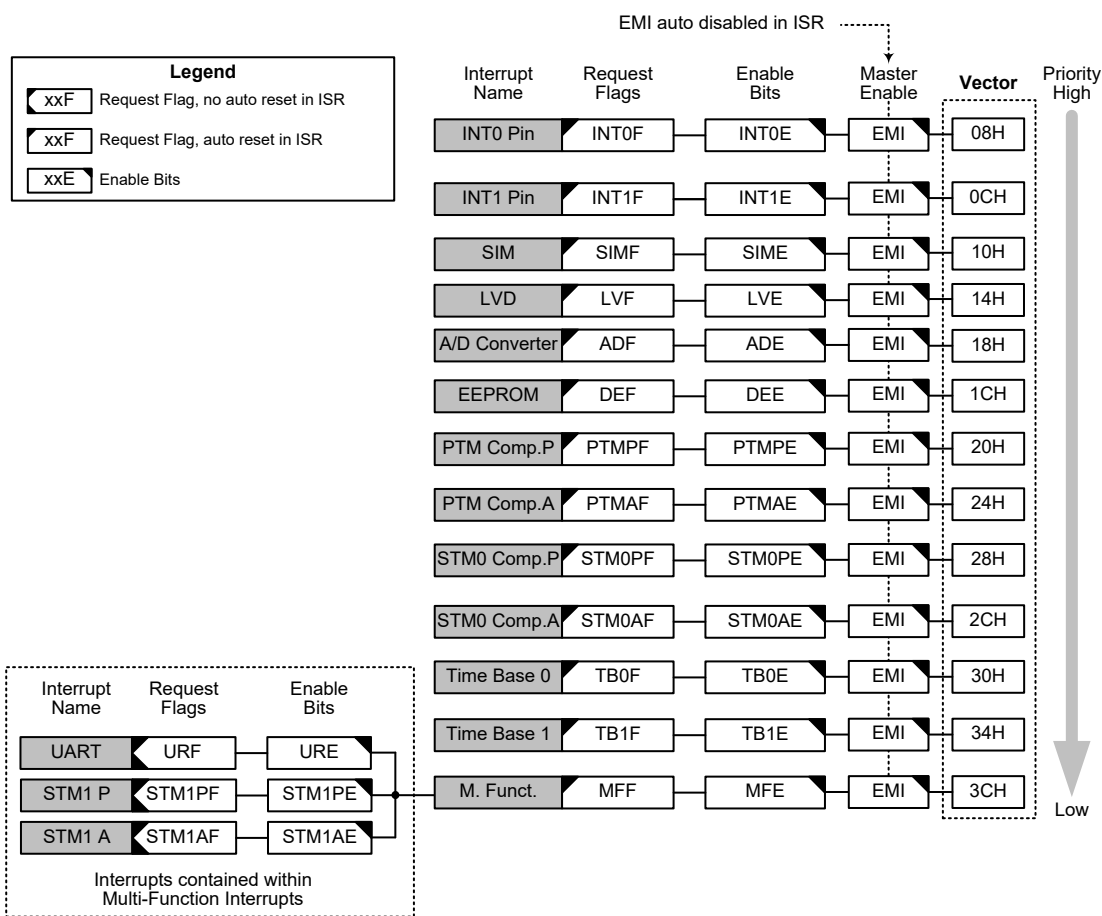
中断操作

若中断事件条件产生, 如一个 TM 比较器 P、比较器 A 匹配或 A/D 转换结束等等, 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

外部中断

通过 INT0~INT1 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，INT0~INT1 引脚的状态发生变化，外部中断请求标志 INT0F~INT1F 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 INT0E~INT1E 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，并且通过引脚共用寄存器选择外部中断脚，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 INT0F~INT1F 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，其上拉电阻选项仍保持有效。

