体脂秤 **Flash** 单片机

HT45F75

版本 : V1.10　日期 : 2016-12-07

www.holtek.com

目 录

特性

CPU 特性

- 工作电压：
  - ♦ f_{SYS}=8MHz：2.2V ~ 5.5V
  - ♦ f_{SYS}=12MHz：2.7V ~ 5.5V
  - ♦ f_{SYS}=20MHz：4.5V ~ 5.5V
- V_{DD}=5V，系统时钟为 20MHz 时，指令周期为 0.2μs
- 提供暂停和唤醒功能，以降低功耗
- 4 种振荡模式：
  - ♦ 外部高频晶振 – HXT
  - ♦ 外部 32.768kHz 晶振 – LXT
  - ♦ 内部高频 RC – HIRC
  - ♦ 内部 32kHz RC – LIRC
- 多种工作模式：正常、低速、空闲和休眠
- 内建 4.8MHz，4.8×2MHz 和 4.8×3MHz 振荡器，无需外部元件
- 所有指令都可在 1~3 个指令周期内完成
- 查表指令
- 115 条功能强大的指令系统
- 8 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器：4K×16
- RAM 数据存储器：256×8
- True EEPROM 存储器：64×8
- IAP 功能
- 看门狗定时器功能
- 27 个双向 I/O 口
- 2 个引脚与外部中断口共用
- 多个定时器模块用于时间测量、捕捉输入、比较匹配输出、PWM 输出及单脉冲输出
- 双时基功能，可提供固定时间的中断信号
- 2 组差分通道 20 位分辨精度的 Delta-Sigma 型 A/D 转换器
- 低电压复位功能
- 低电压检测功能
- 内建一组带旁路功能的 LDO，可提供电源给 PGA、A/D 转换器或外部传感器
- 串行接口模块 – SIM，用于 SPI 或 I^2C 通信
- 带 IR 载波的 UART 模块
- 内置体脂电路
- Flash 程序存储器烧录可达 100,000 次
- Flash 程序存储器数据可保存 10 年以上
- True EEPROM 数据存储器烧录可达 1,000,000 次
- True EEPROM 数据存储器数据可保存 10 年以上
- 封装类型：48-pin LQFP

概述

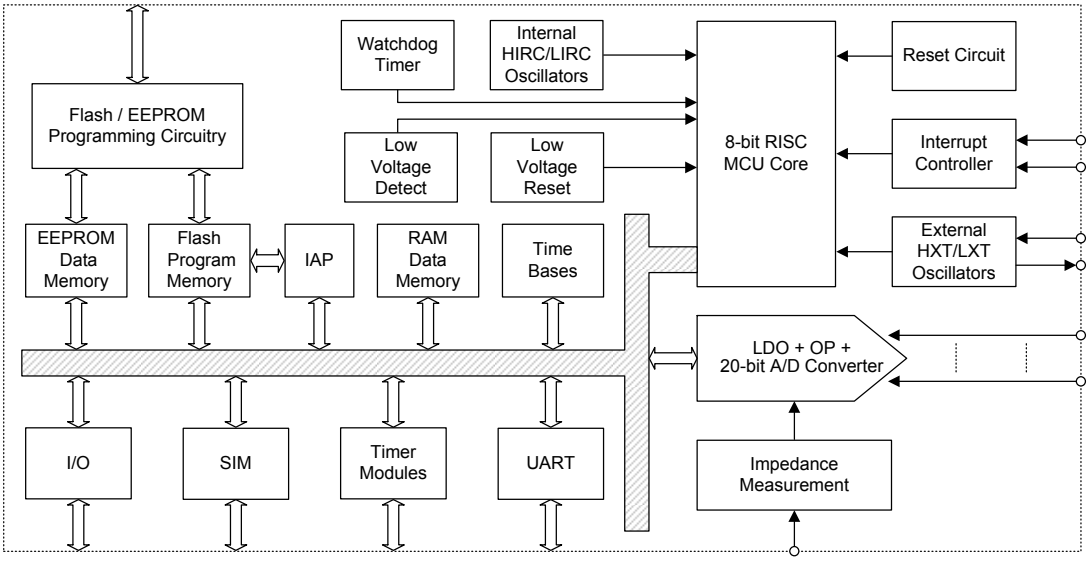
该款 Holtek 单片机是专为人体脂肪秤应用而设计的。体脂秤利用交流信号以电流形式通过人体，计算出人体脂肪值。体脂秤电路主要由测重电路和测脂电路组成，测重电路是通过外部称重传感器的输出信号，经 OPA 后将信号放大，再由 ADC 转换后读取相对应的值，计算体重大小；测脂电路是经由交流信号通过电极片让信号通过人体，利用内部 OPA 将信号放大，并由 ADC 转换后读取相对应的值，测出人体阻抗，最后计算得知相对应的人体脂肪值。

该单片机集成了体脂秤电路，是一款 A/D 型具有 8 位高性能精简指令集的 Flash 单片机，内置的多通道 20-bit Delta-Sigma 型 A/D（ΔΣA/D）转换器专门为需要具有低噪声和高准确度的 A/D 转换的产品而设计。该单片机具有一系列功能和特性，其 Flash 存储器可多次编程的特性给用户提供了极大的方便。存储器方面，还包含了一个 RAM 数据存储器和一个可用于存储序号、校准数据等非易失性数据的 True EEPROM 存储器。

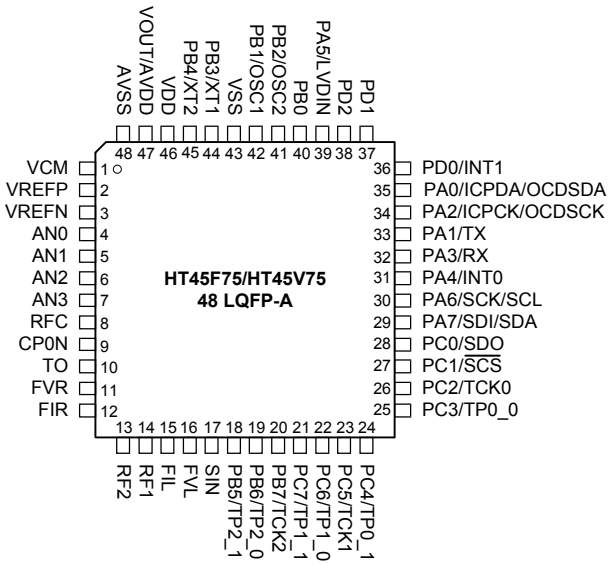
在模拟特性方面，该单片机包含一个多通道 20-bit ΔΣA/D 转换器和一个可编程增益放大器 PGA。还带有多个使用灵活的定时器模块，可提供定时功能、脉冲产生功能及 PWM 产生功能。另外，内部 LDO 功能为内部和外部设备提供了各种电源选项。内部看门狗定时器、低电压复位和低电压检测等内部保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地运行。

该单片机提供了多种内部和外部振荡器功能选项，且内建完整的系统振荡器，无需外接元件。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。

方框图



引脚图



注：1. 若共用脚同时有多种输出，“/”号右侧的引脚名具有更高的优先级。
2. HT45V75 是 HT45F75 的 EV 芯片，OCDSCK 和 OCDSDA 引脚仅存在于 OCDS EV 芯片。

引脚说明

每个引脚的功能如下表所述，而引脚配置的详细内容见规格书其它章节。

引脚名称	功能	OPT	I/T	O/T	说明
PA0/ICPDA/OCDSDA	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
	ICPDA	—	ST	CMOS	ICP 数据 / 地址
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址，仅用于 EV 芯片
PA1/TX	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
	TX	UCR1 UCR2	—	CMOS	UART 发送引脚
PA2/ICPCK/OCDSCK	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
	ICPCK	—	ST	—	ICP 时钟引脚
	OCDSCK	—	ST	—	OCDS 时钟引脚，仅用于 EV 芯片
PA3/RX	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
	RX	UCR1 UCR2	ST	—	UART 接收引脚
PA4/INT0	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
	INT0	INTC0	ST	—	外部中断 0 输入
PA5/LVDIN	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
	LVDIN	LVDC	ST	—	LVD 输入引脚
PA6/SCK/SCL	PA6	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
	SCK	SIMC0	ST	CMOS	SPI 串行时钟
	SCL	SIMC0	ST	CMOS	I^2C 时钟线
PA7/SDI/SDA	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
	SDI	SIMC0	ST	—	SPI 串行数据输入
	SDA	SIMC0	ST	CMOS	I^2C 数据线
PB0	PB0	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
PB1/OSC1	PB1	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OSC1	CO	HXT	—	HXT 输入
PB2/OSC2	PB2	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OSC2	CO	—	HXT	HXT 输出

引脚名称	功能	OPT	I/T	O/T	说明
PB3/XT1	PB3	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	XT1	CO	LXT	—	LXT 输入
PB4/XT2	PB4	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	XT2	CO	—	LXT	LXT 输出
PB5/TP2_1	PB5	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TP2_1	CTRL0 PTM2C0	ST	CMOS	TM2 输入 / 输出
PB6/TP2_0	PB6	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TP2_0	CTRL0 PTM2C0	ST	CMOS	TM2 输入 / 输出
PB7/TCK2	PB7	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TCK2	PTM2C0	ST	—	TM2 时钟输入
PC0/SDO	PC0	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SDO	SIMC0	—	CMOS	SPI 串行数据输出
PC1/$\overline{SCS}$	PC1	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	$\overline{SCS}$	SIMC0	ST	CMOS	SPI 从机选择
PC2/TCK0	PC2	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TCK0	TM0C0	ST	—	TM0 时钟输入
PC3/TP0_0	PC3	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TP0_0	CTRL0 TM0C0	ST	CMOS	TM0 输入 / 输出
PC4/TP0_1	PC4	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TP0_1	CTRL0 TM0C0	ST	CMOS	TM0 输入 / 输出
PC5/TCK1	PC5	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TCK1	PTM1C0	ST	—	TM1 时钟输入
PC6/TP1_0	PC6	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TP1_0	CTRL0 PTM1C0	ST	CMOS	TM1 输入 / 输出

引脚名称	功能	OPT	I/T	O/T	说明
PC7/TP1_1	PC7	PCPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TP1_1	CTRL0 PTM1C0	ST	CMOS	TM1 输入 / 输出
PD0/INT1	PD0	PDPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	INT1	INTC0	ST	—	外部中断 1 输入
PD1~PD2	PDn	PDPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
SIN	SIN	—	—	AO	正弦波输出
FVL	FVL	—	AI	AO	左脚通道 1
FIL	FIL	—	AI	AO	左脚通道 2
RF1	RF1	—	AI	AO	参考 1 阻抗通道
RF2	RF2	—	AI	AO	参考 2 阻抗通道
FIR	FIR	—	AI	AO	右脚通道 2
FVR	FVR	—	AI	AO	右脚通道 1
TO	TO	—	—	AO	OPA 输出
CP0N	CP0N	—	AI	—	峰值检测器输入
RFC	RFC	—	AI	—	ADC 模拟输入
VOUT/ AVDD	VOUT	—	—	PWR	LDO 输出
	AVDD	—	PWR	—	模拟电源电压
AVSS	AVSS	—	PWR	—	模拟地
VCM	VCM	—	—	PWR	ADC 内部共模电压输出
VERFP	VERFP	—	PWR	—	ADC 外部参考电压输入正端
VERFN	VERFN	—	PWR	—	ADC 外部参考电压输入负端
AN0~AN3	ANn	—	AI	—	ADC 输入通道 0~3
VDD	VDD	—	PWR	—	数字电源电压
VSS	VSS	—	PWR	—	数字地

注：I/T：输入类型；　　O/T：输出类型；
OPT：通过配置选项（CO）或者寄存器选项来配置；
PWR：电源；　　CO：配置选项；
ST：施密特触发输入；　　CMOS：CMOS 输出；
AI：模拟输入；　　AO：模拟输出；
HXT：高频晶体振荡器；
LXT：低频晶体振荡器

极限参数

电源供应电压 ..... V_{SS}-0.3V ~ V_{SS}+6.0V
端口输入电压 ..... V_{SS}-0.3V ~ V_{DD}+0.3V
储存温度 ..... -50℃ ~ 125℃
工作温度 ..... -40℃ ~ 85℃
I_{OH}总电流 ..... -100mA
I_{OL}总电流 150mA
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℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD1}	工作电压 (HXT)	—	f_{SYS}=8MHz	2.2	—	5.5	V
			f_{SYS}=12MHz	2.7	—	5.5	V
			f_{SYS}=16MHz	4.5	—	5.5	V
V_{DD2}	工作电压 (HIRC)	—	f_{SYS}=4.8MHz	2.2	—	5.5	V
I_{DD1}	工作电流 (HXT，$f_{SYS}=f_H$，$f_S=f_{SUB}=f_{LXT}$ 或 f_{LIRC})	3V	无负载，f_H=8MHz，LDO、充电泵、ADC off，WDT 使能	—	1.0	1.5	mA
		5V		—	2.5	4	mA
		3V	无负载，f_H=10MHz，LDO、充电泵、ADC off，WDT 使能	—	1.2	2.0	mA
		5V		—	2.8	4.5	mA
		3V	无负载，f_H=12MHz，LDO、充电泵、ADC off，WDT 使能	—	1.5	2.5	mA
		5V		—	3.5	5.5	mA
		5V	无负载，f_H=16MHz，LDO、充电泵、ADC off，WDT 使能	—	4.5	7.0	mA
		5V	无负载，f_H=20MHz，LDO、充电泵、ADC off，WDT 使能	—	5.5	8.5	mA
I_{DD2}	工作电流 (HIRC，$f_{SYS}=f_H$，$f_S=f_{SUB}=f_{LXT}$ 或 f_{LIRC})	3V	无负载，f_H=4.8MHz，LDO、充电泵、ADC off，WDT 使能	—	0.7	1.2	mA
		5V		—	1.5	2.5	mA
		3V	无负载，f_H=4.8×2MHz，LDO、充电泵、ADC off，WDT 使能	—	1.2	2.0	mA
		5V		—	2.8	4.5	mA
		3V	无负载，f_H=4.8×3MHz，LDO、充电泵、ADC off，WDT 使能	—	1.8	3.0	mA
		5V		—	4.0	6.0	mA

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{DD3}	工作电流 (HXT，$f_{SYS}=f_L$，$f_S=f_{SUB}=f_{LXT}$ 或 f_{LIRC})	3V	无负载，f_H=12MHz，$f_L=f_H/2$，ADC off，WDT 使能	—	0.9	1.5	mA
		5V		—	2.1	3.3	mA
		3V	无负载，f_H=12MHz，$f_L=f_H/4$，LDO、充电泵、ADC off，WDT 使能	—	0.6	1.0	mA
		5V		—	1.6	2.5	mA
		3V	无负载，f_H=12MHz，$f_L=f_H/8$，LDO、充电泵、ADC off，WDT 使能	—	0.48	0.8	mA
		5V		—	1.2	2.0	mA
		3V	无负载，f_H=12MHz，$f_L=f_H/16$，LDO、充电泵、ADC off，WDT 使能	—	0.42	0.7	mA
		5V		—	1.1	1.7	mA
		3V	无负载，f_H=12MHz，$f_L=f_H/32$，LDO、充电泵、ADC off，WDT 使能	—	0.38	0.6	mA
		5V		—	1.0	1.5	mA
		3V	无负载，f_H=12MHz，$f_L=f_H/64$，LDO、充电泵、ADC off，WDT 使能	—	0.36	0.55	mA
		5V		—	1.0	1.5	mA
I_{DD4}	工作电流 (LXT，$f_{SYS}=f_{SUB}=f_{LXT}$，$f_S=f_{SUB}=f_{LXT}$)	3V	无负载，LDO、充电泵、ADC off，WDT 使能，LXTLP=0	—	10	20	μA
		5V		—	30	50	μA
		3V	无负载，LDO、充电泵、ADC off，WDT 使能，LXTLP=1	—	10	20	μA
		5V		—	30	50	μA
I_{DD5}	工作电流 (LIRC，$f_{SYS}=f_{SUB}=f_{LIRC}$，$f_S=f_{SUB}=f_{LIRC}$)	3V	无负载，LDO、充电泵、ADC off，WDT 使能	—	10	20	μA
		5V		—	30	50	μA
I_{STB1}	待机电流 (Idle) (HXT，$f_{SYS}=f_H$，$f_S=f_{SUB}=f_{LXT}$ 或 f_{LIRC})	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=12MHz	—	0.6	1.0	mA
		5V		—	1.2	2.0	mA
I_{STB2}	待机电流 (Idle) (HXT，f_{SYS}=off，f_S=T1)	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=12MHz	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
I_{STB3}	待机电流 (Idle) (HXT，f_{SYS}=off，$f_S=f_{SUB}=f_{LXT}$ 或 f_{LIRC})	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=12MHz	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
I_{STB4}	待机电流 (Idle) (HIRC，f_{SYS}=off，$f_S=f_{SUB}=f_{LIRC}$)	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=4.8×3MHz	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
I_{STB5}	待机电流 (Idle) (HXT，$f_{SYS}=f_L$，$f_S=f_{SUB}=f_{LXT}$ 或 f_{LIRC})	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=12MHz/64	—	0.34	0.6	mA
		5V		—	0.85	1.2	mA
I_{STB6}	待机电流 (Idle) (HXT，f_{SYS}=off，$f_S=f_{SUB}=f_{LXT}$ 或 f_{LIRC})	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=12MHz/64	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{STB7}	待机电流 (Idle) (LXT，$f_{SYS}=f_{SUB}=f_{LXT}$，$f_S=f_{SUB}=f_{LXT}$)	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=32768Hz	—	1.9	4.0	μA
		5V		—	3.3	7.0	μA
I_{STB8}	待机电流 (Idle) (LXT，f_{SYS}=off，f_S=T1)	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=32768Hz	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
I_{STB9}	待机电流 (Idle) (LXT，f_{SYS}=off，$f_S=f_{SUB}=f_{LXT}$)	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=32768Hz	—	1.3	3.0	μA
		5V		—	2..2	5.0	μA
I_{STB10}	待机电流 (Idle) (LIRC，f_{SYS}=off，$f_S=f_{SUB}=f_{LIRC}$)	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=32kHz	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
I_{STB11}	待机电流 (Sleep) (HXT，f_{SYS}=off，$f_S=f_{SUB}=f_{LXT}$ 或 f_{LIRC})	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 除能，f_{SYS}=12MHz	—	0.1	1	μA
		5V		—	0.3	2	μA
I_{STB12}	待机电流 (Sleep) (HXT，f_{SYS}=off，$f_S=f_{SUB}=f_{LXT}$)	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=12MHz	—	1.3	5	μA
		5V		—	2.2	10	μA
I_{STB13}	待机电流 (Sleep) (HXT，f_{SYS}=off，$f_S=f_{SUB}=f_{LIRC}$)	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=12MHz	—	1.3	5	μA
		5V		—	2.2	10	μA
I_{STB14}	待机电流 (Sleep) (LXT，f_{SYS}=off，$f_S=f_{SUB}=f_{LXT}$)	3V	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 使能，f_{SYS}=32768Hz	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
I_{STB15}	待机电流 (Sleep) (HXT，f_{SYS}=off，$f_S=f_{SUB}=f_{LXT}$ 或 f_{LIRC})	—	无负载，系统 HALT，LDO、充电泵、ADC off，WDT 除能，f_{SYS}=12MHz，LVR 使能且 LVDEN=1	—	90	120	μA
V_{IL}	I/O 口、TCKn、TPn_0、TPn_1 和 INTn 的低电平输入电压	—	—	0	—	$0.2V_{DD}$	V
		5V	—	0	—	1.5	V
V_{IH}	I/O 口、TCKn、TPn_0、TPn_1 和 INTn 的高电平输入电压	—	—	$0.8V_{DD}$	—	V_{DD}	V
		5V	—	3.5	—	5	V
V_{LVR1}	低电压复位电压	—	LVR 使能，选择 2.1V	-5%	2.1	+5%	V
V_{LVR2}			LVR 使能，选择 2.55V		2.55		V
V_{LVR3}			LVR 使能，选择 3.15V		3.15		V
V_{LVR4}			LVR 使能，选择 3.8V		3.8		V
I_{LVR}	低电压复位电流	—	LVR 使能，LVDEN=0	—	60	90	μA
V_{LVD1}	低电压检测输入引脚电压	—	LVDEN=1，V_{LVD}=1.04V，VLVD[2:0]=000b	-10%	1.04	+10%	V

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{LVD2}	低电压检测电压	—	LVDEN=1，V_{LVD}=2.2V	-5%	2.2	+5%	V
V_{LVD3}			LVDEN=1，V_{LVD}=2.4V		2.4		V
V_{LVD4}			LVDEN=1，V_{LVD}=2.7V		2.7		V
V_{LVD5}			LVDEN=1，V_{LVD}=3.0V		3.0		V
V_{LVD6}			LVDEN=1，V_{LVD}=3.3V		3.3		V
V_{LVD7}			LVDEN=1，V_{LVD}=3.6V		3.6		V
V_{LVD8}			LVDEN=1，V_{LVD}=4.0V		4.0		V
I_{LVD1}	低电压检测电流	—	LVR 除能，LVDEN=1	—	75	120	μA
I_{LVD2}			LVR 使能，LVDEN=1	—	90	150	μA
I_{OL}	I/O 口灌电流	3V	V_{OL}=0.1V_{DD}	18	36	—	mA
		5V	V_{OL}=0.1V_{DD}	40	80	—	mA
I_{OH}	I/O 口源电流	3V	V_{OH}=0.9V_{DD}，PxPS=00	-1.0	-2.0	—	mA
		5V	V_{OH}=0.9V_{DD}，PxPS=00	-2.0	-4.0	—	mA
		3V	V_{OH}=0.9V_{DD}，PxPS=01	-1.75	-3.5	—	mA
		5V	V_{OH}=0.9V_{DD}，PxPS=01	-3.5	-7.0	—	mA
		3V	V_{OH}=0.9V_{DD}，PxPS=10	-2.5	-5.0	—	mA
		5V	V_{OH}=0.9V_{DD}，PxPS=10	-5.0	-10	—	mA
		3V	V_{OH}=0.9V_{DD}，PxPS=11	-5.5	-11	—	mA
		5V	V_{OH}=0.9V_{DD}，PxPS=11	-11	-22	—	mA
R_{PH}	I/O 口上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

交流电气特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{SYS1}	系统时钟 (HXT)	2.2V~5.5V	—	0.4	—	8	MHz
		2.7V~5.5V		0.4	—	10	MHz
		3.3V~5.5V		0.4	—	12	MHz
		4.5V~5.5V		0.4	—	16	MHz
f_{SYS2}	系统时钟 (HIRC)	5V	Ta=25℃	-2%	4.8×2	+2%	MHz
f_{SYS3}	系统时钟 (LXT)	—	—	—	32768	—	Hz
f_{LIRC}	系统时钟 (LIRC)	5V	Ta=25℃	-10%	32	+10%	kHz
		2.2V~5.5V	Ta= -40℃ ~ 85℃	-50%	32	+60%	kHz
t_{SST}	系统启动时间 (从 HALT 中唤醒)	—	f_{SYS}=HXT 或 LXT	—	1024	—	t_{SYS}
			f_{SYS}=HIRC	—	16	—	
			f_{SYS}=LIRC	—	1~2	—	
t_{RSTD}	系统复位延迟时间 (上电复位)	—	—	25	50	100	ms
	系统复位延迟时间 (上电复位以外其它复位)	—	—	8.3	16.7	33.3	ms
t_{INT}	中断脉宽	—	—	10	—	—	μs
t_{LVR}	低电压复位脉宽	—	—	120	240	480	μs
t_{LVD}	低电压中断脉宽	—	—	60	120	240	μs
t_{LVDS}	LVDO 稳定时间	—	LVR 使能，LVD off → on	—	—	15	μs
		—	LVR 除能，LVD off → on	—	—	150	μs
t_{EERD}	EEPROM 读周期	—	—	—	—	4	t_{SYS}
t_{EEWR}	EEPROM 写周期	—	—	1	2	4	ms
t_{TIMER}	TCKn 和定时器捕捉输入脉宽	—	—	0.3	—	—	μs

注：$t_{SYS}=1/f_{SYS}$

LDO+PGA+ADC+VCM 电气特性

Ta=25℃

符号	参数		测试条件		最小	典型	最大	单位
			V_{DD}					
V_{IN}	LDO 电源电压		—	—	2.7	—	5.5	V
I_{VOREG}	LDO 工作电流		—	无负载， LDOVS[1:0]=00b	—	400	520	μA
V_{OREG}	LDO 输出电压 (I_L=0.1mA， V_{IN}>V_{OREG}+0.2V)		—	LDOVS[1:0]=00b	-5%	2.4	+5%	V
			—	LDOVS[1:0]=01b		2.6		
			—	LDOVS[1:0]=10b		2.9		
			—	LDOVS[1:0]=11b		3.3		
	压降 (I_L=10mA)		—	LDOVS[1:0]=00b	—	—	100	mV
			—	LDOVS[1:0]=01b	—	—	130	
			—	LDOVS[1:0]=10b	—	—	180	
			—	LDOVS[1:0]=11b	—	—	200	
	温度漂移	LDOVS[1:0]=00b I_L=100μA	—	Ta=-40℃ ~ 85℃	—	—	200	Ppm/℃
	V_{OREG} 电压漂移		2.7V~5.5V	—	-0.3	—	+0.3	%/V
ΔV_{LOAD}	负载调节率		2.7V	负载 =0mA~10mA， 单片机 HALT， LDO=2.4V，LDO 使能，其它功能除能	—	25	50	mV
V_{CM}	V_{CM} 输出电压		—	VCMS=0， AV_{DD}=3.3V，无负载	-5%	1.05	+5%	V
			—	VCMS=1, AV_{DD}=3.3V, 无负载	-5%	1.25	+5%	V
			—	I_L=200μA	0.98	—	1.02	V
	温度漂移		—	I_L=10μA， Ta= -40℃ ~ 85℃	—	—	200	Ppm/℃
	AV_{DD} 电压漂移		—	无负载， AV_{DD}=2.4V~3.3V	—	100	—	μV/V
t_{VCM}	VCM 打开稳定时间		—	—	10	—	—	ms
I_{CMSRC}	VCM 源电流		—	V_{CM} 下降 2%	1	—	—	mA
I_{CMSNK}	VCM 灌电流		—	V_{CM} 上升 2%	1	—	—	mA
ADC & ADC 内部参考电压 (Delta-Sigma ADC)								
AV_{DD}	VCM、PGA 和 ADC 的电源电压		—	—	2.4	—	3.3	V

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}					
I_{CM}+ I_{PGA}+ I_{ADC}	VCM、PGA 和 ADC 的工作电流	—	VCM 使能， VRBUFP=1， VRBUFN=1	—	—	900	μA
			VCM 使能， VRBUFP=0， VRBUFN=0	—	600	750	
			VCM 除能， VRBUFP=0， VRBUFN=0	—	500	650	
I_{ADSTB}	待机电流	—	系统 HALT，无负载	—	—	1	μA
RS_{AD}	ADC 分辨率	—	—	—	—	20	Bit
NNFC	无噪声码	—	PGA Gain =128 数据传输速率 =10Hz	—	15.4	—	Bit
ENOB	有效位数	—	PGA Gain =128 数据传输速率 =10Hz	—	18.1	—	Bit
f_{AD}	A/D 时钟频率 (f_{MCLK})	—	—	—	4.8	—	MHz
f_{ADO}	ADC 输出数据传输速率	—	f_{MCLK}=4.8MHz FLMS[2:0]=000， ADC CLK=f_{MCLK}/30	5	—	625	Hz
			f_{MCLK}=4.8MHz FLMS[2:0]=010， ADC CLK=f_{MCLK}/12	12	—	1563	
V_{REF+}	参考输入电压	—	VREFS=1， VRBUFP=0， VRBUFN=0	0.96	1.25	2.2	V
V_{REF-}				0	0	1.0	
V_{REF}		—	V_{REF}=(V_{REF+})-(V_{REF-})	0.96	1.25	1.44	
PGA							
V_{DI+}, V_{DI-}	绝对 / 共模输入电压	—	—	0.4	—	AV_{DD}-1.1	V
ΔDI±	差分输入电压范围	—	Gain = PGS×AGS	ΔV_{R-}/Gain	—	ΔV_{R+}/Gain	V
TC_{PGA}	增益温度漂移	—	Ta= -40℃ ~ 85℃	—	5	—	Ppm/℃
温度传感器							
TC_S	传感器的温度漂移	—	Ta= -40℃ ~ 85℃ ΔVR=1.25V， VGS[1:0]=00 (Gain=1)， VRBUFP=0， VRBUFN=0	—	175	—	μV/℃

有效位数 (ENOB)

AV_{DD}=3.3V，V_{REF}=1.25V，f_{MCLK}=4.8MHz，FLMS[2:0]=000

数据传输速率 (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
5	19.7	19.8	19.6	19.7	19.7	19.6	19.2	18.6
10	19.4	19.3	19.3	19.3	19.3	19.1	18.7	18.1
20	19.0	18.8	18.7	18.9	18.8	18.6	18.2	17.5
39	18.4	18.3	18.3	18.3	18.3	18.1	17.7	17.0
78	18.1	17.9	18.0	17.9	17.9	17.6	17.2	16.5
156	17.6	17.4	17.4	17.4	17.3	17.1	16.6	15.9
313	15.8	15.8	15.9	15.8	15.9	15.9	15.8	15.3
625	14.1	14.0	14.0	14.1	14.1	14.0	14.1	14.4

AV_{DD}=3.3V，V_{REF}=1.25V，f_{MCLK}=4.8MHz，FLMS[2:0]=010

数据传输速率 (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
12	19.4	18.8	18.7	18.8	18.8	18.7	18.9	18.1
24	19.0	18.3	18.3	18.3	18.3	18.2	17.9	17.3
49	18.5	17.8	17.8	17.8	17.9	17.7	17.4	16.8
98	18.2	18.2	18.1	18.2	18.1	17.8	17.2	16.4
195	17.9	17.8	17.8	17.8	17.6	17.3	16.7	15.9
391	17.4	17.2	17.2	17.2	17.1	16.8	16.2	15.4
781	16.2	16.1	16.1	16.1	16.1	15.9	15.5	14.8
1563	14.5	14.5	14.5	14.4	14.5	14.5	14.3	14.0

运算放大器电气特性（体脂电路）

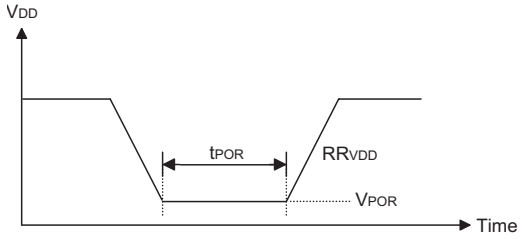
Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	电源电压	—	—	2.2	—	5.5	V
Icc	每个信号放大器的电源电流	5V	I_O = 0A	150	360	500	μA
OP0, OP2							
SR	单位增益的转换速率	3V	R_L = 100kΩ, C_L = 100pF	7.5	—	—	V/μs
GBW	增益带宽积	3V	R_L = 100kΩ, C_L = 100pF	—	—	2	MHz
OP1							
SR	单位增益的转换速率	3V	R_L = 100kΩ, C_L = 100pF	7.5	—	—	V/μs
GBW	增益带宽积	3V	R_L = 100kΩ, C_L = 100pF	—	—	5	MHz

