



内置 LCD & EEPROM 增强 A/D 型 Flash 单片机
HT67F4892

版本 : V1.00 日期 : 2016-11-24

www.holtek.com

目 录

特性	6
CPU 特性	6
周边特性	7
概述	8
方框图	8
引脚图	9
引脚说明	11
极限参数	15
直流电气特性	15
交流电气特性	19
A/D 转换器电气特性	20
参考电压电气特性	20
LVD/LVR 电气特性	21
上电复位特性	21
系统结构	22
时序和流水线结构	22
程序计数器	23
堆栈	23
算术逻辑单元 – ALU	24
Flash 程序存储器	25
结构	25
特殊向量	25
查表	25
查表范例	26
在线烧录 – ICP	27
片上调试 – OCDS	28
RAM 数据存储器	29
结构	29
数据存储器寻址	30
通用数据存储器	30
特殊功能数据存储器	30
特殊功能寄存器	32
间接寻址寄存器 – IAR0, IAR1, IAR2	32
存储器指针 – MP0, MP1L, MP1H, MP2L, MP2H	32
累加器 – ACC	33
程序计数器低字节寄存器 – PCL	34
表格寄存器 – TBLP, TBHP, TBLH	34
状态寄存器 – STATUS	34

EEPROM 数据存储器	36
EEPROM 数据存储器结构	36
EEPROM 寄存器	36
从 EEPROM 中读取数据	38
写数据到 EEPROM	38
写保护	38
EEPROM 中断	38
编程注意事项	39
振荡器	40
振荡器概述	40
系统时钟配置	40
外部晶体 / 陶瓷振荡器 – HXT	41
内部 RC 振荡器 – HIRC	41
外部 32.768kHz 晶体振荡器 – LXT	42
内部 32kHz 振荡器 – LIRC	43
工作模式和系统时钟	44
系统时钟	44
系统工作模式	45
控制寄存器	46
工作模式切换	48
待机电流的注意事项	52
唤醒	52
看门狗定时器	53
看门狗定时器时钟源	53
看门狗定时器控制寄存器	53
看门狗定时器操作	54
复位和初始化	55
复位功能	55
复位初始状态	57
输入 / 输出端口	61
上拉电阻	61
PA 口唤醒	62
输入 / 输出端口控制寄存器	62
引脚共用功能	63
输入 / 输出引脚结构	63
编程注意事项	63
定时器模块 – TM	64
简介	64
TM 操作	64
TM 时钟源	64
TM 中断	64
TM 外部引脚	65
TM 输入 / 输出引脚控制寄存器	65

编程注意事项	67
简易型 TM – CTM.....	68
简易型 TM 操作	68
简易型 TM 寄存器介绍	68
简易型 TM 工作模式	72
周期型 TM – PTM	78
周期型 TM 操作	78
周期型 TM 寄存器介绍	78
周期型 TM 工作模式	82
A/D 转换器.....	91
A/D 转换器简介	91
A/D 转换器寄存器介绍	91
A/D 转换器操作	95
A/D 转换器参考电压	96
A/D 转换器输入信号	96
A/D 转换率和时序图	97
A/D 转换步骤概述	98
编程注意事项	98
A/D 转换功能	98
A/D 程序范例	99
串行接口模块 – SIM.....	101
SPI 接口	101
I ² C 接口	109
LCD 显示存储器.....	119
LCD 驱动输出	119
LCD 寄存器	119
LCD 波形	124
LED 驱动器	128
LED 驱动操作	128
UART 模块串行接口	129
UART 外部引脚接口.....	129
UART 数据传输方案.....	129
UART 状态和控制寄存器.....	130
波特率发生器	134
UART 模块的设置与控制.....	135
UART 发送器.....	137
UART 接收器.....	138
接受错误处理	139
UART 模块中断结构.....	140
UART 模块暂停和唤醒.....	141

中断	142
中断寄存器	142
中断操作	148
外部中断	149
A/D 转换器中断	149
多功能中断	149
SIM 中断	149
UART 中断	150
时基中断	150
EEPROM 中断	151
LVD 中断	151
TM 中断	151
中断唤醒功能	151
编程注意事项	152
低电压检测 – LVD	153
LVD 寄存器	153
LVD 操作	154
配置选项	155
应用电路	155
指令集	156
简介	156
指令周期	156
数据的传送	156
算术运算	156
逻辑和移位运算	156
分支和控制转换	157
位运算	157
查表运算	157
其它运算	157
指令集概要	158
惯例	158
扩展指令集	161
指令定义	163
扩展指令定义	175
封装信息	185
48-pin LQFP (7mm × 7mm) 外形尺寸	186
52-pin LQFP (14mm × 14mm) 外形尺寸	187

特性

CPU 特性

- 工作电压
 - ◆ $f_{\text{SYS}} = 4\text{MHz}$: 2.2V~5.5V
 - ◆ $f_{\text{SYS}} = 8\text{MHz}$: 2.2V~5.5V
 - ◆ $f_{\text{SYS}} = 12\text{MHz}$: 2.7V~5.5V
 - ◆ $f_{\text{SYS}} = 16\text{MHz}$: 4.5V~5.5V
- $V_{\text{DD}} = 5\text{V}$, 系统时钟为 16MHz 时, 指令周期为 0.25 μs
- 提供暂停和唤醒功能, 以降低功耗
- 四种振荡器模式
 - ◆ 内部高速 RC – HIRC
 - ◆ 内部低速 32kHz RC – LIRC
 - ◆ 外部高速晶振 – HXT
 - ◆ 外部低速 32.768kHz 晶振 – LXT
- 多种工作模式: 正常, 低速, 空闲, 休眠
- 高度集成内部 8MHz 振荡器, 无需外接元器件
- 所有的指令都可在 1~3 个指令周期内完成
- 查表指令
- 115 条指令
- 8 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器: $8K \times 16$
- 数据存储器: 384×8
- True EEPROM 存储器: 64×8
- 看门狗定时器功能
- 50 个双向 I/O 口
- 4 个与 I/O 口共用的外部中断输入
- 多个定时器模块用于时间测量、捕捉输入、比较匹配输出、PWM 输出或单脉冲输出
- 双时基功能, 用于产生固定时间中断信号
- 10 个外部通道 12-bit 分辨精度的 A/D 转换器
- LCD 显示器
 - ◆ 52-pin LQFP 封装: $32SEG \times 4COM$ & $32SEG \times 8COM$
 - ◆ 48-pin LQFP 封装: $28SEG \times 4COM$ & $28SEG \times 8COM$
 - ◆ 1/3 或 1/4 Bias
- LED 显示器: $8SEG \times 8COM$
- 串行接口模块 – 用于 SPI 或 I²C 通信
- 全双工通用异步接收 / 发送接口 – UART
- 低电压复位功能
- 低电压检测功能
- 封装类型: 48/52-pin LQFP
- Flash 程序存储器烧录可达 100,000 次
- Flash 程序存储器数据可保存 10 年以上
- True EEPROM 数据存储器烧录可达 1,000,000 次
- True EEPROM 数据存储器数据可保存 10 年以上

概述

该单片机是一款 A/D 型具有 8 位高性能精简指令集的 Flash 存储器，专门为直接连接模拟信号的产品而设计的，如那些来自传感器。该单片机具有一系列功能和特性，其 Flash 存储器可多次编程的特性给用户提供了极大的方便。存储器方面，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。

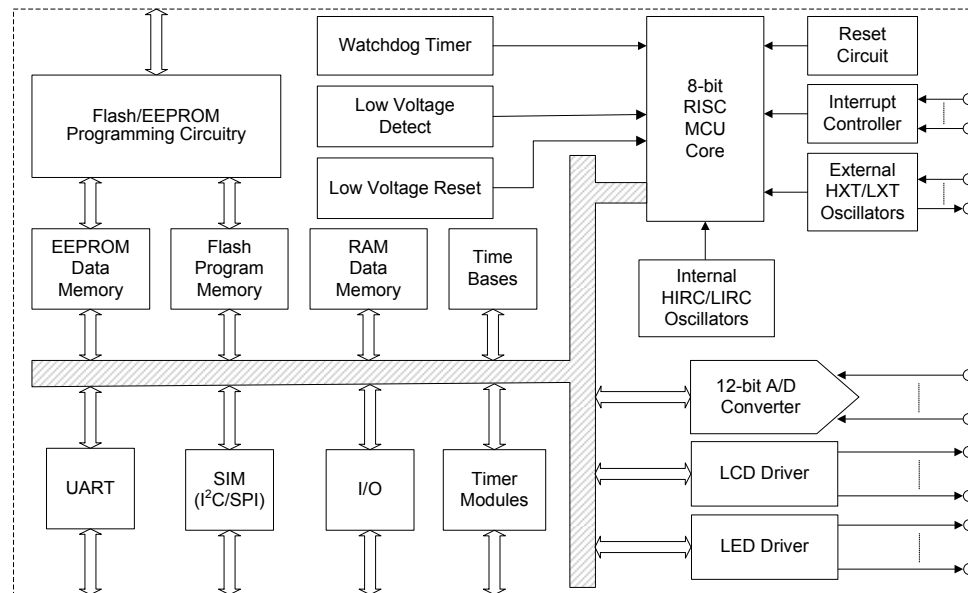
模拟特性包括一个多通道 12-bit A/D 转换器。该单片机带有多个使用灵活的定时器模块，可提供定时功能、脉冲产生功能及 PWM 产生功能。内建完整的 SPI 和 I²C 接口功能，为设计者提供了一个易与外部硬件通信的接口。内部看门狗定时器、低电压复位和低电压检测等内部保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地运行。

该单片机提供了丰富的外部、内部低速和高速振荡器功能选项，且内建两个完整的系统振荡器，无需外围元器件。其在不同工作模式之间动态切换的能力，为用户提供了一个优化单片机操作和减少功耗的手段。

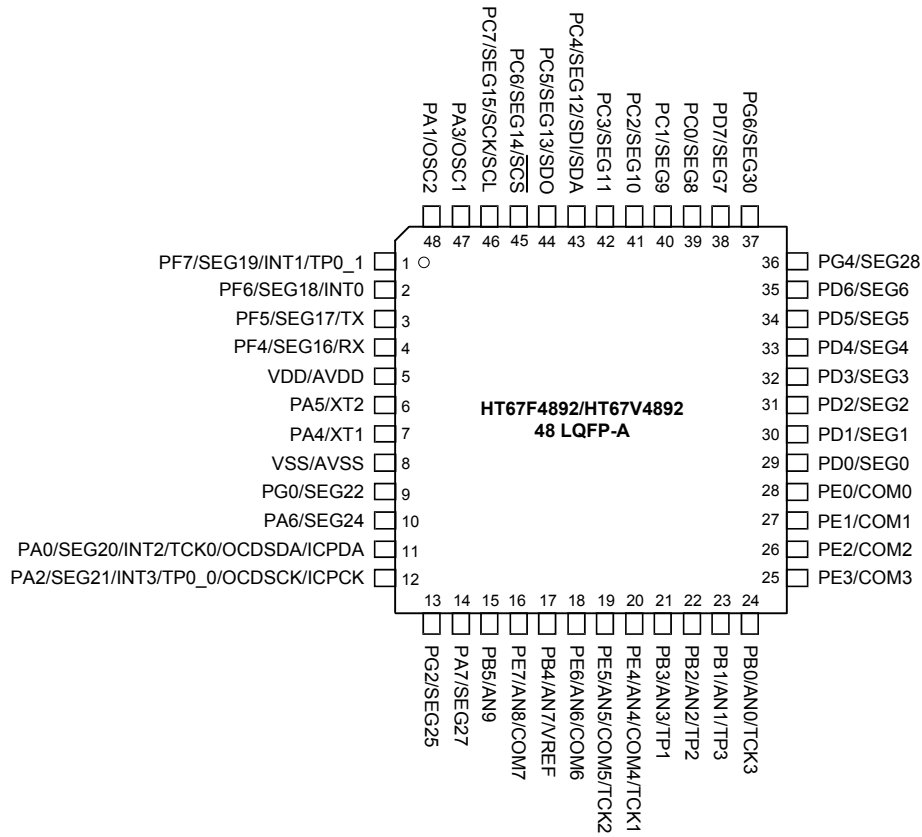
该单片机包含 UART 模块。它可以支持通信应用，如微控制器之间的网络数据通信，个人电脑和外围设备之间的低成本数据链接，便携式和电池供电设备通信等。

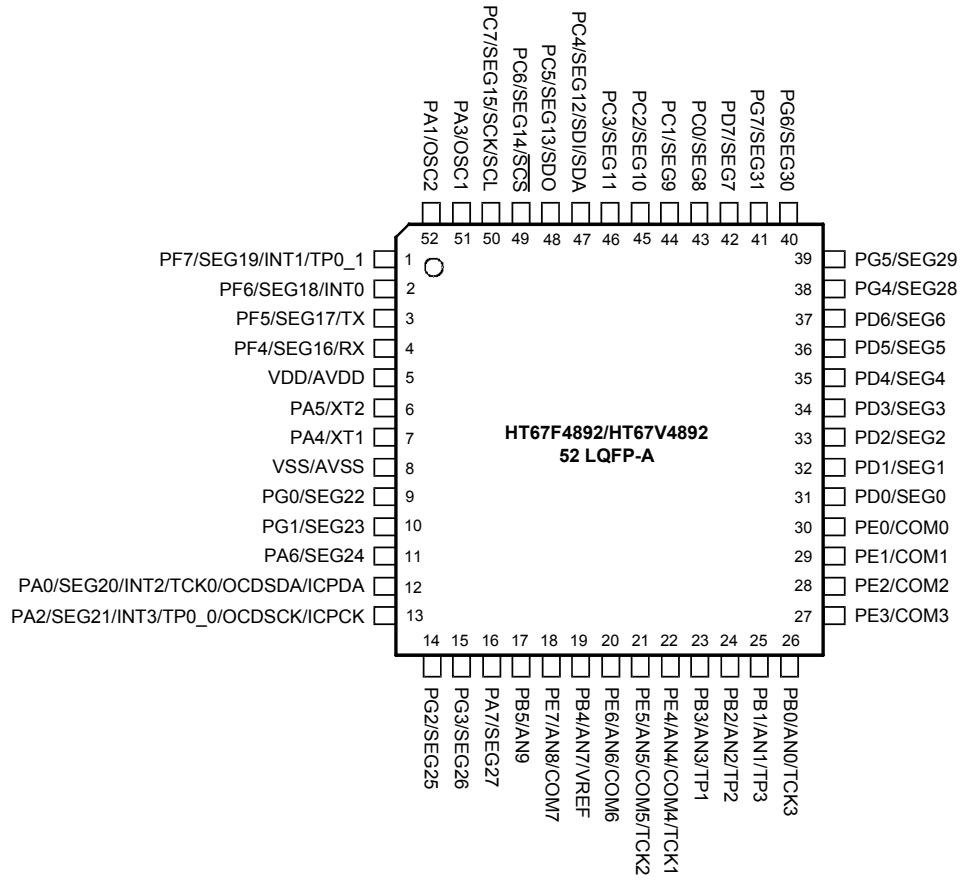
该单片机还具有 LCD 和 LED 驱动器功能，为需要 LCD 和 LED 显示器类型接口的应用提供简易和低成本解决方案。外加 I/O 使用灵活、时基功能等其它特性，使这款单片机实现应用程序，仅需必要的较小外部组件，从而降低组件成本和电路板空间。

方框图



引脚图





- 注：1、若共用引脚同时有多种输出，“/”号右侧的引脚名具有更高的优先级。
2、OCSDA 和 OCDSCK 引脚为 OCDS 专用引脚，仅用于 HT67F4892 的 EV 芯片 HT67V4892。

引脚说明

单片机的所有引脚都以它们的端口名称进行标注，例如 PA0、PA1 等，用于描述这些引脚的数字输入 / 输出功能。然而，这些引脚也与其它功能共用，如 A/D 转换器、定时器模块引脚等。每个引脚的功能如下表所述，而引脚配置的详细内容见规格书其它章节。下述引脚功能表格是针对最大封装提供的引脚，对于小封装会有部分引脚未出现。

引脚名称	功能	OPT	I/T	O/T	说明
PA0/SEG20/ INT2/TCK0/ OCSDA/ ICPDA	PA0	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SEG20	SEGCR2	—	AN	LCD SEG 输出
	INT2	INTEG INTC2	ST	—	外部中断输入 2
	TCK0	—	ST	—	TM0 时钟输入
	OCSDA	—	ST	CMOS	OCDS 地址 / 数据，仅用于 EV 芯片
	ICPDA	—	ST	CMOS	ICP 地址 / 数据
PA1/OSC2	PA1	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	OSC2	CO	—	HXT	HXT 输出引脚
PA2/SEG21/ INT3/TP0_0/ OCDSCK/ ICPCK	PA2	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SEG21	SEGCR2	—	AN	LCD SEG 输出
	INT3	INTEG INTC2	ST	—	外部中断输入 3
	TP0_0	TMPC	ST	CMOS	TM0 输入 / 输出
	OCDSCK	—	ST	—	OCDS 时钟，仅用于 EV 芯片
	ICPCK	—	ST	—	ICP 时钟
PA3/OSC1	PA3	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	OSC1	CO	HXT	—	HXT 输入引脚
PA4/XT1	PA4	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	XT1	FSUBC	LXT	—	LXT 输入引脚
PA5/XT2	PA5	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	XT2	FSUBC	—	LXT	LXT 输出引脚
PA6/SEG24	PA6	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SEG24	SEGCR3	—	AN	LCD SEG 输出
PA7/SEG27	PA7	PAWU PAPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SEG27	SEGCR3	—	AN	LCD SEG 输出
PB0/AN0/ TCK3	PB0	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AN0	ACERL	AN	—	A/D 转换器外部输入 0
	TCK3	—	ST	—	TM3 时钟输入

引脚名称	功能	OPT	I/T	O/T	说明
PB1/AN1/ TP3	PB1	PBPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	AN1	ACERL	AN	—	A/D 转换器外部输入 1
	TP3	TMPC	ST	CMOS	TM3 输入 / 输出
PB2/AN2/ TP2	PB2	PBPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	AN2	ACERL	AN	—	A/D 转换器外部输入 2
	TP2	TMPC	ST	CMOS	TM2 输入 / 输出
PB3/AN3/ TP1	PB3	PBPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	AN3	ACERL	AN	—	A/D 转换器外部输入 3
	TP1	TMPC	ST	CMOS	TM1 输入 / 输出
PB4/AN7/ VREF	PB4	PBPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	AN7	ACERL	AN	—	A/D 转换器外部输入 7
	VREF	ADCR1	AN	—	A/D 转换器参考电压输入引脚
PB5/AN9	PB5	PBPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	AN9	ACERH	AN	—	A/D 转换器外部输入 9
PC0/SEG8	PC0	PCPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG8	SEGCR1	—	AN	LCD SEG 输出
PC1/SEG9	PC1	PCPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG9	SEGCR1	—	AN	LCD SEG 输出
PC2/SEG10	PC2	PCPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG10	SEGCR1	—	AN	LCD SEG 输出
PC3/SEG11	PC3	PCPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG11	SEGCR1	—	AN	LCD SEG 输出
PC4/SEG12/ SDI/SDA	PC4	PCPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG12	SEGCR1	—	AN	LCD SEG 输出
	SDI	SIMC0	ST	—	SPI 串行数据输入
	SDA	SIMC0	ST	NMOS	I ² C 数据线
PC5/SEG13/ SDO	PC5	PCPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG13	SEGCR1	—	AN	LCD SEG 输出
	SDO	SIMC0	—	CMOS	SPI 串行数据输出
PC6/SEG14/ SCS	PC6	PCPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SCS	SIMC0 SIMC2	ST	CMOS	SPI 从机选择
PC7/SEG15/ SCK/SCL	PC7	PCPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG15	SEGCR1	—	AN	LCD SEG 输出
	SCK	SIMC0	ST	CMOS	SPI 串行时钟
	SCL	SIMC0	ST	NMOS	I ² C 时钟线
PD0/SEG0	PD0	PDPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG0	SEGCR0	—	AN	LCD SEG 输出
PD1/SEG1	PD1	PDPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG1	SEGCR0	—	AN	LCD SEG 输出

引脚名称	功能	OPT	I/T	O/T	说明
PD2/SEG2	PD2	PDPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG2	SEGCR0	—	AN	LCD SEG 输出
PD3/SEG3	PD3	PDPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG3	SEGCR0	—	AN	LCD SEG 输出
PD4/SEG4	PD4	PDPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG4	SEGCR0	—	AN	LCD SEG 输出
PD5/SEG5	PD5	PDPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG5	SEGCR0	—	AN	LCD SEG 输出
PD6/SEG6	PD6	PDPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG6	SEGCR0	—	AN	LCD SEG 输出
PD7/SEG7	PD7	PDPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG7	SEGCR0	—	AN	LCD SEG 输出
PE0/COM0	PE0	PEPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	COM0	LCDC0	—	AN	LCD COM 输出
PE1/COM1	PE1	PEPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	COM1	LCDC0	—	AN	LCD COM 输出
PE2/COM2	PE2	PEPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	COM2	LCDC0	—	AN	LCD COM 输出
PE3/COM3	PE3	PEPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	COM3	LCDC0	—	AN	LCD COM 输出
PE4/AN4/ COM4/ TCK1	PE4	PEPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	AN4	ACERL	AN	—	A/D 转换器外部输入 4
	COM4	LCDC0	—	AN	LCD COM 输出
	TCK1	—	ST	—	TM1 时钟输入
PE5/AN5/ COM5/ TCK2	PE5	PEPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	AN5	ACERL	AN	—	A/D 转换器外部输入 5
	COM5	LCDC0	—	AN	LCD COM 输出
	TCK2	—	ST	—	TM2 时钟输入
PE6/AN6/ COM6	PE6	PEPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	AN6	ACERL	AN	—	A/D 转换器外部输入 6
	COM6	LCDC0	—	AN	LCD COM 输出
PE7/AN8/ COM7	PE7	PEPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	AN8	ACERL	AN	—	A/D 转换器外部输入 8
	COM7	LCDC0	—	AN	LCD COM 输出
PF4/SEG16/ RX	PF4	PFPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG16	SEGCR2	—	AN	LCD SEG 输出
	RX	UCR1	ST	—	UART RX 串行数据输入
PF5/SEG17/ TX	PF5	PFPU	ST	CMOS	通用 I/O 口, 可通过寄存器设置上拉电阻
	SEG17	SEGCR2	—	AN	LCD SEG 输出
	TX	UCR1	—	CMOS	UART TX 串行数据输出

引脚名称	功能	OPT	I/T	O/T	说明
PF6/SEG18/ INT0	PF6	PFPUP	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG18	SEGCR2	—	AN	LCD SEG 输出
	INT0	INTEG INTC0	ST	—	外部中断输入 0
PF7/SEG19/ INT1/TP0_1	PF7	PFPUP	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG19	SEGCR2	—	AN	LCD SEG 输出
	INT1	INTEG INTC0	ST	—	外部中断输入 1
	TP0_1	TMPC	ST	CMOS	TM0 输入 / 输出
PG0/SEG22	PG0	PGPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG22	SEGCR2	—	AN	LCD SEG 输出
PG1/SEG23	PG1	PGPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG23	SEGCR2	—	AN	LCD SEG 输出
PG2/SEG25	PG2	PGPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG25	SEGCR3	—	AN	LCD SEG 输出
PG3/SEG26	PG3	PGPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG26	SEGCR3	—	AN	LCD SEG 输出
PG4/SEG28	PG4	PGPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG28	SEGCR3	—	AN	LCD SEG 输出
PG5/SEG29	PG5	PGPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG29	SEGCR3	—	AN	LCD SEG 输出
PG6/SEG30	PG6	PGPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG30	SEGCR3	—	AN	LCD SEG 输出
PG7/SEG31	PG7	PGPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG31	SEGCR3	—	AN	LCD SEG 输出
VDD/AVDD	VDD	—	PWR	—	数字正电源电压
	AVDD	—	PWR	—	模拟正电源电压
VSS/AVSS	VSS	—	PWR	—	数字负电源电压
	AVSS	—	PWR	—	模拟负电源电压

注：I/T：输入类型； O/T：输出类型；
 OPT：通过配置选项 (CO) 或者寄存器选项来配置；
 PWR：电源； ST：斯密特触发输入；
 AN：模拟信号； CMOS：CMOS 输出；
 NMOS：NMOS 输出； HXT：高速晶体振荡器；
 LXT：低速晶体振荡器。

极限参数

电源供应电压	$V_{SS}-0.3V \sim V_{SS}+6.0V$
端口输入电压	$V_{SS}-0.3V \sim V_{DD}+0.3V$
储存温度	$-50^{\circ}C \sim 125^{\circ}C$
工作温度	$-40^{\circ}C \sim 85^{\circ}C$
I_{OH} 总电流	-80mA
I_{OL} 总电流	80mA
总功耗	500mW

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

直流电气特性

$T_a = 25^{\circ}C$

符号	参数	测试条件			最小	典型	最大	单位
		V_{DD}	条件					
V_{DD}	工作电压 (HXT)	—	$f_{SYS} = f_{HXT} = 4MHz$		2.2	—	5.5	V
			$f_{SYS} = f_{HXT} = 8MHz$		—	5.5	5.5	
			$f_{SYS} = f_{HXT} = 12MHz$		—	5.5	5.5	
			$f_{SYS} = f_{HXT} = 16MHz$		—	5.5	5.5	
	工作电压 (HIRC)	—	$f_{SYS} = f_{HIRC} = 8MHz$		2.2	—	5.5	
	工作电压 (LXT)	—	$f_{SYS} = f_{LXT} = 32.768kHz$		2.2	—	5.5	
工作电压 (LIRC)	—	$f_{SYS} = f_{LIRC} = 32kHz$		2.2	—	5.5		

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
I _{DD}	工作电流 (正常模式)	3V	无负载, f _H = 8MHz, ADC off, WDT 使能,	—	1.6	2.4	mA
		5V	f _{SYS} =f _H , f _{SUB} =f _{LXT} 或 f _{LIRC}	—	3.3	5.0	
	工作电流 (HXT)	3V	无负载, f _{SYS} = f _H /2,	—	0.9	1.5	mA
			ADC off, WDT 使能	—	2.5	3.75	
		5V	无负载, f _{SYS} = f _H /4,	—	0.7	1.0	mA
			ADC off, WDT 使能	—	2.0	3.0	
		3V	无负载, f _{SYS} = f _H /8,	—	0.6	0.9	mA
			ADC off, WDT 使能	—	1.6	2.4	
		5V	无负载, f _{SYS} = f _H /16,	—	0.5	0.75	mA
			ADC off, WDT 使能	—	1.5	2.25	
		3V	无负载, f _{SYS} = f _H /32,	—	0.49	0.74	mA
			ADC off, WDT 使能	—	1.45	2.18	
	5V	无负载, f _{SYS} = f _H /64,	—	0.47	0.71	mA	
		ADC off, WDT 使能	—	1.4	2.1		
	工作电流 (低速模式, LXT 或 LIRC)	3V	无负载, f _{SYS} = f _{SUB} = f _{LXT} = 32768Hz,	—	45	75	μA
			ADC off, WDT 使能, LXTLP=0, LVR 使能	—	90	140	
		5V	无负载, f _{SYS} = f _{SUB} = f _{LXT} = 32768Hz,	—	40	70	μA
			ADC off, WDT 使能, LXTLP=1, LVR 使能	—	85	135	
3V		无负载, f _{SYS} = f _{SUB} = f _{LIRC} = 32kHz	—	40	65	μA	
		ADC off, WDT 使能, LVR 使能	—	80	130		

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
I _{STB}	静态电流 (IDLE0 模式, LXT On)	3V	无负载, ADC off,	—	2	4	μA
		5V	WDT 使能, LXTLP=0	—	4	8	
		3V	无负载, ADC off,	—	1.5	3.0	
		5V	WDT 使能, LXTLP=1	—	3.0	6.0	
	静态电流 (IDLE0 模式, LIRC On)	3V	无负载, ADC off,	—	1.5	3.0	μA
		5V	WDT 使能	—	3.0	6.0	
	静态电流 (IDLE0 模式, LXT On)	3V	无负载, ADC off, WDT 使能, LXTLP=1,	—	3.0	6.0	μA
		5V	LCD 使能, (R _T =1170kΩ 无快速充电, V _{LCD} =V _{DD})	—	6.0	12	μA
		3V	无负载, ADC off, WDT 使能, LXTLP=1,	—	14	28	μA
		5V	LCD 使能, (R _T =225kΩ 无快速充电, V _{LCD} =V _{DD})	—	24	48	μA
		3V	无负载, ADC off, WDT 使能, LXTLP=1,	—	5	10	μA
		5V	LCD 使能, (R _T =1170kΩ 快速充电, QCT[2:0]=0, V _{LCD} =V _{DD})	—	9	18	μA
		3V	无负载, ADC off, WDT 使能, LXTLP=1,	—	11	22	μA
		5V	LCD 使能, (R _T =1170kΩ 快速充电, QCT[2:0]=7, V _{LCD} =V _{DD})	—	18	36	μA
静态电流 (IDLE1 模式, LIRC On)	3V	无负载, ADC off,	—	0.5	3.0	mA	
	5V	WDT 使能, f _{sys} =8MHz on	—	1.0	6.0		
静态电流 (SLEEP0 模式, LXT 或 LIRC Off)	3V	无负载, ADC off,	—	0.2	1.0	μA	
	5V	WDT 除能	—	0.4	2.0		
静态电流 (SLEEP1 模式, LXT 或 LIRC On)	3V	无负载, ADC off,	—	1.5	3.0	μA	
	5V	WDT 使能	—	2.5	5.0		
V _{IL}	I/O 口低电平输入电压	—	—	0	—	0.3 V _{DD}	V
V _{IH}	I/O 口高电平输入电压	—	—	0.7 V _{DD}	—	V _{DD}	V
GPIO (除 PD0~PD7 & PE0~PE7 外)							
I _{OL}	I/O 口灌电流	3V	V _{OL} = 0.1V _{DD}	4	8	—	mA
		5V	V _{OL} = 0.1V _{DD}	10	20	—	
I _{OH}	I/O 口源电流	3V	V _{OH} = 0.9V _{DD}	-2	-4	—	mA
		5V	V _{OH} = 0.9V _{DD}	-5	-10	—	
LCD 驱动高速源电流 I/O 口 (PE0~PE7)							
I _{OL}	I/O 口灌电流	3V	V _{OL} = 0.1V _{DD}	8	16	—	mA
		5V	V _{OL} = 0.1V _{DD}	20	40	—	

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
I _{OH}	I/O 口源电流	3V	V _{OH} = 0.9V _{DD}	-2	-4	—	mA
		5V	V _{OH} = 0.9V _{DD}	-5	-10	—	
LCD 驱动可调节源电流 I/O 口 (PD0~PD7)							
I _{OL}	I/O 口灌电流	3V	V _{OL} = 0.1V _{DD}	4	8	—	mA
		5V	V _{OL} = 0.1V _{DD}	10	20	—	
I _{OH}	I/O 口源电流	3V	V _{OH} = 0.9V _{DD} (IOHSn[1:0]=00B, n=0~7)	-2	-4	—	mA
			V _{OH} = 0.9V _{DD} (IOHSn[1:0]=01B, n=0~7)	-0.67	-1.33	—	
			V _{OH} = 0.9V _{DD} (IOHSn[1:0]=10B, n=0~7)	-0.5	-1	—	
			V _{OH} = 0.9V _{DD} (IOHSn[1:0]=11B, n=0~7)	-0.33	-0.66	—	
		5V	V _{OH} = 0.9V _{DD} (IOHSn[1:0]=00B, n=0~7)	-5	-10	—	mA
			V _{OH} = 0.9V _{DD} (IOHSn[1:0]=01B, n=0~7)	-1.67	-3.33	—	
			V _{OH} = 0.9V _{DD} (IOHSn[1:0]=10B, n=0~7)	-1.25	-2.5	—	
			V _{OH} = 0.9V _{DD} (IOHSn[1:0]=11B, n=0~7)	-0.83	-1.67	—	
R _{PH}	I/O 口上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
R _T	LCD 总偏压电阻	3V/5V	—	-30	R _T	+30	%
I _{TOL}	总 I/O 口灌电流	5V	—	80	—	—	mA
I _{TOH}	总 I/O 口源电流	5V	—	-80	—	—	mA

交流电气特性

Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
f _{CPU}	工作时钟	—	2.2~5.5V	DC	—	4	MHz
			2.2~5.5V	DC	—	8	
			2.7~5.5V	DC	—	12	
			4.5~5.5V	DC	—	16	
f _{SYS}	系统时钟 (HIRC)	2.2V~5.5V	—	—	8	—	MHz
f _{HIRC}	通过烧录器调整后的 8MHz HIRC 频率	3V/5V	Ta = 25°C	-1%	8	+1%	MHz
			Ta = -40~85°C	-2%	8	+2%	
		2.2V~5.5V	Ta = 25°C	-2.5%	8	+2.5%	
			Ta = -40°C~85°C	-3%	8	+3%	
f _{LIRC}	系统时钟 (LIRC)	5V	Ta = 25°C	-10%	32	+10%	kHz
t _{TIMER}	TCKn 输入脚最小脉宽	—	—	0.3	—	—	μs
t _{INT}	外部中断最小脉宽	—	—	10	—	—	μs
t _{EERD}	写周期时间	5V	—	—	2	4	t _{SYS}
t _{EEWR}	读周期时间	5V	—	—	4	6	ms
t _{RSTD}	系统复位延迟时间 (POR 复位, LVR 硬件 复位, LVR/WDT 软件 复位)	—	—	25	50	100	ms
	系统复位延迟时间 (WDT 溢出复位)	—	—	8.3	16.7	33.3	ms
t _{SST}	系统启动时间 (上电复位)	—	f _{SYS} = f _{HIRC}	1024	—	—	t _{HIRC}
		—	f _{SYS} = f _{HXT}	1024	—	—	t _{HXT}
	系统启动时间 (从 f _{SYS} off 的状态下唤醒)	—	f _{SYS} = f _{LXT}	1024	—	—	t _{LXT}
		—	f _{SYS} = f _{HXT} ~f _{HXT} / 64	1024	—	—	t _{HXT}
		—	f _{SYS} = f _{HIRC} ~f _{HIRC} / 64	16	—	—	t _{HIRC}
		—	f _{SYS} = 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 _{HXT} 或 f _{HIRC}	2	—	—	f _H
—		f _{SYS} = f _{LXT} 或 f _{LIRC}	2	—	—	f _{SUB}	
t _{SRESET}	软件复位最小延迟脉宽	—	—	45	90	120	μs

注: 1. t_{SYS} = 1/f_{SYS}。

2. 为了保持内部 HIRC 晶体振荡器的频率精度, 0.1μF 去耦电容器应连接在 VDD 和 VSS 并且尽可能地靠近单片机。

A/D 转换器电气特性

Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
AV _{DD}	A/D 转换器工作电压	—	—	2.7	—	5.5	V
V _{ADI}	A/D 转换器输入电压	—	—	0	—	AV _{DD} / V _{REF}	V
V _{REF}	A/D 转换器参考电压	—	—	2	—	AV _{DD}	V
DNL	非线性微分误差	3V	V _{REF} = V _{DD} , t _{ADCK} = 0.5μs	-4	—	+4	LSB
		5V	V _{REF} = V _{DD} , t _{ADCK} = 0.5μs				
		3V	V _{REF} = V _{DD} , t _{ADCK} = 10μs				
		5V	V _{REF} = V _{DD} , t _{ADCK} = 10μs				
INL	非线性积分误差	3V	V _{REF} = V _{DD} , t _{ADCK} = 0.5μs	-7	—	+7	LSB
		5V	V _{REF} = V _{DD} , t _{ADCK} = 0.5μs				
		3V	V _{REF} = V _{DD} , t _{ADCK} = 10μs				
		5V	V _{REF} = V _{DD} , t _{ADCK} = 10μs				
I _{ADC}	A/D 转换器使能的额外电流	3V	无负载 (t _{ADCK} = 0.5μs)	—	0.9	1.35	mA
		5V	无负载 (t _{ADCK} = 0.5μs)	—	1.2	1.8	
t _{ADCK}	A/D 转换器时钟周期	—	—	0.5	—	10	μs
t _{ON2ST}	A/D 转换器 On-to-Start 时间	—	—	2	—	—	μs
t _{ADS}	A/D 采样时间	—	—	—	4	—	t _{ADCK}
t _{ADC}	A/D 转换时间 (包括采样和保持时间)	—	12-bit A/D 转换器	—	16	—	t _{ADCK}

参考电压电气特性

Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{BG}	带缓冲器的参考电压	—	—	-3%	1.09	+3%	V
I _{BG}	带缓冲器的参考电压使能的额外功耗	—	—	—	200	300	μA
t _{BGS}	V _{BG} 开启稳定时间	—	—	—	—	200	μs

 注: V_{BG} 电压可作为 A/D 转换器的内部信号输入。

LVD/LVR 电气特性

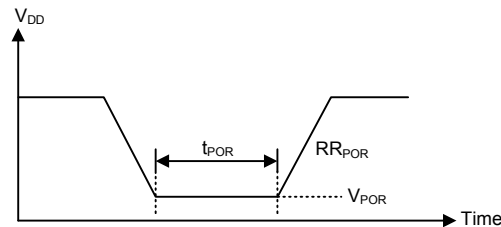
Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{LVR}	低电压复位电压	—	LVR 使能, 电压选择 2.1V	-5%	2.1	+5%	V
			LVR 使能, 电压选择 2.55V		2.55		
			LVR 使能, 电压选择 3.15V		3.15		
			LVR 使能, 电压选择 3.8V		3.8		
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		
I _{LVD}	使能 LVD 的额外功耗	3V	LVD 除能 → LVD 使能 (LVR 使能)	—	30	45	μA
		5V		—	60	90	
t _{LVDS}	LVDO 稳定时间	—	LVR 使能, LVD off → on	—	—	15	μs
t _{LVR}	LVR 复位最小低电压脉宽	—	—	120	240	480	μs
t _{LVD}	LVD 中断最小低电压脉宽	—	—	60	120	240	μs