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。

A/D 转换器中断

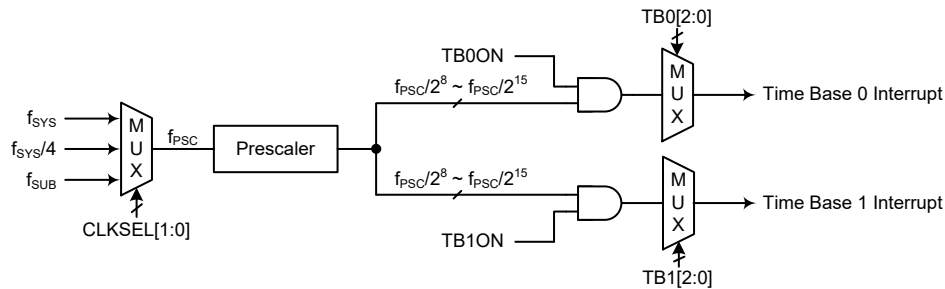
A/D 转换器中断由 A/D 转换动作的结束来控制。当 A/D 转换器中断请求标志 ADF 被置位，即 A/D 转换过程完成时，中断请求发生。当总中断使能位 EMI

和 A/D 中断使能位 ADE 被置位，允许程序跳转到各自的中断向量地址。当中断使能，堆栈未滿且 A/D 转换动作结束时，将调用 A/D 转换器中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 ADF 会自动清零。EMI 位也会被清零以除能其它中断。

时基中断

时基中断提供一个固定周期的中断信号，由各自的定时器功能产生溢出信号控制。当各自的中断请求标志 TB0F~TB1F 被置位时，中断请求发生。当总中断使能位 EMI 和时基使能位 TB0E~TB1E 被置位，允许程序跳转到各自的中断向量地址。当中断使能，堆栈未滿且时基溢出时，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 TB0F~TB1F 会自动复位且 EMI 位会被清零以除能其它中断。

时基中断的目的是提供一个固定周期的中断信号，时钟源来自时钟源 f_{PSC} 。时钟源 f_{PSC} 来自内部时钟源 f_{SYS} 、 $f_{SYS}/4$ 或 f_{SUB} 。 f_{PSC} 输入时钟首先经过分频器，分频率由程序设置 TB0C~TB1C 寄存器相关位获取合适的分频值以提供更长的时基中断周期。相应的控制时基中断周期的时钟源可通过 PSCR 寄存器的 CLKSEL1~CLKSEL0 位选择。



时基中断

• PSCR 寄存器

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

Bit 7~2 未定义，读为“0”

Bit 1~0 **CLKSEL1~CLKSEL0**: 预分频时钟源选择

00: f_{SYS}

01: $f_{SYS}/4$

1x: f_{SUB}

• TBnC 寄存器 (n=0~1)

Bit	7	6	5	4	3	2	1	0
Name	TBnON	—	—	—	—	TBn2	TBn1	TBn0
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TBnON**: 时基 n 控制位

0: 除能

1: 使能

Bit 6~3	未定义，读为“0”
Bit 2~0	TBn2~TBn0 : 选择时基 n 溢出周期
	000: $2^8/f_{PSC}$
	001: $2^9/f_{PSC}$
	010: $2^{10}/f_{PSC}$
	011: $2^{11}/f_{PSC}$
	100: $2^{12}/f_{PSC}$
	101: $2^{13}/f_{PSC}$
	110: $2^{14}/f_{PSC}$
	111: $2^{15}/f_{PSC}$

多功能中断

该单片机中有一种多功能中断，与其它中断不同，它没有独立源，但由其它现有的中断源构成，即 STM1 中断和 UART 中断。

当多功能中断请求标志 MFF 被置位，多功能中断请求产生。当中断使能，堆栈未满，包括在多功能中断中的任意一个中断发生时，将调用多功能中断向量中的一个子程序。当响应中断服务子程序时，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不会自动复位，必须由应用程序清零。

串行接口模块中断

串行接口模块中断，即 SIM 中断。当一个字节数据已由 SIM 接口接收或发送完，或 I²C 从机地址匹配，或 I²C 超时，中断请求标志 SIMF 被置位，SIM 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和串行接口中断使能位 SIME 需先被置位。当中断使能，堆栈未满且以上任一种情况发生时，可跳转至相关多功能中断向量子程序中执行。当响应中断服务子程序时，串行接口中断标志位 SIMF 会自动复位且 EMI 将被自动清零以除能其它中断。

