内置 LCD & EEPROM 低功耗 A/D 型 Flash 单片机

HT67F488/HT67F489

版本：V1.70　日期：2022-12-19

www.holtek.com

目录

注意，规格中提到的 **HT67F488** 产品已终止，目前不再使用。

特性

CPU 特性

- 工作电压
  - ♦ f_{SYS}=4MHz：2.2V~5.5V
  - ♦ f_{SYS}=8MHz：2.2V~5.5V
  - ♦ f_{SYS}=12MHz：2.7V~5.5V
  - ♦ f_{SYS}=16MHz：4.5V~5.5V
- 提供暂停和唤醒功能，以降低功耗
- 振荡器类型
  - ♦ 高速振荡器：HIRC
  - ♦ 低速振荡器：LXT/LIRC
  - ♦ 外部晶振：HXT
- 内建 8MHz 振荡器，无需外部元件
- 多种工作模式：正常、低速、空闲和休眠
- 所有指令都可在 1~3 个指令周期内完成
- 位操作指令：16 位查表功能
- 115 条功能强大的指令系统
- 8 层堆栈

周边特性

- Flash 程序存储器：4K×16~8K×16
- RAM 数据存储：256×8
- True EEPROM 存储器：64×8 (仅适用于 HT67F489)
- 看门狗定时器
- 42 个双向输入 / 输出口
  包括 LCD/LED 驱动输出
- 4 个与 I/O 口复用的外部中断输入
- 多个定时器模块用于时间测量、捕捉输入、比较匹配输出、PWM 输出及单脉冲输出
- 双时基功能用以产生固定的中断信号
- 10 通道 12-bit A/D 转换器
- LCD 显示
  20SEG×8COM & 20SEG×4COM
  1/3 或 1/4 bias
- LED 显示：8SEG×8COM
- 全双工通用异步收发接口 – UART
- 低电压复位功能
- 低电压检测功能
- 封装类型：44-pin LQFP

概述

HT67F488/HT67F489 是一款 A/D 型具有 8 位高性能精简指令集的 Flash 单片机，专门为需要 A/D 转换的产品而设计，如传感器等。该系列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较大的方便。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 (仅 HT67F489)。

在模拟特性方面，该系列单片机包含一个多通道 12 位 A/D 转换器。还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。

该系列单片机提供了丰富的 HXT、HIRC、LXT 和 LIRC 振荡器功能选项，且内建完整的系统振荡器，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。

该系列单片机内含 UART 模块，它可以支持诸如单片机之间的数据通信网络，低成本 PC 和外部设备间的数据连接，便携式和电池供电设备间的通信等。

LCD 和 LED 驱动器功能为一些需要显示类型设备接口的应用提供了一种简单有效的解决方法。

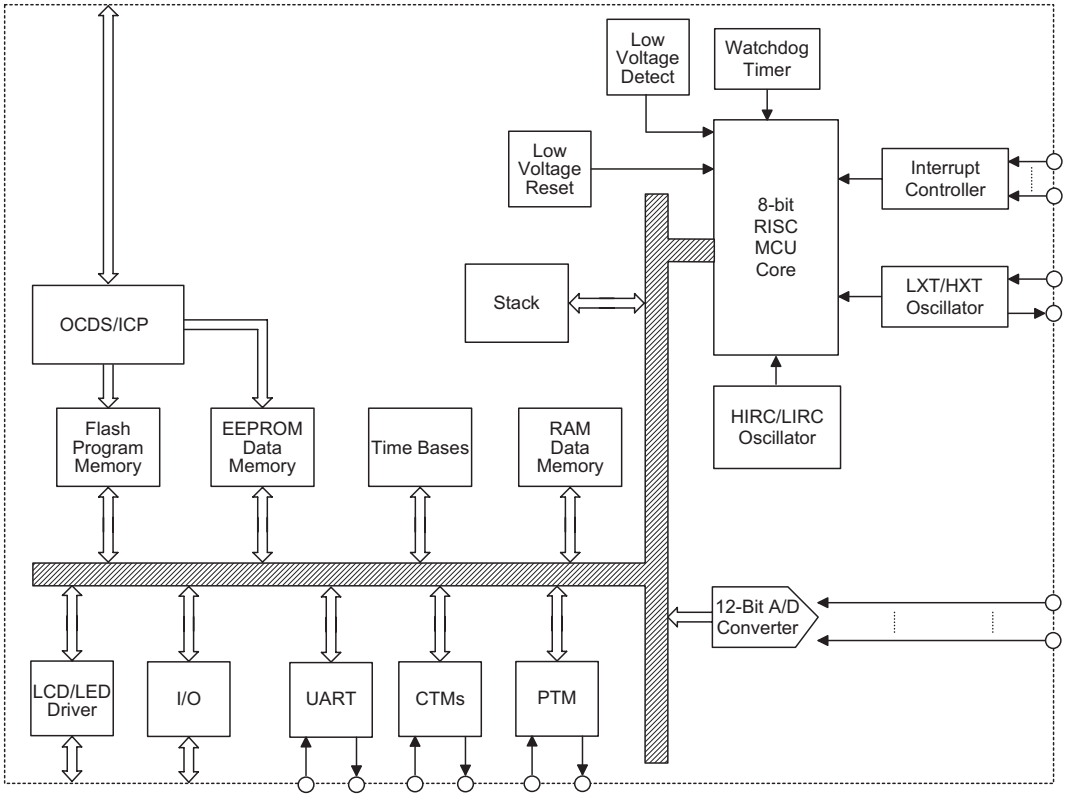
外加时基功能、I/O 使用灵活等其它特性，使该系列单片机可以广泛适用于各种需要 A/D 转换的应用，例如传感器信号处理、电机驱动、工业控制、消费类产品和子系统控制等。

选型表

对此系列的单片机而言，大多数的特性参数都是一样的。主要差异是在于程序存储器的容量和是否有 EEPROM。下标列出各单片机的主要特性。

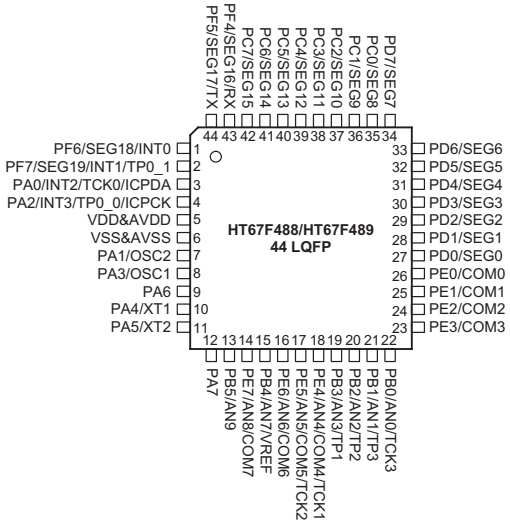
型号	ROM	RAM	EEPROM	I/O	外部中断	A/D	TM 模块	时基	UART	堆栈	封装
HT67F488	4K×16	256×8	—	42	4	12-bit×10	10-bit CTM×3 10-bit PTM×1	2	√	8	44LQFP
HT67F489	8K×16	256×8	64 × 8	42	4	12-bit×10	10-bit CTM×3 10-bit PTM×1	2	√	8	44LQFP

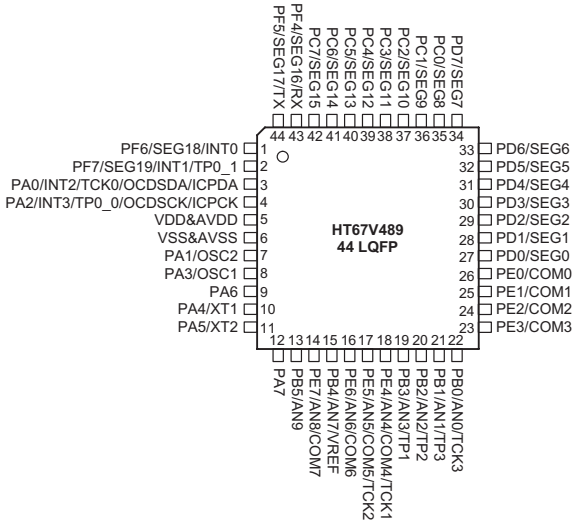
方框图



注：EEPROM 数据存储器仅适用于 HT67F489 单片机。

引脚图





引脚说明

引脚名称	功能	OP	I/T	O/T	说明
PA0/INT2/TCK0/OCDSDA/ICPDA	PA0	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	INT2	—	ST	—	外部中断 2 输入
	TCK0	—	ST	—	TM0 时钟输入
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址，仅用于 EV 芯片
	ICPDA	—	ST	CMOS	ICP 数据 / 地址
PA1/OSC2	PA1	PAPU PAWU	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	OSC	—	HXT	高频率晶振引脚
PA3/OSC1	PA3	PAPU PAWU	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	OSC	HXT	—	高频率晶振引脚
PA6, PA7	PA6, PA7	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
PA2/INT3/TP0_0/OCDSCK/ICPCK	PA2	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	INT3	—	ST	—	外部中断 3 输入
	TP0_0	TMPC	ST	CMOS	TM0 输入 / 输出
	OCDSCK	—	ST	—	OCDS 时钟引脚，仅用于 EV 芯片
	ICPCK	—	ST	—	ICP 时钟引脚
PA4/XT1	PA4	PAPU PAWU	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	CO	LXT	—	LXT 引脚
PA5/XT2	PA5	PAPU PAWU	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	CO	—	LXT	LXT 引脚
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 通道 0
	TCK3	—	ST	—	TM3 时钟输入
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 通道 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 通道 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 通道 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 通道 7
	VREF	ADCR1	AN	—	ADC 参考输入

引脚名称	功能	OP	I/T	O/T	说明
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 通道 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	—	CMOS	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	—	CMOS	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	—	CMOS	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	—	CMOS	LCD SEG 输出
PC4/SEG12	PC4	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG12	SEGCR1	—	CMOS	LCD SEG 输出
PC5/SEG13	PC5	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG13	SEGCR1	—	CMOS	LCD SEG 输出
PC6/SEG14	PC6	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG14	SEGCR1	—	CMOS	LCD SEG 输出
PC7/SEG15	PC7	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG15	SEGCR1	—	CMOS	LCD SEG 输出
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	—	CMOS	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	—	CMOS	LCD SEG 输出
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	—	CMOS	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	—	CMOS	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	—	CMOS	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	—	CMOS	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	—	CMOS	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	—	CMOS	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	—	CMOS	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	—	CMOS	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	—	CMOS	LCD COM 输出

引脚名称	功能	OP	I/T	O/T	说明
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	—	CMOS	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 通道 4
	COM4	LCDC0	—	CMOS	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 通道 5
	COM5	LCDC0	—	CMOS	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 通道 6
	COM6	LCDC0	—	CMOS	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	ACERH	AN	—	A/D 通道 8
	COM7	LCDC0	—	CMOS	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	—	CMOS	LCD SEG 输出
	RX	—	ST	—	UART 接收数据输入
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	—	CMOS	LCD SEG 输出
	TX	—	—	CMOS	UART 发送数据输出
PF6/SEG18/ INT0	PF6	PFPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG18	SEGCR2	—	CMOS	LCD SEG 输出
	INT0	—	ST	—	外部中断 0 输入
PF7/SEG19/ INT1/TP0_1	PF7	PFPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG19	SEGCR2	—	CMOS	LCD SEG 输出
	INT1	—	ST	—	外部中断 1 输入
	TP0_1	TMPC	ST	CMOS	TM0 输入 / 输出
AVDD	AVDD	—	PWR	—	ADC 正电源
VDD	VDD	—	PWR	—	正电源
AVSS	AVSS	—	PWR	—	ADC 负电源
VSS	VSS	—	PWR	—	负电源、接地

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

极限参数

电源供应电压 ..... V_{SS}-0.3V~V_{SS}+6.0V
输入电压 ..... V_{SS}-0.3V~V_{DD}+0.3V
储存温度 ..... -60°C~150°C
工作温度 ..... -40°C~85°C
I_{OH} 总电流 ..... -80mA
I_{OL} 总电流 80mA
总功耗 500mW

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

直流电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压 (HXT)	—	f_{SYS}=4MHz	2.2	—	5.5	V
			f_{SYS}=8MHz	2.2	—	5.5	V
			f_{SYS}=12MHz	2.7	—	5.5	V
			f_{SYS}=16MHz	4.5	—	5.5	V
I_{DD}	工作电流 , 正常模式 $f_{SYS}=f_H$, $f_{SUB}=f_{LXT}$ or f_{LIRC}	3V	无负载 , $f_{SYS}=8MHz$, ADC off, WDT 使能	—	1.6	2.4	mA
		5V		—	3.3	5.0	mA
	工作电流 , 正常模式 , f_H=8MHz	3V	无负载 , $f_{SYS}=f_H/2$, ADC off, WDT 使能	—	0.9	1.5	mA
		5V		—	2.5	3.75	mA
		3V	无负载 , $f_{SYS}=f_H/4$, ADC off, WDT 使能	—	0.7	1.0	mA
		5V		—	2.0	3.0	mA
		3V	无负载 , $f_{SYS}=f_H/8$, ADC off, WDT 使能	—	0.6	0.9	mA
		5V		—	1.6	2.4	mA
		3V	无负载 , $f_{SYS}=f_H/16$, ADC off, WDT 使能	—	0.5	0.75	mA
		5V		—	1.5	2.25	mA
		3V	无负载 , $f_{SYS}=f_H/32$, ADC off, WDT 使能	—	0.49	0.74	mA
		5V		—	1.45	2.18	mA
		3V	无负载 , $f_{SYS}=f_H/64$, ADC off, WDT 使能	—	0.47	0.71	mA
		5V		—	1.4	2.1	mA

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，低速模式 $f_{SYS}=f_L$ (LXT, LIRC)	3V	无负载，f_{SYS}=LXT, ADC off, WDT 使能，LXTLP=0, LVR 使能	—	45	75	μA
		5V		—	90	140	μA
		3V	无负载，f_{SYS}=LXT, ADC off, WDT 使能，LXTLP=1, LVR 使能	—	40	70	μA
		5V		—	85	135	μA
		3V	无负载，f_{SYS}=LIRC, ADC off, WDT 使能，LVR 使能	—	40	65	μA
		5V		—	80	130	μA
I_{IDLE01}	IDLE0 模式待机电流 (LXT on)	3V	无负载，ADC off, WDT 使能，LXTLP=0	—	2	4	μA
		5V		—	4	8	μA
		3V	无负载，ADC off, WDT 使能，LXTLP=1	—	1.5	3.0	μA
		5V		—	3.0	6.0	μA
I_{IDLE02}	IDLE0 模式待机电流 (LIRC on)	3V	无负载，ADC off, WDT 使能	—	1.5	3.0	μA
		5V		—	3.0	6.0	μA
I_{IDLE03}	IDLE0 模式待机电流 (LXT on)	3V	无负载，ADC off, WDT 使能，LXTLP=1, LCD 使能 (R_T=1170K, 无快速充电，$V_{LCD}=V_{DD}$)	—	3	6	μA
		5V		—	6	12	μA
I_{IDLE04}	IDLE0 模式待机电流 (LXT on)	3V	无负载，ADC off, WDT 使能，LXTLP=1, LCD 使能 (R_T=225K, 无快速充电，$V_{LCD}=V_{DD}$)	—	14	28	μA
		5V		—	24	48	μA
I_{IDLE05}	IDLE0 模式待机电流 (LXT on)	3V	无负载，ADC off, WDT 使能，LXTLP=1, LCD 使能 (R_T=1170K, 快速充电，QCT[2:0]=0, $V_{LCD}=V_{DD}$)	—	5	10	μA
		5V		—	9	18	μA
I_{IDLE06}	IDLE0 模式待机电流 (LXT on)	3V	无负载，ADC off, WDT 使能，LXTLP=1, LCD 使能 (R_T=1170K, 快速充电，QCT[2:0]=7, $V_{LCD}=V_{DD}$)	—	11	22	μA
		5V		—	18	36	μA
I_{IDLE1}	IDLE1 模式待机电流 (LIRC on)	3V	无负载，ADC off, WDT 使能，f_{SYS}=8MHz on	—	0.5	3.0	mA
		5V		—	1.0	6.0	mA
I_{SLEEP0}	SLEEP0 模式待机电流 (LXT 和 LIRC off)	3V	无负载，ADC off, WDT 除能	—	0.2	1	μA
		5V		—	0.4	2	μA
I_{SLEEP1}	SLEEP1 模式待机电流 (LXT 或 LIRC on)	3V	无负载，ADC off, WDT 使能	—	1.5	3.0	μA
		5V		—	2.5	5.0	μA
V_{IL}	输入 / 输出口或其它引脚的低电平输入电压	—	—	0	—	$0.3V_{DD}$	V
V_{IH}	输入 / 输出口或其它引脚的高电平输入电压	—	—	$0.7V_{DD}$	—	V_{DD}	V

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
GPIO (除了 PD0~PD7 & PE0~PE7)							
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	—	mA
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	—	mA
大灌电流 I/O 口用于 LED 驱动器 (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	—	mA
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	—	mA
源电流可调节 I/O 口用于 LED 驱动器 (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	—	mA
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	—	mA
			V_{OH}=0.9V_{DD} (IOHSn[1:0]=10B, n=0~7)	-0.5	-1	—	mA
			V_{OH}=0.9V_{DD} (IOHSn[1:0]=11B, n=0~7)	-0.33	-0.66	—	mA
		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	—	mA
			V_{OH}=0.9V_{DD} (IOHSn[1:0]=10B, n=0~7)	-1.25	-2.5	—	mA
			V_{OH}=0.9V_{DD} (IOHSn[1:0]=11B, n=0~7)	-0.83	-1.67	—	mA
R_{PH}	输入 / 输出口上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
R_T	LCD 总偏置电阻	3V/5V	—	-30	R_T	+30	%
I_{TOL}	所有 I/O 口灌电流总和	5V	—	80	—	—	mA
I_{TOH}	所有 I/O 口源电流总和	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	2.2V~5.5V	—	DC	—	4	MHz
		2.2V~5.5V	—	DC	—	8	MHz
		2.7V~5.5V	—	DC	—	12	MHz
		4.5V~5.5V	—	DC	—	16	MHz
f_{SYS}	系统时钟 (HIRC)	2.2V~5.5V	—	—	8	—	MHz
f_{HIRC}	系统时钟 (HIRC)	4.5V~5.5V	Ta=0°C~70°C	-2%	8	+2%	MHz
f_{LIRC}	系统时钟 (LIRC)	5V	Ta=25°C	-10%	32	+10%	kHz
t_{TIMER}	TCKn 输入脉宽	—	—	0.3	—	—	μs
t_{INT}	中断脉宽	—	—	10	—	—	μs
t_{EERD}	EEPROM 读周期	5V	—	—	2	4	t_{SYS}
t_{EEWR}	EEPROM 写周期	5V	—	—	2	4	ms
t_{RSTD}	系统复位延迟时间 (上电复位 , LVR, WDTC/LVRC 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 (从 HALT 中唤醒)	—	f_{SYS}=LXT/HXT	—	1024	—	t_{SYS}
			f_{SYS}=HIRC	—	16	—	
			f_{SYS}=LIRC	—	2	—	
t_{SRESET}	软件复位时间	—	—	45	90	120	μs

注：$t_{SYS}=1/f_{SYS}$

ADC 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 工作电压	—	—	2.7	—	5.5	V
V_{ADI}	A/D 输入电压	—	—	0	—	AV_{DD}/V_{REF}	V
V_{REF}	A/D 参考电压	—	—	2	—	AV_{DD}	V
V_{BG}	参考缓冲电压	—	—	-3%	1.09	+3%	V
DNL	A/D 非线性微分误差	5V	$V_{REF}=AV_{DD}=V_{DD}$ t_{ADCK}=0.5μs	-4	—	+4	LSB
INL	A/D 非线性积分误差	5V	$V_{REF}=AV_{DD}=V_{DD}$ t_{ADCK}=0.5μs	-7	—	+7	LSB
I_{ADC}	使用 A/D 的额外功耗	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	mA
I_{BG}	使用 V_{BG} 的额外功耗	—	—	—	200	300	μA
t_{ADCK}	A/D 转换器时钟周期	—	—	0.5	—	10	μs
t_{ADC}	A/D 转换时间 (包括采样和保持时间)	—	12 bit ADC	—	16	—	t_{ADCK}

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
t_{ADS}	A/D 转换器采样时间	—	—	—	4	—	t_{ADCK}
t_{ON2ST}	A/D 转换器 On-to-Start 时间	—	—	2	—	—	μs
t_{BGS}	V_{BG} 开启到稳定的时间	—	—	—	—	200	μs

LVD&LVR 电气特性

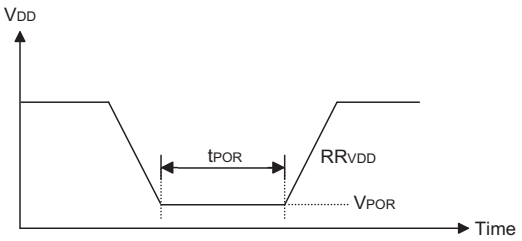
T=25°C

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 , 选择 2.1V	-5%	2.1	+5%	V
			LVR 使能 , 选择 2.55V		2.55		V
			LVR 使能 , 选择 3.15V		3.15		V
			LVR 使能 , 选择 3.8V		3.8		V
V_{LVD}	低电压检测电压	—	LVDEN=1, V_{LVD}=2.0V	-5%	2.0	+5%	V
			LVDEN=1, V_{LVD}=2.2V		2.2		V
			LVDEN=1, V_{LVD}=2.4V		2.4		V
			LVDEN=1, V_{LVD}=2.7V		2.7		V
			LVDEN=1, V_{LVD}=3.0V		3.0		V
			LVDEN=1, V_{LVD}=3.3V		3.3		V
			LVDEN=1, V_{LVD}=3.6V		3.6		V
			LVDEN=1, V_{LVD}=4.0V		4.0		V
I_{LVD}	使用 LVD 的额外功耗	3V	LVD 除能 →LVD 使能 (LVR 使能)	—	30	45	μA
		5V		—	60	90	μA
t_{LVR}	低电压复位时间	—	—	120	240	480	μs
t_{LVD}	低电压中断时间	—	—	20	45	90	μs
t_{LVDS}	LVDO 稳定时间	—	LVR 使能 , LVD off→on	—	—	15	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_{VDD}	上电复位电压速率	—	—	0.035	—	—	V/ms
t_{POR}	V_{DD} 保持为 V_{POR} 的最小时间	—	—	1	—	—	ms

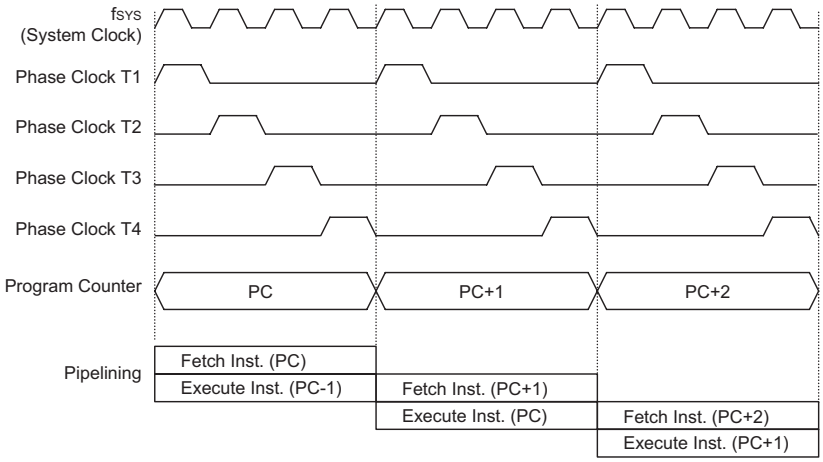


系统结构

内部系统结构是盛群单片机具有良好性能的主要因素。由于采用 RISC 结构，该系列单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的取得和执行同时进行，此举使得除了跳转和调用指令需多一个指令周期外，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、递增、递减和分支等功能，而内部的数据路径则是以通过累加器和 ALU 的方式加以简化。有些寄存器在数据存储器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供具有较大可靠度和灵活性的 I/O 和 A/D 控制系统时，仅需要少数的外部器件。使得这些单片机适用于低成本和大量生产的控制应用。

时序和流水线结构

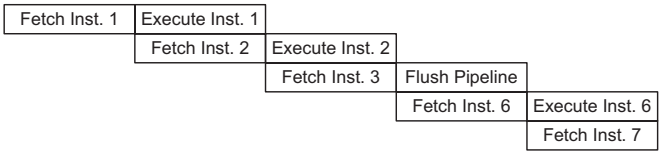
主系统时钟由 HXT、HIRC、LXT 或 LIRC 振荡器提供，它被细分为 T1~T4 四个内部产生的非重叠时序。在 T1 时间，程序计数器自动加一并抓取一条新的指令。剩下的时间 T2~T4 完成译码和执行功能，因此，一个 T1~T4 时钟周期构成一个指令周期。虽然指令的抓取和执行发生在连续的指令周期，但单片机流水线结构会保证指令在一个指令周期内被有效执行。除非程序计数器的内容被改变，如子程序的调用或跳转，在这种情况下指令将需要多一个指令周期的时间去执行。



系统时序和流水线

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

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



指令捕捉

程序计数器

在程序执行期间，程序计数器用来指向下一个要执行的指令地址。除了“JMP”和“CALL”指令需要跳转到一个非连续的程序存储器地址之外，它会在每条指令执行完成以后自动加一。选择不同型号的单片机，程序寄存器的宽度会因程序存储器的容量的不同而不同。只有较低的 8 位，即所谓的程序计数器低字节寄存器 PCL，可以被用户直接读写。

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

单片机型号	程序计数器	
	程序计数器高字节	PCL 寄存器
HT67F488	PC11~PC8	PCL7~PCL0
HT67F489	PC12~PC8	PCL7~PCL0

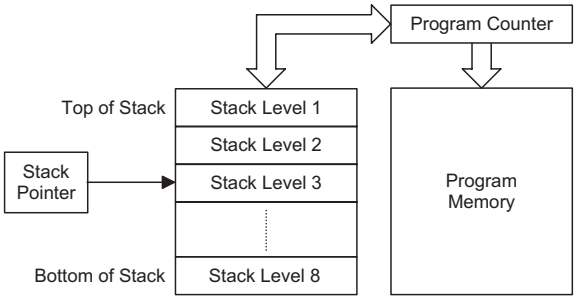
程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。程序计数器的低字节可由程序直接进行读取，PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

堆栈

堆栈是一个特殊的存储空间，用来存储程序计数器中的内容。该系列单片机有 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、LRR、LRRA、LRRCA、LRRC、LRLA、LRL、LRLCA、LRLC
- 递增和递减：INCA、INC、DECA、DEC、LINCA、LINC、LDECA、LDEC
- 分支判断：JMP、SZ、SZA、SNZ、SIZ、SDZ、SIZA、SDZA、CALL、RET、RETI、LSZ、LSZA、LSNZ、LSIZ、LSDZ、LSIZA、LSDZA

Flash 程序存储器

程序存储器用来存放用户代码即储存程序。程序存储器为 Flash 类型意味着可以多次重复编程，方便用户使用同一芯片进行程序的修改。使用适当的单片机编程工具，此所有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。

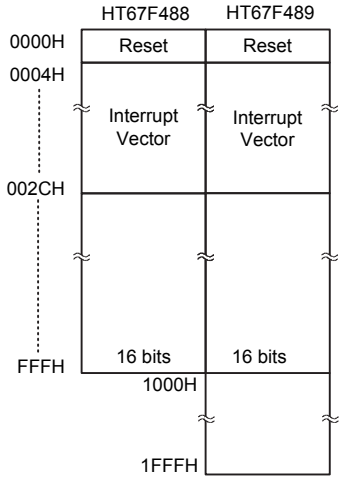
结构

程序存储器的容量为 4K×16 或 8K×16 位，程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口。数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。

单片机型号	容量
HT67F488	4K×16
HT67F489	8K×16

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。



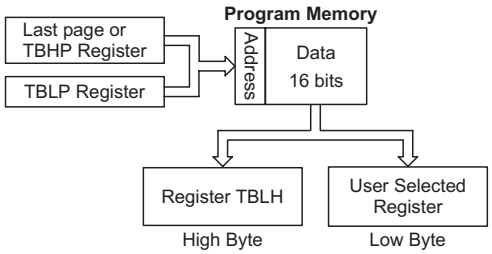
程序存储器结构

查表

程序存储器中的任何地址都可以定义成一个表格，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 TBLH 中。这些寄存器定义表格总的地址。

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

下图是查表中寻址 / 数据流程：



查表范例

以下范例说明表格指针和表格数据如何被定义和执行。这个例子使用的表格数据用 ORG 伪指令储存在存储器中。ORG 指令的值“F00H”指向的地址是 HT67F488 的 4K 程序存储器中最后一页的起始地址。表格指针的初始值设为 06H，这可保证从数据表格读取的第一笔数据位于程序存储器地址 F06H，即最后一页起始地址后的第六个地址。值得注意的是，假如“TABRD [m]”指令被使用，则表格指针指向当前页。在这个例子中，表格数据的高字节等于零，而当“TABRD [m]”指令被执行时，此值将会自动的被传送到 TBLH 寄存器。