上电复位特性

Ta = 25°C

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



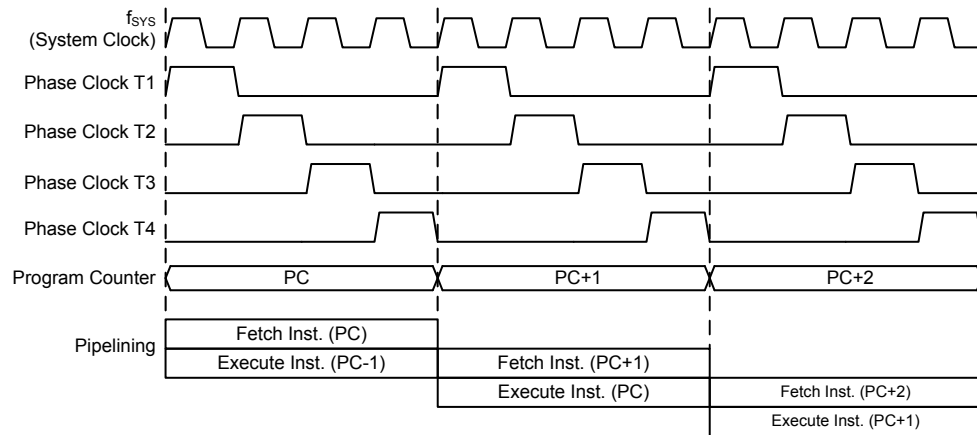
系统结构

内部系统结构是盛群单片机具有良好性能的主要因素。由于采用 RISC 结构该单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的获取和执行同时进行，此举使得除了跳转和调用指令需多一个指令周期外，其它大部分标准指令或扩展指令都能分别在一个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、移位元、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中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供具有最大可靠度和灵活性的 I/O 与 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。

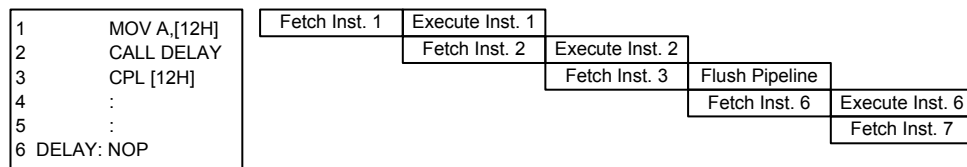
时序和流水线结构

主系统时钟由 HXT, LXT、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。

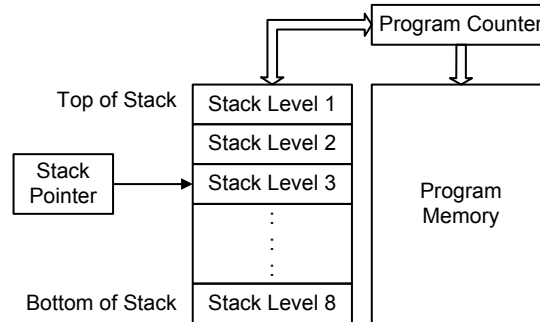
程序计数器	
高字节	PCL 寄存器
PC12~PC8	PCL7~PCL0

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 寄存器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。该单片机含有 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：

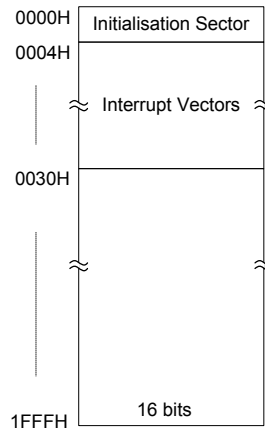
- 算术运算：
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM,
LDAA
- 逻辑运算：
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- 移位运算：
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
LRRR, LRR, LRRCA, LRRR, LRLA, LRL, LRLCA, LRLC
- 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,
LSNZ, LSZ, LSZA, LSIZ, LSIZA, LSDZ, LSDZA

Flash 程序存储器

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

结构

程序存储器的容量为 $8K \times 16$ 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

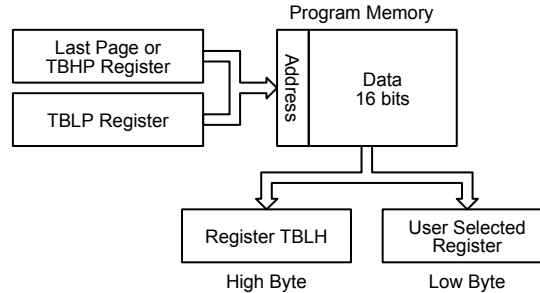
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 0000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

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

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

表格读取程序范例

```

tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
mov a,06h          ; initialise low table pointer - note that this address
                  ; is referenced
mov tblp,a         ; 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

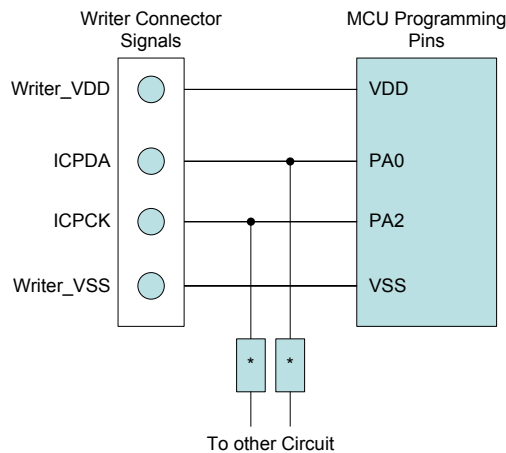
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。除此之外，Holtek 单片机提供 4 线接口的在线烧录方式。用户可将进行过烧录或未经过烧录的单片机芯片连同电路板一起制成，最后阶段进行程序的更新和程序的烧写，在无需去除或重新插入芯片的情况下方便地保持程序为最新版。

Flash MCU 与烧录器引脚对应表如下：

烧录器引脚名称	MCU 在线烧录引脚名称	引脚功能
ICPDA	PA0	烧录串行数据 / 地址
ICPCK	PA2	烧录时钟
VDD	VDD	电源供电
VSS	VSS	地

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

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



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

片上调试 – OCDS

EV 芯片 HT67V4892 用于 HT67F4892 单片机仿真。此 EV 芯片提供片上调试功能 (OCDS – On-Chip Debug) 用于开发过程中的单片机调试。除了片上调试功能方面，EV 芯片和实际单片机在功能上几乎是兼容的。用户可将 OCDSDA 和 OCDSCK 引脚连接至 Holtek HT-IDE 开发工具，从而实现 EV 芯片对实际单片机的仿真。OCDSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户用 EV 芯片进行调试时，实际单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。关于 OCDS 功能的详细描述，请参考“Holtek e-Link for 8-bit MCU OCDS 使用手册”文件。

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

RAM 数据存储器的

RAM 数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。数据存储器分为三个部分，第一部分是 RAM 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 RAM 通用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。第三部分是为 LCD 存储器保留的。这个特殊数据存储器的地址直接映射到 LCD 显示器，写入这部分存储器的数据将直接影响显示的数据。

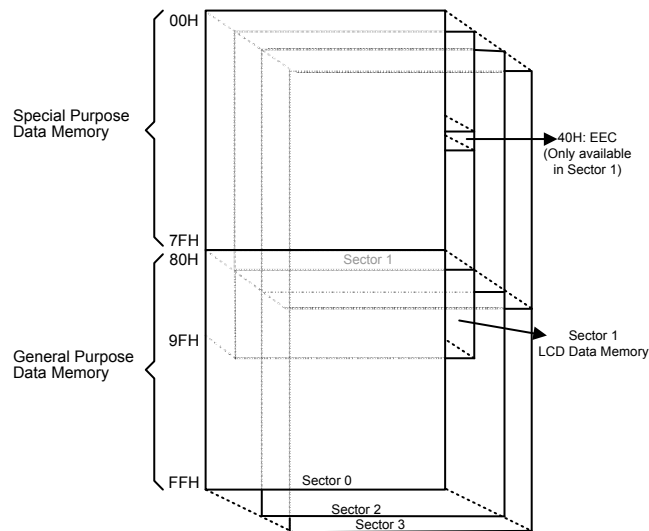
结构

数据存储器被分为多个 Sector，都是 8-bit 存储器。每个 Sector 被分为两种类型，即特殊功能数据存储器 and 通用数据存储器。而 Sector 1 的 80H~9FH 为 LCD 存储器。

单片机数据存储器的起始地址为“00H”。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。

特殊功能数据存储器	LED 存储器		通用数据存储器	
有效 Sector	容量	Sector: 地址	容量	Sector: 地址
0, 1, 2, 3	32×8	1: 80H~9FH	384×8	0: 80H~FFH 2: 80H~FFH 3: 80H~FFH

数据存储器概要



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使用间接寻址访问方式时，通过 MP1L 或 MP2L 寄存器指定。

直接寻址可用于所有 Sector，通过相应的指令可以寻址所有可用的数据存储器空间。所访问的数据存储器位于除 Sector 0 外的任何数据存储器 Sector，扩展指令可代替间接寻址方式用来访问数据存储器。标准指令和扩展指令的主要区别在于扩展指令中的数据存储器地址“m”由 10 个有效位组成，高字节表示 Sector，低字节表示指定的地址。

通用数据存储器

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

特殊功能数据存储器

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

Sector 0,2,3		Sector 1	Sector 0,2,3		Sector 1
00H	IAR0		40H	EEC	
01H	MP0		41H	USR	
02H	IAR1		42H	UCR1	
03H	MP1L		43H	UCR2	
04H	MP1H		44H	BRG	
05H	ACC		45H	TXR/RXR	
06H	PCL		46H		
07H	TBLP		47H	TMPC	
08H	TBLH		48H	TM2C0	
09H	TBHP		49H	TM2C1	
0AH	STATUS		4AH	TM2DL	
0BH			4BH	TM2DH	
0CH	IAR2		4CH	TM2AL	
0DH	MP2L		4DH	TM2AH	
0EH	MP2H		4EH	TM3C0	
0FH	SMOD		4FH	TM3C1	
10H	TBC		50H	TM3DL	
11H	WDTA		51H	TM3DH	
12H	LVDC		52H	TM3AL	
13H	LVRC		53H	TM3AH	
14H	CTRL		54H		
15H	FSUBC		55H		
16H	INTEG		56H		
17H	INTC0		57H		
18H	INTC1		58H	LCDC0	
19H	INTC2		59H	LCDC1	
1AH	MFI0		5AH	SEGCR0	
1BH	MFI1		5BH	SEGCR1	
1CH	MFI2		5CH	SEGCR2	
1DH	MFI3		5DH	SEGCR3	
1EH	PAWU		5EH	PCPU	
1FH	PAPU		5FH	PC	
20H	PA		60H	PCC	
21H	PAC		61H	PDPU	
22H	PBPU		62H	PD	
23H	PB		63H	PDC	
24H	PBC		64H	PEPU	
25H	IOHR0		65H	PE	
26H	IOHR1		66H	PEC	
27H	MFI4		67H	PFPU	
28H	ADRL		68H	PF	
29H	ADRH		69H	PFC	
2AH	ADCR0		6AH	PGPU	
2BH	ADCR1		6BH	PG	
2CH	ACERL		6CH	PGC	
2DH	ACERH		6DH	SIMC0	
2EH	TM0C0		6EH	SIMC1	
2FH	TM0C1		6FH	SIMD	
30H	TM0DL		70H	SIMA/SIMC2	
31H	TM0DH		71H	SIMTOC	
32H	TM0AL		72H		
33H	TM0AH		73H		
34H	TM0RPL		74H		
35H	TM0RPH		75H		
36H	TM1C0		76H		
37H	TM1C1		77H		
38H	TM1DL		78H		
39H	TM1DH		79H		
3AH	TM1AL		7AH		
3BH	TM1AH		7BH		
3CH	INTC3		7CH		
3DH	EEA		7DH		
3EH	EED		7EH		
3FH			7FH		

□ : Unused, read as 00H ■ : Reserved, cannot be changed

特殊功能数据存储结构

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 和 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 以及 IAR2 和 MP2L/MP2H 可以访问所有 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。当对相关间接寻址寄存器进行任何操作时，单片机指向的实际地址是由存储器指针 MP0 所指定的地址，此时间接寻址寄存器 IAR0 用于访问 Sector0 中的数据，而 MP1L/MP1H 和 IAR1、MP2L/MP2H 和 IAR2 可根据 MP1H 或 MP2H 寄存器访问所有的 Sector。直接寻址通过相关的数据存储器寻址指令来访问所有的数据 Sector。

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 mplh, 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 MPL
    inc mp1l           ; increment memory pointer MPL
    sdz block          ; check if last memory location has been cleared
    jmp loop
continue:

```

需注意，范例中并没有确定 RAM 地址。

使用扩展指令直接寻址程序举例

```

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

```

注：“m”是位于任何数据存储单元 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单元，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

该 8-bit 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 TO 和 PDF 标志外，状态寄存器中的位像其它大部分寄存器一样可以被改变。任何数据写入到状态寄存器将不会改变 TO 或 PDF 标志位。另外，执行不同的指令后，与状态寄存器有关的运算可能会得到不同的结果。TO 标志位只会受系统上电、看门狗溢出或执行“CLR WDT”或“HALT”指令影响。PDF 标志位只会受执行“HALT”或“CLR WDT”指令或系统上电影响。

Z、OV、AC、C、SC 和 CZ 标志位通常反映最近运算的状态。

- C: 当加法运算的结果产生进位，或减法运算的结果没有产生借位时，则 C 被置位，否则 C 被清零，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，A 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 也受循环移位指令的影响。

EEPROM 数据存储

此单片机的一个特性是内建 EEPROM 数据存储。 “Electrically Erasable Programmable Read Only Memory” 为电可擦可编程只读存储器，由于其非易失的存储结构，即使在电源掉电的情况下存储器内的数据仍然保存完好。这种存储区扩展了 ROM 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

EEPROM 数据存储容量为 64×8。由于映射方式与程序存储器和数据存储器不同，因此不能像其它类型的存储器一样寻址。使用 Sector 0 中的一个地址和数据寄存器以及 Sector 1 中的一个控制寄存器，可以实现对 EEPROM 的单字节读写操作。

EEPROM 寄存器

有三个寄存器控制内部 EEPROM 数据存储总的操作，地址寄存器 EEA、数据寄存器 EED 及控制寄存器 EEC。EEA 和 EED 位于 Sector 0 中，它们能像其它特殊功能寄存器一样直接被访问。EEC 位于 Sector 1 中，不能被直接访问，仅能通过 MP1L/MP1H 和 IAR1 或 MP2L/MP2H 和 IAR2 进行间接读取或写入。由于 EEC 控制寄存器位于 Sector 1 中的“40H”，在 EEC 寄存器上的任何操作被执行前，MP1L 或 MP2L 必须先设为“40H”，MP1H 或 MP2H 被设为“01H”。

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

EEPROM 寄存器列表

EEA 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	D5	D4	D3	D2	D1	D0
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”

Bit5~0 **EEA5~EEA0**: 数据 EEPROM 地址 bit 5~bit 0

EED 寄存器

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

Bit 7~0 **D7~D0**: 数据 EEPROM 数据 bit 7~bit 0

EEC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	WREN	WR	RDEN	RD
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”

Bit 3 **WREN**: 数据 EEPROM 写使能位

0: 除能

1: 使能

此位为数据 EEPROM 写使能位，向数据 EEPROM 写操作之前需将此位置高。将此位清零时，则禁止向数据 EEPROM 写操作。

Bit 2 **WR**: EEPROM 写控制位

0: 写周期结束

1: 写周期有效

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

Bit 1 **RDEN**: 数据 EEPROM 读使能位

0: 除能

1: 使能

此位为数据 EEPROM 读使能位，向数据 EEPROM 读操作之前需将此位置高。将此位清零时，则禁止向数据 EEPROM 读操作。

Bit 0 **RD**: EEPROM 读控制位

0: 读周期结束

1: 读周期有效

此位为数据 EEPROM 读控制位，由应用程序将此位置高将激活读周期。读周期结束后，硬件自动将此位清零。当 RDEN 未首先置高时，此位置高无效。

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

从 EEPROM 中读取数据

从 EEPROM 中读取数据，EEC 寄存器中的读使能位 RDEN 先置高以启用读功能，EEPROM 中读取数据的地址要先放入 EEA 寄存器中。若 EEC 寄存器中的 RD 位被置高，一个读周期将开始。若 RD 位已置高而 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 先置高以启用写功能，然后 EEC 寄存器中的 WR 位需立即置高以开始写操作，这两条指令必须连续执行。总中断位 EMI 在写周期开始前应当被清零，写周期开始后再将其使能。应注意若 WR 位已置高而 WREN 位还未被设置则不能开始写操作。由于控制 EEPROM 写周期是一个内部时钟，与单片机的系统时钟异步，所以数据写入 EEPROM 的时间将有所延迟。可通过轮询 EEC 寄存器中的 WR 位或判断 EEPROM 写中断以侦测写周期是否完成。若写周期完成，WR 位将自动清除为“0”，通知用户数据已写入 EEPROM。因此，应用程序将轮询 WR 位以确定写周期是否结束。

写保护

防止误写入的写保护有以下几种。单片机上电后控制寄存器中的写使能位将被清除以杜绝任何写入操作。上电后存储器指针高字节寄存器 MP1H 或 MP2H 将重置为“0”，这意味着数据存储区 Sector 0 被选中。由于 EEPROM 控制寄存器位于 Sector 1 中，这增加了对写操作的保护措施。在正常程序操作中确保控制寄存器中的写使能位被清除将能防止不正确的写操作。

EEPROM 中断

EEPROM 写周期结束后将产生 EEPROM 写中断，需先通过设置相关中断寄存器的 DEE 位使能 EEPROM 中断。由于 EEPROM 中断包含在多功能中断中，相应的多功能中断使能位需被设置。当 EEPROM 写周期结束，DEF 请求标志位及其相关多功能中断请求标志位将被置位。若总中断、EEPROM 中断和多功能中断使能且堆栈未满的情况下将跳转到相应的多功能中断向量中执行。当中断被响应，只有多功能中断标志位将自动复位，而 EEPROM 中断标志将通过应用程序手动复位。更多细节将在中断章节讲述。

编程注意事项

必须注意的是数据不会无意写入 EEPROM。在没有写动作时写使能位被正常清零可以增强保护功能。存储器指针高字节寄存器也可以正常清零以阻止进入 EEPROM 控制寄存器存在的 Sector 1。也可将 MP1H 或 MP2H 正常清零从而避免访问到 EEPROM 控制寄存器所在的 Sector 1。尽管没有必要，写一个简单的读回程序以检查新写入的数据是否正确还是应该考虑的。

WREN 位置位后，EEC 寄存器中的 WR 位需立即置位，以确保写周期正确地执行。写周期执行前总中断位 EMI 应先清零，写周期开始执行后再将此位重新使能。注意，单片机不应在 EEPROM 读或写操作完全完成之前进入空闲或休眠模式，否则 EEPROM 读或写操作将失败。

程序范例

● 从 EEPROM 中读取数据 – 轮询法

```
MOV A, EEPROM_ADRES           ; user defined address
MOV EEA, A
MOV A, 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 write
CLR MP1H
MOV A, EED                    ; move read data to register
MOV READ_DATA, A
```

● 写数据到 EEPROM – 轮询法

```
MOV A, EEPROM_ADRES           ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA            ; user defined data
MOV EED, A
MOV A, 040H                   ; setup memory pointer low byte MP1L
MOV MP1L, A                   ; MP1L points to EEC register
MOV A, 01H                    ; setup Memory Pointer high byte MP1H
MOV MP1H, A
CLR EMI
SET IAR1.3                    ; set WREN bit, enable write operations
SET IAR1.2                    ; start Write Cycle - set WR bit
SET EMI
BACK:
SZ IAR1.2                     ; check for write cycle end
JMP BACK
CLR IAR1                      ; disable EEPROM write
CLR MP1H
```

振荡器

不同的振荡器选择可以让使用者在不同的应用需求中实现更大范围的功能。振荡器的灵活性使得在速度和功耗方面可以达到最优化。通过配置选项和相关寄存器选择振荡器及其操作。

振荡器概述

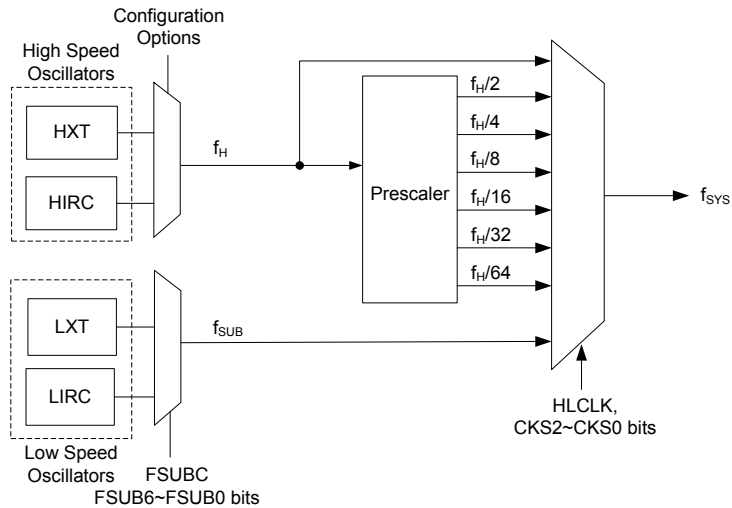
振荡器除了作为系统时钟源，还作为看门狗定时器和时基功能的时钟源。外部振荡器需要一些外围器件，而集成的内部振荡器不需要任何外围器件。它们提供的高速和低速系统振荡器具有较宽的频率范围。通过配置相关控制寄存器可选择所需振荡器。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

类型	名称	频率	引脚
外部高速晶体振荡器	HXT	400kHz~16MHz	OSC1/OSC2
内部高速 RC	HIRC	8MHz	—
外部低速晶体振荡器	LXT	32.768kHz	XT1/XT2
内部低速 RC	LIRC	32kHz	—

振荡器类型

系统时钟配置

该单片机有四个系统振荡器，两个高速振荡器和两个低速振荡器。高速振荡器有外部晶体 / 陶瓷振荡器 HXT 和内部 8MHz RC 振荡器 HIRC。低速振荡器有内部 32kHz RC 振荡器 LIRC 和外部 32.768kHz 振荡器 LXT。使用高速或低速振荡器作为系统时钟的选择是通过设置 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位决定的，系统时钟可动态选择。请注意，两个振荡器必须做出选择，即一个高速和一个低速振荡器。

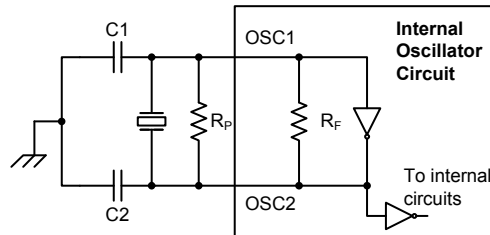


系统时钟配置

外部晶体 / 陶瓷振荡器 – HXT

对于外部晶体 / 陶瓷振荡器，只要简单地将晶体连接至 OSC1 和 OSC2，则会产生振荡所需的相移及反馈，而不需其它外部器件。为保证某些低频率的晶体振荡和陶瓷谐振器的振荡频率更精准，建议连接两个小容量电容 C1 和 C2 到 VSS，具体数值与客户选择的晶振或谐振器有关。此外，有一个配置选项根据振荡器的频率是否为高来设置，等于或高于 1MHz 的频率定义为高，低于 1MHz 频率的定义为低。

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



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

晶体 / 陶瓷振荡器 – HXT

HXT 晶体振荡器 C1 和 C2 值		
晶体频率	C1	C2
12MHz	0 pF	0 pF
8MHz	0 pF	0 pF
4MHz	0 pF	0 pF
1MHz	100 pF	100 pF
455kHz ^(注 2)	100 pF	100 pF

注：1. C1 和 C2 值仅供参考。
2. HXT 模式的配置选项：455kHz。

晶体振荡器电容推荐值

内部 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，不需其它外部器件。内部 RC 振荡器的固定频率为 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减至最低程度。注意如果选择了该内部系统时钟，无需额外的引脚，I/O 引脚可作为普通 I/O 口或其它共用功能使用。

外部 32.768kHz 晶体振荡器 – LXT

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

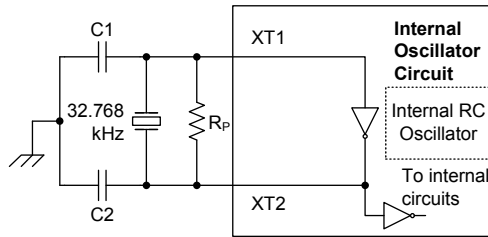
当系统进入空闲 / 休眠模式，系统时钟关闭以降低功耗。然而在某些应用，比如空闲 / 休眠模式下要保持内部定时器功能，必须提供额外的时钟，且与系统时钟无关。

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

FSUBC 寄存器决定 XT1/XT2 脚是用于 LXT 还是作为普通 I/O 口或其它共用功能使用。

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

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



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

外部 LXT 振荡器

LXT 晶体振荡器 C1 和 C2 值		
晶体频率	C1	C2
32.768kHz	10 pF	10 pF
注: 1. C1 和 C2 值仅供参考。 2. 建议 $R_p = 5M\Omega \sim 10M\Omega$ 。		

32.768kHz 晶体振荡器电容推荐值

LXT 振荡器低功耗功能

LXT 振荡器可以工作在快速启动模式或低功耗模式，可通过设置 FSUBC 寄存器中的 LXTLP 位进行模式选择。

● FSUBC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	LXTLP	FSUB6	FSUB5	FSUB4	FSUB3	FSUB2	FSUB1	FSUB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	1	0	1	0

Bit 7 **LXTLP**: LXT 低功耗控制位

- 0: 快速启动模式
- 1: 低功耗模式

Bit 6~0 **FSUB6~FSUB0**: f_{SUB} 时钟源选择位

- 0101010: LIRC
- 1010101: LXT
- 其它值: MCU 复位

系统上电时会清零 LXTLP 位来快速启动 LXT 振荡器。在快速启动模式，LXT 振荡器将起振并快速稳定下来。LXT 振荡器完全起振后，可以通过设置 LXTLP 位为高进入低功耗模式。振荡器可以继续运行，其间耗电将少于快速启动模式。在功耗敏感的应用领域如电池应用方面，功耗必须限制为一个最小值。为了降低功耗，建议系统上电 2 秒后，在应用程序中将 LXTLP 位设为“1”。

应注意的是，无论 LXTLP 位是什么值，LXT 振荡器会一直运作，不同的只是在低功耗模式时启动时间更长。

内部 32kHz 振荡器 – LIRC

内部 32kHz 系统振荡器是一个低频振荡器，由 FSUBC 寄存器进行选择。LIRC 为一个完全集成的 RC 振荡器，它在 5V 电压下运行的典型频率值为 32kHz 且无需外部元件。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响减至最低。因此，在电源为 5V 且温度为 25°C 时，固定的 32kHz 振荡频率容限将达到 10%。

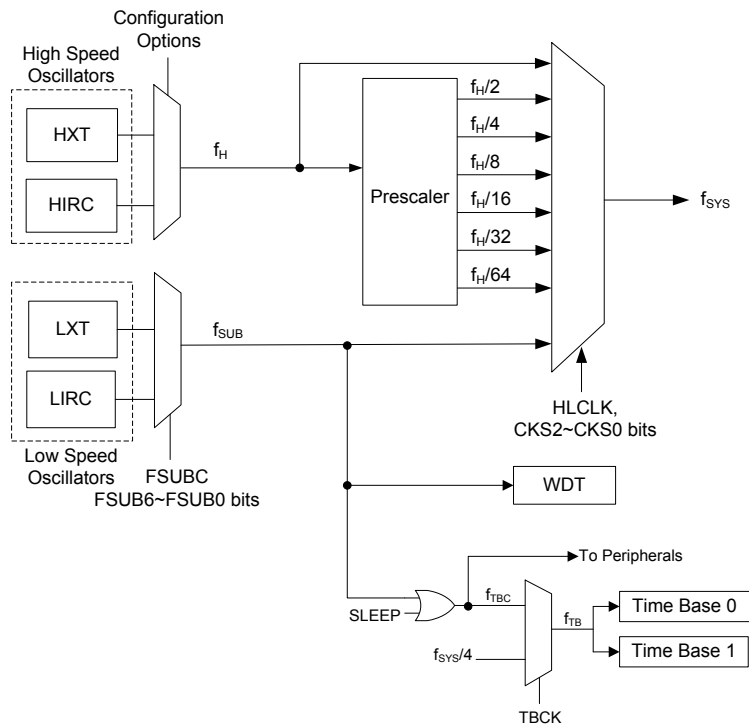
工作模式和系统时钟

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

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。用户使用配置选项和寄存器编程可获取多种时钟，进而使系统时钟获取最大的应用性能。

主系统时钟可来自高频时钟源 f_H 或低频时钟源 f_{SUB} ，通过 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位进行选择。高频时钟来自 HXT 或 HIRC 振荡器，可通过配置选项选择。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 或 LXT 振荡器，可通过 FSUBC 寄存器的 FSUB6~FSUB0 位进行选择。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 到 f_{SUB} 转换时，高速振荡器将停止，因此无 $f_H \sim f_H/64$ 频率可提供给外围电路使用。

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：正常模式和低速模式。剩余的四种工作模式：休眠模式 0、休眠模式 1、空闲模式 0 和空闲模式 1 用于单片机 CPU 关闭时以节省耗电。

工作模式	描述			
	CPU	f_{SYS}	f_{SUB}	f_{TBC}
正常模式	On	$f_H \sim f_H/64$	On	On
低速模式	On	f_{SUB}	On	On
空闲模式 0	Off	Off	On	On
空闲模式 1	Off	On	On	On
休眠模式 0	Off	Off	Off	Off
休眠模式 1	Off	Off	On	Off

正常模式

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

低速模式

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

休眠模式 0

在 HALT 指令执行后且 SMOD 寄存器中 IDLEN 位为低时，系统进入休眠模式。在休眠模式 0 中，CPU 及 f_{SUB} 停止运行，看门狗定时器功能除能。在该模式中 LVDEN 位需置为“0”，否则将不能进入休眠模式 0 中。

休眠模式 1

在 HALT 指令执行后且 SMOD 寄存器中 IDLEN 位为低时，系统进入休眠模式。在休眠模式 1 中，CPU 停止运行。然而若 LVDEN 位为“1”或看门狗定时器功能使能， f_{SUB} 继续运行。

空闲模式 0

执行 HALT 指令后且 SMOD 寄存器中 IDLEN 位为高，CTRL 寄存器中 FSYSON 位为低时，系统进入空闲模式 0。在空闲模式 0 中，CPU 停止，系统振荡器停止，但低速时钟 f_{SUB} 开启。

空闲模式 1

执行 HALT 指令后且 SMOD 寄存器中 IDLEN 位为高，CTRL 寄存器中 FSYSON 位为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，系统振荡器继续运行，该系统振荡器可以为高速或低速系统振荡器。在该模式中，低速时钟 f_{SUB} 开启。

注：如果 LVDEN = 1 且该单片机进入休眠 / 空闲模式，LVD 和 bandgap 功能不会被除能， f_{SUB} 将强制使能。在休眠模式，如果使能休眠模式 1，则除 WDT 和 LVD 外的外围功能都将除能。

控制寄存器

寄存器 SMOD，用于控制单片机内部时钟。

SMOD 寄存器

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

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

000: f_{SUB} (f_{LXT} 或 f_{LIRC})

001: f_{SUB} (f_{LXT} 或 f_{LIRC})

010: $f_H/64$

011: $f_H/32$

100: $f_H/16$

101: $f_H/8$

110: $f_H/4$

111: $f_H/2$

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

Bit 4 未定义，读为 “0”

Bit 3 **LTO**: 低速振荡器就绪标志位

0: 未就绪

1: 就绪

此位为低速系统振荡器就绪标志位，用于表明低速系统振荡器在系统上电复位或经唤醒后何时稳定下来。当系统处于 SLEEP0 模式时，该标志为低。系统时钟来自 LXT 振荡器，系统唤醒后该位转换为高需 1024 个时钟周期；若系统时钟来自 LIRC 振荡器，该位转换为高需 1~2 个时钟周期。

Bit 2 **HTO**: 高速振荡器就绪标志位

0: 未就绪

1: 就绪

此位为高速系统振荡器就绪标志位，用于表明高速系统振荡器何时稳定下来。此标志在系统上电后经硬件清零，高速系统振荡器稳定后变为高电平。

因此，此位在单片机上电后由应用程序读取的总是为 “1”。该标志由休眠模式或空闲模式 0 中唤醒后会处于低电平状态，若使用 HIRC 或 HXT 振荡器，该位将在一定时间后变为高电平状态。

Bit 1 **IDLEN**: 空闲模式控制位

0: 除能

1: 使能

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

Bit 0 **HLCLK**: 系统时钟选择位

0: $f_H/2 \sim f_H/64$ 或 f_{SUB}

1: f_H

此位用于选择 f_H 或 $f_H/2 \sim f_H/64$ 还是 f_{SUB} 作为系统时钟。该位为高时选择 f_H 作为系统时钟，为低时则选择 $f_H/2 \sim f_H/64$ 或 f_{SUB} 作为系统时钟。当系统时钟由 f_H 时钟向 f_{SUB} 时钟转换时， f_H 将自动关闭以降低功耗。

CTRL 寄存器

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

“x”：未知

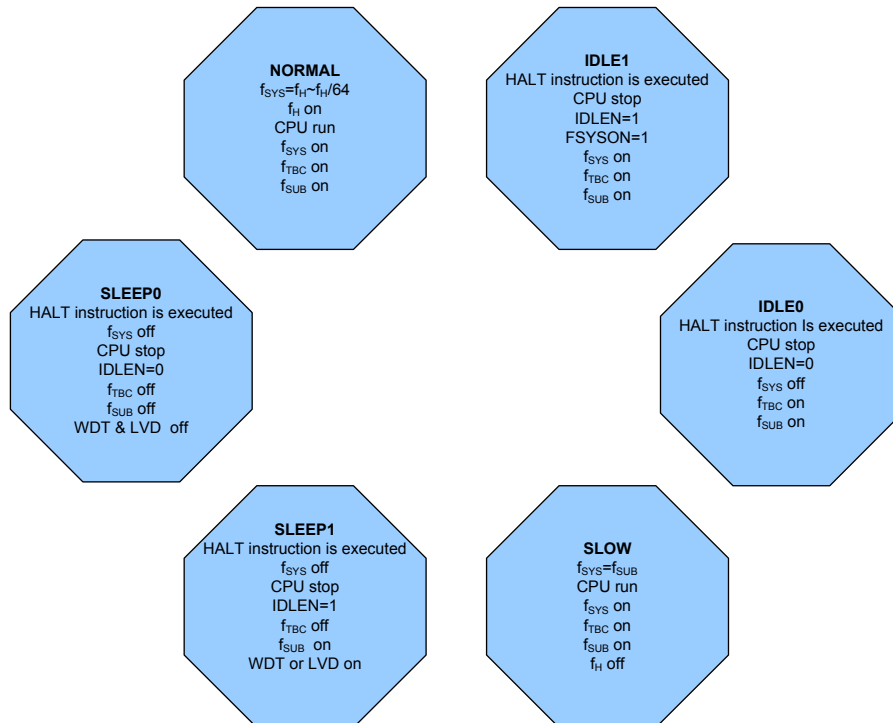
- Bit 7 **FSYSON**: 空闲模式下 f_{sys} 控制位
 0: 除能
 1: 使能
- Bit 6~4 未定义, 读为 “0”
- Bit 3 **FSUBF**: FSUBC 控制寄存器软件复位标志位
 0: 未发生
 1: 发生
 如果 FSUBC 控制寄存器的 FSUB6~FSUB0 位含有未定义值, 该位设置为 1。该位只能通过应用程序清零。
- Bit 2 **LVRF**: LVR 复位标志位
 详见其它章节
- Bit 1 **LRF**: LVRC 控制寄存器软件复位标志位
 详见其它章节
- Bit 0 **WRF**: WDTC 控制寄存器软件复位标志位
 详见其它章节

工作模式切换

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

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

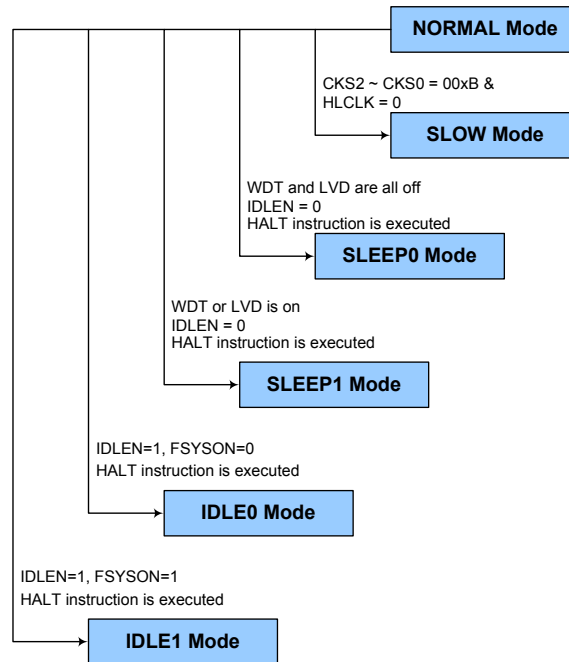
当 HLCLK 位变为低电平时，时钟源将由高速时钟源 f_H 转换成时钟源 $f_H/2 \sim f_H/64$ 或 f_{SUB} 。若时钟源来自 f_{SUB} ，高速时钟源将停止运行以节省耗电。此时须注意， $f_H/16$ 和 $f_H/64$ 内部时钟源也将停止运行，由此会影响到如 TM 等内部功能的工作。所附流程图显示了单片机在不同工作模式间切换时的变化。



正常模式切换到低速模式

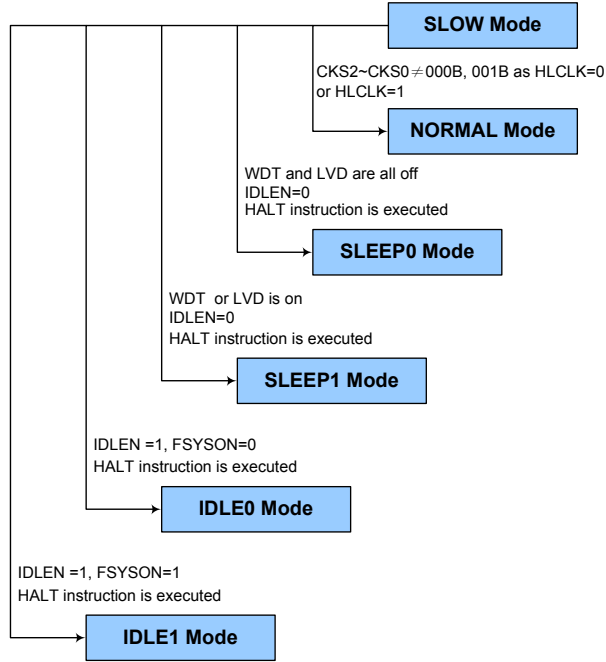
系统运行在正常模式时使用高速系统振荡器，因此较为耗电。可通过设置 SMOD 寄存器中的 HLCLK 位为“0”及 CKS2~CKS0 位为“000”或“001”使系统时钟切换至运行在低速模式下。此时将使用低速系统振荡器以节省耗电。用户可在对性能要求不高的操作中使用此方法以减少耗电。