UART 中断

UART 中断属于多功能中断，由几种 UART 条件来控制。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，UART 中断请求标志 URF 被置位，UART 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、UART 中断使能位 URE 和多功能使能位 MFE 需先被置位。当中断使能，堆栈未满且以上任何一种情况发生时，将调用 UART 中断向量子程序。当响应中断服务子程序时，EMI 位会被清零以除能其它中断，多功能中断请求标志位也将自动清零。而 URF 标志位无法自动清除，需通过应用程序清除。然而 USR 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，详情请参考 UART 章节。

LVD 中断

当低电压检测功能检测到一个低电压时，LVD 中断请求标志 LVF 被置位，LVD 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 和低电压中断使能位 LVE 需先被置位。当中断使能，堆栈未满且低电压条件发生时，将调用 LVD 中断向量子程序。当低电压中断响应，LVD 中断请求标志 LVF 自动清除，EMI 将被自动清零以除能其它中断。

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，将调用 EEPROM 中断向量程序。当 EEPROM 中断响应，相应中断请求标志位 DEF 会自动清零且 EMI 位也会被清零以除能其它中断。

TM 中断

标准型和周期型 TM 各有两个中断，分别来自比较器 P 和比较器 A 匹配。STM1 中断属于多功能中断，而 STM0 和 PTM 中断源有自己独立的向量。所有 TM 均带有两个中断请求标志位及两个使能位。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。

对于 STM1，若要程序跳转到相应中断向量地址，总中断控制位 EMI，STM1 中断使能位 STM1E 和多功能使能位 MFE 需先被置位。当中断使能，堆栈未满且 STM1 比较器匹配情况发生时，可跳转至相关中断向量程序中执行。当 STM1 中断响应，仅 MFF 标志位会被自动清零，STM1 中断请求标志位 STM1F 需通过应用程序手动清零。

对于 STM0 或 PTM，若要程序跳转到相应中断向量地址，总中断控制位 EMI 和相应的中断使能位 STM0E 或 PTME 需先被置位。当中断使能，堆栈未满且 STM0 或 PTM 的比较器 P 或比较器 A 匹配情况发生时，可调用相应的 TM 中断向量程序。当 STM0 或 PTM 中断响应，TM 中断标志位 STM0F 或 PTMF 自动清除，EMI 将被自动清零以除能其它中断。

中断唤醒功能

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

编程注意事项

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

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

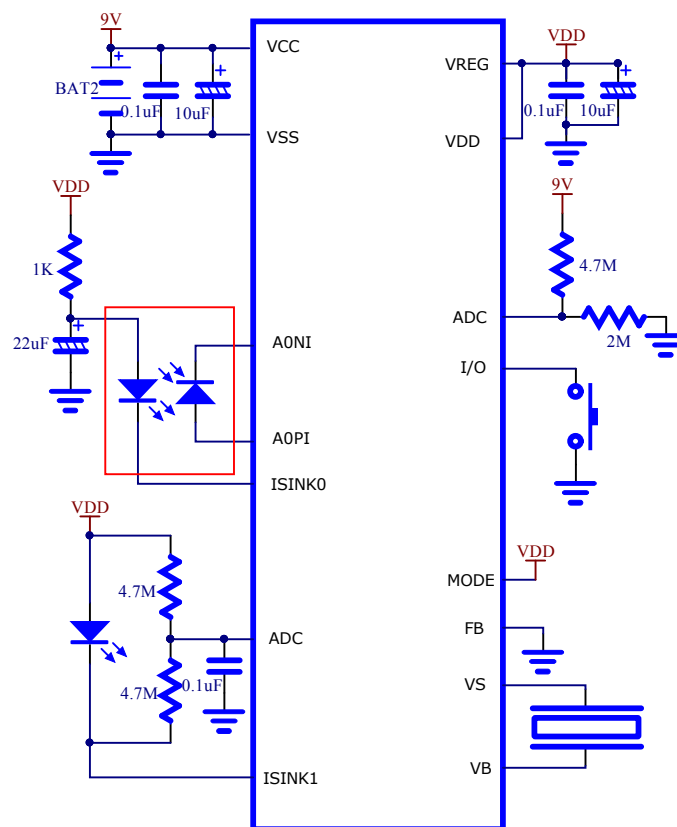
建议在中断服务子程序中不要使用“CALL 子程序”指令。中断通常发生在不可预料的情况或是需要立刻执行的某些应用。假如只剩下一层堆栈且没有控制好中断，当“CALL 子程序”在中断服务子程序中执行时，将破坏原来的控制序列。