TBLH 寄存器为可读 / 写寄存器，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 in current page
tempreg2 db ?              ; temporary register #2 in current page
:
:
mov a,06h                  ; initialise low table pointer - note that this
                           ; address is referenced
mov tblp,a                 ; to the last page or present page
:
:
tabrdl tempreg1            ; transfers value in table referenced by table
                           ; pointer to tempreg1
                           ; Data at program memory address "F06H"
                           ; transferred to tempreg1 and TBLH
dec tblp                   ; reduce value of table pointer by one
tabrdl tempreg2            ; transfers value in table referenced by table
                           ; pointer to tempreg2
                           ; Data at program memory address "F05H"
                           ; transferred to tempreg2 and TBLH
                           ; in this example the data "1AH" is transferred
                           ; to tempreg1 and data "0FH" to register tempreg2
                           ; while the value "00H" will be transferred to
                           ; the high byte register TBLH
:
:
org 0F00h                  ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

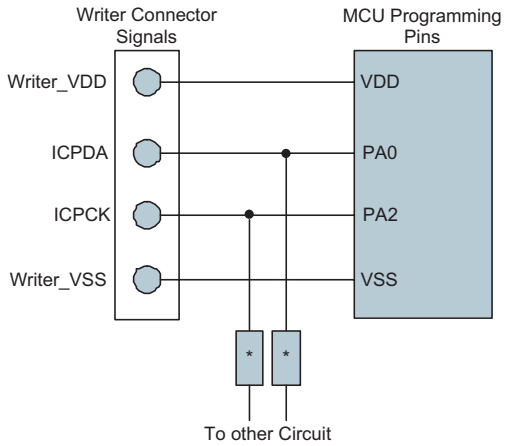
在线烧录

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

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

烧录过程中，用户必须确保 PA0 和 PA2 这两个引脚没有连接至其它输出脚。

程序存储器和 EEPROM 存储器可以通过 4 线的接口在线进行烧录。其中 PA0 用于数据串行下载或上传、PA2 用于串行时钟、两条用于提供电源。芯片在线烧写的详细使用说明超出此文档的描述范围，将由专门的参考文献提供。



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

EV 芯片 HT67V489 用于 HT67F489/488 单片机仿真。此 EV 芯片 HT67V489 提供片上调试功能 (OCDS) 用于开发过程中的 HT67F489/488 单片机调试。除了片上调试功能，HT67F489/488 和 HT67V489 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 HT67V489 对 HT67F489/488 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 HT67V489 进行调试时，HT67F489/488 单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能对 EV 芯片 HT67V489 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 User's Guide”文件。

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

数据存储器

数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。

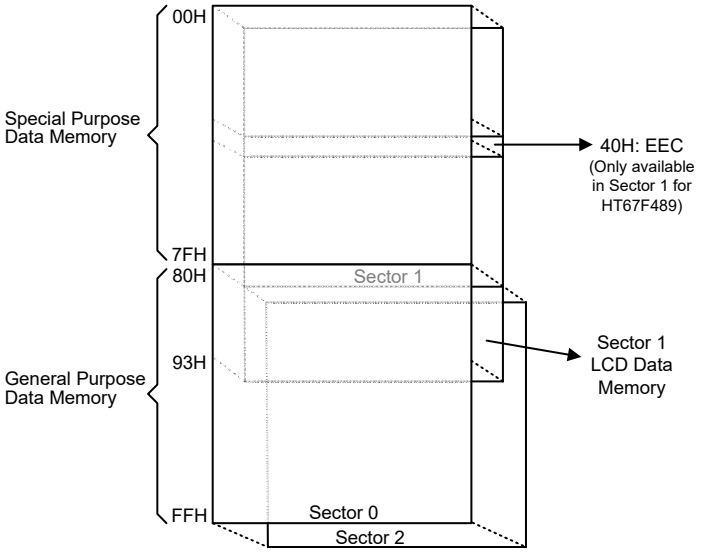
数据存储器分为两个区，第一部分是特殊功能数据存储器。这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二部分数据存储器是做一般用途使用，都可在程序控制下进行读取和写入。

结构

数据存储器被分为 5 个 Sector，都位于 8 位存储器中。每个数据存储器 Sector 分为两类，特殊功能数据存储器和通用数据存储器。

特殊功能数据存储器起始地址为“00H”，而通用数据存储器起始地址为“80H”。大部分特殊功能数据寄存器均可在所有 Sector 被访问，处于“40H”地址的 EEC 寄存器却只能在 Sector 1 中被访问到。

单片机型号	容量	Sectors
HT67F488 HT67F489	通用数据存储器：256×8	0：80H~FFH 1：80H~93H (用于 LCD) 2：80H~FFH



数据存储器结构

数据存储器寻址

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

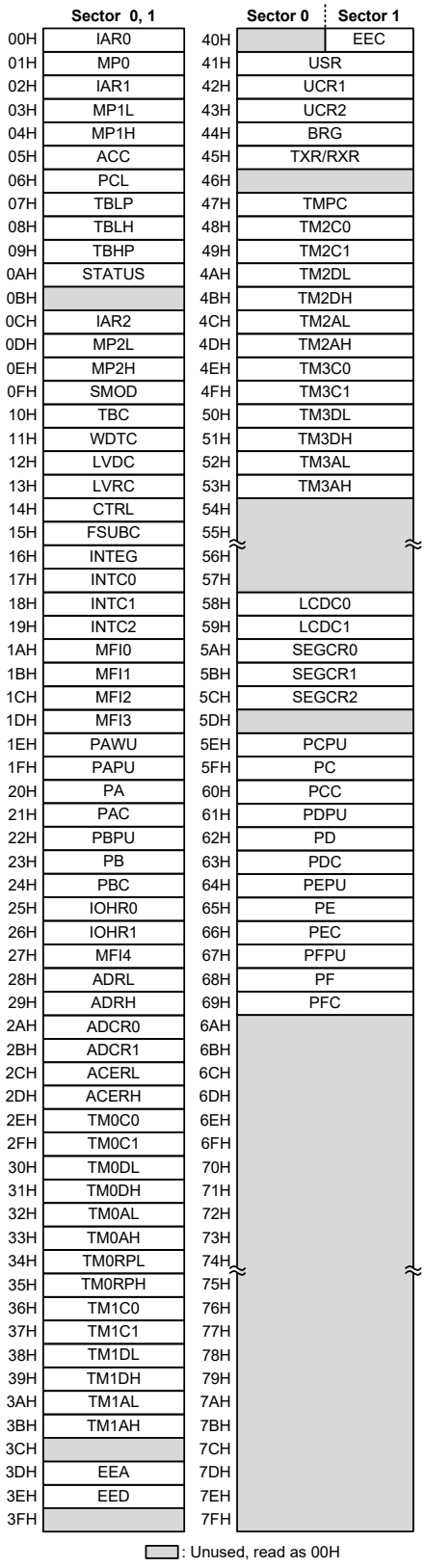
直接寻址可用于所有 Sector，通过相应的指令可以寻址所有可用的数据存储器空间。所访问的数据存储器位于除 Sector 0 外的任何 Sector，可使用扩展指令代替间接寻址方式来访问数据存储器。标准指令和扩展指令的主要区别在于扩展指令中的数据存储器地址“m”是 10 个有效位，高字节表示某一 Sector，低字节表示某一指定地址。

通用数据存储器

通用数据存储器共 256 字节位于 Sector 0、Sector 2 的 80H~FFH。LCD 存储器共 20 字节映射于 Sector 1 的 80H~93H。所有的单片机程序需要一个读 / 写的存储区，让临时数据可以被储存和再使用，该 RAM 区域就是通用数据存储器。这个数据存储区可让使用者进行读取和写入的操作。使用位操作指令可对个别的位做置位或复位的操作，较大地方便了用户在数据存储器内进行位操作。

特殊功能数据存储器

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



特殊功能数据存储器结构

特殊功能寄存器

大部分特殊功能寄存器的细节将在相关功能章节描述，但有几个寄存器需在此章节单独描述。

间接寻址寄存器 – IAR0，IAR1，IAR2

间接寻址寄存器 IAR0、IAR1 和 IAR2 的地址虽位于数据存储区，但其并没有实际的物理地址。间接寻址的方法准许使用间接寻址指针做数据操作，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，将对间接寻址指针 MP0、MP1L/MP1H 或 MP2L/MP2H 所指定的存储器地址产生对应的读 / 写操作。它们总是成对出现，IAR0 和 MP0 可以访问 Sector 0，而 IAR1 和 MP1L/MP1H、IAR2 和 MP2L/MP2H 可以访问任何 Sector。因为这些间接寻址寄存器不是实际存在的，直接读取将返回“00H”的结果，而直接写入此寄存器则不做任何操作。

间接寻址指针 – MP0，MP1L，MP1H，MP2L，MP2H

该系列单片机提供五个间接寻址指针，即 MP0、MP1L、MP1H、MP2L 和 MP2H。由于这些指针在数据存储器中能像普通的寄存器一般被操作，因此提供了一个寻址和数据追踪的有效方法。当对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间接寻址指针所指定的地址。MP0、IAR0 用于访问 Sector 0，而 MP1L/MP1H 和 IAR1、MP2L/MP2H 和 IAR2 可根据 MP1H 或 MP2H 寄存器访问所有的 Sector。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。

间接寻址程序举例 1

```
data .section ´data´
adres1      db ?
adres2      db ?
adres3      db ?
adres4      db ?
block       db ?
code .section at 0 ´code´
org00h
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:
```

间接寻址程序举例 2

```
data .section ´data´
adres1      db ?
adres2      db ?
adres3      db ?
adres4      db ?
block       db ?
code .section at 0 ´code´
org 00h
start:
    mov a, 04h              ; setup size of block
    mov block, a
    mov a, 01h              ; setup the memory sector
    mov mp1h, a
    mov a, offset adres1    ; Accumulator loaded with first RAM address
    mov mp1l, a             ; setup memory pointer with first RAM address
loop:
    clr IAR1                ; clear the data at address defined by MP1L
    inc mp1l                ; increment memory pointer MP1L
    sdz block               ; check if last memory location has been cleared
    jmp loop
continue:
```

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。

使用扩展指令直接寻址程序举例

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

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

累加器 – ACC

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

程序计数器低字节寄存器 – PCL

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

表格寄存器 – TBLP，TBHP，TBLH

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

状态寄存器 – STATUS

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

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

SC、CZ、Z、OV、AC 和 C 标志位通常反映最近运算的状态。

- SC：当 OV 与当前指令操作结果 MSB 执行“XOR”所得结果。
- CZ：不同指令不同标志位的操作结果。详细资料请参考寄存器定义部分。
- 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，或高半字节减法运算的结果没有产生借位时，AC 被置位，否则 AC 被清零。
- Z： 当算术或逻辑运算结果是零时，Z 被置位，否则 Z 被清零。
- OV：当运算结果高两位的进位状态异或结果为 1 时，OV 被置位，否则 OV 被清零。
- PDF：系统上电或执行“CLR WDT”指令会清零 PDF，而执行“HALT”指令则会置位 PDF。
- TO：系统上电或执行“CLR WDT”或“HALT”指令会清零 TO，而当 WDT 溢出则会置位 TO。

另外，当进入一个中断程序或执行子程序调用时，状态寄存器不会自动压入到堆栈保存。假如状态寄存器的内容是重要的且子程序可能改变状态寄存器的话，则需谨慎的去做正确的储存。

STATUS 寄存器

Bit	7	6	5	4	3	2	1	0
Name	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	×	×	0	0	×	×	×	×

"×"：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 数据寄存器

HT67F489 内建 EEPROM 数据存储器。“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 数据寄存器结构

EEPROM 数据寄存器容量为 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 中的一个地址寄存器和一个数据寄存器以及 Sector 1 中的一个控制寄存器，可以实现对 EEPROM 的单字节读写操作。

EEPROM 寄存器

有三个寄存器控制内部 EEPROM 数据存储器总的操作。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 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 必须先设为“40H”，MP1H 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”

Bit 5~0　数据 EEPROM 地址
数据 EEPROM 地址 Bit 5~Bit 0

EED 寄存器

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

Bit 7~0　EEPROM 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，EEPROM 中读取数据的地址要先放入 EEA 寄存器中。EEC 寄存器中的读使能位 RDEN 先置为高以使能读功能。若 EEC 寄存器中的 RD 位被置高，一个读周期将开始。若 RD 位已置为高而 RDEN 位还未被设置则不能开始读操作。若读周期结束，RD 位将自动清除为“0”，数据可以从 EED 寄存器中读取。数据在其它读或写操作执行前将一直保留在 EED 寄存器中。应用程序将轮询 RD 位以确定数据可以有效地被读取。

写数据到 EEPROM

写数据至 EEPROM， EEPROM 中写入数据的地址要先放入 EEA 寄存器中，写入的数据需存入 EED 寄存器中。EEC 寄存器中的写使能位 WREN 先置为高以使能写功能，若 EEC 寄存器中 WR 位被置为高，一个内部写周期将开始。这两个指令必须连续进行。在进行写操作前，要先将总的中断控制位 EMI 清零，当写周期开始后再将其置为 1。需注意的是若 WR 位已置为高而 WREN 位还未被设置则不能开始写操作。由于控制 EEPROM 写周期是一个内部时钟，与单片机的系统时钟异步，所以数据写入 EEPROM 的时间将有所延迟。可通过轮询 EEC 寄存器中的 WR 位或判断 EEPROM 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。上电后存储器指针对，MP1L/MP1H 和 MP2L/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。MP1H/MP2H 也可以正常清零以阻止进入 EEPROM 控制寄存器存在的 Sector 1。 尽管不是必须的，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。

写数据时，WREN 位置为“1”后，WR 须立即设置为高，以确保正确地执行写周期。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。