上电复位特性

Ta=25℃

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

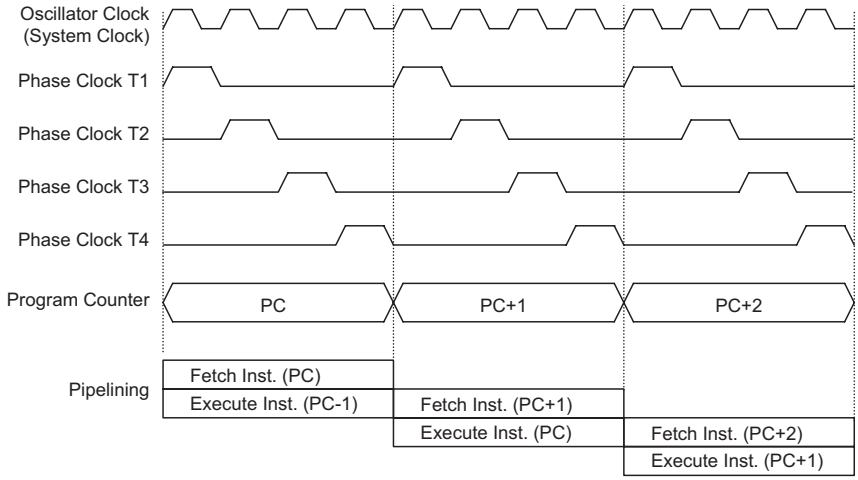


系统结构

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，此单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的取得和执行同时进行，此举使得除了跳转和调用指令需要一个以上指令周期外，大部分的标准指令或扩展指令分别能在一个指令周期或两个指令周期内完成。8 位 ALU 参与指令集中所有的运算，它可完成算术运算、逻辑运算、移位、递增、递减和分支等功能，而内部的数据路径则是以通过累加器和 ALU 的方式加以简化。有些寄存器在数据存储器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供具有最大可靠度和灵活性的 I/O 和 A/D 控制系统时，仅需要少数的外部器件。使得这些单片机适用于低成本和批量生产的控制应用。

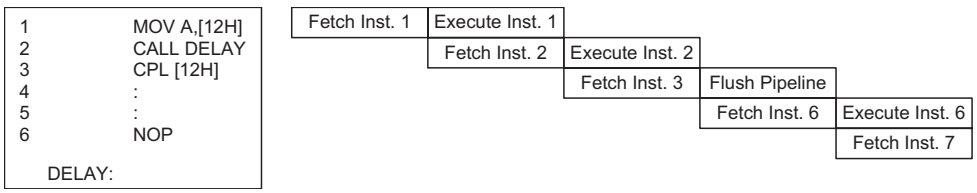
时序和流水线结构

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



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。



指令捕捉

程序计数器

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

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

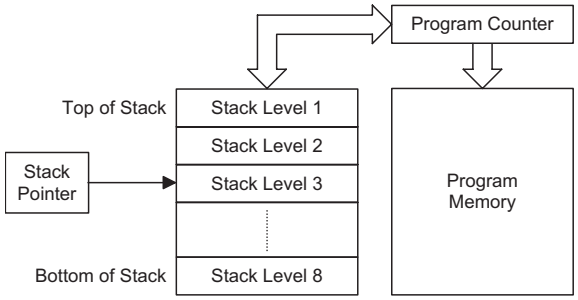
程序计数器	
程序计数器高字节	PCL 寄存器
PC11~PC8	PCL7~PCL0

程序计数器

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

堆栈

堆栈是一个特殊的存储空间，用来存储程序计数器中的内容。该单片机有 8 层堆栈，堆栈既不是数据部分也不是程序空间部分，而且它既不是可读取也不是可写入的。当前层由堆栈指针 (SP) 加以指示，同样也是不可读写的。在子程序调用或中断响应服务时，程序计数器的内容被压入到堆栈中。当子程序或中断响应结束时，返回指令 (RET 或 RETI) 使程序计数器从堆栈中重新得到它以前的值。当一个芯片复位后，堆栈指针将指向堆栈顶部。



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

若堆栈溢出，则首个存入堆栈的程序计数器数据将会丢失。

算术逻辑单元 – ALU

算术逻辑单元是单片机中很重要的部分，执行指令集中的算术和逻辑运算。ALU 连接到单片机的数据总线，在接收相关的指令码后执行需要的算术与逻辑操作，并将结果存储在指定的寄存器，当 ALU 计算或操作时，可能导致进位、借位或其它状态的改变，而相关的状态寄存器会因此更新内容以显示这些改变，ALU 所提供的功能如下：

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

Flash 程序存储器

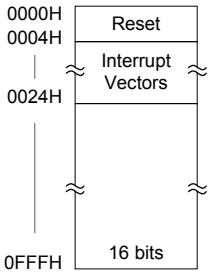
程序存储器用来存放用户代码即储存程序。程序存储器为 FLASH 类型意味着可以多次重复编程，方便用户使用同一芯片进行程序的修改。使用适当的单片机编程工具，此单片机提供用户灵活便利的调试方法和项目开发规划及更新。

结构

程序存储器的容量为 4K×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。

特殊向量

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



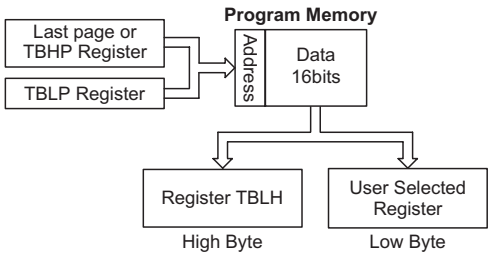
程序存储器结构

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 TBHP 中。这些寄存器定义表格总的地址。

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

下图是查表中寻址 / 数据流程：



查表范例

以下范例说明表格指针和表格数据如何被定义和执行。这个例子使用的表格数据用 ORG 伪指令储存在存储器中。ORG 指令的值“0F00H”指向的地址是 4K 程序存储器中最后一页的起始地址。表格指针低字节寄存器的初始值设为 06H，这可保证从数据表格读取的第一笔数据位于程序存储器地址 0F06H，即最后一页起始地址后的第六个地址。值得注意的是，假如“TABRD [m]”或 LTABRD 指令被使用，则表格指针指向 TBHP 和 TBLP 指定的地址。在这个例子中，表格数据的高字节等于零，而当“TABRD [m]”或 LTABRD 指令被执行时，此值将会自动的被传送到 TBLH 寄存器。

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

表格读取程序举例

```
tempreg1 db ?       ; temporary register #1
tempreg2 db ?       ; temporary register #2
:
:
mov a,06h           ; initialise low table pointer - note that this address
                    ; is referenced
mov tblp,a          ; to the last page or the page that tbhp pointed
mov a,0Fh           ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1      ; transfers value in table referenced by table pointer
                    ; data at program
                    ; memory address "0F06H" transferred to tempreg1 and TBLH
dec tblp            ; reduce value of table pointer by one
tabrd tempreg2      ; transfers value in table referenced by table pointer
                    ; data at program memory address "0F05H" transferred to
                    ; tempreg2 and TBLH in this example the data "1AH" is
                    ; transferred to tempreg1 and data "0FH" to register
                    ; tempreg2
:
:
org 0F00h           ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

在线烧录

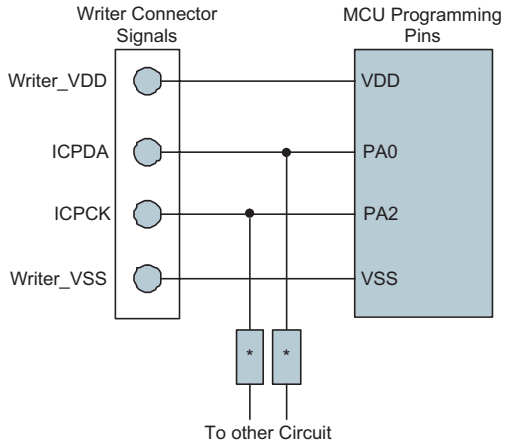
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。

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写，在无需去除或重新插入芯片的情况下方便地保持程序为最新版。

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

芯片内部程序存储器和 EEPROM 存储器都可以通过 4 线的接口在线进行烧录。其中 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。

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



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

片上调试

EV 芯片 HT45V75 用于 HT45F75 单片机仿真。此 EV 芯片提供片上调试功能（OCDS—On-Chip Debug Support）用于开发过程中的单片机调试。除了片上调试功能方面，EV 芯片和实际单片机在功能上几乎是兼容的。用户可将 OCDSDA 和 OCDSCK 引脚连接至 Holtek HT-IDE 开发工具，从而实现 EV 芯片对实际单片机的仿真。OCDSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户用 EV 芯片进行调试时，实际单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。关于 OCDS 功能的详细描述，请参考“Holtek e-Link for 8-bit MCU OCDS User's Guide”文件。

Holtek e-Link 引脚名称	EV 芯片引脚名称	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

在应用烧录 – IAP

该单片机提供 IAP 功能来对 Flash ROM 进行数据和程序更新。用户可自行定义 IAP ROM 地址，但是用户在使用 IAP 功能时必须注意几个特点。

- 擦除块：256 个字 / 块
- 写：4 个字 / 次
- 读：1 个字 / 次

IAP 控制寄存器

位于数据存储器所有 Sector 地址寄存器 FARL/FARH 和数据寄存器 FD0L/FD0H、FD1L/FD1H、FD2L/FD2H 和 FD3L/FD3H，以及位于数据存储器 Sector 1 控制寄存器 FC0 和 FC1，都是与 IAP 相关的 Flash 存取寄存器。由于间接寻址是存取 FC0 和 FC1 寄存器的唯一方式，因此所有与这些寄存器相关的读写操作必须使用间接寻址寄存器 IAR1 或 IAR2，和存储器指针对 MP1L/MP1H 或 MP2L/MP2H 来执行。由于 FC0 和 FC1 控制寄存器位于地址 28H~29H 数据存储器 Sector 1 中，位于 28H~29H 地址范围的值必须首先被写入 MP1L 或 MP2L 存储器指针低字节，且“01”值也被写入 MP1H 或 MP2H 存储器指针高字节。

- **FC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	1	1	0	0	0	0

Bit 7 **CFWEN**：Flash 存储器写使能控制位
0：Flash 存储器写功能除能
1：Flash 存储器写功能已被成功使能
当此位由应用程序清零后，Flash 存储器写功能除能。注意，此位写“1”则会导致无效操作。此位被用来说明 Flash 存储器写功能状态。当此位由硬件置为“1”时，就意味着 Flash 存储器写功能已经成功使能，否则此位为“0”，该功能除能。

Bit 6~4 **FMOD2~FMOD0**：模式选择
000：写程序存储器
001：块擦除程序存储器
010：保留位
011：读程序存储器
100：保留位
101：保留位
110：FWEN 模式—Flash 存储器写功能使能模式
111：保留位

Bit 3 **FWPEN**：Flash 存储器写步骤使能控制
0：除能
1：使能
当此位置为“1”且 FMOD2~FMOD0 为“110”时，IAP 控制器将执行“Flash 存储器写功能使能”步骤。一旦 Flash 存储器写功能成功使能，无需再设置 FWPEN 位。

Bit 2 **FWT**：Flash 存储器写开始控制位
0：不开始 Flash 存储器写过程或 Flash 存储器写过程已完成
1：开始 Flash 存储器写过程
此位由软件置“1”。当 Flash 存储器写过程完成，由硬件清零。

Bit 1 **FRDEN**：Flash 存储器读使能位
0：Flash 存储器读除能
1：Flash 存储器读使能

Bit 0 **FRD**：Flash 存储器读开始控制位
0：不开始 Flash 存储器读过程或 Flash 存储器读过程已完成
1：开始 Flash 存储器读过程
此位由软件置“1”。当 Flash 存储器读过程完成，由硬件清零。

- **FC1 寄存器**

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

Bit 7~0 **D7~D0**：整个芯片复位
当用户写“55H”到该寄存器，将产生一个复位信号将整个单片机复位。

- **FARL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	A7	A6	A5	A4	A3	A2	A1	A0
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 Flash 存储器地址 [7:0]

- **FARH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	A15	A14	A13	A12	A11	A10	A9	A8
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 Flash 存储器地址 [15:8]

- **FD0L 寄存器**

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

Bit 7~0　　第一个 Flash 存储器数据 [7:0]

- **FD0H 寄存器**

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

Bit 7~0　　第一个 Flash 存储器数据 [15:8]

- **FD1L 寄存器**

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

Bit 7~0　　第二个 Flash 存储器数据 [7:0]

- **FD1H 寄存器**

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

Bit 7~0　　第二个 Flash 存储器数据 [15:8]

- **FD2L 寄存器**

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

Bit 7~0　　第三个 Flash 存储器数据 [7:0]

- **FD2H 寄存器**

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

Bit 7~0　　第三个 Flash 存储器数据 [15:8]

- **FD3L 寄存器**

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

Bit 7~0　　第四个 Flash 存储器数据 [7:0]

- **FD3H 寄存器**

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

Bit 7~0　　第四个 Flash 存储器数据 [15:8]

Flash 存储器写功能使能步骤

为使用户可以通过 IAP 控制寄存器来更改 Flash 存储器数据，用户必须首先使能 Flash 存储器写操作，步骤如下：

步骤 1、写“110”到 FMOD2~FMOD0 位，选择 FWEN 模式。

步骤 2、FWPEN 置为“1”。步骤 1 和步骤 2 可同时执行。

步骤 3、模式数据序列 00H、04H、0DH、09H、C3H 和 40H 必须分别写入寄存器 FD1L、FD1H、FD2L、FD2H、FD3L 和 FD3H。

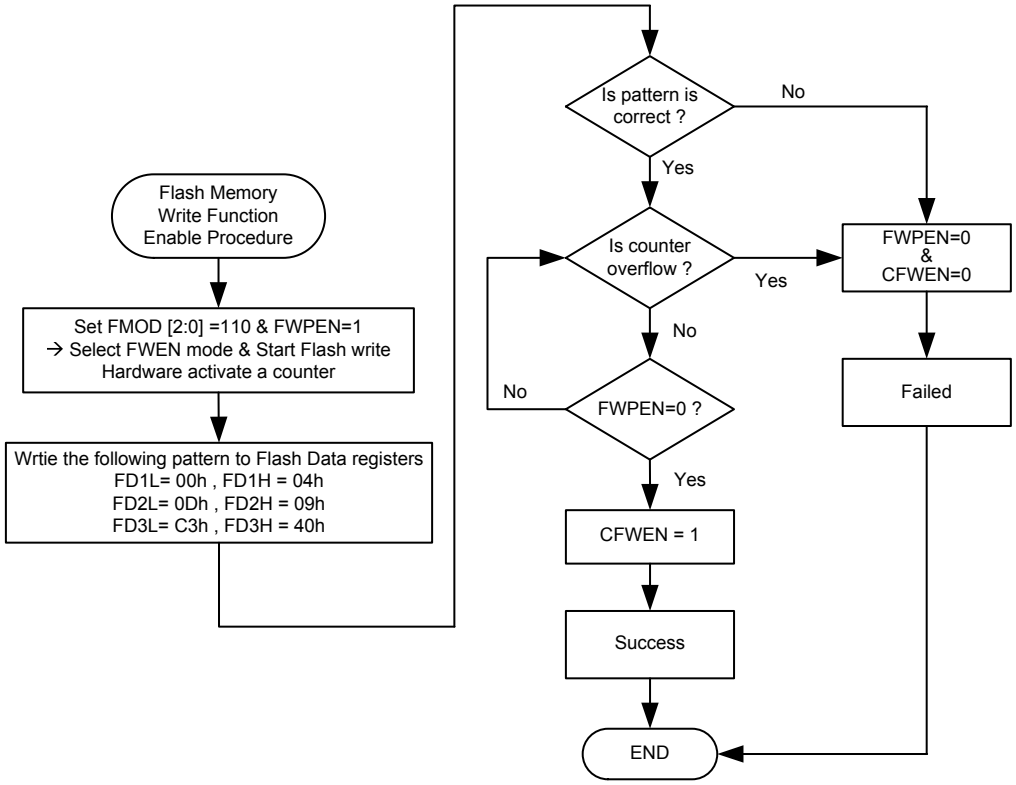
步骤 4、溢出周期为 300μs 的计数器将进行有效计时，此时允许用户将正确的数据序列写入 FD1L/FD1H~FD3L/FD3H 寄存器对。计数器时钟来自 LIRC 振荡器。

步骤 5、如果计数器溢出或数据序列不正确，Flash 存储器写操作不被使能且用户必须再次重复以上步骤。FWPEN 位将自动由硬件清零。

步骤 6、如果计数器溢出前数据序列正确，Flash 存储器写操作将使能且 FWPEN 位将自动由硬件清零。CFWEN 位也由硬件置为“1”，表明 Flash 存储器写操作成功使能。

步骤 7、一旦 Flash 存储器写操作使能，用户可通过 Flash 控制寄存器更改 Flash ROM 数据。

步骤 8、用户可以清零 CFWEN 位来除能 Flash 存储器写操作。



Flash 存储器写功能使能步骤

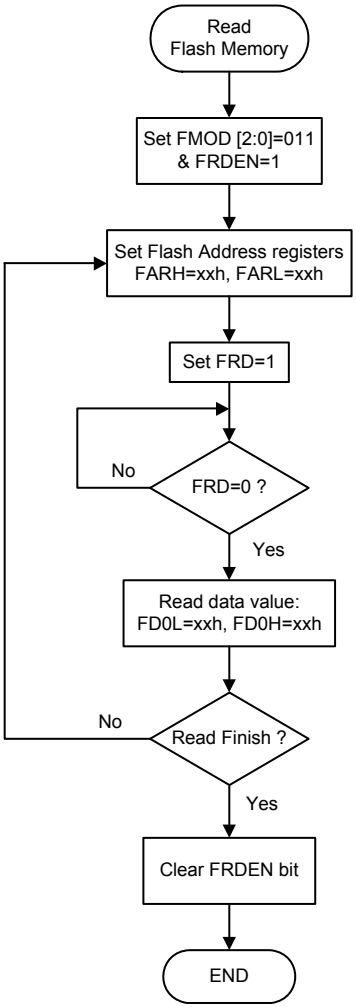
Flash 存储器读 / 写步骤

通过前面的 IAP 步骤成功使能 Flash 存储器写功能后，用户必须先擦除相应的 Flash 存储块，然后再开始 Flash 存储器读 / 写操作。块擦除操作的数量是每块 256 个字，有效的块擦除地址只能由 FARH 寄存器指定，FARL 寄存器不用来指定块擦除地址。

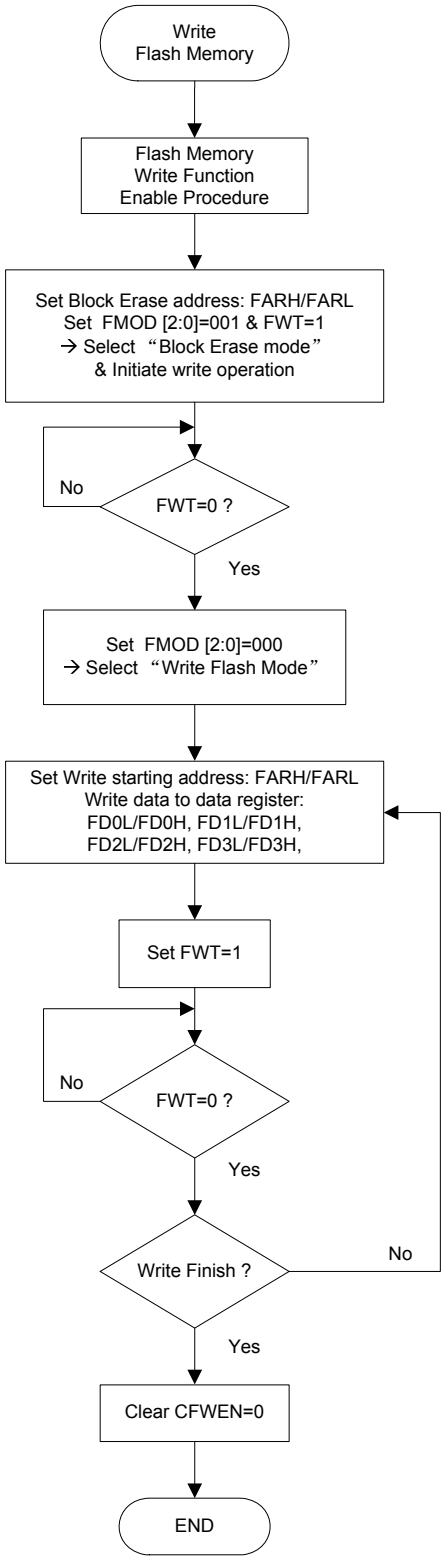
擦除块	FARH	FARL
0	0000 0000	xxxx xxxx
1	0000 0001	xxxx xxxx
2	0000 0010	xxxx xxxx
3	0000 0011	xxxx xxxx
4	0000 0100	xxxx xxxx
5	0000 0101	xxxx xxxx
6	0000 0110	xxxx xxxx
7	0000 0111	xxxx xxxx
8	0000 1000	xxxx xxxx
9	0000 1001	xxxx xxxx
10	0000 1010	xxxx xxxx
11	0000 1011	xxxx xxxx
12	0000 1100	xxxx xxxx
13	0000 1101	xxxx xxxx
14	0000 1110	xxxx xxxx
15	0000 1111	xxxx xxxx

“x”表示不相关

擦除块数量和选择



读 Flash 存储器步骤



写 Flash 存储器步骤

注：当 FWT 或 FRD 位置为“1”，单片机停止工作。

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。

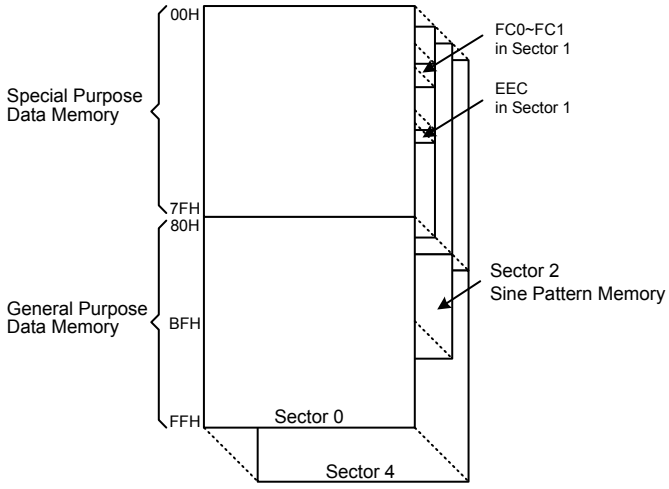
数据存储器分为三部分，第一部分是特殊功能数据存储器。这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二部分数据存储器是做一般用途使用，都可在程序控制下进行读取和写入。第三部分数据存储器用于正弦波形函数。正弦波形存储区的地址与通用数据存储区的地址重叠。

结构

数据存储器被分为若干个 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。位于 Sector 2 地址 80H~BFH 为正弦波形存储器。

特殊功能数据存储器的起始地址为“00H”，而通用数据存储器和正弦波形存储器的起始地址都为“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 寄存器和处于 28H~29H 地址的 FC0 和 FC1 寄存器却只能在 Sector 1 中被访问到。切换不同的数据存储器 Sector 可通过设置正确的存储器指针值实现。

容量	Sectors
通用数据存储器：256×8	0：80H~FFH 2：80H~BFH（用于正弦波形） 3：80H~FFH



数据存储器结构

通用数据存储器

通用数据存储器共 256 字节位于 Sector 0 和 Sector 3 的 80H~FFH。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，极大地方便了用户在数据存储器内进行位操作。

特殊功能数据存储器

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

	Sector 0,2,3	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH		
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH		
10H	INTC0	
11H	INTC1	
12H	INTC2	
13H		
14H	MFI0	
15H	MFI1	
16H	MFI2	
17H	MFI3	
18H	PAWU	
19H	PAPU	
1AH	PA	
1BH	PAC	
1CH	PBPU	
1DH	PB	
1EH	PBC	
1FH	PCPU	
20H	PC	
21H	PCC	
22H	PDPU	
23H	PD	
24H	PDC	
25H		
26H		
27H		
28H		FC0
29H		FC1
2AH		
2BH	LVRC	
2CH	CTRL	
2DH		
2EH	ADRL	
2FH	ADRH	
30H	ADCR0	
31H	ADCR1	
32H	ADCS	
33H	ADRM	
34H	PTM1C0	
35H	PTM1C1	
36H	PTM1DL	
37H	PTM1DH	
38H	PTM1AL	
39H	PTM1AH	
3AH	TM0C0	
3BH	TM0C1	
3CH	TM0DL	
3DH	TM0DH	
3EH	TM0AL	
3FH	TM1AH	

	Sector 0,2,3	Sector 1
40H		EEC
41H	EEA	
42H	EED	
43H	PTM2C0	
44H	PTM2C1	
45H	PTM2DL	
46H	PTM2DH	
47H	PTM2AL	
48H	PTM2AH	
49H	CTRL0	
4AH	PTM1RPL	
4BH	PTM1RPH	
4CH	PTM2RPL	
4DH	PTM2RPH	
4EH	SLEDC0	
4FH	SLEDC1	
50H	PWRC	
51H	PGAC0	
52H	PGAC1	
53H	PGACS	
54H	USR	
55H	UCR1	
56H	UCR2	
57H	BRG	
58H	TXRRXR	
59H	IRCTRL0	
5AH	IRCTRL1	
5BH		
5CH		
5DH		
5EH		
5FH		
60H	SIMC0	
61H	SIMC1	
62H	SIMA/SIMC2	
63H	SIMD	
64H		
65H	SIMTOC	
66H	FARL	
67H	FARH	
68H	FD0L	
69H	FD0H	
6AH	FD1L	
6BH	FD1H	
6CH	FD2L	
6DH	FD2H	
6EH	FD3L	
6FH	FD3H	
70H	SGC	
71H	SGN	
72H	SGDNR	
73H	OPAC	
74H	SWC	
75H	DACO	
76H	FTRC	
77H	SMOD	
78H	LVDC	
79H	INTEG	
7AH	WDTC	
7BH	TBC	
7CH		
7DH		
7EH		
7FH		

: Unused, read as 00H

特殊功能数据存储器结构

特殊功能寄存器

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

间接寻址寄存器 – IAR0, IAR1, IAR2

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

存储器指针 – MP0, MP1L, MP1H, MP2L, MP2H

该单片机提供五个存储器指针，即 MP0、MP1L、MP1H、MP2L 和 MP2H。由于这些指针在数据存储器中能像普通的寄存器一般被操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由存储器指针所指定的地址。MP0、IAR0 用于访问 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

- **Example 1**

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
code .section at 0 code
org 00h
start:
mov a,04h                  ; setup size of block
mov block,a
mov a,offset adres1        ; Accumulator loaded with first RAM address
mov mp0,a                  ; setup memory pointer with first RAM address
loop:
clr IAR0                   ; clear the data at address defined by MP0
inc mp0                    ; increment memory pointer
sdz block                  ; check if last memory location has been cleared
jmp loop
continue:
    :
```

- **Example 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，或低半字节减法运算的结果没有产生借位时，AC 被置位，否则 AC 被清零。
- Z：当算术或逻辑运算结果是零时，Z 被置位，否则 Z 被清零。
- OV：当运算结果高两位的进位状态异或结果为 1 时，OV 被置位，否则 OV 被清零。
- PDF：系统上电或执行“CLR WDT”指令会清零 PDF，而执行“HALT”指令则会置位 PDF。

- TO：系统上电或执行“CLR WDT”或“HALT”指令会清零 TO，而当 WDT 溢出则会置位 TO。

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

STATUS 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SC	CZ	TO	PDF	OV	Z	AC	C
R/W	R	R	R	R	R/W	R/W	R/W	R/W
POR	x	x	0	0	x	x	x	x

“x”为未知

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

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

Bit 5 **TO**：看门狗溢出标志位
0：系统上电或执行“CLR WDT”或“HALT”指令后
1：看门狗溢出发生

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

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

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

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

Bit 0 **C**：进位标志位
0：无进位
1：如果在加法运算中结果产生了进位，或在减法运算中结果不发生借位
C 也受循环移位指令的影响。

EEPROM 数据存储器

此单片机的一个特性是内建 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 数据存储器结构

EEPROM 数据存储器容量为 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 中的一个地址和数据寄存器以及 Sector 1 中的一个控制寄存器，可以实现对 EEPROM 的单字节读写操作。

EEPROM 寄存器

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

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

EEPROM 寄存器列表

EEA 寄存器

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

“x”为未知

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	x	x	x	x	x	x	x	x

“x”为未知

Bit 7~0　数据 EEPROM 数据
数据 EEPROM 数据 Bit 7~Bit 0

EEC 寄存器

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

Bit 7~4　未定义，读为“0”

Bit 3　**WREN**：数据 EEPROM 写使能位
0：除能
1：使能
此位为数据 EEPROM 写使能位，向数据 EEPROM 写操作之前需将此位置高。将此位清零时，则禁止向数据 EEPROM 写操作。

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

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

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

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

从 EEPROM 中读取数据

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

写数据到 EEPROM

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

写保护

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

EEPROM 中断

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

编程注意事项

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

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

程序举例

- 从 **EEPROM** 中读取数据 – 轮询法

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

- 写数据到 **EEPROM** – 轮询法

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

振荡器

不同的振荡器选择可以让使用者在不同的应用需求中实现更大范围的功能。振荡器的灵活性使得在速度和功耗方面可以达到最优化。振荡器选择是通过配置选项和寄存器共同完成的。

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。它们提供的高速和低速系统振荡器具有较宽的频率范围。所有振荡器选择通过配置选项选择。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

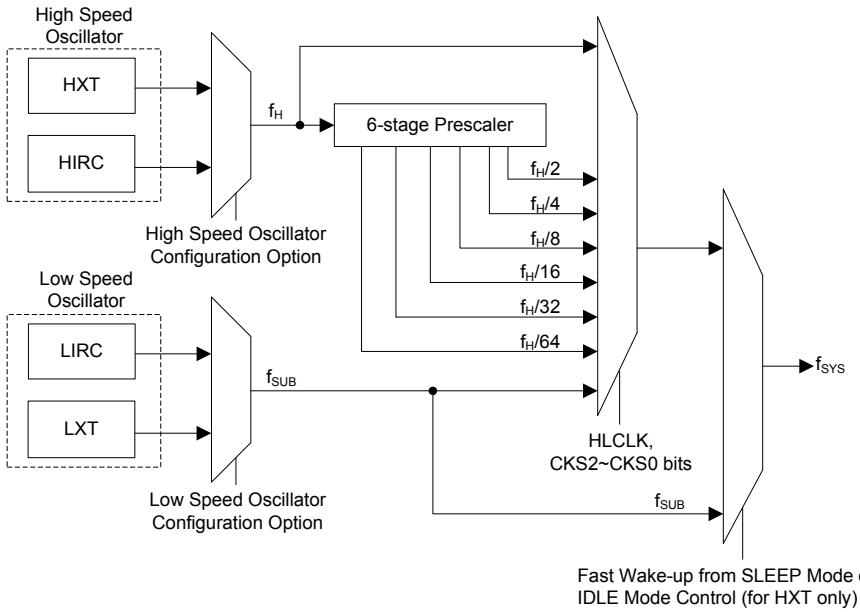
类型	名称	频率	引脚
外部高速晶振	HXT	400kHz~20MHz	OSC1/OSC2
内部高速 RC	HIRC	4.8MHz，4.8×2MHz 或 4.8×3MHz	—
外部低速晶振	LXT	32.768kHz	XT1/XT2
内部低速 RC	LIRC	32kHz	—

振荡器类型

系统时钟配置

此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。高速振荡器有外部晶体 / 陶瓷振荡器—HXT 和内部 4.8MHz、4.8×2MHz 或 4.8×3MHz RC 振荡器—HIRC，低速振荡器有外部 32.768kHz 晶振—LXT 和内部 32kHz 低速振荡器—LIRC。使用高速或低速振荡器作为系统时钟的选择是通过设置 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位决定的，系统时钟可动态选择。

高速或低速振荡器的实际时钟源经由配置选项选择。请注意，两个振荡器必须做出选择，即一个高速和一个低速振荡器。

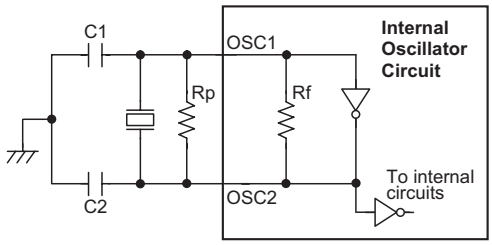


系统时钟配置

外部晶体 / 陶瓷振荡器 – HXT

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。

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



Note: 1. Rp 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
20MHz	0pF	0pF
12MHz	0pF	0pF
8MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	100pF	100pF
注：C1 和 C2 数值仅作参考用		