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

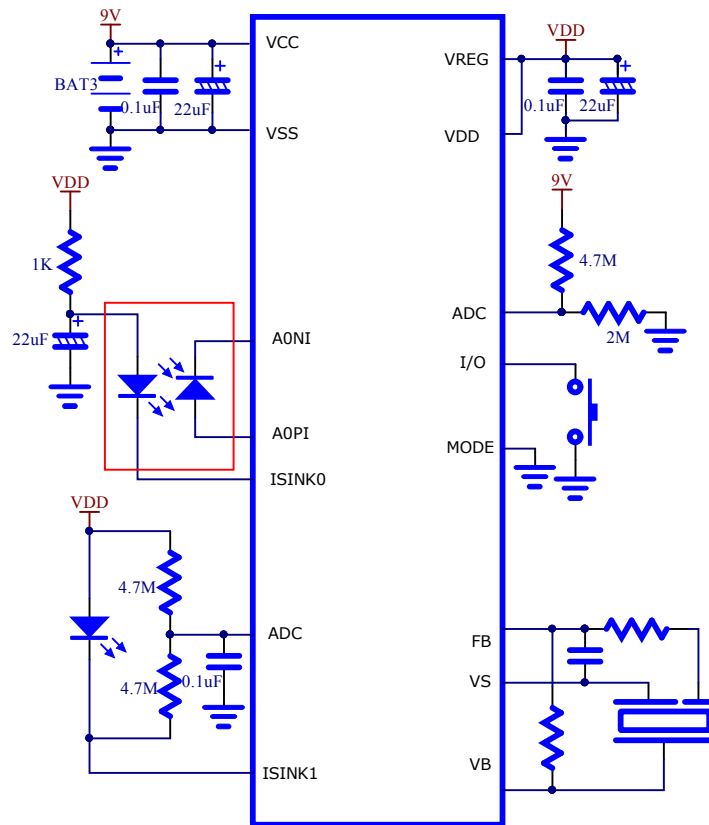
当进入中断服务程序，系统仅将程序计数器的内容压入堆栈，如果中断服务程序会改变状态寄存器或其它的寄存器的内容而破坏控制流程，应事先将这些数据保存起来。若从中断子程序中返回可执行 RET 或 RETI 指令。除了能返回至主程序外，RETI 指令还能自动设置 EMI 位为高，允许进一步中断。RET 指令只能返回至主程序，清除 EMI 位，除能进一步中断。

应用电路

他激式蜂鸣器应用电路



自激式蜂鸣器应用电路



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 0.5 μ s 中执行完成，而分支或调用操作则将在 1 μ s 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入口的 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, x	ACC 与立即数、进位标志相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SBC A,[m]	ACC 与数据存储器、进位标志相减，结果放入 ACC	1	Z, C, AC, OV, 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
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，读取特定页的 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 2 个周期，如果没有发生跳转，则只需一个周期。
2. 任何指令若要改变 PCL 的内容将需要 2 个周期来执行。

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，此举不仅可节省 Flash 存储器空间的使用，同时可提高 CPU 执行效率。