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



低速模式切换到正常模式

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



进入休眠模式 0

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

- 系统时钟、WDT 时钟和时基时钟停止运行，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。
- WDT 将被清除并停止运行。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。

进入休眠模式 1

进入休眠模式 1 的方法仅有一种——应用程序中执行“HALT”指令前需设置寄存器 SMOD 中 IDLEN 位为“0”且 WDT 或 LVD 功能使能。在上述条件下执行该指令后，将发生的情况如下：

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

进入空闲模式 0

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

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

进入空闲模式 1

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

- 系统时钟、时基时钟 f_{TBC} 和 f_{SUB} 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。
- 如果 WDT 功能使能，WDT 将被清零并重新开始计数。如果 WDT 功能除能，WDT 将被清零并停止计数。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。

待机电流的注意事项

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

另外还需注意单片机设为输出的 I/O 引脚上的负载。应将它们设置在有最小拉电流的状态或将它们和其它的 CMOS 输入一样接到没有拉电流的外部电路上。还应注意的是，如果选择 LXT 或 LIRC 振荡器，会导致耗电增加。

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

唤醒

系统进入休眠或空闲模式之后，可以通过以下几种方式唤醒：

- PA 口下降沿
- 系统中断
- WDT 溢出

若单片机由 WDT 溢出唤醒，则会发生看门狗定时器复位。实际的唤醒源可通过检测 TO 和 PDF 位确定。执行 HALT 指令，PDF 将被置位；系统上电或执行清除看门狗定时器指令会清零 PDF。看门狗计数器溢出将会置位 TO 标志位并唤醒系统，此时仅复位程序计数器和堆栈指针，其它标志位保持原有状态。

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

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源来自于内部时钟 f_{SUB} ，而 f_{SUB} 由 LIRC 或 LXT 提供，可通过 FSUBC 寄存器选择。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 WDTC 寄存器中的 WS2~WS0 位来决定。电压为 5V 时内部振荡器 LIRC 的频率大约为 32kHz，这个特殊的内部时钟周期随 V_{DD} 、温度和制成的不同而变化。LXT 振荡器由外部 32.768kHz 晶振提供。

看门狗定时器控制寄存器

WDTC 寄存器用于选择溢出周期，控制 WDT 功能的使能 / 除能及软件复位单片机。

WDTC 寄存器

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

Bit 7~3 **WE4~WE0**: WDT 功能软件控制

10101: 除能
01010: 使能
其它值: MCU 复位

如果这些位因干扰变成其它值或软件设置，导致单片机复位，这需要延迟时间 t_{SRESET} 响应复位且 CTRL 寄存器中的 WRF 标志位会被置位。

Bit 2~0 **WS2~WS0**: WDT 溢出周期选择位

000: $2^8/f_{SUB}$
001: $2^{10}/f_{SUB}$
010: $2^{12}/f_{SUB}$
011: $2^{14}/f_{SUB}$
100: $2^{15}/f_{SUB}$
101: $2^{16}/f_{SUB}$
110: $2^{17}/f_{SUB}$
111: $2^{18}/f_{SUB}$

这三位控制 WDT 时钟源的分频比，从而实现了对 WDT 溢出周期的控制。

CTRL 寄存器

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

“x”：未知

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

Bit 6~4 未定义，读为“0”

Bit 3 **FSUBF**: FSUBF 控制寄存器软件复位标志位
详见其它章节

- Bit 2 **LVRF**: LVR 复位标志位
详见其它章节
- Bit 1 **LRF**: LVR 控制寄存器 LVRC 软件复位标志位
详见其它章节
- Bit 0 **WRF**: WDT 控制寄存器软件复位标志位
0: 未发生
1: 发生
当 WDT 控制寄存器软件复位时, 此位被置为“1”, 且只能通过程序清零。

看门狗定时器操作

当 WDT 溢出时, 它产生一个芯片复位的动作。这也就意味着正常工作期间, 用户需在应用程序中看门狗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, 这些清除指令都不能被正确执行, 此种情况下, 看门狗将溢出以使单片机复位。WDTC 寄存器中的 WE4~WE0 位可使能 / 除能看门狗定时器和复位单片机。如果 WE4~WE0 为 10101B, 则 WDT 除能; 如果 WE4~WE0 为 01010B, 则 WDT 使能; 如果 WE4~WE0 为其它任意值, 则经过延迟时间 t_{SRESET} 后单片机复位。上电后这几位的值为 01010B。

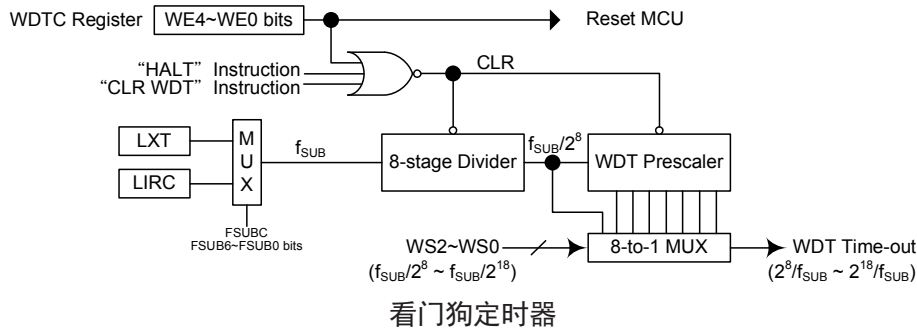
WE4~WE0 位	WDT 功能
10101B	除能
01010B	使能
其它值	MCU 复位

看门狗定时器使能 / 除能控制

程序正常运行时, WDT 溢出将导致芯片复位, 并置位状态标志位 TO。若系统处于休眠或空闲模式, 当 WDT 发生溢出时, 状态寄存器中的 TO, 程序计数器 PC 和堆栈指针 SP 将被复位。有 3 种方法可以用来清除 WDT 的内容。第一种是 WDT 复位, 即往 WE4~WE0 位写入除了 01010B 或 10101B 之外的其它值, 第二种是通过软件清除指令, 第三种是通过“HALT”指令。

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

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



复位和初始化

复位功能是整个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定好的状态并开始执行第一条程序指令。上电复位以后，在程序执行之前，部分重要的内部寄存器将会被设定为预先设定的状态。程序计数器就是其中之一，它会被清除为零，使得单片机从最低的程序存储器地址开始执行程序。

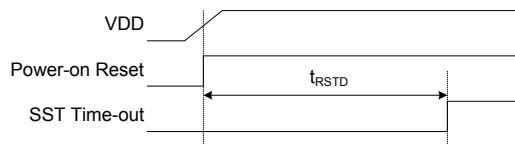
另一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。

复位功能

包括内部事件触发复位，单片机共有多种复位方式：

上电复位

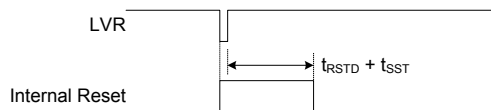
这是最基本且不可避免的复位，发生在单片机上电后。除了保证程序存储器从开始地址执行，上电复位也使得其它寄存器被设定在预设条件。所有的输入/输出端口控制寄存器在上电复位时会保持高电平，以确保上电后所有引脚被设定为输入状态。



上电复位时序图

低电压复位 – LVR

单片机具有低电压复位电路，用来监测它的电源电压。低电压复位功能始终使能于特定的电压值， V_{LVR} 。例如在更换电池的情况下，单片机供应的电压可能会在 $0.9V \sim V_{LVR}$ 之间，这时 LVR 将会自动复位单片机且 CTRL 寄存器中的 LVRF 标志位置位。LVR 包含以下的规格：有效的 LVR 信号，即在 $0.9V \sim V_{LVR}$ 的低电压状态的时间，必须超过 LVR/LVD 电气特性中 t_{LVR} 参数的值。如果低电压存在不超过 t_{LVR} 参数的值，则 LVR 将会忽略它且不会执行复位功能。 V_{LVR} 参数值可通过 LVRC 寄存器中的 LVS7~LVS0 位设置不同的值。若由于受到干扰 LVS7~LVS0 变为其它值时，需经过延迟时间 t_{SRESET} 后响应复位。此时 CTRL 寄存器的 LRF 位被置位。上电后寄存器的值为 01010101B。当单片机进入暂停模式时，LVR 功能会自动除能。



低电压复位时序图

• LVRC 寄存器

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

Bit 7~0 **LVS7~LVS0**: LVR 电压选择

01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V

其它值: MCU 复位 — 寄存器会复位到 POR 值

若低电压情况发生且满足以上定义的低电压复位值, 则单片机复位。当低电压状况保持时间大于 t_{LVR} 后, 响应复位。此种复位寄存器的值保持不变。

除了以上定义的低电压复位值外, 其它值也能导致单片机复位。需要经过一段延迟时间 t_{SRESET} 才响应复位。此种情况下寄存器将复位到原来的 POR 值。

• CTRL 寄存器

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

“x”: 未知

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

Bit 6~4 未定义, 读为 “0”

Bit 3 **FSUBF**: FSUBF 复位控制寄存器软件复位标志位
详见其它章节

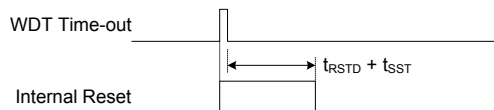
Bit 2 **LVRF**: LVR 复位标志位
0: 未发生
1: 发生
此位在出现指定低压复位情况时被置位, 且仅能由应用程序清零。

Bit 1 **LRF**: LVR 控制寄存器 LVRC 软件复位标志位
0: 未发生
1: 发生
当 LVRC 寄存器的值不属于任何具体定义的 LVR 电压寄存器值时此位被置位。该复位方式与软件复位功能很相似。此位只能由应用程序清零。

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

正常运行时看门狗溢出复位

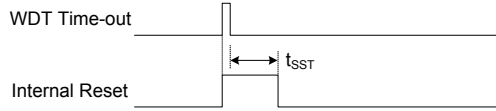
除了看门狗溢出标志位 TO 将被设为 “1” 之外, 正常运行时看门狗溢出复位和 LVR 复位相同。



正常运行时看门狗溢出时序图

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“0”及 TO 位被设为“1”外，大部分的条件保持不变。图中 t_{SST} 的详细说明请参考交流电气特性。



休眠或空闲时看门狗溢出复位时序图

复位初始状态

不同的复位形式以不同的途径影响复位标志位。这些标志位，即 PDF 和 TO 位存放在状态寄存器中，由休眠或空闲模式功能或看门狗计数器等几种控制器操作控制。复位标志位如下所示：

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

注：“u”代表不改变

在单片机上电复位之后，各功能单元初始化的情况，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	复位后都清除，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。若芯片有多种封装类型，表格反应较大的封装的情况。

寄存器	上电复位	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠模式)
IAR0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
IAR1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1L	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu

寄存器	上电复位	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠模式)
TBHP	--- x xxxx	--- u uuuu	--- u uuuu	--- u uuuu
STATUS	xx00 xxxx	uuuu uuuu	uu1u uuuu	uu11 uuuu
IAR2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP2L	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP2H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
SMOD	000- 0011	000- 0011	000- 0011	uuu- uuuu
TBC	0011 - 111	0011 - 111	0011 - 111	uuuu - uuu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
LVDC	-- 00 - 000	-- 00 - 000	-- 00 - 000	-- uu - uuu
LVRC	0101 0101	0101 0101	0101 0101	uuuu uuuu
CTRL	0--- 0x00	0--- u1uu	0--- uuuu	0--- uuuu
FSUBC	0010 1010	0010 1010	0010 1010	uuuu uuuu
INTEG	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC0	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI0	-- 00 -- 00	-- 00 -- 00	-- 00 -- 00	-- uu -- uu
MFI1	-- 00 -- 00	-- 00 -- 00	-- 00 -- 00	-- uu -- uu
MFI2	-- 00 -- 00	-- 00 -- 00	-- 00 -- 00	-- uu -- uu
MFI3	-- 00 -- 00	-- 00 -- 00	-- 00 -- 00	-- uu -- uu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	-- 00 0000	-- 00 0000	-- 00 0000	-- uu uuuu
PB	-- 11 1111	-- 11 1111	-- 11 1111	-- uu uuuu
PBC	-- 11 1111	-- 11 1111	-- 11 1111	-- uu uuuu
IOHR0	0000 0000	0000 0000	0000 0000	uuuu uuuu
IOHR1	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI4	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADRL	xxxx ----	xxxx ----	xxxx ----	uuuu ---- (ADRFS = 0)
				uuuu uuuu (ADRFS = 1)
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS = 0)
				---- uuuu (ADRFS = 1)
ADCR0	0110 0000	0110 0000	0110 0000	uuuu uuuu
ADCR1	00- 0 - 000	00- 0 - 000	00- 0 - 000	uu- u - uuu
ACERL	0000 0000	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (空闲/休眠模式)
ACERH	---- --00	---- --00	---- --00	---- --uu
TM0C0	0000 0---	0000 0---	0000 0---	uuuu u---
TM0C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0DH	---- --00	---- --00	---- --00	---- --uu
TM0AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0AH	---- --00	---- --00	---- --00	---- --uu
TM0RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0RPH	---- --00	---- --00	---- --00	---- --uu
TM1C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1DH	---- --00	---- --00	---- --00	---- --uu
TM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1AH	---- --00	---- --00	---- --00	---- --uu
INTC3	---0 ---0	---0 ---0	---0 ---0	---u ---u
EEA	--00 0000	--00 0000	--00 0000	--uu uuuu
EED	0000 0000	0000 0000	0000 0000	uuuu uuuu
USR	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TXR/RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMPC	---0 0000	---0 0000	---0 0000	---u uuuu
TM2C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2DH	---- --00	---- --00	---- --00	---- --uu
TM2AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2AH	---- --00	---- --00	---- --00	---- --uu
TM3C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM3C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM3DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM3DH	---- --00	---- --00	---- --00	---- --uu
TM3AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM3AH	---- --00	---- --00	---- --00	---- --uu
LCDC0	xxxx -xxx	xxxx -xxx	xxxx -xxx	uuuu -uuu
LCDC1	xxx- xxxx	xxx- xxxx	xxx- xxxx	uuu- uuuu
SEGCR0	1111 1111	1111 1111	1111 1111	uuuu uuuu
SEGCR1	1111 1111	1111 1111	1111 1111	uuuu uuuu
SEGCR2	1111 1111	1111 1111	1111 1111	uuuu uuuu

寄存器	上电复位	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠模式)
SEGCR3	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PE	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFPU	0000 ----	0000 ----	0000 ----	uuuu ----
PF	1111 ----	1111 ----	1111 ----	uuuu ----
PFC	1111 ----	1111 ----	1111 ----	uuuu ----
PGPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PG	1111 1111	1111 1111	1111 1111	uuuu uuuu
PGC	1111 1111	1111 1111	1111 1111	uuuu uuuu
SIMC0	111- 0000	111- 0000	111- 0000	uuu- uuuu
SIMC1	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMTOC	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEC	---- 0000	---- 0000	---- 0000	---- uuuu

注：“u”表示不改变
“x”表示未知
“-”表示未定义

输入 / 输出端口

盛群单片机的输入 / 输出控制具有很大的灵活性。每个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~PG 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	PAC5	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU4	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	PB5	PB4	PB3	PB2	PB1	PB0
PBC	—	—	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PE	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEC	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PF	PF7	PF6	PF5	PF4	—	—	—	—
PFC	PFC7	PFC6	PFC5	PFC4	—	—	—	—
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	—	—	—	—
PG	PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0
PGC	PGC7	PGC6	PGC5	PGC4	PGC3	PGC2	PGC1	PGC0
PGPU	PGPU7	PGPU6	PGPU5	PGPU4	PGPU3	PGPU2	PGPU1	PGPU0

“—”未定义，读为“0”

输入 / 输出逻辑功能寄存器列表

上拉电阻

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

PxPU 寄存器

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

PxPUn: I/O 端口 x 引脚上拉功能控制

- 0: 除能
- 1: 使能

PxPUn 位用于控制引脚的上拉功能。此处“x”可以是 A、B、C、D、E、F 或 G。然而，每个实际使用端口的位可以是不同的。

PA 口唤醒

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

PAWU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAWU7	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

PAWU7n: PA 口唤醒功能控制位

- 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~PGC，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。应注意，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: 输入 / 输出端口 x 引脚类型选择位

- 0: 输出
- 1: 输入

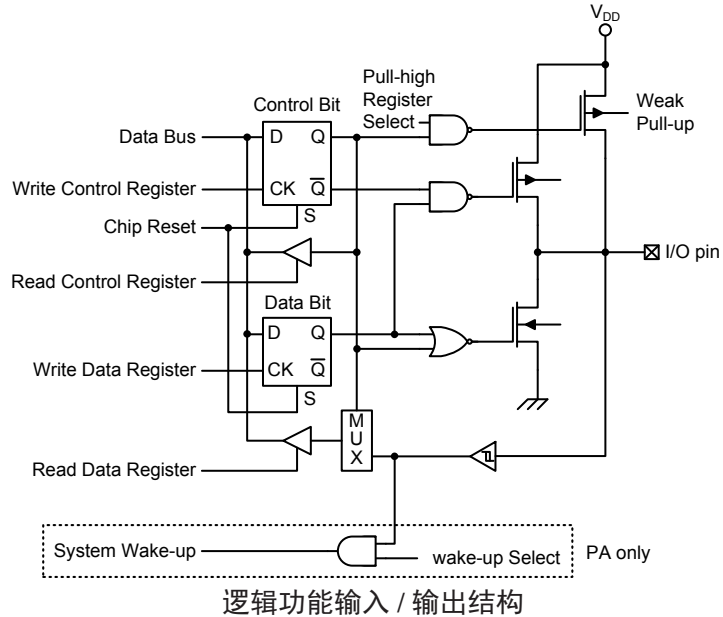
PxCn 位用于引脚类型选择。此处“x”可以是 A、B、C、D、E、F 或 G。然而，每个实际使用端口的位可以是不同的。

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。每个引脚上的功能选择方式都是不同的，且具有优先顺序，从而允许同时选中多种引脚功能。

输入 / 输出引脚结构

下图为输入 / 输出引脚的逻辑功能结构图。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的一个参考。图中的引脚共用结构并非针对所有单片机。



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。如果某些引脚位元被设定输出状态，这些输出引脚会初始为电平输出，除非数据寄存器端口在程序中被预先设定。设置哪些引脚是输入及哪些引脚是输出，可通过设置正确的值到适当的端口控制寄存器，或使用指令“SET [m].i”及“CLR [m].i”来设定端口控制寄存器中个别的位。注意，当使用这些位控制指令时，系统即将产生一个读 - 修改 - 写的操作。单片机需要先读入整个端口上的数据，修改个别的位，然后重新把这些数据写入到输出口。

PA 口的每个引脚都带唤醒功能。当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 端口的任一引脚高到低电平转换，可以设置端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。每个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 / 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独立简易型和周期型定时器章节。

简介

该单片机包含 4 个 TM，分别命名为 TM0、TM1、TM2 和 TM3。每个 TM 可被划分为一个特定的类型，即简易型 TM 或周期型 TM。虽然性质相似，但不同 TM 特性复杂度不同。本章介绍简易型和周期型 TM 的共性，更多详细资料分别见后面各章。两种类型 TM 的特性和区别，详见下表。

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

TM 功能概要

该单片机包含四个 TM，TM0~TM3 分别对应为简易型或周期型，如下表所示。

TM0	TM1	TM2	TM3
10-bit PTM	10-bit CTM	10-bit CTM	10-bit CTM

TM 名称 / 类型参考

TM 操作

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

TM 时钟源

驱动 TM 计数器的时钟源很多。通过设置 TM 控制寄存器的 TnCK2~TnCK0 位来选择所需的时钟源。该时钟源来自系统时钟 f_{SYS} 或内部高速时钟 f_H 或 f_{TBC} 时钟源或外部 TCKn 引脚时钟。TCKn 引脚时钟源用于允许外部信号作为 TM 时钟源或用于事件计数。

TM 中断

简易型和周期型 TM 各自拥有两个内部中断，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，都有一个 TM 输入引脚，分别为 TCK_n。通过设置 TM_nCO 寄存器中的 TnCK2~TnCK0 位，选择 TCK_n 引脚作为 TM_n 的时钟源输入脚。外部时钟源可通过该引脚来驱动内部 TM。TCK_n 引脚可选择上升沿有效或下降沿有效。

每个 TM 有一个或两个输出引脚 TP_n。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 TP_n 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，故在使用 TM 输入和输出功能时需要先行设置相应的引脚共用功能选择寄存器。寄存器中的一个独位用于选择相关引脚作为 TM 输出引脚或其它共用功能。不同类型的 TM 输出引脚的具体情况详见下表。

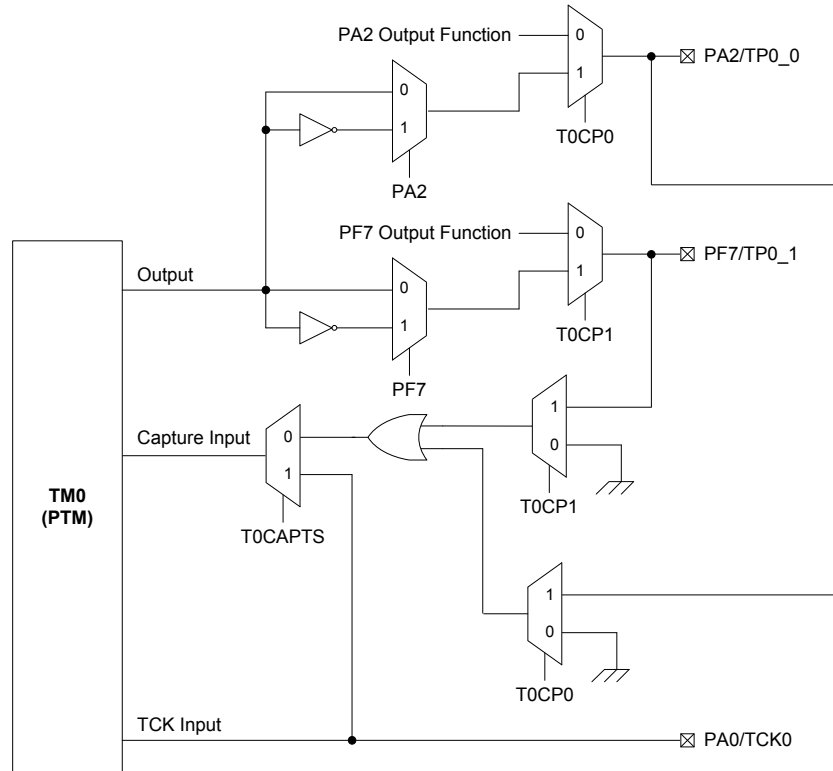
周期型 TM 输出引脚的名称带“_0”或“_1”后缀表示来自多输出引脚。这允许 TM 产生一对互补的输出，可通过 I/O 寄存器的数据位。

TM0	TM1	TM2	TM3
TP0_0, TP0_1	TP1	TP2	TP3

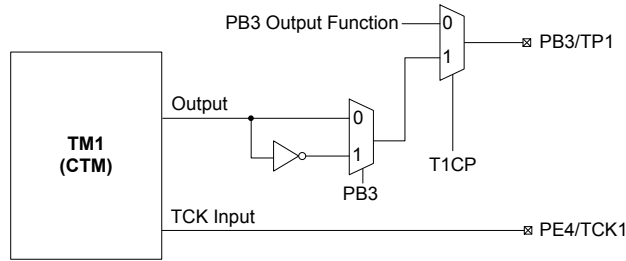
TM 外部引脚

TM 输入 / 输出引脚控制寄存器

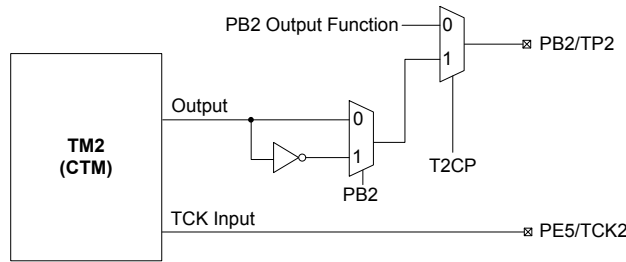
通过设置与 TM 输入 / 输出引脚相关的引脚共用功能选择寄存器的位，选择作为 TM 输入 / 输出功能或其它共用功能。正确设置相应的选择位将引脚用作 TM 输入 / 输出，如果选择位复位为 0，该引脚将保留为其原有的功能。



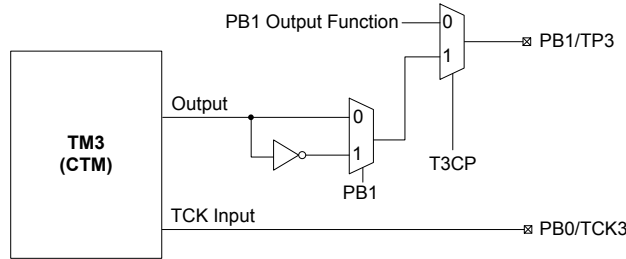
TM0 功能引脚控制方框图



TM1 功能引脚控制方框图



TM2 功能引脚控制方框图



TM3 功能引脚控制方框图

TMPC 寄存器

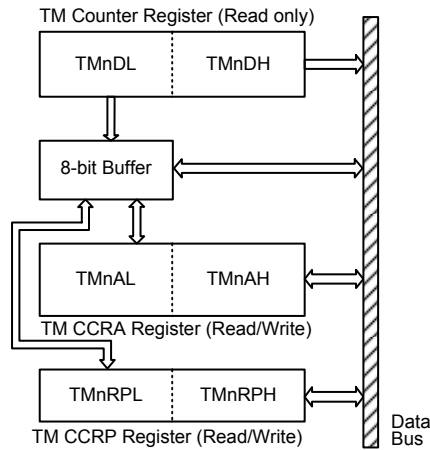
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	T3CP	T2CP	T1CP	T0CP1	T0CP0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

- Bit 7~5 未定义，读为“0”
- Bit 4 **T3CP**: TP3 引脚控制位
0: 除能
1: 使能
- Bit 3 **T2CP**: TP2 引脚控制位
0: 除能
1: 使能
- Bit 2 **T1CP**: TP1 引脚控制位
0: 除能
1: 使能
- Bit 1 **T0CP1**: TP0_1 引脚控制位
0: 除能
1: 使能
- Bit 0 **T0CP0**: TP0_0 引脚控制位
0: 除能
1: 使能

编程注意事项

TM 计数寄存器和捕捉 / 比较寄存器 CCRA 和 CCRP 寄存器，含有低字节和高字节结构。高字节可直接访问，低字节则仅能通过一个内部 8-bit 的缓存器进行访问。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 和 CCRP 寄存器访问方式如下图所示，读写这些成对的寄存器需通过上述的特殊方式。建议使用“MOV”指令按照以下步骤访问 CCRA 和 CCRP 低字节寄存器，即 TMnAL 和 TMnRPL，否则可能导致无法预期的结果。



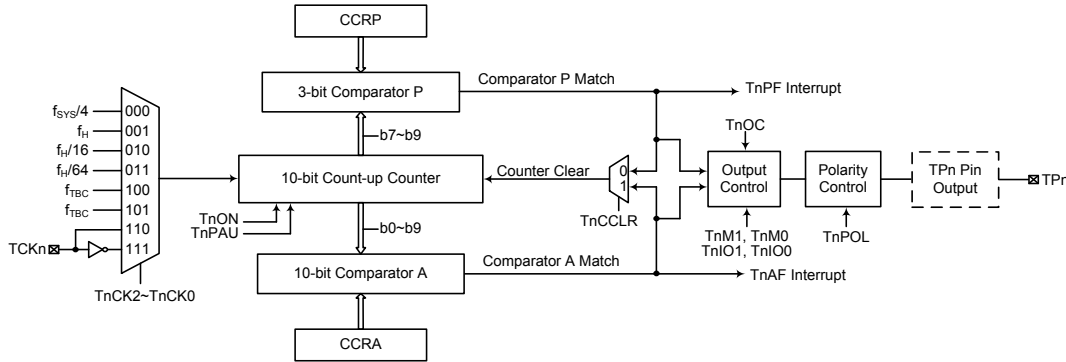
读写流程如下步骤所示：

- 写数据至 CCRA 或 CCRP
 - ◆ 步骤 1. 写数据至低字节寄存器 TMnAL 或 TMnRPL
 - 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 TMnAH 或 TMnRPH
 - 注意，此时数据直接写入高字节寄存器，同时锁存在 8-bit 缓存器中的数据写入低字节寄存器。
- 由计数器寄存器和 CCRA 或 CCRP 中读取数据
 - ◆ 步骤 1. 由高字节寄存器 TMnDH, TMnAH 或 TMnRPH 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 TMnDL、TMnAL 或 TMnRPL 读取数据
 - 注意，此时读取 8-bit 缓存器中的数据。

简易型 TM – CTM

简易型 TM 包括三种工作模式，即比较匹配输出，定时 / 事件计数器和 PWM 输出模式。简易型 TM 由一个外部输入脚控制并驱动一个外部输出脚。

名称	TM 编号	TM 输入引脚	TM 输出引脚
10-bit CTM	1, 2, 3	TCK1, TCK2, TCK3	TP1, TP2, TP3



简易型 TM 方框图 (n = 1~3)

简易型 TM 操作

简易型 TM 核心是一个由用户选择的内部或外部时钟源驱动的 10 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 CCRP 和 CCRA 寄存器中的值进行比较。CCRP 是 3-bit 的，与计数器的高 3 位比较；而 CCRA 是 10-bit 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 TnON 位发生上升沿跳变清除计数器。此外，计数器溢出或比较匹配也会自动清除计数器。上述条件发生时，通常情况会产生 TM 中断信号。简易型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，也可以控制输出脚。所有工作模式的设定都是通过设置相关寄存器来实现的。

简易型 TM 寄存器介绍

简易型 TM 的所有操作由一系列寄存器控制。其中包含一对只读寄存器用来存放 10-bit 计数器的值，一对读 / 写寄存器存放内部 10-bit CCRA 的值，剩下两个控制寄存器设置不同的操作和控制模式以及 CCRP 的 3 个位。

寄存器名称	位							
	7	6	5	4	3	2	1	0
TMnC0	TnPAU	TnCK2	TnCK1	TnCK0	TnON	TnRP2	TnRP1	TnRP0
TMnC1	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnDPX	TnCCLR
TMnDL	D7	D6	D5	D4	D3	D2	D1	D0
TMnDH	—	—	—	—	—	—	D9	D8
TMnAL	D7	D6	D5	D4	D3	D2	D1	D0
TMnAH	—	—	—	—	—	—	D9	D8

10-bit 简易型 TM 寄存器列表 (n = 1~3)

TMnC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TnPAU	TnCK2	TnCK1	TnCK0	TnON	TnRP2	TnRP1	TnRP0
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 TnPAU:** TMn 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，TM 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 TnCK2~TnCK0:** 选择 TMn 计数时钟位
000: $f_{SYS}/4$
001: f_H
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: f_{TBC}
110: TCKn 上升沿时钟
111: TCKn 下降沿时钟
此三位用于选择 TM 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_{TBC} 是其它的内部时钟源，细节方面请参考振荡器章节。
- Bit 3 TnON:** TMn 计数器 On/Off 控制位
0: Off
1: On
此位控制 TM 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 TM。清零此位将停止计数器并关闭 TM 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 TM 处于比较匹配输出模式或 PWM 输出模式时，当 TnON 位经由低到高转换时，TM 输出脚将复位至 TnOC 位指定的初始值。
- Bit 2~0 TnRP2~TnRP0:** TMn CCRP 3-bit 寄存器，与 TMn 计数器 bit 9~bit 7 比较器 P 匹配周期
000: 1024 个 TMn 时钟周期
001: 128 个 TMn 时钟周期
010: 256 个 TMn 时钟周期
011: 384 个 TMn 时钟周期
100: 512 个 TMn 时钟周期
101: 640 个 TMn 时钟周期
110: 768 个 TMn 时钟周期
111: 896 个 TMn 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 TnCCLR 位设定为 0 时，比较结果为 0 并清除内部计数器。TnCCLR 位设为低，内部计数器在比较器 P 比较匹配发生时被重置；由于 CCRP 只与计数器高三位比较，比较结果是 128 时钟周期的倍数。CCRP 被清零时，实际上会使得计数器在最大值溢出。

TMnC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnDPX	TnCCLR
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 **TnM1~TnM0**: 选择 TMn 工作模式位

- 00: 比较匹配输出模式
- 01: 未定义
- 10: PWM 输出模式
- 11: 定时 / 计数器模式

这两位设置 TM 需要的工作模式。为了确保操作可靠，TM 应在 TnM1 和 TnM0 位有任何改变前先关掉。在定时 / 计数器模式，TM 输出脚控制必须除能。

Bit 5~4 **TnIO1~TnIO0**: 选择 TPn 输出功能位

比较匹配输出模式

- 00: 无变化
- 01: 输出低
- 10: 输出高
- 11: 输出翻转

PWM 输出模式

- 00: 强制无效状态
- 01: 强制有效状态
- 10: PWM 输出
- 11: 未定义

定时 / 计数器模式

未使用

此两位用于决定在一定条件达到时 TM 输出脚如何改变状态。这两位值的选择取决于 TM 运行在何种模式下。

在比较匹配输出模式下，TnIO1 和 TnIO0 位决定当比较器 A 比较匹配输出发生时 TM 输出脚如何改变状态。当比较器 A 比较匹配输出发生时 TM 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。TM 输出脚的初始值通过 TMnC1 寄存器的 TnOC 位设置取得。注意，由 TnIO1 和 TnIO0 位得到的输出电平必须与通过 TnOC 位设置的初始值不同，否则当比较匹配发生时，TM 输出脚将不会发生变化。在 TM 输出脚改变状态后，通过 TnON 位由低到高电平的转换复位至初始值。

在 PWM 输出模式，TnIO1 和 TnIO0 用于决定比较匹配条件发生时怎样改变 TM 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 TMn 关闭时改变 TnIO1 和 TnIO0 位的值是很有必要的。若在 TM 运行时改变 TnIO1 和 TnIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **TnOC**: TMn 输出控制位

比较匹配输出模式

- 0: 初始低
- 1: 初始高

PWM 输出模式

- 0: 低有效
- 1: 高有效

这是 TM 输出脚输出控制位。它取决于 TM 此时正运行于比较匹配输出模式还是 PWM 输出模式。若 TM 处于定时 / 计数器模式，则其不受影响。在比较匹配输出模式时，比较匹配发生前其决定 TM 输出脚的逻辑电平值。在 PWM 输出模式时，其决定 PWM 信号是高有效还是低有效。

- Bit 2 **TnPOL**: TMn 输出极性控制位
 0: 同相
 1: 反相
 此位控制 TMn 输出脚的极性。此位为高时 TM 输出脚反相，为低时 TM 输出脚同相。若 TM 处于定时 / 计数器模式时其不受影响。
- Bit 1 **TnDPX**: TMn 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 **TnCCLR**: 选择 TMn 计数器清零条件位
 0: TMn 比较器 P 匹配
 1: TMn 比较器 A 匹配
 此位用于选择清除计数器的方法。简易型 TM 包括两个比较器 – 比较器 A 和比较器 P。这两个比较器每个都可以用作清除内部计数器。TnCCLR 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。TnCCLR 位在 PWM 输出模式时未使用。

TMnDL 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**: TMn 计数器低字节寄存器 bit 7~bit 0
 TMn 10-bit 计数器 bit 7~bit 0

TMnDH 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**: TMn 计数器高字节寄存器 bit 1~bit 0
 TMn 10-bit 计数器 bit 9~bit 8

TMnAL 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**: TMn CCRA 低字节寄存器 bit 7~bit 0
 TMn 10-bit CCRA bit 7~bit 0

TMnAH 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**: TMn CCRA 高字节寄存器 bit 1~bit 0
TMn 10-bit CCRA bit 9~bit 8

简易型 TM 工作模式

简易型 TM 有三种工作模式，即比较匹配输出模式，PWM 输出模式或定时 / 计数器模式。通过设置 TMnC1 寄存器的 TnM1 和 TnM0 位选择任意工作模式。

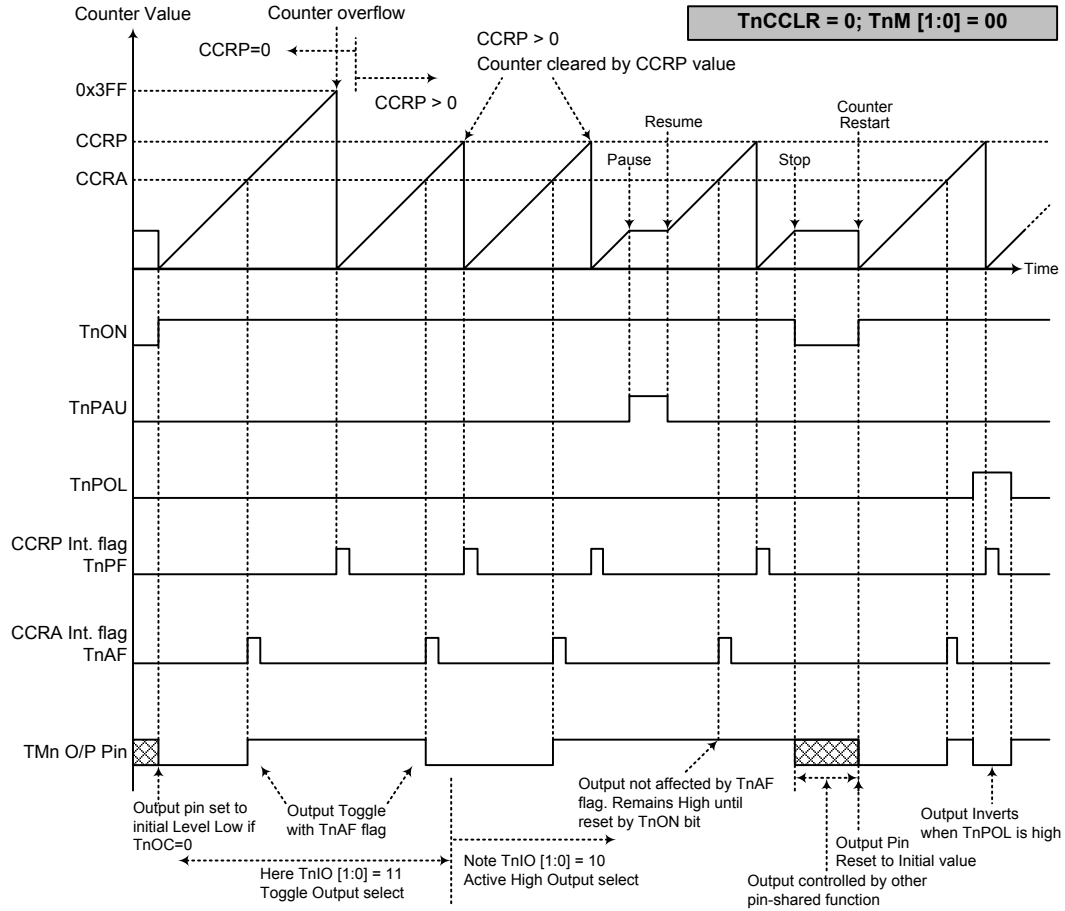
比较匹配输出模式

为使 TM 工作在此模式，TMnC1 寄存器中的 TnM1 和 TnM0 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 TnCCLR 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 TnAF 和 TnPF 将分别置起。