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 振荡器具有三种固定的频率：4.8MHz，4.8×2MHz，4.8×3MHz。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因 V_{DD}、温度以及芯片制成工艺不同的影响减至最低程度。在电源电压为 5V 及温度为 25℃的条件下，4.8×2MHz 这个固定频率的容差为 2%。如果选择了该内部时钟，无需额外的引脚；PB1 和 PB2 可以作为通用 I/O 口使用。

内部 32kHz 振荡器 – LIRC

内部 32kHz 系统振荡器是一个低频振荡器，经由配置选项选择。这种单片机有一个完全集成 RC 振荡器，它在 5V 电压下运行的典型频率值为 32kHz 且无需外部元件。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响减至最低。因此，内部 32kHz 振荡器频率在 25℃温度 5V 电压下的精度保持在 10% 以内。

外部 32.768kHz 晶体振荡器 – LXT

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

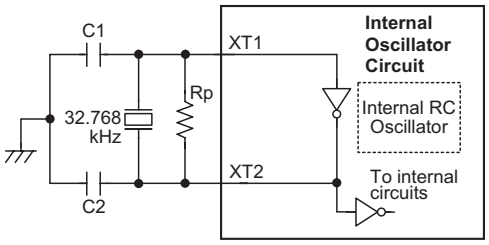
当系统进入空闲 / 休眠模式，系统时钟关闭以降低功耗。然而在某些应用，比如空闲 / 休眠模式下要保持内部定时器功能，必须提供额外的时钟，且与系统时钟无关。

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

一些配置选项决定 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它们之间的连线都应尽可能的接近单片机。



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

外部 LXT 振荡器

LXT 振荡器 C1 和 C2 值		
晶体频率	C1	C2
32.768kHz	10pF	10pF
注：1、C1 和 C2 数值仅作参考用 2、R_P 的建议值为 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 TBC 寄存器中的 LXTLP 位进行模式选择。

LXTLP 位	LXT 工作模式
0	快速启动
1	低功耗

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。

辅助振荡器

低速振荡器除了提供一个系统时钟源外，也用来为看门狗定时器、时基中断和 SIM 提供时钟来源。

工作模式和系统时钟

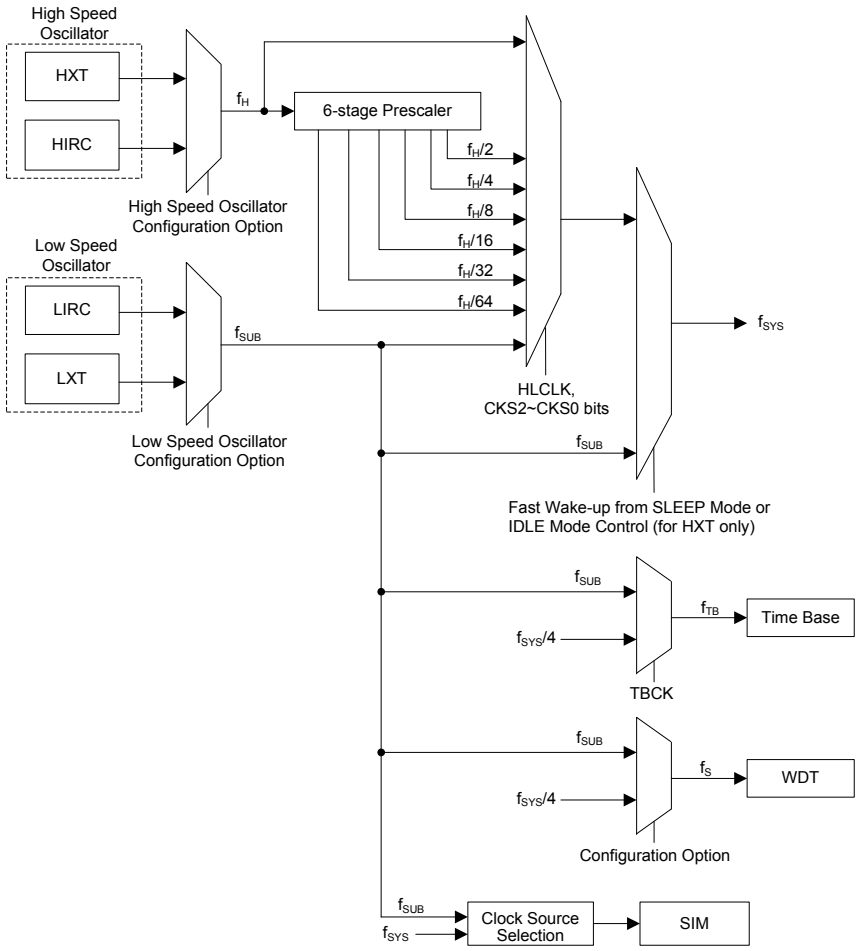
现今的应用要求单片机具有较高的性能及尽可能低的功耗，这种矛盾的要求在便携式电池供电的应用领域尤为明显。高性能所需要的高速时钟将增加功耗，反之亦然。此单片机提供高、低速两种时钟源，它们之间可以动态切换，用户可通过优化单片机操作来获得最佳性能 / 功耗比。

系统时钟

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

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

当唤醒发生后，f_{SUB} 提供次时钟源以更快实现唤醒。f_{SUB} 用于看门狗定时器、时基中断功能、TM 和 SIM 的时钟源。



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_S
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/Off
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	On

正常模式

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

低速模式

此模式的系统时钟虽为较低速时钟源，但单片机仍能正常工作。该低速时钟源来自 LIRC 或 LXT 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、f_{SUB} 及 f_S 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。然而当 WDT 时钟源经配置选项选择为 f_{SUB} 时，若 LVDEN 位为“1”或看门狗定时器功能使能，f_{SUB} 及 f_S 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，但一些外围功能如看门狗定时器、TM 和 SIM 将继续工作。在空闲模式 0 中，系统振荡器停止，看门狗定时器时钟 f_S 开启或关闭由 f_S 所选时钟源决定。若时钟源为 f_{SYS}/4，f_S 关闭；若时钟源为 f_{SUB}，f_S 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，但会提供一个时钟源给一些外围功能如看门狗定时器、TM 和 SIM。在空闲模式 1 中，系统振荡器继续运行，该系统振荡器可以为高速或低速系统振荡器。在该模式中，无论看门狗定时器时钟源来自 f_{SYS}/4 还是 f_{SUB}，f_S 开启。

控制寄存器

寄存器 SMOD 和 CTRL 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	FSTEN	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
POR	0	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 **FSTEN**：快速唤醒控制位（仅用于 HXT）

0：除能
1：使能

此位为快速唤醒控制位，用于决定单片机被唤醒后 f_{SUB} 是否开始工作。若此位为高，当 f_{SUB} 有效时，此时钟源可作为暂用系统时钟以提供一个较快的唤醒时间。

Bit 3 **LTO**：低速振荡器就绪标志位

0：未就绪
1：就绪

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

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 HXT 振荡器，该位将在 1024 个时钟周期后变为高电平状态，若使用 HIRC 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：$f_H/2$~$f_H/64$ 或 f_{SUB}
1：f_H
此位用于选择 f_H 或 $f_H/2$~$f_H/64$ 还是 f_{SUB} 作为系统时钟。该位为高时选择 f_H 作为系统时钟，为低时则选择 $f_H/2$~$f_H/64$ 或 f_{SUB} 作为系统时钟。当系统时钟由 f_H 时钟向 f_{SUB} 时钟转换时，f_H 将自动关闭以降低功耗。

CTRL 寄存器

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

“x”为未知

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

Bit 6~3 未使用，读为“0”

Bit 2 **LVRF**：LVR 复位标志位
详见其它章节

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

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

快速唤醒

单片机进入休眠模式或空闲模式 0 后，系统时钟将停止以降低功耗。然而单片机再次唤醒，原来的系统时钟重新起振、稳定且恢复正常工作需要一定的时间。为确保单片机能够尽快的开始工作，系统提供了一个快速唤醒功能。需提供一个临时时钟源 f_{SUB} 先驱动系统直至原系统振荡器稳定，这个临时时钟来自 LXT 或 LIRC 振荡器。快速启动功能的时钟源为 f_{SUB}，该功能仅在休眠模式 1 和空闲模式 0 中有效。当单片机由休眠模式 0 唤醒时，因 f_{SUB} 已停止，故快速唤醒功能无效。快速唤醒功能使能 / 除能由 SMOD 寄存器中 FSTEN 位控制的。

若 HXT 振荡器作为正常模式的系统时钟，且快速唤醒功能使能，系统唤醒将需 1~2 个 t_{SUB} 时钟周期。系统开始在 f_{SUB} 时钟源下运行直至 1024 个 HXT 时钟周期后 HTO 标志转换为高，系统将切换到 HXT 振荡器运行。

若系统振荡器选用 HIRC，将系统从休眠模式或空闲模式 0 中唤醒需 15~16 个时钟周期；若选用 LIRC，则需 1~2 个周期。快速唤醒位 FSTEN 在这些情况下不受影响。

系统振荡器	FSTEN 位	唤醒时间（休眠模式 0）	唤醒时间（休眠模式 1）	唤醒时间（空闲模式 0）	唤醒时间（空闲模式 1）
HXT	0	1024 个 HXT 周期	1024 个 HXT 周期		1~2 个 HXT 周期
	1	1024 个 HXT 周期	1~2 个 f_{SUB} 周期（系统在 f_{SUB} 下运行 1024 个 HXT 周期后切换到 HXT 振荡器运行）		1~2 个 HXT 周期
HIRC	x	15~16 个 HIRC 周期	15~16 个 HIRC 周期		1~2 个 HIRC 周期
LIRC	x	1~2 个 LIRC 周期	1~2 个 LIRC 周期		1~2 个 LIRC 周期
LXT	x	1024 个 LXT 周期	1024 个 LXT 周期		1~2 个 LXT 周期

“x”表示未知

唤醒时间

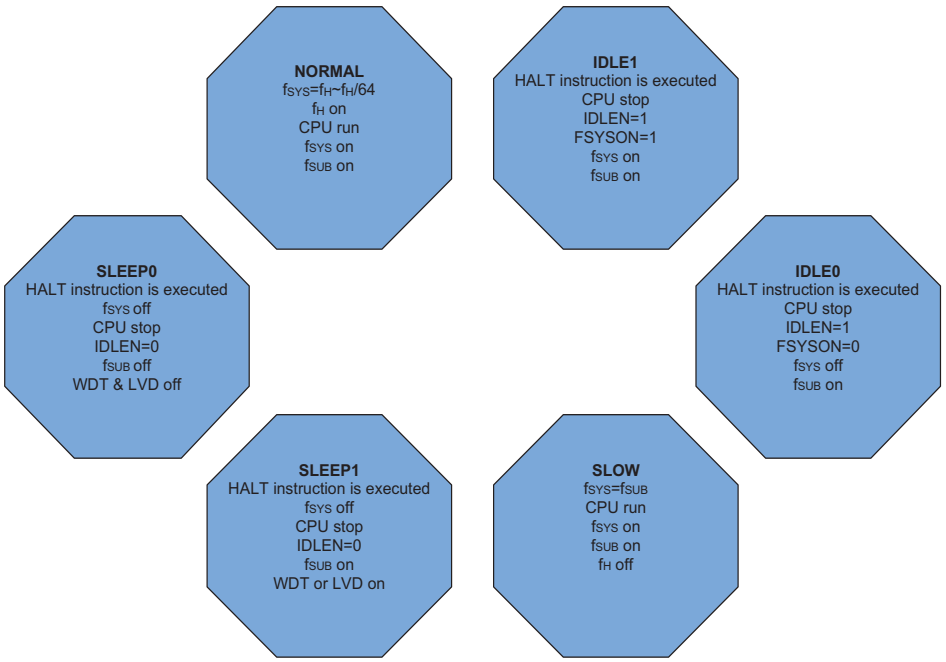
注：若看门狗定时器除能，意味着 LXT 和 LIRC 都关闭，当单片机由休眠模式 0 中唤醒时，快速唤醒功能不可用。

工作模式切换

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

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

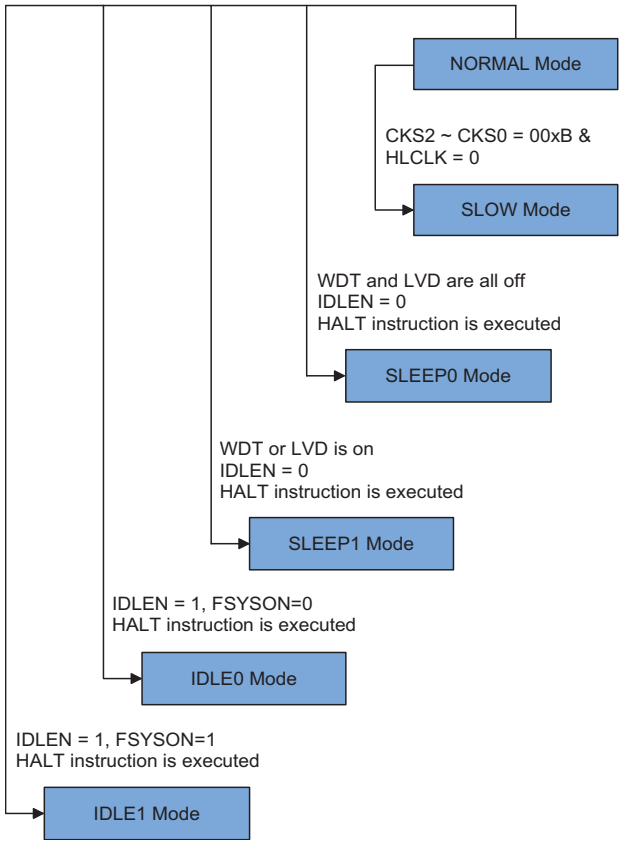
当 HLCLK 位变为低电平时，时钟源将由高速时钟源 f_H 转换成时钟源 $f_H/2$~$f_H/64$ 或 f_{SUB}。若时钟源来自 f_{SUB}，高速时钟源将停止运行以节省耗电。此时须注意，$f_H/16$ 和 $f_H/64$ 内部时钟源也将停止运行，由此会影响到如 TM 和 SIM 等内部功能的工作。所附流程图显示了单片机在不同工作模式间切换时的变化。



正常模式切换到低速模式

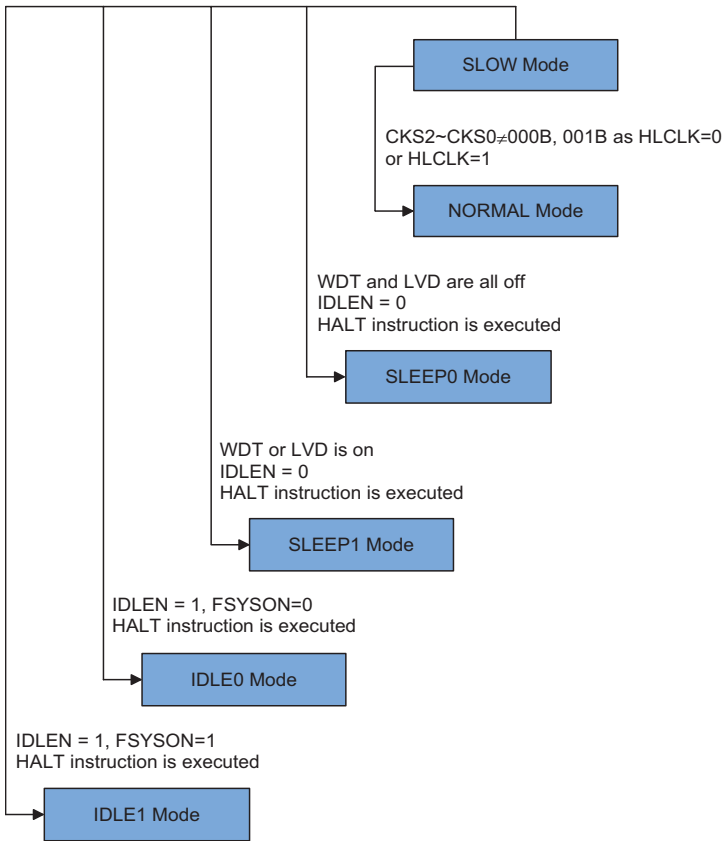
系统运行在正常模式时使用高速系统振荡器，因此较为耗电。可通过设置 SMOD 寄存器中的 HLCLK 位为“0”及 CKS2~CKS0 位为“000”或“001”使系统时钟切换至运行在低速模式下。此时将使用低速系统振荡器以节省耗电。用户可在对性能要求不高的操作中使用此方法以减少耗电。

低速模式的时钟源来自 LIRC 或 LXT 振荡器，因此要求这些振荡器在所有模式切换动作发生前稳定下来。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 LIRC 或 LXT 低速振荡器。切换到使用高速系统时钟振荡器的正常模式需设置 HLCLK 位为“1”，也可设置 HLCLK 位为“0”但 CKS2~CKS0 需设为“010”、“011”、“100”、“101”、“110”或“111”。高频时钟需要一定的稳定时间，通过检测 HTO 位的状态可进行判断。高速振荡器的稳定时间由所使用高速系统振荡器的类型决定。



进入休眠模式 0

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

- 系统时钟和 f_{SUB} 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。
- 无论 WDT 时钟源来自 f_{SUB} 时钟或系统时钟，WDT 都将被清除并停止运行。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。

进入休眠模式 1

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

- 系统时钟停止运行，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 使能且其时钟源来自 f_{SUB}，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，f_{SUB} 时钟将继续运行。
- 数据存储器中的内容和寄存器将保持当前值。
- 若 WDT 使能且其时钟源来自 f_{SUB}，则 WDT 将被清零并重新开始计数；若其时钟源来自系统时钟，则 WDT 将停止运行。
- 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。

进入空闲模式 1

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

- 系统时钟和 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，无论 WDT 时钟源来自 f_{SUB} 或是系统时钟，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。

待机电流的注意事项

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

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

编程注意事项

HXT 和 LXT 振荡器使用相同的 SST 计数器。例如，若系统从休眠模式 0 中唤醒，HXT 和 LXT 振荡器都需从关闭状态快速启动。HXT 振荡器结束其 SST 周期后，LXT 振荡器才开始使用 SST 计数器。

- 若单片机从休眠模式 0 唤醒后进入正常模式，高速系统振荡器需要一个 SST 周期。在 HTO 为“1”后，单片机开始执行首条指令。此时，若 f_{SUB} 时钟来源于 LXT 振荡器，LXT 振荡器可能不是稳定的，上电状态可能会发生类似情况，首条指令执行时 LXT 振荡器还未就绪。
- 若单片机从休眠模式 1 唤醒后进入正常模式，系统时钟源来自 HXT 振荡器且 FSTEN 为“1”，唤醒后，系统时钟可切换至 LXT 或 LIRC 振荡器。
- 一些外围功能，如 WDT、TM 和 SIM，采用系统时钟 f_{SYS} 时，在系统时钟源由 f_H 切换至 f_{SUB} 时，以上这些功能的时钟源也要随之改变。

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_S，而 f_S 的时钟源又是通过配置选项从 f_{SUB} 和 $f_{SYS}/4$ 中选择。f_{SUB} 时钟由 LXT 或 LIRC 振荡器提供，可通过配置选项设置。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。另一个看门狗定时器时钟源选项为 $f_{SYS}/4$。

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。寄存器结合配置选项控制看门狗定时器的工作。

WDTC 寄存器

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

Bit 7~3 **WE4~WE0**：WDT 软件控制位

如果 WDT 配置选项为“总是使能”

10101 或 01010：使能

其它：MCU 复位

如果 WDT 配置选项为“由 WDTC 寄存器控制”

10101：除能

01010：使能

其它：MCU 复位

如果因外部环境干扰使这些位发生改变，则单片机将在 2~3 个 f_{SUB} 时钟周期后复位，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：$2^8/f_S$

001：$2^{10}/f_S$

010：$2^{12}/f_S$

011：$2^{14}/f_S$

100：$2^{15}/f_S$

101：$2^{16}/f_S$

110：$2^{17}/f_S$

111：$2^{18}/f_S$

CTRL 寄存器

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

“x”为未知

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

Bit 6~3 未使用，读为“0”

Bit 2 **LVRF**：LVR 复位标志位
详见其它章节

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

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

看门狗定时器操作

当 WDT 溢出时，它产生一个芯片复位的动作。这也就意味着正常工作期间，用户需在应用程序中看门狗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。通过配置选项选择看门狗定时器的一些选项，如使能 / 除能控制及时钟源选择。除了配置选项使能 / 除能看门狗定时器外，WDTC 寄存器中的 WE4~WE0 位也可用来使能 / 除能看门狗定时器及其复位控制。如果 WDT 配置选项选择“总是使能”，WE4~WE0 位仍对 WDT 功能有影响。如果 WE4~WE0 为“01010”或“10101”，则 WDT 使能。但若因环境干扰致使 WE4~WE0 改变为其它值，则在 2~3 个 f_{SUB} 时钟周期后单片机将产生复位；如果 WDT 配置选项选择“由 WDTC 寄存器控制”，WE4~WE0 位可以选择 WDT 的工作模式。如果 WE4~WE0 为“10101”，WDT 除能。如果 WE4~WE0 为“01010”，WDT 使能。如果 WE4~WE0 由于环境干扰设为其它值，则会在 2~3 个 f_{SUB} 时钟周期后复位单片机。复位后 WE4~WE0 初始化为“01010”。

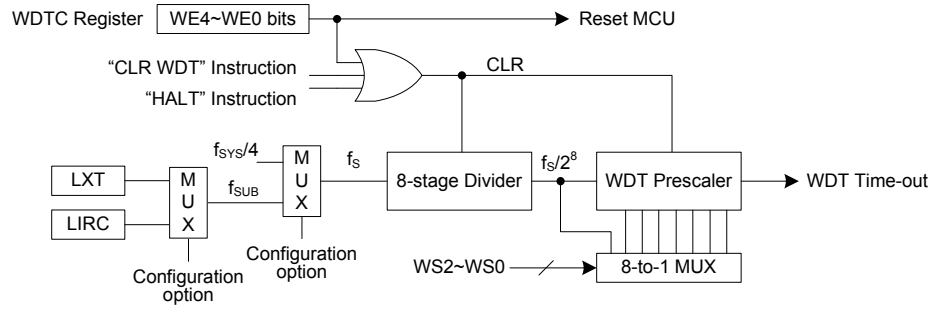
WDT 配置选项	WE4~WE0 位	WDT 功能
总是使能	01010 或 10101	使能
	其它值	单片机复位
由 WDTC 寄存器控制	10101	除能
	01010	使能
	其它值	单片机复位

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 应置位，仅 PC 和堆栈指针复位。有三种方法可以用来清除 WDT 的内容。第一种是 WDT 复位，即写入除 01010 和 10101 外任何值到 WE4~WE0 位，第二种是通过软件清除指令，而第三种是通过“HALT”指令。

只有一种软件指令用于清除看门狗寄存器。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 7.8ms。



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。

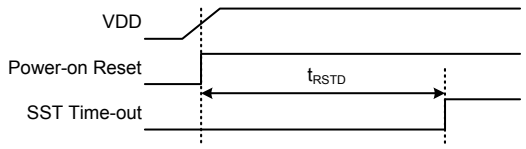
另一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。

复位功能

单片机共有四种内部复位方式：

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。

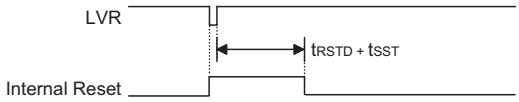


注：t_{RSTD} 为上电延迟时间，典型值为 50ms。

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。低电压复位功能总是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，必须超过交流电气特性中 t_{LVR} 参数的值。如果低电压存在不超过 t_{LVR} 参数的值，则 LVR 将会忽略它且不会执行复位功能。V_{LVR} 参数值可通过 LVRC 寄存器中的 LVS7~LVS0 位设置。若由于受到干扰 LVS7~LVS0 变为其它值时，需经过 2~3 个 f_{SUB} 周期响应复位。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 16.7ms。

低电压复位时序图

- **LVRC 寄存器**

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

Bit 7~0 **LVS7~LVS0**：LVR 电压选择

01010101：2.1V
00110011：2.55V
10011001：3.15V
10101010：3.8V
其它值：MCU 复位（寄存器复位为 POR 值）

若低电压情况发生且满足以上定义的低电压复位值，则单片机复位。需要经过 2~3 个 f_{SUB} 时钟周期响应复位。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 2~3 个 f_{SUB} 时钟周期响应复位。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	—	—	—	—	LVRF	LRF	WRF
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	x	0	0

“x”为未知

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

Bit 6~3 未使用，读为“0”

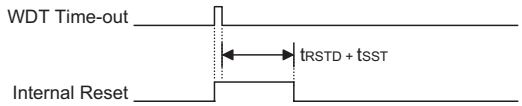
Bit 2 **LVRF**：LVR 复位标志位
0：未发生
1：发生
当特定的低电压复位条件发生时，该位被置为“1”。该位只能由应用程序清零。

Bit 1 **LRF**：LVRC 控制寄存器软件复位标志位
0：未发生
1：发生
如果 LVRC 寄存器包含任何非定义的 LVR 电压值，该位被置为“1”，这类似于软件复位功能。该位只能由应用程序清零。

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

正常运行时看门狗溢出复位

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

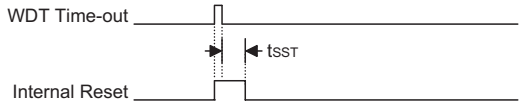


注：t_{RSTD} 为上电延迟时间，典型值为 16.7ms。

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：如果系统时钟为 HIRC，t_{SST} 为 15~16 个时钟周期。
如果系统时钟为 HXT 或者 LXT，t_{SST} 为 1024 个时钟周期。
如果系统时钟为 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

“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
定时模块	所有定时器模块停止
输入 / 输出口	I/O 口设为输入模式，AN0~AN3 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 (正常模式)	WDT 溢出 (正常模式)	WDT 溢出 (HALT)
MP0	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
IAR1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
IAR2	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	---- xxxx	---- uuuu	---- uuuu	---- uuuu
STATUS	xx00 xxxx	xxuu uuuu	xx1u uuuu	xx11 uuuu
SMOD	0000 0011	0000 0011	0000 0011	uuuu uuuu
INTEG	---- 0000	---- 0000	---- 0000	---- uuuu
LVDC	--00 -000	--00 -000	--00 -000	--uu —uuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	LVR 复位 (正常模式)	WDT 溢出 (正常模式)	WDT 溢出 (HALT)
INTC2	--00 --00	--00 --00	--00 --00	--uu —uu
MFI0	--00 --00	--00 --00	--00 --00	--uu —uu
MFI1	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI2	0-00 0-00	0-00 0-00	0-00 0-00	u-uu u-uu
MFI3	--00 --00	--00 --00	--00 --00	--uu —uu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	---- -111	---- -111	---- -111	---- -uuu
PDC	---- -111	---- -111	---- -111	---- -uuu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
TBC	0011 0111	0011 0111	0011 0111	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDPU	---- -000	---- -000	---- -000	---- -uuu
TM0C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
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
SLEDC0	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC1	--00 0000	--00 0000	--00 0000	--uu uuuu
PWRC	00-- -000	00-- -000	00-- -000	uu-- -uuu
PGAC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
PGAC1	10-- 000-	10-- 000-	10-- 000-	uu-- uuu-
PGACS	--00 0000	--00 0000	--00 0000	--uu 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
TXRRXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
IRCTRL0	0000 0000	0000 0000	0000 0000	uuuu uuuu
IRCTRL1	---- ---0	---- ---0	---- ---0	---- ---u
ADCR0	0010 00-0	0010 00-0	0010 00-0	uuuu uu-u
ADCR1	0000 000-	0000 000-	0000 000-	uuuu uuu-

寄存器	上电复位	LVR 复位 (正常模式)	WDT 溢出 (正常模式)	WDT 溢出 (HALT)
ADRL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH	---- xxxx	---- xxxx	---- xxxx	---- uuuu
ADCS	---0 0000	---0 0000	---0 0000	---u uuuu
ADRM	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
LVRC	0101 0101	0101 0101	0101 0101	uuuu uuuu
CTRL	0--- -x00	0--- -x00	0--- -x00	u--- -uuu
PTM1C0	0000 0---	0000 0---	0000 0---	uuuu u---
PTM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DH	---- --00	---- --00	---- --00	---- --uu
PTM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1AH	---- --00	---- --00	---- --00	---- --uu
PTM2C0	0000 0---	0000 0---	0000 0---	uuuu u---
PTM2C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DH	---- --00	---- --00	---- --00	---- --uu
PTM2AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2AH	---- --00	---- --00	---- --00	---- --uu
CTRL0	---- -000	---- -000	---- -000	---- -uuu
PTM1RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	---- --00	---- --00	---- --00	---- --uu
PTM2RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	---- --00	---- --00	---- --00	---- --uu
SIMC0	111- 0000	111- 0000	111- 0000	uuu- uuuu
SIMC1	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMA/ SIMC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMTOC	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEA	--xx xxxx	--xx xxxx	--xx xxxx	--uu uuuu
EED	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
EEC	---- 0000	---- 0000	---- 0000	---- uuuu
FC0	0111 0000	0111 0000	0111 0000	uuuu uuuu
FC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARL	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARH	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0L	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	LVR 复位 (正常模式)	WDT 溢出 (正常模式)	WDT 溢出 (HALT)
FD2L	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	0000 0000	uuuu uuuu
SGC	0000 ----	0000 ----	0000 ----	uuuu ----
SGN	--00 0000	--00 0000	--00 0000	--uu uuuu
SGDNR	---0 0000	---0 0000	---0 0000	---u uuuu
OPAC	0--- 0000	0--- 0000	0--- 0000	u--- uuuu
SWC	0000 0000	0000 0000	0000 0000	uuuu uuuu
DACO	--00 0000	--00 0000	--00 0000	--uu uuuu
FTRC	0--0 --00	0--0 --00	0--0 --00	u--u --uu

注：“-”表示未定义
“x”表示未知
“u”表示不改变

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。

此单片机提供 PA~PD 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	D7	D6	D5	D4	D3	D2	D1	D0
PAPU	D7	D6	D5	D4	D3	D2	D1	D0
PA	D7	D6	D5	D4	D3	D2	D1	D0
PAC	D7	D6	D5	D4	D3	D2	D1	D0
PBPU	D7	D6	D5	D4	D3	D2	D1	D0
PB	D7	D6	D5	D4	D3	D2	D1	D0
PBC	D7	D6	D5	D4	D3	D2	D1	D0
PCPU	D7	D6	D5	D4	D3	D2	D1	D0
PC	D7	D6	D5	D4	D3	D2	D1	D0
PCC	D7	D6	D5	D4	D3	D2	D1	D0
PDPU	—	—	—	—	—	D2	D1	D0
PD	—	—	—	—	—	D2	D1	D0
PDC	—	—	—	—	—	D2	D1	D0

输入 / 输出寄存器列表

上拉电阻

许多产品应用在端口处于输入状态时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~PDPU 来设置，它用一个 PMOS 晶体管来实现上拉电阻功能。

PAPU 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　PA 口 bit 7~bit 0 上拉电阻控制位
0：除能
1：使能

PBPU 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 PB 口 bit 7~bit 0 上拉电阻控制位
0：除能
1：使能

PCPU 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 PC 口 bit 7~bit 0 上拉电阻控制位
0：除能
1：使能

PDPU 寄存器

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

Bit 7~3 未使用，读为“0”
Bit 2~0 PD 口 bit 2~bit 0 上拉电阻控制位
0：除能
1：使能

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	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 PA 口 bit 7~bit 0 唤醒功能控制位
0：除能
1：使能

输入 / 输出端口控制寄存器

每一个输入 / 输出口都具有各自的控制寄存器，即 PAC~PDC，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。