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

助记符	说明	指令周期	影响标志位
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
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	2 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]	读取特定页的 ROM 内容，并送至数据存储器 and TBLH	3 注	无
LTABRDL [m]	读取最后页的 ROM 内容，并送至数据存储器 and TBLH	3 注	无
LITABRD [m]	读表指针 TBLP 自加，读取特定页的 ROM 内容，并送至数据存储器 and 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 and 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 3 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	$ACC \leftarrow ACC + [m] + C$
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	$[m] \leftarrow ACC + [m] + C$
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	$ACC \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC + x$
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	$[m] \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC \text{ "AND" } [m]$
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	$ACC \leftarrow ACC \text{ “AND” } x$
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	$[m] \leftarrow ACC \text{ “AND” } [m]$
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	$Stack \leftarrow Program\ Counter + 1$ $Program\ Counter \leftarrow addr$
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	$[m] \leftarrow 00H$
影响标志位	无
CLR [m].i	Clear bit of Data Memory
指令说明	将指定数据存储器的第 i 位内容清零。
功能表示	$[m].i \leftarrow 0$
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 $TO \ \& \ PDF \leftarrow 0$
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	$[m] \leftarrow \overline{[m]}$
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	$ACC \leftarrow \overline{[m]}$
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	$[m] \leftarrow ACC + 00H$ 或 $[m] \leftarrow ACC + 06H$ 或 $[m] \leftarrow ACC + 60H$ 或 $[m] \leftarrow ACC + 66H$
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	$[m] \leftarrow [m] - 1$
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	$ACC \leftarrow [m] - 1$
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	$TO \leftarrow 0$ $PDF \leftarrow 1$
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	$[m] \leftarrow [m] + 1$
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	$ACC \leftarrow [m] + 1$
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 $\leftarrow$ 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	$ACC \leftarrow [m]$
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	$ACC \leftarrow x$
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	$[m] \leftarrow ACC$
影响标志位	无
NOP	No operation
指令说明	空操作，接下来顺序执行下一条指令。
功能表示	无操作
影响标志位	无
ORA, [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	$ACC \leftarrow ACC \text{ "OR" } [m]$
影响标志位	Z
ORA, 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	$ACC \leftarrow ACC \text{ "OR" } x$
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	$[m] \leftarrow ACC \text{ "OR" } [m]$
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 $\leftarrow$ 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 $\leftarrow$ Stack $ACC \leftarrow x$
影响标志位	无

RETI	Return from interrupt
指令说明	将堆栈寄存器中的程序计数器值恢复且中断功能通过设置 EMI 位重新使能。EMI 是控制中断使能的主控制位。如果在执行 RETI 指令之前还有中断未被响应，则这个中断将在返回主程序之前被响应。
功能表示	Program Counter $\leftarrow$ Stack
影响标志位	EMI $\leftarrow$ 1 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) $\leftarrow$ [m].i (i=0~6) [m].0 $\leftarrow$ [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) $\leftarrow$ [m].i (i=0~6) ACC.0 $\leftarrow$ [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) $\leftarrow$ [m].i (i=0~6) [m].0 $\leftarrow$ C C $\leftarrow$ [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) $\leftarrow$ [m].i (i=0~6) ACC.0 $\leftarrow$ C C $\leftarrow$ [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	$[m].i \leftarrow [m].(i+1) (i=0\sim6)$ $[m].7 \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1) (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
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	$[m].i \leftarrow [m].(i+1) (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1) (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC \leftarrow ACC - [m] - \bar{C}$
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	$ACC \leftarrow ACC - [m] - \bar{C}$
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	$[m] \leftarrow ACC - [m] - \bar{C}$
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	$[m] \leftarrow [m] - 1$ ，如果 $[m]=0$ 跳过下一条指令执行
影响标志位	无
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC
指令说明	将指定数据存储器内容减 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果将存放到累加器，但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	$ACC \leftarrow [m] - 1$ ，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	$[m] \leftarrow FFH$
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	$[m].i \leftarrow 1$
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	$[m] \leftarrow [m] + 1$ ，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	$ACC \leftarrow [m] + 1$ ，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 $[m].i \neq 0$ ，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	指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定存储器，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 $[m] \neq 0$ ，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	$ACC \leftarrow ACC - [m]$
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	$[m] \leftarrow ACC - [m]$
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	$ACC \leftarrow ACC - x$
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	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
影响标志位	无

