SPWM 输出端反相控制 Flash 单片机

JXY-IPS5V

版本 : V1.00　日期 : 2018-02-24

目 录

特性

CPU 特性

- 工作电压：
  - ♦ f_{SYS} = 20MHz：4.0V~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
- 振荡器类型：
  - ♦ 外部高频晶振 – HXT
  - ♦ 内部 16MHz RC – HIRC
  - ♦ 内部 32kHz RC – LIRC
- 多种工作模式：正常、低速、空闲和休眠
- 内建 16MHz 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
- 6 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：256×8
- True EEPROM 存储器：64×8
- 看门狗定时器
- 20 个双向输入 / 输出口
- 与 I/O 口复用的外部中断输入
- 多个定时器模块用于时间测量、捕捉输入、比较匹配输出、PWM 输出及单脉冲输出
- 多通道 12-bit A/D 转换器
- 双时基功能用以产生固定的中断信号
- 过电流保护功能
- 过电压保护功能
- 正弦波 PWM 产生功能
- 交流过零检测功能
- 低电压复位功能
- 低电压检测功能
- UART 接口用于全双工异步通信
- 封装类型：24-pin SSOP
- Flash 程序存储器烧录可达 100,000 次
- Flash 程序存储器数据可保存 10 年以上
- True EEPROM 数据存储器烧录可达 1,000,000 次
- True EEPROM 数据存储器数据可保存 10 年以上

概述

JXY-IPS5V 是一款 A/D 型具有 8 位高性能精简指令集的 Flash 单片机，为弦波输出端反相控制 ASSP MCU。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。

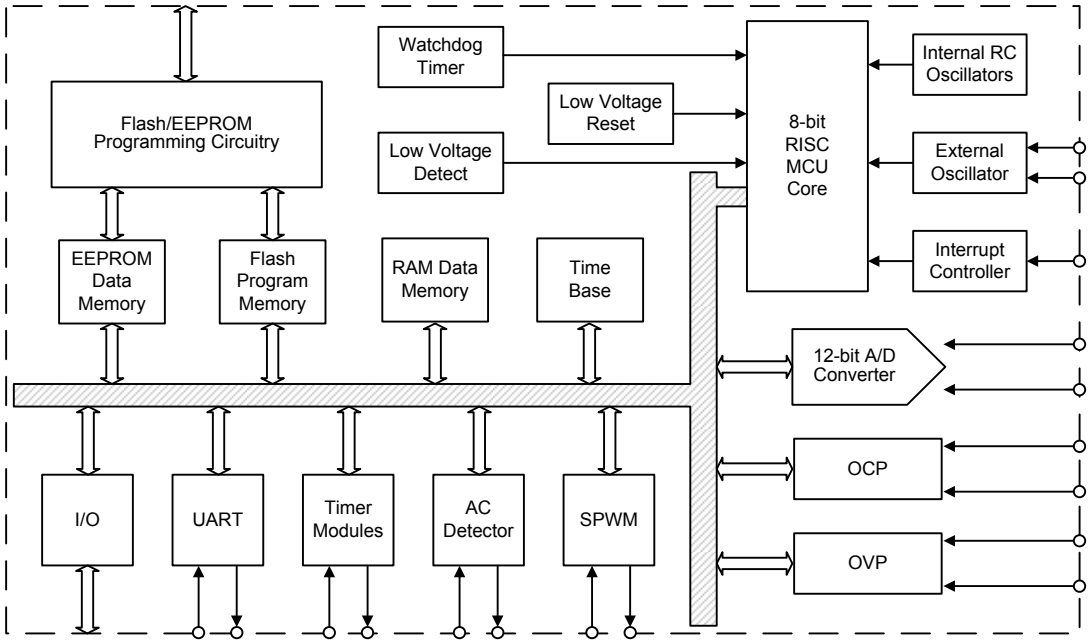
在模拟特性方面，这单片机包含一个多通道 12 位 A/D 转换器功能，弦波 PWM 产生功能。还带有多个使用灵活的定时器模块，可提供定时功能、脉冲产生功能及 PWM 产生等功能。内建的 UART 接口模块可以支持诸如单片机之间的数据通信网络，低成本 PC 和外部设备间的数据连接，便携式和电池供电设备间的通信等。内部看门狗定时器、低电压复位和低电压检测等内部保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地运行。

电源保护方面，该单片机提供了过压保护，过流保护以及交流过零检测功能。

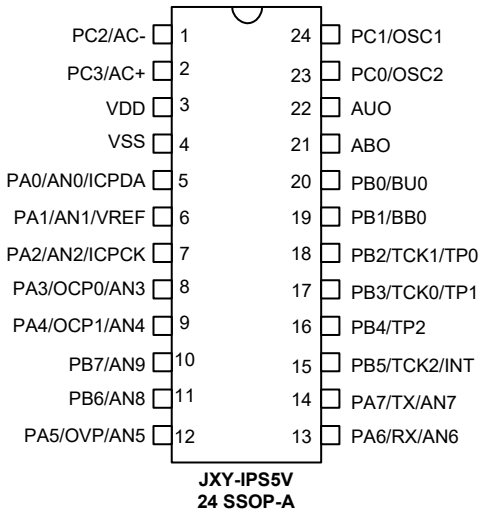
此单片机提供了丰富的内部及外部，高速及低速振荡器功能选项，包含内建完整的系统振荡器，无需外围元器件。其在不同工作模式之间动态切换的能力，为用户提供了一个优化单片机操作和减少功耗的手段。

外加时基功能、I/O 使用灵活等其它特性，使这款单片机可以广泛适用于各种产品，尤其适合于 SPWM 反相输出端控制的应用领域中。

方框图



引脚图



注：若共用脚同时有多种输出，可由相关的软件控制位选择需要的引脚功能。

引脚说明

引脚名称	功能	OP	I/T	O/T	说明
PA0/AN0/ ICPDA	PA0	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	AN0	ADCR0 PAS0	AN	—	A/D 转换器外部信号输入通道
	ICPDA	—	ST	CMOS	ICP 地址 / 数据输入 / 输出
PA1/AN1/ VREF	PA1	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	AN1	ADCR0 PAS0	AN	—	A/D 转换器外部信号输入通道
	VREF	ADCR1 PAS0	AN	—	A/D 转换器参考电压输入
PA2/AN2/ ICPCK	PA2	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	AN2	ADCR0 PAS0	AN	—	A/D 转换器外部信号输入通道
	ICPCK	—	ST	—	ICP 时钟输入
PA3/OCP0/ AN3	PA3	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	OCP0	PAS0	AN	—	OCP 输入
	AN3	ADCR0 PAS0	AN	—	A/D 转换器外部信号输入通道

引脚名称	功能	OP	I/T	O/T	说明
PA4/OCP1/AN4	PA4	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	OCP1	PAS1	AN	—	OCP 输入
	AN4	ADCR0 PAS1	AN	—	A/D 转换器外部信号输入通道
PA5/OVP/AN5	PA5	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	OVP	PAS1	AN	—	OVP 输入
	AN5	ADCR0 PAS1	AN	—	A/D 转换器外部信号输入通道
PA6/RX/AN6	PA6	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	RX	PAS1	ST	—	UART 数据接收端
	AN6	ADCR0 PAS1	AN	—	A/D 转换器外部信号输入通道
PA7/TX/AN7	PA7	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	TX	PAS1	—	CMOS	UART 数据发送端
	AN7	ADCR0 PAS1	AN	—	A/D 转换器外部信号输入通道
PB0/BUO	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	BUO	PBS0	—	CMOS	正弦波 PWM 输出
PB1/BBO	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	BBO	PBS0	—	CMOS	正弦波 PWM 输出
PB2/TCK1/TP0	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TCK1	—	ST	—	TM1 时钟输入
	TP0	PBS0	ST	CMOS	TM0 输入 / 输出
PB3/TCK0/TP1	PB3	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TCK0	—	ST	—	TM0 时钟输入
	TP1	PBS0	ST	CMOS	TM1 输入 / 输出
PB4/TP2	PB4	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TP2	PBS0	ST	CMOS	TM2 输入 / 输出
PB5/TCK2/INT	PB5	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TCK2	—	ST	—	TM2 时钟输入
	INT	—	ST	—	外部中断引脚

引脚名称	功能	OP	I/T	O/T	说明
PB6/AN8	PB6	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AN8	ADCR0 PBS0	AN	—	A/D 转换器外部信号输入通道
PB7/AN9	PB7	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AN9	ADCR0 PBS0	AN	—	A/D 转换器外部信号输入通道
PC0/OSC2	PC0	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OSC2	PCS0	—	HXT	HXT 振荡器引脚
PC1/OSC1	PC1	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OSC1	PCS0	HXT	—	HXT 振荡器引脚
PC2/AC-	PC2	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AC-	PCS0	AN	—	AC 检测输入（负端）
PC3/AC+	PC3	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AC+	PCS0	AN	—	AC 检测输入（正端）
AUO	AUO	—	—	CMOS	正弦波 PWM 输出
ABO	ABO	—	—	CMOS	正弦波 PWM 输出
VDD	VDD	—	PWR	—	正电源
VSS	VSS	—	PWR	—	负电源、接地

注：I/T：输入类型；　　O/T：输出类型
　　OP：通过寄存器选项来配置
　　PWR：电源；　　ST：施密特触发输入
　　CMOS：CMOS 输出
　　AN：模拟输入脚
　　HXT：外部高频晶体振荡器

极限参数

电源供应电压 ..... V_{SS}-0.3V ~ V_{SS}+6.0V
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℃