PAC 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	1	1	1	1	1	1	1	1

Bit 7~0 PA 口 bit 7~bit 0 输入 / 输出控制位
0：输出
1：输入

PBC 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	1	1	1	1	1	1	1	1

Bit 7~0 PB 口 bit 7~bit 0 输入 / 输出控制位
0：输出
1：输入

PCC 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	1	1	1	1	1	1	1	1

Bit 7~0 PC 口 bit 7~bit 0 输入 / 输出控制位
0：输出
1：输入

PDC 寄存器

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

Bit 7~3 未使用，读为“0”
Bit 2~0 PD 口 bit 2~bit 0 输入 / 输出控制位
0：输出
1：输入

输入 / 输出端口源电流控制

该单片机的每个 I/O 口都支持不同的源电流驱动能力。通过设置相应的选择寄存器 SLEDC0 和 SLEDC1，每个 I/O 口可以支持四个级别的源电流驱动能力。用户可参考直流电气特性章节为不同应用选择所需的源电流。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SLEDC0	PBPS3	PBPS2	PBPS1	PBPS0	PAPS3	PAPS2	PAPS1	PAPS0
SLEDC1	—	—	PDPS1	PDPS0	PCPS3	PCPS2	PCPS1	PCPS0

I/O 口源电流控制寄存器列表

SLEDC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PBPS3	PBPS2	PBPS1	PBPS0	PAPS3	PAPS2	PAPS1	PAPS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7~6 **PBPS3~PBPS2**：PB7~PB4 源电流选择位
00：level 0（最小值）
01：level 1
10：level 2
11：level 3（最大值）

Bit 5~4 **PBPS1~PBPS0**：PB3~PB0 源电流选择位
00：level 0（最小值）
01：level 1
10：level 2
11：level 3（最大值）

Bit 3~2 **PAPS3~PAPS2**：PA7~PA4 源电流选择位
00：level 0（最小值）
01：level 1
10：level 2
11：level 3（最大值）

Bit 1~0 **PAPS1~PAPS0**：PA3~PA0 源电流选择位
00：level 0（最小值）
01：level 1
10：level 2
11：level 3（最大值）

SLEDC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	PDPS1	PDPS0	PCPS3	PCPS2	PCPS1	PCPS0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	1	0	1	0	1

Bit 7~6 未使用，读为“0”

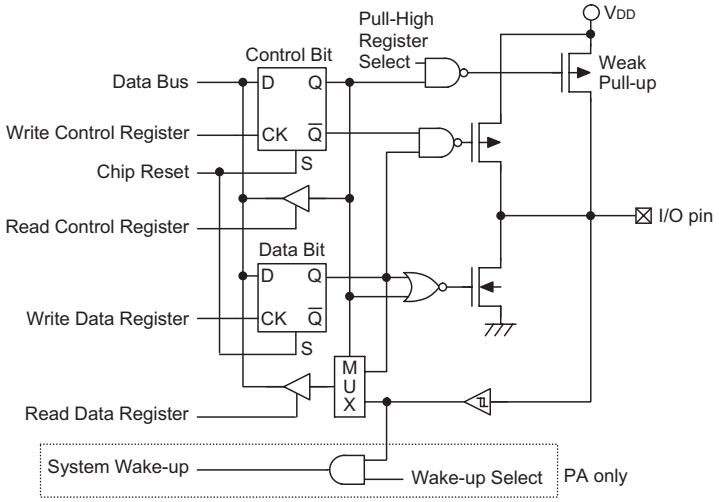
Bit 5~4 **PDPS1~PDPS0**：PD2~PD0 源电流选择位
00：level 0（最小值）
01：level 1
10：level 2
11：level 3（最大值）

Bit 3~2 **PCPS3~PCPS2**：PC7~PC4 源电流选择位
00：level 0（最小值）
01：level 1
10：level 2
11：level 3（最大值）

Bit 1~0 **PCPS1~PCPS0**：PC3~PC0 源电流选择位
00：level 0（最小值）
01：level 1
10：level 2
11：level 3（最大值）

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

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~PDC，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~PD 在程序中被预先设定。设置哪些引脚是输入及哪些引脚是输出，可通过设置正确的值到适当的端口控制寄存器，或使用指令“SET [m].i”及“CLR [m].i”来设定端口控制寄存器中个别的位。注意，当使用这些位控制指令时，系统即将产生一个读 - 修改 - 写的操作。单片机需要先读入整个端口上的数据，修改个别的位，然后重新把这些数据写入到输出端口。

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

定时器模块 – TM

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

这里只介绍各种 TM 的共性，更多详细资料请参考简易型和周期型定时器章节。

简介

该单片机包含 3 个 TM，分别命名为 TM0、TM1 和 TM2。TM0 为 10-bit 简易型 TM，TM1 和 TM2 都为 10-bit 周期型 TM。虽然性质相似，但不同 TM 特性复杂度不同。本章介绍简易型和周期型 TM 的共性，更多详细资料分别见后面各章。两种类型 TM 的特性和区别见下表。

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
10-bit CTM	10-bit PTM	10-bit PTM

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_{SUB} 时钟源或外部 TCKn 引脚。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

两种不同类型的 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/PTMnC0 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 有两个输出引脚 TPn_0 和 TPn_1。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 中输出引脚是不同的，详见下表。

TM 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	寄存器
TP0_0, TP0_1	TP1_0, TP1_1	TP2_0, TP2_1	CTRL0

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 / 输出功能的引脚。某一 TM 输出引脚用作 TM 输入 / 输出功能时，另一引脚将用作普通输入 / 输出功能。

CTRL0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	TP2CPS	TP1CPS	TP0CPS
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3　未使用，读为“0”

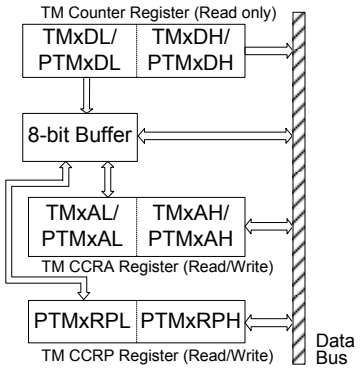
Bit 2　**TP2CPS**：TP2_0，TP2_1 引脚选择
0：TP2_0
1：TP2_1
当 TM2 使能时，如果 TP2_0 作为 TM2 输入 / 输出引脚，则 TP2_1 作为普通输入 / 输出脚使用，反之亦然。

Bit 1　**TP1CPS**：TP1_0，TP1_1 引脚选择
0：TP1_0
1：TP1_1
当 TM1 使能时，如果 TP1_0 作为 TM1 输入 / 输出引脚，则 TP1_1 作为普通输入 / 输出脚使用，反之亦然。

Bit 0　**TP0CPS**：TP0_0，TP0_1 引脚选择
0：TP0_0
1：TP0_1
当 TM0 使能时，如果 TP0_0 作为 TM0 输入 / 输出引脚，则 TP0_1 作为普通输入 / 输出脚使用，反之亦然。

编程注意事项

TM 计数寄存器和捕捉 / 比较寄存器 CCRA、CCRP 为 10-bit 的寄存器，含有低字节和高字节结构。高字节可直接访问，低字节则仅能通过一个内部 8-bit 的缓存器进行访问。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、PTMxAL 或 PTMxRPL，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、PTMxAL 或 PTMxRPL
    - 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、PTMxAH 或 PTMxRPH
    - 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、PTMxDH、TMxAH、PTMxAH 或 PTMxRPH 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、PTMxDL、TMxAL、PTMxAL 或 PTMxRPL 读取数据
    - 注意，此时读取 8-bit 缓存器中的数据。

简易型 TM – CTM（TM0）

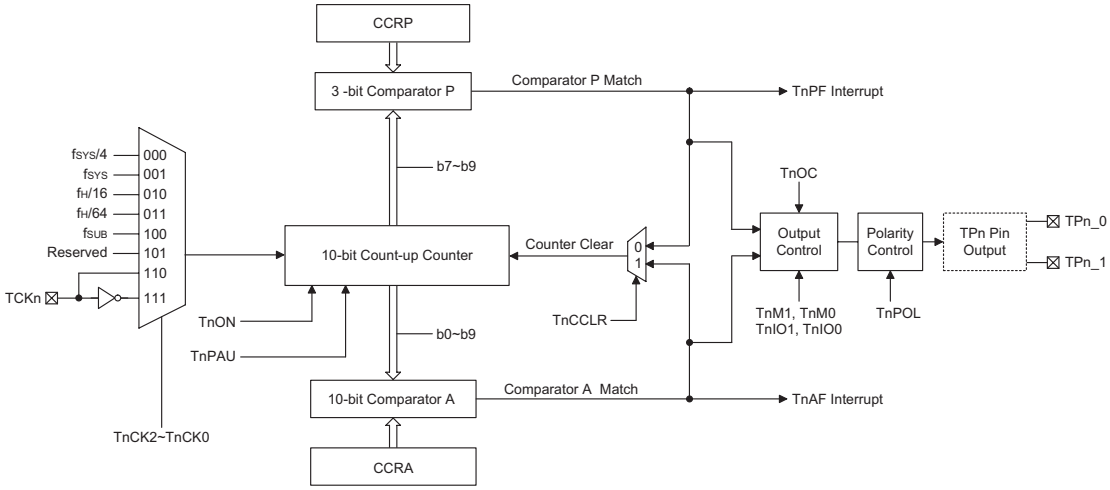
虽然简易型 TM 是这两种 TM 类型中最简单的形式，但仍然包括三种工作模式，即比较匹配输出，定时 / 事件计数器和 PWM 输出模式。简易型 TM 由外部输入脚控制并驱动两个外部输出脚。这两个引脚可以是相同或相反的信号。

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

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 方框图 (n=0)

简易型 TM 寄存器介绍

简易型 TM 的所有操作由六个寄存器控制。其中包含一对只读寄存器用来存放 10 位计数器的值，一对读 / 写寄存器存放 10 位 CCRA 的值，剩下两个控制寄存器设置不同的操作和控制模式以及 CCRP 的 3 个位。

寄存器名称	位							
	7	6	5	4	3	2	1	0
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 TMn 寄存器列表 (n=0)

TMnC0 寄存器 (n=0)

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_{SYS}

010：$f_H/16$

011：$f_H/64$

100：f_{SUB}

101：保留位

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。选择保留时钟输入将有效地除能内部计数器。外部引脚时钟源能被选择在上升沿或下降沿有效。f_{SYS} 是系统时钟，f_H 和 f_{SUB} 是其它的内部时钟源，细节方面请参考振荡器章节。

Bit 3 **TnON**：TMn 计数器 On/Off 控制位

0：Off

1：On

此位控制 TM 的总开关功能。设置此位为高则使能计数器使其运行，清零此位则除能 TM。清零此位将停止计数器并关闭 TM 减少耗电。当此位经由低到高转变时，内部计数器将复位清零；当此位经由高到低转换时，内部计数器将保持其剩余值，直到此位再次改变为高电平。

若 TMn 处于比较匹配输出模式时，当 TnON 位经由低到高转换时，TMn 输出脚将复位至 TnOC 位指定的初始值。

Bit 2~0 **TnRP2~TnRP0**：TMn CCRP 3-bit 寄存器，与 TMn 计数器 bit 9~bit 7 比较

比较器 P 匹配周期

000：1024 个 TMn 时钟周期
001：128 个 TMn 时钟周期
010：256 个 TMn 时钟周期
011：384 个 TMn 时钟周期
100：512 个 TMn 时钟周期
101：640 个 TMn 时钟周期
110：768 个 TMn 时钟周期
111：896 个 TMn 时钟周期

此三位设定内部 CCRP 3-bit 寄存器的值，然后与内部计数器的高三位进行比较。如果 TnCCLR 位设定为 0 时，比较结果为 0 并清除内部计数器。TnCCLR 位设为低，内部计数器在比较器 P 比较匹配发生时被重置；由于 CCRP 只与计数器高三位比较，比较结果是 128 时钟周期的倍数。CCRP 被清零时，实际上会使得计数器在最大值溢出。

TMnC1 寄存器 (n=0)

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：未定义模式
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_0，TPn_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

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_0，TPn_1 输出控制位
比较匹配输出模式
0：初始低
1：初始高
PWM 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_0，TPn_1 输出极性控制位
0：同相
1：反相
此位控制 TPn_0，TPn_1 输出脚的极性。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 (n=0)

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

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

TMnDH 寄存器 (n=0)

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

Bit 7~2 未定义，读为“0”

Bit 1~0 **D9~D8**：TMn 计数器高字节寄存器 bit 1~bit 0
TMn 10-bit 计数器 bit 9~bit 8

TMnAL 寄存器 (n=0)

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

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

TMnAH 寄存器 (n=0)

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

Bit 7~2 未定义，读为“0”

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

简易型 TM 工作模式

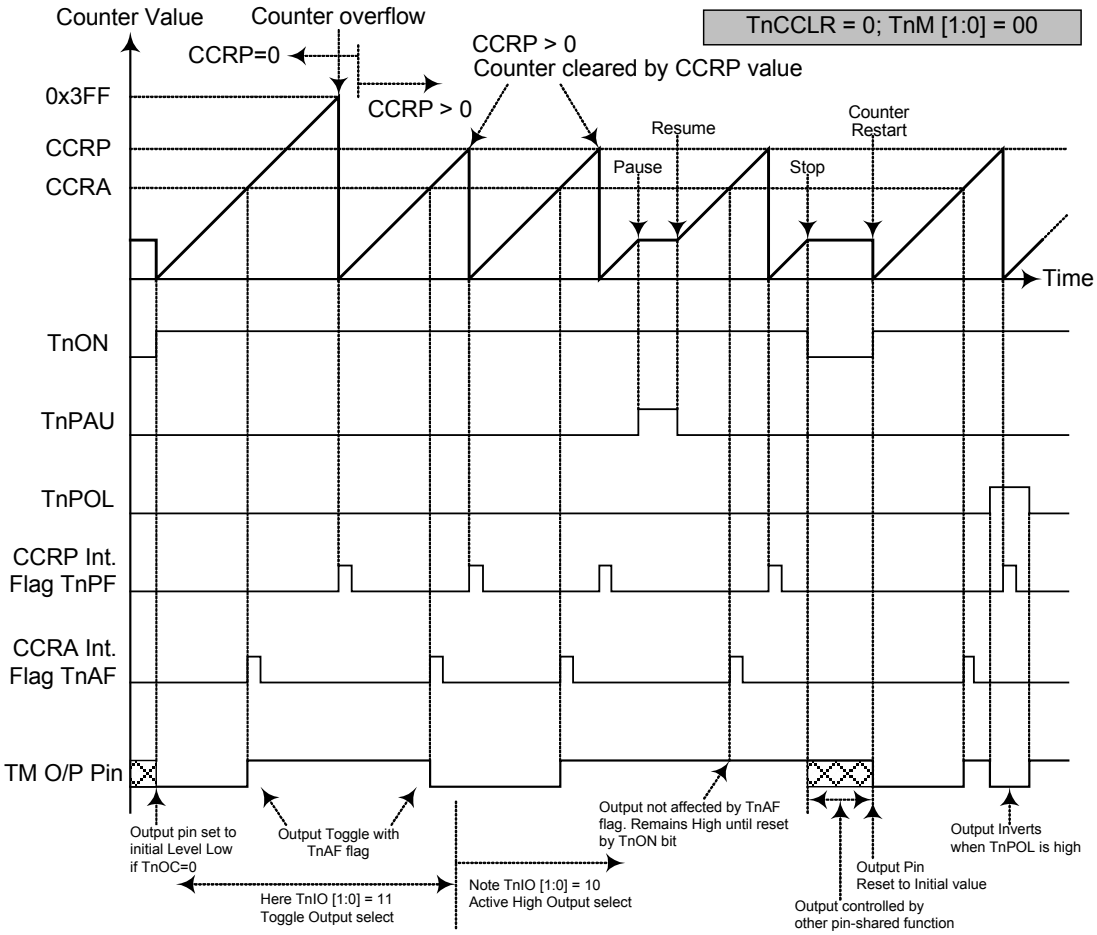
简易型 TM 有三种工作模式，即比较匹配输出模式，PWM 模式或定时 / 计数器模式。通过设置 TMnC1 寄存器的 TnM1 和 TnM0 位选择任意工作模式。

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。

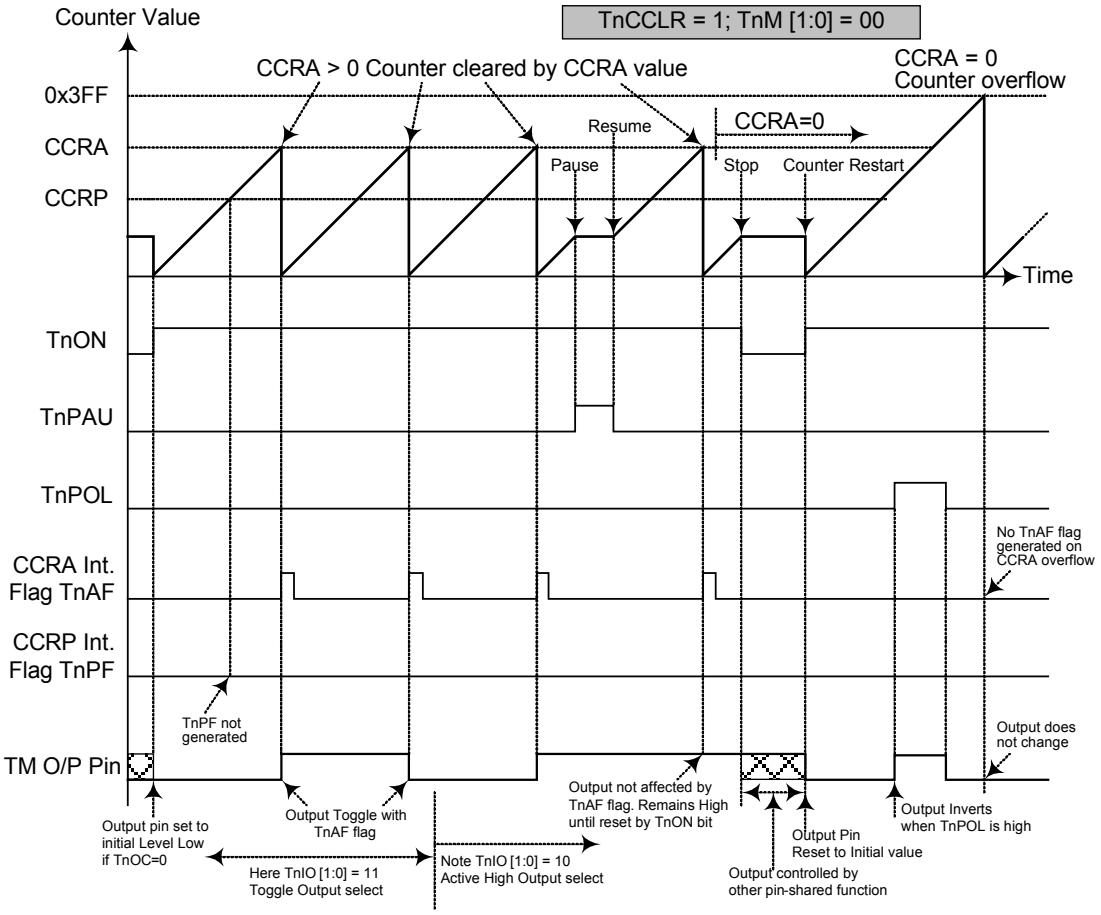
如果 TMnC1 寄存器的 TnCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 TnAF 中断请求标志产生。所以当 TnCCLR 为高时，不产生 TnPF 中断请求标志。如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 TnAF 请求标志。

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。TMn 输出脚初始值，在 TnON 位由低到高电平的变化后通过 TnOC 位设置。注意，若 TnIO1 和 TnIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 – TnCCLR=0 (n=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



比较匹配输出模式 – TnCCLR=1 (n=0)

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

定时 / 计数器模式

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

PWM 输出模式

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

由于 PWM 波形的周期和占空比可调，其波形的选择就极其灵活。在 PWM 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。

- **CTM，PWM 模式，边沿对齐模式，TnDPX=0**

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

若 f_{SYS}=16MHz，TM 时钟源选择 $f_{SYS}/4$，CCRP=100b，CCRA=128，

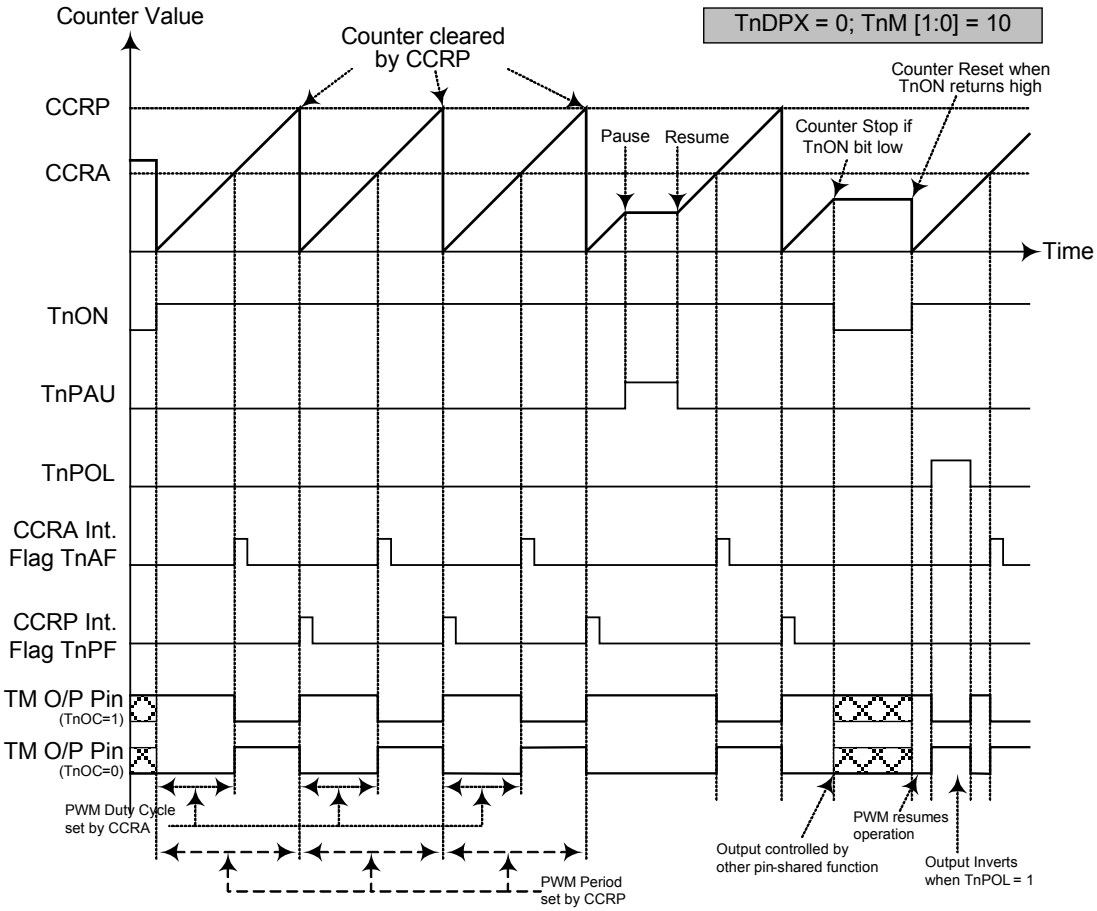
CTM PWM 输出频率 =($f_{SYS}/4$)/512=$f_{SYS}/2048$=7.8125kHz，duty=128/512=25%

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

- **CTM，PWM 模式，边沿对齐模式，TnDPX=1**

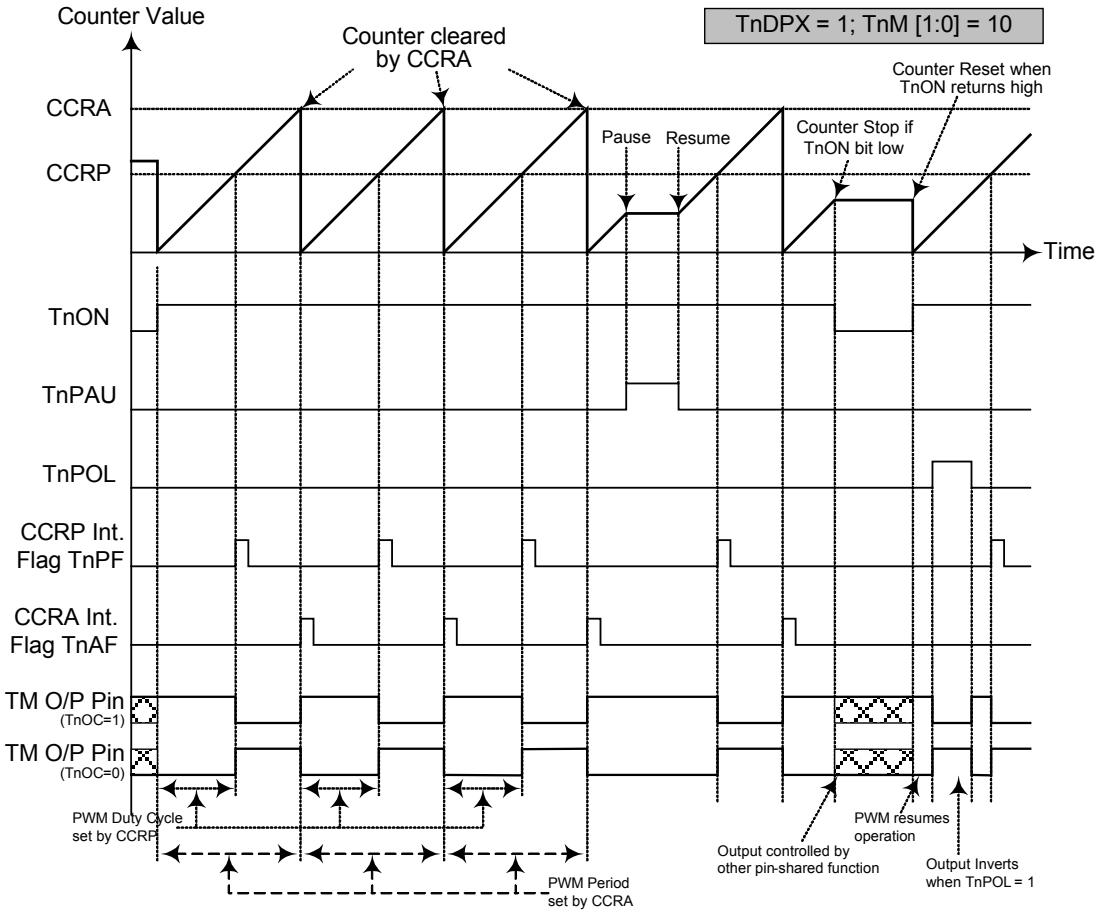
CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

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



PWM 模式 – TnDPX=0 (n=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



PWM 模式 – TnDPX=1 (n=0)

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

周期型 TM – PTM（TM1, TM2）

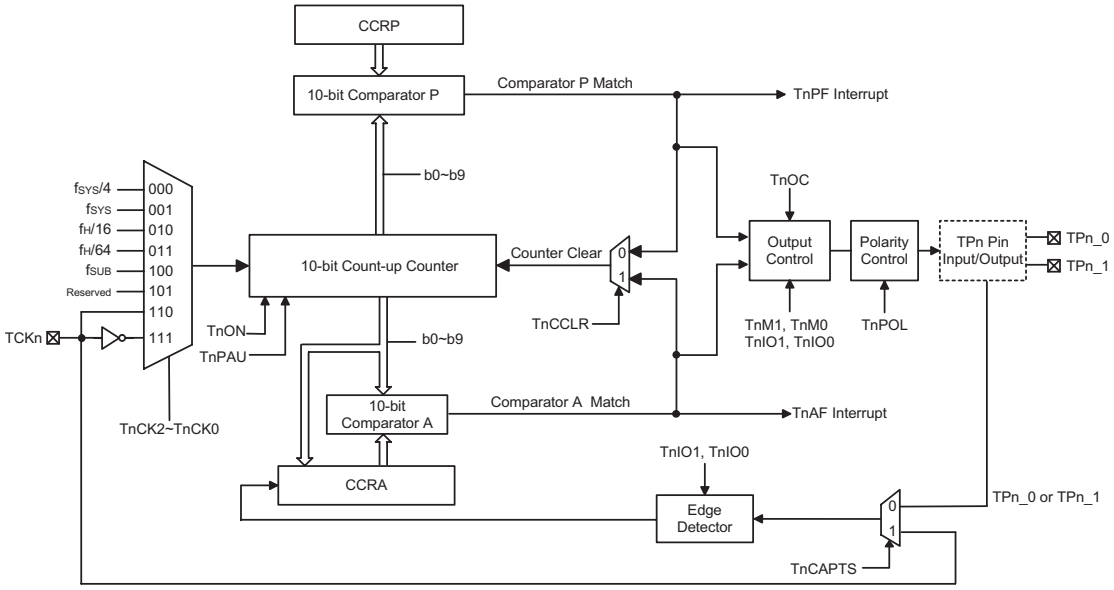
周期型 TM 包括 5 种工作模式，即比较匹配输出、定时 / 事件计数器、捕捉输入、单脉冲输出和 PWM 输出模式。周期型 TM 由一个外部输入脚控制并驱动两个外部输出脚。

名称	TM 编号	TM 输入引脚	TM 输出引脚
10-bit PTM	1, 2	TCK1, TCK2	TP1_0, TP1_1 TP2_0, TP2_1

周期型 TM 操作

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

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。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。



10-bit 周期型 TM 方框图 (n=1 或 2)

周期型 TM 寄存器介绍

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
PTMnC0	TnPAU	TnCK2	TnCK1	TnCK0	TnON	—	—	—
PTMnC1	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnCAPTS	TnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	—	—	—	—	—	—	D9	D8

10-bit 周期型 TM 寄存器列表 (n=1 或 2)

PTMnC0 寄存器 (n=1 或 2)

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

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

Bit 6 ~ 4 **TnCK2~TnCK0**：选择 TMn 计数时钟位
000：$f_{SYS}/4$
001：f_{SYS}
010：$f_H/16$
011：$f_H/64$
100：f_{SUB}
101：保留位
110：TCKn 上升沿
111：TCKn 下降沿
此三位用于选择 TM 的时钟源。选择保留时钟输入将有效地除能内部计数器。外部引脚时钟源能被选择在上升沿或下降沿有效。f_{SYS} 是系统时钟，f_H 和 f_{SUB} 是其它的内部时钟源，细节方面请参考振荡器章节。

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

Bit 2~0 未定义，读为“0”

PTMnC1 寄存器 (n=1 或 2)

Bit	7	6	5	4	3	2	1	0
Name	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnCAPTS	TnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **TnM1~TnM0**：选择 TMn 工作模式位

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 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_0 和 TPn_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 TCKn、TPn_0 和 TPn_1 上升沿输入捕捉

01：在 TCKn、TPn_0 和 TPn_1 下降沿输入捕捉

10：在 TCKn、TPn_0 和 TPn_1 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，TnIO1 和 TnIO0 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 PTMnC1 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_0 和 TPn_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 **TnPOL**：TPn_0 和 TPn_1 输出极性控制位
0：同相
1：反相
此位控制 TPn_0 和 TPn_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 **TnCAPTS**：选择 TMn 捕捉触发源
0：来自 TPn_0 或 TPn_1 引脚
1：来自 TCKn 引脚

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。周期型 TM 包括两个比较器 -- 比较器 A 和比较器 P，两者都可以用作清除内部计数器。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、单脉冲或输入捕捉模式时未使用。

PTMnDL 寄存器 (n=1 或 2)

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

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

PTMnDH 寄存器 (n=1 或 2)

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

Bit 7~2 未定义，读为“0”

Bit 1~0 **D9~D8**：TMn 计数器高字节寄存器 bit 1~bit 0
TMn 10-bit 计数器 bit 9~bit 8

PTMnAL 寄存器 (n=1 或 2)

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

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

PTMnAH 寄存器 (n=1 或 2)

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

Bit 7~2 未定义，读为“0”

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

PTMnRPL 寄存器 (n=1 或 2)

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

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

PTMnRPH 寄存器 (n=1 或 2)

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

Bit 7~2 未定义，读为“0”