如果 TMnC1 寄存器的 TnCCLR 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 TnAF 中断请求标志产生。所以当 TnCCLR 为高时，不产生 TnPF 中断请求标志。如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 TnAF 请求标志。

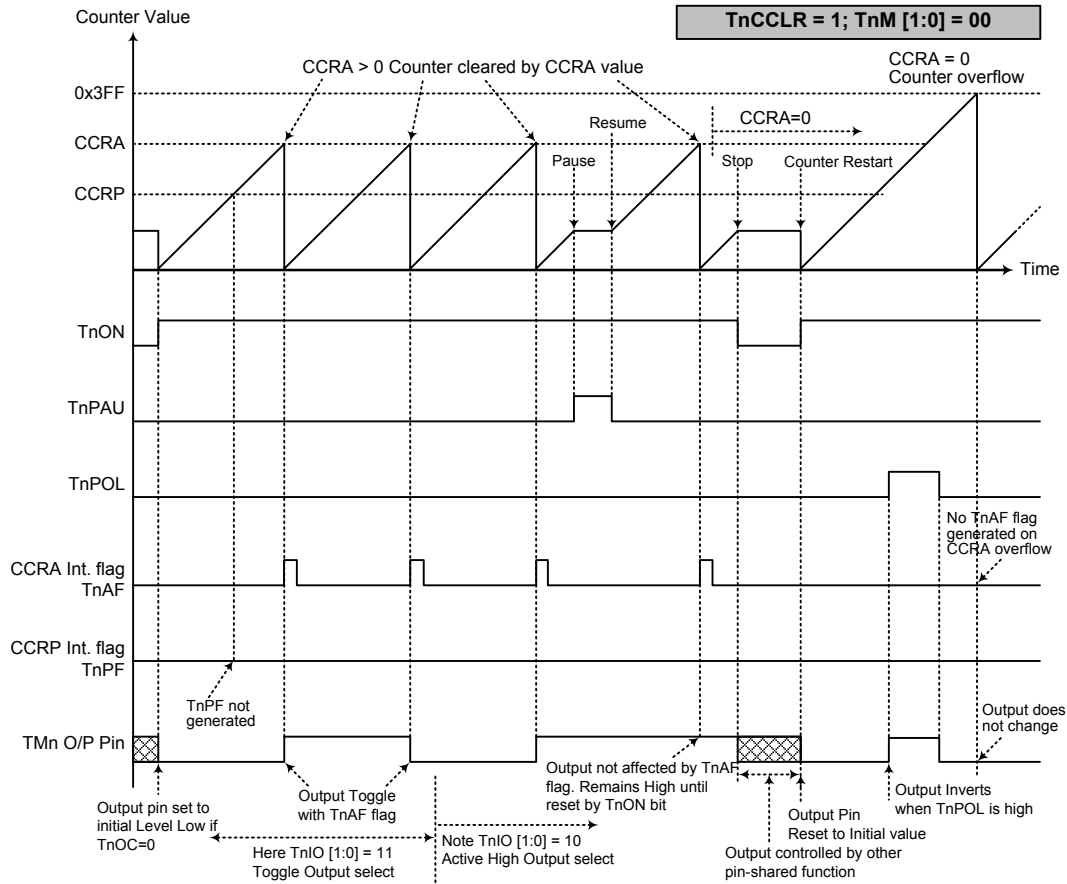
若 CCRA 全为“0”，则计数器将在其最大值 3FF 处溢出。此时不产生 TnAF 中断请求标志位。

正如该模式名所言，当比较匹配发生后，TM 输出脚状态改变。当比较器 A 比较匹配发生后 TnAF 标志产生时，TM 输出脚状态改变。比较器 P 比较匹配发生时产生的 TnPF 标志不影响 TM 输出脚。TM 输出脚状态改变方式由 TMnC1 寄存器中 TnIO1 和 TnIO0 位决定。当比较器 A 比较匹配发生时，TnIO1 和 TnIO0 位决定 TM 输出脚输出高，低或翻转当前状态。TM 输出脚初始值。TMn 输出脚初始值，在 TnON 位由低到高电平的变化后通过 TnOC 位设置。注意，若 TnIO1 和 TnIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 - TnCCLR = 0 (n = 1~3)

- 注: 1. TnCCLR = 0, 比较器 P 匹配将清除计数器。
2. TMn 输出脚仅由 TnAF 标志位控制。
3. 在 TnON 上升沿 TM 输出脚复位至初始值。



比较匹配输出模式 - TnCCLR = 1 (n = 1~3)

- 注：1. TnCCLR = 1，比较器 A 匹配将清除计数器。
 2. TMn 输出脚仅由 TnAF 标志位控制。
 3. 在 TnON 上升沿 TM 输出脚复位至初始值。
 4. 当 TnCCLR = 1 时，TnPF 标志位不会产生。

定时 / 计数器模式

为使 TM 工作在此模式，TMnC1 寄存器中的 TnM1 和 TnM0 位需要设置为“11”。定时 / 计数器模式与比较输出模式操作方式相同，并产生同样的中断请求标志。不同的是，在定时 / 计数器模式下 TM 输出脚未使用。因此，比较匹配输出模式中的描述和时序图可以适用于此功能。该模式中未使用的 TM 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 TM 工作在此模式，TMnC1 寄存器中的 TnM1 和 TnM0 位需要设置为“10”。TM 的 PWM 功能在马达控制，加热控制，照明控制等方面十分有用。给 TM 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就极其灵活。在 PWM 输出模式中，TnCCLR 位不影响 PWM 操作。CCRA 和 CCRP 寄存器决定 PWM 波形，一个用来清除内部计数器并控制 PWM 波形的频率，另一个用来控制占空比。哪个寄存器控制频率或占空比取决于 TMnC1 寄存器的 TnDPX 位。所以 PWM 波形频率和占空比由 CCRA 和 CCRP 寄存器共同决定。

当比较器 A 或比较器 P 比较匹配发生时，将产生 CCRA 或 CCRP 中断标志。TMnC1 寄存器中的 TnOC 位决定 PWM 波形的极性，TnIO1 和 TnIO0 位使能 PWM 输出或将 TM 输出脚置为逻辑高或逻辑低。TnPOL 位对 PWM 输出波形的极性取反。

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

CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	128	256	384	512	640	768	896	1024
Duty	CCRA							

若 $f_{SYS} = 8\text{MHz}$ ，TM 时钟源选择 $f_{SYS}/4$ ，CCRP = 100b，CCRA = 128，

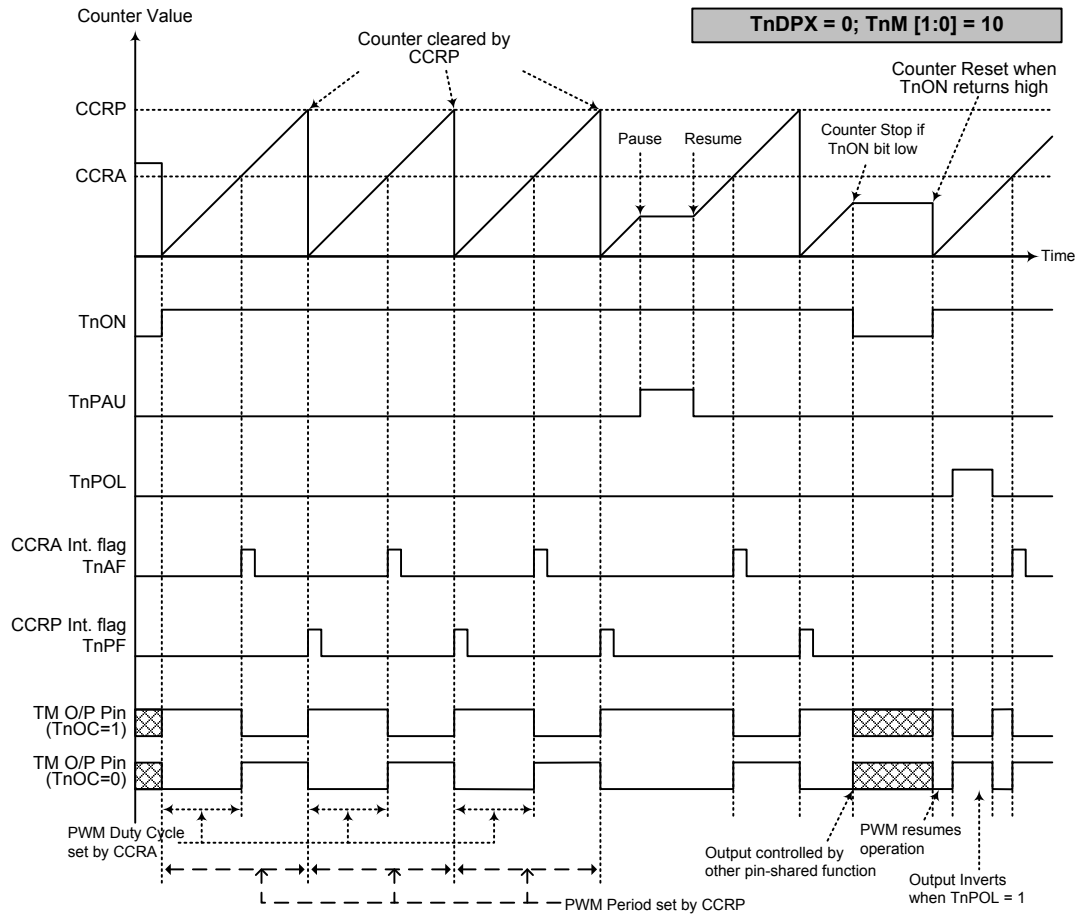
CTM PWM 输出频率 = $(f_{SYS}/4)/512 = f_{SYS}/2048 = 3.90625\text{kHz}$ ，duty = $128/512 = 25\%$

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%

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

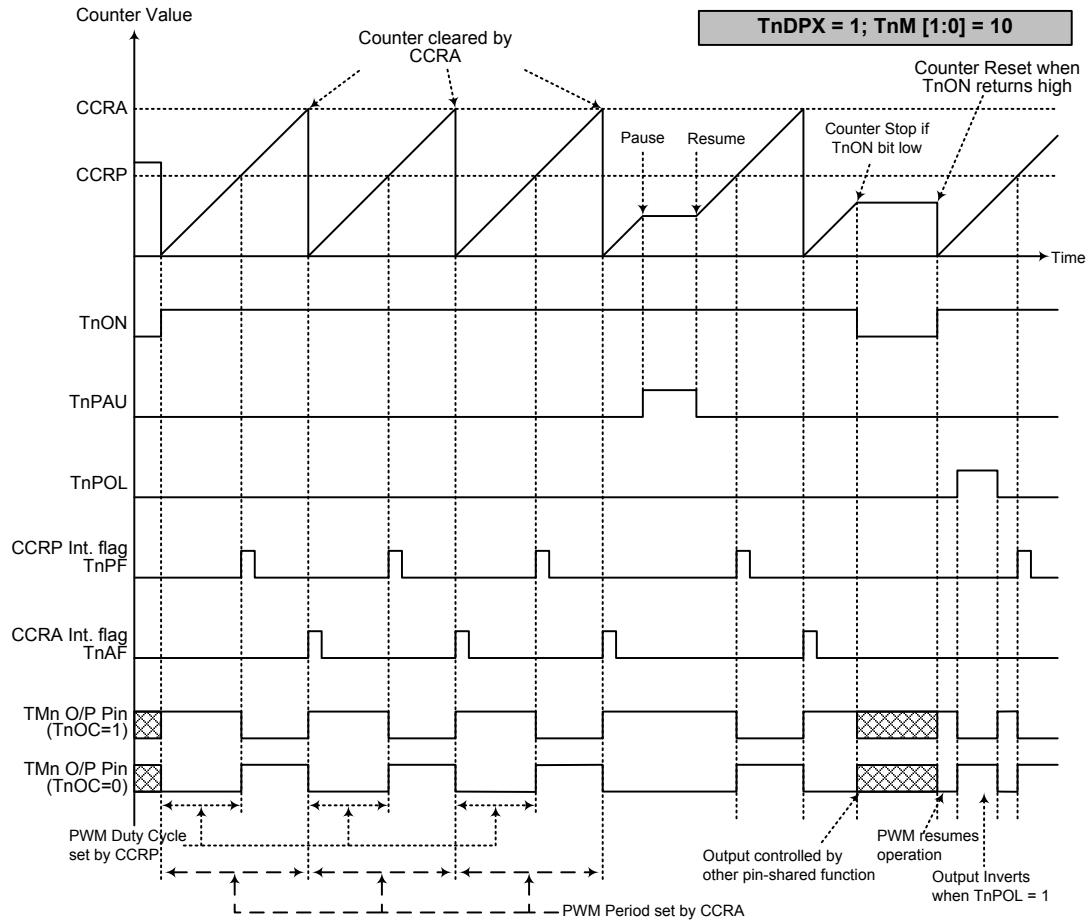
CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

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



PWM 输出模式 - TnDPX = 0 (n = 1~3)

- 注：1. TnDPX = 0, CCRP 清除计数器。
2. 计数器清零并设置 PWM 周期。
3. 即使在 TnIO[1:0] = 00 或 01 时，内部 PWM 功能继续运行。
4. TnCCLR 位不影响 PWM 操作。



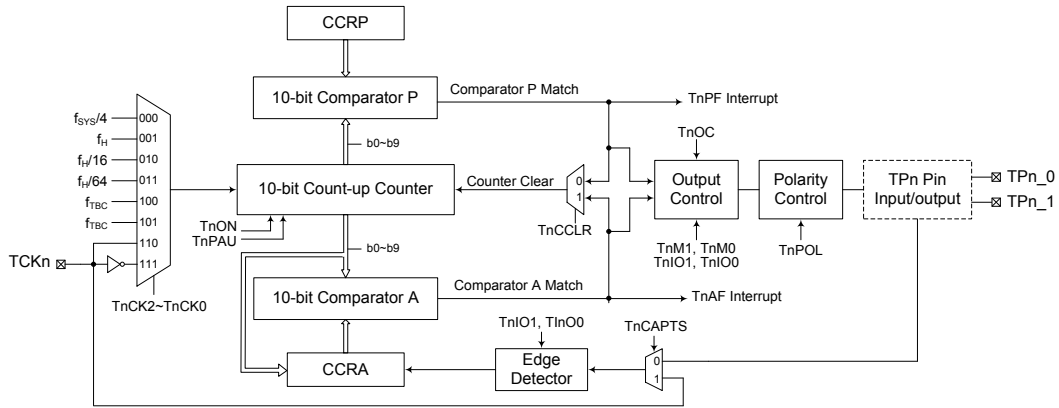
PWM 输出模式 - TnDPX = 1 (n = 1~3)

- 注：1. TnDPX = 1, CCRA 清除计数器。
 2. 计数器清零并设置 PWM 周期。
 3. 即使 TnIO[1:0] = 00 或 01, 内部 PWM 功能继续运行。
 4. TnCCLR 位不影响 PWM 操作。

周期型 TM – PTM

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

名称	TM 编号	TM 输入引脚	TM 输出引脚
10-bit PTM	0	TCK0, TP0_0, TP0_1	TP0_0, TP0_1



周期型 TM 方框图 (n = 0)

周期型 TM 操作

周期型 TM 的核心为 10-bit 向上计数器，由用户选择的内部或外部时钟源驱动。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。CCRA 和 CCRP 的宽度为 10-bit。

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

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 内部计数器的值，两对读 / 写寄存器存放 10-bit CCRA 和 CCRP 的值。剩下两个控制寄存器用来设置不同的操作和控制模式。

寄存器名称	位							
	7	6	5	4	3	2	1	0
TMnC0	TnPAU	TnCK2	TnCK1	TnCK0	TnON	—	—	—
TMnC1	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnCPTS	TnCCLR
TMnDL	D7	D6	D5	D4	D3	D2	D1	D0
TMnDH	—	—	—	—	—	—	D9	D8
TMnAL	D7	D6	D5	D4	D3	D2	D1	D0
TMnAH	—	—	—	—	—	—	D9	D8
TMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
TMnRPH	—	—	—	—	—	—	D9	D8

10-bit 周期型 TM 寄存器列表 (n = 0)

TMnC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TnPAU	TnCK2	TnCK1	TnCK0	TnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **TnPAU**: TMn 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，TM 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 **TnCK2~TnCK0**: TMn 计数器时钟选择位

- 000: $f_{SYS}/4$
- 001: f_H
- 010: $f_H/16$
- 011: $f_H/64$
- 100: f_{TBC}
- 101: f_{TBC}
- 110: TCKn 上升沿时钟
- 111: TCKn 下降沿时钟

此三位用于选择 TM 的时钟源。外部引脚时钟源可被选择在上升沿或下降沿有效。 f_{SYS} 是系统时钟， f_H 和 f_{TBC} 是其它的内部时钟源，细节方面请参考振荡器章节。

Bit 3 **TnON**: TMn 计数器 On/Off 控制位

- 0: Off
- 1: On

此位控制 TM 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 TM。清零此位将停止计数器并关闭 TM 减少耗电。当此位经由低到高转换时，内部计数器将复位清零，当此位由高到低转换时，内部计数器将保持其剩余值直到此位再次转高。

若 TM 处于比较匹配输出模式、PWM 输出模式或单脉冲模式时，当 TnON 位经由低到高转换时，TM 输出脚将复位至 TnOC 位指定的初始值。

Bit 2~0 未定义，读为“0”

TMnC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnCAPTS	TnCCLR
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 **TnM1~TnM0**: TMn 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

这两位为 TM 设置所需工作模式。为确保操作可靠，应在 TnM1 和 TnM0 位发生任何改变之前关闭 TM。在定时 / 计数器模式中，TM 输出引脚控制必须除能。

Bit 5~4 **TnIO1~TnIO0**: 选择 TPn_0, TPn_1 输出功能位

比较匹配输出模式

- 00: 无改变
- 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 TPn_0, TPn_1 或 TCKn 上升沿输入捕捉
- 01: 在 TPn_0, TPn_1 或 TCKn 下降沿输入捕捉
- 10: 在 TPn_0, TPn_1 或 TCKn 双沿输入捕捉
- 11: 输入捕捉除能

定时 / 计数器模式

未使用

这两位用于确定在达到一定条件时 TM 输出脚如何改变状态。此位段所选择的功能取决于 TM 当下的工作模式。

在比较匹配输出模式中，当比较器 A 发生比较匹配时，TnIO1~ TnIO0 决定了 TM 输出脚的状态应如何改变。当比较器 A 发生比较匹配时，TM 输出脚可以设置为切换高，切换低或翻转当前状态。当这两位都为零，则输出将不会发生改变。TM 输出脚的初始值应通过 TMnC1 寄存器中的 TnOC 位进行设置。注意，由 TnIO1 和 TnIO0 位所确定的输出电平必须与利用 TnOC 位所设置的初始值不同，否则在发生比较匹配时 TM 输出脚上将不发生改变。在 TM 输出脚改变状态后，可通过改变 TnON 位的电平从低变高将其复位至初始值。

在 PWM 输出模式，TnIO1~TnIO0 用于决定比较匹配条件发生时 TM 输出脚的状态如何改变。TM 输出功能通过这 2 个位的变化来更新。仅在 TM 关闭后改变 TnIO1~ TnIO0 位的值是很有必要的。若在 TM 运行时改变 TnIO1 和 TnIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **TnOC**: TPn_0, TPn_1 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

此位为 TM 输出脚的输出控制位。其操作取决于 TM 被使用在比较匹配输出模式还是 PWM 输出模式 / 单脉冲输出模式。若 TM 处于定时 / 计数器模式下则不受此位影响。在比较匹配输出模式中，此位决定了在比较匹配发生之前 TM 输出脚的逻辑电平。在 PWM 输出模式中，此位决定了 PWM 信号是为高有效还是低有效。在单脉冲输出模式中，此位决定了当 TnON 位发生从低到高的跳变时 TM 输出脚的逻辑电平。

Bit 2 **TnPOL**: TPn_0, TPn_1 输出极性控制位

- 0: 同相
- 1: 反相

此位控制 TPn_0, TPn_1 输出引脚的极性。当此位置高，TM 输出脚将被反相，而此位为零时则为同相。若 TM 处于定时 / 计数器模式则对其无影响。

Bit 1 **TnCAPTS**: TMn 捕捉触发源选择位
0: 捕捉输入模式的外部时钟源来自 TPn_0, TPn_1 引脚
1: 捕捉输入模式的外部时钟源来自 TCK0 引脚

Bit 0 **TnCCLR**: TMn 计数器清除条件选择位
0: TMn 比较器 P 匹配
1: TMn 比较器 A 匹配

此位用于选择清除计数器的方法。记住周期型 TM 包含两个比较器，即比较器 A 和比较器 P，两者中的任意一个都可以清除内部计数器。当 TnCCLR 位置高，在比较器 A 发生比较匹配时计数器将被清除。当此位为低，计数器将在比较器 P 发生比较匹配或是计数器溢出时被清除。只有在 CCRP 位都被清零时才可执行计数器溢出清除方式。TnCCLR 位在 PWM 输出模式，单脉冲输出模式或输入捕捉模式中未使用。

TMnDL 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**: TMn 计数器低字节寄存器 bit 7~bit 0
TMn 10-bit 计数器 bit 7~bit 0

TMnDH 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**: TMn 计数器高字节寄存器 bit 1~bit 0
TMn 10-bit 计数器 bit 9~bit 8

TMnAL 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**: TMn CCRA 低字节寄存器 bit 7~bit 0
TMn 10-bit CCRA bit 7~bit 0

TMnAH 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**: TMn CCRA 高字节寄存器 bit 1~bit 0
TMn 10-bit CCRA bit 9~bit 8

TMnRPL 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**: TMn CCRP 低字节寄存器 bit 7~bit 0
TMn 10-bit CCRP bit 7~bit 0

TMnRPH 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**: TMn CCRP 高字节寄存器 bit 1~bit 0
TMn 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 TMnC1 寄存器的 TnM1 和 TnM0 位选择任意模式。

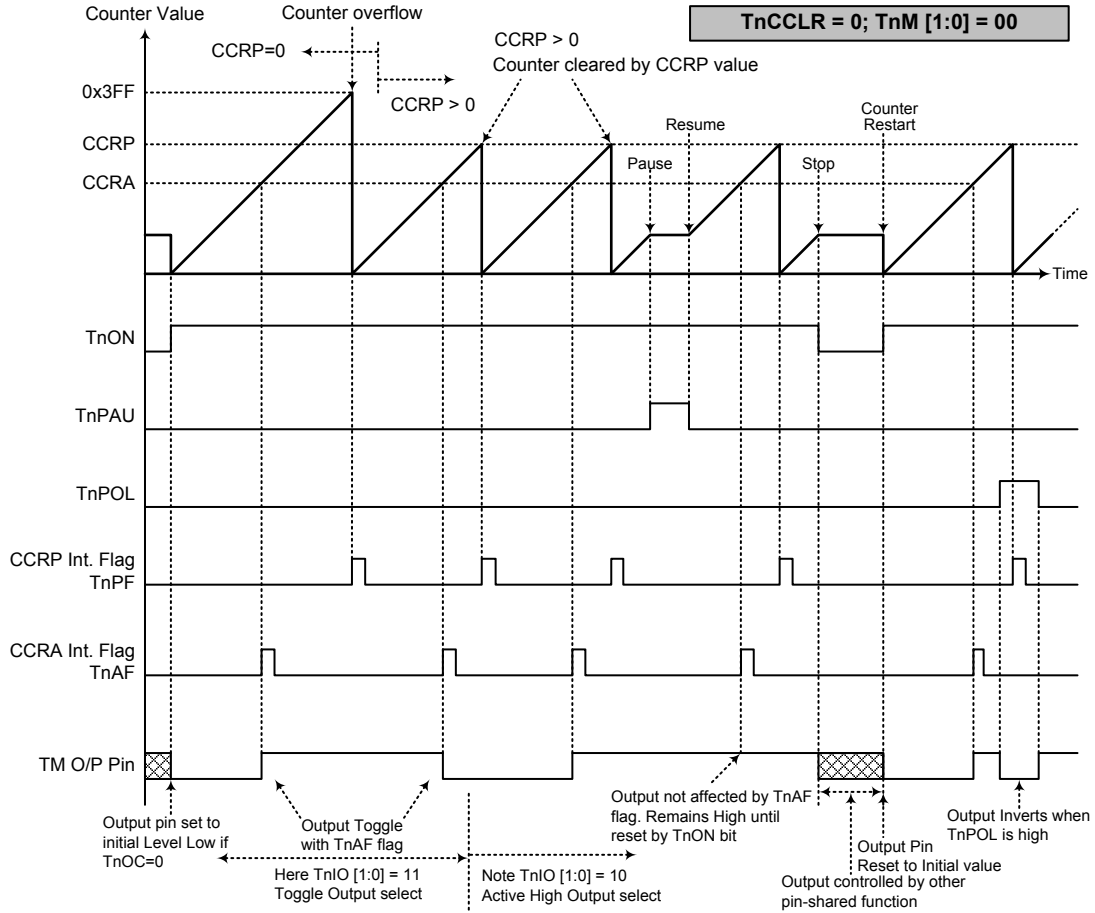
比较匹配输出模式

要工作在此模式，TMnC1 寄存器的 TnM1 和 TnM0 位的值应设为“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 TnCCLR 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 TnAF 和 TnPF 将分别置起。

如果 TMnC1 寄存器的 TnCCLR 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 TnAF 中断请求标志产生。所以当 TnCCLR 为高时，不会产生 TnPF 中断请求标志。在比较匹配输出模式中，CCRA 不能被设置为“0”。

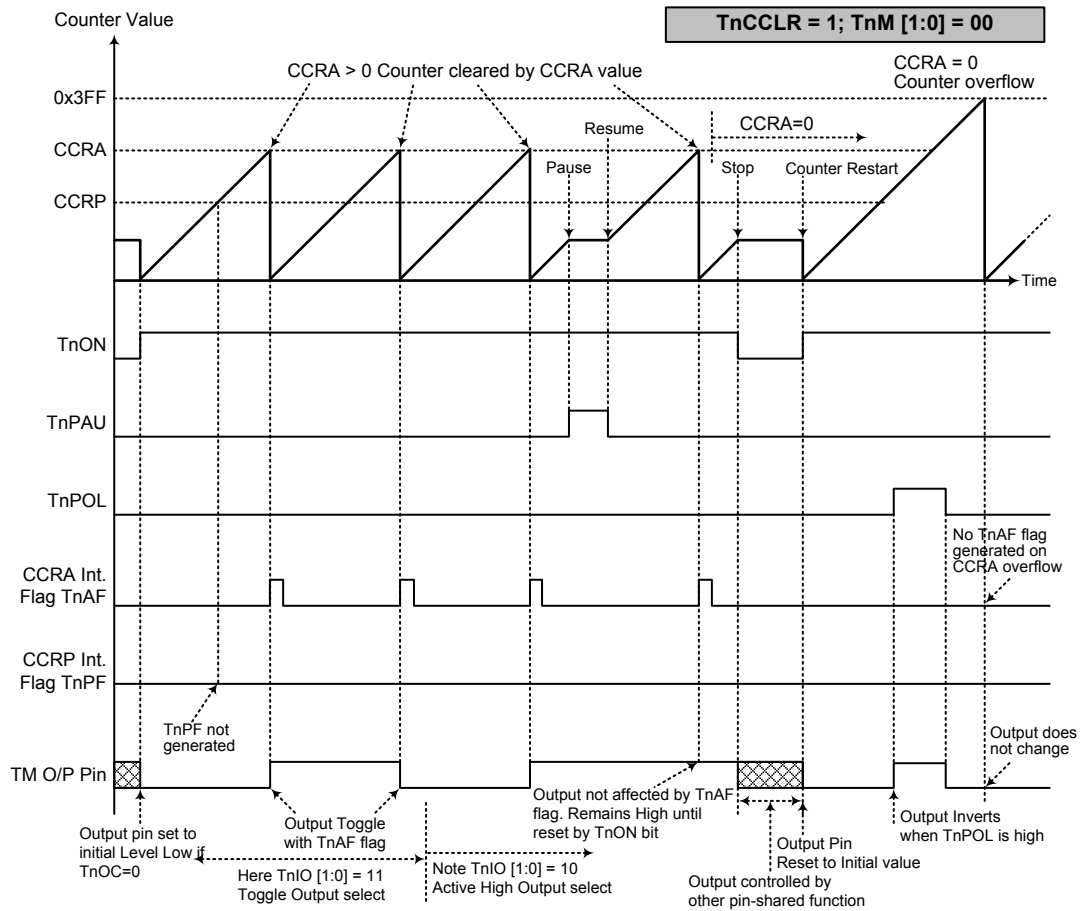
若 CCRA 全为“0”，则计数器将在其最大值 3FF 处溢出。此时不产生 TnAF 中断请求标志位。

正如该模式名所言，当比较匹配发生后，TM 输出脚状态改变。当比较器 A 比较匹配发生后 TnAF 中断请求标志产生时，TM 输出脚状态改变。比较器 P 比较匹配发生时产生的 TnPF 标志不影响 TM 输出脚。TM 输出脚状态改变方式由 TMnC1 寄存器中 TnIO1 和 TnIO0 位决定。当比较器 A 比较匹配发生时，利用 TnIO1 和 TnIO0 位决定 TM 输出脚输出高，低或翻转当前状态。TM 输出脚初始值，在 TnON 位由低到高电平的变化后由 TnOC 位设置。注意，若 TnIO1 和 TnIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 - TnCCLR = 0 (n = 0)

- 注：1. TnCCLR = 0，比较器 P 匹配将清除计数器。
2. TM 输出脚仅由 TnAF 标志位控制。
3. 输出脚在 TnON 位上升沿复位至其初始状态。



比较匹配输出模式 - TnCCLR = 1 (n = 0)

- 注：1. TnCCLR = 1，比较器 A 匹配将清除计数器。
 2. TM 输出脚仅由 TnAF 标志位控制。
 3. 输出脚在 TnON 位上升沿复位至其初始状态。
 4. 当 TnCCLR = 1，不产生 TnPF 标志位。

定时 / 计数器模式

要工作在此模式，TMnC1 寄存器的 TnM1 和 TnM0 位的值需设为“11”。定时 / 计数器模式与比较输出模式操作方式相同，并产生同样的中断请求标志。不同的是，在定时 / 计数器模式下 TM 输出脚未使用。因此，比较匹配输出模式中的描述和时序图可以适用于此功能。该模式中未使用的 TM 输出脚用作普通 I/O 脚或其它引脚共用功能。

PWM 输出模式

要工作在此模式，TMnC1 寄存器的 TnM1 和 TnM0 位的值需设置为“10”。TM 的 PWM 功能在马达控制，加热控制，照明控制等方面十分有用。给 TM 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就极其灵活。在 PWM 输出模式中，TnCCLR 位对 PWM 周期无效。CCRA 和 CCRP 寄存器共同产生 PWM 波形，一个用来清除内部计数器从而控制 PWM 波形频率，另一个则用于控制占空比。因此，PWM 波形频率和占空比可由 CCRA 和 CCRP 寄存器中的值来控制。

当比较器 A 或比较器 P 发生比较匹配时，将产生 CCRA 和 CCRP 的中断标志位。TMnC1 寄存器的 TnOC 位用于选择 PWM 波形所需极性，而 TnIO1 和 TnIO0 位用于使能 PWM 输出或将 TM 输出脚强制固定为高或低电平。TnPOL 位用于取反 PWM 输出波形的极性。

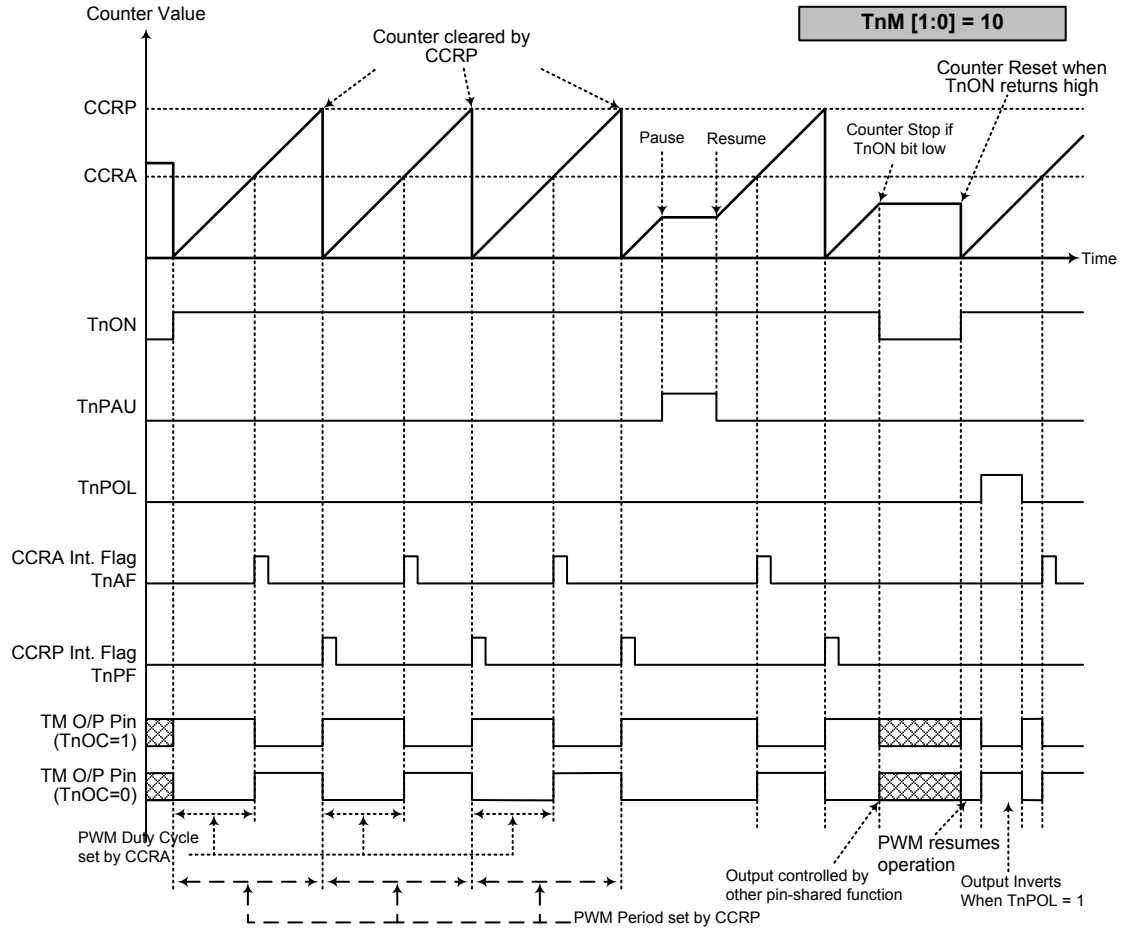
● 10-bit PTM, PWM 输出模式, 边沿对齐模式

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

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

TM PWM 输出频率 = $(f_{SYS}/4)/512 = f_{SYS}/2048 = 3.90625\text{kHz}$ ，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%。



PWM 输出模式 (n = 0)

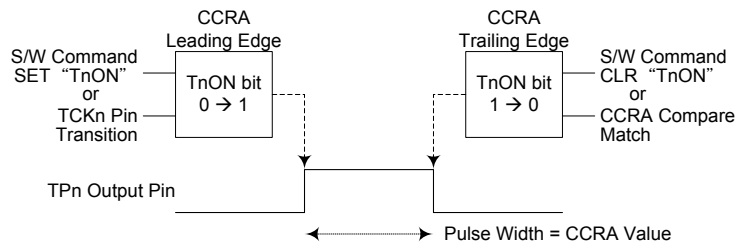
- 注：1. CCRP 将清除计数器。
2. 计数器清除，设置 PWM 周期。
3. 即使在 TnIO[1:0] = 00 或 01 时，内部 PWM 功能继续运行。
4. TnCCLR 位对 PWM 操作无影响。

单脉冲输出模式

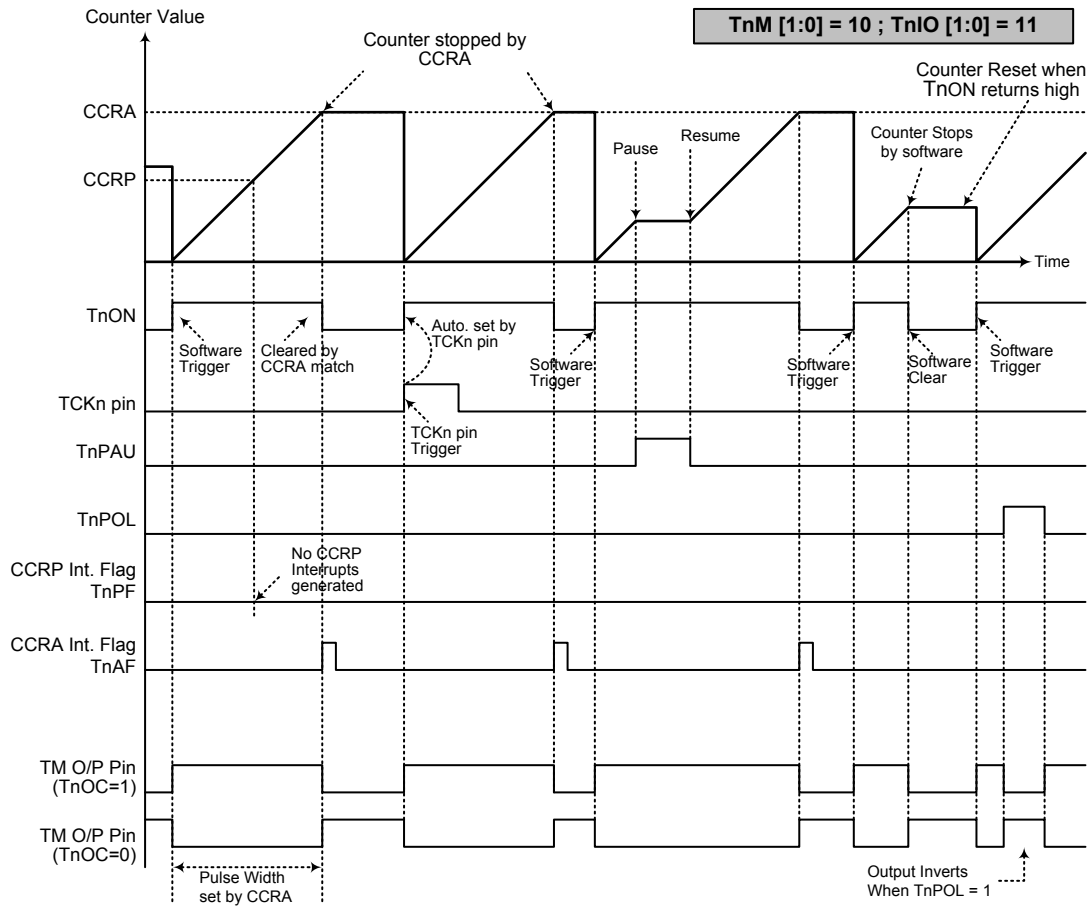
要工作在此模式，TMnC1 寄存器的 TnM1 和 TnM0 位的值需设置为“10”，且 TnIO1 和 TnIO0 位的值也需设置为“11”。正如模式名所言，单脉冲输出模式，在 TM 输出脚将产生一个脉冲输出。