SZ [m]	Skip if Data Memory is 0
指令说明	指定数据存储器的内容会先被读出，后又被重新写入指定数据存储单元内。判断指定数据存储器的内容是否为 0，若为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	如果 [m]=0，跳过下一条指令执行
影响标志位	无
SZA [m]	Skip if Data Memory is 0 with data movement to ACC
指令说明	将指定数据存储器内容复制到累加器，并判断指定数据存储器的内容是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	ACC ←[m]，如果 [m]=0，跳过下一条指令执行
影响标志位	无
SZ [m].i	Skip if bit i of Data Memory is 0
指令说明	判断指定数据存储器的第 i 位是否为 0，若为 0，则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	如果 [m].i=0，跳过下一条指令执行
影响标志位	无
TABRD [m]	Read table (specific page) to TBLH and Data Memory
指令说明	将表格指针 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 (specific page) to TBLH and data memory
指令说明	自加表格指针低字节 TBLP，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	自加表格指针低字节 TBLP，将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定的数据存储器且将高字节移至 TBLH。
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	$ACC \leftarrow ACC + [m] + C$
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	$[m] \leftarrow ACC + [m] + C$
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	$ACC \leftarrow ACC + [m]$
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	$[m] \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC \text{ “AND” } [m]$
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	$[m] \leftarrow ACC \text{ “AND” } [m]$
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	$[m] \leftarrow 00H$
影响标志位	无
LCLR [m].i	Clear bit of Data Memory
指令说明	将指定数据存储器的第 i 位内容清零。
功能表示	$[m].i \leftarrow 0$
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	$[m] \leftarrow \overline{[m]}$
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	$ACC \leftarrow \overline{[m]}$
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	$[m] \leftarrow ACC + 00H$ 或 $[m] \leftarrow ACC + 06H$ 或 $[m] \leftarrow ACC + 60H$ 或 $[m] \leftarrow ACC + 66H$
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	$[m] \leftarrow [m] - 1$
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	$ACC \leftarrow [m] - 1$
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	$[m] \leftarrow [m] + 1$
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	$ACC \leftarrow [m] + 1$
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	$ACC \leftarrow [m]$
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	$[m] \leftarrow ACC$
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	$ACC \leftarrow ACC \text{ "OR" } [m]$
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 and 累加器逻辑或，结果放到数据存储器。
功能表示	$[m] \leftarrow ACC \text{ "OR" } [m]$
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	$[m].(i+1) \leftarrow [m].i \ (i=0\sim6)$ $[m].0 \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i \ (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
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	$[m].(i+1) \leftarrow [m].i \ (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i \ (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
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	$[m].i \leftarrow [m].(i+1) (i=0\sim6)$ $[m].7 \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1) (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
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	$[m].i \leftarrow [m].(i+1) (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位	C
LRRC A [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	$ACC.i \leftarrow [m].(i+1) (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC \leftarrow ACC - [m] - \bar{C}$
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	$[m] \leftarrow ACC - [m] - \bar{C}$
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 3 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 3 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] 指令说明	Skip if increment Data Memory is 0 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 3 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
LSIZA [m] 指令说明	Skip if increment Data Memory is zero with result in ACC 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 3 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
LSNZ [m].i 指令说明	Skip if bit i of Data Memory is not 0 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 3 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 $[m].i \neq 0$ ，跳过下一条指令执行
影响标志位	无
LSNZ [m] 指令说明	Skip if Data Memory is not 0 指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定数据存储器，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 3 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 $[m] \neq 0$ ，跳过下一条指令执行
影响标志位	无
LSUBA, [m] 指令说明	Subtract Data Memory from ACC 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	$ACC \leftarrow ACC - [m]$
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	$[m] \leftarrow ACC - [m]$
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	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
影响标志位	无
LSZ [m]	Skip if Data Memory is 0
指令说明	指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定数据存储器的内容是否为 0，若为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 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 3 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	$ACC \leftarrow [m]$ ，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 3 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 (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
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 (specific page) to TBLH and data memory
指令说明	自加表格指针低字节 TBLP，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	自加表格指针低字节 TBLP，将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定的数据存储器且将高字节移至 TBLH。
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	$ACC \leftarrow ACC \text{ "XOR" } [m]$
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	$[m] \leftarrow ACC \text{ "XOR" } [m]$
影响标志位	Z