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

周期型 TM 工作模式

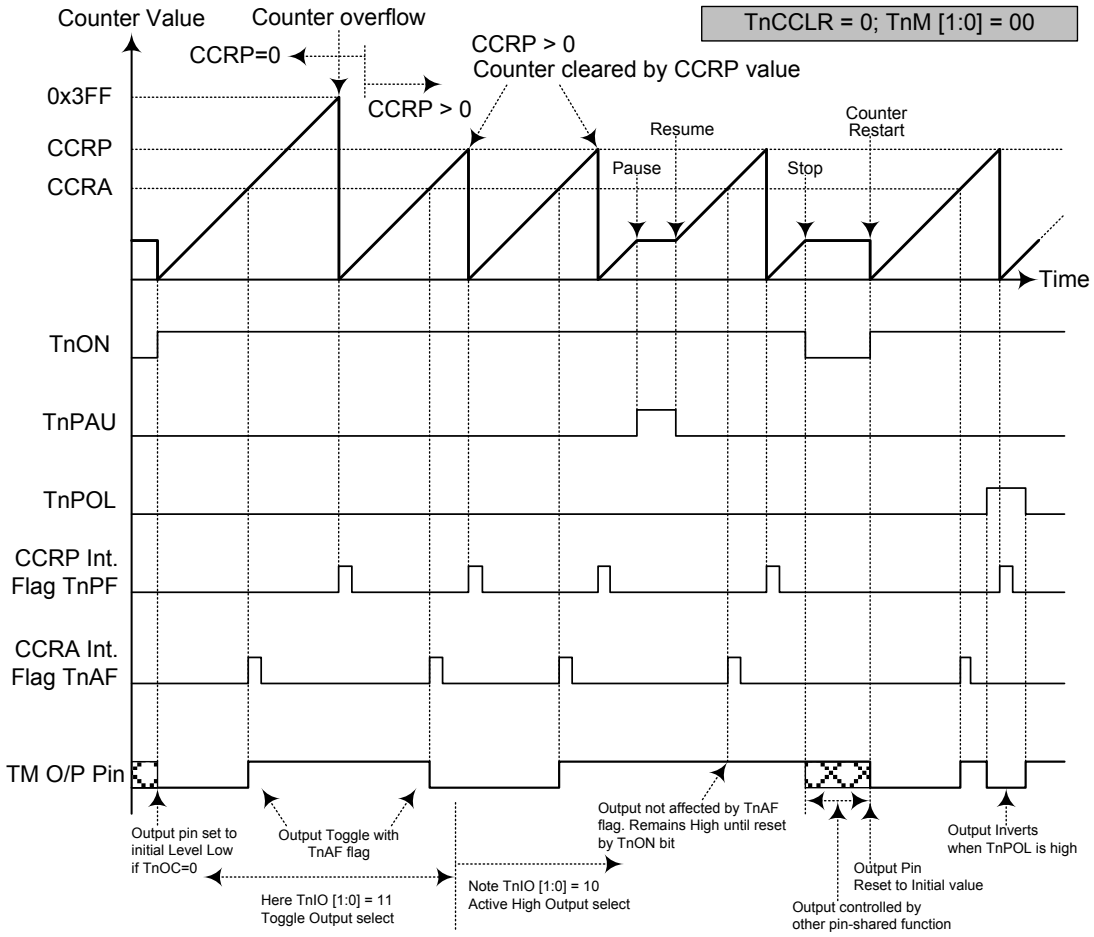
周期型 TM 有五种工作模式，即比较匹配输出模式、PWM 输出模式、单脉冲输出模式、捕捉输入模式或定时 / 计数器模式。通过设置 PTMnC1 寄存器的 TnM1 和 TnM0 位选择任意模式。

比较匹配输出模式

为使 TM 工作在此模式，PTMnC1 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。

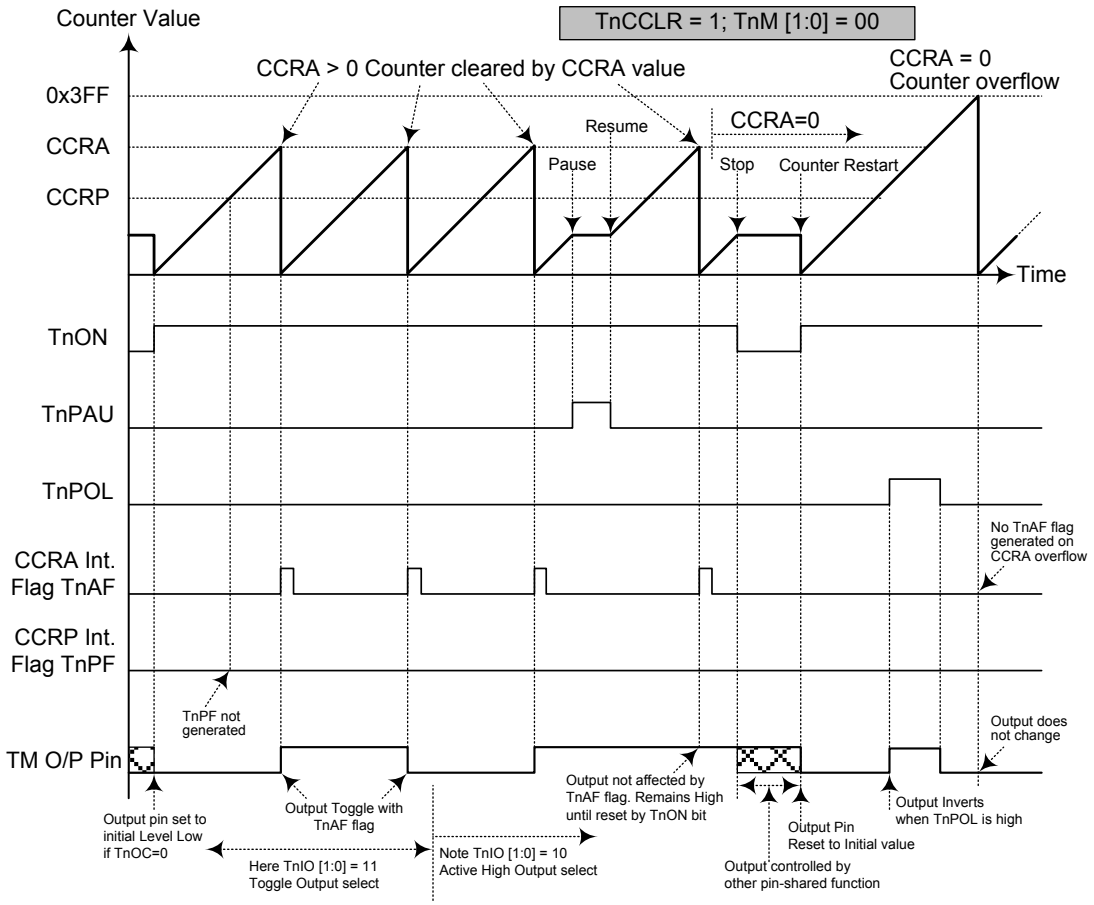
如果 PTMnC1 寄存器的 TnCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 TnAF 中断请求标志产生。所以当 TnCCLR 为高时，不会产生 TnPF 中断请求标志。在比较匹配输出模式中，CCRA 寄存器值不能设为“0”。

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 TnAF 中断请求标志产生时，TM 输出脚状态改变。比较器 P 比较匹配发生时产生的 TnPF 标志不影响 TM 输出脚。TM 输出脚状态改变方式由 PTMnC1 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。TMn 输出脚初始值，在 TnON 位由低到高电平的变化后通过 TnOC 位设置。注意，若 TnIO1 和 TnIO0 位同时为 0 时，引脚输出不变。



比较器匹配输出模式 – TnCCLR = 0 (n=1 或 2)

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



比较器匹配输出模式 – TnCCLR = 1 (n=1 或 2)

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

定时 / 计数器模式

为使 TM 工作在此模式，PTMnC1 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，PTMnC1 寄存器中的 TnM1 和 TnM0 位需要设置为“10”，且 TnIO1 和 TnIO0 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，其波形的选择就极其灵活。在 PWM 模式中，TnCCLR 位对 PWM 周期无影响。CCRP 和 CCRA 寄存器都用于控制 PWM 方波。CCRP 寄存器通过清除内部计数从而控制 PWM 周期，CCRA 寄存器设置 PWM 的占空比。PWM 波形的周期和占空比由 CCRP 和 CCRA 寄存器的值控制。

当比较器 A 或比较器 P 比较匹配发生时，CCRA 和 CCRP 中断标志位分别产生。PTMnC1 寄存器的 TnOC 位选择 PWM 波形的极性，TnIO1 和 TnIO0 位使能 PWM 输出或强制 TM 输出脚为高电平或低电平。TnPOL 位用于 PWM 输出波形的极性反相控制。

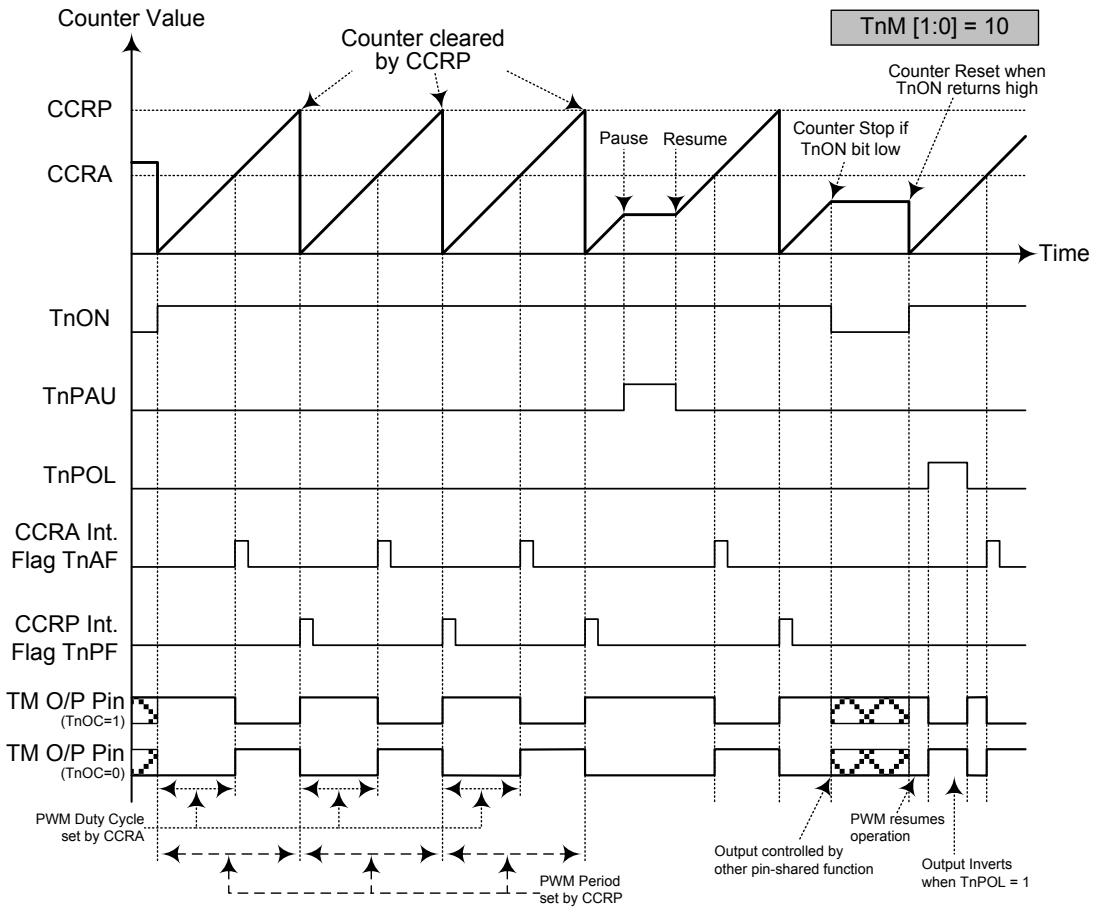
- **10-bit PTM，PWM 模式，边沿对齐模式**

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

若 f_{SYS}=16MHz，TM 时钟源选择 f_{SYS}/4，CCRP=100b 且 CCRA=128，

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

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



PWM 模式 (n=1 或 2)

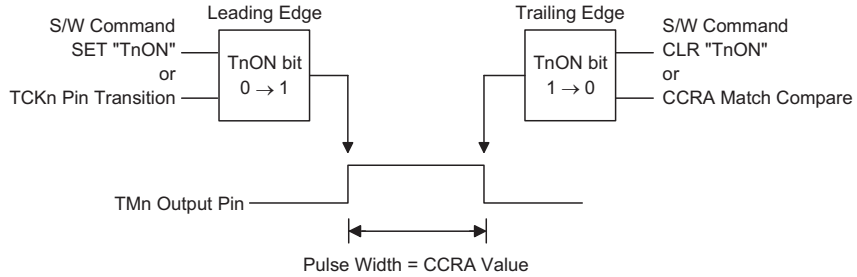
注： 1. CCRP 清除计数器
2. 计数器清除并决定 PWM 周期
3. 当 TnIO[1:0]=00 或 01，PWM 功能不变
4. TnCCLR 位对 PWM 功能无影响

单脉冲输出模式

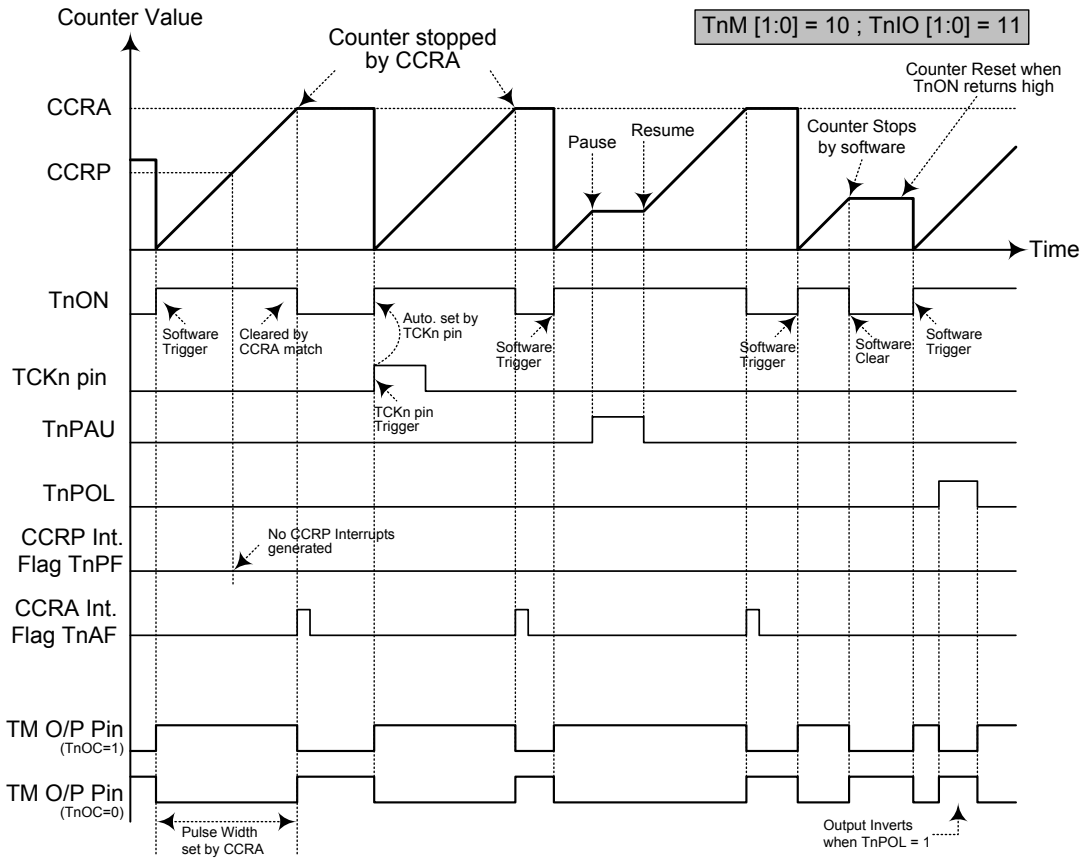
为使 TM 工作在此模式，PTMnC1 寄存器中的 TnM1 和 TnM0 位需要设置为“10”，并且相应的 TnIO1 和 TnIO0 需要设置为“11”。正如模式名所言，单脉冲输出模式，在 TM 输出脚将产生一个脉冲输出。

通过应用程序控制 TnON 位由低到高的转变来触发脉冲前沿输出。而处于单脉冲模式时，TnON 位可在 TCKn 引脚发生有效边沿跳转时自动由低转变为高，进而开始单脉冲输出。当 TnON 位转变为高电平时，计数器将开始运行，并产生脉冲前沿。通过应用程序使 TnON 位清零或比较器 A 比较匹配发生时，产生脉冲后沿。

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



单脉冲产生示意图 (n=1 或 2)



单脉冲模式 (n=1 或 2)

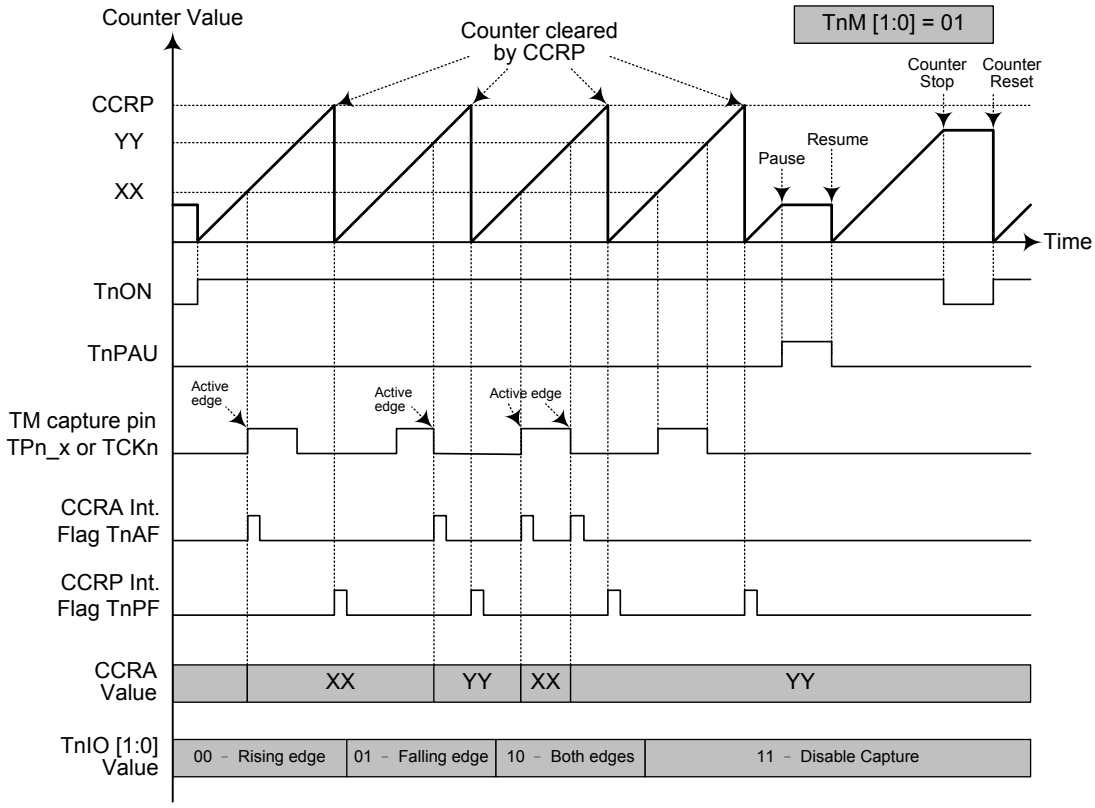
注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 TCKn 脚或设置 TnON 位为高来触发脉冲
4. TCKn 脚有效沿会自动置位 TnON
5. 单脉冲模式中，TnIO[1:0] 需置为“11”，且不能更改。

捕捉输入模式

为使 TM 工作在此模式，PTMnC1 寄存器中的 TnM1 和 TnM0 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。TPn_0、TPn_1 或 TCKn 引脚上的外部信号，通过设置 PTMnC1 寄存器的 TnCAPTS 位选择。可通过设置 PTMnC1 寄存器的 TnIO1 和 TnIO0 位选择有效边沿类型，即上升沿，下降沿或双沿有效。通过应用程序将 TnON 位由低到高转变时，计数器启动。

当 TPn_0、TPn_1 或 TCKn 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。无论 TPn_0、TPn_1 或 TCKn 引脚发生哪种边沿转换，计数器将继续工作直到 TnON 位发生下降沿跳变。当 CCRP 比较匹配发生时计数器复位至零；CCRP 的值通过这种方式控制计数器的最大值。当比较器 P CCRP 比较匹配发生时，也会产生 TM 中断。记录 CCRP 溢出中断信号的值可以测量长脉宽。通过设置 TnIO1 和 TnIO0 位选择 TPn_0、TPn_1 或 TCKn 引脚为上升沿，下降沿或双沿有效。如果 TnIO1 和 TnIO0 位都设置为高，无论 TPn_0、TPn_1 或 TCKn 引脚发生哪种边沿转换都不会产生捕捉操作，但计数器仍会继续运行。

当 TPn_0、TPn_1 或 TCKn 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。TnCCLR，TnOC 和 TnPOL 位在此模式中未使用。

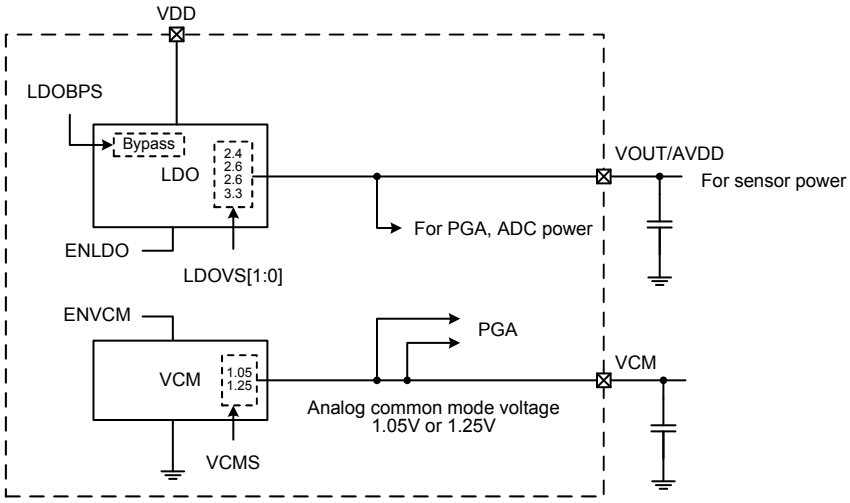


捕捉输入模式 (n=1 或 2)

注： 1. TnM[1:0]=01 并通过 TnIO[1:0] 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

内部电源

该单片机内部集成一个 LDO 和一个 VCM 模块，用于产生稳定的电源电压，其基本功能操作如下图所示。内部的 LDO 电路为 PGA、ADC 或外部器件提供了一个固定电压。VCM 还可以作为 ADC 模块的参考电压。LDO 可提供 2.4V、2.6V、2.9V 或 3.3V 四个输出电压，通过 PWRC 寄存器中的 LDOVS1~LDOVS0 位选择。VCM 可提供 1.05V 或 1.25V 两个输出电压，通过 PGAC1 寄存器中的 VCMS 位选择。LDO 和 VCM 功能分别由 ENLDO 位和 ENVCM 位控制，可选择关闭以降低功耗。另外还可通过 PWRC 寄存器中的 LDOBPS 位控制 LDO 旁路功能的使能 / 除能。



内部电压方框图

寄存器的相关位			输出电压	
ADOFF	**ENLDO**	**ENVCM**	**VOUT/AVDD**	**VCM**
1	0	x	除能	除能
1	1	x	使能	除能
0	0	0	除能	除能
0	1	0	使能	除能
0	0	1	除能	除能
0	1	1	使能	使能

“x”为未知

电源控制表

PWRC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ENLDO	ENVCM	—	—	—	LDOBPS	LDOVS1	LDOVS0
R/W	R/W	R/W	—	—	—	R/W	R/W	R/W
POR	0	0	—	—	—	0	0	0

Bit 7 **ENLDO**：LDO 功能控制位
0：除能
1：使能
如果 LDO 除能，将不产生功耗，LDO 输出脚会处于浮空状态。

Bit 6 **ENVCM**：VCM 功能控制位
0：除能
1：使能
如果 VCM 除能，将不产生功耗，VCM 输出脚会处于浮空状态。

Bit 5~3 未定义，读为“0”

Bit 2 **LDOBPS**：LDO 旁路功能控制位
0：除能
1：使能

Bit 1~0 **LDOVS1~LDOVS0**：LDO 输出电压选择位
00：2.4V
01：2.6V
10：2.9V
11：3.3V

PGAC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	VCMS	INIS	—	—	DCSET2	DCSET1	DCSET0	—
R/W	R/W	R/W	—	—	R/W	R/W	R/W	—
POR	1	0	—	—	0	0	0	—

Bit 7 **VCMS**：A/D 转换器共模电压选择位
0：1.05V
1：1.25V

Bit 6 **INIS**：IN1 和 IN2 输入端连接控制位
详见其它章节

Bit 5~4 未使用，读为“0”

Bit 3~1 **DCSET2~DCSET0**：DI+/DI- 差分输入偏置电压调整控制位
详见其它章节

Bit 0 未使用，读为“0”

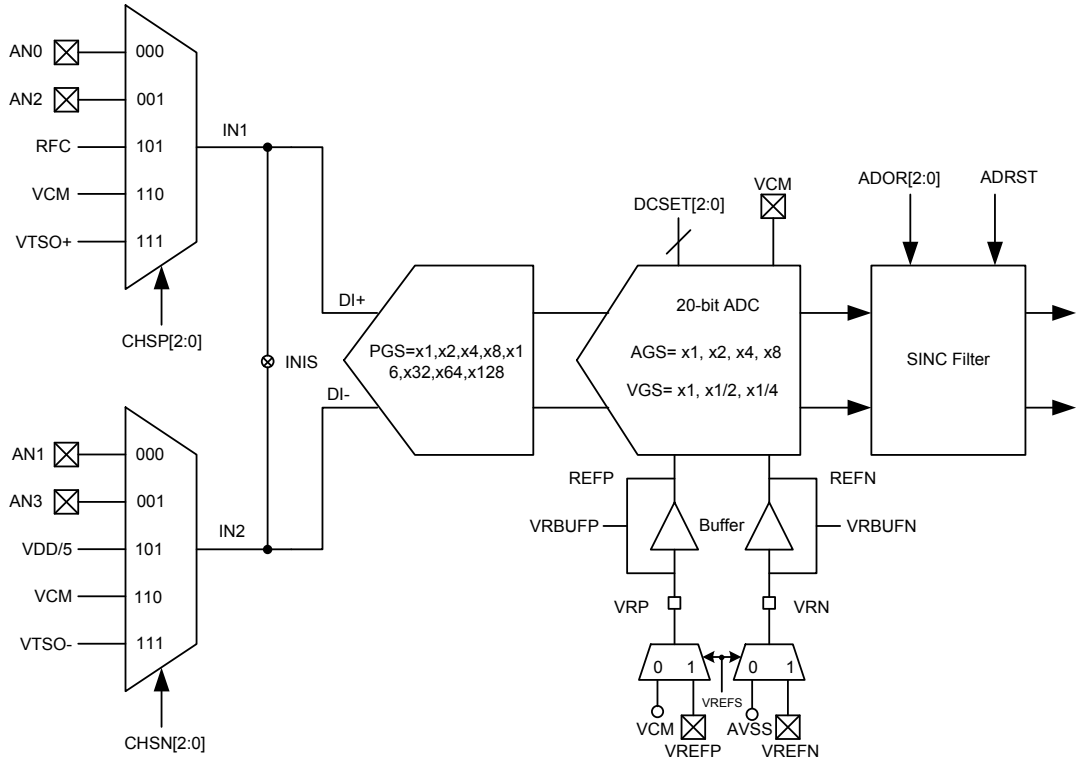
A/D 转换器

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

A/D 简介

此单片机包含一个多通道的 20 位 delta-sigma 型 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 20 位的数字量。

另外，ADC 输入信号的放大增益由 PGA 增益控制、ADC 增益控制和 ADC 参考电压增益控制共同确定。设计者可以选择最佳增益组合为输入信号提供所需的放大增益。下面的方框图说明了 A/D 转换器的基本操作功能。A/D 转换器输入通道由两组差分输入通道组成。在 PGA 进入 20 位 delta-sigma 型 A/D 转换器之前，输入信号被放大。ΔΣA/D 转换调制器将 1-bit 转换后的数据输出到 SINC 滤波器，然后会转换成 20-bit 的数据，并将它们存储到特殊数据寄存器。此外，单片机还提供了一个温度传感器来补偿的 A/D 转换器由温度引起的偏差。这种高精度和高性能的特点，使得该单片机非常适用于体重秤相关产品。



A/D 转换器结构

A/D 数据传输率的定义

Delta-Sigma 型 A/D 转换器的数据传输率可以通过下面的公式计算：

数据传输率 = ADC 时钟 / (FLMS[2:0] × ADOR[2:0])

其中，ADC 时钟来自 f_{MCLK}；FLMS[2:0] 用于选择 ADC 模式和定义 ADC 时钟的分频比，只能为 30 或 12；ADOR[2:0] 定义斩波平均值 CHOP 和过采样率 OSR。

例如，如果需要一个 10Hz 的数据传输率，可以选择一个 4.8MHz 的 ADC 时钟，然后设置 FLMS[2:0]=000b，即 ADC 输出处于正常模式且时钟分频比为 30，最后设置 ADOR[2:0]=001b，则 CHOP=2，OSR=8192。因此，数据传输率 =4.8MHz / (30 × 2 × 8192) = 10Hz。

此外，A/D 转换器还提供一个 3.2kHz 的数据传输率用于自动上秤。

A/D 转换寄存器介绍

A/D 转换器的所有工作由 9 个寄存器控制。3 个只读寄存器来存放 20 位 ADC 数据的值。剩下 6 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
PGAC0	—	VGS1	VGS0	AGS1	AGS0	PGS2	PGS1	PGS0
PGAC1	VCMS	INIS	—	—	DCSET2	DCSET1	DCSET0	—
PGACS	—	—	CHSN2	CHSN1	CHSN0	CHSP2	CHSP1	CHSP0
ADRL	D7	D6	D5	D4	D3	D2	D1	D0
ADRM	D15	D14	D13	D12	D11	D10	D9	D8
ADRH	—	—	—	—	D19	D18	D17	D16
ADCR0	ADRST	ADSLP	ADOFF	ADOR2	ADOR1	ADOR0	—	VREFS
ADCR1	FLMS2	FLMS1	FLMS0	VRBUFN	VRBUFP	ADCDL	EOC	—
ADCS	—	—	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0

A/D 转换寄存器列表

可编程增益放大器 – PGA

有三个与可编程增益相关的控制寄存器，PGAC0、PGAC1 和 PGACS。PGAC0 寄存器用于选择 PGA 增益、ADC 增益和 ADC 参考电压增益。PGAC1 寄存器用于定义输入端连接、差分输入偏置电压调整控制和 V_{CM} 电压选择。PGACS 寄存器用于选择 PGA 的输入端。因此，必须通过 CHSP2~CHSP0 和 CHSN2~CHSN0 位来选择模拟输入通道、RFC 引脚、温度检测输入或内部电源中的哪些被连接到内部差分 A/D 转换器。

PGAC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	VGS1	VGS0	AGS1	AGS0	PGS2	PGS1	PGS0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7　未使用，读为“0”

Bit 6~5　**VGS1~VGS0**：V_{REF} 增益选择位
00：1
01：1/2
10：1/4
11：保留位

Bit 4~3　**AGS1~AGS0**：ADC 增益选择位
00：1
01：2
10：4
11：8

Bit 2~0　**PGS2~PGS0**：PGA 增益选择位
000：1
001：2
010：4
011：8
100：16
101：32
110：64
111：128

PGAC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	VCMS	INIS	—	—	DCSET2	DCSET1	DCSET0	—
R/W	R/W	R/W	—	—	R/W	R/W	R/W	—
POR	1	0	—	—	0	0	0	—

Bit 7　**VCMS**：A/D 转换器共模电压选择位
0：1.05V
1：1.25V

Bit 6　**INIS**：IN1 和 IN2 输入端连接控制位
0：不短接
1：短接

Bit 5~4　未使用，读为“0”

Bit 3~1　**DCSET2~DCSET0**：DI+/DI- 差分输入偏置电压调整控制位
000：+0V
001：+0.25V_R
010：+0.5V_R
011：+0.75V_R
100：+0V
101：-0.25V_R
110：-0.5V_R
111：-0.75V_R