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_{DD}	工作电压	—	f_{SYS}=20MHz	4.0	—	5.5	V
I_{DD}	工作电流，正常模式 $f_{SYS}=f_H$	5V	无负载，f_H=20MHz, ADC off, WDT 使能 (HXT)	—	4.0	6.2	mA
			无负载，f_H=16MHz, ADC off, WDT 使能 (HIRC)	—	2.8	4.2	mA
	工作电流，低速模式 $f_{SYS}=f_{SUB}$=LIRC	5V	无负载，f_{SYS}=LIRC, ADC off, WDT 使能	—	30	50	μA
I_{STB}	待机电流，SLEEP 模式 (LIRC on)	5V	无负载，ADC off, WDT 使能，LVR 除能	—	2.5	5.0	μA
	待机电流，IDLE0 模式 (LIRC on)	5V	无负载，ADC off, WDT 使能，LVR 除能	—	2.5	5.0	μA
	待机电流，IDLE1 模式	5V	无负载，ADC off, WDT 使能，f_{SYS}=20MHz on (HXT)	—	2.2	3.3	mA
			无负载，ADC off, WDT 使能，f_{SYS}=16MHz on (HIRC)	—	1.4	2.1	mA
V_{IL}	PA, PB, PC, INT, TPn 引脚低电平输入电压	5V	—	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	V
V_{IH}	PA, PB, PC, INT, TPn 引脚高电平输入电压	5V	—	3.5	—	5	V
		—		$0.8V_{DD}$	—	V_{DD}	V
V_{LVR}	低电压复位电压	—	LVR 使能，电压选择 3.8V	-5%	3.8	+5%	V
V_{LVD}	低电压检测电压	—	LVD 使能，电压选择 2.0V	-5%	2.0	+5%	V
		—	LVD 使能，电压选择 2.2V	-5%	2.2	+5%	V
		—	LVD 使能，电压选择 2.4V	-5%	2.4	+5%	V
		—	LVD 使能，电压选择 2.7V	-5%	2.7	+5%	V
		—	LVD 使能，电压选择 3.0V	-5%	3.0	+5%	V
		—	LVD 使能，电压选择 3.3V	-5%	3.3	+5%	V
		—	LVD 使能，电压选择 3.6V	-5%	3.6	+5%	V
		—	LVD 使能，电压选择 4.0V	-5%	4.0	+5%	V
I_{OH}	I/O 口源电流	3V	$V_{OH}=0.9V_{DD}$	-5.5	-11	—	mA
		5V	$V_{OH}=0.9V_{DD}$	-11	-22	—	mA
I_{OL}	I/O 口灌电流	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V	$V_{OL}=0.1V_{DD}$	32	64	—	mA
R_{PH}	I/O 口上拉电阻	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_{SYS}	系统时钟（HXT）	4.0~5.5V	—	—	—	20	MHz
	系统时钟（LIRC）	5V	Ta=25°C	-10%	32	+10%	kHz
		2.2V~5.5V	Ta= -40°C~85°C	-30%	32	+60%	kHz
	系统时钟（HIRC）	4.5V~5.5V	Ta=0°C~85°C	-5%	16	+5%	MHz
t_{TIMER}	TCKn 和 TPn 最小输入脉宽	—	—	0.3	—	—	μs
t_{INT}	中断脉冲宽度	—	—	10	—	—	μs
t_{EERD}	EEPROM 读周期	5V	—	—	2	4	t_{SYS}
t_{EEWR}	EEPROM 写周期	5V	—	—	2	4	ms
t_{SST}	系统启动时间 (从暂停模式唤醒，在暂停模式下 f_{SYS} Off)	—	$f_{SYS}=f_{HXT}$ ~ f_{HXT} / 64	128	—	—	t_{HXT}
		—	$f_{SYS}=f_{HIRC}$ ~ f_{HIRC} / 64	16	—	—	t_{HIRC}
		—	$f_{SYS}=f_{LIRC}$	2	—	—	t_{LIRC}
	系统启动时间 (低速模式 ↔ 正常模式 或 $f_H = f_{HIRC}$ ↔ f_{HXT})	—	f_{HXT} off → on (HXTF=1)	1024	—	—	t_{HXT}
		—	f_{HIRC} off → on (HIRCF=1)	16	—	—	t_{HIRC}
	系统启动时间 (从暂停模式唤醒，在暂停模式下 f_{SYS} On)	—	$f_{SYS}=f_H$ ~ f_H / 64, $f_H=f_{HXT}$ 或 f_{HIRC}	2	—	—	t_H
		—	$f_{SYS}=f_{LIRC}$	2	—	—	t_{SUB}
	系统启动时间 (WDT 溢出硬件冷复位)	—	—	0	—	—	f_H
t_{RSTD}	系统复位延迟时间 (上电复位 , LVR 硬件复位 WDTC 软件复位)	5V	—	25	50	100	ms
	系统复位延迟时间 (WDT 硬件复位)	5V	—	8.3	16.7	33.3	ms

过电压保护电气特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{OVP}	工作电流	3V	OVPEN=1 DAC V_{REF}=2.5V	—	—	TBD	μA
		5V	OVPEN=1 DAC V_{REF}=2.5V	—	280	400	μA
V_{OS}	输入失调电压	3V	校准后	-4	—	4	mV
		5V	校准后	-4	—	4	mV
V_{HYS}	迟滞电压	3V	—	20	40	60	mV
		5V	—	20	40	60	mV
V_{CM}	共模电压范围	3V	—	V_{SS}	—	V_{DD} - 1.4	V
		5V	—	V_{SS}	—	V_{DD} - 1.4	V
DNL	DAC 非线性微分误差	3V	DAC V_{REF}=V_{DD}	—	—	±1	LSB
		5V	DAC V_{REF}=V_{DD}	—	—	±1	LSB
INL	DAC 非线性微分误差	3V	DAC V_{REF}=V_{DD}	—	—	±1.5	LSB
		5V	DAC V_{REF}=V_{DD}	—	—	±1.5	LSB

过电流保护电气特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{OCP}	工作电流	3V	OCPnM[1:0]=01B DAC V_{REF}=2.5V	—	—	TBD	μA
		5V	OCPnM[1:0]=01B DAC V_{REF}=2.5V	—	730	1250	μA
V_{OS_CMP}	比较器输入失调电压	3V	未校准 (OCPnCOF[4:0]=10000B)	-15	—	15	mV
		5V	未校准 (OCPnCOF[4:0]=10000B)	-15	—	15	mV
		3V	校准后	-4	—	4	mV
		5V	校准后	-4	—	4	mV
V_{HYS}	迟滞电压	3V	—	20	40	60	mV
		5V	—	20	40	60	mV
V_{CM_CMP}	比较器共模电压范围	3V	—	V_{SS}	—	V_{DD} - 1.4	V
		5V	—	V_{SS}	—	V_{DD} - 1.4	V
V_{OS_OPA}	OPA 输入失调电压	3V	未校准 (OCPnAOF[5:0]=100000B)	-15	—	15	mV
		5V	未校准 (OCPnAOF[5:0]=100000B)	-15	—	15	mV
		3V	校准后	-4	—	4	mV
		5V	校准后	-4	—	4	mV
V_{CM_OPA}	OPA 共模电压范围	3V	—	V_{SS} + 0.2	—	V_{DD} - 1.4	V
		5V	—	V_{SS} + 0.2	—	V_{DD} - 1.4	V
Gain	OPA 增益误差	3V	所有增益	-5	—	5	%
		5V	所有增益	-5	—	5	%
DNL	DAC 非线性微分误差	3V	DAC V_{REF}=V_{DD}	—	—	±1	LSB
		5V	DAC V_{REF}=V_{DD}	—	—	±1	LSB
INL	DAC 非线性微分误差	3V	DAC V_{REF}=V_{DD}	—	—	±1.5	LSB
		5V	DAC V_{REF}=V_{DD}	—	—	±1.5	LSB

A/D 转换器电气特性

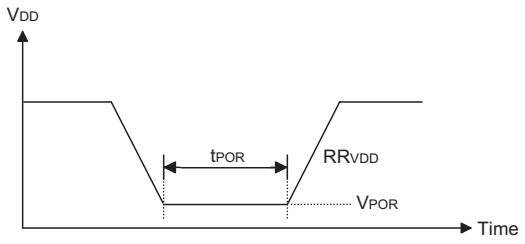
Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
AV_{DD}	A/D 转换器工作电压	—	—	2.7	—	5.5	V
V_{AD}	A/D 转换器输入电压	—	—	0	—	AV_{DD}/ V_{REF}	V
V_{REF}	A/D 转换器参考电压	—	—	2	—	AV_{DD}	V
DNL	A/D 转换器非线性微分误差	3V/5V	V_{REF}=AV_{DD}=V_{DD} t_{AD}=0.5μs/10μs	-3	—	+3	LSB
INL	A/D 转换器非线性积分误差	3V/5V	V_{REF}=AV_{DD}=V_{DD} t_{AD}=0.5μs/10μs	-4	—	+4	LSB
I_{ADC}	A/D 转换器使能的额外电流	3V	无负载，t_{AD}=0.5μs	—	1.0	2.0	mA
		5V		—	1.5	3.0	mA
t_{AD}	A/D 转换器时钟周期	2.7~5.5V	—	0.5	—	10	μs
t_{ADC}	A/D 转换时间 (包括采样和保持时间)	2.7~5.5V	12-bit ADC	16	—	20	t_{AD}
t_{ADS}	A/D 转换采样时间	2.7~5.5V	—	—	4	—	t_{AD}
t_{ON2ST}	A/D 转换器 On-to-Start 时间	2.7~5.5V	—	4	—	—	μs