程序举例

从 EEPROM 中读取数据 — 轮询法

```
MOV A, EEPROM_ADRES        ; user defined address
MOV EEA, A
MOV A, 040H                ; setup memory pointer MP1L
MOV MP1L, A                ; MP1 points to EEC register
MOV A, 01H                 ; setup Section Pointer MP1H
MOV MP1H, A
SET IAR1.1                 ; set RDEN bit, enable read operations
SET IAR1.0                 ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                  ; check for read cycle end
JMP BACK
CLR IAR1                   ; disable EEPROM read/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 MP1L
MOV MP1L, A                ; MP1 points to EEC register
MOV A, 01H                 ; setup Section Pointer MP1H
```

```
MOV MP1H, A
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit- executed
                          ; immediately after set WREN bit
SET EMI
BACK:
SZ IAR1.2                 ; check for write cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read/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较佳的优化。振荡器选择是通过寄存器共同完成的。

振荡器概述

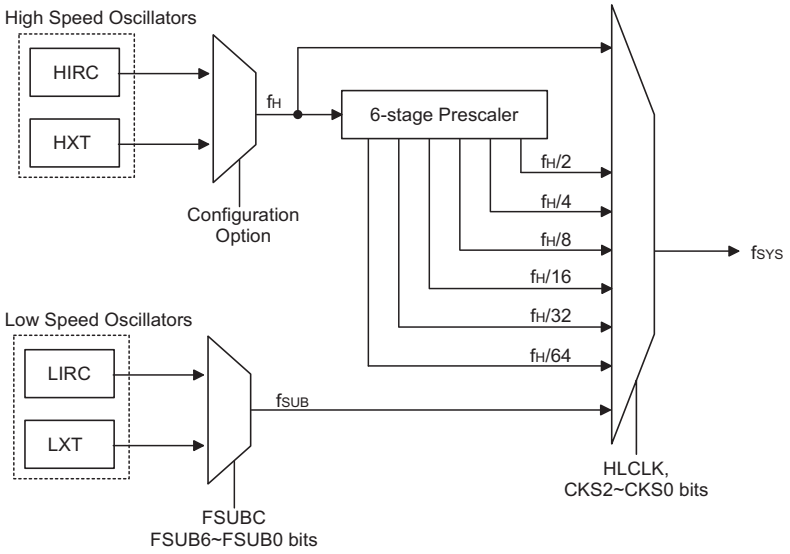
振荡器除了作为系统时钟源，还作为看门狗定时器和时基功能的时钟源。外部振荡器需要一些外围器件，而集成的两个内部振荡器不需要任何外围器件。它们提供高速和低速系统振荡器具有较宽的频率范围。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

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

振荡器类型

系统时钟配置

该系列单片机有四个系统振荡器，包括两个高速振荡器和两个低速振荡器。高速振荡器为内部 8MHz RC 振荡器 HIRC 和外部晶体 / 陶瓷振荡器 HXT。两个低速振荡器包括外部 32.768kHz 振荡器和内部 32kHz 振荡器。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。低速或高速系统时钟频率由 SMOD 寄存器的 HLCLK 位及 CKS2~CKS0 位决定的。请注意，两个振荡器必须做出选择，即一个高速和一个低速振荡器。

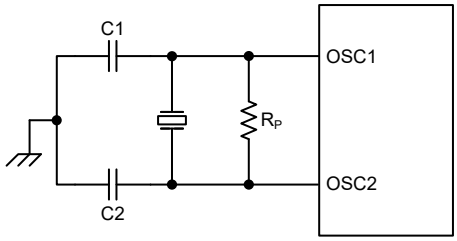


系统时钟配置选项

外部晶体 / 陶瓷振荡器 – HXT

外部高频晶体 / 陶瓷振荡器可通过配置选项选择。对于晶体振荡器，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体 / 陶瓷晶振有关。根据外部晶振频率的高 (≥1MHz) 或低 (<1MHz)，须在配置选项中对 HXT 模式进行相对应设置。

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他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

晶体振荡器 C1 和 C2 值		
晶体频率	C1	C2
12MHz	0pF	0pF
8MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	100pF	100pF
455kHz (注 2)	100pF	100pF
注：1. C1 和 C2 数值仅作参考用 2. XTAL 模式配置选项：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 振荡器具有一固定频率：8MHz。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因 V_{DD}、温度以及芯片制成工艺不同的影响较大程度地降低。如果选择了该内部时钟，无需额外的引脚。

外部 32.768kHz 晶体振荡器 – LXT

外部 32.768kHz 晶体振荡器是一个低频振荡器，经由 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 32768Hz 晶振以帮助起振。对于那些要求精确频率的场合中，可能需要这些元件来对由制程产生的误差提供频率补偿。在系统上电期间，LXT 振荡器启动需要一定的延时。

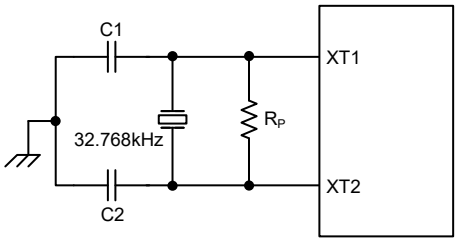
当系统进入空闲 / 休眠模式，系统时钟关闭以降低功耗。然而在某些应用，比如空闲 / 休眠模式下要保持内部定时器功能，必须提供额外的时钟，且与系统时钟无关。

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

由 FSUBC 寄存器决定是否 XT1/XT2 脚是用于 LXT 还是作为普通 I/O 口使用。

- 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他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.

外部晶体振荡器

LXT 振荡器 C1 和 C2 值		
晶振频率	C1	C2
32.768kHz	10pF	10pF
注：1、C1 和 C2 数值仅作参考用 2、R_{P2} 的建议值为 5M~10MΩ		

32.768kHz 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 **FSUB[6:0]**：f_{SUB} 系统时钟选择位
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 系统振荡器也是一个低频振荡器。这种单片机有一个完全集成 RC 振荡器，它在 5V 电压下运行的典型频率值为 32kHz 且无需外部元件。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响较大程度地降低。因此，内部 32kHz 振荡器频率在 25°C 温度 5V 电压下的精度保持在 10% 以内。

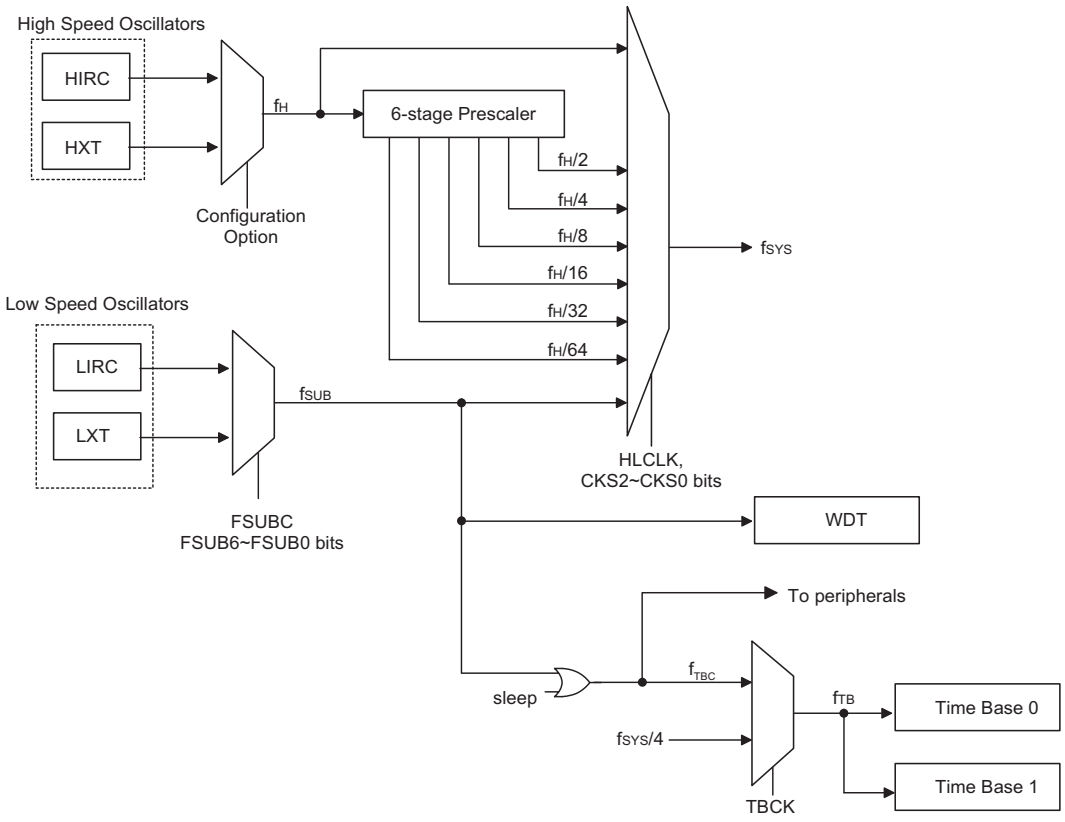
工作模式和系统时钟

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

系统时钟

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

主系统时钟可来自高频时钟源 f_H 或低频时钟源 f_{SUB}，通过 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位进行选择。高频时钟来自 HIRC/HXT 振荡器，低频系统时钟源来自内部时钟 f_{SUB}，若 f_{SUB} 被选择，可通过 FSUBC 寄存器的 FSUB6~FSUB0 位设定为 LXT 或 LIRC 振荡器。其它系统时钟还有高速系统振荡器的分频 $f_H/2$~$f_H/64$。



系统时钟选项

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

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 4 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	说明			
	CPU	f_{SYS}	f_{SUB}	f_{TBC}
正常模式	On	f_H~f_H/64	On	On
低速模式	On	f_{SUB}	On	On
空闲模式 0	Off	Off	On	On
空闲模式 1	Off	On	On	On
休眠模式 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 HIRC/HXT 振荡器。高速振荡器频率可被分为 1~64 的不等比率，实际的比率由 SMOD 寄存器中的 CKS2~CKS0 位及 HLCLK 位选择的。单片机使用高速振荡器分频作为系统时钟可减少工作电流。

低速模式

此模式的系统时钟虽为较低速时钟源，但单片机仍能正常工作。该低速时钟源可来自 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 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，低频时钟 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。低频时钟 f_{SUB} 开启。

注：若 LVDEN=1，进入休眠或空闲模式时，LVD 和 bandgap 不会关掉，并且 f_{SUB} 也会跟着强迫使能。休眠模式 1 时，其它外围设备将除能，而 WDT 或 LVD 可使能。

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，该标志位为低。若系统时钟来自 LXT 振荡器，系统唤醒后该位转换为高需要 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/HXT 振荡器，则只需 15~16 个时钟周期即可。

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

Bit 0 **HLCLK**：系统时钟选择位
0：$f_H/2$~$f_H/64$ or f_{SUB}
1：f_H
此位用于选择 f_H 或 $f_H/2$~$f_H/64$ 还是 f_{SUB} 作为系统时钟。该位为高时选择 f_H 作为系统时钟，为低时则选择 $f_H/2$~$f_H/64$ 或 f_{SUB} 作为系统时钟。当系统时钟由 f_H 时钟向 f_{SUB} 时钟转换时，f_H 将自动关闭以降低功耗。

CTRL 寄存器

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

“×”：未知

Bit 7 **FSYSON**：IDLE 模式时，f_{SYS} 控制位
0：除能
1：使能

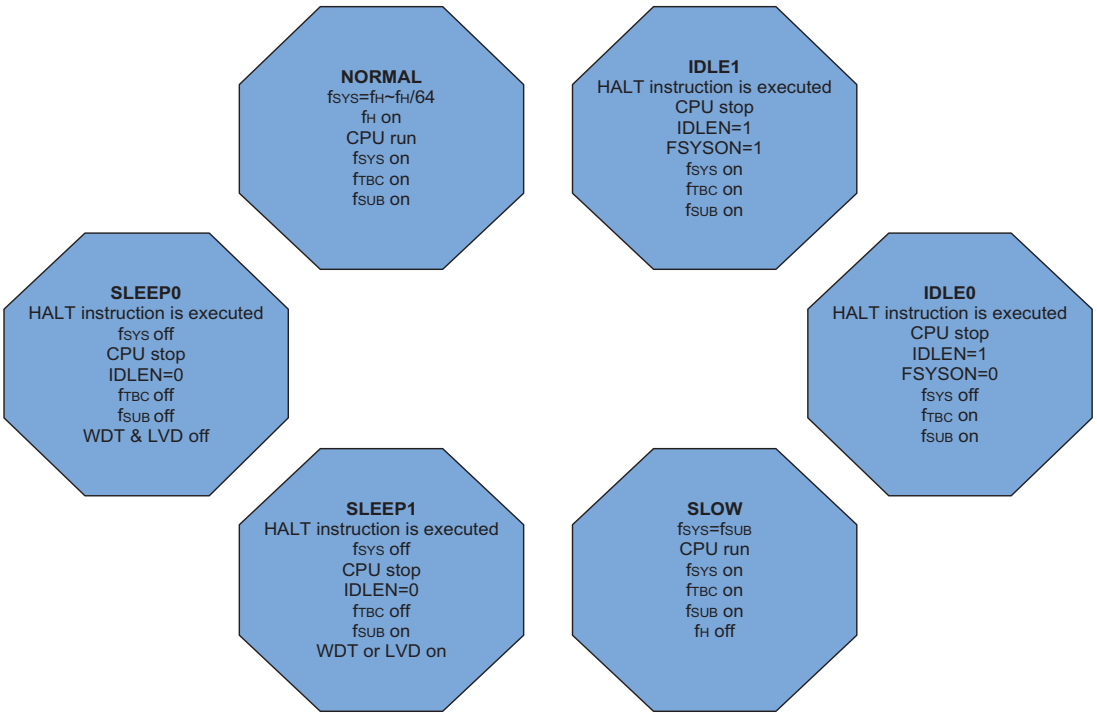
Bit 6~4 未定义，读为“0”

Bit 3 **FSUBF**：FSUBC 寄存器软件复位标志
0：未发生
1：发生
该位可通过程序清零，但不能由程序置 1。

Bit 2 **LVRF**：LVR 复位标志
0：未发生
1：发生
当低电压复位情况发生时此位设置为“1”。此位只能通过程序清零。

Bit 1 **LRF**：LVR 控制寄存器软件复位标志
0：未发生
1：发生
当 LVRC 寄存器包含任何未定义的 LVR 电压值时此位设置为“1”，用作软件复位功能。此位只能通过程序清零。

Bit 0 **WRF**：WDT 控制寄存器软件复位标志
0：未发生
1：发生
当 WDT 控制寄存器软件复位时此位设置为“1”。此位只能通过程序清零。



工作模式切换

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

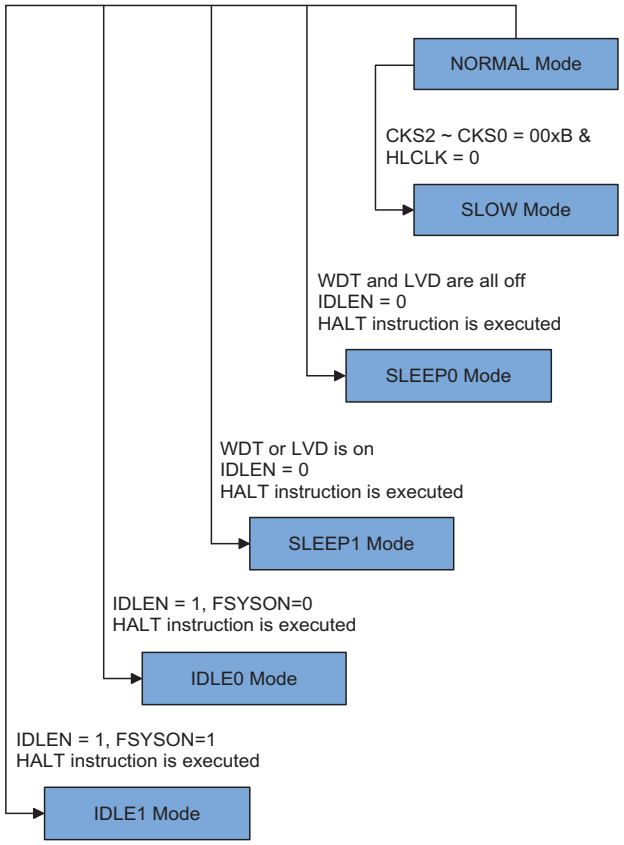
简单来说，正常模式和低速模式间的切换仅需设置 SMOD 中的 HLCLK 位及 CKS2~CKS0 位即可实现，而正常模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SMOD 寄存器中的 IDLEN 位和 CTRL 寄存器中的 FSYSON 位决定的。

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 $f_H/2$~$f_H/64$ 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

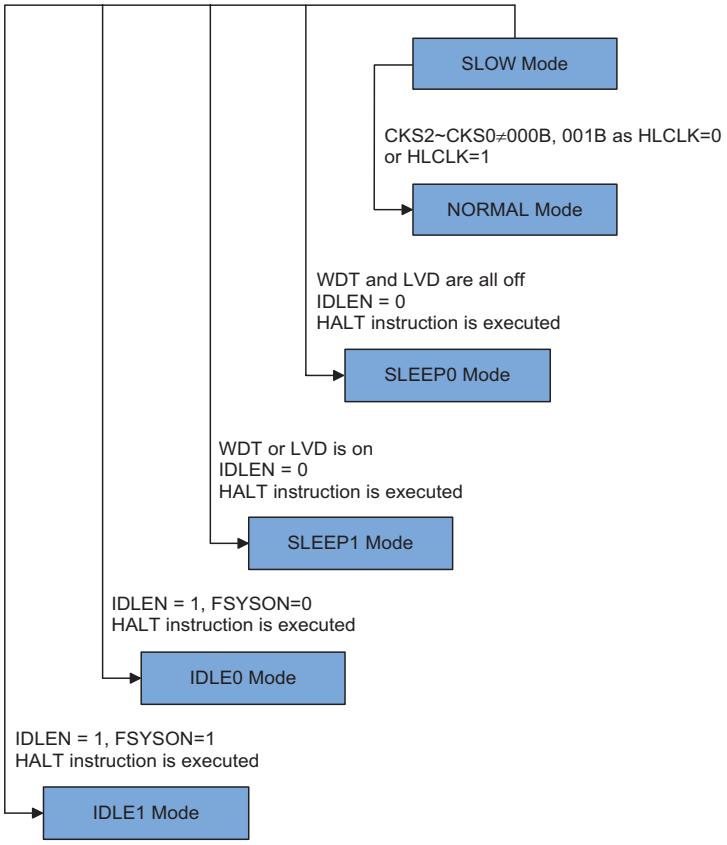
系统运行在正常模式时使用高速系统振荡器，因此较为耗电。可通过设置 SMOD 寄存器中的 HLCLK 位为“0”及 CKS2~CKS0 位为“000”或“001”使系统时钟切换至运行在低速模式下。此时将使用低速系统振荡器以节省耗电。用户可在对性能要求不高的操作中使用此方法以减少耗电。

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



低速模式切换到正常模式

在低速模式系统使用 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。



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 将被清零并停止运行。
- 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。
- 若 WDT 使能，则 WDT 将被清零并重新开始计数。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。

进入空闲模式 0

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

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

进入空闲模式 1

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

- 系统时钟、时基时钟 f_{TBC} 和低频时钟 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。
- 若 WDT 使能，WDT 将被清零并重新开始计数。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。

待机电流的注意事项

由于单片机进入休眠或空闲模式的主要原因是将 MCU 的电流降低到尽可能低，可能到只有几个微安的级别（空闲模式 1 除外），所以如果要将电路的电流进一步降低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入 / 输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。

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

若由 WDT 溢出唤醒，则会发生看门狗定时器复位。可以通过状态寄存器中 TO 和 PDF 位来判断它的唤醒源。系统上电或执行清除看门狗的指令，会清零 PDF；执行 HALT 指令，PDF 将被置位。看门狗计数器溢出将会置位 TO 标志并唤醒系统，这种复位会重置程序计数器和堆栈指针，其它标志保持原有状态。

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

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源来自于内部时钟 f_{SUB}，而 f_{SUB} 的时钟源由 LXT 或 LIRC 振荡器提供，可通过 FSUBC 寄存器选择。看门狗定时器的时钟源可分频为 2^8~2^{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 的周期大约为 32kHz。需要注意的是，这个特殊的内部时钟周期随 V_{DD}、温度和制成的不同而变化。LXT 振荡器由一个外部 32.768kHz 晶振提供。

看门狗定时器控制寄存器

WDTC 寄存器用于控制 WDT 功能的使能 / 除能及选择溢出周期。

WDTC 寄存器

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

Bit 7~3 **WE4~WE0**：WDT 软件控制
10101：除能
01010：使能
其它值：MCU 复位
如果单片机复位且由干扰引起，需要一个延迟时间 t_{SRESET} 响应复位。且在复位后 CTRL 寄存器中的 WRF 标志位会被置位。

Bit 2~0 **WS2~WS0**：WDT 溢出周期选择位
000：$2^8/f_{SUB}$
001：$2^{10}/f_{SUB}$

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

这三位控制 WDT 时钟源的分频比，从而实现对 WDT 溢出周期的控制。

CTRL 寄存器

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

“×”：未知

Bit 7 **FSYSON**：IDLE 模式时，f_{SYS} 控制位
详见别处的描述。

Bit 6~4 未定义，读为“0”

Bit 3 **FSUBF**：FSUBC 寄存器软件复位标志
详见别处的描述。

Bit 2 **LVRF**：LVR 复位标志
详见别处的描述。

Bit 1 **LRF**：LVR 控制寄存器软件复位标志
详见别处的描述。

Bit 0 **WRF**：WDT 控制寄存器软件复位标志
0：未发生
1：发生
当 WDT 控制寄存器软件复位时此位设置为“1”。此位只能通过程序清零。

看门狗定时器操作

当 WDT 溢出时，看门狗定时器产生一个芯片复位的动作。这也就意味着正常工作期间，用户需在应用程序中看门狗溢出前有策略地清看门狗定时器以防止其产生复位，可使用清除看门狗指令实现。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 位可提供使能 / 除能控制以及控制看门狗定时器复位操作。如果 WE4~WE0 设置为“10101B”，则 WDT 除能；若设置为“01010B”，则 WDT 使能；如果 WE4~WE0 设置为除“10101B”和“01010B”以外的其它任意值，则经过一个延迟时间 t_{SRESET} 后单片机复位。上电后这些位初始化为“01010B”。

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

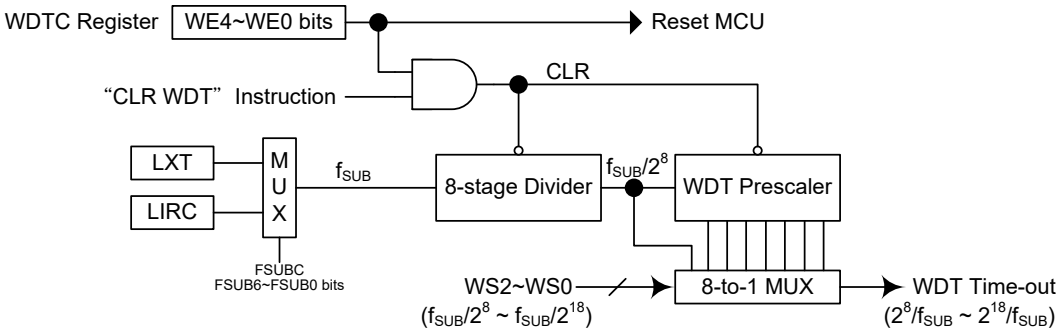
看门狗定时器使能 / 除能控制

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 将被置位。有三种方法可以用来清除 WDT 的内容。第一种是 WDT 复位，即将 WE4~WE0 位设置成除了“01010B”和“10101B”外的任

意值；第二种是通过看门狗定时器软件清除指令，而第三种是通过“HALT”指令。

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

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



看门狗定时器

复位和初始化

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

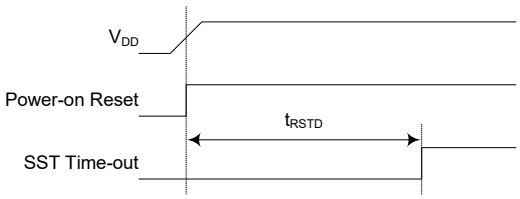
另一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。

复位功能

包括内部事件触发复位，单片机共有四种复位方式：

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。

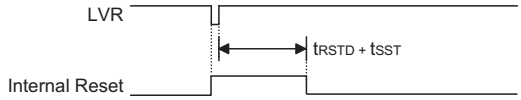


注：t_{RSTD} 为上电延迟时间，典型值为 50ms

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~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~V_{LVR} 的低电压状态的时间，必须超过 LVD & LVR 电气特性中 t_{LVR} 参数的值。如果低电压存在不超过 t_{LVR} 参数的值，则 LVR 将会忽略它且不会执行复位功能。实际的 V_{LVR} 参数值可通过 LVRC 寄存器中的 LVS 位进行选择。若由于受到干扰 LVS7~LVS0 变为其它值时，需经过一个延迟时间 t_{SRESET} 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。正常执行时 LVR 会于休眠或空闲时自动除能关闭。



注：t_{RSTD} 为上电延迟时间，典型值为 50ms

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

Bit 7~0 **LVS7~LVS0**：LVR 电压选择
01010101：2.1V
00110011：2.55V
10011001：3.15V
10101010：3.8V
其它值：MCU 复位 (寄存器设置为 POR 值)
若低电压情况发生且满足以上定义的任何一个低电压复位值，则单片机复位。当低电压状况保持时间大于 t_{LVR} 后响应复位。此时复位后寄存器内容保持不变。除了以上定义的四个低电压复位值外，其它值也能导致单片机复位。需要经过一个延迟时间 t_{SRESET} 响应复位。但此时复位后寄存器内容恢复到 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	×	0	0

“×”：未知

Bit 7 **FSYSON**：IDLE 模式时，f_{SYS} 控制位
详见别处的描述。

Bit 6~4 未定义，读为“0”

Bit 3 **FSUBF**：FSUBC 寄存器软件复位标志
详见别处的描述。

Bit 2 **LVRF**：LVR 复位标志
0：未发生
1：发生

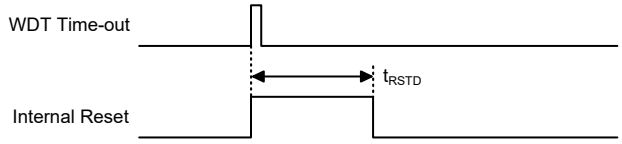
当低电压复位情况发生时此位设置为“1”。此位只能通过程序清零。

Bit 1 **LRF**：LVR 控制寄存器软件复位标志
0：未发生
1：发生
当 LVRC 寄存器包含任何未定义的 LVR 电压值时此位设置为“1”，相当于软件复位功能。此位只能通过程序清零。

Bit 0 **WRF**：WDT 控制寄存器软件复位标志
详见别处的描述。

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。

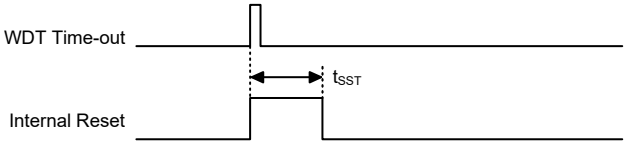


注：t_{RSTD} 为上电延迟时间，典型值为 16.7ms

正常运行时看门狗溢出复位时序图

休眠或空闲时看门狗溢出复位

休眠或空闲时看门狗溢出复位和其它种类的复位有些不同。除了程序计数器与堆栈指针将被清“0”及 TO 位被设为“1”外，绝大部分的条件保持不变。图中 t_{SST} 的详细说明请参考交流电气特性。



注：如果系统时钟源为 HIRC/HXT，则 t_{SST} 为 15~16 个时钟周期。
如果系统时钟源为 LXT，则 t_{SST} 为 1024 个时钟周期。
如果系统时钟源为 LIRC，则 t_{SST} 为 1~2 个时钟周期。

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形，列于下表。

项目	复位后情况
程序计数器	清除为零
中断	所有中断被除能
看门狗定时器	WDT 清除并重新计数
定时 / 计数器	所有定时 / 计数器停止
输入 / 输出口	I/O 口设为输入模式，AN0~AN9 设为 A/D 输入
堆栈指针	堆栈指针指向堆栈顶端

不同的复位形式对单片机内部寄存器的影响是不同的。为保证复位后程序能正常执行，了解寄存器在特定条件复位后的设置是非常重要的。下表即为不同方式复位后内部寄存器的状况。

寄存器	上电复位	LVR 复位	WDT 溢出 (正常模式)	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
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
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	---x xxxx	---u uuuu	---u uuuu	---u uuuu
STATUS	xx00 xxxx	uuuu uuuu	uu1u uuuu	uu11 uuuu
SMOD	000- 0011	000- 0011	000- 0011	uuu- uuuu
LVDC	--00 -000	--00 -000	--00 -000	--uu -uuu
LVRC	0101 0101	0101 0101	0101 0101	uuuu uuuu
CTRL	0--- 0x00	0--- uuuu	0--- uuuu	0--- uuuu
INTEG	0000 0000	0000 0000	0000 0000	uuuu uuuu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