通过应用程序控制 TnON 位由低到高的转变来触发脉冲前沿输出。但在单脉冲模式中，TnON 位可利用外部 TCKn 引脚有效边沿跳转时自动由低转变为高，进而启动单脉冲输出。当 TnON 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 TnON 位保持高电平。通过应用程序使 TnON 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 TnON 位并产生单脉冲输出后沿。CCRA 的值通过这种方式控制脉冲宽度。比较器 A 比较匹配发生时，会产生 TM 中断。TnON 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。在单脉冲模式中，CCRP 寄存器和 TnCCLR 位未使用。



单脉冲产生示意图 (n = 0)



单脉冲输出模式 (n = 0)

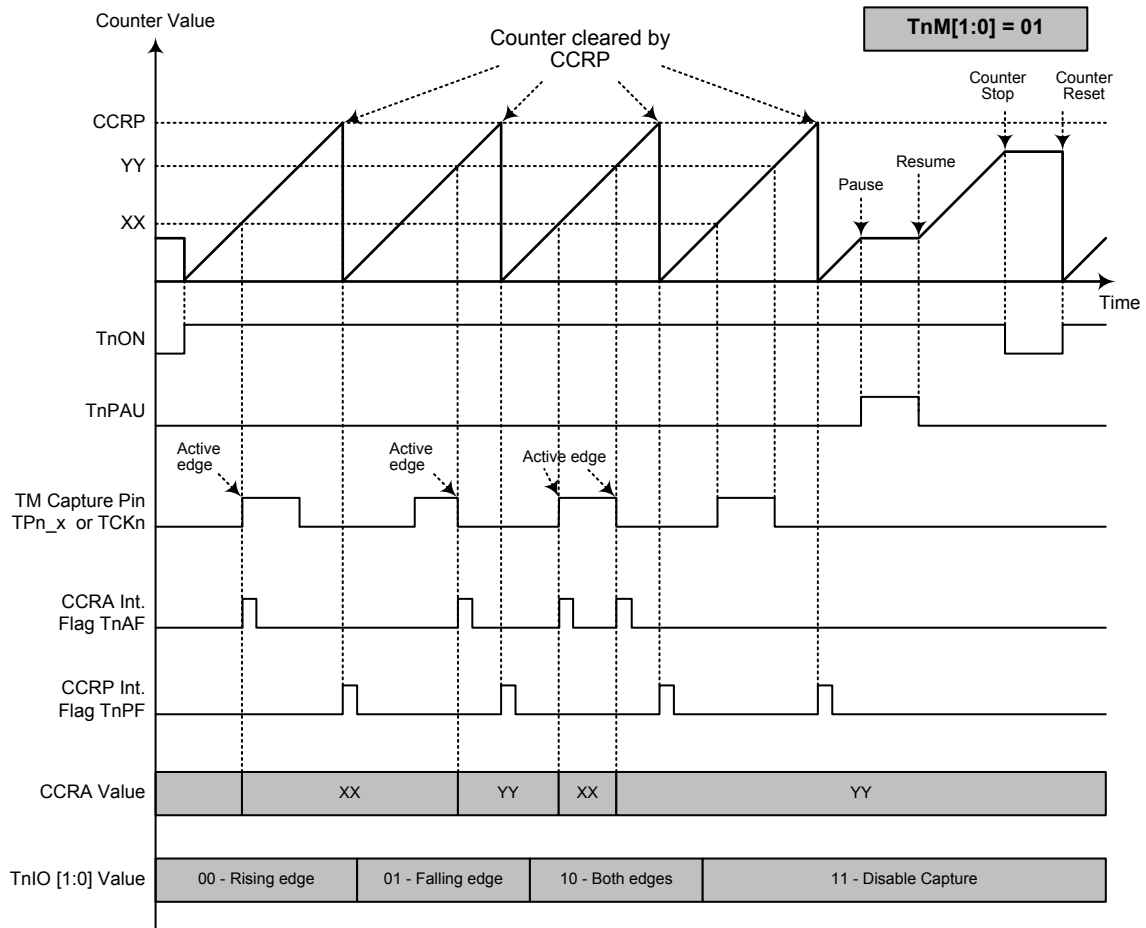
- 注：1. CCRA 将停止计数器。
 2. CCRP 未使用。
 3. 脉冲由 TCKn 引脚或设定 TnON 位为高电平触发。
 4. TCKn 引脚有效边沿将自动将 TnON 位置高。
 5. 在单脉冲模式中，TnIO[1:0] 必须设置为“11”且保持不变。

捕捉输入模式

要工作在此模式，TMnC1 寄存器的 TnM1 和 TnM0 位的值需设置为“01”。此模式使能外部信号捕捉并保存内部计数器当前值，因此被用于诸如脉冲宽度测量的应用中。TPn_0, TPn_1 和 TCKn 引脚上的外部信号可通过设置 TMnC1 寄存器的 TnCPTS 位进行选择。输入引脚有效边沿可为上升沿，下降沿或双沿有效。有效边沿转换类型可通过设置 TMnC1 寄存器的 TnIO1 和 TnIO0 位进行选择。当利用应用程序启动 TnON 位由低到高转变时，计数器开始运行。

当 TPn_0, TPn_1 和 TCKn 引脚出现所需边沿转换时，计数器当前值被锁存到 CCRA 寄存器，并产生 TM 中断。无论 TPn_0, TPn_1 和 TCKn 引脚发生什么事件，计数器继续工作直到 TnON 位发生下降沿跳变。当 CCRP 比较匹配发生时计数器复位至零；CCRP 的值通过这种方式控制计数器的最大值。当比较器 P CCRP 比较匹配发生时，也会产生 TM 中断。计数 CCRP 溢出中断信号的值可以测量长脉宽。如果 TnIO1 和 TnIO0 位都设为高，则无论 TPn_0, TPn_1 和 TCKn 引脚发生什么事件都不会产生捕捉操作，但必须注意，计数器将继续运行。

由于 TPn_0, TPn_1 和 TCKn 引脚与其它功能共用，则 TM 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。TnCCLR, TnOC 和 PTnPOL 位在此模式中未使用。



捕捉输入模式 (n = 0)

- 注：1. TnM[1:0] = 01，有效边沿由 TnIO[1:0] 位段设置。
 2. TM 捕捉输入引脚有效边沿将计数器值传送到 CCRA。
 3. TnCCLR 位未使用。
 4. 无输出功能 - TnOC 和 TnPOL 位未使用。
 5. CCRP 决定计数器的值，当 CCRP 等于零时计数器有一个最大计数值。

A/D 转换器

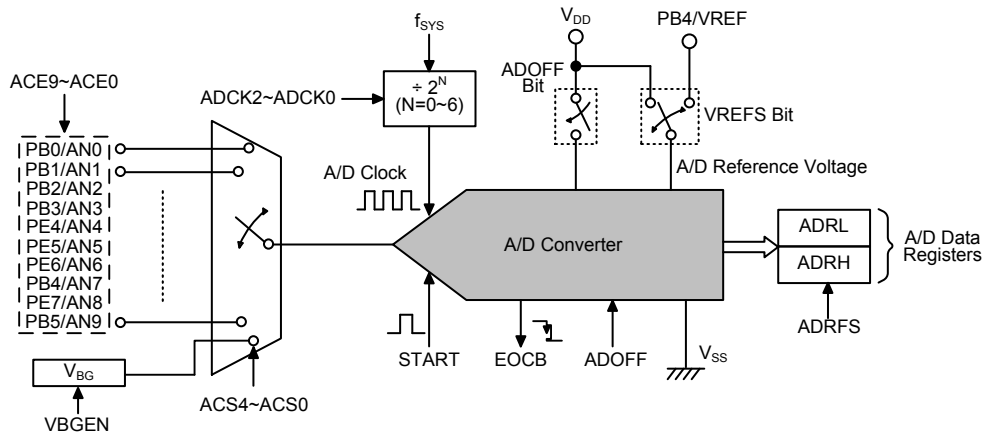
对于大多数电子系统而言，处理现实世界的模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先需要通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

A/D 转换器简介

该单片机包含一个多通道的 A/D 转换器，它们可以直接接入外部模拟信号（来自传感器或其它控制信号）并直接将这些信号转换成 12-bit 数字量。

输入通道	A/D 通道选择位	输入引脚
10	ACS4, ACS3~ACS0	AN0~AN9

下面框图显示了 A/D 转换器内部结构及其相关寄存器。



A/D 转换器结构

A/D 转换器寄存器介绍

A/D 转换器的所有工作由六个寄存器控制。一对只读寄存器来存放 12-bit A/D 转换数据的值，剩下的四个控制寄存器设置 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
ADRL (ADRFS = 0)	D3	D2	D1	D0	—	—	—	—
ADRL (ADRFS = 1)	D7	D6	D5	D4	D3	D2	D1	D0
ADRH (ADRFS = 0)	D11	D10	D9	D8	D7	D6	D5	D4
ADRH (ADRFS = 1)	—	—	—	—	D11	D10	D9	D8
ADCR0	START	EOCB	ADOFF	ADRFS	ACS3	ACS2	ACS1	ACS0
ADCR1	ACS4	VBGEN	—	VREFS	—	ADCK2	ADCK1	ADCK0
ACERL	ACE7	ACE6	ACE5	ACE4	ACE3	ACE2	ACE1	ACE0
ACERH	—	—	—	—	—	—	ACE9	ACE8

A/D 转换器寄存器列表

A/D 转换器数据寄存器 – ADRL, ADRH

对于具有 12-bit A/D 转换器的单片机，需要两个数据寄存器存放转换结果，一个高字节寄存器 ADRH 和一个低字节寄存器 ADRL。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-bit 中的 12-bit，其数据存储格式由 ADCR0 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”。

ADRFS	ADRH								ADRL							
	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	—	—	—	—
1	—	—	—	—	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D 转换器数据寄存器

A/D 转换器控制寄存器 – ADCR0, ADCR1, ACERL, ACERH

寄存器 ADCR0、ADCR1、ACERL 和 ACERH 用来控制 A/D 转换器的功能和操作。这些 8-bit 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 转换器时钟源，并控制和监测 A/D 转换器的开始和转换结束状态。寄存器 ADCR0 的 ACS[3:0] 位和 ADCR1 的 ACS4 位定义 A/D 转换器外部输入通道编号。由于每个单片机只包含一个实际的模数转换硬件电路，因此模拟信号中的每一个都需要分别被发送到转换器。由 ACS4~ACS0 位段决定模拟通道引脚或内部 V_{BG} 实际连接到内部 A/D 转换器。

ACERL 和 ACERH 寄存器中的 ACE9~ACE0 位段用来定义 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。若相关位置高，选择该引脚作为 A/D 转换器输入；若相关位清零，该引脚作为通用 I/O 或其它引脚共用功能。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。

● ADCR0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	START	EOCB	ADOFF	ADRFS	ACS3	ACS2	ACS1	ACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	1	0	0	0	0	0

Bit 7 START: 启动 A/D 转换位
 0 → 1 → 0: 启动 A/D 转换
 0 → 1: A/D 转换器复位，且 EOCB 置高
 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，将启动 A/D 转换过程。当此位置高，A/D 转换器被复位。

Bit 6 EOCB: A/D 转换结束标志位
 0: A/D 转换结束
 1: A/D 转换中
 该只读标志位用于表明 A/D 转换过程是否完成。当 A/D 转换时，此位被置高。

Bit 5 ADOFF: A/D 转换器电源 On/Off 控制位
 0: A/D 转换器电源开启
 1: A/D 转换器电源关闭
 此位控制 A/D 内部功能的电源。该位需清零以使能 A/D 转换器。如果该位设为高将关闭 A/D 转换器以降低功耗。A/D 转换器即使没有转换，仍然会产生功耗。因此此举在功耗敏感的应用中值得考虑。
 注：1. 在单片机进入空闲 / 休眠模式前，建议将 ADOFF 设置为 1 以节省耗电。
 2. ADOFF = 1 时将关闭 A/D 转换器。

- Bit 4 **ADRF5**: A/D 转换器数据格式控制位
 0: A/D 转换器数据格式 → ADRL = D[11:4]; ADRL = D[3:0]
 1: A/D 转换器数据格式 → ADRL = D[11:8]; ADRL = D[7:0]
 此位控制存放在两个 A/D 数据寄存器中的 12-bit A/D 转换结果的格式。细节方面请参考 A/D 转换器数据寄存器章节。
- Bit 3~0 **ACS3~ACS0**: A/D 转换器通道选择位 (当 ACS4 为 “0” 时)
 0000: AN0
 0001: AN1
 0010: AN2
 0011: AN3
 0100: AN4
 0101: AN5
 0110: AN6
 0111: AN7
 1000: AN8
 1001~1111: AN9
 这些位用于选择要转换的输入通道。由于每个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。如果 ACS4 置高, 内部 Bandgap V_{BG} 信号被连接到内部 A/D 转换器。

● **ADCR1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ACS4	VBGEN	—	VREFS	—	ADCK2	ADCK1	ADCK0
R/W	R/W	R/W	—	R/W	—	R/W	R/W	R/W
POR	0	0	—	0	—	0	0	0

- Bit 7 **ACS4**: 内部 V_{BG} 作为 A/D 转换器输入信号控制位
 0: 除能
 1: 使能
 当此位置高, V_{BG} 连接到 A/D 转换器。VBGEN 需先设定以使能 bandgap 电路的 V_{BG} 电压用于 A/D 转换器。当此位置高, 内部 Bandgap V_{BG} 信号被连接到内部 A/D 转换器, 且其它 A/D 输入通道不连接。
- Bit 6 **VBGEN**: 内部 V_{BG} 控制位
 0: 除能
 1: 使能
 该位用于控制 A/D 转换器的内部 Bandgap 电路开启 / 关闭。当此位置高, Bandgap 电压 V_{BG} 用于 A/D 转换器。当 LVR/LVD 除能时, 如果 V_{BG} 不作为 A/D 转换器输入, Bandgap 参考电路将自动关闭以降低功耗。如果 V_{BG} 使能且作为 A/D 转换器输入, 实现 A/D 转换前, Bandgap 电路稳定需要一定的延时 t_{BGS} 。
- Bit 5 未定义, 读为 “0”
- Bit 4 **VREFS**: 选择 A/D 转换器参考电压
 0: 内部 A/D 转换器电源
 1: 外部 VREF 引脚
 此位用于选择 A/D 转换器参考电压。如果此位置高, 选择外部 VREF 引脚作为 A/D 转换器参考电压。如果此位置低, 选择电源 VDD 引脚作为 A/D 转换器参考电压。VREF 引脚与其它功能引脚共用, 当外部 VREF 引脚作为 A/D 转换器参考电压时, VREF 引脚的其它共用功能需除能。
- Bit 3 未定义, 读为 “0”

- Bit 2~0 **ADCK2~ADCK0**: 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: 未定义

这三位用于选择 A/D 转换器的时钟源。

ACS4	VBGEN	LVR/LVD	VBG	Bandgap 参考电压
x	0	使能	Off 连接到 GND	On
x	0	除能	Off 连接到 GND	Off
x	1	x	On	On

“x”：无关

Bandgap 参考电压 On/Off 真值表

● **ACERL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ACE7	ACE6	ACE5	ACE4	ACE3	ACE2	ACE1	ACE0
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 **ACE7**: 定义 PB4 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN7
- Bit 6 **ACE6**: 定义 PE6 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN6
- Bit 5 **ACE5**: 定义 PE5 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN5
- Bit 4 **ACE4**: 定义 PE4 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN4
- Bit 3 **ACE3**: 定义 PB3 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN3
- Bit 2 **ACE2**: 定义 PB2 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN2
- Bit 1 **ACE1**: 定义 PB1 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN1
- Bit 0 **ACE0**: 定义 PB0 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN0

• ACERH 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	ACE9	ACE8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

- Bit 7~2 未定义，读为“0”
- Bit 1 **ACE9**: 定义 PB5 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN9
- Bit 0 **ACE8**: 定义 PE7 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN8

A/D 转换器操作

ADCR0 寄存器中的 START 位用于启动 A/D 转换。当单片机设定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。当 START 位从逻辑低到逻辑高，然后没有再到逻辑低，ADCR0 寄存器中的 EOCB 位将会置高且 A/D 转换器发生复位。START 位用于控制 A/D 转换器整体的开启操作。

ADCR0 寄存器中的 EOCB 位用于表明模数转换过程是否完成。在转换周期结束后，EOCB 位会被清零。此外，中断控制寄存器内相应的 A/D 中断请求标志位也会置位，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到相应的 A/D 内部中断入口。如果 A/D 内部中断被除能，可以让单片机轮询 ADCR0 寄存器中的 EOCB 位，检测此位是否被清除，以作为另一种侦测 A/D 转换周期结束的方法。

A/D 转换器的时钟源是源于系统时钟 f_{SYS} 。虽然 A/D 时钟源是由系统时钟 f_{SYS} 和 ADCR1 寄存器中 ADCK[2:0] 位决定。但可选择的最大 A/D 时钟源还有一些限制。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必须小心。如果系统时钟速度为 4MHz 时，ADCK[2:0] 位不能设为“000”或“110”。否则所得的 A/D 时钟周期将小于最小时钟周期或大于最大时钟周期，导致所得 A/D 转换结果不准确。

参考如下表，其中用星号 * 标示的值取决于单片机，必须特别注意，这些值可能超出定义的 A/D 时钟周期范围。

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

A/D 时钟周期范例

ADCR0 寄存器的 ADOFF 位用于控制 A/D 转换电路电源的开启 / 关闭。该位必须清零以开启 A/D 转换器电源。当设置 ADOFF 位清零开启 A/D 转换器内部电路时，在 A/D 转换成功初始化前需一段延时。如时序图所示。即使通过设置 ACERL 和 ACERH 的 ACE9~ACE0 控制位，选择无引脚作为 A/D 输入，如果 ADOFF 设为“0”，那么仍然会产生功耗。因此当未使用 A/D 转换器功能时，在功耗敏感的应用中建议设置 ADOFF 为高以减少功耗。

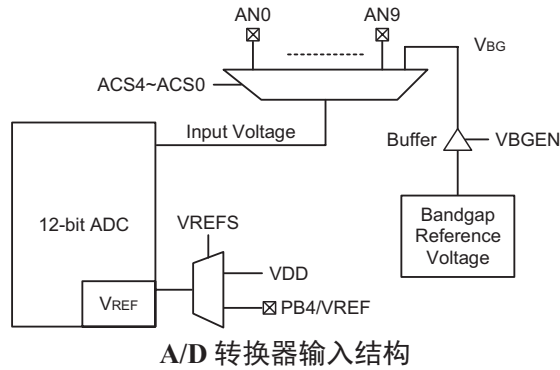
A/D 转换器参考电压

A/D 转换器的参考电压可以来自正电源引脚 VDD 或来自 VREF 引脚上的外部参考源。通过配置 VREFS 位进行选择。由于 VREF 引脚与其它功能共用，当 VREFS 设置为高时，VREF 引脚被选定并自动除能其它引脚功能。

A/D 转换器输入信号

所有的 A/D 模拟输入引脚都与 I/O 引脚及其它功能共用。使用 ACERL 和 ACERH 寄存器中的 ACE9~ACE0 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 ACE9~ACE0 位设置为高对应的引脚作为 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。

A/D 转换器的参考电压可以来自正电源引脚 VDD 或来自 VREF 引脚上的外部参考源。通过配置 ADCR1 寄存器中的 VREFS 位进行选择。模拟输入值不得超过 V_{REF} 的值。

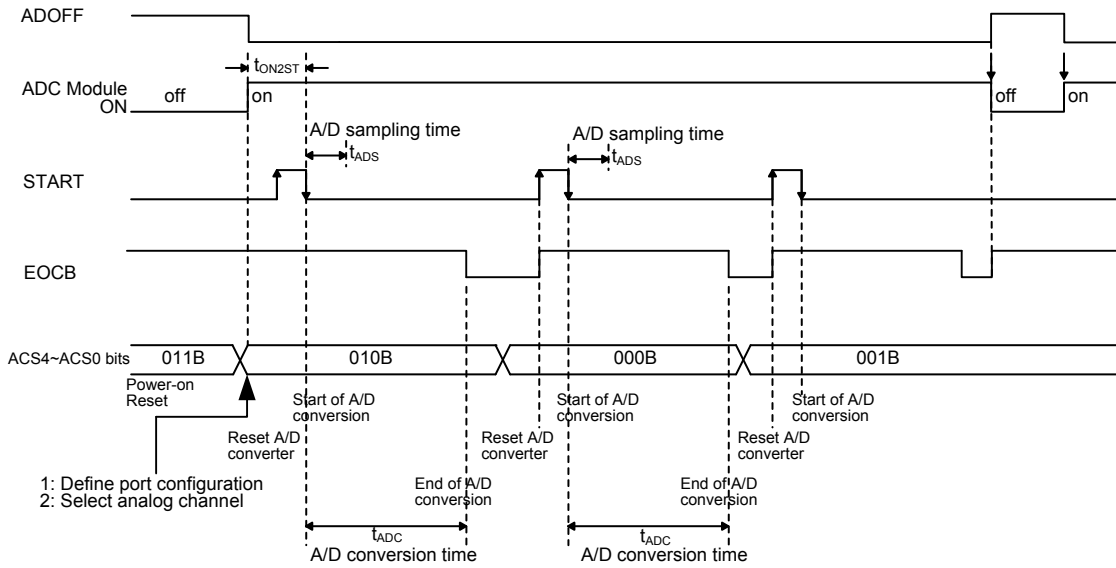


A/D 转换率和时序图

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 4 个 A/D 时钟周期，而数据转换需 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 16 个 t_{ADCK} ， t_{ADCK} 为 A/D 时钟周期。



A/D 转换时序

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 ADCR1 寄存器中的 ADCK [2:0] 位，选择所需的 A/D 转换器转换时钟。
- 步骤 2
将 ADCR0 寄存器中的 ADOFF 位清零以使能 A/D 转换器。
- 步骤 3
通过配置 ADCR1 和 ADCR0 寄存器的 ACS4~ACS0 位段，选择连接至内部 A/D 转换器的通道。
- 步骤 4
通过配置 ADERH 和 ACERL 寄存器的 ACE9~ACE0 位段，选择连接至内部 A/D 转换器的引脚。
- 步骤 5
如果要使用 A/D 转换中断，则中断控制寄存器需要正确地设置，以确保 A/D 中断功能是有用的。总中断控制位 EMI 以及 A/D 转换器中断位 ADE 需要预先置位为“1”。
- 步骤 6
现在可以通过设定 ADCR0 寄存器中的 START 位从低到高再到低，开始 A/D 转换的过程。注意，此位的初始状态为零。
- 步骤 7
可以让单片机轮询 ADCR0 寄存器中的 EOCB 位，以作为侦测 A/D 转换周期结束的方法。EOCB 位清零。A/D 转换结束，并可从 ADRH 和 ADRL 寄存器读取输出数据。另一种方法是，若中断使能且堆栈未滿，则程序等待 A/D 中断发生。

注：若使用轮询 ADCR0 寄存器中的 EOCB 位的方法以检测模数转换过程是否完成，上述中断使能步骤可以忽略。

编程注意事项

在单片机工作时，如果 A/D 转换器未使用，通过设置 ADCR0 寄存器中的 ADOFF 为高，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

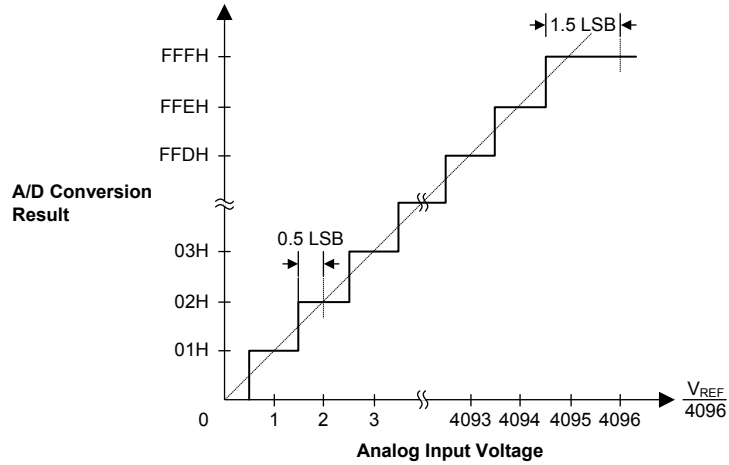
该单片机含有一个 12-bit A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于实际 A/D 转换器参考电压 V_{REF} ，因此每一位可表示 $V_{REF}/4096$ 的模拟输入值。

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

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。应注意的是， V_{REF} 为实际进入 A/D 转换器的参考电压，由 VREFS 位进行选择。



理想的 A/D 转换功能

A/D 程序范例

下面两个范例程序用来说明怎样设置和实现 A/D 转换。第一个范例是轮询 ADCR0 寄存器中的 EOCB 位来判断 A/D 转换何时完成，而第二个范例则使用中断的方式判断。

范例 1: 使用轮询 EOCB 的方式来检测转换结束

```

clr ADE                ; disable A/D converter interrupt
mov a, 03H
mov ADCR1, a           ; select fsys/8 as A/D clock and switch off VBG
clr ADOFF
mov a, 0Fh            ; setup ACERL to configure pins AN0~AN3
mov ACERL, a
mov a, 00h
mov ACERH, a
mov a, 00h
mov ADCR0, a         ; enable and connect AN0 channel to A/D converter
:
:
Start_conversion:
clr START
set START            ; reset A/D
clr START            ; start A/D
Polling_EOC:
sz EOCB              ; poll the ADCR0 register EOCB bit to detect end
                    ; of A/D conversion
jmp polling_EOC      ; continue polling
mov a, ADRL           ; read low byte conversion result value
mov adrl_buffer, a   ; save result to user defined register
mov a, ADRH           ; read high byte conversion result value
mov adrh_buffer, a   ; save result to user defined register
:
jmp start_conversion ; start next A/D conversion
    
```

注：ADOFF 位设为“1”将关闭 A/D 转换器。

范例 2：使用中断的方式来检测转换结束

```

clr ADE                ; disable A/D converter interrupt
mov a, 03H
mov ADCR1, a          ; select fsys/8 as A/D clock and switch off VBG
clr ADOFF
mov a, 0Fh            ; setup ACERL to configure pins AN0~AN3
mov ACERL, a
mov a, 00h
mov ACERH, a
mov a, 00h
mov ADCR0, a          ; enable and connect AN0 channel to A/D converter
:
:
Start_conversion:
clr START
set START              ; reset A/D
clr START              ; start A/D
clr ADF                ; clear A/D converter interrupt request flag
set ADE                ; enable A/D converter interrupt
set EMI                ; enable global interrupt
:
:
; A/D converter 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, ADRL            ; read low byte conversion result value
mov adrl_buffer, a   ; save result to user defined register
mov a, ADRH            ; read high byte conversion result value
mov adrh_buffer, a   ; save result to user defined register
:
:
EXIT_ISR:
mov a, status_stack
mov STATUS, a         ; restore STATUS from user defined memory
mov a, acc_stack      ; restore ACC from user defined memory
clr ADF                ; clear A/D converter interrupt flag
reti

```

注：ADOFF 位设为“1”将关闭 A/D 转换器。

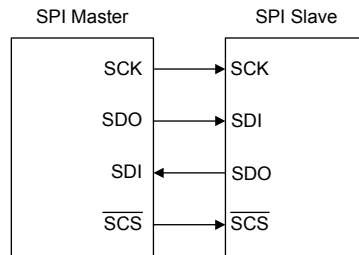
串行接口模块 – SIM

该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。SIM 接口的引脚与其它 I/O 引脚共用，所以要使用 SIM 功能前应设置相关引脚共用功能选择位。因为这两种接口共用引脚和寄存器，所以要通过一个 SIMC0 寄存器中的 SIM2~SIM0 位来选择哪一种通信接口。如果 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 脚的上拉电阻。

SPI 接口

SPI 接口常用于与外部设备如传感器或闪存等通信。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 \overline{SCS} 引脚。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 主 / 从机连接方式

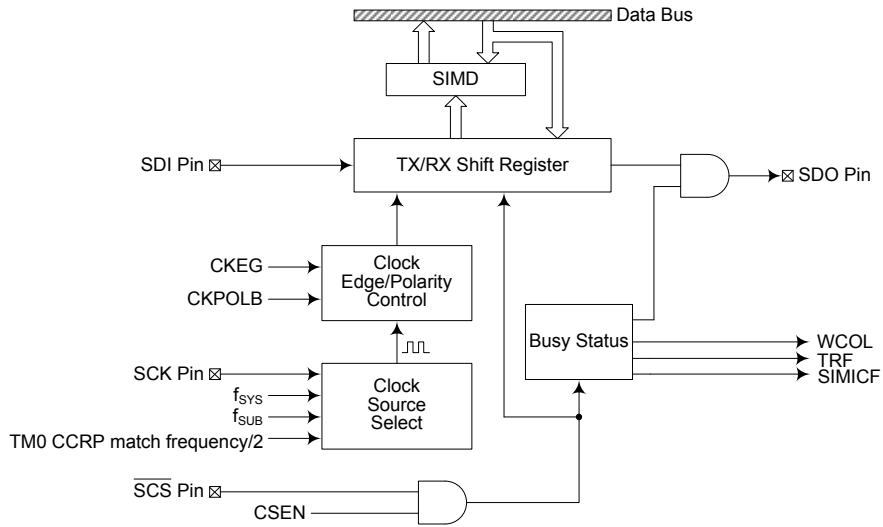
SPI 接口操作

SPI 接口是一个全双工串行数据传输器。SPI 接口的四线为：SDI、SDO、SCK 和 \overline{SCS} 。SDI 和 SDO 是数据的输入和输出线。SCK 是串行时钟线， \overline{SCS} 是从机的选择线。SPI 的接口引脚与普通 I/O 口和 I²C 的功能脚共用，须预先设定 SIMC0 和 SIMC2 寄存器的对应位。通过 SIMC0 寄存器的 SIMEN 位控制 SPI 接口的除能或使能。连接到 SPI 接口的单片机以从主 / 从模式进行通信，且主机完成所有的数据传输初始化，并控制时钟信号。由于单片机只有一个 \overline{SCS} 引脚，所以只能拥有一个从机。可通过软件控制 \overline{SCS} 引脚使能与除能，设置 CSEN 位为“1”使能 \overline{SCS} 功能，设置 CSEN 位为“0”， \overline{SCS} 引脚将处于浮空状态。

该单片机的 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 和两个控制寄存器 SIMC0, SIMC2。注意，SIMC1 寄存器仅用于 I²C 接口功能。

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

SPI 寄存器列表

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

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 接口功能中的的寄存器 SIMA 是同一个寄存器。SPI 功能不会用到寄存器 SIMC1，SIMC1 只适用于 I²C 中。寄存器 SIMC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 SIMC2 用于其它的控制功能如 LSB/MSB 选择，写冲突标志位等。

● SIMC0 寄存器

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

Bit 7~5 **SIM2~SIM0: SIM 工作模式控制位**

- 000: SPI 主机模式, SPI 时钟为 $f_{SYS}/4$
- 001: SPI 主机模式, SPI 时钟为 $f_{SYS}/16$
- 010: SPI 主机模式, SPI 时钟为 $f_{SYS}/64$
- 011: SPI 主机模式, SPI 时钟为 f_{SUB}
- 100: SPI 主机模式, SPI 时钟为 TM0 CCRP 匹配频率 /2
- 101: SPI 从机模式
- 110: I²C 从机模式
- 111: 非使用模式

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 的主时钟频率及 I²C 或 SPI 功能。SPI 时钟源可来自于系统时钟也可以选择来自 TM0 或 f_{SUB} 。若选择的是作为 SPI 从机, 则其时钟源从外部主机而得。

Bit 4 未定义, 读为“0”

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

此位段在 SPI 模式中未使用, 当工作在 SPI 模式时, 请忽略该位段。

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 脚处于浮空状态, 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: 未发生 SIM SPI 未完成现象
- 1: 发生 SIM SPI 未完成现象

此位仅在 SIM 工作在 SPI 从机模式时有效。若 SPI 工作在从机模式且 SIMEN 和 CSEN 都置高, 但 SCS 在 SPI 数据传输结束前由外部主机拉高, 则 SIMICF 与 TRF 都将被置位。此时若中断功能使能则将发生中断。但若此位由应用程序软件置位, TRF 位将不会被置“1”。

• SIMC2 寄存器

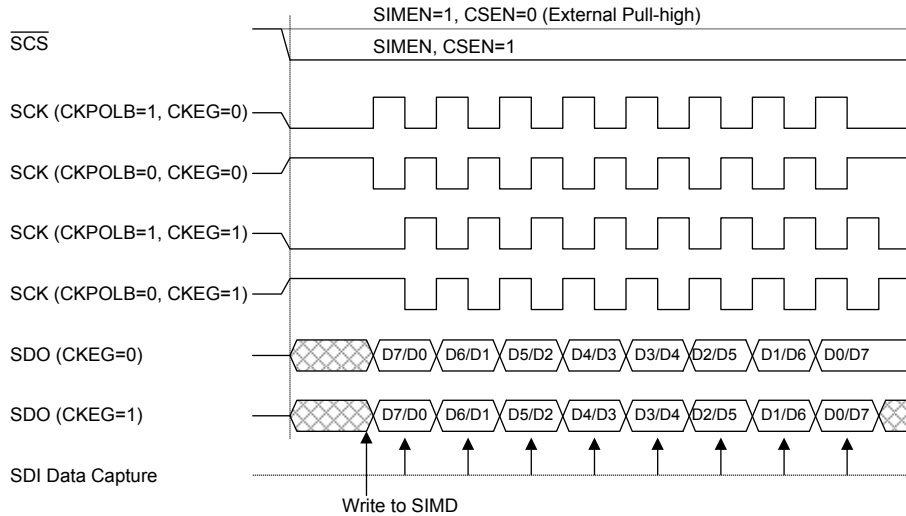
Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7~6 **D7~D6**: 未定义位
用户可通过应用程序对着两位进行读写。
- Bit 5 **CKPOLB**: SPI 时钟线的基础状态位
0: 当时钟无效时, SCK 口为高电平
1: 当时钟无效时, SCK 口为低电平
此位决定了时钟线的基础状态, 当时钟无效时, 若此位为高, SCK 为低电平。若此位为低, SCK 为高电平。
- Bit 4 **CKEG**: SPI SCK 有效时钟边沿类型位
CKPOLB = 0
0: SCK 为高电平且在 SCK 上升沿抓取数据
1: SCK 为高电平且在 SCK 下降沿抓取数据
CKPOLB = 1
0: SCK 为低电平且在 SCK 下降沿抓取数据
1: SCK 为低电平且在 SCK 上升沿抓取数据
CKEG 和 CKPOLB 位用于设置 SPI 总线上时钟信号输入和输出方式。在执行数据传输前, 这两位必须被设置, 否则将产生错误的时钟边沿信号。CKPOLB 位决定时钟线的基础状态, 若时钟无效且此位为高, 则 SCK 为低电平, 若时钟无效且此位为低, 则 SCK 为高电平。CKEG 位决定有效时钟边沿类型, 取决于 CKPOLB 的状态。
- Bit 3 **MLS**: SPI 数据移位命令位
0: LSB 优先传送
1: MSB 优先传送
数据移位选择位, 用于选择数据传输时高位优先传输还是低位优先传输。此位设置为高时高位优先传输, 为低时低位优先传输。
- Bit 2 **CSEN**: SPI \overline{SCS} 引脚控制位
0: 除能
1: 使能
CSEN 位用于 \overline{SCS} 引脚的使能 / 除能控制。此位为低时, \overline{SCS} 除能并处于浮空状态。此位为高时, 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, 数据在传输时被写入 SIMD 寄存器。若数据正在被传输时, 此操作无效。此位可被应用程序清零。
- Bit 0 **TRF**: SPI 发送 / 接收结束标志位
0: 数据正在发送
1: 数据发送结束
TRF 位为发送 / 接收结束标志位, 当 SPI 数据传输结束时, 此位自动置高, 但须通过应用程序设置为“0”。此位也可用于产生中断。