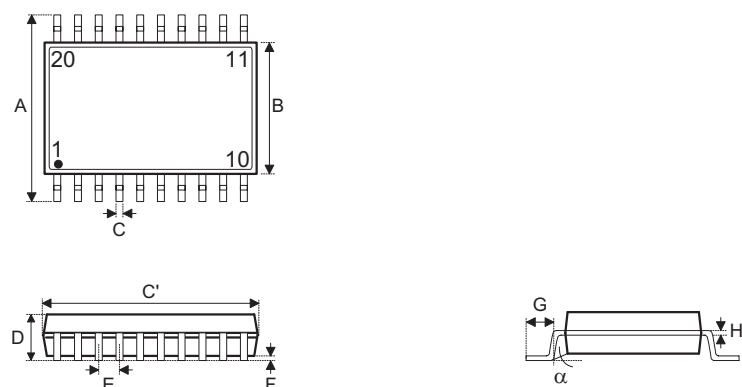
封装信息

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

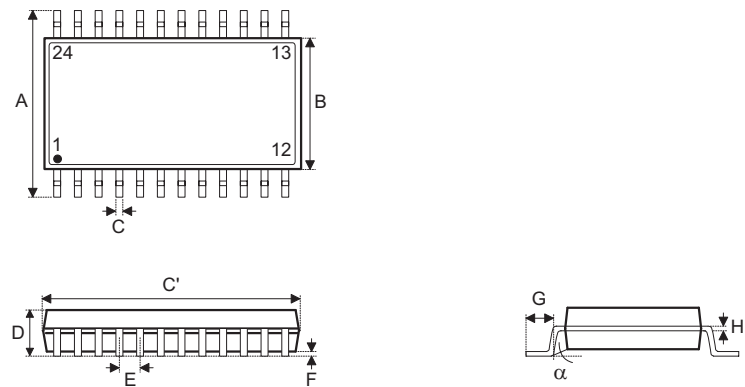
20-pin SOP (300mil) 外形尺寸



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

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

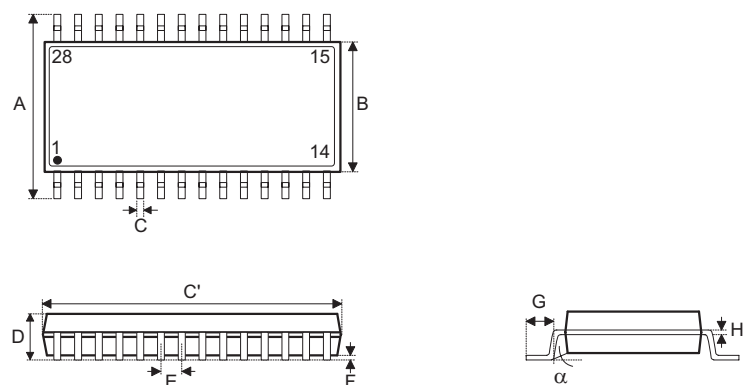
24-pin SOP (300mil) 外形尺寸



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

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

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°

Copyright© 2022 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

本文件出版时 HOLTEK 已针对所载信息为合理注意，但不保证信息准确无误。文中提到的信息仅是提供作为参考，且可能被更新取代。HOLTEK 不担保任何明示、默示或法定的，包括但不限于适合商品化、令人满意的质量、规格、特性、功能与特定用途、不侵害第三方权利等保证责任。HOLTEK 就文中提到的信息及该信息之应用，不承担任何法律责任。此外，HOLTEK 并不推荐将 HOLTEK 的产品使用在会由于故障或其他原因而可能会对人身安全造成危害的地方。HOLTEK 特此声明，不授权将产品使用于救生、维生或安全关键零部件。在救生 / 维生或安全应用中使用 HOLTEK 产品的风险完全由买方承担，如因该等使用导致 HOLTEK 遭受损害、索赔、诉讼或产生费用，买方同意出面进行辩护、赔偿并使 HOLTEK 免受损害。HOLTEK (及其授权方，如适用) 拥有本文件所提供信息 (包括但不限于内容、数据、示例、材料、图形、商标) 的知识产权，且该信息受著作权法和其他知识产权法的保护。HOLTEK 在此并未明示或暗示授予任何知识产权。HOLTEK 拥有不事先通知而修改本文件所载信息的权利。如欲取得最新的信息，请与我们联系。