TBC	0011 -111	0011 -111	0011 -111	uuuu -uuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	-000 -000	-000 -000	-000 -000	-uuu -uuu
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

寄存器	上电复位	LVR 复位	WDT 溢出 (正常模式)	WDT 溢出 (空闲 / 休眠模式)
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
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 ----
TMPC	---0 0000	---0 0000	---0 0000	---u uuuu
IOHR0	0000 0000	0000 0000	0000 0000	uuuu uuuu
IOHR1	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADRL (ADRFS=0)	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRL (ADRFS=1)	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH (ADRFS=0)	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH (ADRFS=1)	---- xxxx	---- xxxx	---- xxxx	---- uuuu
ADCR0	0110 0000	0110 0000	0110 0000	uuuu uuuu
ADCR1	00-0 -000	00-0 -000	00-0 -000	uu-u -uuu
ACERL	1111 1111	1111 1111	1111 1111	uuuu uuuu
ACERH	---- --11	---- --11	---- --11	---- --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

寄存器	上电复位	LVR 复位	WDT 溢出 (正常模式)	WDT 溢出 (空闲 / 休眠模式)
TM1AH	---- --00	---- --00	---- --00	---- --uu
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
FSUBC	0010 1010	0010 1010	0010 1010	uuuu uuuu
LCDC0	0000 -000	0000 -000	0000 -000	uuuu -uuu
LCDC1	000- 0000	000- 0000	000- 0000	uuu- uuuu
SEGCR0	0000 0000	0000 0000	0000 0000	uuuu uuuu
SEGCR1	0000 0000	0000 0000	0000 0000	uuuu uuuu
SEGCR2	---- 0000	---- 0000	---- 0000	---- uuuu
EEA	--00 0000	--00 0000	--00 0000	--uu uuuu
EED	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEC	---- 0000	---- 0000	---- 0000	---- 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

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 / 输出口控制具有很大的灵活性。大部分引脚可在用户程序控制下被设定为输入或输出。所有引脚的上拉电阻设置以及指定引脚的唤醒设置也都由软件控制，这些特性也使得此类单片机在广泛应用上都能符合开发的需求。

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PBPU	—	—	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PB	—	—	PB5	PB4	PB3	PB2	PB1	PB0
PBC	—	—	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PEPU	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PE	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEC	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	—	—	—	—
PF	PF7	PF6	PF5	PF4	—	—	—	—
PFC	PFC7	PFC6	PFC5	PFC4	—	—	—	—

“—”：未定义，读为“0”

PAWUn：PA 唤醒功能控制
0：除能
1：使能

PAn/PBn/PCn/PDn/PEn/PFn：I/O 口数据位
0：数据 0
1：数据 1

PACn/PBCn/PCCn/PDCn/PECn/PFCn：I/O 口类型选择
0：输出
1：输入

PAPUn/PBPUn/PCPUn/PDPUn/PEPUn/PFPUn：上拉功能控制
0：除能
1：使能

上拉电阻

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

PA 口唤醒

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

PAWU 寄存器

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

Bit 7~0　PA 口 bit 7~bit 0 唤醒功能控制位
0：除能
1：使能

引脚共用功能

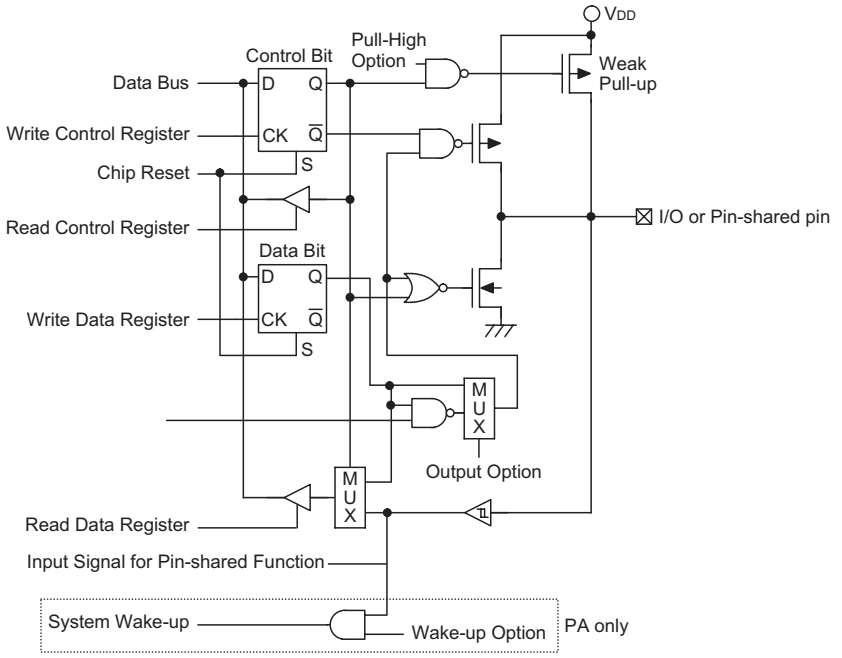
引脚的多功能可以增加单片机应用的灵活性。有限的引脚个数将会限制设计者，而引脚的多功能将会解决很多此类问题。此外，这些引脚功能可以通过一系列寄存器进行设定。

输入 / 输出端口控制寄存器

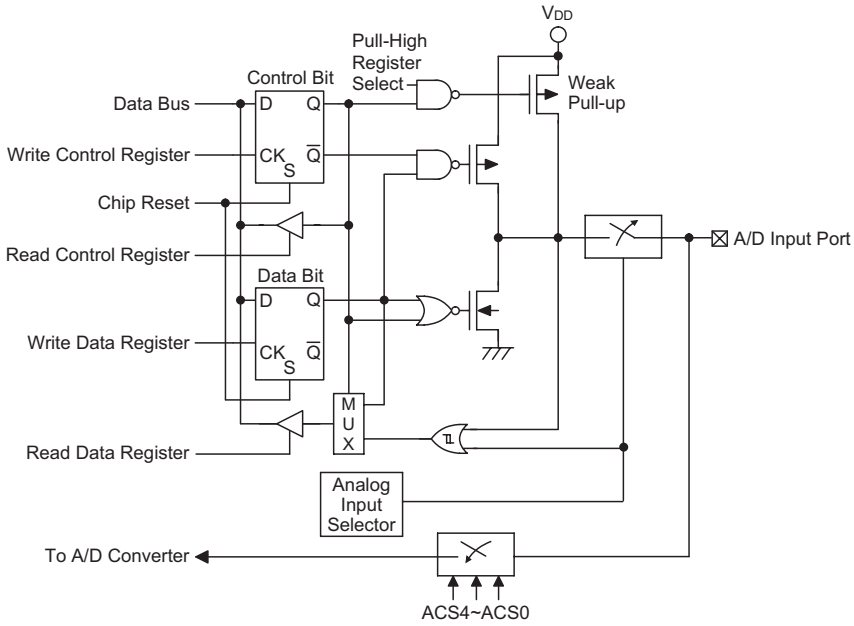
每一个输入 / 输出口都具有各自的控制寄存器，即 PAC~PFC，用来控制输入 / 输出状态。从而每个 I/O 引脚都可以通过软件控制，动态的设置为 CMOS 输出或输入。所有的 I/O 端口的引脚都各自对应于 I/O 端口控制的某一位。若 I/O 引脚要实现输入功能，则对应的控制寄存器的位需要设置为“1”。这时程序指令可以直接读取输入脚的逻辑状态。若控制寄存器相应的位被设定为“0”，则此引脚被设置为 CMOS 输出。当引脚设置为输出状态时，程序指令读取的是输出端口寄存器的内容。注意，如果对输出口做读取动作时，程序读取到的是内部输出数据锁存器中的状态，而不是输出引脚上实际的逻辑状态。

输入 / 输出引脚结构

下图为输入 / 输出引脚的内部结构图。输入 / 输出引脚的准确逻辑结构图可能与此图不同，这里只是为了方便对 I/O 引脚功能的理解提供的一个参考。图中的引脚共用结构并非针对所有单片机。



通用输入 / 输出端口



A/D 输入 / 输出结构

编程注意事项

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

PA 口的每个引脚都带唤醒功能。单片机处于休眠或空闲模式时，有很多方法可以唤醒单片机，其中之一就是通过 PA 任一引脚电平从高到低转换的方式，可以设置 PA 口一个或多个引脚具有唤醒功能。

定时器模块 – TM

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

这里只介绍各种 TM 的共性，更多详细资料请参考相关定时器章节。

简介

该系列单片机包含一个 10-bit 周期型 TM 和三个 10-bit 简易型 TM，分别命名为 TM0 和 TM1~TM3。虽然性质相似，但不同 TM 特性复杂度不同。本章介绍简易型和周期型 TM 的共性，更多详细资料分别见后面各章。两种类型 TM 的特性和区别见下表。

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

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

两种不同类型的 TM 提供从简单的定时操作到 PWM 信号产生等多种功能。理解 TM 操作的关键是比较 TM 内独立运行的计数器的值与内部比较器的预置值。当计数器的值与比较器的预置值相同时，则比较匹配，TM 中断信号产生，清

零计数器并改变 TM 输出引脚的状态。用户选择内部时钟或外部时钟来驱动内部 TM 计数器。

TM 时钟源

驱动 TM 计数器的时钟源很多。通过设置 TM 控制寄存器的 TnCK2~TnCK0 位，选择所需的时钟源。该时钟源来自系统时钟 f_{SYS} 或内部高速时钟 f_H 或 f_{TBC} 时钟源或外部 TCKn 引脚时钟的分频比。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 都有两个内部中断，分别是内部比较器 A 或比较器 P，当比较匹配发生时产生 TM 中断。当 TM 中断产生时，计数器清零并改变 TM 输出引脚的状态。

TM 外部引脚

无论哪种类型的 TM，都有一个 TM 输入引脚 TCKn。通过设置 TMnC0 寄存器中的 TnCK2~TnCK0 位，选择 TM 功能并将该引脚作为 TM 时钟源输入脚。外部时钟源可通过该引脚来驱动内部 TM。外部 TM 输入脚也与其它功能共用，但是，如果设置适当值给 TnCK2~TnCK0，该引脚会连接到内部 TM。TM 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。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 TPn 输出引脚也被 TM 用来产生 PWM 输出波形。当 TM 输出引脚与其它功能共用时，TM 输出功能需要通过寄存器先被设置。寄存器中的一个单独位用于决定其相关引脚用于外部 TM 输出还是用于其它功能。不同类型 TM 中输出引脚的个数是不同的，详见下表。

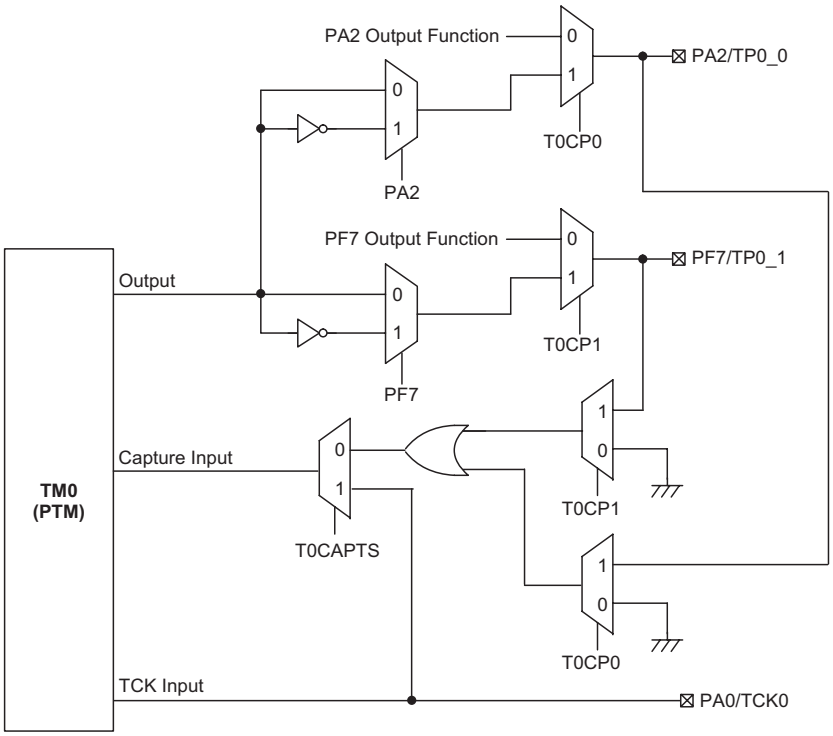
周期型 TM 引脚名称带有“_n”后缀。引脚名称带“_0”或“_1”后缀表示来自多输出引脚的 TM。这允许 TM 产生一对互补输出，可通过 I/O 寄存器数据位选择。

TM0	TM1	TM2	TM3
TP0_0, TP0_1	TP1	TP2	TP3

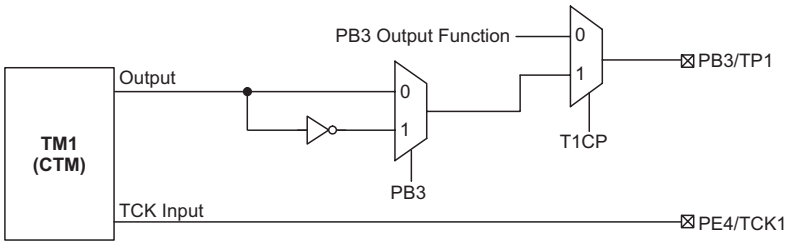
TM 输出引脚

TM 输入 / 输出引脚控制寄存器

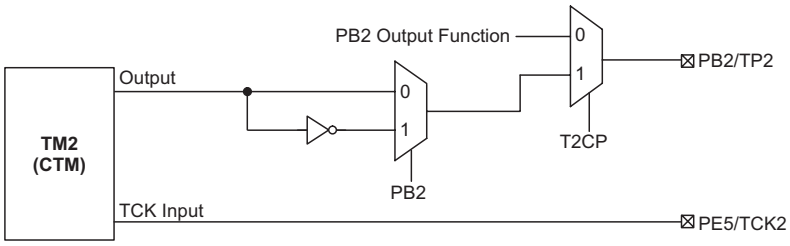
通过设置一个与 TM 输入 / 输出引脚相关的寄存器的一位，选择作为 TM 输入 / 输出功能或其它共用功能。设定为高时，相关引脚用作 TM 输入 / 输出，清零时将保持原来的功能。



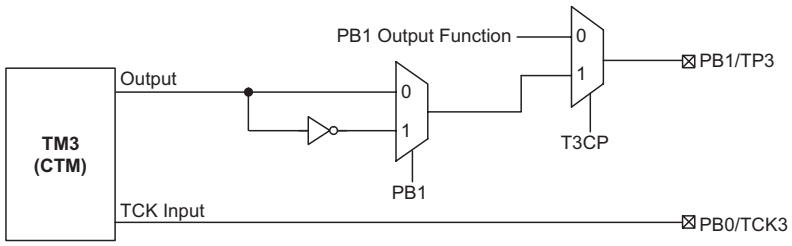
TM0 功能引脚控制方框图



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

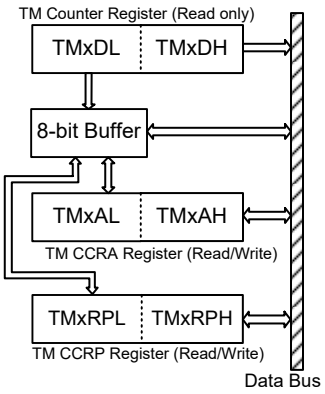
Bit 2　**T1CP**：TP1 引脚控制位
0：除能
1：使能

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 TM0 寄存器 CCRP 为 10-bit 的寄存器，含有低字节和高字节结构。高字节可直接访问，低字节则仅能通过一个内部 8-bit 的缓存器进行访问。值得注意的是 8-bit 缓存器的存取数据及相关低字节的读写操作仅在其相应的高字节读取操作执行时发生。CCRA 和 CCRP 寄存器访问方式如下图所示，读写这些成对的寄存器需通过特殊的方式。建议使用“MOV”指令按照以下步骤访问 CCRA 或 CCRP 低字节寄存器，TMxAL 或 TMxRPL，否则可能导致无法预期的结果。



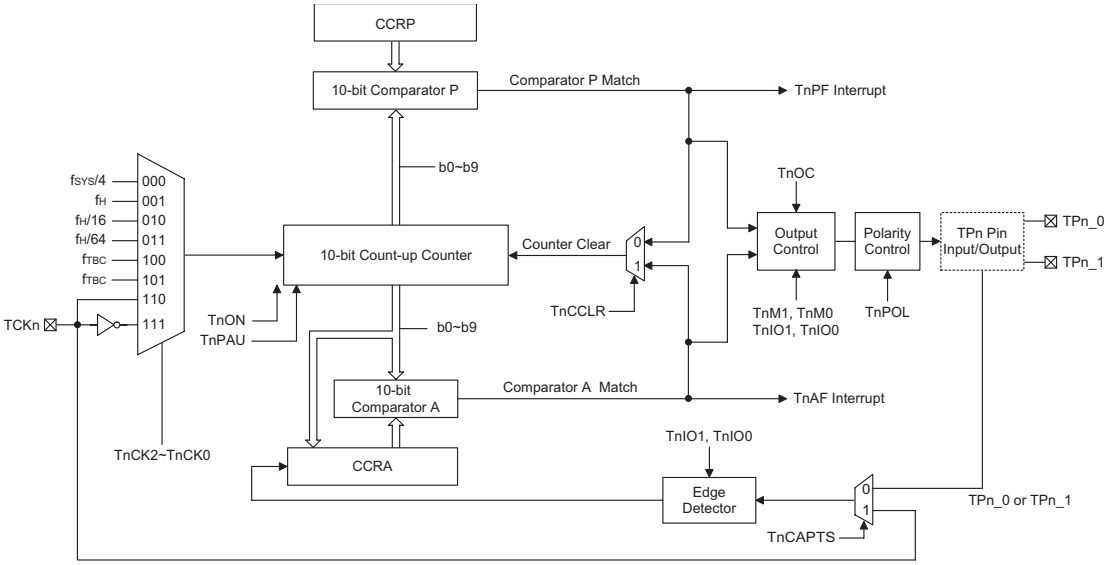
读写流程如下步骤所示：

- 写数据至 CCRA 或 CCRP
  - ♦ 步骤 1. 写数据至低字节寄存器 TMxAL 或 TMxRPL
    – 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 TMxAH 或 TMxRPH
    – 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 TMxDH、TMxAH 或 TMxRPH 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 TMxDL、TMxAL 或 TMxRPL 读取数据
    – 注意，此时读取 8-bit 缓存器中的数据。

周期型 TM – PTM

周期型 TM 包括 5 种工作模式，即比较匹配输出、定时 / 事件计数器、捕捉输入、单脉冲输出和 PWM 输出模式。周期型 TM 也由一个外部输入脚控制并驱动两个外部输出脚。

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



周期型 TM 框图 (n=0)

周期型 TM 操作

周期型 TM 的核心是一个由用户选择的内部或外部时钟源驱动的 10 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 10 位宽度。

通过应用程序改变 10 位计数器值的唯一方法是使 T0ON 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寄存器来实现的。

周期型 TM 寄存器介绍

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
TM0C0	T0PAU	T0CK2	T0CK1	T0CK0	T0ON	—	—	—
TM0C1	T0M1	T0M0	T0IO1	T0IO0	T0OC	T0POL	T0CAPTS	T0CCLR
TM0DL	D7	D6	D5	D4	D3	D2	D1	D0
TM0DH	—	—	—	—	—	—	D9	D8

寄存器名称	位							
	7	6	5	4	3	2	1	0
TM0AL	D7	D6	D5	D4	D3	D2	D1	D0
TM0AH	—	—	—	—	—	—	D9	D8
TM0RPL	D7	D6	D5	D4	D3	D2	D1	D0
TM0RPH	—	—	—	—	—	—	D9	D8

10-bit 周期型 TM 寄存器列表

TM0C0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	T0PAU	T0CK2	T0CK1	T0CK0	T0ON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **T0PAU**：TM0 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，并从此值开始继续计数。

Bit 6~4 **T0CK2~T0CK0**：选择 TM0 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：TCK0 上升沿时钟
111：TCK0 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 **T0ON**：TM0 计数器 On/Off 控制
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时 (通过 T0OC 位指定)，当 T0ON 位经由低到高的转换时，TM 输出脚将重置其初始值。

Bit 2~0 未定义，读为“0”

TM0C1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	T0IO1	T0IO0	T0OC	T0POL	T0CAPTS	T0CCLR
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 **T0M1~T0M0**：选择 TM0 工作模式
00：比较匹配输出模式
01：捕捉输入模式
10：PWM 模式或单脉冲输出模式
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 T0M1 和 T0M0 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 **T0IO1~T0IO0**：选择 TP0_0 和 TP0_1 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 模式 / 单脉冲输出模式
00：强制无效状态
01：强制有效状态
10：PWM 输出
11：单脉冲输出
捕捉输入模式
00：在 TM 捕捉输入引脚上升沿输入捕捉
01：在 TM 捕捉输入引脚下降沿输入捕捉
10：在 TM 捕捉输入引脚双沿输入捕捉
11：输入捕捉除能
定时 / 计数器模式
未使用
此两位用于决定在一定条件达到时 TM 输出脚如何改变状态。这两位值的选择决定 TM 运行在哪种模式下。
在比较匹配输出模式下，T0IO1 和 T0IO0 位决定当从比较器 A 比较匹配输出发生时 TM 输出脚如何改变状态。当从比较器 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 TM0C1 寄存器的 T0OC 位设置取得。注意，由 T0IO1 和 T0IO0 位得到的输出电平必须与通过 T0OC 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 T0ON 位由低到高电平的转换复位至初始值。
在 PWM 模式，T0IO1 和 T0IO0 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 TM1 关闭时改变 T0IO1 和 T0IO0 位的值是很有必要的。若在 TM 运行时改变 T0IO1 和 T0IO0 的值，PWM 输出的值是无法预料的。

Bit 3 **T0OC**：TP0_0 和 TP0_1 输出控制位
比较匹配输出模式
0：初始低
1：初始高
PWM 模式 / 单脉冲输出模式
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 模式 / 单脉冲输出模式。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 模式时，其决定 PWM 信号是高有效还是低有效。

Bit 2 **T0POL**：TP0_0 和 TP0_1 输出极性控制位
0：同相
1：反相
此位控制 TP0_0 和 TP0_1 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 **T0CAPTS**：TM0 捕捉输入触发源选择位
0：来自 TP0_0 或 TP0_1 引脚
1：来自 TCK0 引脚

Bit 0 **T0CCLR**：选择 TM0 计数器清零条件位
0：TM0 比较器 P 匹配
1：TM0 比较器 A 匹配
此位用于选择清除计数器的方法。周期型 TM 包括两个比较器 - 比较器 A 和比较器 P。这两个比较器每个都可以用作清除内部计数器。T0CCLR 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。T0CCLR 位在 PWM，单脉冲或输入捕捉模式时未使用。

TM0DL 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 **TM0DL**：TM0 计数器低字节寄存器 bit 7~bit 0
TM0 10-bit 计数器 bit 7~bit 0

TM0DH Register

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 **TM0DH**：TM0 计数器高字节寄存器 bit1~bit 0
TM0 10-bit 计数器 bit 9~bit 8

TM0AL 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 **TM0AL**：TM0 CCRA 低字节寄存器 bit 7~bit 0
TM0 10-bit CCRA bit 7~bit 0

TM0AH 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 **TM0AH**：TM0 CCRA 高字节寄存器 bit 1~bit 0
TM0 10-bit CCRA bit 9~bit 8

TM0RPL 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 **TM0RPL**：TM0 CCRP 低字节寄存器 bit 7~bit 0
TM0 10-bit CCRP bit 7~bit 0

TM0RPH 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 **TM0RPH**：TM0 CCRP 高字节寄存器 bit 1~bit 0
TM0 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 TM0C1 寄存器的 T0M1 和 T0M0 位选择任意模式。

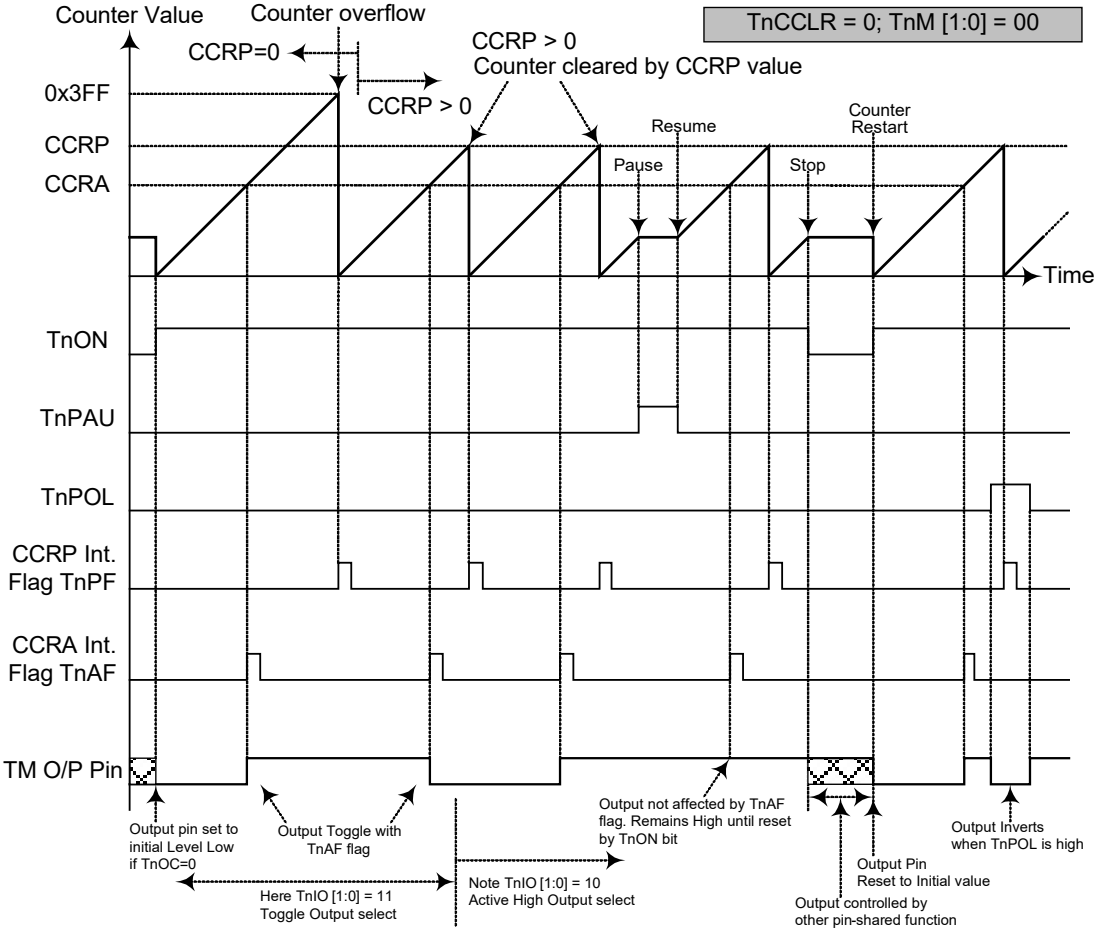
比较匹配输出模式

为使 TM 工作在此模式，TM0C1 寄存器中的 T0M1 和 T0M0 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 T0CCLR 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 T0AF 和 T0PF 将分别置位。

如果 TM0C1 寄存器的 T0CCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅产生 T0AF 中断请求标志。所以当 T0CCLR 为高时，不会产生 T0PF 中断请求标志。在比较匹配输出模式下，CCRA 不能设为“0”。

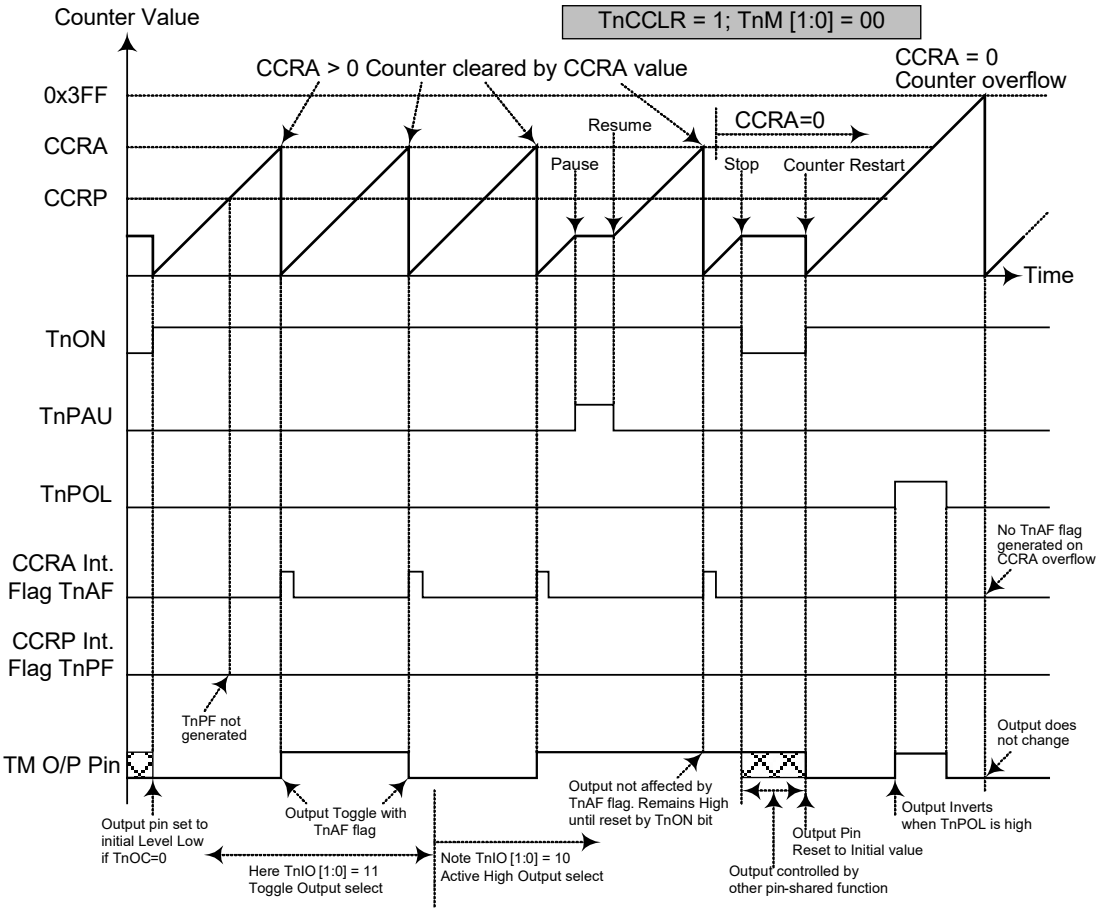
正如该模式名所言，当比较匹配发生后，TM 输出脚状态改变。当比较器 A 比较匹配发生后 T0AF 标志产生时，TM 输出脚状态改变。比较器 P 比较匹配发生时产生的 T0PF 标志不影响 TM 输出脚。TM 输出脚状态改变方式由 TM0C1

寄存器中 T0IO1 和 T0IO0 位决定。当比较器 A 比较匹配发生时，T0IO1 和 T0IO0 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，既可以通过 T0ON 位由低到高电平的变化设置，也可以由 T0OC 位设置。注意，若 T0IO1 和 T0IO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 -- TnCCLR=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. 在 TnON 上升沿 TM 输出脚复位至初始值
4. n=0



比较匹配输出模式 -- TnCCLR=1

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. 在 TnON 上升沿 TM 输出脚复位至初始值
4. 当 TnCCLR=1 时，不会产生 TnPF 标志
5. n=0

定时 / 计数器模式

为使TM工作在此模式，TM0C1寄存器中的T0M1和T0M0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，TM0C1 寄存器中的 T0M1 和 T0M0 位需要设置为“10”，且 T0IO1 和 T0IO0 位也需要设置为“10”。TM 的 PWM 功能在马达控制、加热控制、照明控制等方面十分有用。给 TM 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就较为灵活。在 PWM 模式中，T0CCLR 位不影响 PWM 周期。CCRA 和 CCRP 寄存器决定 PWM 波形，一个用来清除内部计数器并控制 PWM 波形的频率，另一个用来控制占空比。所以 PWM 波形由 CCRA 和 CCRP 寄存器共同决定。

当比较器 A 或比较器 P 比较匹配发生时，将产生 CCRA 或 CCRP 中断标志。TM0C1 寄存器中的 T0OC 位决定 PWM 波形的极性，T0IO1 和 T0IO0 位使能 PWM 输出或将 TM 输出脚置为逻辑高或逻辑低。T0POL 位对 PWM 输出波形的极性取反。

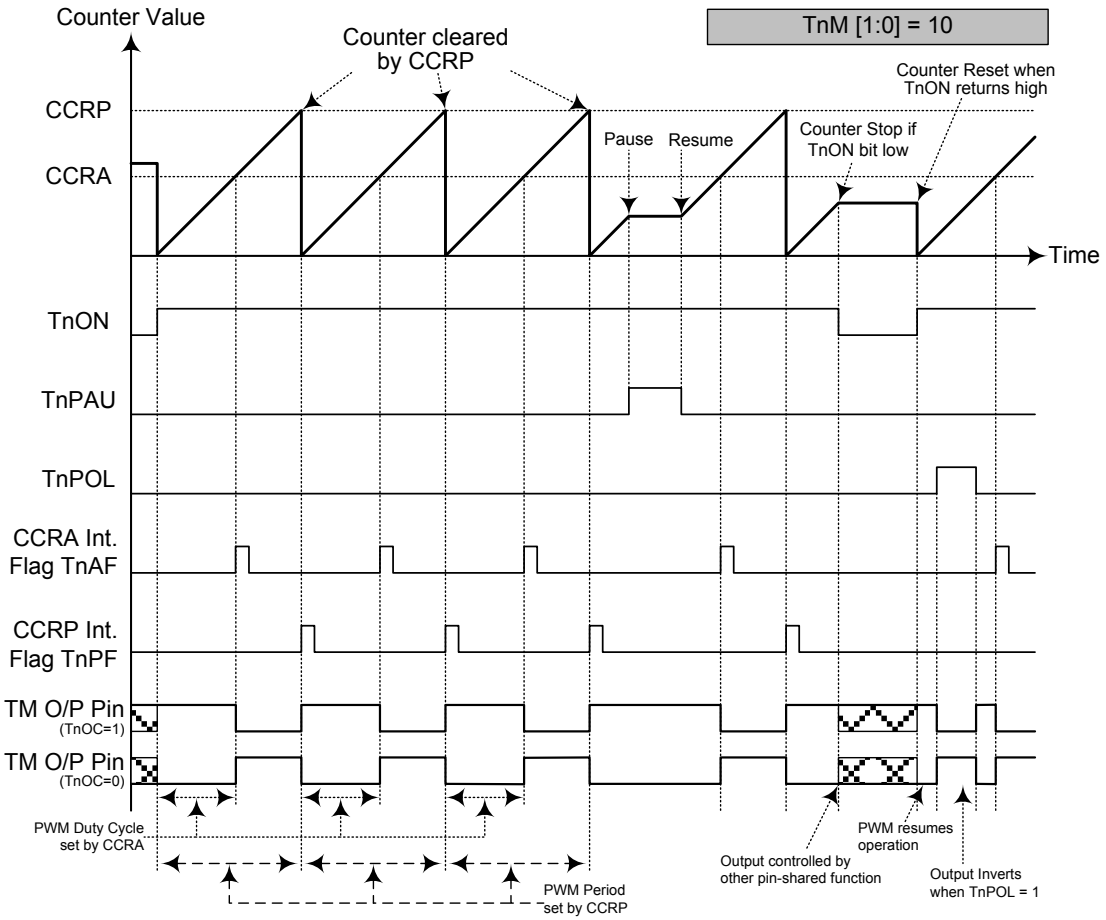
- **10-bit PTM, PWM 模式 , 边沿对齐模式**

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

若 f_{SYS}=8MHz，TM 时钟源选择 f_{SYS}/4，CCRP=512，CCRA=128，

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

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



PWM 输出模式

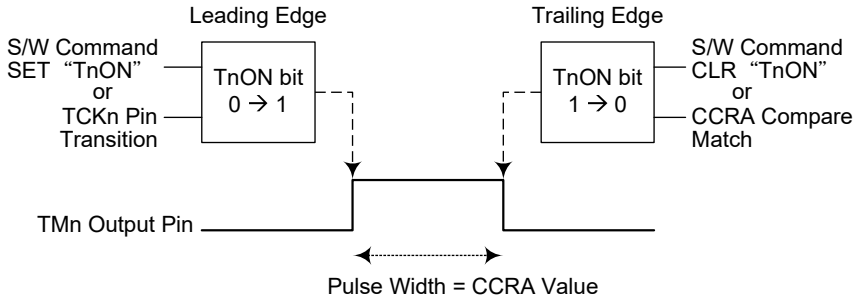
注：1. CCRP 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 TnIO1，TnIO0=00 或 01，PWM 功能不变
4. TnCCLR 位不影响 PWM 操作