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	—	—	10	—	—	μs

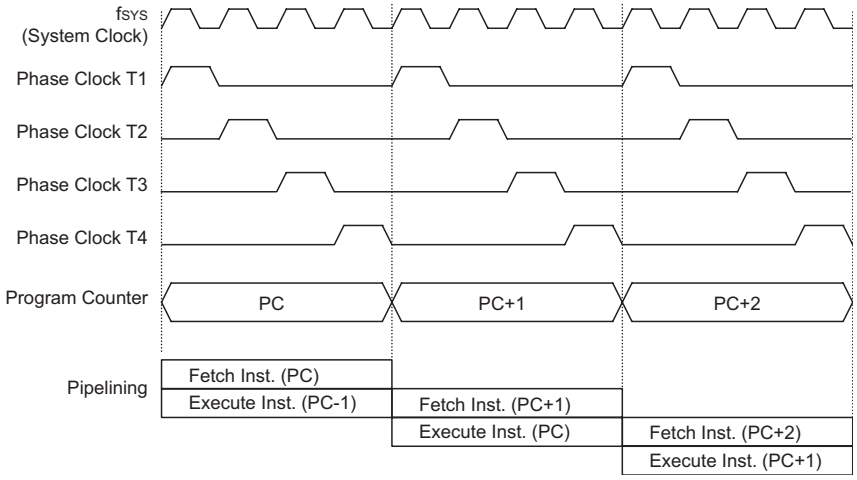


系统结构

内部系统结构是该单片机具有良好性能的主要因素。由于采用 RISC 结构，该单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的取得和执行同时进行，此举使得除了跳转和调用指令需多一个指令周期外，其它大部分标准指令或扩展指令都能分别在一个或两个指令周期内完成。8 位 ALU 参与指令集中所有的运算，它可完成算术运算、逻辑运算、移位、递增、递减和分支等功能，而内部的数据路径则是以通过累加器和 ALU 的方式加以简化。有些寄存器在数据存储器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供具有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大量生产的控制应用。

时序和流水线结构

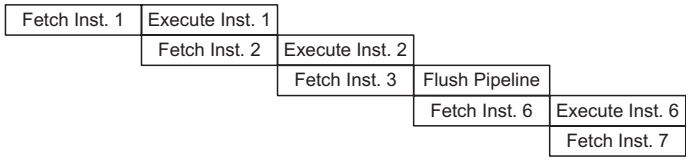
主系统时钟由 HXT、HIRC 或 LIRC 振荡器提供，它被细分为 T1~T4 四个内部产生的非重叠时序。在 T1 时间，程序计数器自动加一并抓取一条新的指令。剩下的时间 T2~T4 完成译码和执行功能，因此，一个 T1~T4 时钟周期构成一个指令周期。虽然指令的抓取和执行发生在连续的指令周期，但单片机流水线结构会保证指令在一个指令周期内被有效执行。除非程序计数器的内容被改变，如子程序的调用或跳转，在这种情况下指令将需要多一个指令周期的时间去执行。



系统时序和流水线

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

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



指令捕捉

程序计数器

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

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

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

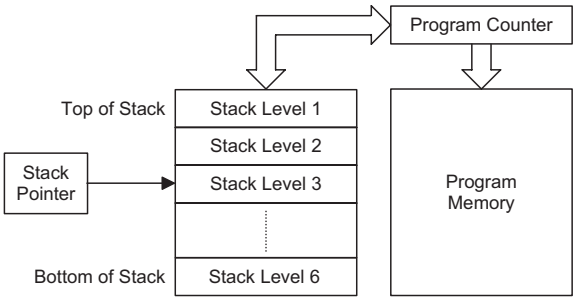
程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。程序计数器的低字节可由程序直接进行读取，PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

堆栈

堆栈是一个特殊的存储空间，用来存储程序计数器中的内容。该单片机有 6 层堆栈，堆栈既不是数据部分也不是程序空间部分，而且它既不是可读取也不是可写入的。当前层由堆栈指针 (SP) 加以指示，同样也是不可读写的。在子程序调用或中断响应服务时，程序计数器的内容被压入到堆栈中。当子程序或中断响应结束时，返回指令 (RET 或 RETI) 使程序计数器从堆栈中重新得到它以前的值。当一个芯片复位后，堆栈指针将指向堆栈顶部。

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

若堆栈溢出，则首个存入堆栈的程序计数器数据将会丢失。



算术逻辑单元 – ALU

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

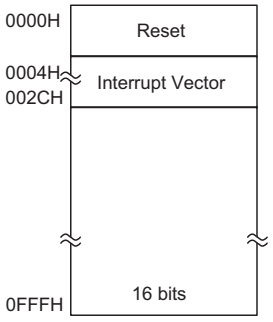
- 算术运算：ADD，ADDM，ADC，ADCM，SUB，SUBM，SBC，SBCM，DAA，LADD，LADDM，LADC，LADCM，LSUB，LSUBM，LSBC，LSBCM，LDAA
- 逻辑运算：AND，OR，XOR，ANDM，ORM，XORM，CPL，CPLA，LAND，LANDM，LOR，LORM，LXOR，LXORM，LCPL，LCPLA
- 移位运算：RRA，RR，RRCA，RRC，RLA，RL，RLCA，RLC，LRR，LRRA，LRRCA，LRRC， LRLA，LRLCA，LRLC
- 递增和递减：INCA，INC，DECA，DEC，LINCA，LINC，LDECA，LDEC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

Flash 程序存储器

程序存储器用来存放用户代码或程序。此单片机的程序存储器为 Flash 类型即意味着可以进行多次重复编程，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

此单片机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。



程序存储器结构

特殊向量

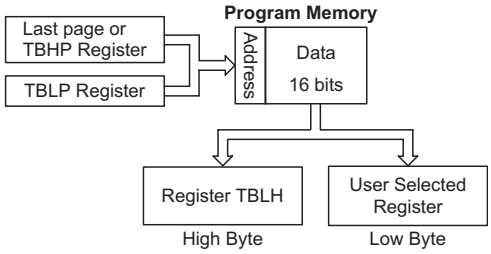
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 0000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

查表

程序存储器中的任何地址都可以定义成一个表格，以便储存固定的数据。使用表格时，表格指针必须先行设定，其方式是将表格的地址放在表格指针寄存器 TBLP 和 TBHP 中。这两个寄存器可以定义整个表格地址。

在设定完表格指针后，若数据存储器 [m] 位于数据存储器 Sector0，表格数据可以使用“TABRD [m]”或“TABRDL [m]”指令分别从程序存储器查表读取。如果存储器 [m] 位于数据存储器其它 Sector，表格数据可以使用“LTABRD [m]”或“LTABRDL [m]”指令分别从程序存储器查表读取。当这些指令执行时，程序存储器中表格数据低字节，将被传送到使用者所指定的数据存储器 [m]，程序存储器中表格数据的高字节，则被传送到 TBLH 特殊寄存器。