Bit 0　未使用，读为“0”

PGACS 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	CHSN2	CHSN1	CHSN0	CHSP2	CHSP1	CHSP0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6　未使用，读为“0”

Bit 5~3　**CHSN2~CHSN0**：PGA 输入负端选择位

000：AN1
001：AN3
010：保留位
011：保留位
100：保留位
101：VDD/5
110：VCM
111：温度传感器 VTSO-

Bit 2~0　**CHSP2~CHSP0**：PGA 输入正端选择位

000：AN0
001：AN2
010：保留位
011：保留位
100：保留位
101：RFC—RFC 为单端输入脚，如果 PGA 正端选择了 RFC 引脚，则 PGA 负端输入必须选择 VCM、AN1 或 AN3
110：VCM
111：温度传感器 VTSO+

注：若给 PGA 的 DI+ 端分配了单端输入脚，那么 DI- 输入端需选择 VCM。

A/D 转换器数据寄存器 – ADRL, ADRM, ADRH

对于具有 20 位 ΔΣA/D 转换器的单片机，需要 3 个数据寄存器存放转换结果，两个高字节寄存器 ADRH、ADRM 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。D0~D19 是 A/D 换转数据结果位。

ADRH 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D19	D18	D17	D16
R/W	—	—	—	—	R	R	R	R
POR	—	—	—	—	x	x	x	x

“x”为未知

Bit 7~4　未使用，读为“0”

Bit 3~0　A/D 数据寄存器 bit 19~bit 16

ADRM 寄存器

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

“x”为未知

Bit 7~0　A/D 数据寄存器 bit 15~bit 8

ADRL 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	x	x	x	x	x	x	x	x

“x”为未知

Bit 7~0　A/D 数据寄存器 bit 7~bit 0

A/D 转换控制寄存器 – ADCR0, ADCR1, ADCS

寄存器 ADCR0、ADCR1 和 ADCS 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内部 A/D 转换器的参考源，A/D 时钟源，A/D 输出数据传输率，并控制和监视 A/D 转换器的开始和转换结束状态等。

ADCR0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ADRST	ADSLP	ADOFF	ADOR2	ADOR1	ADOR0	—	VREFS
R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	R/W
POR	0	0	1	0	0	0	—	0

Bit 7　**ADRST**：ADC 软件复位控制位
0：除能
1：使能
此位用来复位 ADC 内部数字 SINC 滤波器。通常此位为低，但如果设为高再被清零，将初始化 A/D 转换过程。

Bit 6　**ADSLP**：ADC 休眠模式控制位
0：正常模式
1：休眠模式
此位用于控制 ADC 休眠模式。该位被置高将迫使 ADC 进入休眠模式，这样可以减少功耗并缩短 ADC 启动时间。

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. 无论 ADSLP 和 ADRST 位如何设置，ADOFF=1 将关闭 ADC 模块的电源。
3. ADOFF、ADSLP 和 ADRST 位的关系将在 A/D 操作中描述。

Bit 4~2 **ADOR2~ADOR0**：输出数据传输率选择
正常模式：输出数据传输率
000：CHOP=2，OSR=16384
001：CHOP=2，OSR=8192
010：CHOP=2，OSR=4096
011：CHOP=2，OSR=2048
100：CHOP=2，OSR=1024
101：CHOP=2，OSR=512
110：CHOP=2，OSR=256
111：CHOP=2，OSR=128
低延迟模式：输出数据传输率
000：CHOP=1，OSR=16384
001：CHOP=1，OSR=8192
010：CHOP=1，OSR=4096
011：CHOP=1，OSR=2048
100：CHOP=1，OSR=1024
101：CHOP=1，OSR=512
110：CHOP=1，OSR=256
111：CHOP=1，OSR=128

Bit 1 未定义，读为“0”

Bit 0 **VREFS**：选择 ADC 参考电压
0：内部参考电压 (VCM，AVSS)
1：外部参考电压 (VREFP，VREFN)

ADCR1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	FLMS2	FLMS1	FLMS0	VRBUFN	VRBUFP	ADCDL	EOC	—
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 **FLMS2~FLMS0**：ADC 输出数据模式和时钟分频比选择
000：正常模式，ADC 时钟 /30
010：正常模式，ADC 时钟 /12
100：低延迟模式，ADC 时钟 /30
110：低延迟模式，ADC 时钟 /12
其它值：保留位

Bit 4 **VRBUFN**：VRN 缓存器使能位
0：除能
1：使能

Bit 3 **VRBUFP**：VRP 缓存器使能位
0：除能
1：使能

Bit 2 **ADCDL**：A/D 转换数据锁存功能控制
0：除能
1：使能
如果使能 A/D 转换数据锁存功能，最新转换的数据将被锁存，且不会更新后面的转换结果直到该功能被除能。虽然转换后的数据被锁存到数据寄存器，A/D 转换电路仍正常运行，但并不产生中断，EOC 也不改变。建议在读取 ADRL、ADRM 和 ADRH 寄存器中的转换数据之前先将该位置高。读取后该位会被清零以除能 A/D 数据锁存功能，以便下一笔转换结果的存储。这样可以防止在 A/D 转换过程中得到不需要的数据。

Bit 1 **EOC**：A/D 转换结束标志
0：A/D 转换中
1：A/D 转换结束
此位必须通过软件清除。

Bit 0 未定义，读为“0”

ADCS 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5　未定义，读为“0”

Bit 4~0　**ADCK4~ADCK0**：选择 ADC 时钟源 (f_{MCLK})

00000~11110：f_{SYS}/2/(ADCK[4:0]+1)

11111：f_{SYS}

由于 ADC 的时钟源 f_{MCLK} 通常设计为 4.8MHz，单片机可能会选择在不同的系统时钟下工作，因此，设计者应通过设置 ADCK4~ADCK0 位以获得固定 4.8MHz 的 ADC 工作时钟。例如，若系统时钟为 9.6MHz，ADCK[4:0] 需设为 0 以使 f_{MCLK} =4.8MHz。

A/D 操作

该 A/D 转换器提供了三种工作模式，暂停模式、休眠模式和复位模式，分别由 ADCR0 寄存器中的 ADOFF、ADSLP 和 ADRST 位控制。下表列出了工作模式的选择。

ADOFF	ADSLP	ADRST	工作模式	说明
1	x	x	暂停模式	PGA 关闭，ADC 关闭
0	1	x	休眠模式	PGA 开启，ADC 关闭
0	0	1	复位模式	PGA 开启，ADC 开启，SINC 复位

“x”为未知

A/D 工作模式选择

要打开 A/D 转换器，首先应除能 A/D 转换器的暂停和休眠模式，以确保 A/D 转换器可以通电。ADCR0 寄存器中的 ADRST 位，用于上电后打开和复位 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，一个模数转换后的数据就会开始在 SINC 滤波器中进行转换。设置完成后，A/D 转换器可以开始工作。这三位用于控制内部模数转换器的开启动作。

ADCR1 寄存器中的 EOC 位用于表明模数转换过程的完成。在转换周期结束后，EOC 位会被单片机自动地置为“1”。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 EOC 位，检查此位是否被置高，以作为另一种侦测 A/D 转换周期结束的方法。A/D 转换数据将不断更新，如果 A/D 转换数据锁存功能使能，最新的转换数据会被锁存，这样后面再转换的数据不会被保存，直到该功能被关闭。

A/D 转换器的时钟源通常固定在 4.8MHz，来自系统时钟 f_{SYS} 或其分频，分频系数由 ADCS 寄存器中的 ADCK4~ADCK0 位决定，以获得固定 4.8MHz 的 ADC 时钟源。

A/D 转换器参考电压来自内部电源电压引脚 VCM 和 AVSS 或外部参考源引脚 VREFP 和 VREFN，可通过 ADCR0 寄存器的 VREFS 位来选择。

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
  使能 LDO 和 VCM，以提供电源给 PGA 和 ADC。
- 步骤 2
  通过 PGAC0 寄存器，选择 PGA、ADC 和 V_{REF} 的增益。
- 步骤 3
  通过 PGAC1 寄存器，选择 PGA 的输入引脚连接和 V_{CM} 选项。
- 步骤 4
  通过 ADCS 寄存器中的 ADCK4~ADCK0 位，选择所需的 4.8MHz A/D 转换时钟。
- 步骤 5
  通过 ADCR0 寄存器中的 ADOR2~ADOR0 位，选择输出数据传输率。
- 步骤 6
  通过 PGACS 寄存器中的 CHSP2~CHSP0 和 CHSN2~CHSN0 位，选择连接至内部 PGA 的通道。
- 步骤 7
  通过 ADCR0 寄存器中的 ADOFF 和 ADSLP 位，关闭暂停和休眠模式。
- 步骤 8
  通过置高 ADCR0 寄存器中的 ADRST 位来复位 A/D 转换器，清除该位来释放复位状态。
- 步骤 9
  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 转换中断功能是激活的。总中断控制位 EMI 需要置位为“1”，以及 A/D 转换器中断位 ADE 也需要置位为“1”。
- 步骤 10
  可以轮询 ADCR1 寄存器中的 EOC 位，检查模数转换过程是否完成。当此位成为逻辑高时，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、ADRM 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 ADCR1 寄存器中 EOC 位的状态的方法来检查转换过程是否结束时，则中断使能的步骤可以省略。

编程注意事项

在编程时，如果 A/D 转换器未使用，通过设置 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。

A/D 转换功能

单片机含有一组 20 位的 ΔΣA/D 转换器，它的转换范围为 -524288~524287（十进制）。转换后的数据以二进制补码的形式表示，最高位是转换数据的符号位。由于模拟输入最大值等于 V_{CM} 或 ΔV_{REF} 的电压值，因此每一位可表示 (V_{CM} 或 ΔV_{REF})/524288 的模拟输入值。

$$1\ LSB = (V_{CM} 或\ \Delta V_{REF}) \div 524288$$

通过下面的等式可估算 A/D 转换后的数据：

$$\Delta SI\_I = (PGAGN \times ADGN \times \Delta DI\pm) + (DCSET \times \Delta VR\_I)$$

$$\Delta VR\_I = VREGN \times \Delta VR\pm$$

$$A/D\ 转换数据 = (\Delta SI\_I \div \Delta VR\_I) \times K$$

其中，$K=2^{19}$

注：PGAGN、ADGN 和 VREGN 的值由 PGS、AGS、VGS 控制位决定。

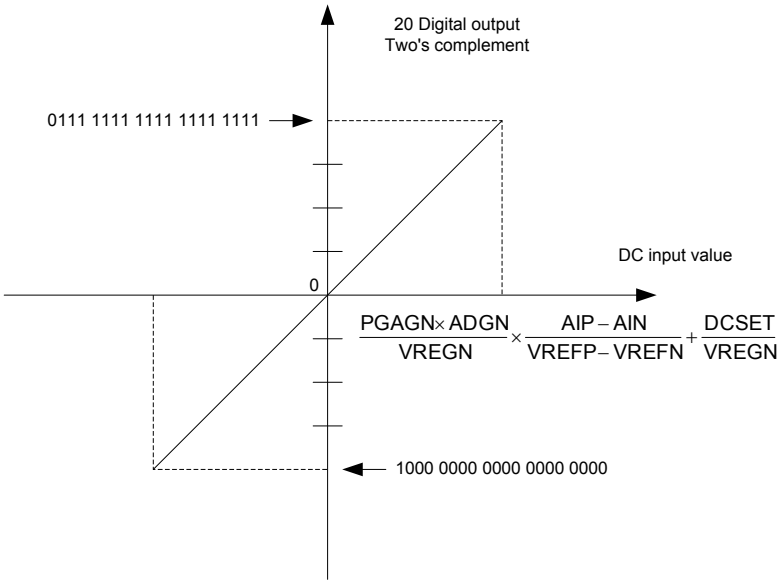
ΔSI_I：处理后的差分输入信号
PGAGN：PGA 增益
ADGN：ADC 增益
ΔDI±：差分输入信号
DCSET：偏置电压
ΔVR±：差分参考电压
ΔVR_I：处理后的差分参考输入电压
VREGN：参考电压增益

由于数字系统设计的 ΔΣA/D 转换器，其转换的最大值为 524287，最小值为 -524288，因此有一个中间值 0。A/D 转换数据公式说明了转换值的变化范围。

A/D 转换数据 （二进制补码，十六进制值）	十进制值
0x7FFFF	524287
0x80000	-524288

上面的 A/D 转换数据表说明了 A/D 转换值的范围。

下图显示直流输入电压值和A/D转换数据（以二进制补码形式表示）之间的关系。



A/D 转换数据

A/D 转换数据与输入电压和 PGA 的选择有关。A/D 转换输出数据以二进制补码的形式表示，代码的长度为 20 位，最高位为符号位。最高位“0”表示输出为正数，最高位“1”表示输出为负数。所以最大值是 524287，最小值是 -524288。如果输入信号大于最大值，转换后的数据上限为 524287；如果输入信号小于最小值，转换后的数据下限为 -524288。

A/D 转换数据转为电压值

设计者可以通过下面的公式来恢复转换后的数据。

如果 MSB=0（正转换数据）：

输入电压 = (转换数据 -0) × (LSB/PGA)

如果 MSB=1（负转换数据）：

输入电压 = (转换数据的补码 -0) × (LSB/PGA)

注：补码 = 反码 +1

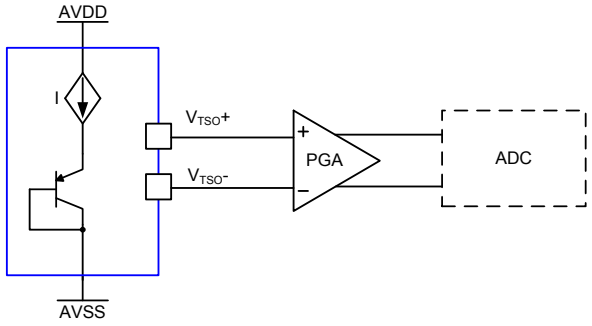
A/D 转换应用范例

范例：使用查询 EOC 的方式来检测转换结束

```
#include    ht45f75.inc
data .section       'data'
adc_result_data_l db?
adc_result_data_m db?
adc_result_data_h db?
code .section 'code'
start:
clr ADE                     ; disable ADC interrupt
mov a, 0C3H                 ; Power control for PGA, ADC
mov PWRC, a                 ; PWRC=11000011, LDO enable, VCM enable,
                            ; LDO Bypass disable, LDO output voltage: 3.3V
mov a, 000H
mov PGAC0, a                ; PGA gain=1, ADC gain=1, VREF gain=1
mov a, 080H
mov PGAC1, a                ; VCM=1.25V, INIS, DCSET2~0 in default value
set VRBUFP                  ; enable buffer for VREF+
set VRBUFN                  ; enable buffer for VREF-
set VREFS                   ; for using external reference
clr ADOR2                   ; for 10Hz output data rate, ADOR[2:0]=001,
                            ; FLMS[2:0]=000
clr ADOR1
set ADOR0
clr FLMS2
clr FLMS1
clr FLMS0
clr ADOFF                   ; ADC exit power down mode.
set ADRST                   ; ADC in reset mode
clr ADRST                   ; ADC in conversion (continues mode)
clr EOC                     ; Clear "EOC" flag
loop:
snz EOC                     ; Polling "EOC" flag
jmp loop                    ; Wait for read data
clr adc_result_data_h
clr adc_result_data_m
clr adc_result_data_l
mov a, ADRL
mov adc_result_data_l, a    ; Get Low byte ADC value
mov a, ADRM
mov adc_result_data_m, a    ; Get Middle byte ADC value
mov a, ADRH
mov adc_result_data_h, a    ; Get High byte ADC value
get_adc_value_ok:
clr EOC                     ; Clearing read flag
jmp loop                    ; for next data read
end
```

温度传感器

该单片机提供了一个内部温度传感器以补偿温度对其的影响。将 PGA 输入通道连接到 VTSO+ 和 VTSO-，A/D 转换器可以获得温度信息，设计者可以根据 A/D 转换结果做一些调整。下图说明了温度传感器的功能操作。



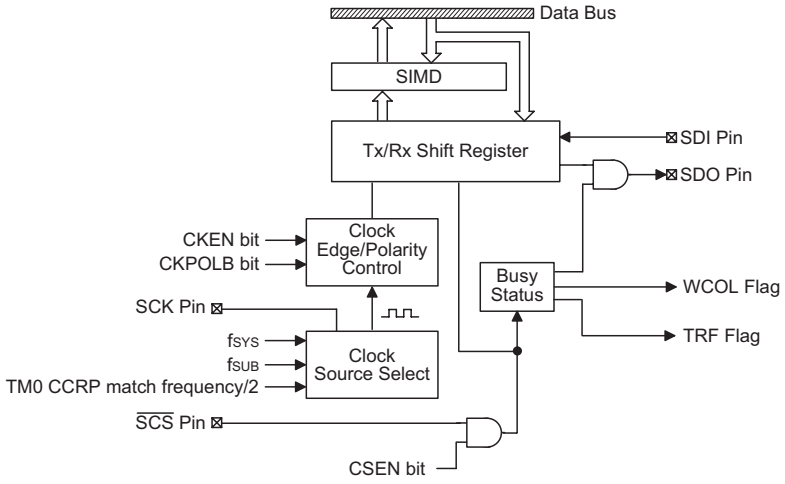
串行接口模块 – SIM

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

SPI 接口

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

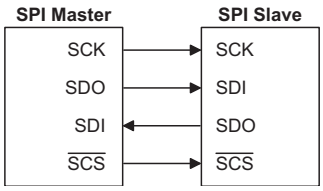
SPI 通信模式为全双工模式，且能以主 / 从模式的工作方式进行通信，单片机既可以做为主机，也可以做为从机。虽然 SPI 接口理论上允许一个主机控制多个从机，但此处的 SPI 中只有一个片选信号引脚。若主机需要控制多个从机，可使用输入 / 输出引脚选择从机。



SPI 方框图

SPI 接口操作

SPI 接口是一个全双工串行数据传输器。SPI 接口的四线为：SDI、SDO、SCK 和 $\overline{SCS}$。SDI 和 SDO 是数据的输入和输出线。SCK 是串行时钟线，$\overline{SCS}$ 是从机的选择线。SPI 的接口引脚与普通 I/O 口和 I^2C 的功能脚共用。通过设定 SIMC0/SIMC2 寄存器的对应位，来使能 SPI 接口。SPI 可以通过 SIMC0 寄存器中的 SIMEN 位来除能或使能。连接到 SPI 接口的单片机以从主 / 从模式进行通信，且主机完成所有的数据传输初始化，并控制时钟信号。由于单片机只有一个 $\overline{SCS}$ 引脚，所以只能拥有一个从机设备。可通过软件控制 $\overline{SCS}$ 引脚使能与除能，设置 CSEN 位为“1”使能 $\overline{SCS}$ 功能，设置 CSEN 位为“0”，$\overline{SCS}$ 引脚将处于浮空状态。



SPI 主 / 从机连接方式

该单片机的 SPI 功能具有以下特点：

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

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

SPI 寄存器

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDBC1	SIMDBC0	SIMEN	SIMICF
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF

SIM 寄存器列表

SIMD 用于存储发送和接收的数据。这个寄存器由 SPI 和 I^2C 功能所共用。在单片机尚未将数据写入到 SPI 总线中时，要传输的数据应先存在 SIMD 中。SPI 总线接收到数据之后，单片机就可以从 SIMD 数据寄存器中读取。所有通过 SPI 传输或接收的数据都必须通过 SIMD 实现。

- **SIMD 寄存器**

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

“x”为未知

单片机中也有两个控制 SPI 接口功能的寄存器，SIMC0 和 SIMC2。应注意的是 SIMC2 与 I^2C 接口功能中的寄存器 SIMA 是同一个寄存器。SPI 功能不会用到寄存器 SIMC1，SIMC1 只适用于 I^2C 中。寄存器 SIMC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 SIMC2 用于其它的控制功能如 LSB/MSB 选择，写冲突标志位等。

- **SIMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDBC1	SIMDBC0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7~5 **SIM2~SIM0**：SIM 工作模式控制位
000：SPI 主机模式；SPI 时钟为 $f_{SYS}/4$
001：SPI 主机模式；SPI 时钟为 $f_{SYS}/16$
010：SPI 主机模式；SPI 时钟为 $f_{SYS}/64$
011：SPI 主机模式；SPI 时钟为 f_{SUB}
100：SPI 主机模式；SPI 时钟为 TM0 CCRP 匹配频率 /2
101：SPI 从机模式
110：I^2C 从机模式
111：未使用模式
这几位用于设置 SIM 功能的工作模式，用于选择 SPI 的主从模式和 SPI 的主机时钟频率及 I^2C 或 SPI 功能。SPI 时钟源可来自于系统时钟也可以选择来自 f_{SUB} 或 TM0。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 未使用，读为“0”

Bit 3~2 **SIMDBC1~SIMDBC0**：I^2C 去抖时间选择位
00：无去抖时间
01：2 个系统时钟去抖时间
1x：4 个系统时钟去抖时间

Bit 1 **SIMEN**：SIM 控制位
0：除能
1：使能
此位为 SIM 接口的开 / 关控制位。此位为“0”时，SIM 接口除能，SDI、SDO、SCK 和 SCS 或 SDA 和 SCL 脚处于浮空状态，SIM 工作电流减小到最小值。此位为“1”时，SIM 接口使能。若 SIM 经由 SIM2~SIM0 位设置为工作在 SPI 接口，当 SIMEN 位由低到高转变时，SPI 控制寄存器中的设置不会发生变化，其首先应在应用程序中初始化。若 SIM 经由 SIM2~SIM0 位设置为工作在 I^2C 接口，当 SIMEN 位由低到高转变时，I^2C 控制寄存器中的设置，如 HXT 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I^2C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0 **SIMICF**：SIM 未完成标志位
0：未发生
1：发生
此位由 SCS 确定。当 $\overline{SCS}$ 为“1”时，SPI 计数器清零，同时产生一个中断并将此位设置为“1”。

- **SIMC2 寄存器**

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

Bit 7~6 未定义位
用户可通过软件程序对这两位进行读写。

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

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

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

Bit 2 **CSEN**：SPI $\overline{\text{SCS}}$ 引脚控制位
0：除能
1：使能
CSEN 位用于 $\overline{\text{SCS}}$ 引脚的使能 / 除能控制。此位为低时，$\overline{\text{SCS}}$ 除能并处于浮空状态。此位为高时，SCS 作为选择脚。

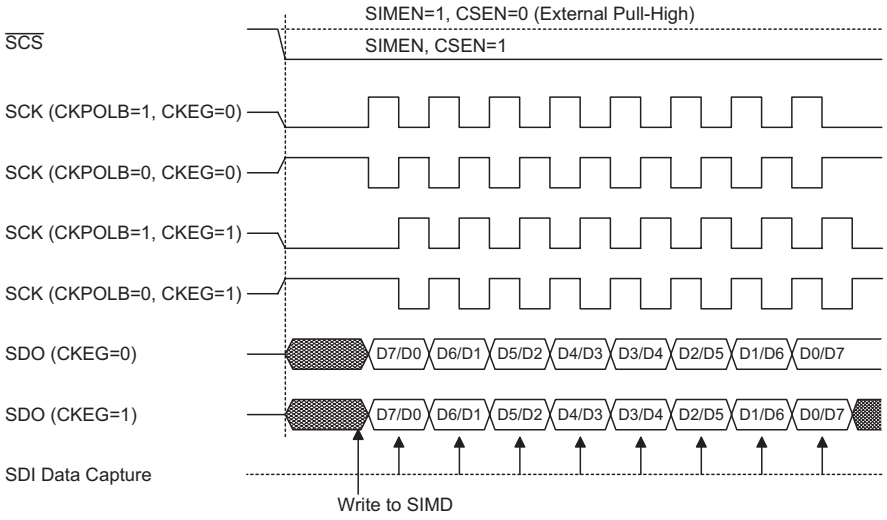
Bit 1 **WCOL**：SPI 写冲突标志位
0：无冲突
1：冲突
WCOL 标志位用于监测数据冲突的发生。此位为高时，数据在传输时被写入 SIMD 寄存器。若数据正在被传输时，此操作无效。此位可被应用程序清零。

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

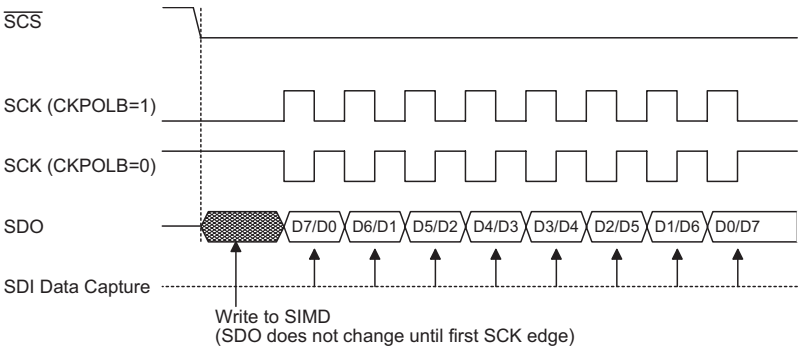
SPI 通信

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

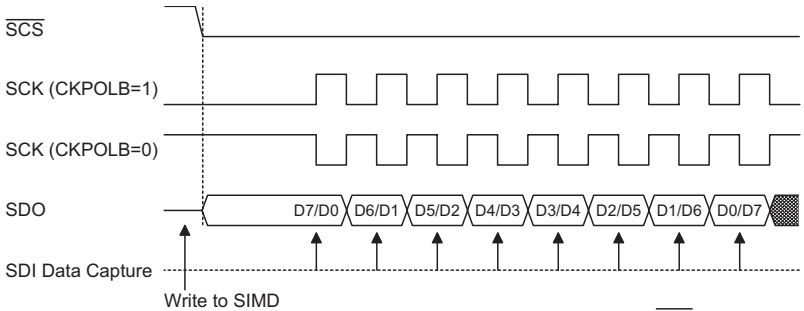
即使在单片机处于空闲模式，SPI 功能仍将继续执行。



SPI 主机模式时序

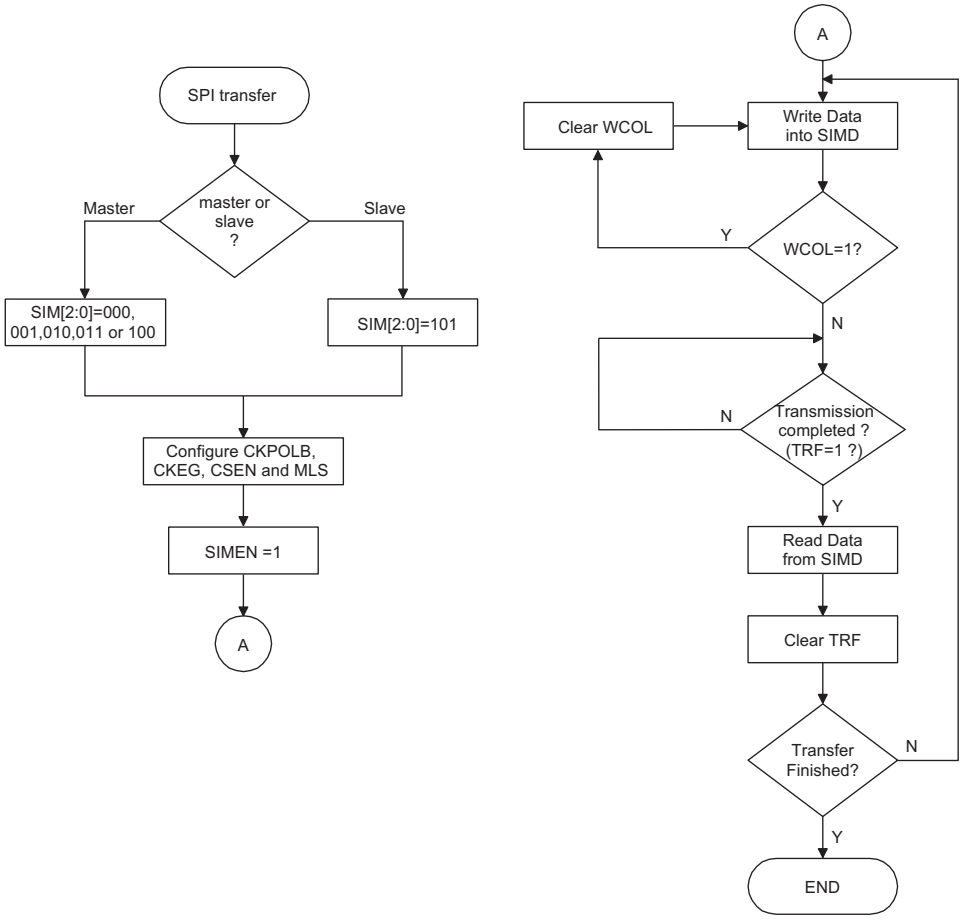


SPI 从机模式时序 – CKEG=0



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

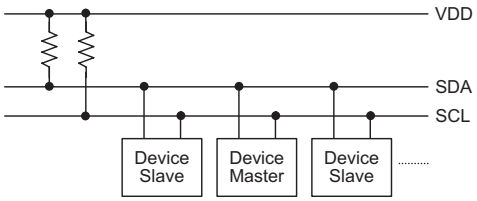
SPI 从机模式时序 – CKEG=1



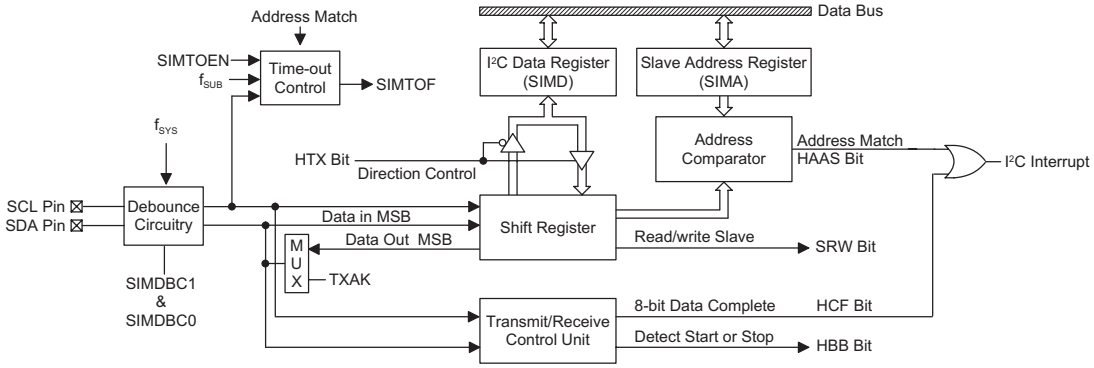
SPI 传输控制流程图

I²C 接口

I²C 可以和传感器、EEPROM 内存等外部硬件接口进行通信。最初是由飞利浦公司研制，是适用于同步串行数据传输的双线式低速串行接口。I²C 接口具有两线通信，非常简单的通信协议和在同一总线上和多个设备进行通信的能力的优点，使之在很多的应用场合中大受欢迎。



I²C 主从总线连接图

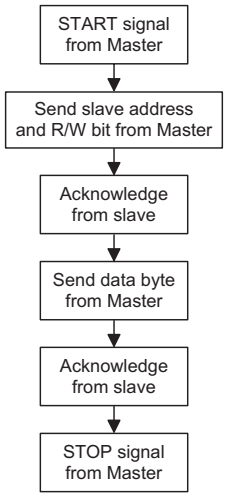


I²C 方框图

I²C 接口操作

I²C 串行接口是一个双线的接口，有一条串行数据线 SDA 和一条串行时钟线 SCL。由于可能有多个设备在同一条总线上相互连接，所以这些设备的输出都是开漏型输出。因此应在这些输出口上都应加上拉电阻。应注意的是，I²C 总线上的每个设备都没有选择线，但分别与唯一的地址一一对应，用于 I²C 通信。

如果有两个设备通过双向的 I²C 总线进行通信，那么就存在一个主机和一个从机。主机和从机都可以用于传输和接收数据，但只有主机才可以控制总线动作。那些处于从机模式的设备，要在 I²C 总线上传输数据只有两种方式，一是从机发送模式，二是从机接收模式。即使 I²C 设备被激活，上拉电阻控制功能和 SCL/SDA 引脚功能仍有效，其上拉电阻功能由相关上拉电阻控制寄存器控制。