5. n=0

单脉冲输出模式

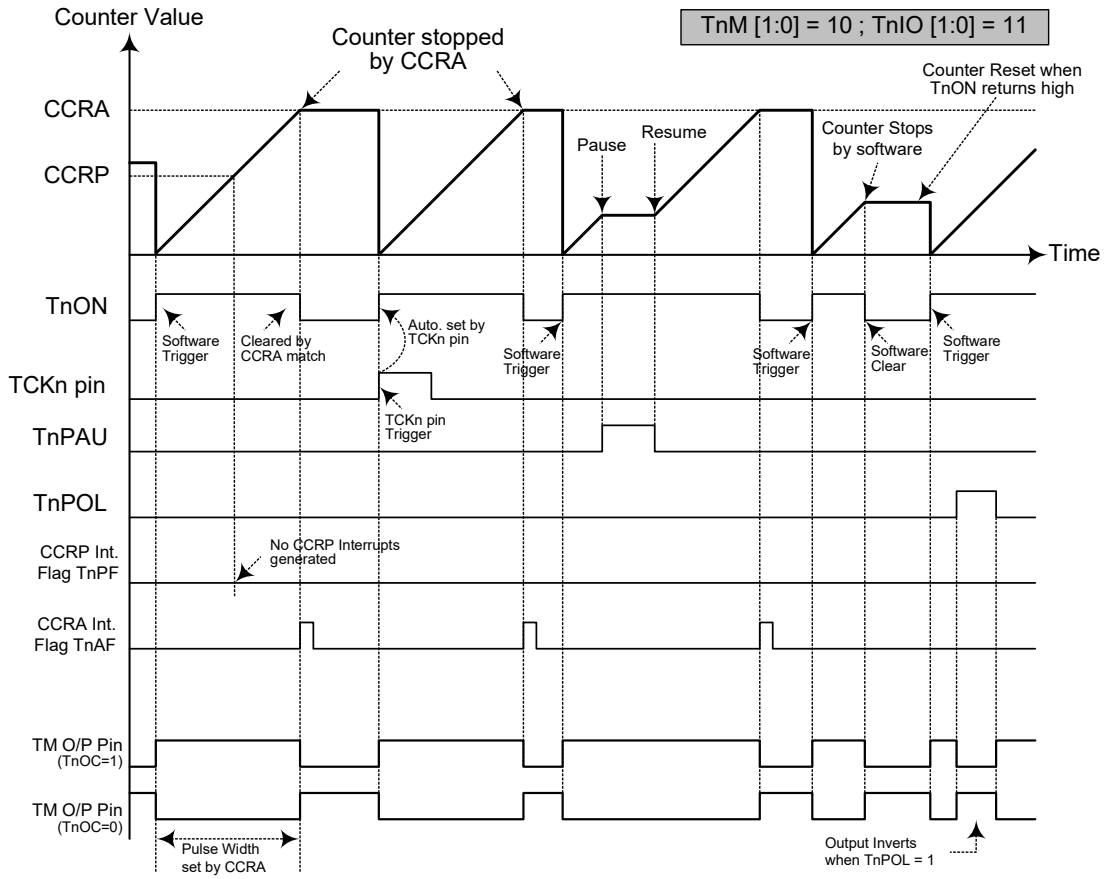
为使 TM 工作在此模式，TM0C1 寄存器中的 T0M1 和 T0M0 位需要设置为“10”，同时 T0IO1 和 T0IO0 位需要设置为“11”。正如模式名所言，单脉冲输出模式，在 TM 输出脚将产生一个脉冲输出。

脉冲输出可以通过应用程序控制 T0ON 位由低到高的转变来触发。而处于单脉冲输出模式时，T0ON 位在 TCK0 脚自动由低转变为高，进而初始化单脉冲输出状态。当 T0ON 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 T0ON 位保持高电平。通过应用程序使 T0ON 位清零或比较器 A 比较匹配发生时，产生脉冲下降沿。

然而，比较器 A 比较匹配发生时，会自动清除 T0ON 位并产生单脉冲输出下降沿。CCRA 的值通过这种方式控制脉冲宽度。比较器 A 比较匹配发生时，也会产生 TM 中断。T0ON 位在计数器重启时会发生由低到高的转变，此时计数器才复位至零。在单脉冲输出模式中，CCRP 寄存器和 T0CCLR 位未使用。



单脉冲产生示意图 (n=0)



单脉冲输出模式

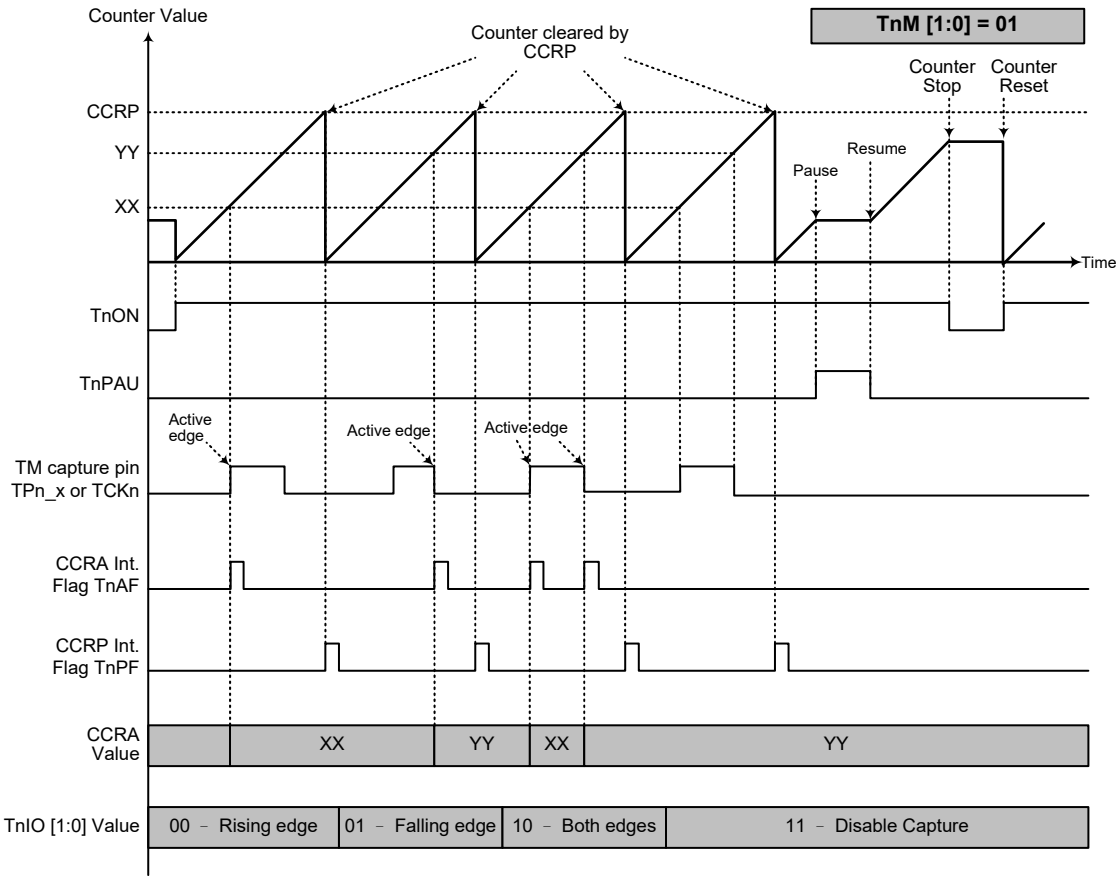
注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 TCKn 脚或设置 TnON 位为高来触发脉冲
4. TCKn 脚有效沿会自动置位 TnON
5. 单脉冲输出模式中，TnIO[1:0] 需置位“11”，且不能更改
6. n=0

捕捉输入模式

为使 TM 工作在此模式，TM0C1 寄存器中的 T0M1 和 T0M0 位需要设置为“01”。此模式使能外部信号捕捉并保存内部计数器当前值，因此被用于诸如脉冲宽度测量的应用中。TP0_0、TP0_1 或 TCK0 脚上的外部信号，通过设置 TM0C0 寄存器的 T0CAPTS 位选择。可通过设置 TM0C1 寄存器的 T0IO1 和 T0IO0 位选择有效边沿类型，即上升沿，下降沿或双沿有效。计数器在 T0ON 位由低到高转变时启动并通过应用程序初始化。

当 TP0_0、TP0_1 或 TCK0 引脚出现有效边沿转换时，计数器当前值被锁存到 CCRA 寄存器，并产生 TM 中断。不考虑 TP0_0、TP0_1 或 TCK0 引脚事件，计数器继续工作直到 T0ON 位发生下降沿跳变。当 CCRP 比较匹配发生时计数器复位至零；CCRP 的值通过这种方式控制计数器的最大值。当比较器 P CCRP 比较匹配发生时，也会产生 TM 中断。记录 CCRP 溢出中断信号的值可以测量脉宽。通过设置 T0IO1 和 T0IO0 位选择 TP0_0、TP0_1 或 TCK0 引脚为上升沿、下降沿或双沿有效。不考虑 TP0_0、TP0_1 或 TCK0 引脚引脚事件，如果 T0IO1 和 T0IO0 位设置为高，不会产生捕捉操作，但计数器继续运行。

当 TP0_0、TP0_1 或 TCK0 引脚与其它功能共用，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。T0CCLR、T0OC 和 T0POL 位在此模式中未使用。

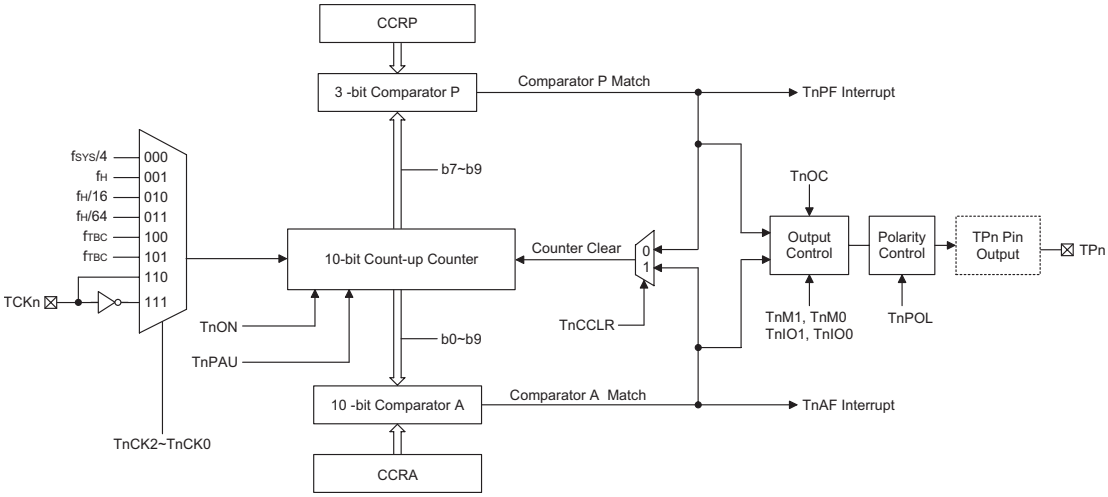


捕捉输入模式

注：1. TnM1，TnM0=01 并通过 TnIO1 和 TnIO0 位设置有效边沿
2. TM 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中
3. TnCCLR 位未使用
4. 无输出功能 -- TnOC 和 TnPOL 位未使用
5. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大
6. n=0

简易型 TM – CTM

虽然简易型 TM 是三种 TM 类型中最简单的形式，但仍然包括三种工作模式，即比较匹配输出，定时 / 事件计数器和 PWM 输出模式。简易型 TM 也由一个外部输入脚控制并驱动一个外部输出脚 TPn (n=1~3)。



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 位的，与计数器的高 3 位比较；而 CCRA 是 10 位的，与计数器的所有位比较。

通过应用程序改变 10 位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

每个简易型 TM 的所有操作由对应的六个寄存器控制。一对只读寄存器用来存放 10 位计数器的值，一对读 / 写寄存器存放 10 位 CCRA 的值，剩下两个控制寄存器设置不同的操作和控制模式以及 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

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。当此位由低到高转换时，内部计数器值将复位为零；当此位由高到低转换时，内部计数器将保持其剩余值。
若 TM 处于比较匹配输出模式时 (通过 TnOC 位指定)，当 TnON 位经由低到高的转换时，TM 输出脚将重置其初始值。

Bit 2~0 **TnRP2~TnRP0**：TMn CCRP 3-bit 寄存器，对应于 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，该比较结果用于清除内部计数器。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 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 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。这两位值的选择决定 TM 运行在哪种模式下。
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 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**：TPn 输出控制位
比较匹配输出模式
0：初始低
1：初始高
PWM 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 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 模式时，其决定 PWM 信号是高有效还是低有效。

Bit 2 **TnPOL**：TPn 输出极性控制位
0：同相
1：反相
此位控制 TPn 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 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 **TMnDL**：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 **TMnDH**：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 **TMnAL**：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　**TMnAH**：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

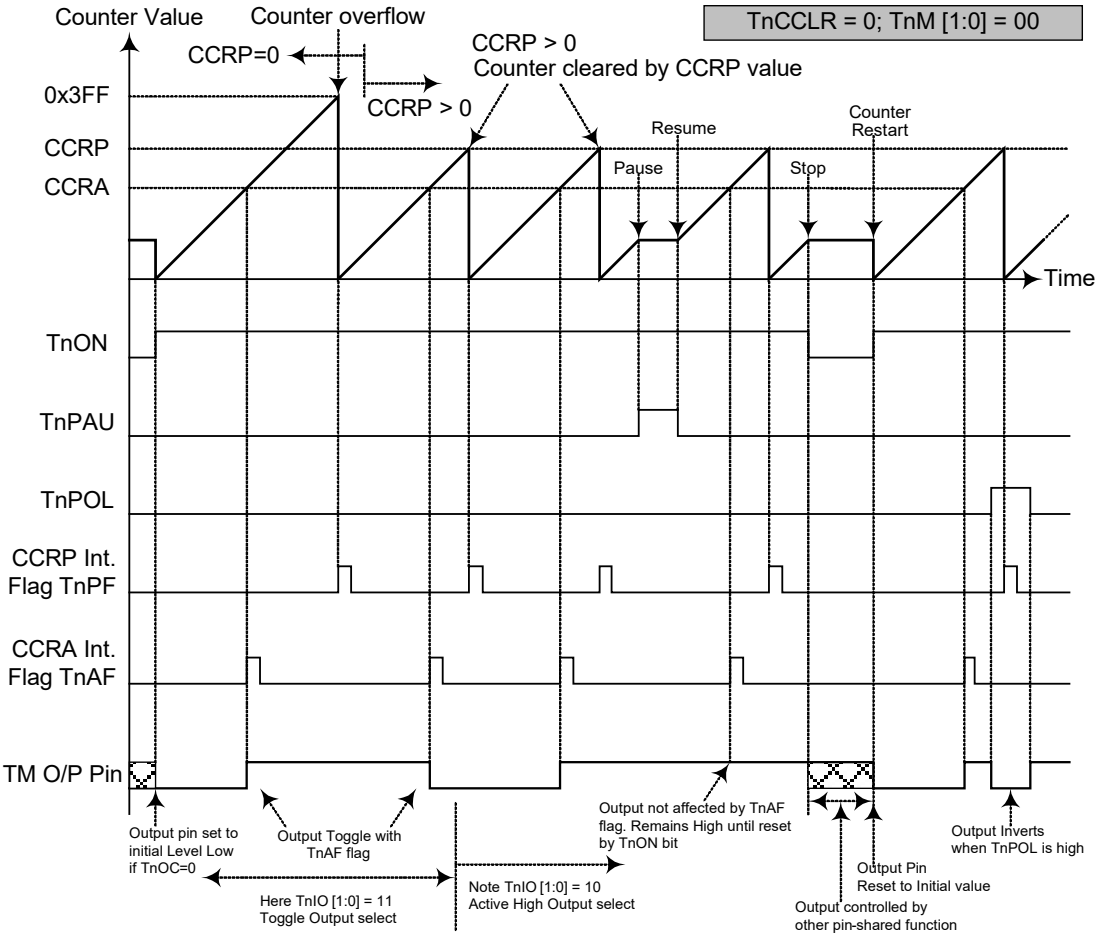
简易型 TM 有三种工作模式，即比较匹配输出模式，PWM 模式或定时 / 计数器模式。通过设置 TMnC1 寄存器的 TnM1 和 TnM0 位选择任意工作模式。

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。

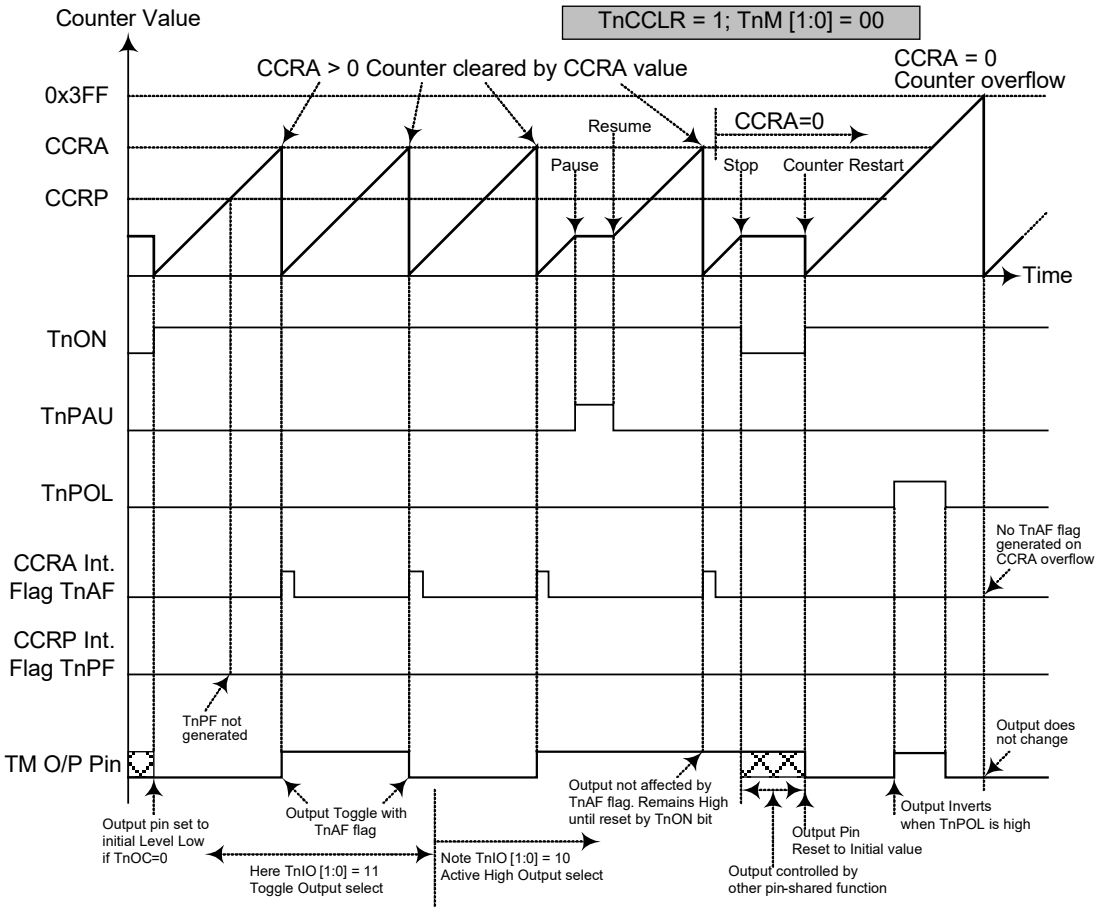
如果 TMnC1 寄存器的 TnCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 TnAF 中断请求标志产生。所以当 TnCCLR 为高时，不产生 TnPF 中断请求标志。如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 TnAF 请求标志。

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，既可以通过 TnON 位由低到高电平的变化设置，也可以由 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

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. 在 TnON 上升沿 TM 输出脚复位至初始值
4. n=1~3



比较匹配输出模式 -- TnCCLR=1

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. 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
5. n=1~3

定时 / 计数器模式

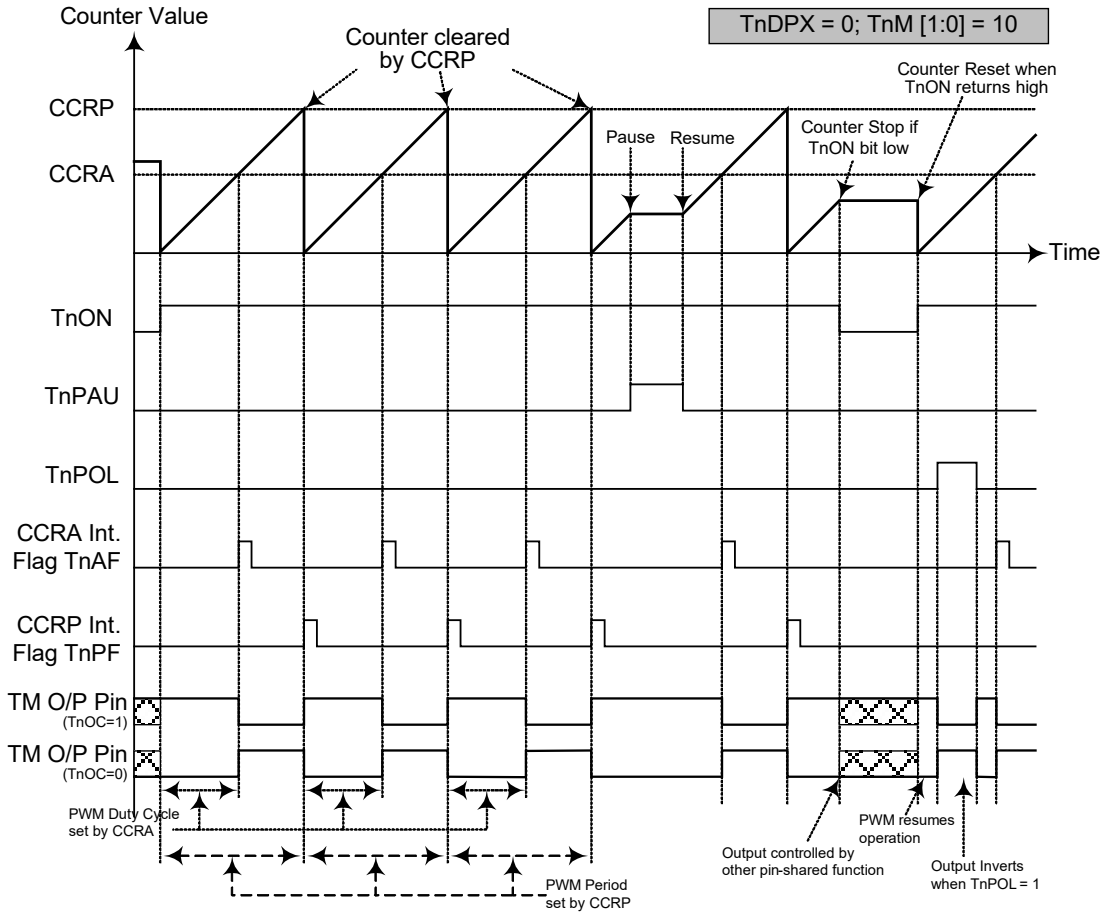
为使 TM 工作在此模式，TMnC1 寄存器中的 TnM1 和 TnM0 位需要设置为“11”。定时 / 计数器模式与比较输出模式操作方式相同，并产生同样的中断请求标志。不同的是，在定时 / 计数器模式下 TM 输出脚未使用。因此，比较匹配输出模式中的描述和时序图可以适用于此功能。该模式中未使用的 TM 输出脚用作普通 I/O 脚或其它功能。

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。

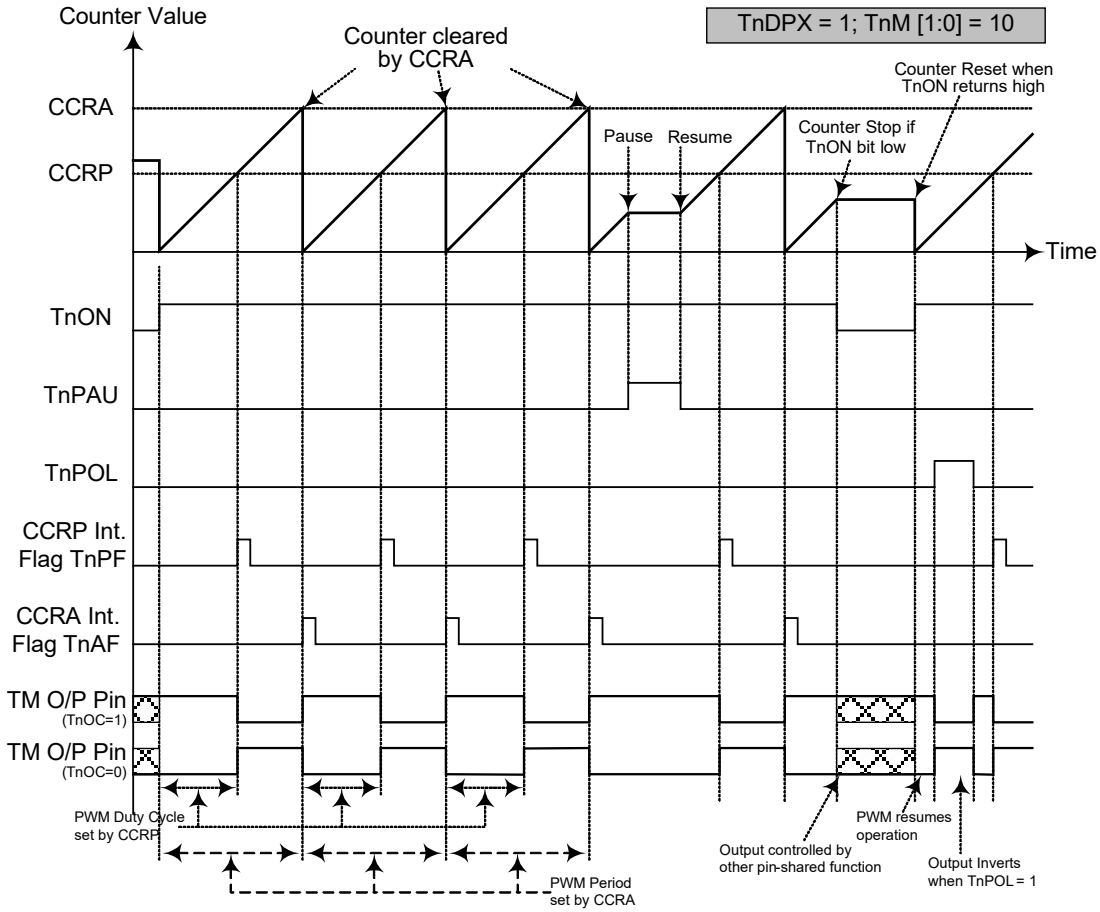
由于 PWM 波形的周期和占空比可调，其波形的选择就较为灵活。在 PWM 模式中，TnCCLR 位不影响 PWM 操作。CCRA 和 CCRP 寄存器决定 PWM 波形，一个用来清除内部计数器并控制 PWM 波形的频率，另一个用来控制占空比。哪个寄存器控制频率或占空比取决于 TMnC1 寄存器的 TnDPX 位。所以 PWM 波形频率和占空比由 CCRA 和 CCRP 寄存器共同决定。

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。



PWM 模式 -- TnDPX=0

注：1. TnDPX=0，CCRP 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 TnIO1，TnIO0=00 或 01，PWM 功能不变
4. TnCCLR 位不影响 PWM 操作
5. n=1~3



PWM 模式 -- TnDPX=1

注：1. TnDPX=1，CCRA 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 TnIO1，TnIO0=00 或 01，PWM 功能不变
4. TnCCLR 位不影响 PWM 操作
5. n=1~3

A/D 转换器

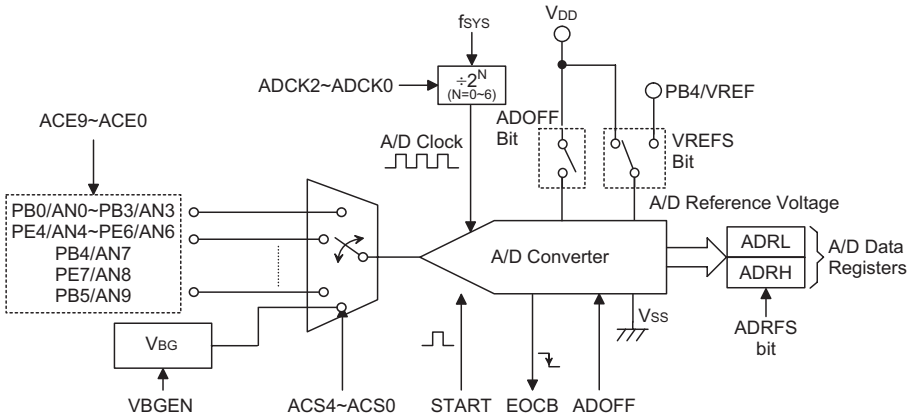
对于大多数电子系统而言，处理现实世界的模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先需要通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

A/D 简介

该系列单片机都包含一个多通道的 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 位的数字量。

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

下图显示了 A/D 转换器内部结构和相关的寄存器。



A/D 转换器结构

A/D 转换寄存器介绍

A/D 转换器的所有工作由六个寄存器控制。一对只读寄存器来存放 12 位 ADC 数据的值。剩下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 转换寄存器列表

A/D 转换器数据寄存器 – ADRL, ADRH

对于具有 12 位 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 位中的 12 位，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 换转数据结果位。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	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D 数据寄存器

A/D 转换控制寄存器 – 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 位的寄存器定义包括选择连接至内部 A/D 转换器的模拟通道，数字化数据格式，A/D 时钟源，并控制和监视 A/D 转换器的开始和转换结束状态。寄存器 ADCR0 的 ACS3~ACS0 位和 ADCR1 的 ACS4 位定义 A/D 转换器输入通道编号。由于每个单片机只包含一个实际的模数转换电路，因此这 10 个模拟输入中的每一个都需要分别被发送到转换器。ACS4~ACS0 位的功能决定选择哪个模拟输入通道或内部 VBG 被连接到内部 A/D 转换器。

ACERL 控制寄存器中的 ACE7~ACE0 位以及 ACERH 控制寄存器中的 ACE9~ACE8 位用来定义 PB 和 PE 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。相应位设为高将选择 A/D 输入功能，清零将选择 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
0→1：重置 A/D 转换，并且设置 EOCB 为“1”
此位用于初始化 A/D 转换过程。通常此位为低，但如果设为高再被清零，将初始化 A/D 转换过程。当此位为高，将重置 A/D 转换器。

Bit 6 **EOCB**：A/D 转换结束标志
0：A/D 转换结束
1：A/D 转换中
此位用于表明 A/D 转换过程的完成。当转换正在进行时，此位为高。

Bit 5 **ADOFF**：ADC 模块电源开 / 关控制位
0：ADC 模块电源开
1：ADC 模块电源关
此位控制 A/D 内部功能的电源。该位被清零将使能 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。由于 A/D 转换器在不执行转换动作时都会产生一定的功耗，所以这在电源敏感的电池应用中需要多加注意。
注：1. 建议在进入空闲 / 休眠模式前，设置 ADOFF=1 以减少功耗。
2. ADOFF=1 将关闭 ADC 模块的电源。

Bit 4 **ADRFS**：ADC 数据格式控制位
0：ADC 数据高字节是 ADRH 的 bit 7~bit 0，低字节是 ADRL 的 bit 7~bit 4
1：ADC 数据高字节是 ADRH 的 bit 3~bit 0，低字节是 ADRL 的 bit 7~bit 0
此位控制存放在两个 A/D 数据寄存器中的 12 位 A/D 转换结果的格式。细节方面请参考 A/D 数据寄存器章节。

Bit 3~0 **ACS3~ACS0**：选择 A/D 通道 (ACS4 为“0”)
0000：AN0
0001：AN1
0010：AN2
0011：AN3
0100：AN4
0101：AN5
0110：AN6
0111：AN7
1000：AN8
1001~1111：AN9
这四位是 A/D 通道选择控制位。由于只包含一个内部 A/D 转换电路，因此通过这些位将 10 个 A/D 输入连接到转换器。如果 ADCR1 寄存器中的 ACS4 设为高，内部 V_{BG} 电路将被连接到内部 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**：选择内部 V_{BG} 作为 ADC 输入控制位
0：除能
1：使能
此位使能 V_{BG} 连接到 A/D 转换器。VBGEN 位必须先被置位使能 V_{BG} 电压能隙电路被用于 A/D 转换器。当 ACS4 设为高，V_{BG} 带隙电压将连接到 A/D 转换器，其它 A/D 输入通道断开。

Bit 6 **VBGEN**：内部 V_{BG} 控制位
0：除能
1：使能
此位控制连接到 A/D 转换器的内部充电泵电路开 / 关功能。当此位设为高，带隙电压 V_{BG} 连接至 A/D 转换器。如果 V_{BG} 未连接至 A/D 转换器且 LVR/LVD 除能，带隙参考电压电路自动关闭以减少功耗。当 V_{BG} 打开连接至 A/D 转换器，在 A/D 转换动作执行前，带隙电路稳定需一段时间 t_{BG}。

Bit 5 未定义，读为“0”

Bit 4 **VREFS**：选择 ADC 参考电压
0：内部 ADC 电源
1：VREF 引脚
此位用于选择 A/D 转换器的参考电压。如果该位设为高，A/D 转换器参考电压来源于外部 VREF 引脚。如果该位设为低，内部参考电压来源于电源电压 VDD 引脚。

Bit 3 未定义，读为“0”

Bit 2~0 **ADCK2~ADCK0**：选择 ADC 时钟源
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。

● 间隙参考电压 on/off 真值表

ACS4	VBGEN	LVR/LVD	VBG	间隙参考电压
×	0	使能	Off (接地)	On
×	0	除能	Off (接地)	Off
×	1	×	On	On

×：无关

● **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	1	1	1	1	1	1	1	1

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	—	—	—	—	—	—	1	1

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 操作

ADCR0 寄存器中的 START 位，用于打开和复位 A/D 转换器。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，但不再回到逻辑低时，ADCR0 寄存器中的 EOCB 位置“1”，复位模数转换器。START 位用于控制内部模数转换器的开启动作。

ADCR0 寄存器中的 EOCB 位用于表明模数转换过程的完成。在转换周期结束后，EOCB 位会被单片机自动地置为“0”。此外，也会置位中断控制寄存器内相应的 A/D 中断请求标志位，如果中断使能，就会产生对应的内部中断信号。A/D 内部中断信号将引导程序到相应的 A/D 内部中断入口。如果 A/D 内部中断被禁止，可以让单片机轮询 ADCR0 寄存器中的 EOCB 位，检查此位是否被清除，以作为另一种侦测 A/D 转换周期结束的方法。

A/D 转换器的时钟源为系统时钟 f_{SYS} 或其分频，而分频系数由 ADCR1 寄存器中的 ADCK2~ADCK0 位决定。

虽然 A/D 时钟源是由系统时钟 f_{SYS} 和 ADCK2~ADCK0 位决定，但可选择的最大 A/D 时钟源则有一些限制。由于允许的 A/D 时钟周期 t_{ADCK} 的范围为 0.5μs~10μs，所以选择系统时钟速度时就必须小心。如果系统时钟速度为 4MHz 时，ADCK2~ADCK0 位不能设为“000”或“110”。必须保证设定的 A/D 转换时钟周期不小于时钟周期的最小值或大于时钟周期的最大值，否则将会产生不准确的 A/D 转换值。使用者可以参考下面的表格，被标上星号 * 的数值是不允许的，因为它们的 A/D 转换时钟周期不在规定的范围内。

f_{SYS}	A/D 时钟周期 (t_{ADCK})							
	ADCK2, ADCK1, ADCK0 =000 (f_{SYS})	ADCK2, ADCK1, ADCK0 =001 ($f_{SYS}/2$)	ADCK2, ADCK1, ADCK0 =010 ($f_{SYS}/4$)	ADCK2, ADCK1, ADCK0 =011 ($f_{SYS}/8$)	ADCK2, ADCK1, ADCK0 =100 ($f_{SYS}/16$)	ADCK2, ADCK1, ADCK0 =101 ($f_{SYS}/32$)	ADCK2, ADCK1, ADCK0 =110 ($f_{SYS}/64$)	ADCK2, ADCK1, ADCK0 =111
1MHz	1μs	2μs	4μs	8μs	16μs*	32μs*	64μs*	未定义
2MHz	500ns	1μs	2μs	4μs	8μs	16μs*	32μs*	未定义
4MHz	250ns*	500ns	1μs	2μs	4μs	8μs	16μs*	未定义
8MHz	125ns*	250ns*	500ns	1μs	2μs	4μs	8μs	未定义

A/D 时钟周期范例

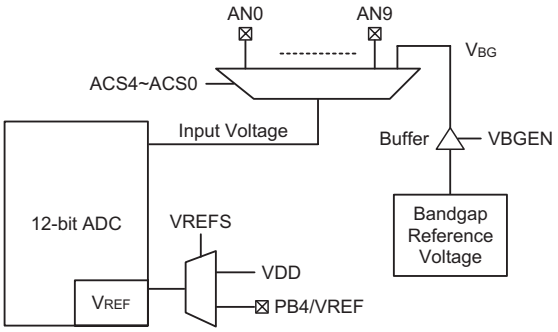
ADCR0 寄存器的 ADOFF 位用于控制 A/D 转换电路电源的开 / 关。该位必须清零以开启 A/D 转换器电源。即使通过清除 ACERL 寄存器的 ACE7~ACE0 位以及 ACERH 寄存器中的 ACE9~ACE8 位，选择无引脚作为 A/D 输入，如果 ADOFF 设为“0”，那么仍然会产生功耗。因此当未使用 A/D 转换器功能时，在功耗敏感的应用中建议设置 ADOFF 为高以减少功耗。

A/D 转换器参考电压来自正电源电压引脚 VDD 或外部参考源引脚 VREF，可通过 VREFS 位来选择。由于 VREF 引脚与其它功能共用，当 VREFS 设为高，选择 VREF 引脚功能且其它引脚功能将自动除能。

A/D 输入引脚

所有的 A/D 模拟输入引脚都与 PB 和 PE 端口的 I/O 引脚及其它功能共用。使用 ACERL 寄存器中的 ACE7~ACE0 位以及 ACERH 寄存器中的 ACE9~ACE8 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。如果引脚的对应位 ACE9~ACE0 设为高，那么该引脚作为 A/D 转换输入且原引脚功能除能。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，PBC 和 PEC 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 ACE9~ACE0 位使能 A/D 输入时，端口控制寄存器的状态将被重置。

A/D 转换器有自己的参考电压引脚 VREF，而通过设置 ADCR1 寄存器的 VREFS 位，参考电压也可以选择来自电源电压引脚。模拟输入值一定不能超过 V_{REF} 值。



A/D 输入结构

A/D 转换步骤

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 ADCK2~ADCK0 位，选择所需的 A/D 转换时钟。
- 步骤 2
  清零 ADCR0 寄存器中的 ADOFF 位使能 A/D。
- 步骤 3
  通过 ADCR1 和 ADCR0 寄存器中的 ACS4~ACS0 位，选择连接至内部 A/D 转换器的通道。
- 步骤 4
  通过 ACERL 寄存器中的 ACE7~ACE0 位以及 ACERH 寄存器中的 ACE9~ACE8 位，选择哪些引脚规划为 A/D 输入引脚。

- 步骤 5

  如果要使用中断，则中断控制寄存器需要正确地设置，以确保 A/D 转换功能是激活的。总中断控制位 EMI 需要置位为“1”，以及 A/D 转换器中断位 ADE 也需要置位为“1”。

- 步骤 6

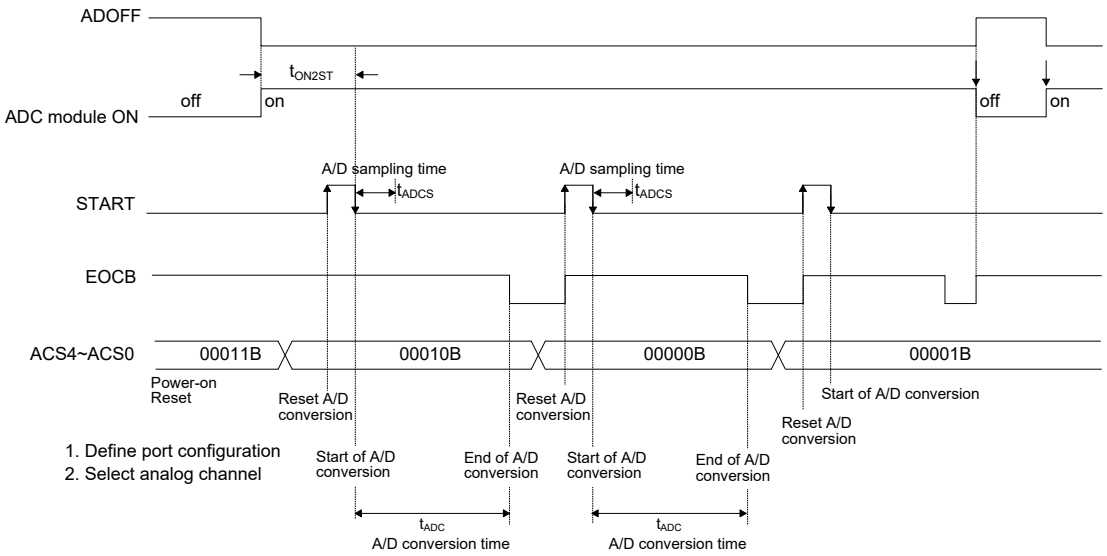
  现在可以通过设定 ADCR0 寄存器中的 START 位从“0”到“1”再回到“0”，开始模数转换的过程。注意，该位需初始化为“0”。

- 步骤 7

  可以轮询 ADCR0 寄存器中的 EOCB 位，检查模数转换过程是否完成。当此位成为逻辑低时，表示转换过程已经完成。转换完成后，可读取 A/D 数据寄存器 ADRL 和 ADRH 获得转换后的值。另一种方法是，若中断使能且堆栈未满，则程序等待 A/D 中断发生。

  注：若使用轮询 ADCR0 寄存器中 EOCB 位的状态的方法来检查转换过程是否结束时，则中断使能的步骤可以省略。

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



A/D 转换时序图

编程注意事项

在编程时，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

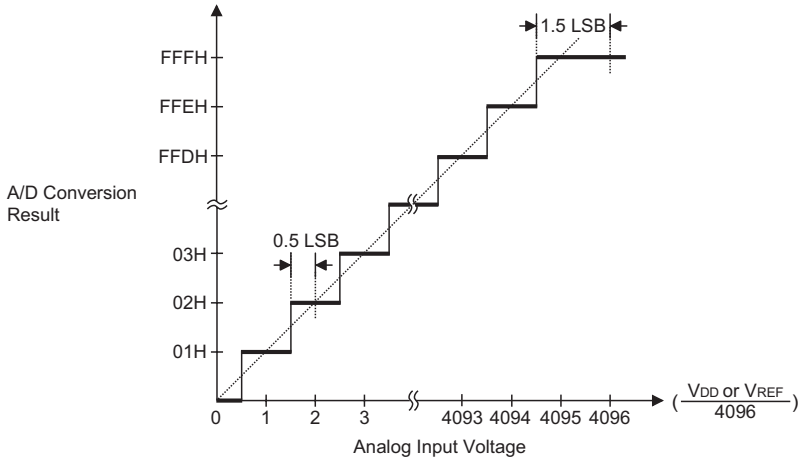
单片机含有一组 12 位的 A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于 V_{DD} 或 V_{REF} 的电压值，因此每一位可表示 V_{DD} 或 $V_{REF}/4096$ 的模拟输入值。

$$1\ LSB=(V_{DD} 或 V_{REF}) \div 4096$$

通过下面的等式可估算 A/D 转换器输入电压值：

A/D 输入电压 = A/D 数字输出值 × (V_{DD} 或 V_{REF}) ÷ 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_{DD} 或 V_{REF} 之前的 1.5 LSB 处改变。



理想的 A/D 转换功能

A/D 转换应用范例

下面两个范例程序用来说明怎样使用 A/D 转换。第一个范例是轮询 ADCR0 寄存器中的 EOCB 位来判断 A/D 转换是否完成；第二个范例则使用中断的方式判断。

范例：使用查询 EOCB 的方式来检测转换结束