下图是查表中寻址 / 数据流程：



查表范例

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

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

表格读取程序范例

```
tempreg1 db ?       ; temporary register #1
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 tblp pointed
mov a,0fh           ; initialise high table pointer
mov tbhp,a          ; it is not necessary to set tbhp if executing tabrdl
:
:
tabrdl tempreg1     ; transfers value in table referenced by table pointer
                    ; to tempreg1
```

```
                    ; Data at program memory address "0F06H" transferred to
                    ; tempreg1 and TBLH
dec tblp            ; reduce value of table pointer by one
tabrdl tempreg2     ; transfers value in table referenced by table pointer
                    ; to tempreg2
                    ; Data at program memory address "0F05H" transferred to
                    ; tempreg2 and TBLH
                    ; in this example the data "1AH" is transferred to
                    ; tempreg1 and data "0FH" to
                    ; register tempreg2 while the value "00H" will be
                    ; transferred to the high byte register TBLH
:
:
org 0F00h           ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

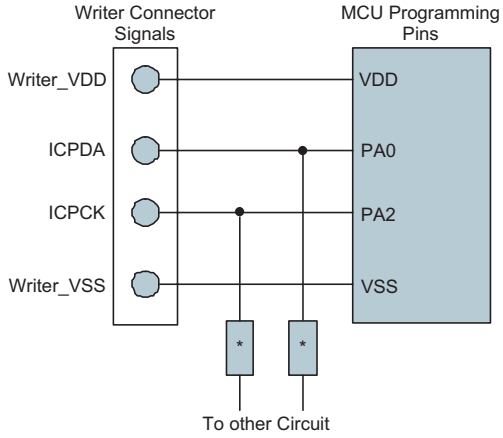
在线烧录

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

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

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

烧录过程中，用户必须确保 ICPDA 和 ICPCK 这两个数据和时钟烧录引脚没有连接其它输出脚。



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。

数据存储器

数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。

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

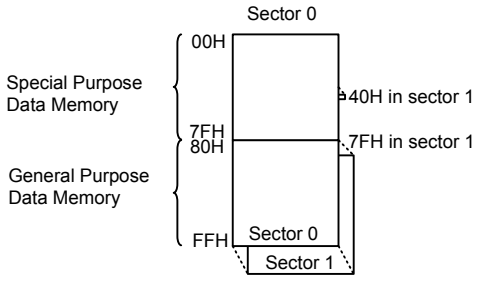
切换不同的数据存储器 Sector 可通过设置正确的存储器指针值实现。

结构

数据存储器被分为 2 个 Sector，都位于 8 位存储器中。数据存储器的 Sector 0 分为两类，特殊功能数据存储器和通用数据存储器。

特殊功能数据存储器位于 Sector 0 和 Sector 1 的 00H ~ 7FH，而通用数据存储器的地址范围为 Sector 0 和 Sector 1 的 80H ~ FFH。大部分特殊功能数据寄存器均可在 Sector 0 被访问，而寄存器 EEC 却只能在 Sector 1 中被访问到。

容量	Sector
通用数据存储器：256×8	0：80H~FFH 1：80H~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 1 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 "SET [m].i" 和 "CLR [m].i" 位操作指令可对个别的位做置位或复位的操作，极大地方便了用户在数据存储器内进行位操作。

特殊功能数据存储器

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

	Sector 0	Sector 1
00H	IAR0	Unused
01H	MP0	Unused
02H	IAR1	Unused
03H	MP1L	Unused
04H	MP1H	Unused
05H	ACC	Unused
06H	PCL	Unused
07H	TBLP	Unused
08H	TBLH	Unused
09H	TBHP	Unused
0AH	STATUS	Unused
0BH	Unused	Unused
0CH	IAR2	Unused
0DH	MP2L	Unused
0EH	MP2H	Unused
0FH	Unused	Unused
10H	INTC0	Unused
11H	Unused	Unused
12H	PA	Unused
13H	PAC	Unused
14H	PAPU	Unused
15H	PAWU	Unused
16H	Unused	Unused
17H	RSTFC	Unused
18H	Unused	Unused
19H	Unused	Unused
1AH	PB	Unused
1BH	PBC	Unused
1CH	PBPU	Unused
1DH	PC	Unused
1EH	PCC	Unused
1FH	PCPU	Unused

	Sector 0	Sector 1
20H	TBC0	Unused
21H	TBC1	Unused
22H	PSCR	Unused
23H	WDTC	Unused
24H	LVDC	Unused
25H	Unused	Unused
26H	RSTC	Unused
27H	Unused	Unused
28H	ADRL	Unused
29H	ADRH	Unused
2AH	ADCR0	Unused
2BH	ADCR1	Unused
2CH	SCC	Unused
2DH	HXTC	Unused
2EH	HIRCC	Unused
2FH	Unused	Unused
30H	INTEG	Unused
31H	INTC1	Unused
32H	INTC2	Unused
33H	MFI0	Unused
34H	MFI1	Unused
35H	MFI2	Unused
36H	Unused	Unused
37H	Unused	Unused
38H	Unused	Unused
39H	Unused	Unused
3AH	PAS0	Unused
3BH	PAS1	Unused
3CH	PBS0	Unused
3DH	IFS0	Unused
3EH	PCS0	Unused
3FH	Unused	Unused

	Sector 0	Sector 1
40H	Unused	EEC
41H	EEA	Unused
42H	EED	Unused
43H	TM0C0	Unused
44H	TM0C1	Unused
45H	TM0DL	Unused
46H	TM0DH	Unused
47H	TM0AL	Unused
48H	TM0AH	Unused
49H	TM1C0	Unused
4AH	TM1C1	Unused
4BH	TM1DL	Unused
4CH	TM1DH	Unused
4DH	TM1AL	Unused
4EH	TM1AH	Unused
4FH	TM2C0	Unused
50H	TM2C1	Unused
51H	TM2DL	Unused
52H	TM2DH	Unused
53H	TM2AL	Unused
54H	TM2AH	Unused
55H	TM2RP	Unused
56H	Unused	Unused
57H	Unused	Unused
58H	USR	Unused
59H	UCR1	Unused
5AH	UCR2	Unused
5BH	BRG	Unused
5CH	TXR/RXR	Unused
5DH	Unused	Unused
5EH	Unused	Unused
5FH	Unused	Unused

	Sector 0	Sector 1
60H	SPWMC0	Unused
61H	SPWMC1	Unused
62H	SPWML	Unused
63H	SPWMH	Unused
64H	SFIFOL	Unused
65H	SFIFOH	Unused
66H	MTF	Unused
67H	SPWMDT	Unused
68H	ACDC	Unused
69H	ACDOFF	Unused
6AH	OCP0C0	Unused
6BH	OCP0C1	Unused
6CH	OCP0REF	Unused
6DH	OCP0ACAL	Unused
6EH	OCP0CCAL	Unused
6FH	OCP1C0	Unused
70H	OCP1C1	Unused
71H	OCP1REF	Unused
72H	OCP1ACAL	Unused
73H	OCP1CCAL	Unused
74H	OVPC	Unused
75H	OVPREF	Unused
76H	OVPCCAL	Unused
77H	Unused	Unused
78H	Unused	Unused
79H	Unused	Unused
7AH	Unused	Unused
7BH	Unused	Unused
7CH	Unused	Unused
7DH	Unused	Unused
7EH	Unused	Unused
7FH	Unused	Unused

: 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间接寻址指针做数据操作，以取代定义实际存储器地址的直接存储器寻址方法。在间接寻址寄存器 IAR0、IAR1 和 IAR2 上的任何动作，将对间接寻址指针 MP0、MP1L/MP1H 或 MP2L/MP2H 所指定的存储器地址产生对应的读 / 写操作。它们总是成对出现，IAR0 和 MP0 可以访问 Sector 0，而 IAR1 和 MP1L/MP1H、IAR2 和 MP2L/MP2H 可以访问任何 Sector。因为这些间接寻址寄存器不是实际存在的，直接读取将返回“00H”的结果，而直接写入此寄存器则不做任何操作。

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 MP1
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/W	R/W	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

该单片机内建 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 中的两个地址寄存器和一个数据寄存器以及 Sector 1 中的一个控制寄存器，可以实现对 EEPROM 的单字节读写操作。

EEPROM 寄存器

有三个寄存器控制内部 EEPROM 数据存储器总的操作。地址寄存器 EEA、数据寄存器 EED 及控制寄存器 EEC。EEA 和 EED 位于 Sector 0 中，它们能像其它特殊功能寄存器一样直接被访问。EEC 位于 Sector 1 中，不能被直接访问，仅能通过 MP1L/MP1H 和 IAR1 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/MP2L 必须先设为“40H”，MP1H/MP2H 被设为“01H”。

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

EEPROM 寄存器列表

EEA 寄存器

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

Bit 7 ~ 6　未定义，读为“0”

Bit 5 ~ 0　**D5~D0**：数据 EEPROM 地址
数据 EEPROM 地址 Bit 5~Bit 0

EED 寄存器

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

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

EEC 寄存器

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

Bit 7～4　未定义，读为“0”

Bit 3　**WREN**：数据 EEPROM 写使能位
0：除能
1：使能
此位为数据 EEPROM 写使能位，在执行数据 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：激活写周期
此位为数据 EEPROM 写控制位，由应用程序将此位置高将开始一个写周期。写周期结束后，硬件自动将此位清零。若 WREN 位没有提前置高的话，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，执行数据 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：激活读周期
此位为数据 EEPROM 读控制位，由应用程序将此位置高将开始一个读周期。读周期结束后，硬件自动将此位清零。若 RDEN 位没有提前置高的话，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 位还未被置高则不能开始读操作。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，将要写入的数据需存入 EED 寄存器中。EEC 寄存器中的写使能位 WREN 先置为高以使能写操作，然后 EEC 寄存器中 WR 位需立即置为高以开始一个写周期，这两条指令须连续执行。在进行写操作之前应先将总中断使能位 EMI 清零，写周期开始后再将其置位。若 WR 位已置为高而 WREN 位还未被置高则不能开始写操作。由于控制 EEPROM 写周期是一个内部时钟，与单片机的系统时钟异步，所以数据写入 EEPROM 的时间将有所延迟。可通过轮询 EEC 寄存器中的 WR 位或判断 EEPROM 中断以侦测写周期是否完成。若写周期完成，WR 位将

自动清除为“0”，通知用户数据已写入 EEPROM。因此，应用程序将轮询 WR 位以确定写周期是否结束。

写保护

防止误写入的写保护有以下几种。单片机上电后控制寄存器中的写使能位将被清除以杜绝任何写入操作。上电后存储器指针对，MP1L/MP1H 和 MP2L/MP2H 将重置为“0”，这意味着数据存储区 Sector 0 被选中。由于 EEPROM 控制寄存器位于 Sector 1 中，这增加了对写操作的保护措施。在正常程序操作中确保控制寄存器中的写使能位被清除将能防止不正确的写操作。

EEPROM 中断

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

编程注意事项

必须注意的是数据不会无意写入 EEPROM。在没有写动作时写使能位被正常清零可以增强保护功能。存储器指针高字节 MP1H 或 MP2H 也可以正常清零以阻止进入 EEPROM 控制寄存器存在的 Sector 1。尽管没有必要，写一个简单的读回程序以检查新写入的数据是否正确还是应该考虑的。WREN 位置位后，EEC 寄存器中的 WR 位需立即置位，否则 EEPROM 写周期将不能被执行。

写数据时，WREN 位置为“1”后，WR 须立即设置为高，以确保正确地执行写周期。写周期执行前总中断位 EMI 应先清零，写周期开始执行后再将此位重新置 1。注意，单片机应在 EEPROM 读或写操作完全完成之后才能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 MP1L
MOV MP1L, A                        ; MP1L points to EEC register
MOV A, 01H                         ; setup Section Pointer MP1H
MOV MP1H, A
SET IAR1.1                         ; set EERDEN bit, enable read operations
SET IAR1.0                         ; start Read Cycle - set EERD bit
BACK:
SZ IAR1.0                          ; check for read cycle end
JMP BACK
CLR IAR1                           ; disable EEPROM read/write
CLR MP1H
MOV A, EED                         ; move read data to register
MOV READ_DATA, A
```