I²C 寄存器

I²C 总线有三个控制寄存器 SIMC0、SIMC1 和 SIMTOC，一个地址寄存器 SIMA 及一个数据寄存器 SIMD。SIMD 寄存器，SPI 章节中已有介绍，用于存储正在传输和接收的数据，当单片机将数据写入 I²C 总线之前，实际将被传输的数据存放在寄存器 SIMD 中。从 I²C 总线接收到数据之后，单片机就可以从寄存器 SIMD 中得到这个数据。I²C 总线上的所有传输或接收到的数据都必须通过 SIMD。应注意的是 SIMA 也有另外一个名字，SIMC2，使用 SPI 功能时会用到。I²C 接口会用到寄存器 SIMC0 中的 SIMEN 位和 SIM2~SIM0 位。SIMTOC 寄存器用于 I²C 超时功能控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDBC1	SIMDBC0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	RNIC	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C 寄存器列表

- **SIMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDBC1	SIMDBC0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7~5 **SIM2~SIM0**：SIM 工作模式控制位
000：SPI 主机模式；SPI 时钟为 $f_{SYS}/4$
001：SPI 主机模式；SPI 时钟为 $f_{SYS}/16$
010：SPI 主机模式；SPI 时钟为 $f_{SYS}/64$
011：SPI 主机模式；SPI 时钟为 f_{SUB}
100：SPI 主机模式；SPI 时钟为 TM0 CCRP 匹配频率 /2
101：SPI 从机模式
110：I^2C 从机模式
111：未使用模式
这几位用于设置 SIM 功能的工作模式，用于选择 SPI 的主从模式和 SPI 的主机时钟频率及 I^2C 或 SPI 功能。SPI 时钟源可来自于系统时钟也可以选择来自 f_{SUB} 或 TM0。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 未使用，读为“0”

Bit 3~2 **SIMDBC1~SIMDBC0**：I^2C 去抖时间选择位
00：无去抖时间
01：2 个系统时钟去抖时间
1x：4 个系统时钟去抖时间

Bit 1 **SIMEN**：SIM 控制位
0：除能
1：使能
此位为 SIM 接口的开 / 关控制位。此位为“0”时，SIM 接口除能，SDI、SDO、SCK 和 SCS 或 SDA 和 SCL 脚处于浮空状态，SIM 工作电流减小到最小值。此位为“1”时，SIM 接口使能。若 SIM 经由 SIM2~SIM0 位设置为工作在 SPI 接口，当 SIMEN 位由低到高转变时，SPI 控制寄存器中的设置不会发生变化，其首先应在应用程序中初始化。若 SIM 经由 SIM2~SIM0 位设置为工作在 I^2C 接口，当 SIMEN 位由低到高转变时，I^2C 控制寄存器中的设置，如 HXT 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I^2C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0 **SIMICF**：SIM 未完成标志位
此位不用于 I^2C 模式，在 I^2C 模式下运行时请忽略该标志位。

- **SIMC1 寄存器**

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

Bit 7 **HCF**：I^2C 总线数据传输结束标志位
0：数据正在被传输
1：8 位数据传输完成
数据正在传输时该位为低。当 8 位数据传输完成时，此位为高并产生一个中断。

Bit 6 **HAAS**：I^2C 地址匹配标志位
0：地址不匹配
1：地址匹配
此标志位用于决定从机地址是否与主机发送地址相同。若地址匹配此位为高，否则此位为低。

Bit 5 **HBB**：I^2C 总线忙标志位
0：I^2C 总线闲
1：I^2C 总线忙
当检测到 START 信号时 I^2C 忙，此位变为高电平。当检测到 STOP 信号时 I^2C 总线停止，该位变为低电平。

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

Bit 3 **TXAK**：I^2C 总线发送确认标志位
0：从机发送确认标志
1：从机没有发送确认标志
单片机接收 8 位数据之后会将该位在第九个时钟传到总线上。如果单片机想要接收更多的数据，则应在接收数据之前将此位设置为“0”。

Bit 2 **SRW**：I^2C 从机读 / 写位
0：从机应处于接收模式
1：从机应处于发送模式
SRW 位是从机读写位。决定主机是否希望传输或接收来自 I^2C 总线的数据。当传输地址和从机的地址相同时，HAAS 位会被设置为高，主机将检测 SRW 位来决定进入发送模式还是接收模式。如果 SRW 位为高时，主机会请求从总线上读数据，此时设备处于传输模式。当 SRW 位为“0”时，主机往总线上写数据，设备处于接收模式以读取该数据。

Bit 1 **RNIC**：I^2C 运行是否使用内部时钟控制位
0：I^2C 运行使用内部时钟
1：I^2C 运行不使用内部时钟
I^2C 模块可以在不使用内部时钟的情况下运行，若 SIM 中断使能就会产生一个中断，可用于休眠模式、空闲（低速）模式和正常（低速）模式。
注：若 RNIC=1 且单片机处于暂停模式，从机接收器仍能很好地工作，但从机发送器不能工作，因为它需要系统时钟。

Bit 0 **RXAK**：I^2C 总线接收确认标志位
0：从机接收到确认标志
1：从机没有接收到确认标志
RXAK 位是接收确认标志位。如果 RXAK 位被重设为“0”即 8 位数据传输之后，设备在第九个时钟有接受到一个正确的确认位。如果单片机处于发送状态，发送方会检查 RXAK 位来判断接收方是否愿意继续接收下一个字节。因此直到 RXAK 为“1”时，传输方停止发送数据。这时，传输方将释放 SDA 线，主机发出停止信号。

SIMD 用于存储发送和接收的数据。这个寄存器由 SPI 和 I^2C 功能所共用。在单片机尚未将数据写入到 I^2C 总线中时，要传输的数据应存在 SIMD 中。I^2C 总线接收到数据之后，单片机就可以从 SIMD 数据寄存器中读取。所有通过 I^2C 传输或接收的数据都必须通过 SIMD 实现。

- **SIMD 寄存器**

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

“x”为未知

- **SIMA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~1 **IICA6~IICA0**：I^2C 从机地址位

IICA6~IICA0 是从机地址对应的 6~0 位。此寄存器也在 SPI 接口功能中使用，但其名称改为 SIMC2。SIMA 寄存器用于存放 7 位从机地址，寄存器 SIMA 中的第 7~1 位是单片机的从机地址，位 0 未定义。如果接至 I^2C 的主机发送处的地址和寄存器 SIMA 中存储的地址相符，那么就选中了这个从机。应注意的是寄存器 SIMA 和 SPI 接口使用的寄存器 SIMC2 是同一个寄存器。

Bit 0 无定义

此位可通过软件程序进行读写。

- **SIMTOC 寄存器**

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

Bit 7 **SIMTOEN**：I^2C 超时控制位

0：除能

1：使能

Bit 6 **SIMTOF**：I^2C 超时标志位

0：未发生

1：发生

该位在超时发生时置位，可由软件清零。

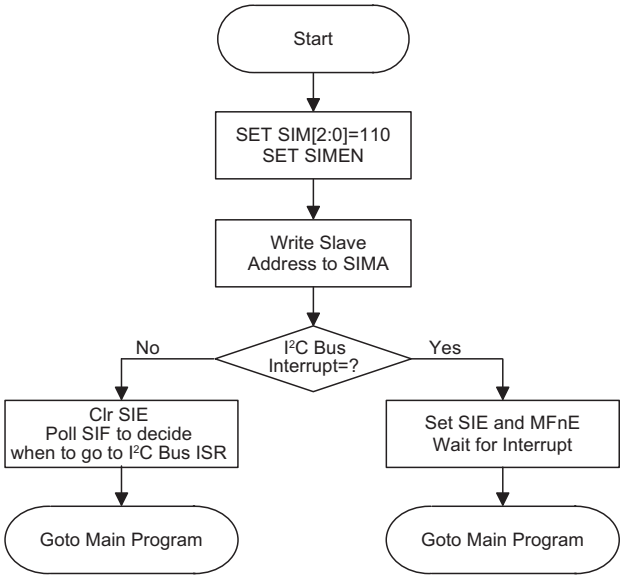
Bit 5~0 **SIMTOS5~SIMTOS0**：I^2C 超时时间选择位

I^2C 超时时钟源是 $f_{SUB}/32$，I^2C 超时时间计算方法：([SIMTOS5:SIMTOS0]+1)×(32/f_{SUB})

I²C 总线通信

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

- 步骤 1
  设置 SIMC0 寄存器中 SIM2~SIM0 为“110”和 SIMEN 位为“1”，以使能 I²C 总线。
- 步骤 2
  向 I²C 总线地址寄存器 SIMA 写入从机地址。
- 步骤 3
  设置 SIE 位和中断控制寄存器中的 SIM 多功能中断使能位，以使能 SIM 中断和多功能中断。



I²C 总线初始化流程图

I²C 总线起始信号

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

从机地址

总线上的所有从机都会侦测由主机发出的起始信号。发送起始信号后，紧接着主机会发送从机地址以选择要进行数据传输的从机。所有在 I²C 总线上的从机接收到 7 位地址数据后，都会将其与各自内部的地址进行比较。如果从机从主机上接收到的地址与自身内部的地址相匹配，则会产生一个 I²C 总线中断信号。地址位接下来的一位为读 / 写状态位 (即第 8 位)，将被保存到 SIMC1 寄存器的 SRW 位，随后发出一个低电平应答信号（即第 9 位）。当单片机从机的地址匹配时，会将状态标志位 HAAS 置位。I²C 总线有两个中断源，当程序运行至中断服务子程序时，通过检测 HAAS 位以确定 I²C 总线中断是来自从机地址匹配，还是来自 8 位数据传递完毕。当是从机地址匹配发生中断时，则从机或是用于发送模式并将数据写进 SIMD 寄存器，或是用于接收模式并从 SIMD 寄存器中读取空值以释放 SCL 线。

I²C 总线读 / 写信号

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

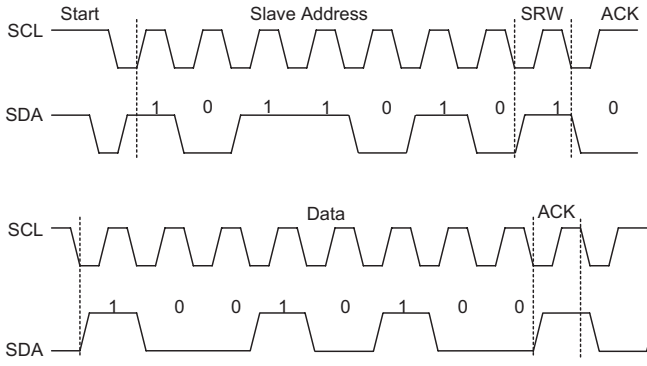
I²C 总线从机地址确认信号

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

I²C 总线数据和确认信号

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

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

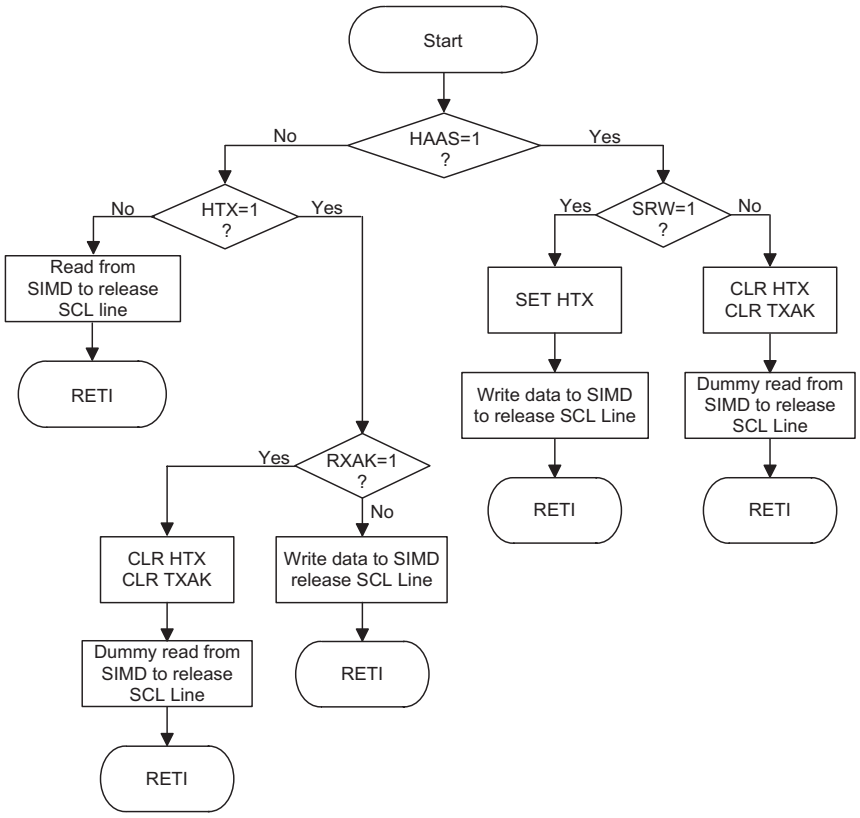


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	-------	S	SA	SR	M	D	A	D	A	-------	P

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

I^2C 通信时序图



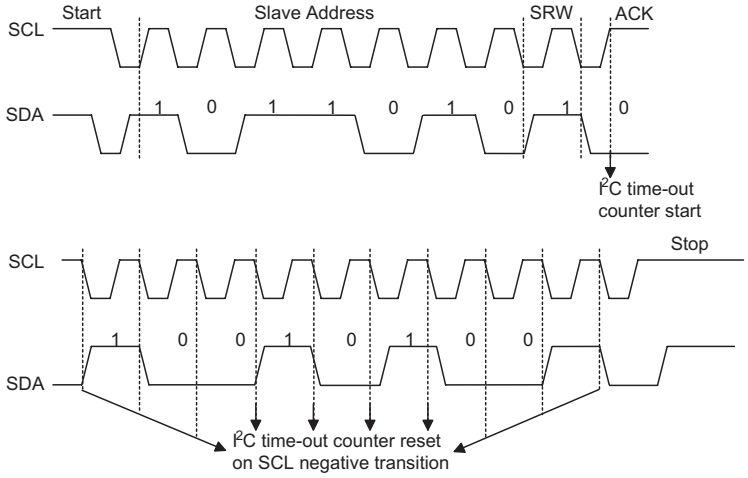
I^2C 总线 ISR 流程图

I²C 超时功能

为了减少由于接收错误时钟源而产生的 I²C 锁定问题，系统提供了超时功能。在固定时间内如果 I²C 总线未接收到时钟源，则 I²C 电路和 SIMC1 寄存器将会复位，SIMTOC 寄存器的 SIMTOF 位将置高。超时功能的使能 / 除能和超时时间都由 SIMTOC 寄存器控制。

I²C 超时操作

超时计数器在 I²C 总线接收到“START”信号和“地址匹配”条件时，超时计数器开始计数，并在 SCL 下降沿处清零。在下一个 SCL 下降沿来临之前，如果等待时间大于 SIMTOC 寄存器设定的超时时间，则会发生超时现象。当 I²C 接收到“STOP”信号，超时计数器将停止计数。64 个超时时间可由 SIMTOC 寄存器的 SIMTOS0~SIMTOS5 位选择。



I²C 超时时序图

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

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

超时发生后的 I²C 寄存器

带 IR 载波的 UART 模块串行接口

UART 模块特性

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

UART 模块概述

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

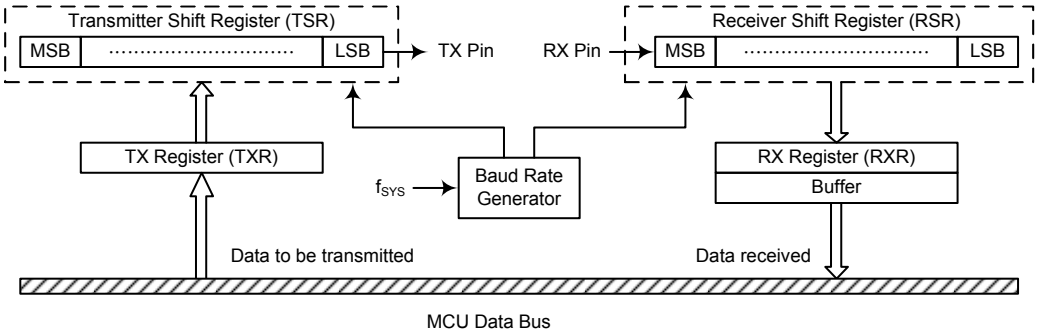
UART 外部引脚接口

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 脚为 UART 发送脚，当 UCR2 控制寄存器中的 TXEN 位为“0”时，该引脚没有被配置作为 UART 发送脚，则可作为普通 I/O 脚或其它共用功能。相似地，RX 为 UART 接收脚，当 UCR2 控制寄存器中的 RXEN 位为“0”时，该引脚没有被配置为 UART 接收脚，则可作为普通 I/O 脚或其它共用功能。UARTEN、TXEN 和 RXEN 位置高时，将自动设置这些 I/O 脚或其它共用功能脚作为 TX 输出和 RX 输入，并且除能 TX 和 RX 引脚上的上拉电阻功能。

UART 数据传输方案

下图显示了 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 TXRRXR 寄存器。



UART 数据传输方案

UART 状态和控制寄存器

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

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
TXRRXR	TXRXD7	TXRXD6	TXRXD5	TXRXD4	TXRXD3	TXRXD2	TXRXD1	TXRXD0
BRG	BRGD7	BRGD6	BRGD5	BRGD4	BRGD3	BRGD2	BRGD1	BRGD0

UART 寄存器列表

USR 寄存器

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

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

Bit 7 **PERR**：奇偶校验出错标志位

0：奇偶校验正确

1：奇偶校验出错

PERR 是奇偶校验出错标志位。若 PERR=0，奇偶校验正确；若 PERR=1，接收到的数据奇偶校验出错。只有使能了奇偶校验此位才有效。可使用软件清除该标志位，即先读取 USR 寄存器再读 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	x	0

“x”为未知

Bit 7 **UARTEN**：UART 功能使能位
0：UART 除能，TX 和 RX 脚用作普通 I/O 口或其它引脚共用功能
1：UART 使能，TX 和 RX 脚作为 UART 功能引脚
此位为 UART 的使能位。UARTEN=0，UART 除能，RX 和 TX 可用作普通的输入输出口；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 引脚作为普通的输入输出端口使用。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 引脚作为普通的输入输出端口使用。

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 引脚可作为普通输入输出端口使用。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 引脚可作为普通输入输出端口使用。

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 且在空闲模式 0 或休眠模式下，RX 引脚的下降沿将唤醒单片机。若 WAKE=0 且在空闲或休眠模式下，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。

TXRRXR 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TXRXD7	TXRXD6	TXRXD5	TXRXD4	TXRXD3	TXRXD2	TXRXD1	TXRXD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”为未知

Bit 7~0 **TXRXD7~TXRXD0**：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	BRGD7	BRGD6	BRGD5	BRGD4	BRGD3	BRGD2	BRGD1	BRGD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”为未知

Bit 7~0 **BRGD7~BRGD0**：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)	f_{SYS} / [64 (N+1)]	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 4MHz 时钟频率且 BRGH=0，若期望的波特率为 4800，计算它的 BRG 寄存器的值 N，实际波特率和误差。

根据上表，波特率 BR = f_{SYS} / [64 (N+1)]

转换后的公式 N = [f_{SYS} / (BR×64)] - 1

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

取最接近的值，十进制 12 写入 BRG 寄存器，实际波特率如下：

BR = 4000000 / [64×(12 + 1)] = 4808

因此，误差 = (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	BRGH=0								
	f_{SYS}=4MHz			f_{SYS}=3.579545MHz			f_{SYS}=7.159MHz		
	BRG	Kbaud	误差 (%)	BRG	Kbaud	误差 (%)	BRG	Kbaud	误差 (%)
0.3	207	0.300	0.16	185	0.300	0.00	—	—	—
1.2	51	1.202	0.16	46	1.190	-0.83	92	1.203	0.23
2.4	25	2.404	0.16	22	2.432	1.32	46	2.380	-0.83
4.8	12	4.808	0.16	11	4.661	-2.90	22	4.863	1.32
9.6	6	8.929	-6.99	5	9.321	-2.90	11	9.332	-2.90
19.2	2	20.833	8.51	2	18.643	-2.90	5	18.643	-2.90
38.4	—	—	—	—	—	—	2	32.286	-2.90
57.6	0	62.500	8.51	0	55.930	-2.90	1	55.930	-2.90
115.2	—	—	—	—	—	—	0	111.859	-2.90

BRGH=0 时的波特率和误差

波特率 K/BPS	BRGH=1								
	f_{SYS}=4MHz			f_{SYS}=3.579545MHz			f_{SYS}=7.159MHz		
	BRG	Kbaud	误差 (%)	BRG	Kbaud	误差 (%)	BRG	Kbaud	误差 (%)
0.3	—	—	—	—	—	—	—	—	—
1.2	207	1.202	0.16	185	1.203	0.23	—	—	—
2.4	103	2.404	0.16	92	2.406	0.23	185	2.406	0.23
4.8	51	4.808	0.16	46	4.76	-0.83	92	4.811	0.23
9.6	25	9.615	0.16	22	9.727	1.32	46	9.520	-0.83
19.2	12	19.231	0.16	11	18.643	-2.90	22	19.454	1.32
38.4	6	35.714	-6.99	5	37.286	-2.90	11	37.286	-2.90
57.6	3	62.5	8.51	3	55.930	-2.90	7	55.930	-2.90
115.2	1	125	8.51	1	111.86	-2.90	3	111.86	-2.90
250	0	250	0	—	—	—	—	—	—

BRGH=1 时的波特率和误差

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，其可用作普通 I/O 口或其它引脚共用功能。当 UART 被除能时将清空缓冲器，所有缓冲器中的数据将被忽略，另外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

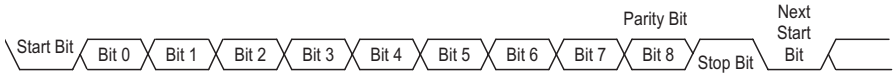
数据传输格式由数据长度、是否校验、校验类型、地址位以及停止位长度组成。它们都是由 UCR1 寄存器的各个位控制的。BNO 决定数据传输是 8 位还是 9 位；PRT 决定校验类型；PRTEN 决定是否选择奇偶校验；而 STOPS 决定选用 1 位还是 2 位停止位。下表列出了各种数据传输格式。地址位用来确定此帧是否为地址。停止位的长度和数据位的长度无关。

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 引脚用作普通 I/O 口或其它引脚共用功能。

发送数据

当 UART 发送数据时，数据从移位寄存器中移到 TX 引脚上，其低位在前高位在后。在发送模式中，TXR 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

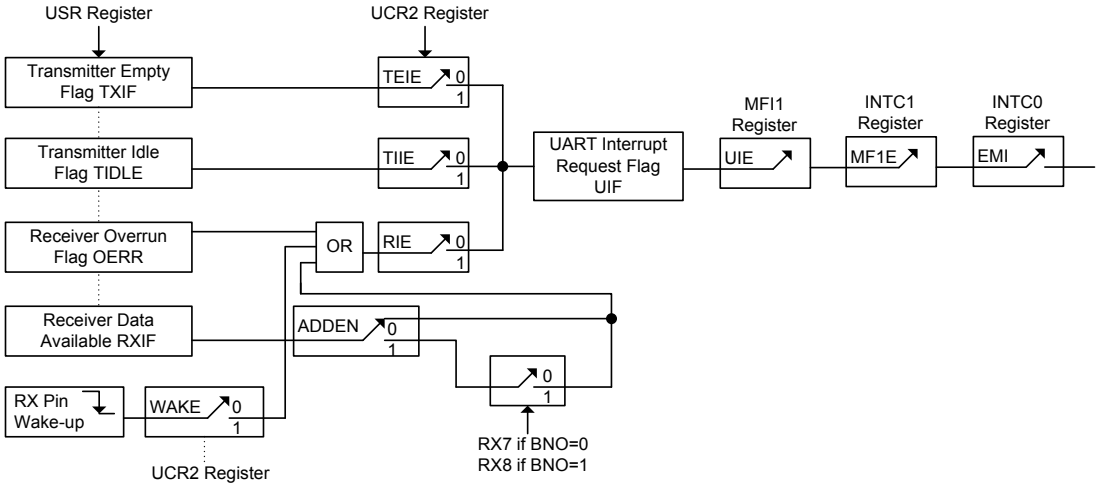
若接收到数据出现奇偶校验错误，USR 寄存器中只读标志 PERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。它同数据一起存储在缓冲器中，可被任何复位清除。注意，FERR 和 PERR 与相应的数据一起存储有缓冲器中，在读取数据之前必须先访问错误标志位。

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将单片机从空闲模式 0 或休眠模式中唤醒。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

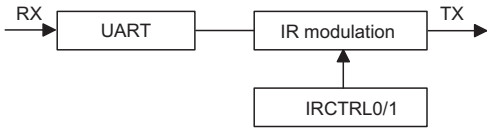
MCU 系统时钟关闭后 UART 模块将停止运行。当传送数据时 MCU 执行 HALT 指令并关闭系统时钟，发送将停止直到 UART 模块时钟再次使能。同样地，当接收数据时 MCU 执行 HALT 指令并关闭系统时钟，数据接收也会停止。当 MCU 进入空闲或休眠模式，USR、UCR1、UCR2、接收 / 发送寄存器以及 BRG 寄存器都不会受到影响。建议在 MCU 进入空闲或休眠模式前先确保数据发送或接收已完成。

UART 功能中包括了 RX 引脚的唤醒功能，由 UCR2 寄存器中 WAKE 位控制。进入空闲模式 0 或休眠模式前，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。

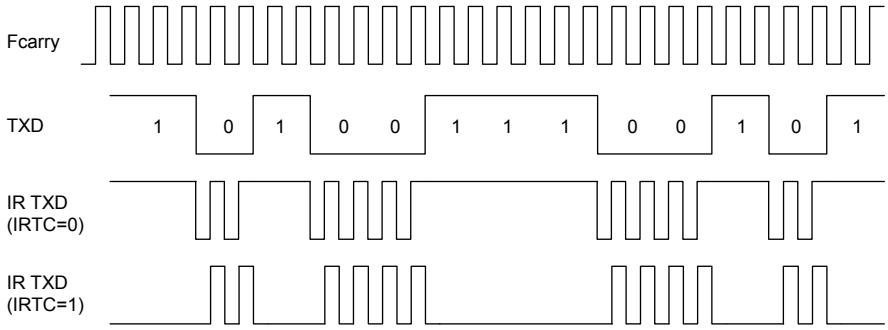
若要唤醒并产生 UART 中断，除了唤醒使能控制位和接收中断使能控制位需置位外，全局中断允许位 EMI 和 UART 中断使能控制位 UIE 也必须置位；若这两控制位没有被置位，那么，单片机将可以被唤醒但不会产生中断。同样唤醒后系统需一定的延时才能正常工作，然后才会产生 UART 中断。

IR 调制接口

该 UART 模块内置一个 IR 调制接口。IR 调制频率由 IRCTRL0 寄存器控制，其值为 0~127 之间任意整数。



红外调制方式：当 TXD=0，才会产生红外线混波并将数据输出。为了满足 PNP 和 NPN 红外驱动管的需求，可以通过 IRCTRL0 寄存器的第 7 位 IRTC 控制红外调制输出的极性。IRTC=0 为正极输出，适用于 PNP 晶体驱动管；IRC=1 为负极输出，适用于 NPN 晶体驱动管。



IRCTRL0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	IRTC	IRDC6	IRDC5	IRDC4	IRDC3	IRDC2	IRDC1	IRDC0
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 **IRTC**：红外调制输出极性选择
0：负
1：正

Bit 6~0 **IRDC6~IRDC0**：红外调制频率分频系数
Fcarry=(f_{SYS}/(IRDC+1))/2

IRCTRL1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	IRME0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 未使用，读为“0”

Bit 0 **IRME0**：UART TX 红外调制控制位
0：除能
1：使能

中断

中断是单片机一个重要功能。当外部事件或内部功能如定时器模块或 A/D 转换器有效，并且产生中断时，系统会暂时中止当前的程序而转到执行相对应的中断服务程序。此单片机提供多个外部中断和内部中断功能，外部中断由 INT0~INT1 引脚动作产生，而内部中断由各种内部功能，如定时器模块、时基、LVD、EEPROM 和 A/D 转换器等产生。

中断寄存器

中断控制基本上是在一定单片机条件发生时设置请求标志位，应用程序中中断使能位的设置是通过位于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~MFI3 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=0~1
多功能	MFnE	MFnF	n=0~3
A/D 转换器	ADE	ADF	—
时基	TBnE	TBnF	n=0~1
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
SIM	SIE	SIF	—
I^2C 超时	I2CTOE	I2CTOF	—
UART	UIE	UIF	—
TM	TnPE	TnPF	n=0~2
	TnAE	TnAF	

中断寄存器位命名模式

寄存器名称	位							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	MF0F	INT1F	INT0F	MF0E	INT1E	INT0E	EMI
INTC1	MF1F	TB1F	TB0F	ADF	MF1E	TB1E	TB0E	ADE
INTC2	—	—	MF3F	MF2F	—	—	MF3E	MF2E
MFI0	—	—	T0AF	T0PF	—	—	T0AE	T0PE
MFI1	UIF	SIF	DEF	LVF	UIE	SIE	DEE	LVE
MFI2	I2CTOF	—	T1AF	T1PF	I2CTOE	—	T1AE	T1PE
MFI3	—	—	T2AF	T2PF	—	—	T2AE	T2PE

中断寄存器列表

INTEG 寄存器

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

Bit 7~4　未定义，读为“0”

Bit 3~2　**INT1S1~INT1S0**：INT1 脚中断边沿控制位
00：除能
01：上升沿
10：下降沿
11：双沿

Bit 1~0　**INT0S1~INT0S0**：INT0 脚中断边沿控制位
00：除能
01：上升沿
10：下降沿
11：双沿

INTC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	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	MF1F	TB1F	TB0F	ADF	MF1E	TB1E	TB0E	ADE
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 **MF1F**：多功能中断 1 中断请求标志位
0：无请求
1：中断请求

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

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

Bit 4 **ADF**：A/D 转换器中断请求标志位
0：无请求
1：中断请求

Bit 3 **MF1E**：多功能中断 1 中断控制位
0：除能
1：使能

Bit 2 **TB1E**：时基 1 中断控制位
0：除能
1：使能

Bit 1 **TB0E**：时基 0 中断控制位
0：除能
1：使能

Bit 0 **ADE**：A/D 转换器中断控制位
0：除能
1：使能

INTC2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	MF3F	MF2F	—	—	MF3E	MF2E
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 **MF3F**：多功能中断 3 中断请求标志位
0：无请求
1：中断请求

Bit 4 **MF2F**：多功能中断 2 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 **MF3E**：多功能中断 3 中断控制位
0：除能
1：使能

Bit 0 **MF2E**：多功能中断 2 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	UIF	SIF	DEF	LVF	UIE	SIE	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 **UIF**：UART 中断请求标志位
0：无请求
1：中断请求

Bit 6 **SIF**：SIM 中断请求标志位
0：无请求
1：中断请求

Bit 5 **DEF**：EEPROM 中断请求标志位
0：无请求
1：中断请求

Bit 4 **LVF**：LVD 中断请求标志位
0：无请求
1：中断请求

Bit 3 **UIE**：UART 中断控制位
0：除能
1：使能

Bit 2 **SIE**：SIM 中断控制位
0：除能
1：使能

Bit 1 **DEE**：EEPROM 中断控制位
0：除能
1：使能

Bit 0 **LVE**：LVD 中断控制位
0：除能
1：使能

MFI2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	I2CTOF	—	T1AF	T1PF	I2CTOE	—	T1AE	T1PE
R/W	R/W	—	R/W	R/W	R/W	—	R/W	R/W
POR	0	—	0	0	0	—	0	0

Bit 7 **I2CTOF**：I^2C 超时中断请求标志位
0：无请求
1：中断请求

Bit 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 **I2CTOE**：I^2C 超时中断控制位
0：除能
1：使能