```
clr ADE                    ; disable ADC 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,01h
mov ADCR0,a                ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr START                  ; high pulse on start bit to initiate conversion
set START                  ; reset A/D
clr START                  ; start A/D
polling_EOC:
sz 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
```

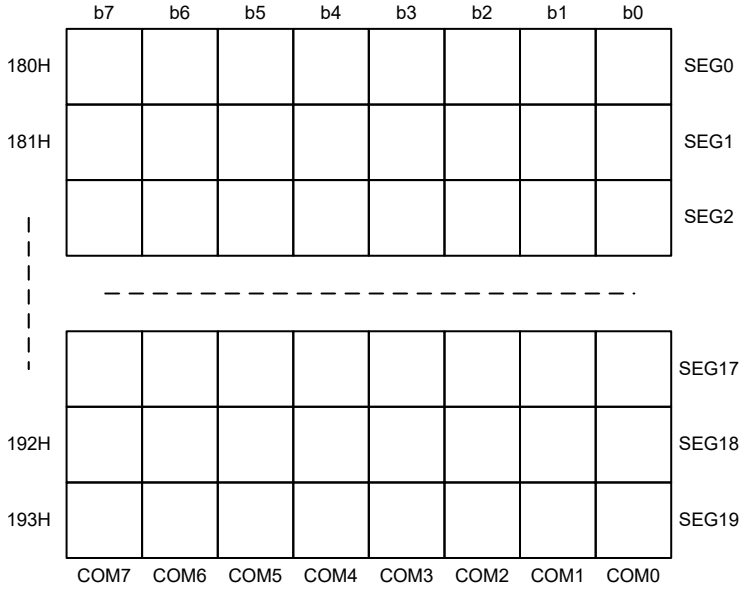
范例：使用中断的方式来检测转换结束