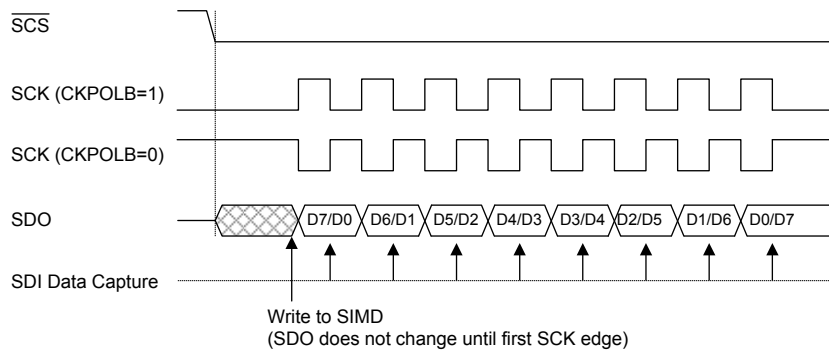
SPI 通信

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

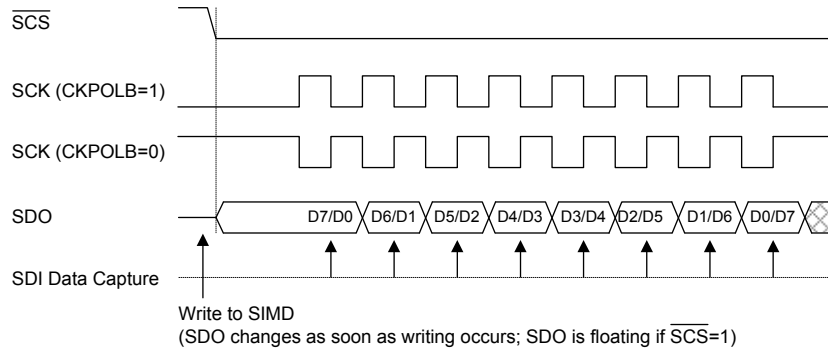
即使在单片机处于空闲模式，SPI 功能仍将继续执行。



SPI 主机模式时序

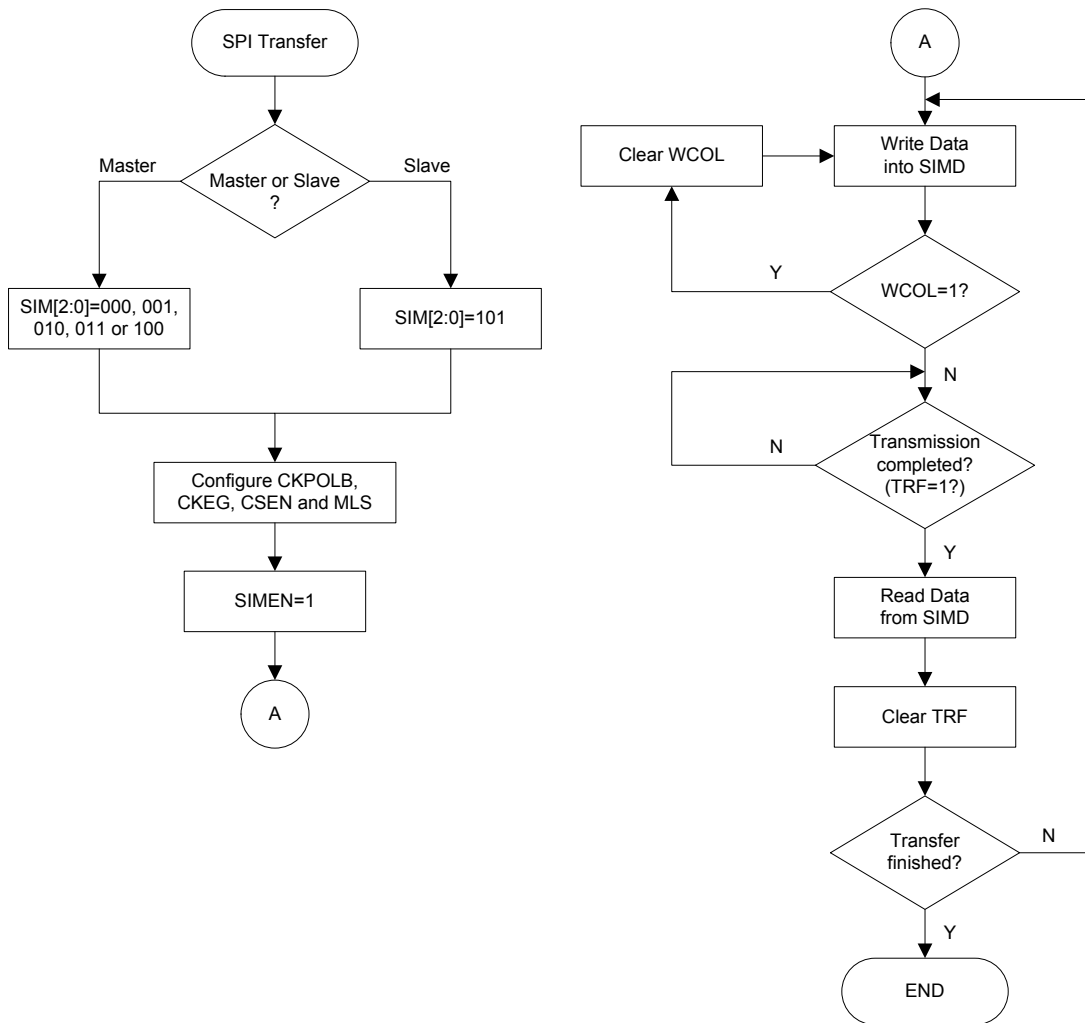


SPI 从机模式时序 - CKEG = 0



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

SPI 从机模式时序 - CKEG = 1



SPI 传输控制流程图

SPI 总线使能 / 除能

使能 SPI 总线，设置 $CSEN = 1$ 和 $\overline{SCS} = 0$ ，然后等待数据写入 SIMD (TXRX 缓存器) 寄存器。对于主机模式，数据写入 SIMD (TXRX 缓存器) 寄存器后，将自动开始数据发送和接收操作。当所有数据都传输完成，TRF 位将置位。对于从机模式，当 SCK 线接收到时钟脉冲时，TXRX 缓存器内的数据将移出或数据移入 SDI 脚。

当 SPI 总线除能时，SCK，SDI，SDO 和 \overline{SCS} 通过相应的引脚共用控制位设置为 I/O 口或其它功能。

SPI 操作

SPI 通过 4 线接口完成主机或从机通信。

SIMC2 寄存器内的 CSEN 位为 SPI 接口通信总的控制位。将该位设为“1”时， \overline{SCS} 线有效，且用于控制 SPI 接口。如果 CSEN 位为低，SPI 接口将除能，且 \overline{SCS} 线处于浮空状态，此时不能用于控制 SPI 接口。如果 SIMC0 寄存器的 CSEN 和 SIMEN 位为高，SDI 将浮空且 SDO 线为高。如果处于主机模式，SCK 线的高低取决于 SIMC2 寄存器的 CKPOLB 位设置的时钟的极性。如果处于从机模式，SCK 将浮空。如果 SIMEN 位为低，总线将除能且 \overline{SCS} ，SDI，SDO 和 SCK 都为 I/O 引脚或其它功能。处于主机模式，时钟信号总是由主机生成。当数据写入 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。处于从机模式，时钟信号将来自于外部主机用于数据的接收和发送。下面介绍主 / 从模式数据传输步骤。

● 主机模式

● 步骤 1

设置 SIMC0 控制寄存器中的 SIM2~SIM0 位，选择时钟源和主机模式。

● 步骤 2

设置 CSEN 和 MLS 位，选择高位或低位数据优先传送，这必须与从机一致。

● 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，数据被存储在 TXRX 缓存器中。再使用 SCK 和 \overline{SCS} 信号线将数据输出。跳至步骤 5。

对于读操作：使用 SDI 信号线将 TXRX 缓存器中的数据移出，并全部锁存至 SIMD 寄存器内。

● 步骤 5

检查 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

检查 TRF 位或等待 SPI 串行总线中断发生。

● 步骤 7

从 SIMD 寄存器读数据。

● 步骤 8

清除 TRF。

● 步骤 9

跳回至步骤 4。

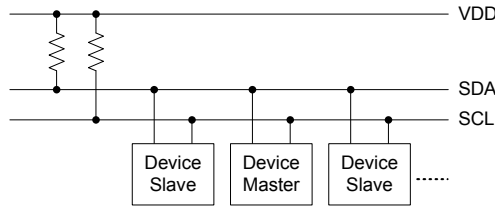
- 从机模式
 - 步骤 1
设置 SIMC0 控制寄存器中的 SIM2~SIM0 位，选择从机模式。
 - 步骤 2
设置 CSEN 和 MLS 位，选择高位或低位数据优先传送，二者必须与主机一致。
 - 步骤 3
设置 SIMC0 控制寄存器中的 SIMEN 位，使能 SPI 接口功能。
 - 步骤 4
对于写操作：写数据到 SIMD 寄存器，实际上，数据被存储在 TXRX 缓存器中。
等待主机时钟信号 SCK 和 \overline{SCS} 信号。跳至步骤 5。
对于读操作：使用 SDI 信号线将 TXRX 缓存器中的数据移出，并全部锁存至 SIMD 寄存器内。
 - 步骤 5
检查 WCOL 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
检查 TRF 位或等待 SPI 串行总线中断发生。
 - 步骤 7
从 SIMD 寄存器读数据。
 - 步骤 8
清除 TRF。
 - 步骤 9
跳回至步骤 4。

错误检测

SIMC2 寄存器中的 WCOL 位用来数据传输期间监测数据冲突的发生。此位由 SPI 串行接口设置为高但必须由应用程序清零。在数据传输期间，如果写数据到 SIMD 寄存器，此时数据冲突发生且不允许数据继续被写入。

I²C 接口

I²C 可以和传感器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。

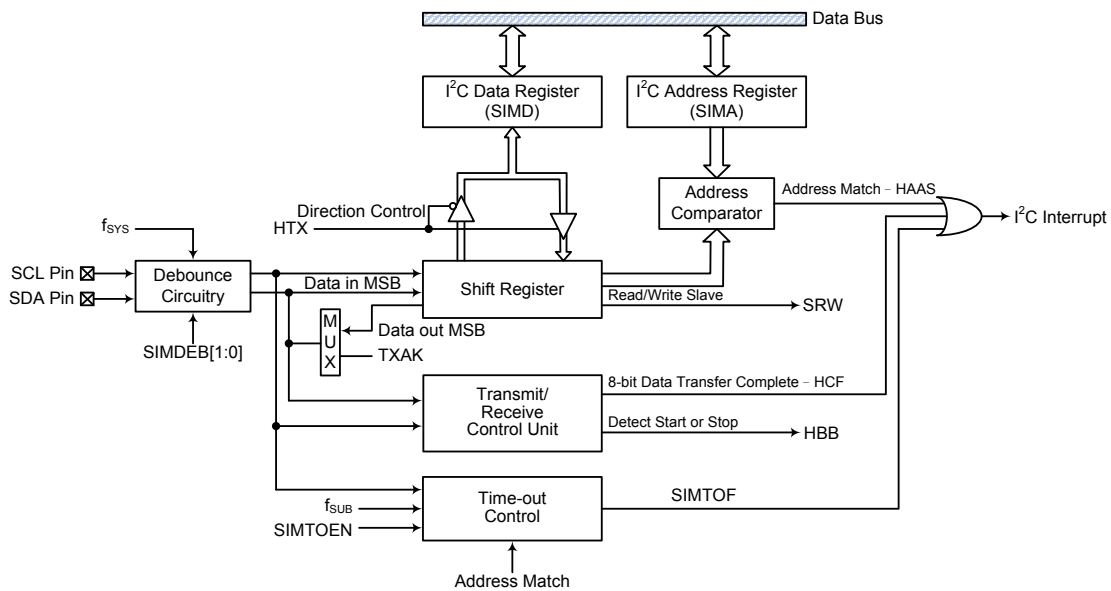


I²C 主从总线连接图

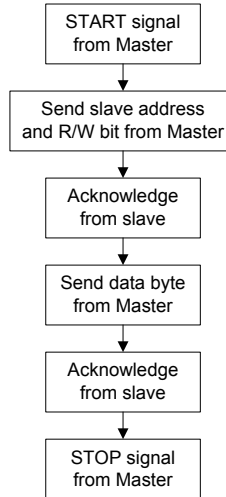
I²C 接口操作

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应加上拉电阻。应注意的是：I²C 总线上的每个设备都没有选择线，但分别与唯一的地址匹配，用于 I²C 通信。

如果有两个设备通过双向的 I²C 总线进行通信，那么就存在一个主机和一个从机。主机和从机都可以用于传输和接收数据，但只有主机才可以控制总线动作。那些处于从机模式的设备，要在 I²C 总线上传输数据只有两种方式，一是从机发送模式，二是从机接收模式。



I²C 方框图



SIMDEB1 和 SIMDEB0 位决定 I²C 接口的去抖时间。这个功能可以使用内部时钟在外部时钟上增加一个去抖间隔，会减小时钟线上毛刺发生的可能性，以避免单片机发生错误动作。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

I²C 寄存器

I²C 总线的两个控制寄存器是 SIMC0 和 SIMC1。一个地址寄存器 SIMA 和一个数据寄存器 SIMD。SIMD 寄存器用于存储发送和接收的数据。这个寄存器在前文 SPI 章节已说明。在单片机尚未将数据写入到 I²C 总线中时，要传输的数据应存在 SIMD 中。I²C 总线接收到数据之后，单片机就可以从 SIMD 数据寄存器中读取。所有通过 I²C 传输或接收的数据都必须通过 SIMD 实现。应注意的是寄存器 SIMA 和 SPI 接口使用的寄存器 SIMC2 是同一个寄存器。SIMC0 寄存器中的 SIMEN 位和 SIM2~SIM0 仅用于 I²C 接口功能。

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	A6	A5	A4	A3	A2	A1	A0	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C 寄存器列表

• 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 TM0 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也可以选择来自 TM0 或 f_{SUB} 。若选择的是作为 SPI 从机, 则其时钟源从外部主机而得。

Bit 4 未定义, 读为“0”

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

- 00: 无去抖时间
- 01: 2 个系统时钟去抖时间
- 1x: 4 个系统时钟去抖时间

Bit 1 **SIMEN: SIM 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**

此位在 I²C 模式中未使用, 当工作在 I²C 模式时, 请忽略该位段。

● 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-bit 数据传输完成
 HCF 是数据传输标志位。数据正在传输时该位为低。当 8-bit 数据传输完成，此位为高并产生一个中断。
- Bit 6 **HAAS**: I²C 总线地址匹配标志位
 0: 地址不匹配
 1: 地址匹配
 HAAS 标志位为地址匹配标志位。此标志位用于决定从机地址是否与主机发送地址相同。若地址匹配此位为高，否则此位为低。
- Bit 5 **HBB**: I²C 总线忙标志位
 0: I²C 总线空闲
 1: I²C 总线忙
 HBB 标志位为 I²C 忙标志位。当检测到 START 信号时 I²C 忙，此位变为高电平。当检测到 STOP 信号时 I²C 总线停止，该位变为低电平。
- Bit 4 **HTX**: I²C 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 总线发送确认标志位
 0: 从机发送确认标志
 1: 从机没有发送确认标志
 TXAK 位为发送确认标志位。从机接收 8-bit 数据之后会将该位在第九个时钟传到总线上。如果从机想要接收更多的数据，则应在接收数据之前将此位设置为“0”。
- Bit 2 **SRW**: I²C 从机读 / 写标志位
 0: 从机应处于接收模式
 1: 从机应处于发送模式
 SRW 位是 I²C 从机读写标志位。决定主机是否希望发送或接收来自 I²C 总线的的数据。当传输地址和从机的地址相匹配时，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该数据。
- Bit 1 **IAMWU**: I²C 地址匹配唤醒控制位
 0: 除能
 1: 使能
 此位应设置为“1”使能 I²C 地址匹配以使系统从休眠或空闲模式中唤醒。若进入休眠或空闲模式前 IAMWU 已经设置以使能 I²C 地址匹配唤醒功能，在系统唤醒后须软件清除此位以确保单片机正确地运行。
- Bit 0 **RXAK**: I²C 总线接收确认标志位
 0: 从机接收到确认标志
 1: 从机没有接收到确认标志
 RXAK 位是接收确认标志位。如果 RXAK 位被重设为“0”即 8-bit 数据传输之后，从机在第九个时钟有接收到一个正确的确认位。如果从机处于发送状态，则从机将会检查 RXAK 位来判断主机接收方是否愿意继续接收下一个字节。因此直到 RXAK 为“1”时，从机传输方停止发送数据。这时，传输方将释放 SDA 线，主机发出停止信号释放 I²C 总线。

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

● SIMA 寄存器

Bit	7	6	5	4	3	2	1	0
Name	A6	A5	A4	A3	A2	A1	A0	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 A6~A0: I²C 从机地址位

A6~A0 是从机地址对应的 bit 6~bit 0。

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-bit 从机地址，寄存器 SIMA 中的第 7~1 位是单片机的从机地址，位 0 未定义。

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

Bit 0 D0: 未定义位

此位可通过软件程序进行读写。

● 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

当超时电路发生超事件，SIMTOF 标志位置高，该位由软件程序清零。

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

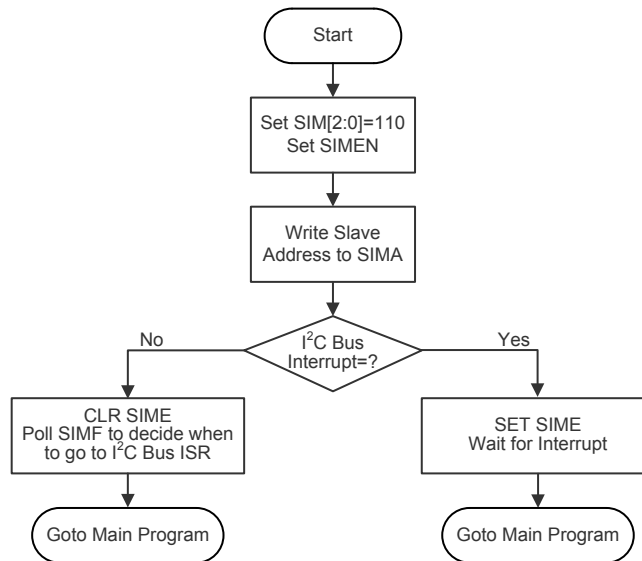
I²C 超时时钟源是 $f_{SUB}/32$ 。

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

I²C 总线通信

I²C 总线上的通信需要四步完成，一个起始信号，一个从机地址发送，一个数据传输，还有一个停止信号。当起始信号被写入 I²C 总线时，总线上的所有从机都会接收到这个起始信号并且被通知总线上会即将有数据到达。数据的前 7 位是从机地址，高位在前，低位在后。如果发出的地址和从机地址匹配，SIMC1 寄存器的 HAAS 位会被置位，同时产生 I²C 中断。进入中断服务程序后，系统要检测 HAAS 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-bit 数据传递完毕，还是来自 I²C 超时发生。在数据传输中，注意的是，在 7-bit 从机地址被发送后，接下来的一位，即第 8 位，是读 / 写控制位，该位的值会反映到 SRW 位中。从机通过检测 SRW 位以确定主控制器是要进入发送模式还是接收模式。在 I²C 总线开始传送数据前，需要先初始化 I²C 总线，初始化 I²C 总线步骤如下：

- 步骤 1
设置 SIMC0 寄存器中 SIM2~SIM0 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 SIME 中断使能位，以使能 SIM 中断和多功能中断。



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

I²C 总线从机地址

I²C 总线上的所有从机都会侦测由主机发出的起始信号。发送起始信号后，紧接着主机会发送从机地址以选择要进行数据传输的从机。所有在 I²C 总线上的从机接收到 7-bit 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，则从机会产生一个 I²C 总线中断信号。地址位接下来的一位为读 / 写状态位 (即第 8 位)，将被保存到 SIMC1 寄存器的 SRW 位，随后发出一个低电平应答信号 (即第 9 位)。当单片机从机的地址匹配时，会将状态标志位 HAAS 置位。

I²C 总线有三个中断源，当程序运行至中断服务子程序时，通过检测 HAAS 和 SIMTOF 位以确定 I²C 总线中断是来自从机地址匹配，还是来自 8-bit 数据传输完毕，还是来自 I²C 超时发生。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 寄存器中虚读数据以释放 SCL 线。

I²C 总线读 / 写信号

SIMC1 寄存器的 SRW 位用来表示主机是要从 I²C 总线上读取数据还是要将数据写到 I²C 总线上。从机则通过检测该位以确定自己是作为发送方还是接收方。当 SRW 置“1”，表示主机要从 I²C 总线上读取数据，从机则作为发送方，将数据写到 I²C 总线；当 SRW 清“0”，表示主机要写数据到 I²C 总线上，从机则做为接收方，从 I²C 总线上读取数据。

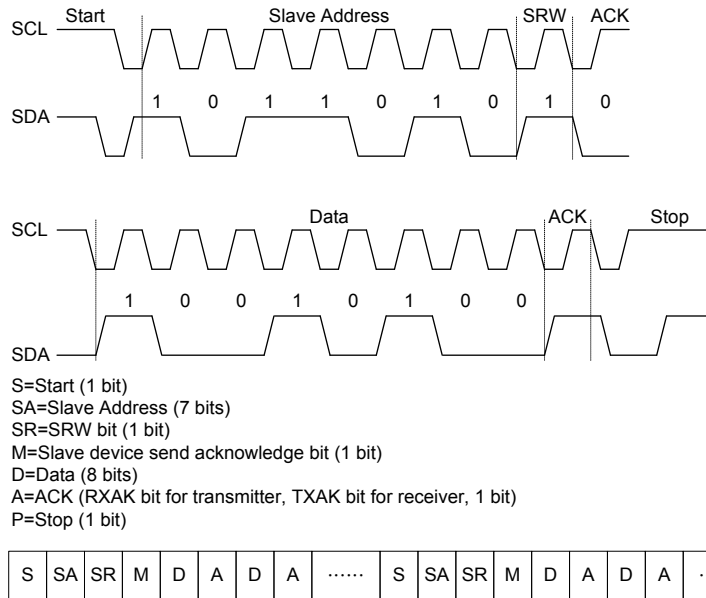
I²C 总线从机地址和确认信号

主机发送呼叫地址后，当 I²C 总线上的任何从机内部地址与其匹配时，会发送一个应答信号。此应答信号会通知主机有从机已经接收到了呼叫地址。如果主机没有收到应答信号，则主机必须发送停止 (STOP) 信号以结束通信。当 HAAS 为高时，表示从机接收到的地址与自己内部地址匹配，则从机需检查 SRW 位，以确定自己是作为发送方还是作为接收方。如果 SRW 位为高，从机须设置成发送方，这样会置位 SIMC1 寄存器的 HTX 位。如果 SRW 位为低，从机须设置成接收方，这样会清零 SIMC1 寄存器的 HTX 位。

I²C 总线数据和确认信号

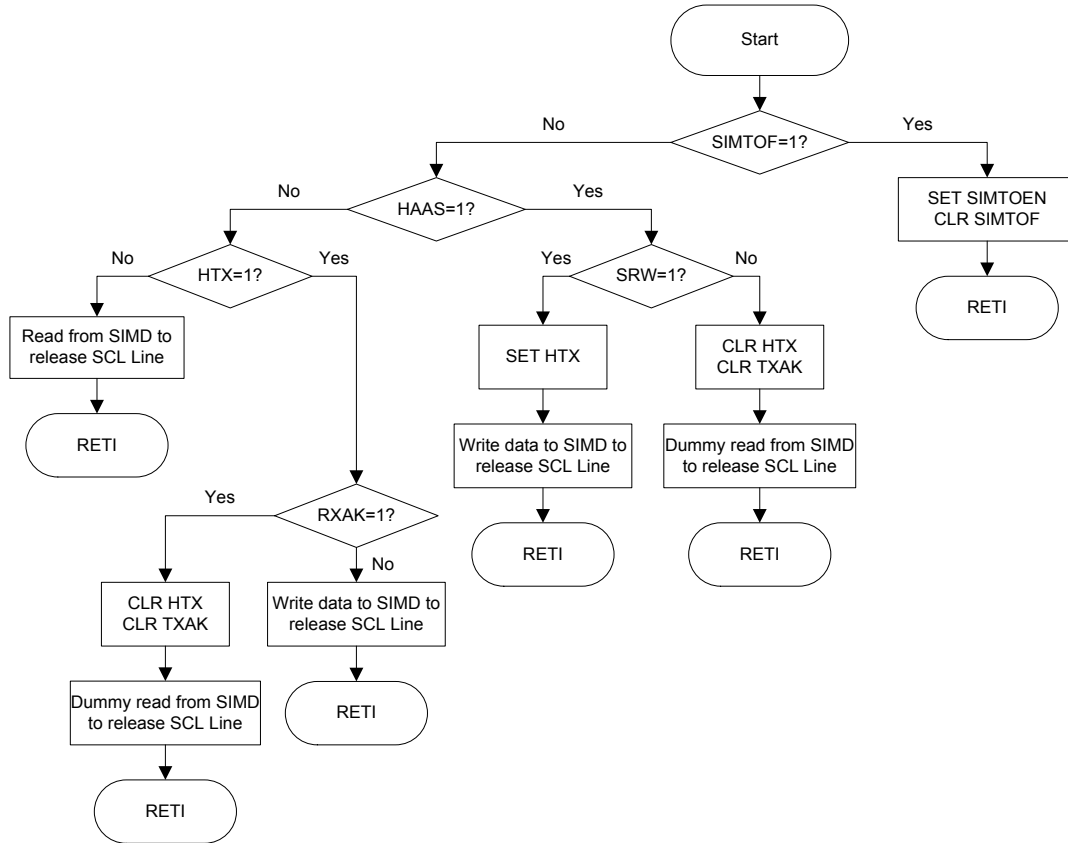
在从机确认接收到从地址后，将进行 8-bit 宽度的数据传输。这个数据传输顺序是的高位在前，低位在后。接收方在接收到 8-bit 数据后必须发出一个应答信号 (“0”) 以继续接收下一个数据。如果从机发送方没接收到主机接收方的应答信号，从机发送方将释放 SDA 线，同时，主机将发出 STOP 信号以释放 I²C 总线。所传送的数据存储在 SIMD 寄存器中。如果设置成发送方，从机必须先将欲传输的数据写到 SIMD 寄存器中；如果设置成接收方，从机必须从 SIMD 寄存器读取数据。

当从机接收方想要继续接收下一个数据时，必须在第 9 个时钟发出应答信号 (TXAK)。被设为发送方的从机将检测寄存器 SIMC1 中的 RXAK 位以判断是否传输下一个字节的数据，如果单片机不传输下一个字节，那么它将释放 SDA 线并等待接收主机的停止信号。



I²C 通信时序图

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



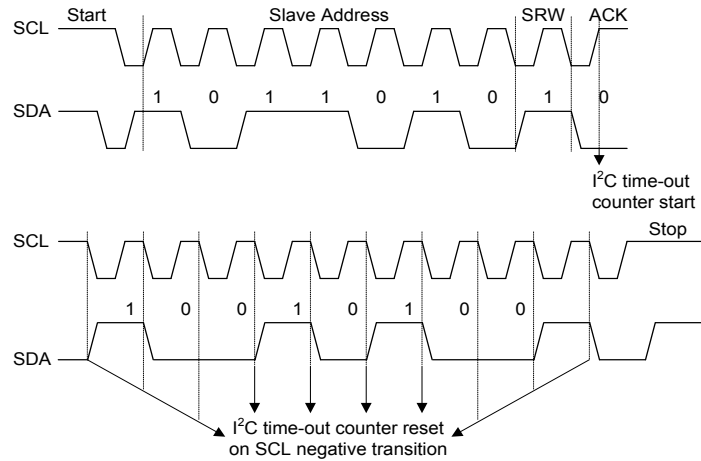
I²C 总线 ISR 流程图

I²C 超时功能

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 SIMC1 寄存器将复位。在一定超时时间后，SIMTOC 寄存器的 SIMTOF 位将被置高。超时功能的使能 / 除能和超时周期由 SIMTOC 寄存器控制。

I²C 超时控制

超时计数器在 I²C 总线“START”和“地址匹配”条件时开始计数，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。共有 64 个超时周期，可通过 SIMTOC 寄存器的 SIMTOS[5:0] 位进行选择。



I²C 超时方框图

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。超时中断使用的也是 I²C 中断向量。当 I²C 超时发生时，I²C 内部电路会被复位，寄存器也将发生如下复位情况。

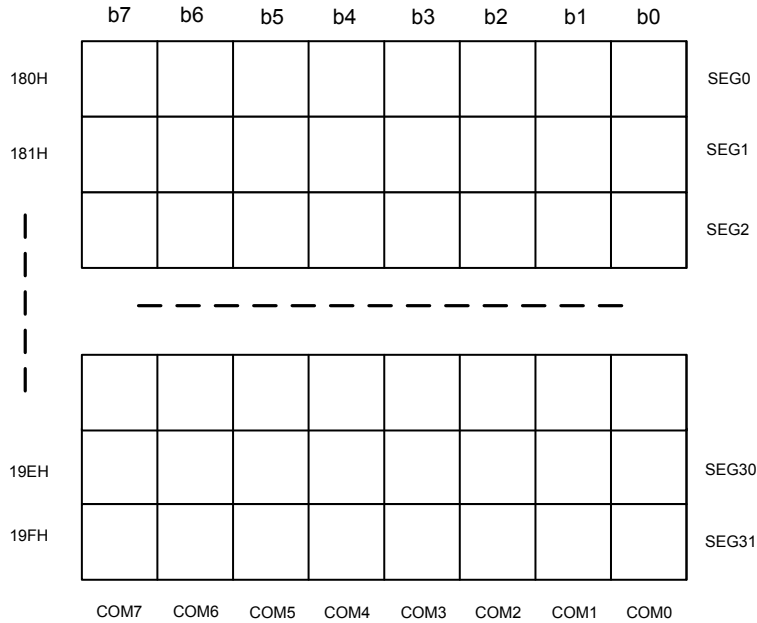
寄存器	I ² C 超时发生后
SIMD, SIMA, SIMC0	无变化
SIMC1	复位到 POR 状态

超时后的 I²C 寄存器状态

LCD 显示存储器

该单片机提供了 LCD 显示数据存储器。该存储区域位于 RAM Sector1 中的 80H~9FH。使用存储器指针 MP1H 可实现在 RAM 和 LCD 显示存储器之间的切换。当 MP1H=01H，数据将写入 LCD 显示数据存储器的 80H~9FH 中。当此位为“01H”外的其它值时，数据将写入通用数据存储器的 80H~9FH 中。

LCD 显示存储器能被读出和写入，但是只能通过间接寻址模式，并使用 MP1L/MP1H 来进行。当数据被写入显示数据区域，这些数据自动地被 LCD 驱动器读取来产生相应的 LCD 驱动信号。把“1”或“0”写入显示存储器的相应位，可以控制显示或不显示。下图为显示存储器和 LCD 显示模块之间的映射关系。



LCD 驱动输出

LCD 驱动器提供 32×4/32×8 或 28×4/28×8 的输出数目。该 LCD 驱动器只有 R 型偏置。LCD 时钟来自内部时钟源 f_{SUB} ， f_{SUB} 的时钟源来自于 LIRC 振荡器或者 LXT 振荡器。

LCD 寄存器

LCDC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	LCDEN	TYPE	DTYC	BIAS	—	RSEL2	RSEL1	RSEL0
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 **LCDCEN**: LCD 使能 / 除能控制

0: 除能

1: 使能

注意当 LCD 输出和 A/D 通道引脚共用时，LCD 驱动器和 A/D 转换器不能同时使能。

- Bit 6 **TYPE:** LCD 波形类型控制
 0: A 型
 1: B 型
- Bit 5 **DTYC:** 定义 LCD 占空比
 0: 1/4 占空比 (LCD COM: COM0~COM3)
 0: 1/8 占空比 (LCD COM: COM0~COM7)
 注: 如果 DTYC = 1, COM4~COM7 引脚配置为 LCD COM 口。如果 DTYC = 0, COM4~COM7 引脚配置为 I/O 口。
- Bit 4 **BIAS:** LCD 偏压选择
 0: 1/3 Bias
 1: 1/4 Bias
- Bit 3 未定义, 读为 “0”
- Bit 2~0 **RSEL2~RSEL0:** 总电阻 R_T 选择位
 000: 1170k Ω
 001: 225 k Ω
 010: 60 k Ω
 011: 快速充电模式, 在 60 k Ω ~1170 k Ω 间转换
 1xx: 快速充电模式, 在 60 k Ω ~ 225 k Ω 间转换
 注: 当 Bias 寄存器选择为 1/3 Bias, 总电阻为 $R_T/3$; 1/4Bias, 总电阻为 $R_T/4$ 。
 该单片机的 LCD 显示器具有低电压快速。在快速充电模式中, 在液晶显示器开始时 (即当前的 LCD COM 口变化), LCD 提供 $R_T = 60$ k Ω 的 LCD 偏置电流。
 在快速充电后, 偏置电阻将变为 225k Ω / 1170k Ω 。

LCDC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	QCT2	QCT1	QCT0	—	VLCD3	VLCD2	VLCD1	VLCD0
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 **QCT2~QCT0:** 快速充电时间选择位
 000: $1 \times t_{SUB}$
 001: $2 \times t_{SUB}$
 010: $3 \times t_{SUB}$
 011: $4 \times t_{SUB}$
 100: $5 \times t_{SUB}$
 101: $6 \times t_{SUB}$
 110: $7 \times t_{SUB}$
 111: $8 \times t_{SUB}$
 $t_{SUB} = 1/f_{SUB}$
- Bit 6~4 未定义, 读为 “0”
- Bit 3~0 **VLCD3~VLCD0:** V_{LCD} 选择位
 0000: $8/16 \times V_{DD}$
 0001: $9/16 \times V_{DD}$
 0010: $10/16 \times V_{DD}$
 0011: $11/16 \times V_{DD}$
 0100: $12/16 \times V_{DD}$
 0101: $13/16 \times V_{DD}$
 0110: $14/16 \times V_{DD}$
 0111: $15/16 \times V_{DD}$
 1000~1111: $16/16 \times V_{DD}$

SEGCR0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SEG7C	SEG6C	SEG5C	SEG4C	SEG3C	SEG2C	SEG1C	SEG0C
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

- Bit 7 **SEG7C**: 选择 SEG7 或 PD7
 0: SEG7
 1: PD7
- Bit 6 **SEG6C**: 选择 SEG6 或 PD6
 0: SEG6
 1: PD6
- Bit 5 **SEG5C**: 选择 SEG5 或 PD5
 0: SEG5
 1: PD5
- Bit 4 **SEG4C**: 选择 SEG4 或 PD4
 0: SEG4
 1: PD4
- Bit 3 **SEG3C**: 选择 SEG3 或 PD3
 0: SEG3
 1: PD3
- Bit 2 **SEG2C**: 选择 SEG2 或 PD2
 0: SEG2
 1: PD2
- Bit 1 **SEG1C**: 选择 SEG1 或 PD1
 0: SEG1
 1: PD1
- Bit 0 **SEG0C**: 选择 SEG0 或 PD0
 0: SEG0
 1: PD0

SEGCR1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SEG15C	SEG14C	SEG13C	SEG12C	SEG11C	SEG10C	SEG9C	SEG8C
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

- Bit 7 **SEG15C**: 选择 SEG15 或 PC7
 0: SEG15
 1: PC7
- Bit 6 **SEG14C**: 选择 SEG14 或 PC6
 0: SEG14
 1: PC6
- Bit 5 **SEG13C**: 选择 SEG13 或 PC5
 0: SEG13
 1: PC5
- Bit 4 **SEG12C**: 选择 SEG12 或 PC4
 0: SEG12
 1: PC4

- Bit 3 **SEG11C**: 选择 SEG11 或 PC3
 0: SEG11
 1: PC3
- Bit 2 **SEG10C**: 选择 SEG10 或 PC2
 0: SEG10
 1: PC2
- Bit 1 **SEG9C**: 选择 SEG9 或 PC1
 0: SEG9
 1: PC1
- Bit 0 **SEG8C**: 选择 SEG8 或 PC0
 0: SEG8
 1: PC0

SEGCR2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SEG23C	SEG22C	SEG21C	SEG20C	SEG19C	SEG18C	SEG17C	SEG16C
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

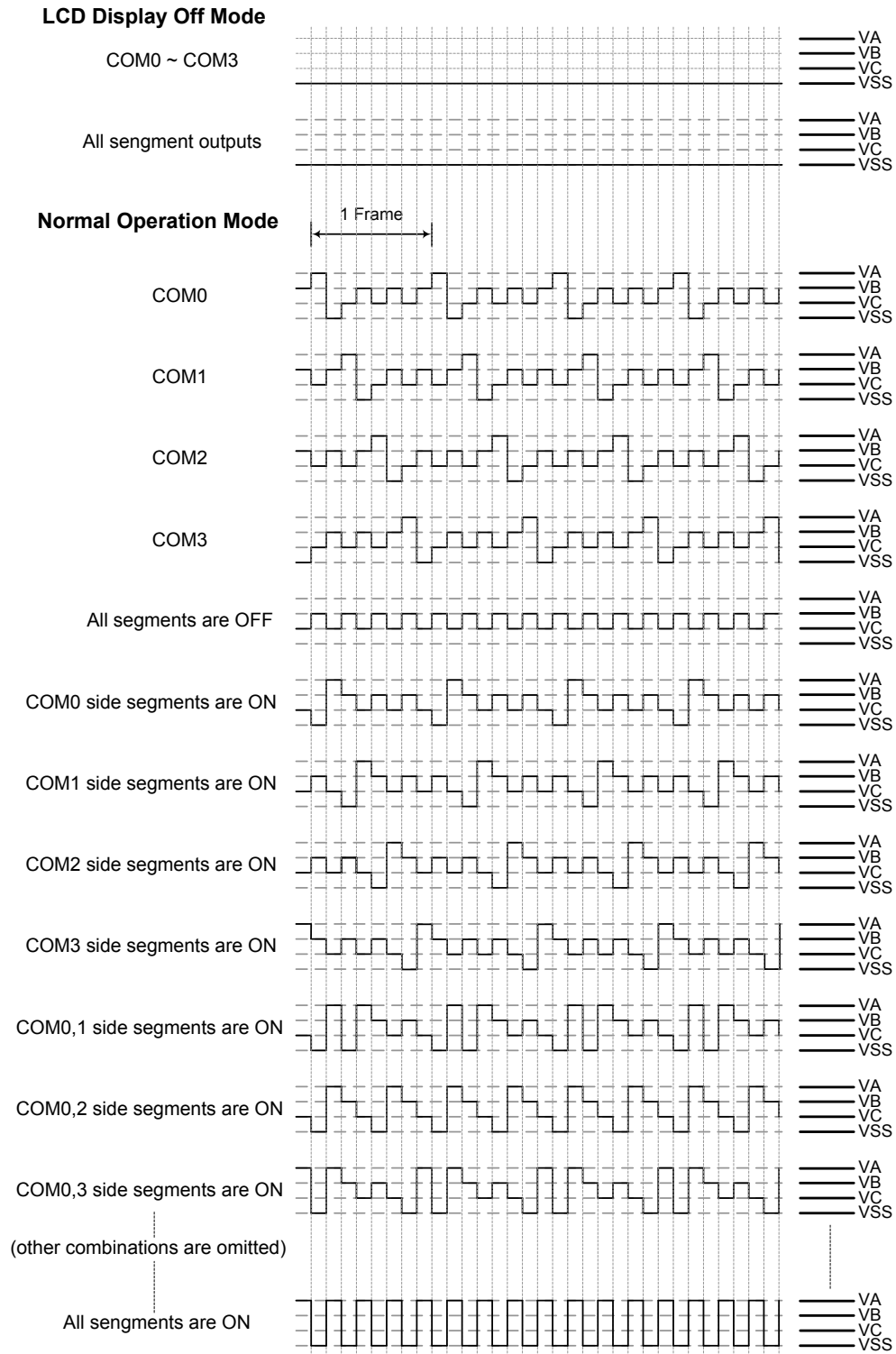
- Bit 7 **SEG23C**: 选择 SEG23 或 PG1
 0: SEG23
 1: PG1
- Bit 6 **SEG22C**: 选择 SEG22 或 PG0
 0: SEG22
 1: PG0
- Bit 5 **SEG21C**: 选择 SEG21 或 PA2
 0: SEG21
 1: PA2
- Bit 4 **SEG20C**: 选择 SEG20 或 PA0
 0: SEG20
 1: PA0
- Bit 3 **SEG19C**: 选择 SEG19 或 PF7
 0: SEG19
 1: PF7
- Bit 2 **SEG18C**: 选择 SEG18 或 PF6
 0: SEG18
 1: PF6
- Bit 1 **SEG17C**: 选择 SEG17 或 PF5
 0: SEG17
 1: PF5
- Bit 0 **SEG16C**: 选择 SEG16 或 PF4
 0: SEG16
 1: PF4