- **写数据到 EEPROM – 轮询法**

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

振荡器

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

振荡器概述

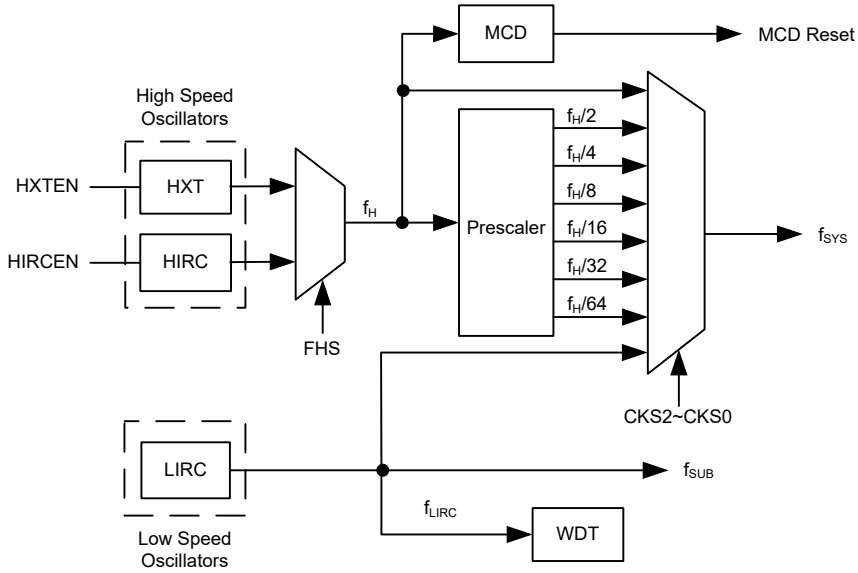
振荡器除了作为系统时钟源，还作为看门狗定时器和时基功能的时钟源。外部振荡器需要一些外围器件，而集成的两个内部振荡器不需要任何外围器件。它们提供高速和低速系统振荡器具有较宽的频率范围。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

类型	名称	频率	引脚
外部高速晶振	HXT	20MHz	OSC1/OSC2
内部高速 RC	HIRC	16MHz	—
内部低速 RC	LIRC	32kHz	—

振荡器类型

系统时钟配置

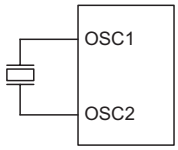
该单片机有三个系统振荡器，包括两个高速振荡器和一个低速振荡器。高速振荡器为内部 16MHz RC 振荡器 (HIRC) 和外部高速晶振 (HXT)。低速振荡器为内部 32kHz RC 振荡器 (LIRC)。使用高速或低速振荡器作为系统时钟的选择是通过设置 SCC 寄存器中的 CKS2~CKS0 位决定的，系统时钟可动态选择。请注意，两个振荡器必须做出选择，即一个高速和一个低速振荡器。



系统时钟配置

外部晶体振荡器 – HXT

HXT 是一个外部高频晶体振荡器，可通过 SCC 寄存器的 FHS 位进行选择。时钟频率固定为 20MHz。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，而不需要其它外部器件。



外部晶体振荡器 – HXT

内部 RC 振荡器 – HIRC

内部 RC 振荡器是一个完全内部集成的 RC 振荡器，不需其它外部器件。内部 RC 振荡器具有一固定频率：16MHz。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。如果选择了该内部时钟，无需额外的引脚，I/O 引脚可以作为通用 I/O 口使用。

内部 32kHz 振荡器 – LIRC

内部 32kHz 系统振荡器也是一个完全内部集成的 RC 振荡器。它在 5V 电压下运行的典型频率值为 32kHz 且无需外部元件。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响减至最低。因此，内部 32kHz 振荡器频率在 25°C 温度 5V 电压下的精度保持在 10% 以内。

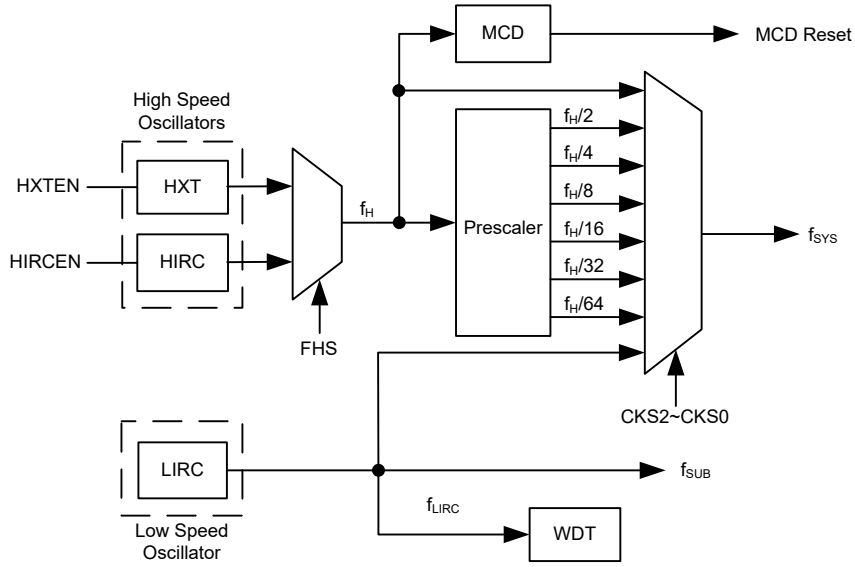
工作模式和系统时钟

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

系统时钟

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

主系统时钟可来自高频时钟源 f_H 或低频时钟源 f_{SUB}，通过 SCC 寄存器中的 CKS2~CKS0 位进行选择。高频时钟来自 HXT 或 HIRC 振荡器，通过 SCC 寄存器的 FHS 位选择。低频系统时钟源来自 LIRC 振荡器。其它系统时钟还有高速系统振荡器的分频 $f_H/2$~$f_H/64$。



时钟配置

注：当系统时钟源 f_{SYS} 由 f_H 到 f_{SUB} 转换时，高速振荡器可通过程序设置关闭将停止以节省耗电，或通过设置对应高频振荡器使能控制位继续为外围电路提供 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、空闲模式 0、空闲模式 1 和空闲模式 2 用于单片机 CPU 关闭时以节省耗电。

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

注：1. 在低速模式下，f_H 时钟开启或关闭由对应高频振荡器使能位控制。
2. 因 LIRC 为 WDT 功能提供时钟，休眠模式下，由于看门狗定时器继续使能，f_{LIRC} 时钟信号开启。

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 SCC 寄存器中的 CKS2~CKS0 位选择的。单片机使用高速振荡器分频作为系统时钟可减少工作电流。

低速模式

此模式的系统时钟虽为较低速时钟源，但单片机仍能正常工作。该低速时钟源 f_{SUB} 来自 LIRC 振荡器。单片机在此模式中运行所耗工作电流较低。在低速模式下，f_H 可通过应用程序关闭。

休眠模式

在 HALT 指令执行后且 SCC 寄存器中 FHIDEN 和 FSIDEN 位为低时，系统进入休眠模式。在休眠模式中，CPU 停止运行。由于 WDT 始终使能，f_{LIRC} 可继续运行。

空闲模式 0

执行 HALT 指令后且 SCC 寄存器中 FHIDEN 位为低，FSIDEN 位为高，系统进入空闲模式 0。在空闲模式 0 中，CPU 停止，但低速振荡器将开启以驱动一些外围功能继续工作。

空闲模式 1

执行 HALT 指令后且 SCC 寄存器中 FHIDEN 位为高，FSIDEN 位为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，高速和低速系统振荡器都会开启提供一个时钟源给一些外围功能。

空闲模式 2

执行 HALT 指令后且 SCC 寄存器中 FHIDEN 位为高， FSIDEN 位为低时，系统进入空闲模式 2。在空闲模式 2 中，CPU 停止，但高速振荡器会开启以保持一些外围功能继续工作。

MCD (Missing Clock Detector) 功能

该单片机支持 MCD 功能，用于检测高频振荡器使能后的操作。若高频振荡器使能后一定时间内未检测到时钟，则说明振荡器未成功起振，MCD 功能将产生复位信号将单片机复位。

控制寄存器

寄存器 SCC 用于整体控制单片机内部时钟。寄存器 HXTC 用于对 HXT 振荡器控制，HIRCC 寄存器为 HIRC 振荡器控制寄存器。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	FHS	—	FHIDEN	FSIDEN
HXTC	—	—	—	—	—	—	HXTF	HXTEN
HIRCC	—	—	—	—	—	—	HIRCF	HIRCEN

系统工作模式控制寄存器列表

SCC 寄存器

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

Bit 7 ~ 5 **CKS2 ~ CKS0**：系统时钟选择位

000：f_H
001：$f_H/2$
010：$f_H/4$
011：$f_H/8$
100：$f_H/16$
101：$f_H/32$
110：$f_H/64$
111：f_{SUB}

这三位用于选择系统时钟源。除了由振荡器直接提供的系统时钟源 f_H 或 f_{SUB} 外，也可使用高频振荡器的分频作为系统时钟。

Bit 4 未定义，读为“0”

Bit 3 **FHS**：高频时钟选择位

0：HIRC
1：HXT

Bit 2 未定义，读为“0”