```
clr ADE                     ; disable ADC interrupt
mov a,03H
mov 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,01h
mov ADCR0,a                 ; enable and connect AN0 channel to A/D converter
Start_conversion:
clr START                   ; high pulse on START bit to initiate conversion
set START                   ; reset A/D
clr START                   ; start A/D
clr ADF                     ; clear ADC interrupt request flag
set ADE                     ; enable ADC interrupt
set EMI                     ; enable global interrupt
:
:
                            ; ADC interrupt service routine
ADC_ISR:
mov acc_stack,a             ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a          ; save STATUS to user defined memory
:
:
mov a,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_INT_ISR:
mov a,status_stack
mov STATUS,a                ; restore STATUS from user defined memory
mov a,acc_stack             ; restore ACC from user defined memory
reti
```

LCD 显示存储器

该系列单片机为LCD显示提供一个嵌入式数据存储区域。这个区域位于Sector 1的80H~93H。区域指针MP1H是通用存储器和LCD显示存储器之间切换的开关。当MP1H写入“01H”时，任何数据写入80H~93H将会影响LCD的显示。当MP1H设为“01H”外的其它值，任何数据写入80H~93H意味着访问一般意义上的数据存储器。

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和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驱动器的输出数目为20×8或20×4。LCD驱动器偏压产生方式为R型。LCD的时钟频率为f_{SUB}，来自LXT或LIRC振荡器。

LCD 控制寄存器

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 **LCDEN**：LCD 使能 / 除能控制位
0：除能
1：使能
关于引脚共用的LCD引脚和ADC引脚，若二者同时使能，LCD引脚享有更高优先级。

Bit 6 **TYPE**：LCD 类型选择位
0：A 型
1：B 型

Bit 5 **DTYC**：LCD 占空比选择位
0：1/4 占空比 (LCD COM：COM0~COM3)
1：1/8 占空比 (LCD COM：COM0~COM7)
若 DTYC=1，则 COM4~COM7 引脚设为 LCD COM；若 DTYC=0，则 COM4~COM7 为普通 I/O。

Bit 4 **BIAS**：LCD 偏置选择位
0：1/3 bias
1：1/4 bias

Bit 3 未定义，读为“0”

Bit 2~0 **RSEL2~RSEL0**：总偏置电阻 (R_T) 选择位
000：1170K
001：225K
010：60K
011：快速充电模式，切换至 60K~1170K 之间
1xx：快速充电模式，切换至 60K~225K 之间
注：1/3 偏置电压为 $R_T/3$，1/4 偏置电压为 $R_T/4$。

该系列单片机为 LCD 显示提供低功耗快速充电模式。在快速充电模式下，当 LCD 显示更新即 LCD COM 改变状态时，LCD 偏置电流通过选择 R_T=60K 产生，R_T 为总偏置电阻。快速充电后，总偏置电阻切换至 225K/1170K。

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 7~5 **QCT[2:0]**：快速充电时间选择位
000：1 t_{SUB}
001：2 t_{SUB}
010：3 t_{SUB}
011：4 t_{SUB}
100：5 t_{SUB}
101：6 t_{SUB}
110：7 t_{SUB}
111：8 t_{SUB}
$t_{SUB}=1/f_{SUB}$

Bit 4 未定义，读为“0”

Bit 3~0 **VLCD[3:0]**：VLCD 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	0	0	0	0	0	0	0	0

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	0	0	0	0	0	0	0	0

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	—	—	—	—	SEG19C	SEG18C	SEG17C	SEG16C
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”

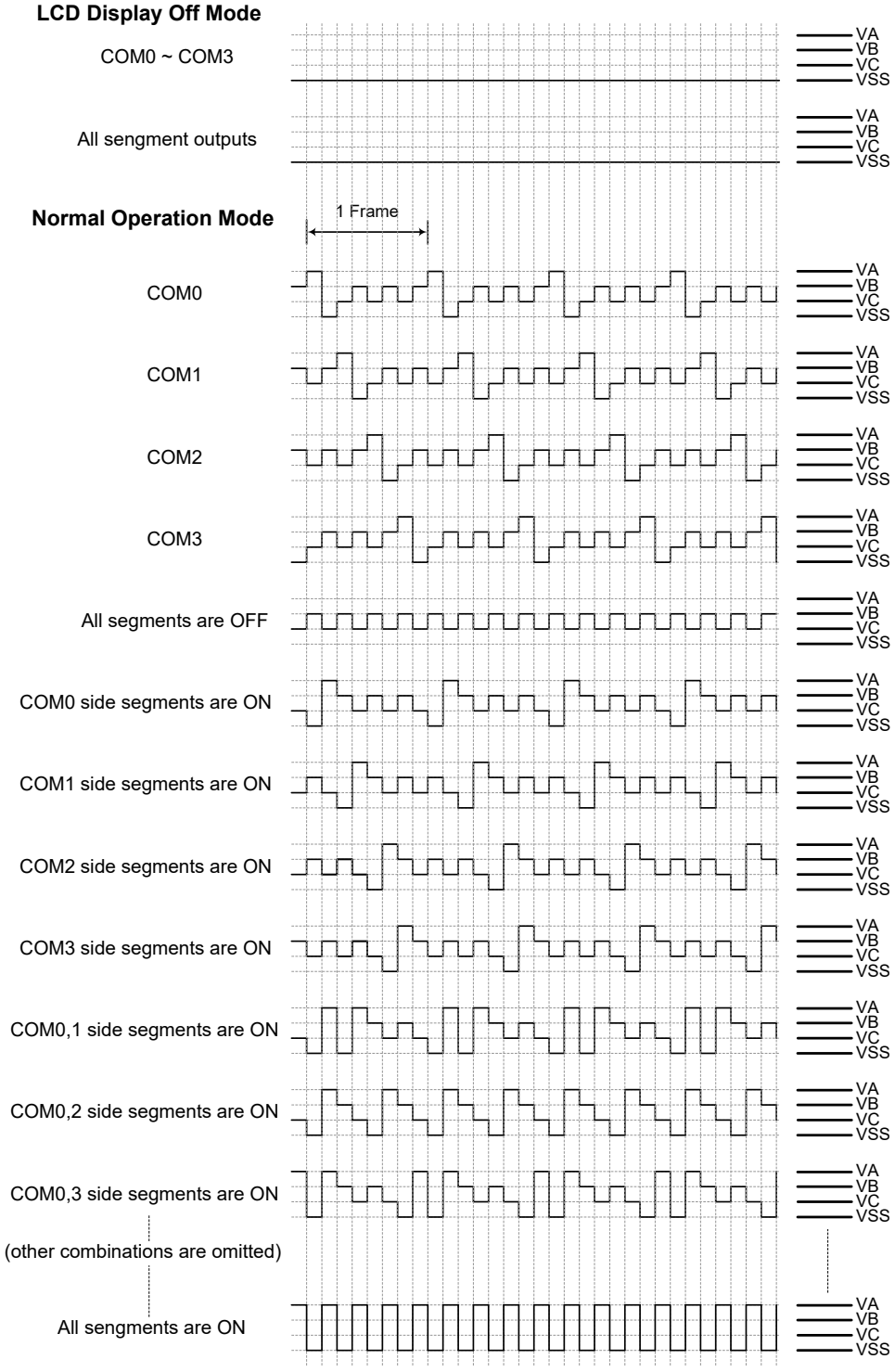
Bit 3 **SEG19C**：选择 SEG19 或 PF7
0：SEG19
1：PF7

Bit 2 **SEG18C**：选择 SEG18 或 PF6
0：SEG18
1：PF6

Bit 1 **SEG17C**：选择 SEG17 或 PF5
0：SEG17
1：PF5

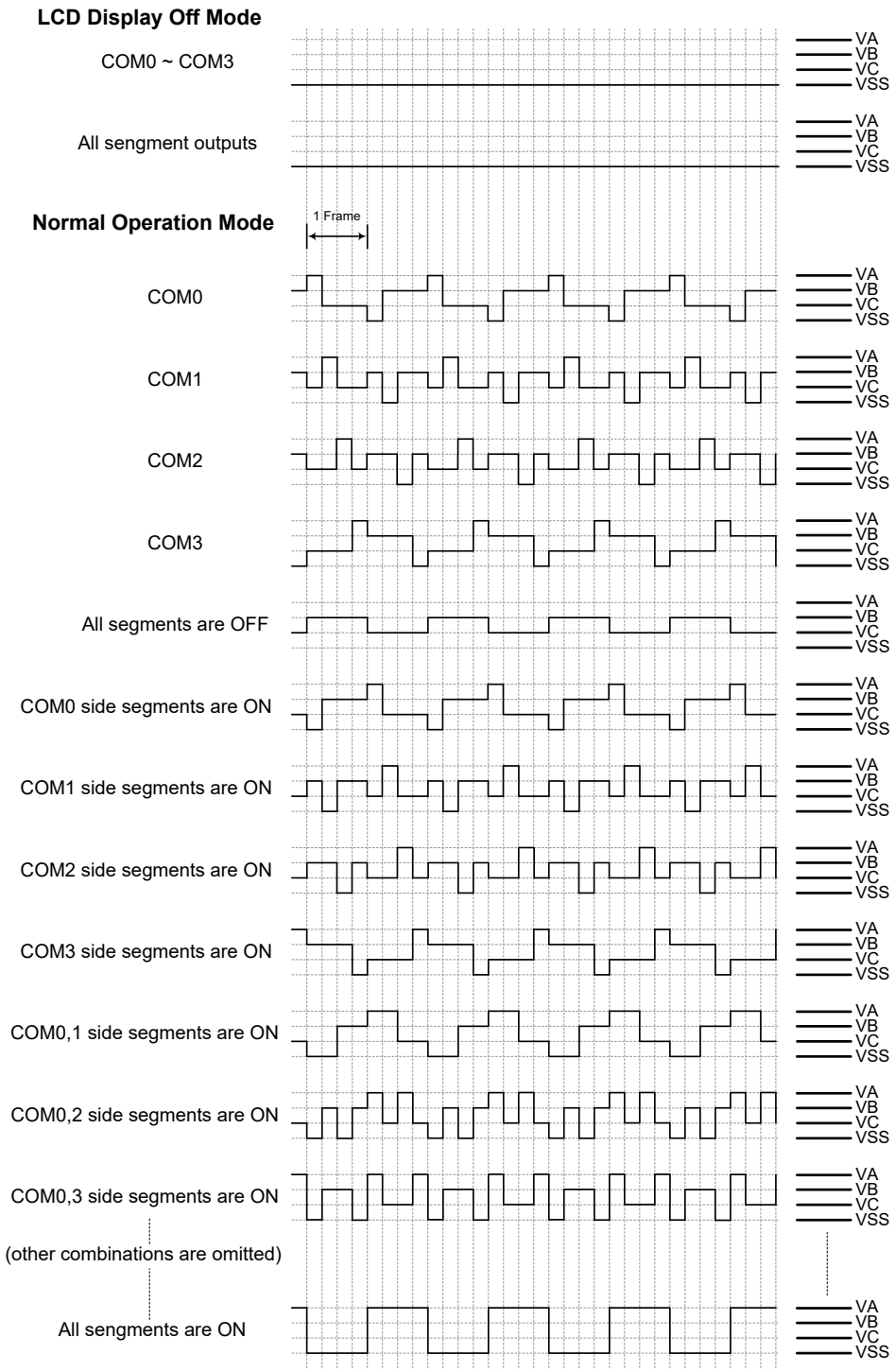
Bit 0 **SEG16C**：选择 SEG16 或 PF4
0：SEG16
1：PF4

LCD 波形时序图



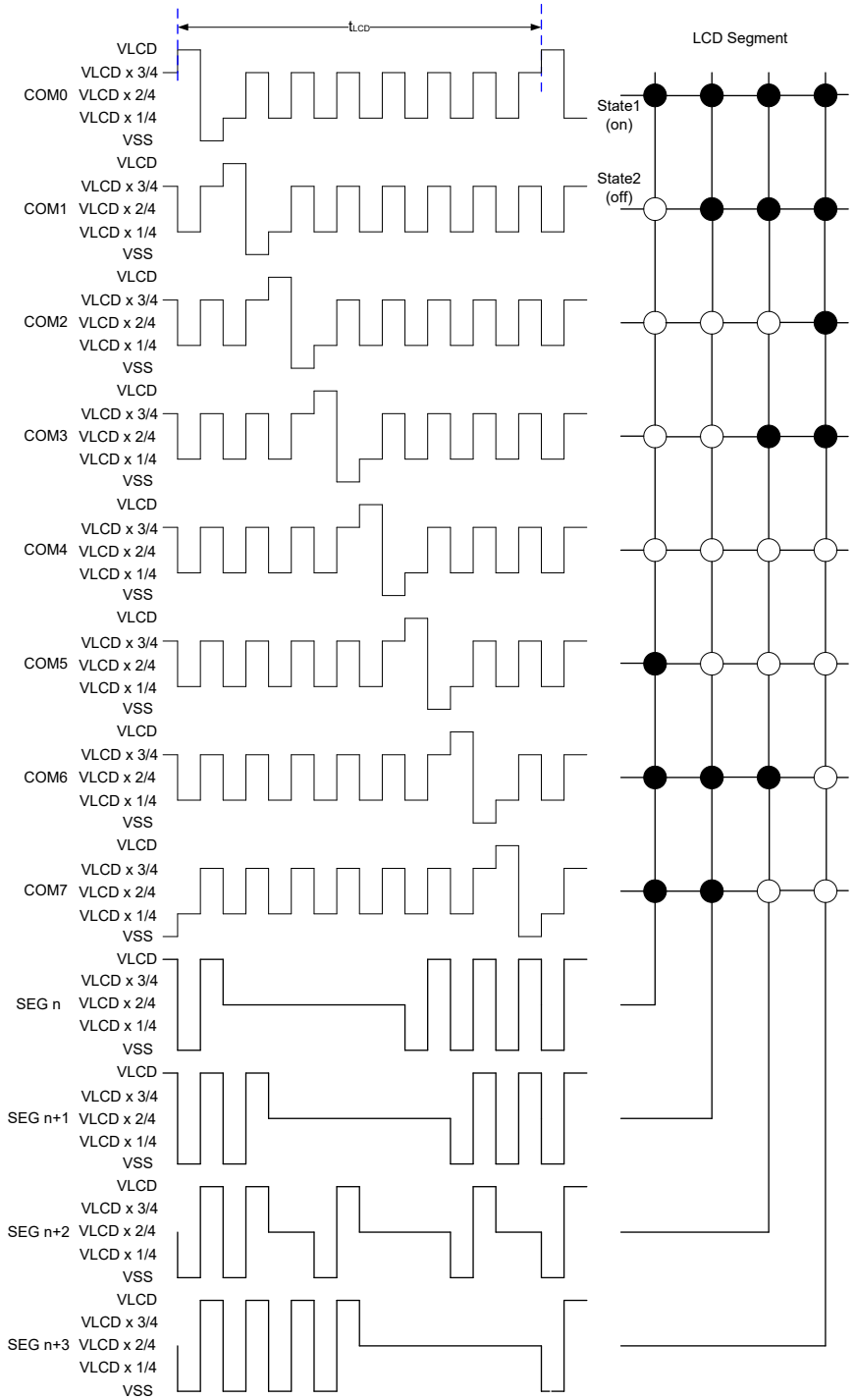
$V_A=V_{LCD}$，$V_B=V_{LCD}\times2/3$，$V_C=V_{LCD}\times1/3$，$V_{LCD}=V_{DD}$

LCD 驱动输出 (1/4 duty, 1/3 bias, A 型)



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

LCD 驱动输出 (1/4 duty, 1/3 bias, B 型)



$V_{LCD}=V_{DD}$

LCD 驱动输出 (1/8 duty, 1/4 bias, A 型)

LED 驱动器

该系列单片机具有 LED 驱动功能，通过输出大电流驱动外部 LED 设备。

LED 驱动操作

单片机 LED 大电流输出驱动引脚如下表所示。

单片机型号	LED 驱动引脚
HT67F488 HT67F489	PD0~PD7 (大源电流) PE0~PE7 (大灌电流)

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

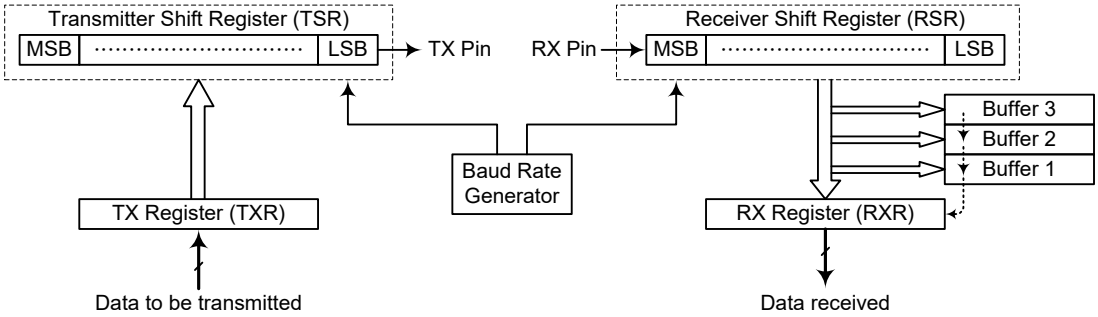
IOHSn[1:0]：PDn (n=0~7) 引脚输出 I_{OH} 选择位
00：全源电流驱动
01：1/3 源电流驱动
10：1/4 源电流驱动
11：1/6 源电流驱动

UART 接口

该系列单片机具有一个全双工的异步串行通信接口——UART，可以很方便的与其它具有串行口的芯片通信。UART 具有许多功能特性，发送或接收串行数据时，将数据组成一个 8 位或 9 位的数据块，连同数据特征位一并传输。具有检测数据覆盖或帧错误等功能。UART 功能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。

UART 模块特性如下：

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



UART 数据传输方框图

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 引脚分别为 UART 接口的发送和接收引脚。在使用 UART 功能之前，要先正确设置相关的引脚共用功能控制寄存器选择 RX 和 TX 引脚功能。若 UARTEN 位或 TXEN/RXEN 位为“0”时，TX 和 RX 引脚功能除能，这两个引脚可作为普通 I/O 口或共用功能引脚。若 UARTEN 位和 TXEN/RXEN 位均为“1”时，将会自动设置 TX 脚为输出，RX 为输入状态，同时断开 RX 和 TX 引脚上的上拉电阻。若 TX 和 RX 引脚与 LCD 输出共用引脚且 UART 接口与 LCD 驱动同时使能，则 LCD 驱动拥有较高优先级，相应引脚将为 LCD 输出功能。

UART 数据传输方案

方框图显示了 UART 的整体结构。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的控制下将 TSR 寄存器中数据一位位地移到 TX 引脚上，低位在前。TXR 寄存器被映射到单片机的数据存储器中，而发送移位寄存器没有实际地址，所以发送移位寄存器不可直接操作。

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

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/RXT。

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
BRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0

USR 寄存器

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

Bit 7 **PERR**：奇偶校验出错标志位
0：奇偶校验正确
1：奇偶校验出错
PERR 是奇偶校验出错标志位。若 PERR=0，奇偶校验正确；若 PERR=1，接收到的数据奇偶校验出错。只有使能了奇偶校验此位才有效。可使用软件清除该标志位，即先读取 USR 寄存器再读 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 NF=0，没有受到噪声干扰；若 NF=1，UART 接收数据时受到噪声干扰。它与 RXIF 在同周期内置位，但不会与溢出标志位同时置位。可使用软件清除该标志位，即先读取 USR 寄存器再读 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 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 OERR=0，没有溢出错误；若 OERR=1，发生了溢出错误，它将影响下一组数据的接收。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 RIDLE=0，正在接收数据；若 RIDLE=1，接收器空闲。在接收到停止位和下一个数据的起始位之间，RIDLE 被置位，表明 UART 空闲，RX 脚处于逻辑高状态。

Bit 2 **RXIF**：接收寄存器状态标志位
0：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 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，如果 UCR2 寄存器中的 RIE=1，则会触发中断。当接收数据时检测到一个或多个错误时，相应的标志位 NF、FERR 或 PERR 会在同一周期内置位。读取 USR 寄存器再读 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**：数据发送完成标志位
0：数据传输中
1：无数据传输
TIDLE 是数据发送完成标志位。若 TIDLE=0，数据传输中。当 TXIF=1 且数据发送完毕或者暂停字被发送时，TIDLE 置位。TIDLE=1，TX 引脚空闲且处于逻辑高状态。读取 USR 寄存器再写 TXR 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 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 存器再写 TXR 寄存器将清除 TXIF。当 TXEN 被置位，由于发送缓冲器未满，TXIF 也会被置位。

UCR1 寄存器

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

详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	×	0

“×”未知

Bit 7 **UARTEN**：UART 功能使能位
0：UART 除能，TX 和 RX 脚为 I/O 或其它共用功能引脚
1：UART 使能，TX 和 RX 脚作为 UART 功能引脚
此位为 UART 的使能位。若 TX 与 RX 引脚与 LCD 输出共用，且 UART 接口与 LCD 驱动同时使能，则 LCD 驱动拥有较高优先级，相应引脚将为 LCD 输出功能。UARTEN=0，UART 除能，RX 和 TX 可用作普通的输入输出口或其它共用功能引脚；UARTEN=1，UART 使能，TX 和 RX 将分别由 TXEN 和 RXEN 控制。当 UART 被除能将清除缓冲器，所有缓冲器中的数据将被忽略，另外波特率计

数器、错误和状态标志位被复位，TXEN、RXEN、TXBRK、RXIF、OERR、FERR、PERR 和 NF 清零而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2 和 BRG 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零，所有发送和接收将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

Bit 6 **BNO**：发送数据位数选择位
0：8-bit 传输数据
1：9-bit 传输数据
BNO 是发送数据位数选择位。BNO=1，传输数据为 9 位；BNO=0，传输数据为 8 位。若选择了 9 位数据传输格式，RX8 和 TX8 将分别存储接收和发送数据的第 9 位。

Bit 5 **PREN**：奇偶校验使能位
0：奇偶校验除能
1：奇偶校验使能
此位为奇偶校验使能位。PREN=1，使能奇偶校验；PREN=0，除能奇偶校验。

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

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

Bit 2 **TXBRK**：暂停字发送控制位
0：没有暂停字要发送
1：发送暂停字
TXBRK 是暂停字发送控制位。TXBRK=0，没有暂停字要发送，TX 引脚正常操作；TXBRK=1，将会发送暂停字，发送器将发送逻辑“0”。若 TXBRK 为高，缓冲器中数据发送完毕后，发送器将至少保持 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 位的格式中有效，用来存储接收数据的第 9 位。BNO 是用来控制传输位数是 8 位还是 9 位。

Bit 0 **TX8**：发送 9-bit 数据传输格式中的第 8 位 (只写)
此位只有在传输数据为 9 位的格式中有效，用来存储发送数据的第 9 位。BNO 是用来控制传输位数是 8 位还是 9 位。

UCR2 寄存器

UCR2 是 UART 的第二个控制寄存器，它的主要功能是控制发送器、接收器以及各种 UART 中断源的使能或除能。它也可用来控制波特率，使能接收唤醒和地址侦测。

详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TXEN**：UART 发送使能位
0：UART 发送除能
1：UART 发送使能
此位为发送使能位。TXEN=0，发送将被除能，发送器立刻停止工作。另外

缓冲器将被复位，此时 TX 引脚为 I/O 或其它共用功能引脚。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，此时 TX 引脚可作为 I/O 或其它共用功能引脚。

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，此时 RX 引脚为 I/O 或其它共用功能引脚。若 RXEN=1 且 UARTEN=1，则接收将被使能，RX 引脚将由 UART 来控制。在数据传输时清除 RXEN 将中止数据接收且复位接收器，此时 RX 引脚可作为 I/O 或其它共用功能引脚。

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) 或第 9 位 (BON=1) 为高，那么接到的是地址而非数据。若相应的中断使能且接收到的值最高位为 1，那么中断请求标志将会被置位，若最高位为 0，那么将不会产生中断且收到的数据也会被忽略。

Bit 3 **WAKE**：RX 脚下降沿唤醒功能使能位
0：RX 脚下降沿唤醒功能除能
1：RX 脚下降沿唤醒功能使能
此位为接收唤醒功能的使能和除能位。若 WAKE=1 且在 SLEEP 模式下，RX 引脚的下降沿将唤醒单片机 (请参考不同暂停模式下 RX 引脚的唤醒功能)。若 WAKE=0 且在 SLEEP 模式下，RX 引脚的任何边沿都不能唤醒单片机。

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

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	×	×	×	×	×	×	×	×

“×”：未知

Bit 7~0 **TXRX7~TXRX0**：UART 发送 / 接收数据位 Bit 7~Bit 0

BRG 寄存器

Bit	7	6	5	4	3	2	1	0
Name	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	×	×	×	×	×	×	×	×

“×”：未知

Bit 7~0 **BRG7~BRG0**：波特率值

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

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

若 BRGH=1，波特率 = $f_{SYS}/[16\times(N+1)]$。

波特率发生器

UART 自身具有一个波特率发生器，通过它可以设定数据传输速率。波特率是由一个独立的内部 8 位计数器产生，它由 BRG 寄存器和 UCR2 寄存器的 BRGH 位来控制。BRGH 是决定波特率发生器处于高速模式还是低速模式，从而决定计算公式的选用。BRG 寄存器的值 N 可根据下表中的公式计算，N 的范围是 0 到 255。

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

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 的值不连续，所以实际波特率和理论值之间有一个偏差。

下面举例怎样计算 BRG 寄存器中的值 N 和误差。

波特率和误差的计算

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 $BR=\frac{f_{SYS}}{[64(N+1)]}$

转换后的公式 $N=\frac{f_{SYS}}{(BR\times64)}-1$

带入参数 $N=\frac{4000000}{(4800\times64)}-1=12.0208$

取最接近的值，十进制 12 写入 BRG 寄存器，实际波特率如下：

$BR=\frac{4000000}{[64(12+1)]}=4808$

因此，误差 = $\frac{4808-4800}{4800}$ =0.16%

下面两表给出 BRGH 取不同值时的实际波特率和误差。

波特率 K/BPS	f_{SYS}=8MHZ					
	Baud Rates for BRGH=0			Baud Rates for BRGH=1		
	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

UART 模块的设置与控制

UART 采用标准的不归零码传输数据，这种方法通常被称为 NRZ 法。它由 1 位起始位，8 位或 9 位数据位和 1 位或者两位停止位组成。奇偶校验是由硬件自动完成的，可设置成奇校验、偶校验和无校验三种格式。常用的数据传输格式由 8 位数据位，1 位停止位，无校验组成，用 8、N、1 表示，它是系统上电的默认格式。数据位数、停止位数和奇偶校验由 UCR1 寄存器的 BNO、PRT、PREN 和 STOPS 设定。用于数据发送和接收的波特率由一个内部的 8 位波特率发送器产生，数据传输时低位在前高位在后。尽管 UART 发送器和接收器在功能上相互独立，但它们使用相同的数据传输格式和波特率，在任何情况下，停止位是必须的。

UART 的使能和除能

UART 是由 UCR1 寄存器的 UARTEN 位来使能和除能的。若 UARTEN、TXEN 和 RXEN 都为高，则 TX 和 RX 分别为 UART 的发送端口和接收端口。若没有数据发送，TX 引脚默认状态为高电平。

UARTEN 清零将除能 TX 和 RX，使其可作为 I/O 或其它共用功能引脚。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。

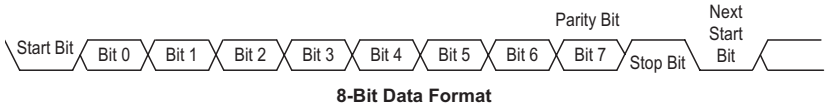
数据位、停止位位数以及奇偶校验的选择

数据传输格式由数据长度、是否校验、校验类型、地址位以及停止位长度组成。它们都是由 UCR1 寄存器的各个位控制的。BNO 决定数据传输是 8 位还是 9 位；PRT 决定校验类型；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。地址位用来确定此帧是否为地址。停止位的长度和数据位的长度无关。

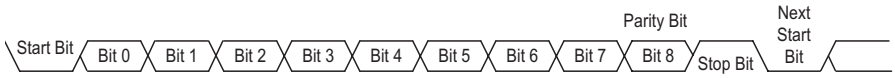
起始位	数据位	地址位	校验位	停止位
8 位数据位				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
9 位数据位				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

发送和接收数据格式

下图是传输 8 位和 9 位数据的波形。



8-Bit Data Format



9-Bit Data Format

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。TSR 不像其它寄存器一样映射到数据存储器，所以应用程序不能对其进行读写操作。TXEN=1，发送使能，但若 TXR 寄存器没有数据或者波特率没有设置，发送器将不会工作。先写 TXR 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 引脚可作为 I/O 或其它共用功能引脚。

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。如果选择 9 位数据传输格式，最高位 MSB 存储在 UCR1 寄存器的 TX8 中。

发送器初始化可由如下步骤完成：

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

如果要发送多个数据只需重复上一步骤。

当 TXIF=0 时，数据将禁止写入 TXR 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，其它数据可以写入而不会覆盖以前的数据。若 TEIE=1，TXIF 标志位会影响中断。在数据传输时，写 TXR 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 指令会将数据直接加载到 TSR 寄存器中，数据传输立刻开始且 TXIF 置位。当一帧数据发送完毕，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

清除 TXIF 和 TIDLE 软件执行次序相同。

发送暂停字

若 TXBRK=1，下一帧将会发送暂停字。它是由一个起始位、13×N（N=1，2……）位逻辑 0 组成。置位 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。需要注意思的是，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 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 RXR 寄存器。RX 引脚上的每一位数据会被采样三次以判断其逻辑状态。RSR 不像其它寄存器一样映射在数据存储器，所以应用程序不能对其进行读写操作。

接收数据

当 UART 接收数据时，数据低位在前高位在后，连续地从 RX 引脚进入。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，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。

接收器的初始化可由如下步骤完成：

- 正确地设置 BNO、PRT、PREN 和 STOPS 位以确定数据长度、校验类型和停止位长度。
- 设置 BRG 寄存器，选择期望的波特率。
- 置高 RXEN，使能 UART 发送器且使 RX 作为 UART 的接收端。

此时接收器被使能并检测起始位。

接收数据将会发生如下事件：

- 当 RXR 寄存器中有一帧以上的数据时，USR 寄存器中的 RXIF 位将会置位。
- 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

接收暂停字

UART 接收任何暂停字都会当作帧错误处理。接收器只根据 BNO 和 STOPS 位确定一帧数据的长度。若暂停字位数大于 BNO 和 STOPS 位指定的长度，接收器认为接收已完毕，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。若暂停字较长，接收器收到起始位、数据位将会置位 FERR 标志，且在下一起始位前必须检测到有效的停止位。暂停字只会被认为包含信息 0 且会置位 FERR 标志。暂停字将会加载到缓冲器中，在接收到停止位前不会再接收数据，没有检测到停止位也会置位只读标志位 RIDLE。

UART 接收到暂停字会产生以下事件：

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

空闲状态

当 UART 接收数据时，即在起初位和停止位之间，USR 寄存器的接收标志位 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。

接收错误处理

UART 会产生几种接收错误，下面部分将描述各错误以及怎样处理。

溢出——OERR 标志

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

产生溢出错误时将会发生以下事件：

- USR 寄存器中 OERR 被置位。
- 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

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，此位置位的同时由 RXIF 请求中断。

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

若在停止位上检测到 0，USR 寄存器中只读标志 FERR 置位。若选择两位停止位，此两位都必须为高，否则将置位 FERR。它同数据一起存储在缓冲器中，可被任何复位清零。

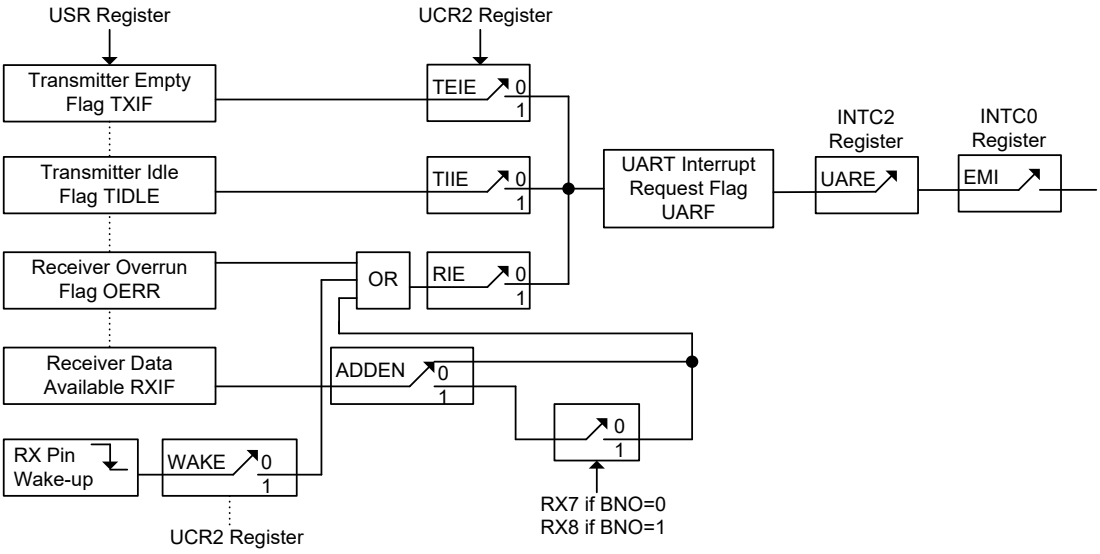
奇偶校验错误——PERR 标志

若接收到数据出现奇偶校验错误，USR 寄存器中只读标志 PERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。它同数据一起存储在缓冲器中，可被任何复位清除。注意，FERR 和 PERR 与相应的数据一起存储有缓冲器中，在读取数据之前必须先访问错误标志位。

UART 模块中断结构

UART 拥有一个单独的中断。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。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。这其中的四种情况在 USR 寄存器中有相关的标志位，若 UCR2 寄存器中相应中断允许位被置位，USR 寄存器中标志位将会产生中断。发送器有两个相应的中断允许位而接收器共用一个中断允许位。这些允许位可用于禁止个别的 UART 中断源。地址检测也是 UART 的中断源，它没有相应的标志位，若 UCR2 寄存器中 ADDEN=1，当检测到地址将会产生 UART 中断。RX 引脚唤醒也可以产生 UART 中断，它没有相应的标志位，当 UXR2 中的 WAKE 和 RIE 位被置位，RX 引脚上有下降沿可以唤醒单片机。应注意，RX 唤醒中断发生时，系统必须延时 t_{SST} 才能正常工作。

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。整体 UART 中断的使能或除能可由中断控制寄存器中的相关中断使能控制位控制，其中断请求由 UART 模块决定。



UART 中断框图

地址检测模式

置位 UCR2 寄存器中的 ADDEN 将启动地址检测模式。若此位为“1”，可产生接收数据有效中断，其请求标志位为 RXIF。若 ADDEN 有效，只有在接收到数据最高位为 1 才会产生中断，中断允许位 UARE 和 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，每接收到一个有效数据便会置位 RXIF，而不用考虑数据的最后一位。地址检测和奇偶校验在功能上相互排斥，若地址检测模式使能，必须保证操作的正确，同时必须将奇偶检验使能位清零，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

UART 模块暂停和唤醒

MCU 进入暂停模式后 UART 模块将暂停运行。当 UART 进入暂停模式后，UART 模块的时钟源将除能。当传送数据时 UART 进入暂停模式，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 UART 进入暂停模式，数据接收也会停止。当 UART 电路进入暂停模式，USR、UCR1、UCR2 、接收 / 发送寄存器以及 BRG 寄存器都不会受到影响。建议在 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。进入暂停模式前，若该标志位与 UART 允许位 UARTEN、接收器允许位 RXEN 和接收器中断位 RIE 都被置位，则 RX 引脚的下降沿可唤醒单片机。唤醒后系统需延时 t_{SST} 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；若这两控制位没有被置位，那么，单片机将可以被唤醒但不会产生中断。同样唤醒后系统需一定的延时才能正常工作，然后才会产生 UART 中断。

下面表格说明了在各暂停模式下 UART 模块 RX 引脚的唤醒功能。

工作模式	说明				RX 引脚唤醒功能
	CPU	f_{SYS}	f_H	f_{SUB}	
空闲模式 0	Off	Off	Off	On	当 CPU 进入空闲模式 0， 即使 UCR2.2(RIE)=1, UCR2.3(WAKE)=1, RX 引脚的下降沿不會开启 f_{SYS}，也不會唤醒 CPU。
空闲模式 1	Off	On	On	On	当 UCR2.2(RIE)=1、UCR2.3(WAKE)=1 且 CPU 进入空闲模式 1： (1) 若 UART 没有在传输数据，RX 引脚的下降沿将开启 f_{SYS} 而 CPU 仍关闭。 (2) 若 UART 正在传输数据，CPU 将在数据传输完成后唤醒。 注：若 RIE=0、WAKE=1 且 UART 正在传输数据，CPU 在数据传输完成后也不唤醒。
空闲模式 1	Off	On ($f_{SYS}=f_H$~$f_H/64$)	On	Off	当 UCR2.2(RIE)=1、UCR2.3(WAKE)=1 且 CPU 进入空闲模式 1： (1) 若 UART 没有在传输数据，RX 引脚的下降沿将开启 f_{SYS} 而 CPU 仍关闭。 (2) 若 UART 正在传输数据，CPU 将在数据传输完成后唤醒。 注：若 RIE=0、WAKE=1 且 UART 正在传输数据，CPU 在数据传输完成后也不唤醒。
休眠模式 0/1	Off	Off	Off	On/Off	当 UCR2.2(RIE)=1、UCR2.3(WAKE)=1 且 CPU 进入休眠模式， RX 引脚的下降沿将开启 f_{SYS} 并唤醒 CPU。

中断

中断是单片机一个重要功能。当外部事件或内部功能如定时器模块或 A/D 转换器有效，并且产生中断时，系统会暂时中止当前的程序而转到执行相对应的中断服务程序。该系列单片机提供多个外部中断和内部中断功能，外部中断由 INT0~INT3 引脚动作产生，而内部中断由各种内部功能，如定时器模块、时基、LVD、EEPROM、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。寄存器总的分为三类。第一类是 INTC0~INTC2 寄存器，用于设置基本的中断；第二类是 MFI0~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	请求标志位	注释
总中断	EMI	—	—
INTn 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	—
TM	TnPE	TnPF	n=0~3
	TnAE	TnAF	

注：EEPROM 中断仅存在于 HT67F489 单片机。

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
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

注：EEPROM 中断仅存在于 HT67F489 单片机。

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 中断请求标志位
0：无请求
1：中断请求

Bit 5 **INT1F**：INT1 中断请求标志位
0：无请求
1：中断请求

Bit 4 **INT0F**：INT0 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 中断控制位
0：除能
1：使能

Bit 2 **INT1E**：INT1 中断控制位
0：除能
1：使能

Bit 1 **INT0E**：INT0 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 中断请求标志位
0：无请求
1：中断请求

Bit 5 **MF2F**：多功能中断 2 中断请求标志位
0：无请求
1：中断请求

Bit 4 **MF1F**：多功能中断 1 中断请求标志位
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 中断控制位
0：除能
1：使能

Bit 1 **MF2E**：多功能中断 2 中断控制位
0：除能
1：使能

Bit 0 **MF1E**：多功能中断 1 中断控制位
0：除能
1：使能

INTC2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	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 中断请求标志位
0：无请求
1：中断请求

Bit 6 **INT3F**：INT3 中断请求标志位
0：无请求
1：中断请求

Bit 5 **INT2F**：INT2 中断请求标志位
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 中断控制位
0：除能
1：使能

Bit 2 **INT3E**：INT3 中断控制位
0：除能
1：使能

Bit 1 **INT2E**：INT2 中断控制位
0：除能
1：使能

Bit 0 **UARE**：UART 中断控制位
0：除能
1：使能

MFI0 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

MFI1 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

MFI2 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

MFI3 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

MFI4 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 1 中断请求标志位
0：无请求
1：中断请求

Bit 6 **TB0F**：时基 0 中断请求标志位
0：无请求
1：中断请求

Bit 5 **DEF**：数据 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 1 中断控制位
0：除能
1：使能

Bit 2 **TB0E**：时基 0 中断控制位
0：除能
1：使能

Bit 1 **DEE**：数据 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

注：EEPROM 中断仅存在于 HT67F489 单片机。

中断操作

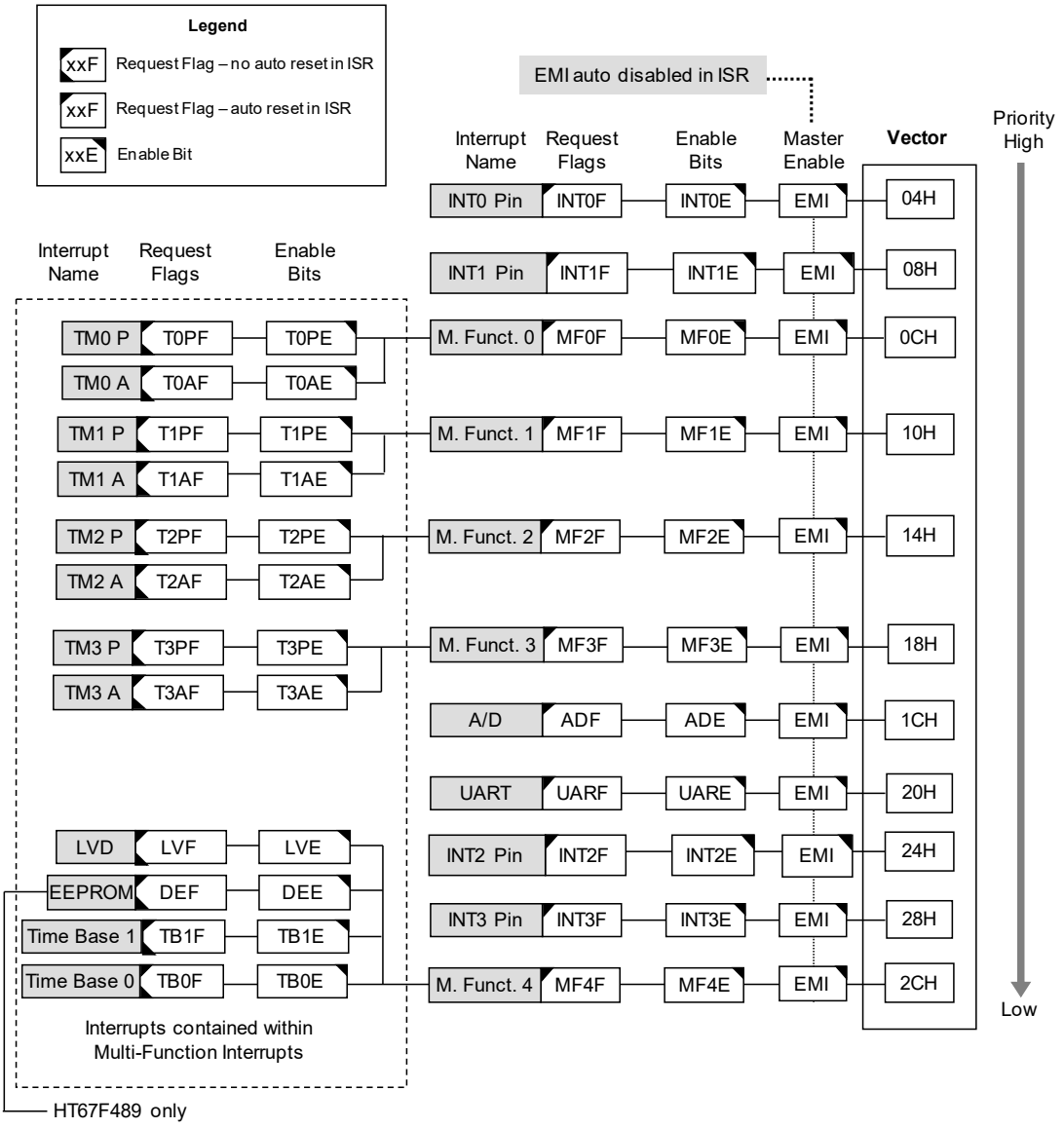
若中断事件条件产生，如一个 TM 比较器 P、比较器 A 匹配或 A/D 转换结束等等，相关中断请求标志将置起。中断标志产生后程序是否会跳转至相关中断向量执行是由中断使能位的条件决定的。若使能位为“1”，程序将跳至相关中断向量中执行；若使能位为“0”，即使中断请求标志置起中断也不会发生，程序也不会跳转至相关中断向量执行。若总中断使能位为“0”，所有中断都将除能。

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

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

如果某个中断服务子程序正在执行时，有另一个中断要求立即响应，那么 EMI 位应在程序进入中断子程序后置位，以允许此中断嵌套。如果堆栈已满，即使

此中断使能，中断请求也不会被响应，直到 SP 减少为止。如果要求立刻动作，则堆栈必须避免成为储满状态。请求同时发生时，执行优先级如下流程图所示。所有被置起的中断请求标志都可把单片机从休眠或空闲模式中唤醒，若要防止唤醒动作发生，在单片机进入休眠或空闲模式前应将相应的标志置起。



中断结构

外部中断

通过 INT0~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，此引脚将被作为外部中断脚使用。此时该引脚必须通过设置控制寄存器，将该引脚设置为输入口。当中断使能，堆栈未满并且外部中断脚状态改变，将调用外部中断向量子程序。当响应外部中断服务子程序时，中断请求标志位 INT0F~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。

多功能中断

该系列单片机中有多达四个多功能中断，与其它中断不同，它没有独立源，但由其它现有的中断源构成，即 TM 中断、LVD 中断、EEPROM 中断和时基中断。

当多功能中断中任何一种中断请求标志 MF0F~MF4F 被置位，多功能中断请求产生。当中断使能，堆栈未满，包括在多功能中断中的任意一个中断发生时，将调用多功能中断向量中的一个子程序。当响应中断服务子程序时，相关的多功能请求标志位会自动复位且 EMI 位会自动清零以除能其它中断。

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

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被置位，即 A/D 转换过程完成时，中断请求发生。当总中断使能位 EMI 和 A/D 中断使能位 ADE 被置位，允许程序跳转到相应的中断向量地址。当中断使能，堆栈未满且 A/D 转换动作结束时，将调用 A/D 中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 ADF 会自动清零。EMI 位也会被清零以除能其它中断。

UART 中断

UART 模块中，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 引脚唤醒都会触发产生 UART 中断。当整体 UART 中断请求标志位被置位，即以上任何一种情况发生时，中断请求发生。当总中断使能位 EMI 和 UART 中断使能位 UARE 被置位，允许程序跳转到相应的中断向量地址。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 会自动清零。EMI 位也会被清零以除能其它中断。而 UART 模块中的只读中断标志位仅在 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。当总中断使能位 EMI、时基使能位 TB0E 或 TB1E 和相应的多功能中断使能位被置位，允许程序跳转到各自的中断向量地址。当中断使能，堆栈未满且时基溢出时，将调用它们各自的中断向量子程序。当响应中断服务子程序时，EMI 位会被清零以除能其它中断，多功能中断请求标志也可自动清除，但相应的中断请求标志位 TB0F 或 TB1F 需在应用程序中清零。

时基中断的目的是提供一个固定周期的中断信号，时钟源来自内部时钟源 f_{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

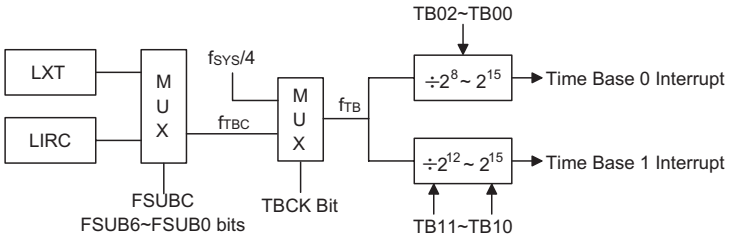
Bit 7 **TBON**：TB0 和 TB1 控制位
0：除能
1：使能

Bit 6 **TBCK**：选择 f_{TB} 时钟位
0：f_{TBC}
1：$f_{SYS}/4$

Bit 5~4 **TB11~TB10**：选择时基 1 溢出周期位
00：$4096/f_{TB}$
01：$8192/f_{TB}$
10：$16384/f_{TB}$
11：$32768/f_{TB}$

Bit 3 未定义，读为“0”

Bit 2~0 **TB02~TB00**：选择时基 0 溢出周期位
000：$256/f_{TB}$
001：$512/f_{TB}$
010：$1024/f_{TB}$
011：$2048/f_{TB}$
100：$4096/f_{TB}$
101：$8192/f_{TB}$
110：$16384/f_{TB}$
111：$32768/f_{TB}$



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子程序中执行。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子程序中执行。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

周期型和简易型 TM 都有两个中断。所有的 TM 中断也属于多功能中断。周期型和简易型 TM 都有两个中断请求标志位 TnPF、TnAF 及两个使能位 TnPE、TnAE。当 TM 比较器 P 或 A 匹配情况发生时，任意 TM 中断请求标志被置位，TM 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、相应 TM 中断使能位和相关多功能中断使能位 MFnE 需先被置位。当中断使能，堆栈未满且 TM 比较器匹配情况发生时，可跳转至相关多功能中断向量子程序中执行。当 TM 中断响应，EMI 将被自动清零以除能其它中断，相关 MFnF 标志也可自动清除，但 TM 中断请求标志需在应用程序中手动清除。

中断唤醒功能

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

编程注意事项

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

多功能中断中所含中断相应程序执行时，多功能中断请求标志 MF0F~MF4F 可以自动清零，但各自的请求标志需在应用程序中手动清除。

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

所有中断在休眠或空闲模式下都具有唤醒功能，当中断请求标志发生由低到高

的转变时都可产生唤醒功能。若要避免相应中断产生唤醒动作，在单片机进入休眠或空闲模式前需先将相应请求标志置为高。

当进入中断服务程序，系统仅将程序计数器的内容压入堆栈，如果中断服务程序会改变状态寄存器或其它的寄存器的内容而破坏控制流程，应事先将这些数据保存起来。

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

低电压检测 – LVD

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

LVD 寄存器

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

LVDC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	—	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

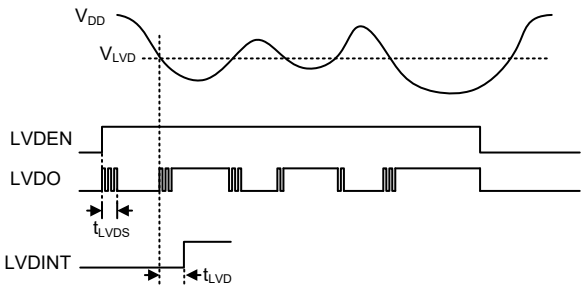
Bit 3　未定义，读为“0”

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

LVD 操作

通过比较电源电压 V_{DD} 与存储在 LVDC 寄存器中的预置电压值的结果，低电压检测功能工作。其设置的范围为 2.0V~4.0V。当电源电压 V_{DD} 低于预置电压值