SEGCR3 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SEG31C	SEG30C	SEG29C	SEG28C	SEG27C	SEG26C	SEG25C	SEG24C
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

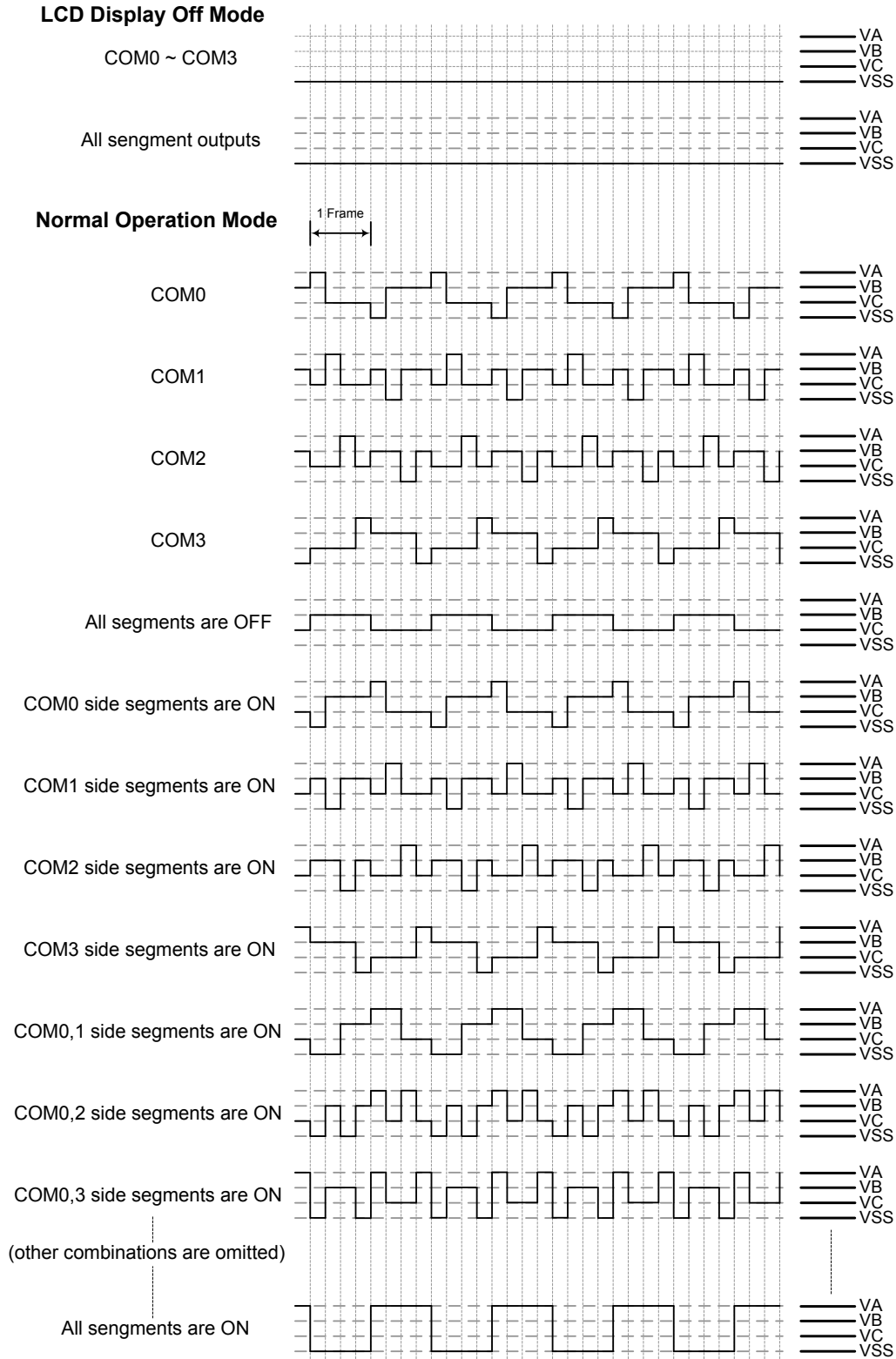
- Bit 7 **SEG31C**: 选择 SEG31 或 PG7
 0: SEG31
 1: PG7
- Bit 6 **SEG30C**: 选择 SEG30 或 PG6
 0: SEG30
 1: PG6
- Bit 5 **SEG29C**: 选择 SEG29 或 PG5
 0: SEG29
 1: PG5
- Bit 4 **SEG28C**: 选择 SEG28 或 PG4
 0: SEG28
 1: PG4
- Bit 3 **SEG27C**: 选择 SEG27 或 PA7
 0: SEG27
 1: PA7
- Bit 2 **SEG26C**: 选择 SEG26 或 PG3
 0: SEG26
 1: PG3
- Bit 1 **SEG25C**: 选择 SEG25 或 PG2
 0: SEG25
 1: PG2
- Bit 0 **SEG24C**: 选择 SEG24 或 PA6
 0: SEG24
 1: PA6

LCD 波形



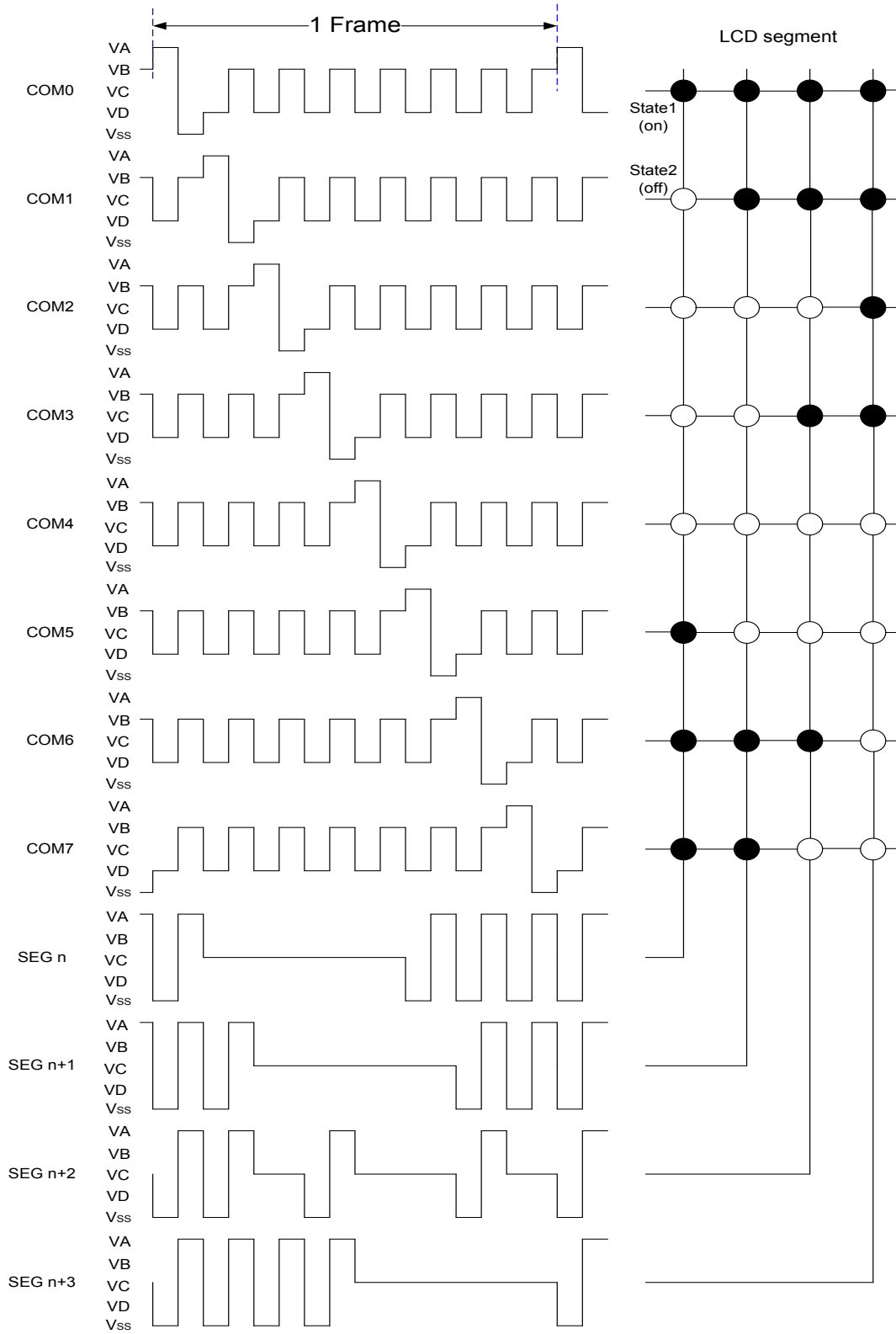
LCD 驱动输出 - A 型, 1/4 Duty, 1/3 Bias

注: $V_A = V_{LCD}$, $V_B = V_{LCD} \times 2/3$, $V_C = V_{LCD} \times 1/3$ 。

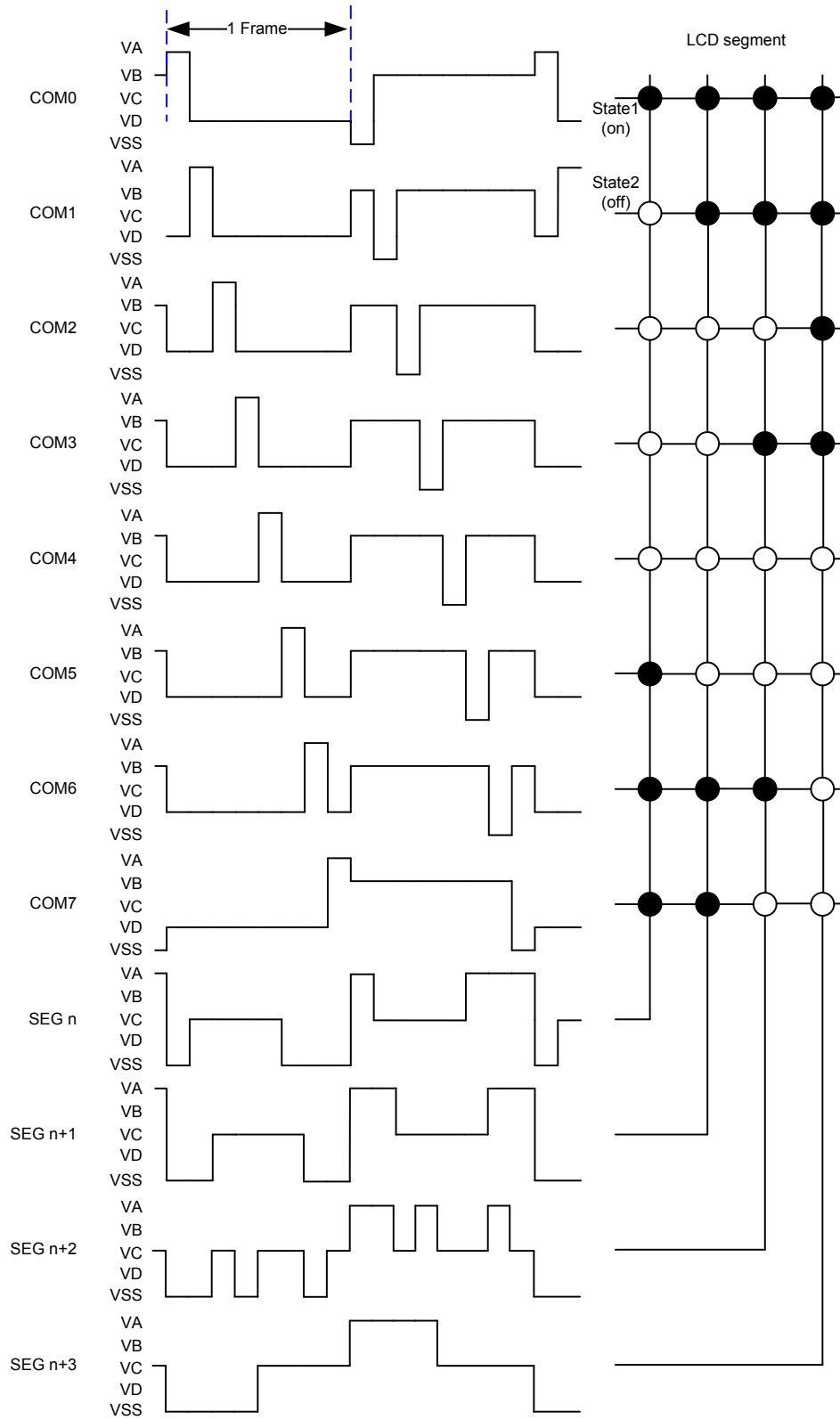


LCD 驱动输出 - B 型, 1/4 Duty, 1/3 Bias

注: $V_A = V_{LCD}$, $V_B = V_{LCD} \times 2/3$, $V_C = V_{LCD} \times 1/3$ 。



LCD 驱动输出 - A 型, 1/8 Duty, 1/4 Bias



LCD 驱动输出 - B 型, 1/8 Duty, 1/4 Bias

LED 驱动器

该单片机包含一个 LED 驱动功能，提供高电流输出驱动能力，可用于驱动外部 LED。

LED 驱动操作

该单片机的多个 I/O 引脚具有提供 LED 高电流驱动的能力，如下表所示：

LED 驱动引脚
PD0~PD7 (高源电流) PE0~PE7 (高灌电流)

LED 驱动寄存器

IOHR0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	IOHS31	IOHS30	IOHS21	IOHS20	IOHS11	IOHS10	IOHS01	IOHS00
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

IOHR1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	IOHS71	IOHS70	IOHS61	IOHS60	IOHS51	IOHS50	IOHS41	IOHS40
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

IOHSn[1:0]: PDn (n = 0~7) 脚 I_{OH} 驱动能力选择位

- 00: GPIO 总源电流驱动能力
- 01: GPIO 1/3 源电流驱动能力
- 10: GPIO 1/4 源电流驱动能力
- 11: GPIO 1/6 源电流驱动能力

UART 模块串行接口

该单片机具有一个全双工异步串行通讯接口—UART，可以很方便地与其它具有串行口的外部设备进行通信。UART 具有许多功能特性，在发送或接收串行数据时，将数据组成一个 8 位或 9 位的数据帧，逐帧进行传输。它还具有检测数据溢出或帧错误等功能。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。

内建的 UART 功能包含以下特性：

- 全双工通用异步收发器
- 8 位或 9 位字符长度
- 奇校验，偶校验或无校验
- 1 位或 2 位停止位
- 8-bit 预分频波特率发生器
- 奇偶校验，帧错误，噪声和溢出错误检测
- 支持地址检测中断 (最后一位 = 1)
- 独立的发送和接收使能
- 2-byte Deep FIFO 接收数据缓存器
- 发送和接收中断触发条件如下：
 - ◆ 发送器为空
 - ◆ 发送器空闲
 - ◆ 接收器已满
 - ◆ 接收器溢出
 - ◆ 地址模式检测
 - ◆ RX 引脚唤醒功能

UART 外部引脚接口

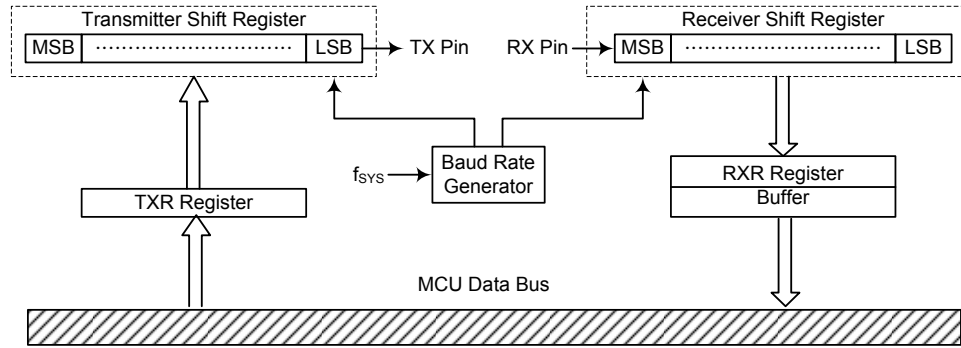
内部 UART 有两个外部引脚 TX 和 RX，可与外部串行接口进行通信。TX 和 RX 分别为 UART 发送脚和接收脚，与 I/O 口或其它功能共用引脚。当 UARTEN、TXEN 和 RXEN 位置高时，将自动设置这些 I/O 脚或其它共用功能脚作为 TX 输出和 RX 输入，并且除能 TX 和 RX 引脚上的上拉电阻功能。当 UARTEN、TXEN 或 RXEN 位清零除能，TX 或 RX 引脚除能，可作为普通 I/O 脚或其它用于功能。

UART 数据传输方案

下图显示了 UART 的整体数据传输结构安排。通过应用程序将需要发送的数据首先写入 TXR 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所控制的速率下将 TSR 寄存器中数据一位位地移到 TX 引脚上，低位在前。只有 TXR 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。

要接收的数据在波特率发生器所控制的速率下，低位在前高位在后，从外部引脚 RX 进入接收移位寄存器 RSR。当移位寄存器接收满，接收的数据从移位寄存器移入可被用户程序操作的 RXR 寄存器中。RXR 寄存器被映射到单片机数据存储器中，而接收器移位寄存器没有实际地址，所以接收器移位寄存器不可直接操作。

需要注意的是，上述发送寄存器 TXR 和接收寄存器 RXR，其实是共享一个地址的数据寄存器 TXR/RXR 寄存器，该寄存器用于数据发送和数据传输。



UART 数据传输方案

UART 状态和控制寄存器

与 UART 功能相关的有五个寄存器 — 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

USR 寄存器为 UART 的状态寄存器，可通过程序读取以确定 UART 当前状态。此寄存器中所有标志位为只读。各个标志位的详细说明如下：

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。此只读标志位 PERR=0，表示奇偶校验正确；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 RXR 寄存器来清除此位。

Bit 6 **NF**: 噪声干扰标志位

- 0: 没有受到噪声干扰
- 1: 受到噪声干扰

NF 是噪声干扰标志位。若此只读标志位 NF=0，表明没有受到噪声干扰；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 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
FREE 是帧错误标志位。若此只读标志 FREE=0, 没有帧错误发生; 若 FREE=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 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。若此只读标志位 OERR=0, 没有溢出错误; 若 OERR=1, 发生了溢出错误, 它将禁止 RXR 寄存器下一组数据的接收。可通过软件清除该标志位, 即先读取 USR 寄存器再读 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。若此只读标志位 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 RXR 状态标志位
0: RXR 寄存器为空
1: RXR 寄存器含有有效数据
RXIF 是接收寄存器状态标志位。当此只读标志位 RXIF=0, RXR 寄存器为空; 当 RXIF=1, RXR 寄存器接收到新数据。当数据从移位寄存器加载到 RXR 寄存器中时, 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。当 RXIF 位为高, 读取 USR 寄存器再读 RXR 寄存器, 如果 RXR 寄存器中没有新的数据, 那么将清除 RXIF 标志。
- Bit 1 **TIDLE:** 发送空闲状态位
0: 正在发送数据
1: 发送器空闲
TIDLE 是数据发送完成标志位。若此只读标志位 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。当 TIDLE 位为高, 读取 USR 寄存器再写 TXR 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 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 数据寄存器为空)
TXIF 是发送数据寄存器为空标志位。若此只读标志位 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 寄存器将清除 TXIF。应注意当 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 引脚处于浮空状态; UARTEN=1, UART 使能, TX 和 RX 将分别由 TXEN 和 RXEN 控制。当 UART 被除能将清空缓冲器, 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

此位用于选择数据长度格式, 可选择长度为 8-bit 或 9-bit。BNO=1, 传输数据为 9-bit; BNO=0, 传输数据为 8-bit。若选择了 9-bit 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。

注: 1. 如果 BNO=1 (9-bit 数据传输), 奇偶功能使能, 数据帧的第九位是奇偶位, 将不会传输到 RX8。

2. 如果 BNO=0 (8-bit 数据传输), 奇偶功能使能, 数据帧的第八位是奇偶位, 将不会传输到 RX7。

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。最高有效位置换成一个奇偶校验位。

Bit 4 PRT: 奇偶校验选择位

0: 偶校验

1: 奇校验

此位为奇偶校验类型选择位。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 8 位 (只读)
 此位只有在传输数据为 9-bit 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-bit 还是 9-bit。
- Bit 0 **TX8**: 发送 9-bit 数据传输格式中的第 8 位 (只写)
 此位只有在传输数据为 9-bit 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-bit 还是 9-bit。

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。另外缓冲器将被复位。此情况下, TX 引脚将设置为浮空状态。
 若 TXEN=1 且 UARTEN=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。若此情况发生, TX 引脚将设置为浮空状态。
- 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。另外缓冲器将被复位。此情况下, RX 引脚将设置为浮空状态。若 RXEN=1 且 UARTEN=1, 则接收将被使能, RX 引脚将由 UART 来控制。在数据接收时清除 RXEN 将中止数据接收且复位接收器。若此情况发生, RX 引脚将设置为浮空状态。
- 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 (BON=0 时对应 RX7) 或第 9 位 (BON=1 时对应 RX8) 为高, 那么接收到的是地址而非数据。若相应的中断使能, 则每次接收到的数据的地址位 (根据 BNO 的值来确定是第 8 或是第 9 位) 置位时都将产生中断请求。若地址检测功能使能, 所接收的地址位为 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_{SYS} 关闭时有效。若 UART 时钟源 f_{SYS} 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_{SYS} 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_{SYS} , 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 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 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

数据寄存器 TXR/RXR 用于存储 TX 引脚的发送数据和 RX 引脚的接收数据

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

波特率发生器

UART 自身具有一个波特率发生器, 通过它可以设定串行数据通信速率。波特率由一个独立的内部 8-bit 计数器所控制, 其周期取决于两个因素。第一个因素为波特率寄存器 BRG 中的值, 第二个因素则是控制寄存器 UCR2 中的 BRGH 位的值。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。下列波特率计算公式中所用 BRG 寄存器的值 N 决定分频因数。注意 BRG 寄存器中的 N 为十进制数, 其范围是 0 到 255。

UCR2 BRGH 位	0	1
波特率 (BR)	$f_{\text{SYS}}/[64(N+1)]$	$f_{\text{SYS}}/[16(N+1)]$

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 N 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。

BRG 寄存器

Bit	7	6	5	4	3	2	1	0
Name	BRG7	BRG 6	BRG 5	BRG 4	BRG 3	BRG 2	BRG 1	BRG 0
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 **BRG 7~BRG 0**: 波特率值
 通过对 UCR2 寄存器的 BRGH 位进行编程选择上述相关规格以及 BRG 寄存器中的所需值可设定所需波特率。
 注: 若 BRGH=0, 波特率= $f_{\text{SYS}}/[64 \times (N+1)]$; 若 BRGH=1, 波特率= $f_{\text{SYS}}/[16 \times (N+1)]$ 。

波特率和误差计算

系统选用 4M 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 $BR = f_{sys}/[64(N+1)]$

转换后的公式 $N = f_{sys}/[(BR \times 64)] - 1$

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 $= (4808 - 4800)/4800 = 0.16\%$

下表给出 BRFH 取不同值时的实际波特率和误差。

波特率 K/BPS	$f_{sys}=8MHz$					
	BRGH=0 时的波特率			BRGH=1 时的波特率		
	BRG	Kbaud	误差 (%)	BRG	Kbaud	误差 (%)
0.3	—	—	—	—	—	—
1.2	103	1.202	0.16	—	—	—
2.4	51	2.404	0.16	207	2.404	0.16
4.8	25	4.808	0.16	103	4.808	0.16
9.6	12	9.615	0.16	51	9.615	0.16
19.2	6	17.8857	-6.99	25	19.231	0.16
38.4	2	41.667	8.51	12	38.462	0.16
57.6	1	62.500	8.51	8	55.556	-3.55
115.2	0	125	8.51	3	125	8.51
250	—	—	—	1	250	0

波特率和误差值

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 2 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，无校验，1 位停止位组成，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 等位设定。用于数据发送和接收的波特率由一个内部的 8-bit 波特率发生器产生，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 接口的使能和除能

UART 是由 UCR1 寄存器的 UARTEN 位来使能和除能的。若 UARTEN、TXEN 和 RXEN 都置位，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，通过配置相关引脚共用控制位使其可作为普通输入 / 输出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 UCR1 寄存器中的 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。

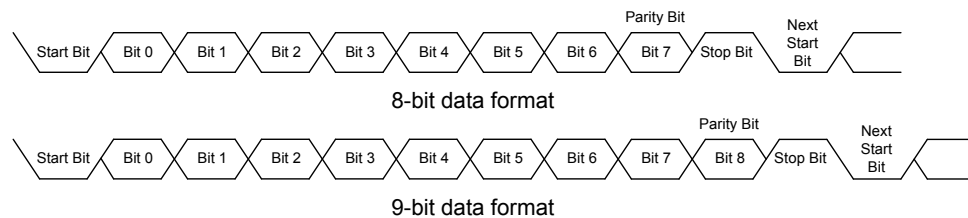
数据位、奇偶校验和停止位选择

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等多种因素组成。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；PRTEN 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-bit 数据位				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
9-bit 数据位				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

发送接收数据格式

下图为 8-bit 和 9-bit 数据格式的发送和接收波形。



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 提供，应用程序只须将发送数据写入 TXR 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 寄存器加载到 TSR 寄存器。应注意的是 TSR 不像其它寄存器一样直接映射到数据存储器，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 寄存器没有加载数据且波特率没有设置一个移位时钟源，发送器将不会工作。先写 TXR 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 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 TX 引脚可通过配置相关引脚共用控制位作为普通输入 / 输出引脚或其它引脚复用功能。

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 寄存器在内部总线和发送移位寄存器间形成一个缓冲。应注意的是如果选择 9-bit 数据传输格式，最高位 MSB 存储在 UCR1 寄存器的 TX8 中。发送器的触发可由如下步骤完成：

- 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，选择所需波特率。
- 置高 TXEN，确保 TX 作为 UART 的发送方引脚。
- 读取 USR 寄存器，然后将待发数据写入 TXR 寄存器。注意，此步骤会清除 TXIF 标志位。

要继续发送数据可重复上述动作。需要注意的是，当 TXIF=0 时，数据将禁止写入 TXR 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 寄存器

只读标志位 TXIF 由 UART 硬件置位。若 TXIF=1，TXR 寄存器为空，其它数据可以写入 TXR 而不会覆盖以前的数据。若 TEIE=1，TXIF 标志位会产生中断。在数据传输时，写 TXR 寄存器指令会将待发数据暂存在 TXR 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 寄存器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。在发送停止位或暂停帧后，表示一帧数据发送完毕，TIDLE 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 寄存器

注意，清除 TXIF 和 TIDLE 软件执行次序相同。

发送暂停

若 TXBRK=1，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 和停止位组成。应用程序置位 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；当应用程序清除了 TXBRK，发送器将传输最后一帧暂停字再加上一位或者两位停止位。暂停字后的高电平保证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 外部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，数据从 RSR 寄存器中加载到为空的接收数据寄存器 RXR 中。RX 引脚上的每一位数据会被采样三次以判断其逻辑状态。应注意 RSR 不像其它寄存器一样映射在数据存储区，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，连续地从外部 RX 引脚进入。在读取模式中，RXR 寄存器在内部总线和接收移位寄存器间形成一个缓冲。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。注意应用程序必须保证在接收完第三帧前读取 RXR 寄存器，否则忽略第三帧数据并且发生溢出错误。接收器的启动可由如下步骤完成：

- 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，选择所需波特率。
- 置高 RXEN，确保 RX 引脚作为 UART 的接收方引脚。

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 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 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 RXR 寄存器

接收暂停

UART 接收任何暂停字都会当作帧错误处理。接收器只根据 BNO 的设置外加一个 STOPS 位确定一帧数据的长度。若暂停字位数远大于 13 位，接收完 BNO 位指定的长度外加一个停止位后认为接收已完毕，RXIF 和 FERR 置位，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。且在下一起始位前必须检测到有效的停止位。接收器不会将线上的暂停状态当成下一个起始位。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。注意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。
- 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，即在起始位和停止位之间，USR 寄存器的接收标志位 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 由接收器的边缘触发置位。若 RIE=1，数据从移位寄存器 RSR 加载到 RXR 寄存器时产生中断，同样地，溢出也会产生中断。

接受错误处理

UART 会产生几种接收错误，下面部分将描述各错误以及 UART 如何处理。

溢出错误 – OERR 标志位

RXR 寄存器是一个两层的 FIFO 缓冲器，它能在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 RXR 寄存器，否则发生溢出错误，通过溢出错误标志位 OERR 显示。

产生溢出错误时将会发生以下事件：

- USR 寄存器中 OERR 被置位。
- 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 RXR 寄存器可将 OERR 清零。

噪声干扰 – NF 标志位

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 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。

应注意，先读取 USR 寄存器再读取 RXR 寄存器可将 NF 清零。

帧错误 – FERR 标志位

如果检测到 0 而不是停止位，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 RXR 寄存器中，此标志位可被任何复位清零。

奇偶校验错误 – PERR 标志位

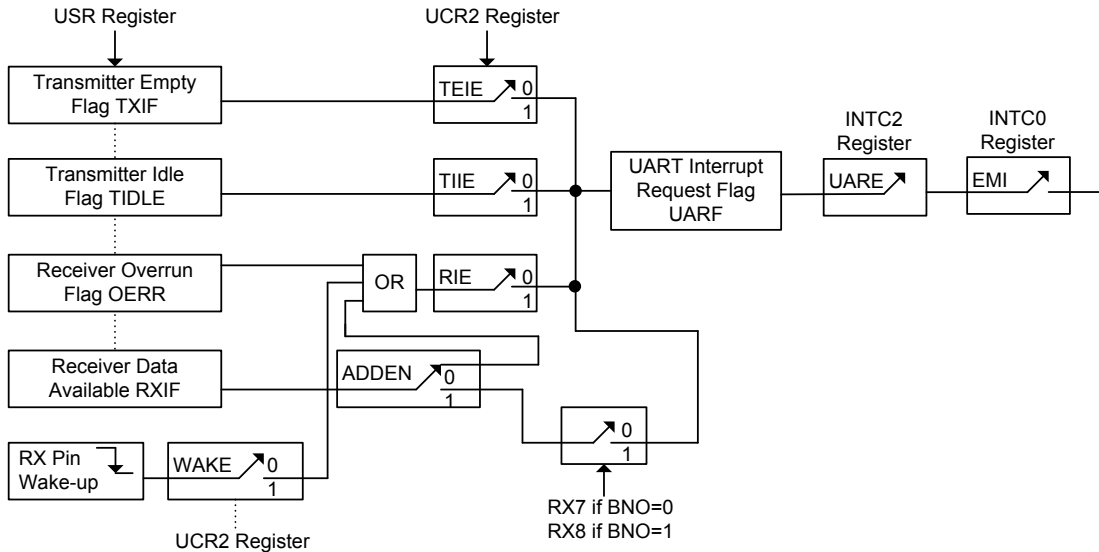
若接收到数据出现奇偶校验错误，USR 寄存器中只读标志 PERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。此标志位同接收的数据分别记录在 USR 寄存器和 RXR 寄存器中，此标志位可被任何复位清零。注意，在读取相应的数据之前必须先访问 USR 寄存器中的 FERR 和 PERR 错误标志位。

UART 模块中断结构

以下几种情况将产生 UART 中断，即发送寄存器为空、发送器空闲、接收器数据有效、接收器溢出、地址检测和 RX 引脚唤醒都会产生中断。在出现上述情况时，将产生一个低脉冲以引起单片机注意。当其中任何一种情况发生时，若其对应的中断控制位使能、整体 UART 中断允许且堆栈未满，程序将会跳转到相应的中断向量执行中断服务程序，而后再返回主程序。其中四种情况，若其 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 使其功能，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_{SYS} 关闭时，UCR2 寄存器中的 WAKE 和 RIE 位被置位，由 RX 引脚上的下降沿唤醒单片机时将会产生 RX 唤醒中断。

注意，USR 寄存器标志位为只读状态，软件不能对其进行设置，在进入相应中断服务程序时也不能清除这些标志位，其它中断亦是如此。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。UART 总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，以确定屏蔽或同意 UART 模块的中断请求。



UART 中断结构

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 使能，则在数据有效时只有在接收到的数据最高位为 1 才会产生中断，注意中断使能位 EMI 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，每次置位 RXIF 都会产生接收有效数据中断，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，为保证操作的正确必须将奇偶检验使能位清零以除能奇偶校验。

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

ADDEN 位功能表

UART 模块暂停和唤醒

当 f_{SYS} 关闭，UART 功能停止。此时该模块的所有时钟源关闭。若 f_{SYS} 在数据传输进行中关闭，发送将停止直到 UART 模块时钟再次使能。同样地，当接收数据时执行 HALT 指令使 UART 进入暂停模式，数据接收也会停止。当 MCU 进入暂停模式，注意 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 MCU 进入暂停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 位控制使能或除能。UART 时钟 f_{SYS} 关闭，若该标志位与 UART 使能位 UARTEN、接收器使能位 RXEN 和接收器中断位 RIE 都被置位，则 RX 引脚的下降沿可触发 RX 引脚唤醒 UART 中断。应注意唤醒后系统需要一定的系统时钟周期才能正常工作，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，全局中断允许位 EMI 和 UART 中断使能控制位 UARE 也必须置位；若这些控制位没有被置位，那么，单片机将可以被唤醒但不会产生中断。注意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。

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，系统会暂停当前的程序而转到执行相对应的中断服务程序。此单片机提供许多外部中断和内部中断功能，外部中断由 INT0~INT3 引脚动作产生，而内部中断由各种内部功能，如 TM、时基、SIM、LVD、EEPROM、UART 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专用数据存储器中的一系列寄存器控制的，如表所示。寄存器总的分为三类。第一类是 INTC0~INTC3 寄存器，用于设置基本的中断；第二类是 MFI0~MFI4 寄存器，用于设置多功能中断；最后一种有 INTEG 寄存器，用于设置外部中断边沿触发类型。

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

功能	使能位	请求标志	注释
总中断	EMI	—	—
INTn 引脚	INTnE	INTnF	n = 0~3
A/D 转换器	ADE	ADF	—
多功能	MFnE	MFnF	n = 0~4
时基	TBnE	TBnF	n = 0~1
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
UART	UARE	UARF	—
SIM	SIME	SIMF	—
TM	TnPE	TnPF	n = 0~3
	nAE	TnAF	

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	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	MF0F	INT1F	INT0F	MF0E	INT1E	INT0E	EMI
INTC1	ADF	MF3F	MF2F	MF1F	ADE	MF3E	MF2E	MF1E
INTC2	MF4F	INT3F	INT2F	UARF	MF4E	INT3E	INT2E	UARE
INTC3	—	—	—	SIMF	—	—	—	SIME
MFI0	—	—	T0AF	T0PF	—	—	T0AE	T0PE
MFI1	—	—	T1AF	T1PF	—	—	T1AE	T1PE
MFI2	—	—	T2AF	T2PF	—	—	T2AE	T2PE
MFI3	—	—	T3AF	T3PF	—	—	T3AE	T3PE
MFI4	TB1F	TB0F	DEF	LVF	TB1E	TB0E	DEE	LVE

中断寄存器列表

INTEG 寄存器

Bit	7	6	5	4	3	2	1	0
Name	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
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 **INT3S1~INT3S0**: INT3 脚中断边沿控制位

- 00: 除能
- 01: 上升沿
- 10: 下降沿
- 11: 双沿

Bit 5~4 **INT2S1~INT2S0**: INT2 脚中断边沿控制位

- 00: 除能
- 01: 上升沿
- 10: 下降沿
- 11: 双沿

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	—	MF0F	INT1F	INT0F	MF0E	INT1E	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 未定义, 读为“0”

Bit 6 **MF0F**: 多功能中断 0 请求标志位

- 0: 无请求
- 1: 中断请求

Bit 5 **INT1F**: 外部中断 1 中断请求标志位

- 0: 无请求
- 1: 中断请求

Bit 4 **INT0F**: 外部中断 0 中断请求标志位

- 0: 无请求
- 1: 中断请求

Bit 3 **MF0E**: 多功能中断 0 控制位

- 0: 除能
- 1: 使能

Bit 2 **INT1E**: 外部中断 1 中断控制位

- 0: 除能
- 1: 使能

- Bit 1 **INT0E**: 外部中断 0 中断控制位
 0: 除能
 1: 使能
- Bit 0 **EMI**: 总中断控制位
 0: 除能
 1: 使能

INTC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ADF	MF3F	MF2F	MF1F	ADE	MF3E	MF2E	MF1E
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 **ADF**: A/D 转换器中断请求标志位
 0: 无请求
 1: 中断请求
- Bit 6 **MF3F**: 多功能中断 3 请求标志位
 0: 无请求
 1: 中断请求
- Bit 5 **MF2F**: 多功能中断 2 请求标志位
 0: 无请求
 1: 中断请求
- Bit 4 **MF1F**: 多功能中断 1 请求标志位
 0: 无请求
 1: 中断请求
- Bit 3 **ADE**: A/D 转换器中断控制位
 0: 除能
 1: 使能
- Bit 2 **MF3E**: 多功能中断 3 控制位
 0: 除能
 1: 使能
- Bit 1 **MF2E**: 多功能中断 2 控制位
 0: 除能
 1: 使能
- Bit 0 **MF1E**: 多功能中断 1 控制位
 0: 除能
 1: 使能