Bit 1 **FHIDEN**：CPU 停止时高频时钟控制位

0：除能
1：使能

Bit 0 **FSIDEN**：CPU 停止时低频时钟控制位
0：除能
1：使能
此位用来控制在执行 HALT 指令 CPU 关闭后低速振荡器是否停止。若选择 LIRC 作为低速振荡器时钟源，刚 LIRC 振荡器是由该位与 WDT 功能控制位共同控制的。但因 WDT 功能始终使能，即使该位被清零，LIRC 振荡器也会继续运行。

HXTC 寄存器

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

Bit 7~2 未定义，读为“0”

Bit 1 **HXTF**：HXT 时钟稳定标志位
0：未稳定
1：稳定
此位用于指示 HXT 振荡器是否稳定。当设置 HXTEN 为“1”使能 HXT 振荡器后，HXTF 会自动清零。当 HXT 时钟稳定下来后该位被置位。

Bit 0 **HXTEN**：HXT 振荡器控制位
0：除能
1：使能

HIRCC 寄存器

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

Bit 7~2 未定义，读为“0”

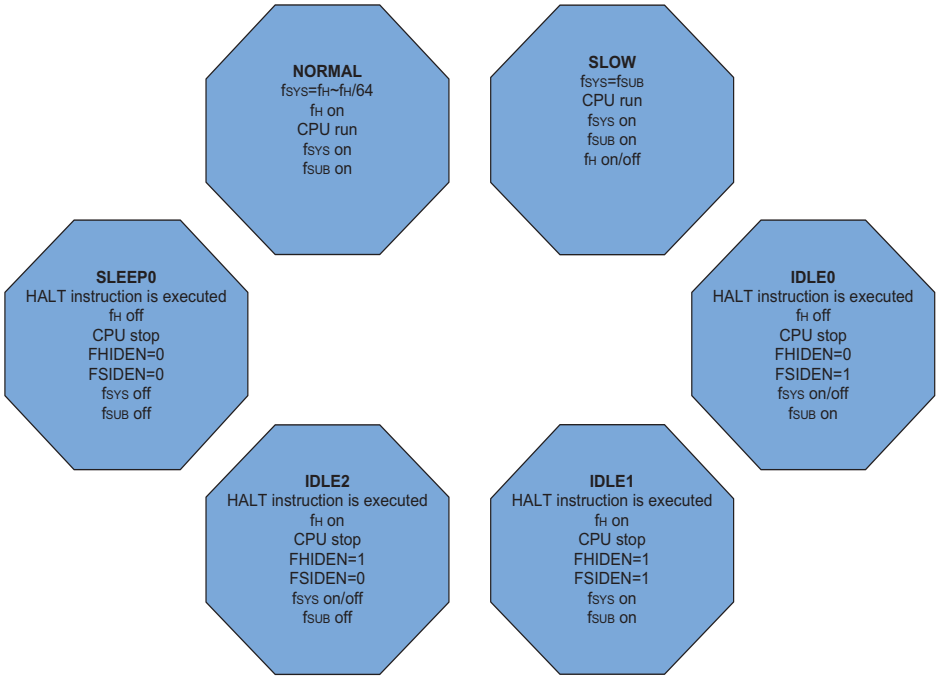
Bit 1 **HIRCF**：HIRC 时钟稳定标志位
0：未稳定
1：稳定
此位用于指示 HIRC 振荡器是否稳定。当设置 HIRCEN 为“1”使能 HIRC 振荡器后，HIRCF 位会自动清零。当 HIRC 时钟稳定下来后该位被置位。HIRC 振荡器从使能到稳定下来需要 16 个时钟周期。

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

工作模式切换

单片机可在各个工作模式间自由切换，使得用户可根据所需选择最佳的性能 / 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。

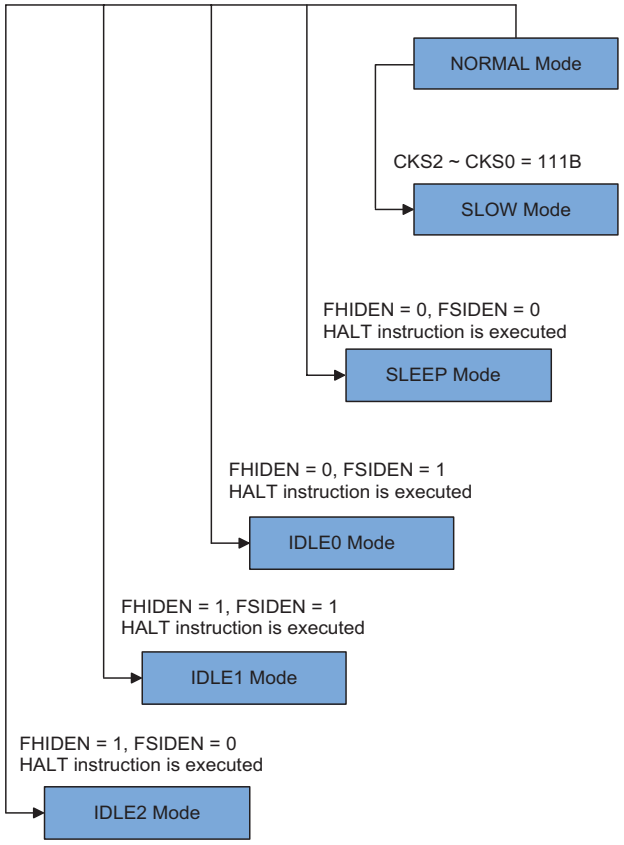
简单来说，正常模式和低速模式间的切换仅需设置 SCC 中的 CKS2~CKS0 位即可实现，而正常模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 位和 FSIDEN 位决定的。



正常模式切换到低速模式

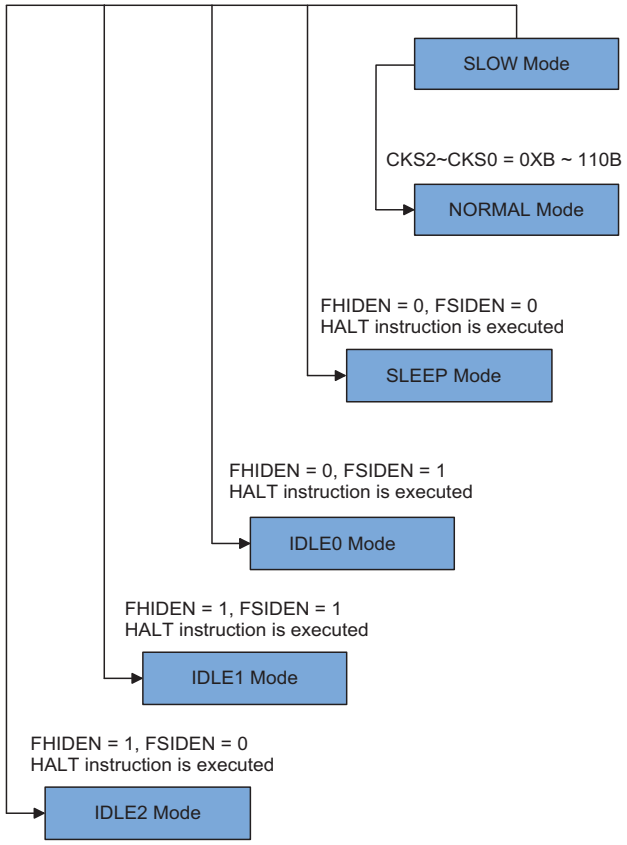
系统运行在正常模式时使用高速系统振荡器，因此较为耗电。可通过设置 SCC 寄存器中的 CKS2~CKS0 位为“111”使系统时钟切换至运行在低速模式下。此时将使用低速系统振荡器以节省耗电。用户可在对性能要求不高的操作中使用此方法以减少耗电。

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



低速模式切换到正常模式

在低速模式系统使用 LIRC 低速振荡器。切换到使用高速系统时钟振荡器的正常模式需设置 CKS2~CKS0 为“000”~“110”。高频时钟需要一定的稳定时间，通过检测 HXTF 或 HIRCF 位的状态可进行判断。



进入休眠模式

进入休眠模式的方法仅有一种——应用程序中执行“HALT”指令前需设置寄存器 SCC 中 FHIDEN 位和 FSIDEN 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。
- 数据存储器中的内容和寄存器将保持当前值。
- WDT 将被清零并重新计数。
- 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 SCC 中 FSIDEN 位为“1”且 FHIDEN 位为“0”。在上述条件下执行该指令后，将发生的情况如下：

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

进入空闲模式 1

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

- f_H 时钟和低频时钟 f_{SUB} 都开启，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。
- 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 2

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

- f_H 时钟开启而低频时钟 f_{SUB} 关闭，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 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单片机的电流降低到尽可能低，可能到只有几个微安的级别（空闲模式 1 和空闲模式 2 除外），所以如果要将电路的电流降到最低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入 / 输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。

另外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 LIRC 振荡器使能，会导致耗电增加。在空闲模式 1 和空闲模式 2 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，驱动 CPU 的系统时钟将停止运行。当单片机唤醒后，之前的系统时钟从重新启动到稳定允许正常操作需要一定时间。

系统进入休眠或空闲模式之后，可以通过以下几种方式唤醒：

- PA 口下降沿
- 系统中断
- WDT 溢出

系统上电或执行清除看门狗的指令，会清零 PDF；执行 HALT 指令，PDF 将被置位。看门狗计数器溢出将会置位 TO 标志并唤醒系统，这种复位会重置程序计数器和堆栈指针，其它标志保持原有状态。