Bit 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

MFI3 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

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。

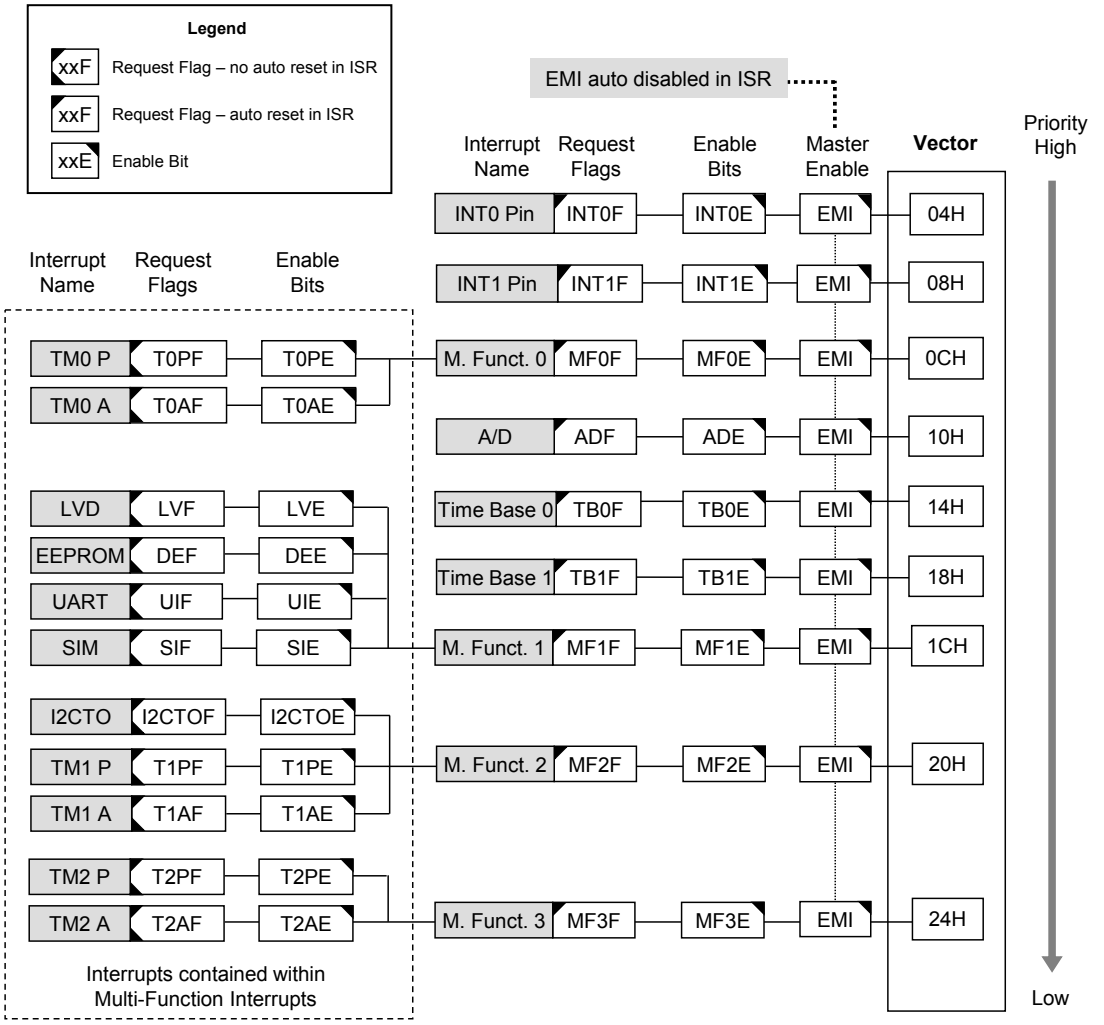
当中断发生时，下条指令的地址将被压入堆栈。相应的中断向量地址加载至 PC 中。系统将从此向量取下条指令。中断向量处通常为跳转指令，以跳转到相应的中断服务程序。中断服务程序必须以“RETI”指令返回至主程序，以继续执行原来的程序。

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。

外部中断

通过 INT0~INT1 引脚上的信号变化可控制外部中断。当触发沿选择位设置好触发类型，INT0~INT1 引脚的状态发生变化，外部中断请求标志 INT0F~INT1F 被置位时外部中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和相应中断使能位 INT0E~INT1E 需先被置位。此外，必须使用 INTEG 寄存器使能外部中断功能并选择触发沿类型。外部中断引脚和普通 I/O 口共用，如果相应寄存器中的中断使能位被置位，此引脚将被作为外部中断脚使用。此时该引脚必须通过设置控制寄存器，将该引脚设置为输入口。当中断使能，堆栈未满并且外部中断脚状态改变，将调用外部中断向量子程序。当响应外部中断服务子程序时，中断请求标志位 INT0F~INT1F 会自动复位且 EMI 位会被清零以除能其它中断。注意，即使此引脚被用作外部中断输入，其上拉电阻选项仍保持有效。寄存器 INTEG 被用来选择有效的边沿类型，来触发外部中断。可以选择上升沿还是下降沿或双沿触发都产生外部中断。注意 INTEG 也可以用来除能外部中断功能。



中断结构

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

多功能中断

此单片机中有多达 4 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、UART 中断、SIM 中断、I^2C 超时中断和 EEPROM 中断。

当多功能中断中任何一种中断请求标志 MFnF 被置位，多功能中断请求产生。当中断使能，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、UART 中断、SIM 中断、I^2C 超时中断和 EEPROM 中断的请求标志位不会自动复位，必须由应用程序清零。

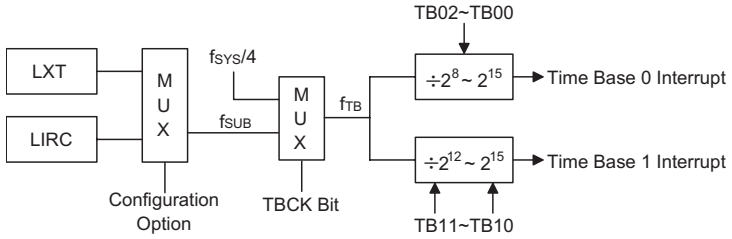
串行接口模块中断

串行接口模块中断，即 SIM 中断，属于多功能中断。当一个字节数据已由 SIM 接口接收或发送完，中断请求标志 SIF 被置位，SIM 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI、串行接口中断使能位 SIE 和多功能中断使能位需先被置位。当中断使能，堆栈未满且一个字节数据已被传送或接收完毕时，可跳转至相关多功能中断向量子程序中执行。当串行接口中断响应，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 SIF 标志需在应用程序中手动清除。

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 和时基使能位 TB0E 或 TB1E 被置位，允许程序跳转到各自的中断向量地址。当中断使能，堆栈未满且时基溢出时，将调用它们各自的中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 TB0F 或 TB1F 会自动复位且 EMI 位会被清零以除能其它中断。

时基中断的目的是提供一个固定周期的中断信号，时钟源来自内部时钟源 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。



时基中断

TBC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TBON	TBCK	TB11	TB10	LXTLP	TB02	TB01	TB00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	1	0	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_{SUB}
1：$f_{SYS}/4$

Bit 5~4 **TB11~TB10**：选择时基 1 溢出周期位
00：$2^{12}/f_{TB}$
01：$2^{13}/f_{TB}$
10：$2^{14}/f_{TB}$
11：$2^{15}/f_{TB}$

Bit 3 **LXTLP**：LXT 低功耗控制位
0：快速启动模式
1：低功耗模式

Bit 2~0 **TB02~TB00**：选择时基 0 溢出周期位
000：$2^{8}/f_{TB}$
001：$2^{9}/f_{TB}$
010：$2^{10}/f_{TB}$
011：$2^{11}/f_{TB}$
100：$2^{12}/f_{TB}$
101：$2^{13}/f_{TB}$
110：$2^{14}/f_{TB}$
111：$2^{15}/f_{TB}$

I^2C 超时中断

I^2C 超时中断也属于多功能中断。当 I^2C 超时计数器溢出时，中断请求标志 I2CTOF 被置位，I^2C 超时中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI、I^2C 超时中断使能位 I2CTOE 和多功能中断使能位需先被置位。当中断使能，堆栈未满且 I^2C 超时发生时，可跳转至相关多功能中断向量子程序中执行。当 I^2C 超时中断响应，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 I2CTOF 标志需在应用程序中手动清除。

UART 中断

UART 中断也属于多功能中断。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，UART 中断请求标志 UIF 被置位，UART 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI、UART 中断使能位 UIE 和相应多功能中断使能位需先被置位。当中断使能，堆栈未满且以上任何一种情况发生时，可跳转至相关多功能中断向量子程序中执行。当 UART 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，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。

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

简易型和周期型 TM 都有两个中断。所有的 TM 中断也属于多功能中断。简易型和周期型 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，低电压或比较器输入改变都可能导致其相应的中断标志被置位，由此产生中断，因此必须注意避免伪唤醒情况的发生。若中断唤醒功能被除能，单片机进入休眠或空闲模式前相应中断请求标志应被置起。中断唤醒功能不受中断使能位的影响。

编程注意事项

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

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

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

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

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

此单片机具有低电压检测功能，即 LVD。该功能使能用于监测电源电压 V_{DD} 或 LVDIN 引脚输入电压，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 LVD 功能的工作条件。LVDO 位被置位时低电压情况发生，若 LVDO 位为低表明 V_{DD} 电压或 LVDIN 引脚输入电压高于参考电压。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”

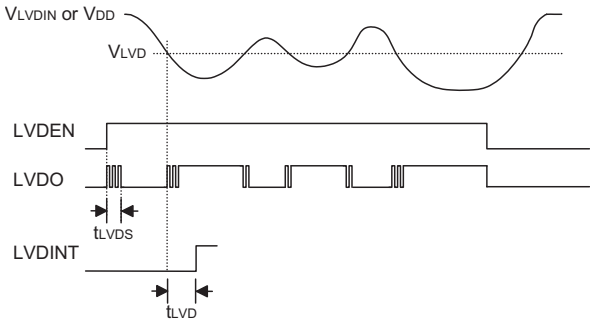
Bit 2~0　**VLVD2~VLVD0**：LVD 功能工作条件选择位
000：LVDIN 引脚，$V_{LVDIN} \leq 1.04V$
001：2.2V
010：2.4V
011：2.7V
100：3.0V
101：3.3V
110：3.6V
111：4.0V

当 VLVD2~VLVD0=000B 时，LVD 功能是通过比较 LVDIN 引脚输入电压与参考电压来实现的。

当 VLVD2~VLVD0 ≠ 000B 时，LVD 功能是通过比较 VDD 电压与参考电压来实现的。

LVD 操作

通过比较电源电压 V_{DD} 或 LVDIN 引脚输入电压与存储在 LVDC 寄存器中的预置电压值的结果，低电压检测功能工作。当电源电压 V_{DD} 或 LVDIN 引脚输入电压低于预置电压值时，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} 或 LVDIN 引脚输入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} 或 LVDIN 引脚输入电压降至小于 LVD 预置电压值时，中断请求标志位 LVF 将被置位，中断产生，单片机将从休眠或空闲模式中被唤醒。若不要求低电压检测的唤醒功能使能，在单片机进入休眠或空闲模式前应将 LVF 标志置为高。

当 LVD 功能使能后，建议在使能相关中断前先将 LVD 中断标志位清零以避免错误操作。

体脂测量功能

体脂电路由一个正弦波发生器、一个放大器和一个滤波器组成。该电路具有最大的灵活性的设计，且功能集成度高，可实现身体脂肪测量的功能。该电路是由 LDO 供电。

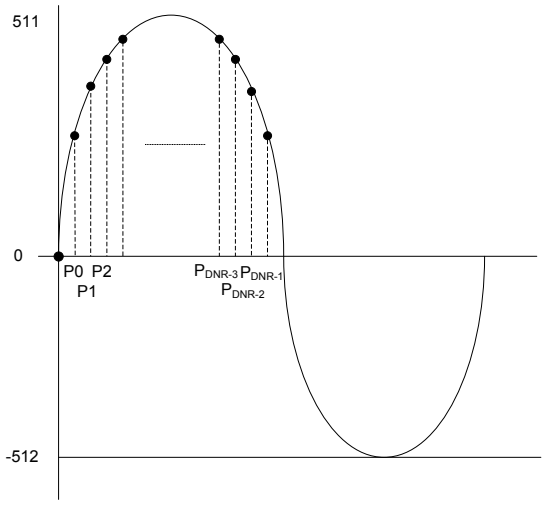
正弦波发生器

正弦波发生器由分频器、计数器、RAM、10 位 DAC 和 OP0 组成。该电路会产生一个频率范围为 5kHz ~ 200kHz 的正弦波，并使用 32 × 9 位的 RAM 来模拟正弦波形。分频器将以 DN/M 的倍数来生成时钟用于计数器。了解正弦波是如何产生时需注意以下几点：

- 系统时钟 /M = 正弦波频率
- 系统时钟 × (DN/M) = 计数器的计数率
- M 必须是 N 和 8 倍数
- M = N×DN
- DNR = DN/2
- DN：正弦波周期数据的数值 (DN ≤ 64)
- DNR：存储在 RAM 中的正弦波半周期数据的数值 (DNR ≤ 32)

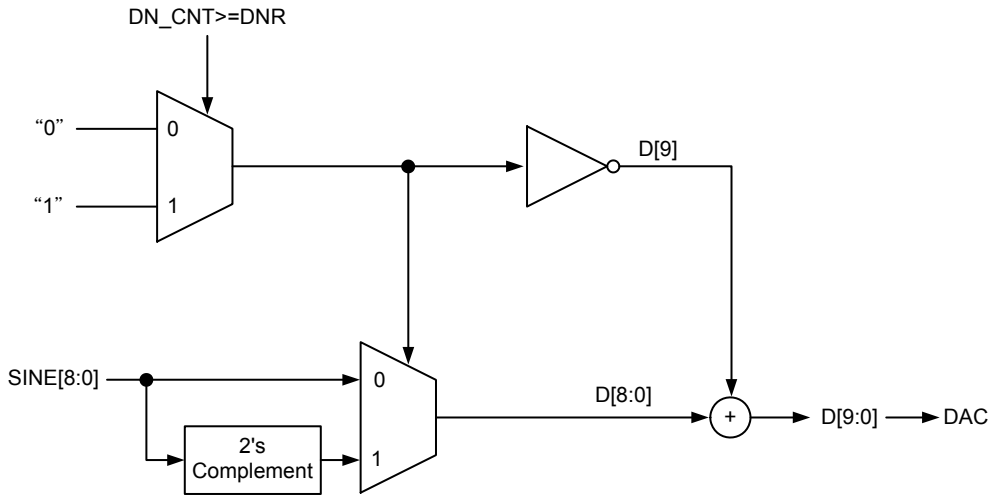
更多细节请参考下表和下图：

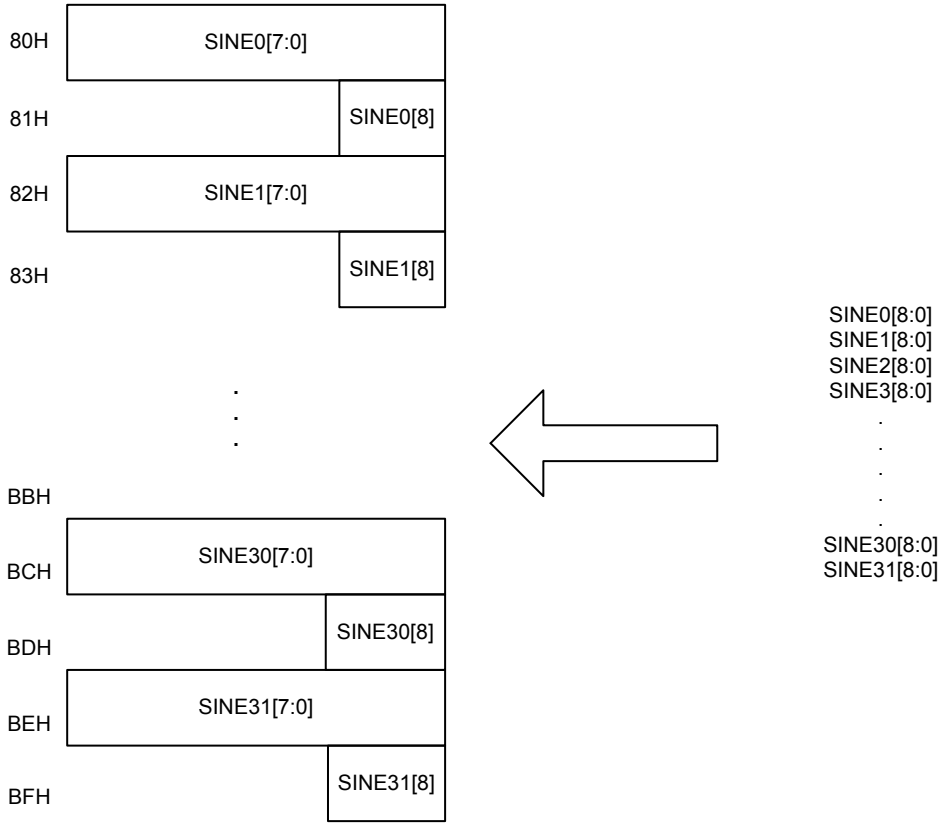
系统时钟	4.8MHz				9.6MHz				14.4MHz			
正弦波频率 (kHz)	200	100	50	5	200	100	50	5	200	100	50	5
M	24	48	96	960	48	96	192	1920	72	144	288	2880
N	1	1	2	20	1	2	4	40	3	3	6	60
DN	24	48	48	48	48	48	48	48	24	48	48	48
DNR	12	24	24	24	24	24	24	24	12	24	24	24



电路只能产生一个 P0~P_{DNR-1} 的半正弦波，并将数据存储在地址范围为 80H~BFH 的 RAM Sector 2 中。正弦波数据 bit[7:0] 存储在偶数地址，而正弦波数据 bit[8] 存储在奇数地址。一旦正弦波发生器使能，CPU 将无法对这个 RAM 区域进行读写，但正弦波发生器可读取 RAM 数据并将其传送到 10 位 DAC。

单片机从 RAM 中读取半正弦波数据，然后在 SIN 引脚上生成实际正弦波形，如下图所示：





RAM (Sector 2) **SINE Wave Pattern**

SGC 寄存器

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

Bit 7 **SGEN**：正弦波发生器使能位
0：除能
1：使能
该位为 0 时，OP0 和 10 位 DAC 将处于暂停模式。

Bit 6~5 未定义，读为“0”

Bit 4 **BREN**：偏置电阻使能位
0：除能 — 暂停模式
1：使能 — 正常模式
该位使能时会产生一个 0.5 × AV_{DD} 电压到 OPA1 和 OPA2 的正相输入端。

Bit 3~0 未定义，读为“0”

SGN 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 **D5~D0**：正弦波发生器数据
系统倍频 N 等于 D[5:0] + 1。

SGDNR 寄存器

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

Bit 7~5 未定义，读为“0”

Bit 4~0 **D4~D0**：采样数据数
正弦波半周期数据的数值存储在 RAM 的 Sector 2 中，DNR 等于 D[4:0] + 1。

放大器

放大器由 OP1、OP2、6 位 DAC 和模拟开关组成。OP2 是一个增益倍数为 1~5 的差分放大器。6 位 DAC 为 OP2 的正相输入端提供了一个参考电压。用户可以通过打开和关闭开关 SW0 ~ SW7 来获得参考电阻上的电压和人体电阻上的电压。

通过 SW0 ~ SW7 的开关方式可以获取人体阻抗和参考阻抗，具体内容请参考下表。

开关	SW0	SW1	SW2	SW3	SW4	SW5	SW6	SW7
足部阻抗	O						O	O
参考阻抗 1kΩ		O			O	O		
参考阻抗 200Ω		O	O	O				

O：开关 On

OPAC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	OPAEN	—	—	—	OP2G3	OP2G2	OP2G1	OP2G0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **OPAEN**：运算放大器控制位
0：除能
1：使能
该位为 0 时，OP1、OP2 和 6 位 DAC 将处于暂停模式。

Bit 6~4 未定义，读为“0”

Bit 3~0 **OP2G3~OP2G0**：OP2 增益控制位
0001：1.14
0010：1.31
0011：1.5
0100：1.73
0101：2
0110：2.33
0111：2.75
1000：3.285
1001：4
1010：5
其它值：1

SWC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
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 **SW7**：开关 7 控制位
0：Off
1：On

Bit 6 **SW6**：开关 6 控制位
0：Off
1：On

Bit 5 **SW5**：开关 5 控制位
0：Off
1：On

Bit 4 **SW4**：开关 4 控制位
0：Off
1：On

Bit 3 **SW3**：开关 3 控制位
0：Off
1：On

Bit 2 **SW2**：开关 2 控制位
0：Off
1：On

Bit 1 **SW1**：开关 1 控制位
0：Off
1：On

Bit 0 **SW0**：开关 0 控制位
0：Off
1：On

DACO 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 **D5~D0**：6 位 DAC 输出电压
输出电压 = $0.5AV_{DD} \times ((D[5:0] + 1) / 64)$

滤波器

滤波器由 CP0、PMOS 晶体管和一些模拟开关组成。它还具有一个峰值检测功能，可以将外部电容存储的峰值传输到 ADC。开关 SW8 和 SW9 用于电容放电。

FTRC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	FTREN	—	—	HYSEN	—	—	SW9	SW8
R/W	R/W	—	—	R/W	—	—	R/W	R/W
POR	0	—	—	0	—	—	0	0

Bit 7 **FTREN**：滤波器控制位
0：除能
1：使能
该位为 0 时，CP0 和 PMOS 将处于暂停模式。

Bit 6~5 未定义，读为“0”

Bit 4 **HYSEN**：保留位

Bit 3~2 未定义，读为“0”

Bit 1 **SW9**：开关 9 控制位
0：Off
1：On

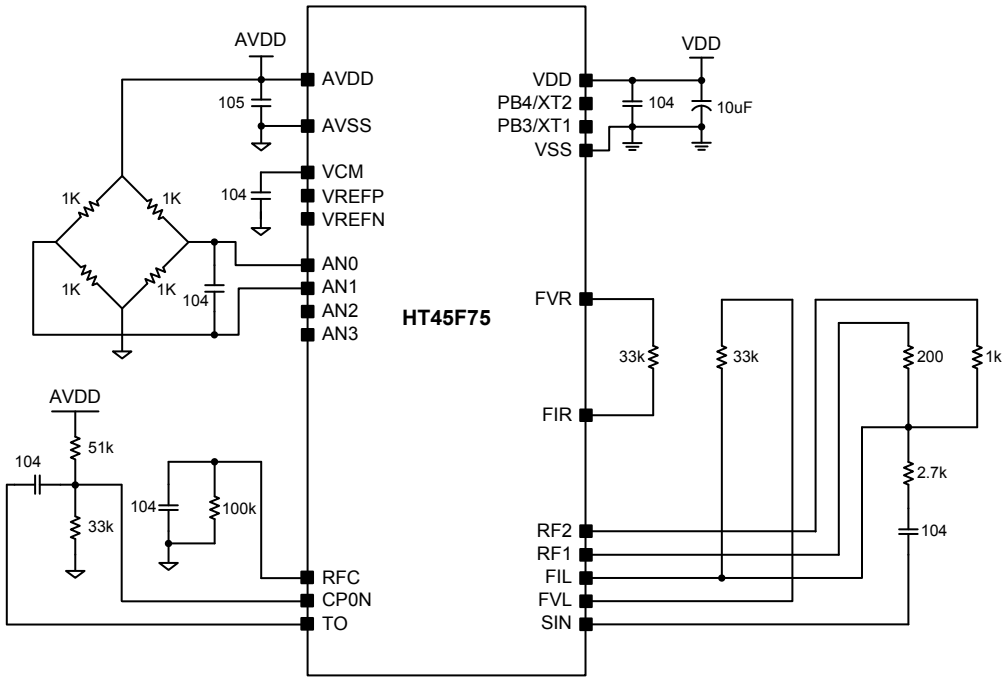
Bit 0 **SW8**：开关 8 控制位
0：Off
1：On

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
振荡器选项	
1	高速振荡器类型选择 – f_H： 1. HXT 2. HIRC
2	HIRC 频率选择： 1. 4.8MHz 2. 9.6MHz 3. 14.4MHz
3	低速振荡器类型选择 – f_{SUB}： 1. LXT 2. LIRC
看门狗选项	
4	看门狗定时器功能： 1. 总是使能 2. 由 WDTC 寄存器控制
5	WDT 时钟选择 – f_S： 1. f_{SUB} 2. $f_{SYS}/4$

应用电路


AVDD
105
AVSS
VCM
104
VREFP
VREFN
1K
1K
1K
1K
104
AN0
AN1
AN2
AN3
HT45F75
AVDD
51k
104
33k
104
100k
RFC
CP0N
TO
VDD
VDD
PB4/XT2
PB3/XT1
VSS
104
10uF
FVR
33k
33k
200
1k
FIR
2.7k
104
RF2
RF1
FIL
FVL
SIN

指令集

简介

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

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

指令周期

大部分的操作均只需要一个指令周期来执行。分支、调用或查表则需要两个指令周期。一个指令周期相当于四个系统时钟周期，因此如果在 8MHz 的系统时钟振荡器下，大部分的操作将在 0.5μs 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,[m]	ACC 与数据存储器、进位标志的反相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	1 注	Z, C, AC, OV, SC, CZ
SBC A, x	ACC 与立即数、进位标志相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
DAA [m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	1 注	C
逻辑运算			
AND A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	1	Z
OR A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	1	Z
XOR A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	1	Z
ANDM A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	1 注	Z
ORM A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	1 注	Z
XORM A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	1 注	Z
AND A, x	ACC 与立即数做“与”运算，结果放入 ACC	1	Z
OR A, x	ACC 与立即数做“或”运算，结果放入 ACC	1	Z
XOR A, x	ACC 与立即数做“异或”运算，结果放入 ACC	1	Z
CPL [m]	对数据存储器取反，结果放入数据存储器	1 注	Z
CPLA [m]	对数据存储器取反，结果放入 ACC	1	Z
递增和递减			
INCA [m]	递增数据存储器，结果放入 ACC	1	Z
INC [m]	递增数据存储器，结果放入数据存储器	1 注	Z
DECA [m]	递减数据存储器，结果放入 ACC	1	Z
DEC [m]	递减数据存储器，结果放入数据存储器	1 注	Z

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

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

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

扩展指令集

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

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

助记符	说明	指令周期	影响标志位
LSET [m].i	置位数据存储器的位	2 注	无
转移			
LSZ [m]	如果数据存储器为零，则跳过下一条指令	2 注	无
LSZA [m]	数据存储器送至 ACC，如果内容为零，则跳过下一条指令	1 注	无
LSNZ [m]	如果数据存储器不为零，则跳过下一条指令	2 注	无
LSZ [m].i	如果数据存储器的第 i 位为零，则跳过下一条指令	2 注	无
LSNZ [m].i	如果数据存储器的第 i 位不为零，则跳过下一条指令	2 注	无
LSIZ [m]	递增数据存储器，如果结果为零，则跳过下一条指令	2 注	无
LSDZ [m]	递减数据存储器，如果结果为零，则跳过下一条指令	2 注	无
LSIZA [m]	递增数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	2 注	无
LSDZA [m]	递减数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	2 注	无
查表			
LTABRD [m]	读取当前页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
LTABRDL [m]	读取最后页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
LITABRD [m]	读表指针 TBLP 自加，读取当前页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
LITABRDL [m]	读表指针 TBLP 自加，读取最后页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
其它指令			
LCLR [m]	清除数据存储器	2 注	无
LSET [m]	置位数据存储器	2 注	无
LSWAP [m]	交换数据存储器的高低字节，结果放入数据存储器	2 注	无
LSWAPA [m]	交换数据存储器的高低字节，结果放入 ACC	2	无

注：1. 对扩展跳转指令而言，如果比较的结果牵涉到跳转即需 4 个周期，如果没有发生跳转，则只需两个周期。

2. 任何扩展指令若要改变 PCL 的内容将需要 3 个周期来执行。

指令定义

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NOP No operation
指令说明 空操作，接下来顺序执行下一条指令。
功能表示 PC ← PC+1
影响标志位 无

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

SUB A, [m] Subtract Data Memory from ACC

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

功能表示 ACC ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

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

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

功能表示 [m] ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

SUB A, x Subtract immediate Data from ACC

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

功能表示 ACC ← ACC – x

影响标志位 OV、Z、AC、C、SC、CZ

SWAP [m] Swap nibbles of Data Memory

指令说明 将指定数据存储器的低 4 位和高 4 位互相交换。

功能表示 [m].3~[m].0 ↔ [m].7~[m].4

影响标志位 无

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

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

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

影响标志位 无

SZ [m] Skip if Data Memory is 0

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

功能表示 如果 [m]=0, 跳过下一条指令执行

影响标志位 无

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

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

功能表示 ACC ← [m]，如果 [m]=0，跳过下一条指令执行

影响标志位 无

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

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

功能表示 如果 [m].i=0，跳过下一条指令执行

影响标志位 无

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

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

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

影响标志位 无

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

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

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

影响标志位 无

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

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

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

影响标志位 无

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

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

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

XOR A, x Logical XOR immediate data to ACC
指令说明 将累加器的数据与立即数逻辑异或，结果存放到累加器。
功能表示 ACC ← ACC“XOR”x
影响标志位 Z

扩展指令定义

扩展指令被用来直接存取存储在任何数据存储器 Sector 中的数据。

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

功能表示 [m] ← ACC – [m] – $\overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

LSDZ [m] Skip if Decrement Data Memory is 0

指令说明 将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 [m] ← [m] -1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

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

指令说明 将指定数据存储器内容减 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果将存放到累加器，但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 ACC ← [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 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 [m] ← [m]+1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

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

指令说明 将指定数据存储器的内容加 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果会被存放到累加器，但是指定数据存储器的内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 ACC ← [m]+1，如果 ACC=0 跳过下一条指令执行

影响标志位 无

LSNZ [m].i Skip if bit i of Data Memory is not 0

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

功能表示 如果 [m].i≠0，跳过下一条指令执行

影响标志位 无

LSNZ [m] Skip if Data Memory is not 0

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

功能表示 如果 [m]≠0，跳过下一条指令执行

影响标志位 无

LSUB A, [m] Subtract Data Memory from ACC

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

功能表示 ACC ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

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

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

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

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

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

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

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

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

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

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

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

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

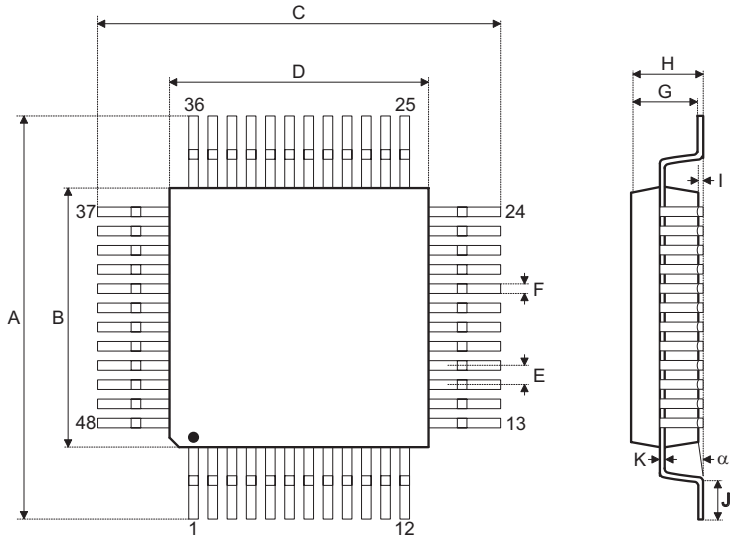
封装信息

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

封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

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

48-pin LQFP (7mm × 7mm) 外形尺寸



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

符号	尺寸 (单位：mm)		
	最小	正常	最大
A	—	9.00 BSC	—
B	—	7.00 BSC	—
C	—	9.00 BSC	—
D	—	7.00 BSC	—
E	—	0.50 BSC	—
F	0.17	0.22	0.27
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°