时，LVDO 位被置为高，表明低电压产生。低电压检测功能由一个自动使能的参考电压提供。若 LVDEN 位为高，当单片机掉电时低电压检测器保持有效状态。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。若 LVDEN 位为高，当单片机掉电时低电压检测器保持有效状态。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。

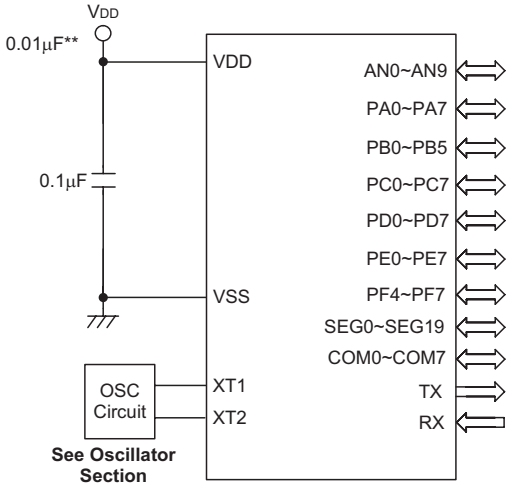
当 LVD 功能使能时，建议先清除 LVD 标志位，在使能中断功能以避免错误动作。

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


VDD
0.01μF**
VDD
0.1μF
VSS
OSC
Circuit
XT1
XT2
See Oscillator
Section
AN0~AN9
PA0~PA7
PB0~PB5
PC0~PC7
PD0~PD7
PE0~PE7
PF4~PF7
SEG0~SEG19
COM0~COM7
TX
RX

指令集

简介

任何单片机成功运作的核心在于它的指令集，此指令集为一组程序指令码，用来指导单片机如何去执行指定的工作。在 HOLTEK 单片机中，提供了丰富且灵活的指令，共超过六十条，程序设计者可以事半功倍地实现它们的应用。

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

指令周期

大部分的操作均只需要一个指令周期来执行。分支、调用或查表则需要两个指令周期。一个指令周期相当于四个系统时钟周期，因此如果在 8MHz 的系统时钟振荡器下，大部分的操作将在 0.5μs 中执行完成，而分支或调用操作则将在 1μs 中执行完成。虽然需要两个指令周期的指令通常指的是 JMP、CALL、RET、RETI 和查表指令，但如果牵涉到程序计数器低字节寄存器 PCL 也将多花费一个周期去加以执行。即指令改变 PCL 的内容进而导致直接跳转至新地址时，需要多一个周期去执行，例如“CLR PCL”或“MOV PCL, A”指令。对于跳转指令必须注意的是，如果比较的结果牵涉到跳转动作将多花费一个周期，如果没有则需一个周期即可。

数据的传送

单片机程序中数据传送是使用最为频繁的操作之一，使用三种 MOV 的指令，数据不但可以从寄存器转移至累加器 (反之亦然)，而且能够直接移动立即数到累加器。数据传送最重要的应用之一是从输入端口接收数据或传送数据到输出端口。

算术运算

算术运算和数据处理是大部分单片机应用所必需具备的能力，在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, x	ACC 与立即数、进位标志相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SBC A,[m]	ACC 与数据存储器、进位标志相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	1 注	Z, C, AC, OV, SC, CZ
DAA [m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	1 注	C
逻辑运算			
AND A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	1	Z
OR A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	1	Z
XOR A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	1	Z
ANDM A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	1 注	Z
ORM A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	1 注	Z
XORM A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	1 注	Z
AND A, x	ACC 与立即数做“与”运算，结果放入 ACC	1	Z
OR A, x	ACC 与立即数做“或”运算，结果放入 ACC	1	Z
XOR A, x	ACC 与立即数做“异或”运算，结果放入 ACC	1	Z
CPL [m]	对数据存储器取反，结果放入数据存储器	1 注	Z
CPLA [m]	对数据存储器取反，结果放入 ACC	1	Z
递增和递减			
INCA [m]	递增数据存储器，结果放入 ACC	1	Z
INC [m]	递增数据存储器，结果放入数据存储器	1 注	Z
DECA [m]	递减数据存储器，结果放入 ACC	1	Z
DEC [m]	递减数据存储器，结果放入数据存储器	1 注	Z

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

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

注：1. 对跳转指令而言，如果比较的结果牵涉到跳转即需 2 个周期，如果没有发生跳转，则只需一个周期。
2. 任何指令若要改变 PCL 的内容将需要 2 个周期来执行。

扩展指令集

扩展指令用来提供更大范围的数据存储器寻址。当被存取的数据存储器位于 Sector 0 之外的任何数据存储器 Sector，扩展指令可直接存取数据存储器而无需使用间接寻址，此举不仅可节省 Flash 存储器空间的使用，同时可提高 CPU 执行效率。

助记符		说明	指令周期	影响标志位
算术运算				
LADD	A,[m]	ACC 与数据存储器相加，结果放入 ACC	2	Z, C, AC, OV, SC
LADDM	A,[m]	ACC 与数据存储器相加，结果放入数据存储器	2注	Z, C, AC, OV, SC
LADC	A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	2	Z, C, AC, OV, SC
LADCM	A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	2注	Z, C, AC, OV, SC
LSUB	A,[m]	ACC 与数据存储器相减，结果放入 ACC	2	Z, C, AC, OV, SC, CZ
LSUBM	A,[m]	ACC 与数据存储器相减，结果放入数据存储器	2注	Z, C, AC, OV, SC, CZ
LSBC	A,[m]	ACC 与数据存储器、进位标志相减，结果放入 ACC	2	Z, C, AC, OV, SC, CZ
LSBCM	A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	2注	Z, C, AC, OV, SC, CZ
LDAA	[m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	2注	C
逻辑运算				
LAND	A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	2	Z
LOR	A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	2	Z
LXOR	A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	2	Z
LANDM	A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	2注	Z
LORM	A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	2注	Z
LXORM	A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	2注	Z
LCPL	[m]	对数据存储器取反，结果放入数据存储器	2注	Z
LCPLA	[m]	对数据存储器取反，结果放入 ACC	2	Z
递增和递减				
LINCA	[m]	递增数据存储器，结果放入 ACC	2	Z
LINC	[m]	递增数据存储器，结果放入数据存储器	2注	Z
LDECA	[m]	递减数据存储器，结果放入 ACC	2	Z
LDEC	[m]	递减数据存储器，结果放入数据存储器	2注	Z
移位				
LRRA	[m]	数据存储器右移一位，结果放入 ACC	2	无
LRR	[m]	数据存储器右移一位，结果放入数据存储器	2注	无
LRRCA	[m]	带进位将数据存储器右移一位，结果放入 ACC	2	C
LRRC	[m]	带进位将数据存储器右移一位，结果放入数据存储器	2注	C
LRLA	[m]	数据存储器左移一位，结果放入 ACC	2	无
LRL	[m]	数据存储器左移一位，结果放入数据存储器	2注	无
LRLCA	[m]	带进位将数据存储器左移一位，结果放入 ACC	2	C
LRLC	[m]	带进位将数据存储器左移一位，结果放入数据存储器	2注	C
数据传送				
LMOV	A,[m]	将数据存储器送至 ACC	2	无
LMOV	[m],A	将 ACC 送至数据存储器	2注	无

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

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

指令定义

ADC A, [m] Add Data Memory to ACC with Carry
指令说明 将指定的数据存储器、累加器内容以及进位标志相加，结果存放到累加器。
功能表示 ACC ← ACC + [m] + C
影响标志位 OV、Z、AC、C、SC

ADCM A, [m] Add ACC to Data Memory with Carry
指令说明 将指定的数据存储器、累加器内容和进位标志位相加，结果存放到指定的数据存储器。
功能表示 [m] ←ACC + [m] + C
影响标志位 OV、Z、AC、C、SC

ADD A, [m] Add Data Memory to ACC
指令说明 将指定的数据存储器和累加器内容相加，结果存放到累加器。
功能表示 ACC ← ACC + [m]
影响标志位 OV、Z、AC、C、SC

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

ADDM A, [m] Add ACC to Data Memory
指令说明 将指定的数据存储器和累加器内容相加，结果存放到指定的数据存储器。
功能表示 [m] ←ACC + [m]
影响标志位 OV、Z、AC、C、SC

AND A, [m] Logical AND Data Memory to ACC
指令说明 将累加器中的数据和指定数据存储器内容做逻辑与，结果存放到累加器。
功能表示 ACC ← ACC“AND”[m]
影响标志位 Z

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

ANDM A, [m]	Logical AND ACC to Data Memory
指令说明	将指定数据存储器内容和累加器中的数据做逻辑与，结果存放到数据存储器。
功能表示	[m] ← ACC“AND”[m]
影响标志位	Z

CALL addr	Subroutine call
指令说明	无条件地调用指定地址的子程序，此时程序计数器先加 1 获得下一个要执行的指令地址并压入堆栈，接着载入指定地址并从新地址继续执行程序，由于此指令需要额外的运算，所以为一个 2 周期的指令。
功能表示	Stack ← Program Counter + 1 Program Counter ← addr
影响标志位	无

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

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

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

CPL [m] Complement Data Memory
指令说明 将指定数据存储器中的每一位取逻辑反，
相当于从 1 变 0 或 0 变 1。
功能表示 [m] ← [$\overline{m}$]
影响标志位 Z

CPLA [m] Complement Data Memory with result in ACC
指令说明 将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1，而结果被储存回累加器且数据存储器中的内容不变。
功能表示 ACC←[$\overline{m}$]
影响标志位 Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory
指令说明 将累加器中的内容转换为 BCD (二进制转成十进制) 码。如果低四位的值大于“9”或 AC=1，那么 BCD 调整就执行对原值加“6”，否则原值保持不变；如果高四位的值大于“9”或 C=1，那么 BCD 调整就执行对原值加“6”。BCD 转换实质上是根据累加器和标志位执行 00H，06H，60H 或 66H 的加法运算，结果存放到数据存储器。只有进位标志位 C 受影响，用来指示原始 BCD 的和是否大于 100，并可以进行双精度十进制数的加法运算。
功能表示 [m] ← ACC + 00H 或
[m] ← ACC + 06H 或
[m] ← ACC + 60H 或
[m] ← ACC + 66H
影响标志位 C

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

DECA [m] Decrement Data Memory with result in ACC
指令说明 将指定数据存储器的内容减 1，把结果存放回累加器并保持指定数据存储器的内容不变。
功能表示 ACC ← [m] – 1
影响标志位 Z

HALT	Enter power down mode
指令说明	此指令终止程序执行并关掉系统时钟，RAM 和寄存器的内容保持原状态，WDT 计数器和分频器被清“0”，暂停标志位 PDF 被置位 1，WDT 溢出标志位 TO 被清 0。
功能表示	TO ← 0 PDF ← 1
影响标志位	TO、PDF

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

INCA [m]	Increment Data Memory with result in ACC
指令说明	将指定数据存储器的内容加 1，结果存放回累加器并保持指定的数据存储器内容不变。
功能表示	ACC ← [m] + 1
影响标志位	Z

JMP addr	Jump unconditionally
指令说明	程序计数器的内容无条件地由被指定的地址取代，程序由新的地址继续执行。当新的地址被加载时，必须插入一个空指令周期，所以此指令为 2 个周期的指令。
功能表示	Program Counter ← addr
影响标志位	无

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

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

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

NOP No operation
指令说明 空操作，接下来顺序执行下一条指令。
功能表示 无操作
影响标志位 无

OR A, [m] Logical OR Data Memory to ACC
指令说明 将累加器中的数据和指定的数据存储器内容逻辑或，
结果存放到累加器。
功能表示 ACC ← ACC“OR”[m]
影响标志位 Z

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

ORM A, [m] Logical OR ACC to Data Memory
指令说明 将存在指定数据存储器中的数据和累加器逻辑或，
结果放到数据存储器。
功能表示 [m] ← ACC“OR”[m]
影响标志位 Z

RET Return from subroutine
指令说明 将堆栈寄存器中的程序计数器值恢复，
程序由取回的地址继续执行。
功能表示 Program Counter←Stack
影响标志位 无

RET A, x Return from subroutine and load immediate data to ACC
指令说明 将堆栈寄存器中的程序计数器值恢复且累加器载入指定的
立即数，程序由取回的地址继续执行。
功能表示 Program Counter ← Stack
ACC←x
影响标志位 无

RETI	Return from interrupt
指令说明	将堆栈寄存器中的程序计数器值恢复且中断功能通过设置 EMI 位重新使能。EMI 是控制中断使能的主控制位。如果在执行 RETI 指令之前还有中断未被响应，则这个中断将在返回主程序之前被响应。
功能表示	Program Counter ←Stack EMI ← 1
影响标志位	无

RL [m]	Rotate Data Memory left
指令说明	将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。
功能表示	[m].(i+1) ← [m].i (i=0~6) [m].0 ← [m].7
影响标志位	无

RLA [m]	Rotate Data Memory left with result in ACC
指令说明	将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位，结果送到累加器，而指定数据存储器的内容保持不变。
功能表示	ACC.(i+1) ← [m].i (i=0~6) ACC.0 ←[m].7
影响标志位	无

RLC [m]	Rotate Data Memory Left through Carry
指令说明	将指定数据存储器的内容连同进位标志左移 1 位，第 7 位取代进位标志且原本的进位标志移到第 0 位。
功能表示	[m].(i+1) ← [m].i (i=0~6) [m].0 ← C C ← [m].7
影响标志位	C

RLC A [m]	Rotate Data Memory left through Carry with result in ACC
指令说明	将指定数据存储器的内容连同进位标志左移 1 位，第 7 位取代进位标志且原本的进位标志移到第 0 位，移位结果送回累加器，但是指定数据寄存器的内容保持不变。
功能表示	ACC.(i+1) ← [m].i (i=0~6) ACC.0 ← C C ← [m].7
影响标志位	C

RR [m] Rotate Data Memory right
指令说明 将指定数据存储器的内容循环右移 1 位且第 0 位移到第 7 位。
功能表示 [m].i ← [m].(i+1) (i=0~6)
[m].7 ← [m].0
影响标志位 无

RRA [m] Rotate Data Memory right with result in ACC
指令说明 将指定数据存储器的内容循环右移 1 位，第 0 位移到第 7 位，移位结果存放到累加器，而指定数据存储器的内容保持不变。
功能表示 ACC.i ← [m].(i+1) (i=0~6)
ACC.7 ← [m].0
影响标志位 无

RRC [m] Rotate Data Memory right through Carry
指令说明 将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位。
功能表示 [m].i ← [m].(i+1) (i=0~6)
[m].7← C
C ← [m].0
影响标志位 C

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

SBC A, [m] Subtract Data Memory from ACC with Carry
指令说明 将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – [m] – $\overline{C}$
影响标志位 OV、Z、AC、C、SC、CZ

SBC A, x Subtract immediate data from ACC with Carry
指令说明 将累加器减去立即数以及进位标志的反，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – [m] – $\overline{C}$
影响标志位 OV、Z、AC、C、SC、CZ

SBCM A, [m] Subtract Data Memory from ACC with Carry and result in Data Memory
指令说明 将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 [m] ← ACC – [m] – $\overline{C}$
影响标志位 OV、Z、AC、C、SC、CZ

SDZ [m] Skip if Decrement Data Memory is 0
指令说明 将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 [m] ← [m] – 1，如果 [m]=0 跳过下一条指令执行
影响标志位 无

SDZA [m] Skip if decrement Data Memory is zero with result in ACC
指令说明 将指定数据存储器内容减 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果将存放到累加器，但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 ACC ← [m] – 1，如果 ACC=0 跳过下一条指令执行
影响标志位 无

SET [m] Set Data Memory
指令说明 将指定数据存储器的每一位设置为 1。
功能表示 [m] ← FFH
影响标志位 无

SET [m].i Set bit of Data Memory
指令说明 将指定数据存储器的第 i 位置位为 1。
功能表示 [m].i ← 1
影响标志位 无

SIZ [m] Skip if increment Data Memory is 0
指令说明 将指定的数据存储器的内容加 1，判断是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 [m] ←[m] + 1，如果 [m]=0 跳过下一条指令执行
影响标志位 无

SIZA [m] Skip if increment Data Memory is zero with result in ACC
指令说明 将指定数据存储器的内容加 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果会被存放到累加器，但是指定数据存储器的内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 ACC ←[m] + 1，如果 ACC=0 跳过下一条指令执行
影响标志位 无

SNZ [m].i Skip if bit i of Data Memory is not 0
指令说明 判断指定数据存储器的第 i 位，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果为 0，则程序继续执行下一条指令。
功能表示 如果 [m].i≠0，跳过下一条指令执行
影响标志位 无

SNZ [m] Skip if Data Memory is not 0
指令说明 指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定存储器，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为2 个周期的指令。如果结果为0，则程序继续执行下一条指令。
功能表示 如果 [m]≠0，跳过下一条指令执行
影响标志位 无

SUB A, [m] Subtract Data Memory from ACC

指令说明 将累加器的内容减去指定的数据存储器的数据，把结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。

功能表示 ACC ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

SUBM A, [m] Subtract Data Memory from ACC with result in Data Memory

指令说明 将累加器的内容减去指定数据存储器的数据，结果存放到指定的数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。

功能表示 [m] ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

SUB A, x Subtract immediate Data from ACC

指令说明 将累加器的内容减去立即数，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。

功能表示 ACC ← ACC – x

影响标志位 OV、Z、AC、C、SC、CZ

SWAP [m] Swap nibbles of Data Memory

指令说明 将指定数据存储器的低 4 位和高 4 位互相交换。

功能表示 [m].3~[m].0 ↔ [m].7~[m].4

影响标志位 无

SWAPA [m] Swap nibbles of Data Memory with result in ACC

指令说明 将指定数据存储器的低 4 位与高 4 位互相交换，再将结果存放到累加器且指定数据寄存器的数据保持不变。

功能表示 ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

影响标志位 无

SZ [m] Skip if Data Memory is 0

指令说明 指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定数据存储器的内容是否为 0，若为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 如果 [m]=0，跳过下一条指令执行

影响标志位 无

SZA [m] Skip if Data Memory is 0 with data movement to ACC

指令说明 将指定数据存储器内容复制到累加器，并判断指定数据存储器的内容是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 ACC ←[m]，如果 [m]=0，跳过下一条指令执行

影响标志位 无

SZ [m].i Skip if bit i of Data Memory is 0

指令说明 判断指定数据存储器的第 i 位是否为 0，若为 0，则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 如果 [m].i=0，跳过下一条指令执行

影响标志位 无

TABRD [m] Read table (specific page) to TBLH and Data Memory

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

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

影响标志位 无

TABRDL [m] Read table (last page) to TBLH and Data Memory

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

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

影响标志位 无

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

指令说明 自加表格指针低字节 TBLP，将表格指针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定的数据存储器且将高字节移至 TBLH。

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

影响标志位 无

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

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

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

影响标志位 无

XOR A, [m] Logical XOR Data Memory to ACC

指令说明 将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。

功能表示 ACC ← ACC“XOR”[m]

影响标志位 Z

XORM A, [m] Logical XOR ACC to Data Memory

指令说明 将累加器的数据和指定的数据存储器内容逻辑异或，结果放到数据存储器。

功能表示 [m] ← ACC“XOR”[m]

影响标志位 Z

XOR A, x Logical XOR immediate data to ACC

指令说明 将累加器的数据与立即数逻辑异或，结果存放到累加器。

功能表示 ACC ← ACC“XOR”x

影响标志位 Z

扩展指令定义

扩展指令被用来直接存取存储在任何数据存储器 Sector 中的数据。

LADC A, [m] Add Data Memory to ACC with Carry
指令说明 将指定的数据存储器、累加器内容以及进位标志相加，结果存放到累加器。
功能表示 ACC ← ACC + [m] + C
影响标志位 OV、Z、AC、C、SC

LADCM A, [m] Add ACC to Data Memory with Carry
指令说明 将指定的数据存储器、累加器内容和进位标志位相加，结果存放到指定的数据存储器。
功能表示 [m] ←ACC + [m] + C
影响标志位 OV、Z、AC、C、SC

LADD A, [m] Add Data Memory to ACC
指令说明 将将指定的数据存储器和累加器内容相加，结果存放到累加器。
功能表示 ACC ← ACC + [m]
影响标志位 OV、Z、AC、C、SC

LADDM A, [m] Add ACC to Data Memory
指令说明 将指定的数据存储器和累加器内容相加，结果存放到指定的数据存储器。
功能表示 [m] ←ACC + [m]
影响标志位 OV、Z、AC、C、SC

LAND A, [m] Logical AND Data Memory to ACC
指令说明 将累加器中的数据和指定数据存储器内容做逻辑与，结果存放到累加器。
功能表示 ACC ← ACC“AND”[m]
影响标志位 Z

LANDM A, [m] Logical AND ACC to Data Memory
指令说明 将指定数据存储器内容和累加器中的数据做逻辑与，结果存放到数据存储器。
功能表示 [m] ← ACC“AND”[m]
影响标志位 Z

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

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

LCPL [m] Complement Data Memory
指令说明 将指定数据存储器中的每一位取逻辑反，
相当于从 1 变 0 或 0 变 1。
功能表示 [m] ← $[\overline{m}]$
影响标志位 Z

LCPLA [m] Complement Data Memory with result in ACC
指令说明 将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1，结果被存放回累加器且数据寄存器的内容保持不变。
功能表示 ACC←$[\overline{m}]$
影响标志位 Z

LDAA [m] Decimal-Adjust ACC for addition with result in Data Memory
指令说明 将累加器中的内容转换为 BCD (二进制转成十进制) 码。如果低四位的值大于“9”或 AC=1，那么 BCD 调整就执行对低四位加“6”，否则低四位保持不变；如果高四位的值大于“9”或C=1，那么BCD调整就执行对高四位加“6”。BCD 转换实质上是根据累加器和标志位执行 00H，06H，60H 或 66H 的加法运算，结果存放到数据存储器。只有进位标志位 C 受影响，用来指示原始 BCD 的和是否大于 100，并可以进行双精度十进制数的加法运算。
功能表示 [m] ← ACC + 00H 或
[m] ← ACC + 06H 或
[m] ← ACC + 60H 或
[m] ← ACC + 66H
影响标志位 C

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

LDECA [m] Decrement Data Memory with result in ACC
指令说明 将指定数据存储器的内容减 1，把结果存放回累加器并保持指定数据存储器的内容不变。
功能表示 ACC ← [m] – 1
影响标志位 Z

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

LINCA [m] Increment Data Memory with result in ACC
指令说明 将指定数据存储器的内容加 1，结果存放回累加器并保持指定的数据存储器内容不变。
功能表示 ACC ← [m] + 1
影响标志位 Z

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

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

LOR A, [m] Logical OR Data Memory to ACC
指令说明 将累加器中的数据和指定的数据存储器内容逻辑或，结果存放到累加器。
功能表示 ACC ← ACC“OR”[m]
影响标志位 Z

LORM A, [m] Logical OR ACC to Data Memory
指令说明 将存在指定数据存储器中的数据和累加器逻辑或，结果放到数据存储器。
功能表示 [m] ← ACC "OR" [m]
影响标志位 Z

LRL [m] Rotate Data Memory left
指令说明 将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。
功能表示 [m].(i+1) ← [m].i (i=0~6)
[m].0 ← [m].7
影响标志位 无

LRLA [m] Rotate Data Memory left with result in ACC
指令说明 将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位，结果送到累加器，而指定数据存储器的内容保持不变。
功能表示 ACC.(i+1) ← [m].i (i=0~6)
ACC.0 ←[m].7
影响标志位 无

LRLC [m] Rotate Data Memory Left through Carry
指令说明 将指定数据存储器的内容连同进位标志左移 1 位，第 7 位取代进位标志且原本的进位标志移到第 0 位。
功能表示 [m].(i+1) ← [m].i (i=0~6)
[m].0 ← C
C ← [m].7
影响标志位 C

LRLC A [m] Rotate Data Memory left through Carry with result in ACC
指令说明 将指定数据存储器的内容连同进位标志左移 1 位，第 7 位取代进位标志且原本的进位标志移到第 0 位，移位结果送回累加器，但是指定数据寄存器的内容保持不变。
功能表示 ACC.(i+1) ← [m].i (i=0~6)
ACC.0 ← C
C ← [m].7
影响标志位 C

LRR [m] Rotate Data Memory right
指令说明 将指定数据存储器的内容循环右移 1 位且第 0 位移到第 7 位。
功能表示 [m].i ← [m].(i+1) (i=0~6)
[m].7 ← [m].0
影响标志位 无

LRRA [m] Rotate Data Memory right with result in ACC
指令说明 将指定数据存储器的内容循环右移 1 位，第 0 位移到第 7 位，移位结果存放到累加器，而指定数据存储器的内容保持不变。
功能表示 ACC.i ← [m].(i+1) (i=0~6)
ACC.7 ← [m].0
影响标志位 无

LRRC [m] Rotate Data Memory right through Carry
指令说明 将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位。
功能表示 [m].i ← [m].(i+1) (i=0~6)
[m].7← C
C ← [m].0
影响标志位 C

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

LSBC A, [m] Subtract Data Memory from ACC with Carry
指令说明 将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – [m] – $\overline{C}$
影响标志位 OV、Z、AC、C、SC、CZ

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

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

功能表示 [m] ← ACC – [m] – $\overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

LSDZ [m] Skip if Decrement Data Memory is 0

指令说明 将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 [m] ← [m] – 1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

LSDZA [m] Skip if decrement Data Memory is zero with result in ACC

指令说明 将指定数据存储器内容减 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果将存放到累加器，但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 ACC ← [m] – 1，如果 ACC=0 跳过下一条指令执行

影响标志位 无

LSET [m] Set Data Memory

指令说明 将指定数据存储器的每一个位置位为 1。

功能表示 [m] ← FFH

影响标志位 无

LSET [m].i Set bit of Data Memory

指令说明 将指定数据存储器的第 i 位置位为 1。

功能表示 [m].i ← 1

影响标志位 无

LSIZ [m] Skip if increment Data Memory is 0

指令说明 将指定的数据存储器的内容加 1，判断是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 [m] ←[m] + 1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

LSIZA [m] Skip if increment Data Memory is zero with result in ACC

指令说明 将指定数据存储器的内容加 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果会被存放到累加器，但是指定数据存储器的内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 ACC ←[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 3 个周期的指令。如果结果为 0，则程序继续执行下一条指令。

功能表示 如果 [m].i≠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 指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定数据存储器，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果为 0，则程序继续执行下一条指令。

功能表示 如果 [m]≠0，跳过下一条指令执行

影响标志位 无

LSUB A, [m] Subtract Data Memory from ACC

指令说明 将累加器的内容减去指定的数据存储器的数据，把结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。

功能表示 ACC ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

LSUBM A, [m] Subtract Data Memory from ACC with result in Data Memory
指令说明 将累加器的内容减去指定数据存储器的数据，结果存放到指定的数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 [m] ← ACC – [m]
影响标志位 OV、Z、AC、C、SC、CZ

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

LSWAPA [m] Swap nibbles of Data Memory with result in ACC
指令说明 将指定数据存储器的低 4 位和高 4 位互相交换，再将结果存放到累加器且指定数据寄存器的数据保持不变。
功能表示 ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0
影响标志位 无

LSZ [m] Skip if Data Memory is 0
指令说明 指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定数据存储器的内容是否为 0，若为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 如果 [m]=0，跳过下一条指令执行
影响标志位 无

LSZA [m] Skip if Data Memory is 0 with data movement to ACC
指令说明 将指定数据存储器内容复制到累加器，并判断指定数据存储器的内容是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 ACC ← [m]，如果 [m]=0，跳过下一条指令执行
影响标志位 无

LSZ [m].i Skip if bit i of Data Memory is 0
指令说明 判断指定数据存储器的第 i 位是否为 0，若为 0，则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 如果 [m].i=0，跳过下一条指令执行
影响标志位 无

LTABRD [m] Move the ROM code (specific page) to TBLH and data memory
指令说明 将表格指针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定数据存储器且将高字节移至 TBLH。
功能表示 [m] ← 程序代码 (低字节)
TBLH ← 程序代码 (高字节)
影响标志位 无

LTABRDL [m] Read table (last page) to TBLH and Data Memory
指令说明 将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定数据存储器且将高字节移至 TBLH。
功能表示 [m] ← 程序代码 (低字节)
TBLH ← 程序代码 (高字节)
影响标志位 无

LITABRD [m] Increment table pointer low byte first and read table (specific page) to TBLH and data memory
指令说明 自加表格指针低字节 TBLP，将表格指针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定的数据存储器且将高字节移至 TBLH。
功能表示 [m] ← 程序代码 (低字节)
TBLH ← 程序代码 (高字节)
影响标志位 无

LITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and data memory
指令说明 自加表格指针低字节 TBLP，将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定的数据存储器且将高字节移至 TBLH。
功能表示 [m] ← 程序代码 (低字节)
TBLH ← 程序代码 (高字节)
影响标志位 无

LXOR A, [m]	Logical XOR Data Memory to ACC
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。
功能表示	ACC ← ACC“XOR”[m]
影响标志位	Z

LXORM A, [m]	Logical XOR ACC to Data Memory
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果放到数据存储器。
功能表示	[m] ← ACC“XOR”[m]
影响标志位	Z

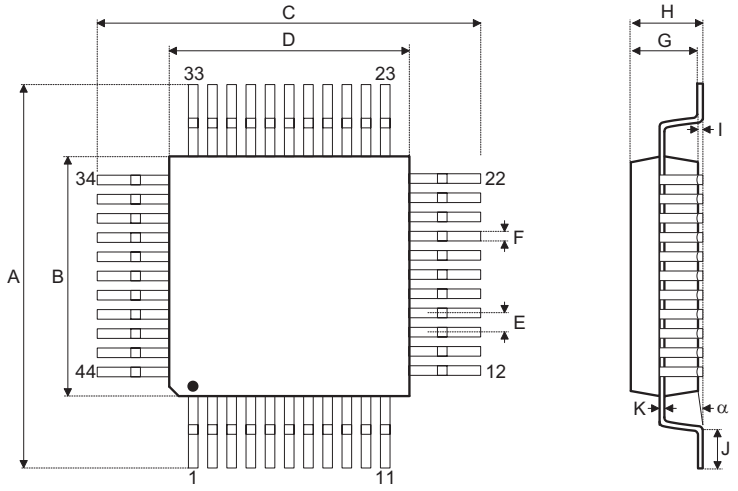
封装信息

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

封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

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

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



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

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