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

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源来自于内部时钟由 LIRC 振荡器提供。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。

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、复位及选择溢出周期。

WDTC 寄存器

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

Bit 7~ 3 **WE4 ~ WE0**：WDT 功能软件控制

10101 或 01010：使能

其它值：MCU 复位

若因外部环境因素使这些位改写为除 10101 和 01010 外的其它值时，将引起单片机复位，需要 2~3 个 LIRC 周期响应复位。且 RSTFC 寄存器中的 WRF 标志位会被置位。

Bit 2~ 0 **WS2 ~ WS0**：WDT 溢出周期选择位

000：$2^8/f_{LIRC}$

001：$2^{10}/f_{LIRC}$

010：$2^{12}/f_{LIRC}$

011：$2^{14}/f_{LIRC}$

100：$2^{15}/f_{LIRC}$

101：$2^{16}/f_{LIRC}$

110：$2^{17}/f_{LIRC}$

111：$2^{18}/f_{LIRC}$

这三位控制 WDT 时钟源的分频比，从而实现对 WDT 溢出周期的控制。

看门狗定时器操作

当 WDT 溢出时，看门狗定时器产生一个芯片复位的动作。这也就意味着正常工作期间，用户需在应用程序中看门狗溢出前有策略地清看门狗定时器以防止其产生复位，可使用清除看门狗指令实现。无论什么原因，程序失常跳转到一个未知的地址或进入一个死循环，看门狗清除指令都不能被正确执行，此种情况下，看门狗将溢出以使单片机复位。WDTC 寄存器中的 WE4~WE0 位可提供使能控制以及控制看门狗定时器复位操作。如果 WE4~WE0 设置为“10101B”或“01010B”，则 WDT 使能；如果 WE4~WE0 设置为除“10101B”和“01010B”以外的其它任意值，则经过 2~3 个 f_{LIRC} 时钟周期后单片机复位。上电后这些位初始化为“01010B”。

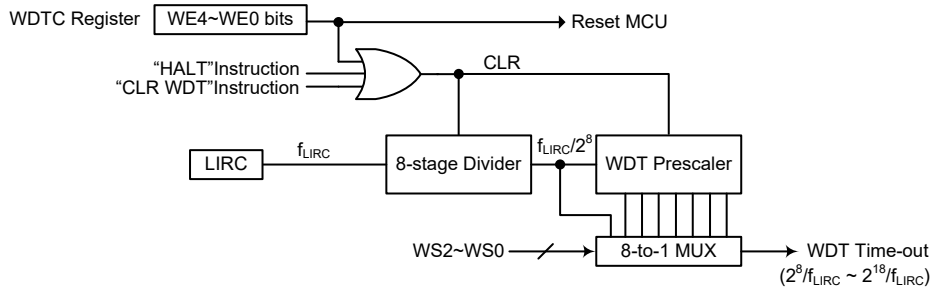
WE4 ~ WE0 位	WDT 功能
10101B 或 01010B	使能
其它值	MCU 复位

看门狗定时器使能控制

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，仅程序计数器 PC 和堆栈指针 SP 复位。有三种方法可以用来清除 WDT 的内容。第一种是 WDT 软件复位，即将 WE4~WE0 位设置成除了“01010B”和“10101B”外的任意值；第二种是通过看门狗定时器软件清除指令，而第三种是通过“HALT”指令。

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

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

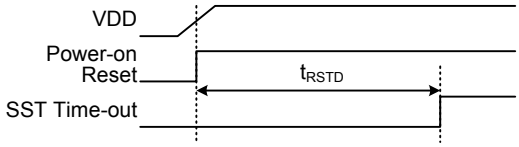
另一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。

复位功能

包括内部和外部事件触发复位，单片机有多种复位方式：

上电复位

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

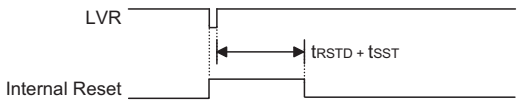


注：t_{RSTD} 为上电延迟时间，典型值为 50ms。

上电复位时序图

低电压复位 – LVR

单片机具有低电压复位电路，用来监测它的电源电压。低电压复位功能始终使能于 3.8V。例如在更换电池的情况下，单片机供应的电压可能会在 0.9V~V_{LVR} 之间，这时 LVR 将会自动复位单片机且 RSTFC 寄存器中的 LVRF 标志位置位。LVR 包含以下的规格：有效的 LVR 信号，即在 0.9V~V_{LVR} 的低电压状态的时间，必须超过交流电气特性中 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。正常执行时 LVR 会于休眠或空闲时自动除能关闭。



注：t_{RSTD} 为上电延迟时间，典型值为 50ms。

低电压复位时序图

- **RSTFC 寄存器**

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

“x”为未知

Bit 7~4 未定义，读为“0”

Bit 3 **ERSTF**：RSTC 寄存器软件复位标志位
0：未发生
1：发生
当 RSTC 寄存器被改写为未定义的值发生软件复位时此位设置为“1”，可通过应用程序清零。

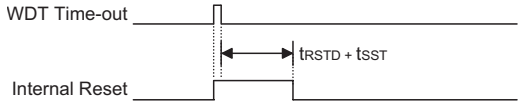
Bit 2 **LVRF**：LVR 复位标志
0：未发生
1：发生
当发生低电压复位时此位被置为“1”，可通过应用程序清零。

Bit 1 未定义，读为“0”

Bit 0 **WRF**：WDT 控制寄存器软件复位标志
0：未发生
1：发生
当 WDT 控制寄存器软件复位发生时，此位被置为“1”，且只能通过应用程序清零。

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。

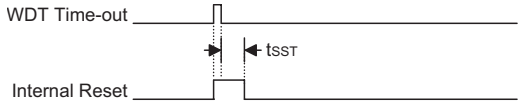


注：t_{RSTD} 为上电延迟时间，典型值为 16.7ms。

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：若系统时钟来自 HIRC, 则 t_{SST}=15~16 时钟周期
若系统时钟来自 HXT, 则 t_{SST}=128 时钟周期
若系统时钟来自 LIRC, 则 t_{SST}=1~2 时钟周期

休眠或空闲时看门狗溢出复位时序图

复位初始状态

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

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

注：“u”代表不改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

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

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

寄存器	上电复位	LVR 复位	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
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	xx0u uuuu	xx1u uuuu	uu11 uuuu
MP2L	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- 0x-0	---- uu-u	---- uu-u	---- uu-u
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu

寄存器	上电复位	LVR 复位	WDT 溢出 (正常模式)	WDT 溢出 (HALT)
PBPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	---- 1111	---- 1111	---- 1111	---- uuuu
PCC	---- 1111	---- 1111	---- 1111	---- uuuu
PCPU	---- 0000	---- 0000	---- 0000	---- uuuu
TBC0	0--- -000	0--- -000	0--- -000	u--- -uuu
TBC1	0--- -000	0--- -000	0--- -000	u--- -uuu
PSCR	---- --00	---- --00	---- --00	---- --uu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
LVDC	--00 0000	--00 0000	--00 0000	--uu uuuu
RSTC	0101 0101	0101 0101	0101 0101	uuuu uuuu
ADRL(ADRFS=0)	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRL(ADRFS=1)	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH(ADRFS=0)	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH(ADRFS=1)	---- xxxx	---- xxxx	---- xxxx	---- uuuu
ADCR0	0100 0000	0100 0000	0100 0000	uuuu uuuu
ADCR1	0000 0000	0000 0000	0000 0000	uuuu uuuu
SCC	000- 0-00	000- 0-00	000- 0-00	uuu- u-uu
HXTC	---- --00	---- --00	---- --00	---- --00
HIRCC	---- --01	---- --01	---- --01	---- --01
INTEG	---- 0000	---- 0000	---- 0000	---- 0000
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI0	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI1	--00 --00	--00 --00	--00 --00	--uu --uu
MFI2	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	00-0 0000	00-0 0000	00-0 0000	uu-u uuuu
PCS0	---- 0000	---- 0000	---- 0000	---- uuuu
EEA	--00 0000	--00 0000	--00 0000	--uu uuuu
EED	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEC	---- 0000	---- 0000	---- 0000	---- uuuu
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
TM1C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	LVR 复位	WDT 溢出 (正常模式)	WDT 溢出 (HALT)
TM1DH	---- --00	---- --00	---- --00	---- --uu
TM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1AH	---- --00	---- --00	---- --00	---- --uu
TM2C0	0000 0---	0000 0---	0000 0---	uuuu u---
TM2C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2DH	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2AH	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2RP	0000 0000	0000 0000	0000 0000	uuuu uuuu
USR	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SPWMC0	1000 -100	1000 -100	1000 -100	uuuu -uuu
SPWMC1	111- 0000	111- 0000	111- 0000	uuu- uuuu
SPWML	0000 0000	0000 0000	0000 0000	uuuu uuuu
SPWMH	---- 0000	---- 0000	---- 0000	---- uuuu
SFIFOL	0000 0000	0000 0000	0000 0000	uuuu uuuu
SFIFOH	0--- 0000	0--- 0000	0--- 0000	u--- uuuu
MTF	0000 0000	0000 0000	0000 0000	uuuu uuuu
SPWMDT	---- 0000	---- 0000	---- 0000	---- uuuu
ACDC	0000 0--0	0000 0--0	0000 0--0	uuuu u--u
ACDOFF	0000 0000	0000 0000	0000 0000	uuuu uuuu
OCP0C0	00-- ----	00-- ----	00-- ----	uu-- ----
OCP0C1	0-00 0000	0-00 0000	0-00 0000	0-00 0000
OCP0REF	0000 0000	0000 0000	0000 0000	uuuu uuuu
OCP0ACAL	0010 0000	0010 0000	0010 0000	uuuu uuuu
OCP0CCAL	0001 0000	0001 0000	0001 0000	uuuu uuuu
OCP1C0	00-- ----	00-- ----	00-- ----	uu-- ----
OCP1C1	0-00 0000	0-00 0000	0-00 0000	0-00 0000
OCP1REF	0000 0000	0000 0000	0000 0000	uuuu uuuu
OCP1ACAL	0010 0000	0010 0000	0010 0000	uuuu uuuu
OCP1CCAL	0001 0000	0001 0000	0001 0000	uuuu uuuu
OVPC	0--0 -000	0--0 -000	0--0 -000	u--u -uuu
OVPREF	0000 0000	0000 0000	0000 0000	uuuu uuuu
OVPCCAL	0001 0000	0001 0000	0001 0000	uuuu uuuu