INTC2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	MF4F	INT3F	INT2F	UARF	MF4E	INT3E	INT2E	UARE
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 **MF4F**: 多功能中断 4 请求标志位
 0: 无请求
 1: 中断请求
- Bit 6 **INT3F**: 外部中断 3 中断请求标志位
 0: 无请求
 1: 中断请求
- Bit 5 **INT2F**: 外部中断 2 请求标志位
 0: 无请求
 1: 中断请求

- Bit 4 **UARF**: UART 中断请求标志位
0: 无请求
1: 中断请求
- Bit 3 **MF4E**: 多功能中断 4 控制位
0: 除能
1: 使能
- Bit 2 **INT3E**: 外部中断 3 控制位
0: 除能
1: 使能
- Bit 1 **INT2E**: 外部中断 2 控制位
0: 除能
1: 使能
- Bit 0 **UARE**: UART 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	—	—	—	SIMF	—	—	—	SIME
R/W	—	—	—	R/W	—	—	—	R/W
POR	—	—	—	0	—	—	—	0

- Bit 7~5 未定义，读为“0”
- Bit 4 **SIMF**: SIM 中断请求标志位
0: 无请求
1: 中断请求
- Bit 3~1 未定义，读为“0”
- Bit 0 **SIME**: SIM 中断控制位
0: 除能
1: 使能

MF10 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	T0AF	T0PF	—	—	T0AE	T0PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6 未定义，读为“0”
- Bit 5 **T0AF**: TM0 比较器 A 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **T0PF**: TM0 比较器 P 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 3~2 未定义，读为“0”
- Bit 1 **T0AE**: TM0 比较器 A 匹配中断控制位
0: 除能
1: 使能
- Bit 0 **T0PE**: TM0 比较器 P 匹配中断控制位
0: 除能
1: 使能

MF11 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	T1AF	T1PF	—	—	T1AE	T1PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6 未定义，读为“0”
- Bit 5 **T1AF**: TM1 比较器 A 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **T1PF**: TM1 比较器 P 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 3~2 未定义，读为“0”
- Bit 1 **T1AE**: TM1 比较器 A 匹配中断控制位
0: 除能
1: 使能
- Bit 0 **T1PE**: TM1 比较器 P 匹配中断控制位
0: 除能
1: 使能

MF12 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	T2AF	T2PF	—	—	T2AE	T2PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6 未定义，读为“0”
- Bit 5 **T2AF**: TM2 比较器 A 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **T2PF**: TM2 比较器 P 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 3~2 未定义，读为“0”
- Bit 1 **T2AE**: TM2 比较器 A 匹配中断控制位
0: 除能
1: 使能
- Bit 0 **T2PE**: TM2 比较器 P 匹配中断控制位
0: 除能
1: 使能

MF13 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	T3AF	T3PF	—	—	T3AE	T3PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6 未定义，读为“0”
- Bit 5 **T3AF**: TM3 比较器 A 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **T3PF**: TM3 比较器 P 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 3~2 未定义，读为“0”
- Bit 1 **T3AE**: TM3 比较器 A 匹配中断控制位
0: 除能
1: 使能
- Bit 0 **T3PE**: TM3 比较器 P 匹配中断控制位
0: 除能
1: 使能

MF14 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TB1F	TB0F	DEF	LVF	TB1E	TB0E	DEE	LVE
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

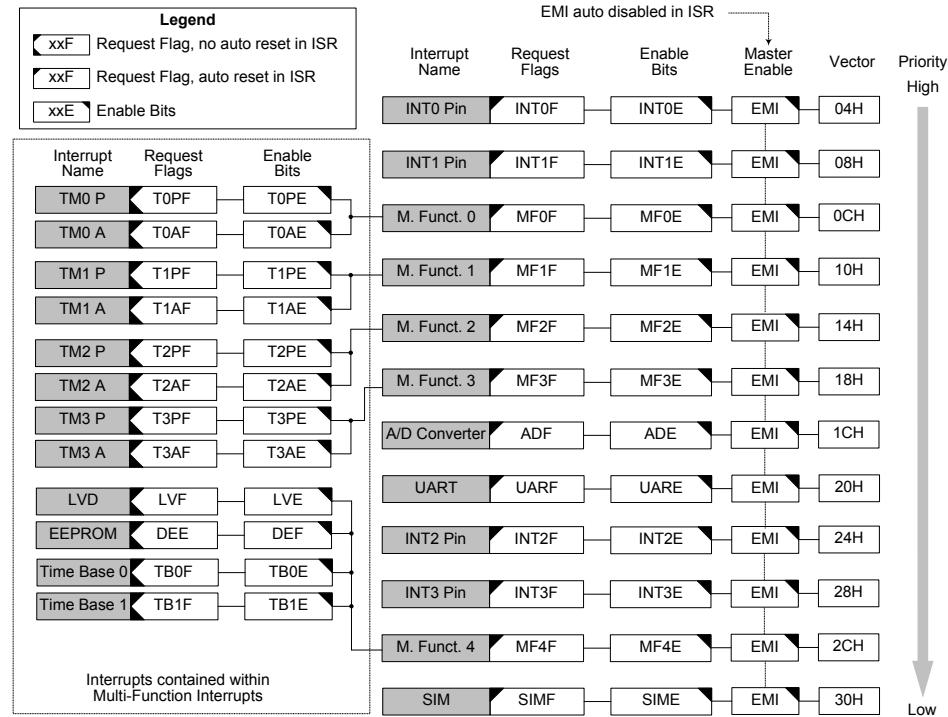
- Bit 7 **TB1F**: 时基中断 1 请求标志位
0: 无请求
1: 中断请求
- Bit 6 **TB0F**: 时基中断 0 请求标志位
0: 无请求
1: 中断请求
- Bit 5 **DEF**: EEPROM 中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **LVF**: LVD 中断请求标志位
0: 无请求
1: 中断请求
- Bit 3 **TB1E**: 时基中断 1 控制位
0: 除能
1: 使能
- Bit 2 **TB0E**: 时基中断 0 控制位
0: 除能
1: 使能
- Bit 1 **DEE**: EEPROM 中断控制位
0: 除能
1: 使能
- Bit 0 **LVE**: LVD 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，下一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。系统将从此中断向量取下条指令。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”指令返回至主程序断点处，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~INT3 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~INT3 引脚的状态发生变化，外部中断请求标志 INT0F~INT3F 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~INT3E 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和通过相关引脚共用选择位选择此引脚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~INT3F 会自动复位且 EMI 位会被清零以除能其它中断。注意，即使此引脚被用作外部中断输入，其配置选项中的上拉电阻仍保持有效。寄存器 INTEG 被用来选择有效的边沿类型，来触发外部中断。可以选择上升沿还是下降沿或双沿触发都产生外部中断。注意 INTEG 也可以用来除能外部中断功能。

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。若要跳转到相应中断向量地址，总中断控制位 EMI 和 A/D 转换器中断使能位 ADE 需先被置位。当中断使能，堆栈未满且 A/D 转换动作结束时，可跳转至 A/D 转换器中断向量子程序中执行。当 A/D 转换器中断响应时，相应的中断请求标志位 ADF 会自动清零且 EMI 位会被清零以除能其它中断。

多功能中断

此单片机中有五种多功能中断，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 TM 中断、EEPROM 中断、LVD 中断和时基中断。

当多功能中断中任何一种中断请求标志 MF_nF 被置位，多功能中断请求产生。当中断使能，堆栈未满，包括在多功能中断中的任意一个中断发生时，将调用多功能中断向量中的一个子程序。当响应中断服务子程序时，相关的多功能请求标志位会自动复位且 EMI 位会自动清零以除能其它中断。

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

SIM 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，堆栈未满且上述任一情况发生时，可跳转至 SIM 中断向量子程序中执行。当串行接口中断响应，SIMF 请求标志可自动复位，EMI 将被自动清零以除能其它中断。

UART 中断

UART 传输中断由几种 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 UARF 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 和 UART 中断使能位 UARE 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，相应的中断请求标志位 UARF 会自动复位且 EMI 位会被清零以除能其它中断。而 UART 模块 USR 寄存器中的标志位仅在 UART 特定动作发生时才会自动被清除。更多关于 UART 中断的细节请参考 UART 章节。

时基中断

时基中断也属于多功能中断。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 或 TB1F 被置位时，中断请求发生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和时基使能位 TBOE 或 TB1E 和相关的多功能中断使能位需先被置位。当中断使能，堆栈未满且时基溢出时，将调用它们各自的中断向量子程序。当响应中断服务子程序时，只有相应的多功能中断请求标志位会自动清零且 EMI 位会被清零以除能其它中断。由于 TBnF 中断请求标志位不会自动清零，需要通过应用程序清零。

时基中断的目的是提供一个固定周期的中断信号。其时钟源来自内部时钟源 f_{TB} 。 f_{TB} 输入时钟首先经过分频器，分频率由程序设置 TBC 寄存器相关位获取合适的分频值以提供更长的时基中断周期。控制时基中断频率 f_{TB} 的时钟源有几种，如在系统工作模式章节所示。

TBC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TBON	TBCK	TB11	TB10	—	TB02	TB01	TB00
R/W	R/W	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	0	0	1	1	—	1	1	1

Bit 7 **TB0ON**: 时基 0 和时基 1 控制位

0: 除能
1: 使能

Bit 6 **TBCK**: f_{TB} 时钟源选择位

0: f_{TBC}
1: $f_{SYS}/4$

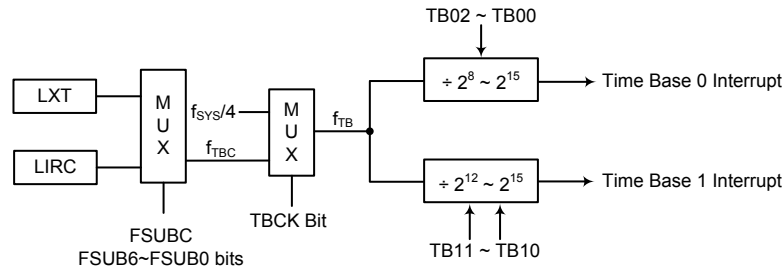
Bit 5~4 **TB11~TB10**: 选择时基 1 溢出周期

00: $2^{12}/f_{TB}$
01: $2^{13}/f_{TB}$
10: $2^{14}/f_{TB}$
11: $2^{15}/f_{TB}$

Bit 3 未定义，读为“0”

Bit 2~0 **TB02~TB00**: 选择时基 0 溢出周期

000: $2^8/f_{TB}$
001: $2^9/f_{TB}$
010: $2^{10}/f_{TB}$
011: $2^{11}/f_{TB}$
100: $2^{12}/f_{TB}$
101: $2^{13}/f_{TB}$
110: $2^{14}/f_{TB}$
111: $2^{15}/f_{TB}$



时基中断

EEPROM 中断

EEPROM 中断也属于多功能中断。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、EEPROM 中断使能位 DEE 和相应多功能中断使能位需先被置位。当中断使能，堆栈未满且 EEPROM 写周期结束时，可跳转至相关多功能中断向量程序执行。当 EEPROM 中断响应，EMI 将被自动清零以除能其它中断，多功能中断请求标志也可自动清除，但 DEF 标志需在应用程序中手动清除。

LVD 中断

LVD 中断也属于多功能中断。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、低电压中断使能位 LVE 和相应多功能中断使能位需先被置位。当中断使能，堆栈未满且低电压条件发生时，可跳转至相关多功能中断向量程序执行。当低电压中断响应，EMI 将被自动清零以除能其它中断，多功能中断请求标志也可自动清除，但 LVF 标志需在应用程序中手动清除。

TM 中断

简易型和周期型 TM 各有两个中断，来自比较器 P 匹配和比较器 A 匹配，都 TM 中断都属于多功能中断。所有类型的 TM 都有两个中断请求标志位 TnPF 和 TnAF 及两个使能位 TnPE 和 TnAE。当 TM 比较器 P 和比较器 A 匹配情况发生时，相应 TM 中断请求标志被置位，TM 中断请求产生。

若要程序跳转到相应中断向量地址，总中断控制位 EMI 和相应 TM 中断使能位和相关多功能中断使能位 MFnE 需先被置位。当中断使能，堆栈未满且 TM 比较器匹配情况发生时，可跳转至相关多功能中断向量程序执行。当 TM 中断响应，EMI 将被自动清零以除能其它中断，相关 MFnE 标志也可自动清除，但 TM 中断请求标志需通过应用程序清零。

中断唤醒功能

每个中断都具有将处于休眠或空闲模式的单片机唤醒的能力。当中断请求标志由低到高转换时唤醒动作产生，其与中断是否使能无关。因此，尽管单片机处于休眠或空闲模式且系统振荡器停止工作，如有外部中断脚上产生外部边沿跳变，低电压或比较器输出位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应用程序清除。

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 MFnF 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。

低电压检测 – LVD

该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。低电压检测会有一些的功耗，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	—	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7~6 未定义，读为“0”

Bit 5 **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 未定义，读为“0”

Bit 2~0 **VLVD2~VLVD0**: LVD 电压选择位

000: 2.0V

001: 2.2V

010: 2.4V

011: 2.7V

100: 3.0V

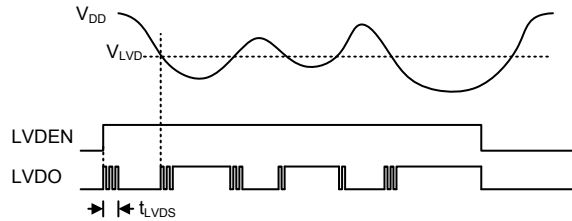
101: 3.3V

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。当单片机进入休眠模式时，LV DEN 位为高，低电压检测器有效。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。



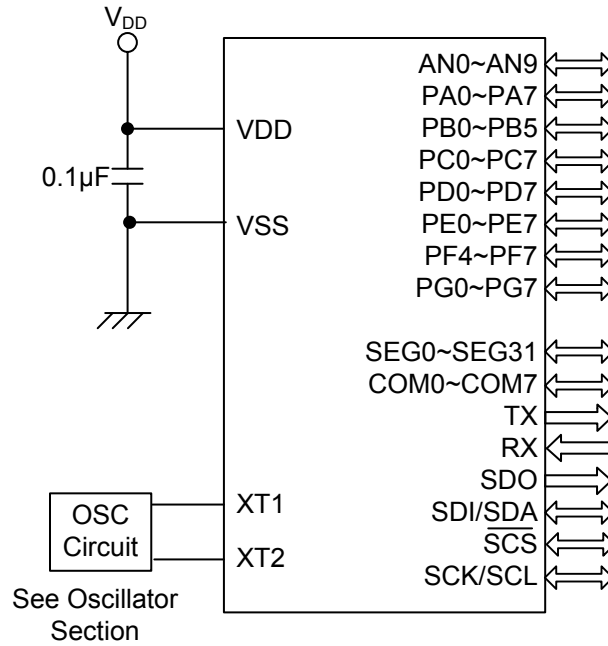
LVD 操作低电压检测器也有自己的中断功能，也是属于多功能中断的一种，它是除了轮询 LVDO 位之外的另一种检测低电压的方法。中断条件产生置位 LVDO 并延时 t_{LVD} 后，中断产生。此种情况下，若 V_{DD} 降至小于 LVD 预置电压值时，中断请求标志位 LVF 将被置位，中断产生，单片机将从休眠或空闲模式中被唤醒。若不要求低电压检测的唤醒功能使能，在单片机进入休眠或空闲模式前应将 LVF 标志置为高。当 LVD 功能使能后，建议在使能相关中断前先将 LVD 中断标志位清零以避免错误操作。

配置选项

配置选项在烧写程序时写入芯片。通过 HT-IDE 的软件开发环境，使用者在开发过程中可以选择配置选项。当配置选项烧入单片机后，无法再通过应用程序修改。所有位必须按系统的需要定义，具体内容可参考下表：

序号	选项
1	高速振荡器类型选择 f _H – HXT 或 HIRC

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他们的应用。

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，使用几种 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，在盛群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 全都包含在盛群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盛群单片机的特性之一。这特性对于输出端口位的设置尤其有用，其中个别的位或端口的引脚可以使用“SET [m].i”或“CLR [m].i”指令来设定其为高位或低位。如果没有这特性，程序设计师必须先读入输出口的 8 位数据，处理这些数据，然后再输出正确的新数据。这种读入 - 修改 - 写出的过程现在则被位运算指令所取代。

查表运算

数据的储存通常由寄存器完成，然而当处理大量固定的数据时，它的存储量常常造成对个别存储器的不便。为了改善此问题，盛群单片机允许在程序存储器中建立一个表格作为数据可直接存储的区域，只需要一组简易的指令即可对数据进行查表。

其它运算

除了上述功能指令外，其它指令还包括用于省电的“HALT”指令和使程序在极端电压或电磁环境下仍能正常工作的看门狗定时器控制指令。这些指令的使用则请查阅相关的章节。

指令集概要

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

惯例

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

助记符	说明	指令周期	影响标志位
算术运算			
ADD A,[m]	ACC 与数据存储器相加，结果放入 ACC	1	Z, C, AC, OV, SC
ADDM A,[m]	ACC 与数据存储器相加，结果放入数据存储器	1 ^注	Z, C, AC, OV, SC
ADD A, x	ACC 与立即数相加，结果放入 ACC	1	Z, C, AC, OV, SC
ADC A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	1	Z, C, AC, OV, SC
ADCM A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	1 ^注	Z, C, AC, OV, SC
SUB A, x	ACC 与立即数相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SUB A,[m]	ACC 与数据存储器相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SUBM A,[m]	ACC 与数据存储器相减，结果放入数据存储器	1 ^注	Z, C, AC, OV, SC, CZ
SBC A,[m]	ACC 与数据存储器、进位标志的反相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	1 ^注	Z, C, AC, OV, SC, CZ
SBC A, x	ACC 与立即数、进位标志相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
DAA [m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	1 ^注	C
逻辑运算			
AND A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	1	Z
OR A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	1	Z
XOR A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	1	Z
ANDM A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	1 ^注	Z
ORM A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	1 ^注	Z
XORM A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	1 ^注	Z
AND A, x	ACC 与立即数做“与”运算，结果放入 ACC	1	Z
OR A, x	ACC 与立即数做“或”运算，结果放入 ACC	1	Z
XOR A, x	ACC 与立即数做“异或”运算，结果放入 ACC	1	Z
CPL [m]	对数据存储器取反，结果放入数据存储器	1 ^注	Z

助记符	说明	指令周期	影响标志位
CPLA [m]	对数据存储器取反, 结果放入 ACC	1	Z
递增和递减			
INCA [m]	递增数据存储器, 结果放入 ACC	1	Z
INC [m]	递增数据存储器, 结果放入数据存储器	1 ^注	Z
DECA [m]	递减数据存储器, 结果放入 ACC	1	Z
DEC [m]	递减数据存储器, 结果放入数据存储器	1 ^注	Z
移位			
RRA [m]	数据存储器右移一位, 结果放入 ACC	1	无
RR [m]	数据存储器右移一位, 结果放入数据存储器	1 ^注	无
RRCA [m]	带进位将数据存储器右移一位, 结果放入 ACC	1	C
RRC [m]	带进位将数据存储器右移一位, 结果放入数据存储器	1 ^注	C
RLA [m]	数据存储器左移一位, 结果放入 ACC	1	无
RL [m]	数据存储器左移一位, 结果放入数据存储器	1 ^注	无
RLCA [m]	带进位将数据存储器左移一位, 结果放入 ACC	1	C
RLC [m]	带进位将数据存储器左移一位, 结果放入数据存储器	1 ^注	C
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 and TBLH	2 ^注	无
TABRDL [m]	读取最后页的 ROM 内容, 并送至数据存储器 and TBLH	2 ^注	无
ITABRD [m]	读表指针 TBLP 自加, 读取当前页的 ROM 内容, 并送至数据存储器 and TBLH	2 ^注	无

助记符	说明	指令周期	影响标志位
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 and TBLH	2 ^注	无
其它指令			
NOP	空指令	1	无
CLR [m]	清除数据存储器	1 ^注	无
SET [m]	置位数据存储器	1 ^注	无
CLR WDT	清除看门狗定时器	1	TO, PDF
SWAP [m]	交换数据存储器的高低字节，结果放入数据存储器	1 ^注	无
SWAPA [m]	交换数据存储器的高低字节，结果放入 ACC	1	无
HALT	进入暂停模式	1	TO, PDF

- 注: 1. 对跳转指令而言，如果比较的结果牵涉到跳转即需 3 个周期，如果没有发生跳转，则只需一个周期。
 2. 任何指令若要改变 PCL 的内容将需要 2 个周期来执行。
 3. 对于“CLR WDT”指令而言，TO 和 PDF 标志位也许会受执行结果影响，“CLR WDT”被执行后，TO 和 PDF 标志位会被清除，否则 TO 和 PDF 标志位保持不变。

扩展指令集

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

助记符	说明	指令周期	影响标志位
算术运算			
LADD A,[m]	ACC 与数据存储器相加，结果放入 ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	ACC 与数据存储器相加，结果放入数据存储器	2 ^注	Z, C, AC, OV, SC
LADC A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	2	Z, C, AC, OV, SC
LADCM A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	2 ^注	Z, C, AC, OV, SC
LSUB A,[m]	ACC 与数据存储器相减，结果放入 ACC	2	Z, C, AC, OV, SC, CZ
LSUBM A,[m]	ACC 与数据存储器相减，结果放入数据存储器	2 ^注	Z, C, AC, OV, SC, CZ
LSBC A,[m]	ACC 与数据存储器、进位标志的反相减，结果放入 ACC	2	Z, C, AC, OV, SC, CZ
LSBCM A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	2 ^注	Z, C, AC, OV, SC, CZ
LDA A [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

助记符	说明	指令周期	影响标志位
LRLC [m]	带进位将数据存储器左移一位，结果放入数据存储器	2 ^注	C
数据传送			
LMOV A,[m]	将数据存储器送至 ACC	2	无
LMOV [m],A	将 ACC 送至数据存储器	2 ^注	无
位运算			
LCLR [m].i	清除数据存储器的位	2 ^注	无
LSET [m].i	置位数据存储器的位	2 ^注	无
转移			
LSZ [m]	如果数据存储器为零，则跳过下一条指令	2 ^注	无
LSZA [m]	数据存储器送至 ACC，如果内容为零，则跳过下一条指令	1 ^注	无
LSNZ [m]	如果数据存储器不为零，则跳过下一条指令	2 ^注	无
LSZ [m].i	如果数据存储器的第 i 位为零，则跳过下一条指令	2 ^注	无
LSNZ [m].i	如果数据存储器的第 i 位不为零，则跳过下一条指令	2 ^注	无
LSIZ [m]	递增数据存储器，如果结果为零，则跳过下一条指令	2 ^注	无
LSDZ [m]	递减数据存储器，如果结果为零，则跳过下一条指令	2 ^注	无
LSIZA [m]	递增数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	2 ^注	无
LSDZA [m]	递减数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	2 ^注	无
查表			
LTABRD [m]	读取当前页的 ROM 内容，并送至数据存储器 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，读取当前页的 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 4 个周期，如果没有发生跳转，则只需两个周期。

2. 任何扩展指令若要改变 PCL 的内容将需要 3 个周期来执行。

指令定义

ADC A, [m]	Add Data Memory to ACC with Carry
指令说明	将指定的数据存储器、累加器内容以及进位标志相加，结果存放到累加器。
功能表示	$ACC \leftarrow ACC + [m] + C$
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
ADDMA, [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
ANDA, [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
ANDMA, [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 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，结果存放在数据存储器。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 ← 0 PDF ← 1
影响标志位	TO、PDF
INC [m]	Increment Data Memory
指令说明	将指定数据存储器的内容加 1。
功能表示	[m] ← [m] + 1
影响标志位	Z
INCA [m]	Increment Data Memory with result in ACC
指令说明	将指定数据存储器的内容加 1，结果存放回累加器并保持指定的数据存储器内容不变。
功能表示	ACC ← [m] + 1
影响标志位	Z
JMP addr	Jump unconditionally
指令说明	程序计数器的内容无条件地由被指定的地址取代，程序由新的地址继续执行。当新的地址被加载时，必须插入一个空指令周期，所以此指令为 2 个周期的指令。
功能表示	Program Counter ← addr
影响标志位	无
MOV A, [m]	Move Data Memory to ACC
指令说明	将指定数据存储器的内容复制到累加器。
功能表示	ACC ← [m]
影响标志位	无
MOV A, x	Move immediate data to ACC
指令说明	将 8 位立即数载入累加器。
功能表示	ACC ← x
影响标志位	无

MOV [m], A	Move ACC to Data Memory
指令说明	将累加器的内容复制到指定的数据存储器。
功能表示	$[m] \leftarrow \text{ACC}$
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	$\text{PC} \leftarrow \text{PC}+1$
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	$\text{ACC} \leftarrow \text{ACC} \text{ "OR" } [m]$
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	$\text{ACC} \leftarrow \text{ACC} \text{ "OR" } x$
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	$[m] \leftarrow \text{ACC} \text{ "OR" } [m]$
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	$\text{Program Counter} \leftarrow \text{Stack}$
影响标志位	无
RETA, 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	$\text{Program Counter} \leftarrow \text{Stack}$ $\text{ACC} \leftarrow x$
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相应，则这个中断将在返回主程序之前被相应。
功能表示	Program Counter ← Stack EMI ← 1
影响标志位	无
RL [m]	Rotate Data Memory left
指令说明	将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。
功能表示	$[m].(i+1) \leftarrow [m].i \ (i=0\sim6)$ $[m].0 \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i \ (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
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	$[m].(i+1) \leftarrow [m].i \ (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i \ (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
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=0~6) $[m].7 \leftarrow [m].0$
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=0~6) $ACC.7 \leftarrow [m].0$
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=0~6) $[m].7 \leftarrow C$ $C \leftarrow [m].0$
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=0~6) $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
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

<p>SBC A, x 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Subtract immediate data from ACC with Carry 将累加器减去立即数以及进位标志，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。</p> <p>$ACC \leftarrow ACC - [m] - \bar{C}$</p> <p>OV、Z、AC、C、SC、CZ</p>
<p>SBCM A, [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Subtract Data Memory from ACC with Carry and result in Data Memory 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。</p> <p>$[m] \leftarrow ACC - [m] - \bar{C}$</p> <p>OV、Z、AC、C、SC、CZ</p>
<p>SDZ [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Skip if Decrement Data Memory is 0 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。</p> <p>$[m] \leftarrow [m] - 1$，如果 $[m]=0$ 跳过下一条指令执行</p> <p>无</p>
<p>SDZA [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Decrement data memory and place result in ACC, skip if 0 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。</p> <p>$ACC \leftarrow [m] - 1$，如果 $ACC=0$ 跳过下一条指令执行</p> <p>无</p>
<p>SET [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Set Data Memory 将指定数据存储器的每一位设置为 1。</p> <p>$[m] \leftarrow FFH$</p> <p>无</p>

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	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

<p>SUB A, [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>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。</p> <p>$ACC \leftarrow ACC - [m]$</p> <p>OV、Z、AC、C、SC、CZ</p>
<p>SUBM A, [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Subtract Data Memory from ACC with result in Data Memory 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。</p> <p>$[m] \leftarrow ACC - [m]$</p> <p>OV、Z、AC、C、SC、CZ</p>
<p>SUB A, x 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Subtract immediate Data from ACC 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。</p> <p>$ACC \leftarrow ACC - x$</p> <p>OV、Z、AC、C、SC、CZ</p>
<p>SWAP [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Swap nibbles of Data Memory 将指定数据存储器的低 4 位和高 4 位互相交换。</p> <p>$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$</p> <p>无</p>
<p>SWAPA [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Swap nibbles of Data Memory with result in ACC 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。</p> <p>$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$</p> <p>$ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$</p> <p>无</p>
<p>SZ [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Skip if Data Memory is 0 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。</p> <p>如果 $[m]=0$，跳过下一条指令执行</p> <p>无</p>

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	$ACC \leftarrow [m]$ ，如果 $[m]=0$ ，跳过下一条指令执行
影响标志位	无
SZ [m].i	Skip if bit i of Data Memory is 0
指令说明	判断指定数据存储器的第 i 位是否为 0，若为 0，则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	如果 $[m].i=0$ ，跳过下一条指令执行
影响标志位	无
TABRD [m]	Read table (specific page) to TBLH and Data Memory
指令说明	将表格指针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定数据存储器且将高字节移至 TBLH。
功能表示	$[m] \leftarrow$ 程序代码 (低字节) $TBLH \leftarrow$ 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	$[m] \leftarrow$ 程序代码 (低字节) $TBLH \leftarrow$ 程序代码 (高字节)
影响标志位	无
ITABRD [m]	Increment table pointer low byte first and read table to TBLH and data memory
指令说明	将自加表格指针 TBLP 所指的程序代码低字节 (当前页) 移至指定的数据存储器且将高字节移至 TBLH。
功能表示	$[m] \leftarrow$ 程序代码 (低字节) $TBLH \leftarrow$ 程序代码 (高字节)
影响标志位	无

ITABRDL [m]	Increment table pointer low byte first and read table(last page) to TBLH and data memory
指令说明	将自加表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定的数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
XOR A, [m]	Logical XOR Data Memory to ACC
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或, 结果存放到累加器。
功能表示	ACC ← ACC “XOR” [m]
影响标志位	Z
XORM A, [m]	Logical XOR ACC to Data Memory
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或, 结果放到数据存储器。
功能表示	[m] ← ACC “XOR” [m]
影响标志位	Z
XOR A, x	Logical XOR immediate data to ACC
指令说明	将累加器的数据与立即数逻辑异或, 结果存放到累加器。
功能表示	ACC ← ACC “XOR” x
影响标志位	Z

扩展指令定义

扩展指令被用来直接存取存储在任何数据存储器 Sector 中的数据。

LADC A, [m]	Add Data Memory to ACC with Carry
指令说明	将指定的数据存储器、累加器内容以及进位标志相加，结果存放到累加器。
功能表示	$ACC \leftarrow ACC + [m] + C$
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	指将指定数据存储器的 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
LORA, [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，结果存放回累加器。
功能表示	$ACC \leftarrow ACC \text{ "OR" } [m]$
影响标志位	Z

LORMA, [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	$[m] \leftarrow \text{ACC} \text{ "OR" } [m]$
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	$\text{ACC}.(i+1) \leftarrow [m].i \ (i=0\sim6)$ $\text{ACC}.0 \leftarrow [m].7$
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
LRLCA [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	$\text{ACC}.(i+1) \leftarrow [m].i \ (i=0\sim6)$ $\text{ACC}.0 \leftarrow C$ $C \leftarrow [m].7$
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=0~6) $[m].7 \leftarrow [m].0$
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=0~6) $ACC.7 \leftarrow [m].0$
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=0~6) $[m].7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位	C
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC
指令说明	将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位，移位结果送回累加器，但是指定数据寄存器的内容保持不变。
功能表示	$ACC.i \leftarrow [m].(i+1)$ (i=0~6) $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
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

<p>LSBCMA, [m]</p> <p>指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Subtract Data Memory from ACC with Carry and result in Data Memory</p> <p>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。</p> <p>$[m] \leftarrow ACC - [m] - \bar{C}$</p> <p>OV、Z、AC、C、SC、CZ</p>
<p>LSDZ [m]</p> <p>指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Skip if Decrement Data Memory is 0</p> <p>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。</p> <p>$[m] \leftarrow [m] - 1$，如果 $[m]=0$ 跳过下一条指令执行</p> <p>无</p>
<p>LSDZA [m]</p> <p>指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Skip if decrement Data Memory is zero with result in ACC</p> <p>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。</p> <p>$ACC \leftarrow [m] - 1$，如果 $ACC=0$ 跳过下一条指令执行</p> <p>无</p>
<p>LSET [m]</p> <p>指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Set Data Memory</p> <p>将指定数据存储器的每一个位置位为1。</p> <p>$[m] \leftarrow FFH$</p> <p>无</p>
<p>LSET [m].i</p> <p>指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Set bit of Data Memory</p> <p>将指定数据存储器的第i位置位为1。</p> <p>$[m].i \leftarrow 1$</p> <p>无</p>

LSIZ [m]	Skip if increment Data Memory is 0
指令说明	将指定的数据存储器内容加 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
LSIZA [m]	Skip if increment Data Memory is zero with result in ACC
指令说明	将指定数据存储器内容加 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
LSNZ [m].i	Skip if bit i of Data Memory is not 0
指令说明	判断指定数据存储器的第 i 位，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果为 0，则程序继续执行下一条指令。
功能表示	如果 $[m].i \neq 0$ ，跳过下一条指令执行
影响标志位	无
LSNZ [m]	Skip if Data Memory is not 0
指令说明	判断指定数据存储器，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果为 0，则程序继续执行下一条指令。
功能表示	如果 $[m] \neq 0$ ，跳过下一条指令执行
影响标志位	无
LSUB A, [m]	Subtract Data Memory from ACC
指令说明	将累加器的内容减去指定的数据存储器的数据，把结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$ACC \leftarrow ACC - [m]$
影响标志位	OV、Z、AC、C、SC、CZ

LSUBMA, [m] 指令说明	Subtract Data Memory from ACC with result in Data Memory 将累加器的内容减去指定数据寄存器的数据，结果存放到指定的数据寄存器。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] 指令说明	Swap nibbles of Data Memory 将指定数据寄存器的低 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] 指令说明	Swap nibbles of Data Memory with result in ACC 将指定数据寄存器的低 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] 指令说明	Skip if Data Memory is 0 判断指定数据寄存器的内容是否为 0，若为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	如果 $[m]=0$ 跳过下一条指令执行
影响标志位	无
LSZA [m] 指令说明	Skip if Data Memory is 0 with data movement to ACC 将指定数据寄存器内容复制到累加器，并判断指定数据寄存器的内容是否为 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	$ACC \leftarrow [m]$ ，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 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	如果 [m].i=0，跳过下一条指令执行
影响标志位	无
LTABRD [m]	Move the ROM code to TBLH and data memory
指令说明	将表格指针 TBLP 所指的程序代码低字节 (当前页) 移至指定数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
LTABRDL [m]	Read table (last page) to TBLH and Data Memory
指令说明	将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
LITABRD [m]	Increment table pointer low byte first and read table to TBLH and data memory
指令说明	将自加表格指针 TBHP 和 TBLP 所指的程序代码低字节移至指定的数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
LITABRDL [m]	Increment table pointer low byte first and read table(last page) to TBLH and data memory
指令说明	将自加表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定的数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无

LXOR A, [m]	Logical XOR Data Memory to ACC
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。
功能表示	$ACC \leftarrow ACC \text{ "XOR" } [m]$
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

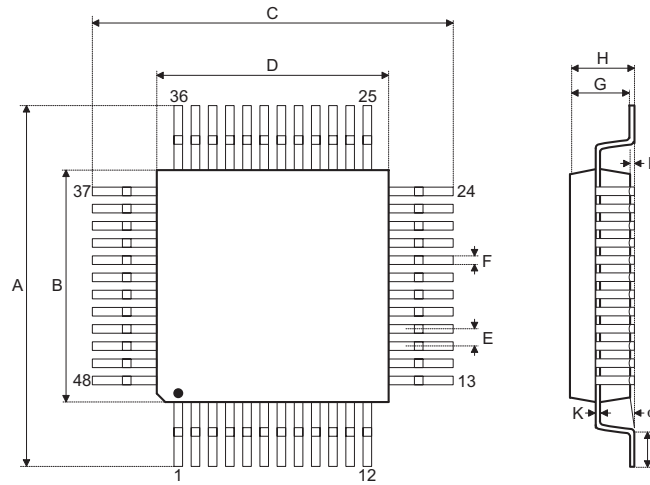
封装信息

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

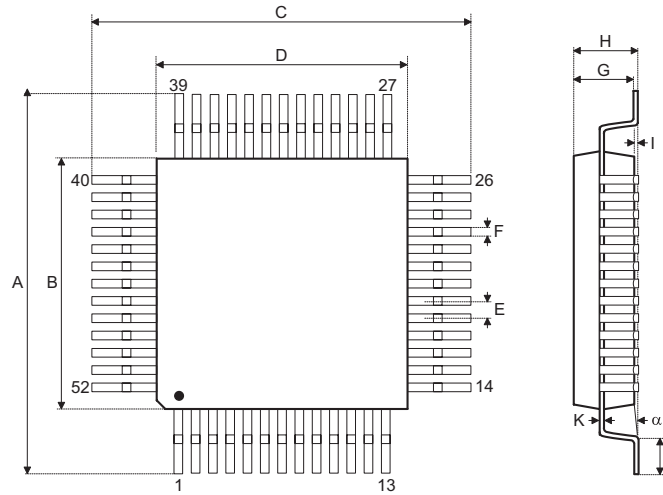
48-pin LQFP (7mm × 7mm) 外形尺寸



符号	尺寸 (单位: inch)		
	最小	正常	最大
A	—	0.354 BSC	—
B	—	0.276 BSC	—
C	—	0.354 BSC	—
D	—	0.276 BSC	—
E	—	0.020 BSC	—
F	0.007	0.009	0.011
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

符号	尺寸 (单位: mm)		
	最小	正常	最大
A	—	9.000 BSC	—
B	—	7.000 BSC	—
C	—	9.000 BSC	—
D	—	7.000 BSC	—
E	—	0.500 BSC	—
F	0.170	0.220	0.270
G	1.350	1.400	1.450
H	—	—	1.600
I	0.050	—	0.150
J	0.450	0.600	0.750
K	0.090	—	0.200
α	0°	—	7°

52-pin LQFP (14mm × 14mm) 外形尺寸



符号	尺寸 (单位: inch)		
	最小	正常	最大
A	0.622	0.630	0.638
B	0.547	0.551	0.555
C	0.622	0.630	0.638
D	0.547	0.551	0.555
E	—	0.039 BSC	—
F	0.015	—	0.019
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.008
J	0.018	—	0.030
K	0.005	—	0.007
α	0°	—	7°

符号	尺寸 (单位: mm)		
	最小	正常	最大
A	15.80	16.00	16.20
B	13.90	14.00	14.10
C	15.80	16.00	16.20
D	13.90	14.00	14.10
E	—	1.00 BSC	—
F	0.39	—	0.48
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.20
J	0.45	—	0.75
K	0.13	—	0.18
α	0°	—	7°

Copyright© 2016 by HOLTEK SEMICONDUCTOR INC.

使用指南中所出现的信息在出版当时相信是正确的，然而盛群对于说明书的使用不负任何责任。文中提到的应用目的仅仅是用来做说明，盛群不保证或表示这些没有进一步修改的应用将是适当的，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。盛群产品不授权使用于救生、维生从机或系统中做为关键从机。盛群拥有不事先通知而修改产品的权利，对于最新的信息，请参考我们的网址 <http://www.holtek.com/zh/>.