注： “u”表示不改变
“x”表示未知
“-”表示未定义

输入 / 输出端口

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

该单片机提供 PA~PC 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
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	—	—	—	—	PC3	PC2	PC1	PC0
PCC	—	—	—	—	PCC3	PCC2	PCC1	PCC0
PCPU	—	—	—	—	PCPU3	PCPU2	PCPU1	PCPU0

PAWUn：PA 唤醒功能控制

0：除能

1：使能

PAn/PBn/PCn：I/O 口数据位

0：数据 0

1：数据 1

PACn/PBCn/PCCn：I/O 口输入 / 输出类型选择

0：输出

1：输入

PAPUn/PBPUn/PCPUn：上拉功能控制

0：除能

1：使能

上拉电阻

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

需要注意的是，当作为数字输入或 NMOS 输出时，上拉电阻才会受 PAPU~PCPU 寄存器控制开启，其它状态均为关闭。

PA 口唤醒

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

需要注意的是，当引脚用作普通 I/O 口数字输入且单片机为 HALT 状态时，唤醒才会受 PAWU 寄存器控制开启，其它状态均为关闭。

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

引脚共用功能

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

还需要注意的一点是，确保所需的引脚共用功能被正确地选择和取消。要选择所需的引脚共用功能，首先应通过相应的引脚共用控制寄存器正确地选择该功能，然后再配置相应的外围功能设置以使能外围功能。要正确地取消引脚共用功能，首先应除能外围功能，然后再修改相应的引脚共用控制寄存器以选择其它的共用功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	—	PBS04	PBS03	PBS02	PBS01	PBS00
PCS0	—	—	—	—	PCS03	PCS02	PCS01	PCS00
IFS0	—	—	—	—	—	—	—	IFS00

引脚共用功能控制寄存器列表

PAS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PAS07~PAS06**：PA3 功能选择
00：PA3
01：PA3
10：OCP0
11：AN3

Bit 5~4 **PAS05~PAS04**：PA2 功能选择
00：PA2
01：PA2
10：PA2
11：AN2

Bit 3~2 **PAS03~PAS02**：PA1 功能选择
00：PA1
01：PA1
10：VREF
11：AN1

Bit 1~0 **PAS01~PAS00**：PA0 功能选择
00：PA0
01：PA0
10：PA0
11：AN0

PAS1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAS17	PAS16	PAS15	PA1S4	PAS13	PAS12	PAS11	PAS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PAS17~PAS16**：PA7 功能选择
00：PA7
01：TX
10：PA7
11：AN7

Bit 5~4 **PAS15~PAS14**：PA6 功能选择
00：PA6
01：RX
10：PA6
11：AN6

Bit 3~2 **PAS13~PAS12**：PA5 功能选择
00：PA5
01：PA5
10：OVP
11：AN5

Bit 1~0 **PAS11~PAS10**：PA4 功能选择
00：PA4
01：PA4
10：OCP1
11：AN4

PBS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	—	PBS04	PBS03	PBS02	PBS01	PBS00
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 **PBS07**：PB7 功能选择
0：PB7
1：AN9

Bit 6 **PBS06**：PB6 功能选择
0：PB6
1：AN8

Bit 5 未定义，读为“0”

Bit 4 **PBS04**：PB4 功能选择
0：PB4/TP2（此时的 TP2 引脚为 TM2 捕捉输入脚）
1：TP2（此时的 TP2 引脚为 TM2 功能输出脚）

Bit 3 **PBS03**：PB3 功能选择
0：PB3
1：TP1

Bit 2 **PBS02**：PB2 功能选择
0：PB2
1：TP0

Bit 1 **PBS01**：PB1 功能选择
0：PB1
1：BBO

Bit 0 **PBS00**：PB0 功能选择
0：PB0
1：BUO

PCS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PCS03	PCS02	PCS01	PCS00
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 **PCS03**：PC3 功能选择
0：PC3
1：AC+

Bit 2 **PCS02**：PC2 功能选择
0：PC2
1：AC-

Bit 1 **PCS01**：PC1 功能选择
0：PC1
1：OSC1

Bit 0 **PCS00**：PC0 功能选择
0：PC0
1：OSC2

IFS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	IFS00
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1　未定义，读为“0”

Bit 0　**IFS00**：TM2 捕捉输入源选择
0：TP2 引脚
1：ACDO

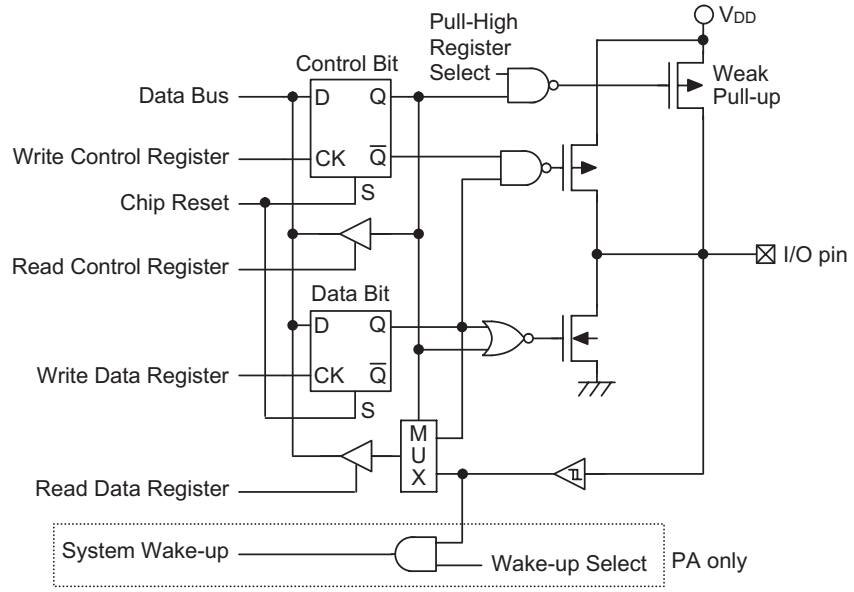
RSTC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	RST7	RST6	RST5	RST4	RST3	RST2	RST1	RST0
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

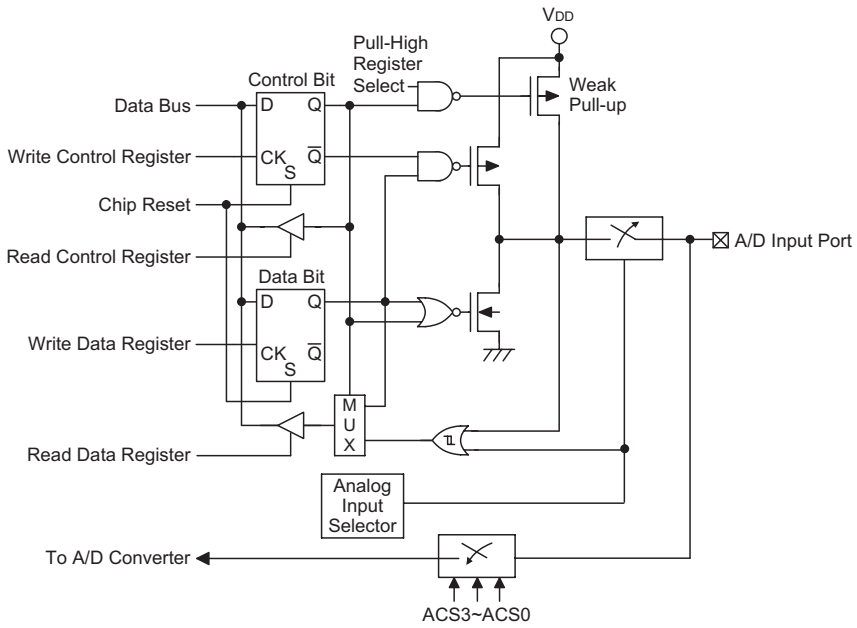
Bit 7~0　**RST7~RST0**：PA2 引脚配置
01010101B/10101010B：作为通用输入 / 输出引脚功能 PA2
其它值：MCU 复位
若此寄存器的值被修改为除“01010101”和“10101010”外的其它值，单片机将复位。复位动作发生在 2 ～ 3 个 LIRC 时钟周期后，且 RSTFC 寄存器的 ERSTF 位将置为“1”。

输入 / 输出引脚结构

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



通用输入 / 输出端口



A/D 输入 / 输出结构

编程注意事项

在编程中，最先要考虑的是端口的初始化。复位之后，所有的输入 / 输出数据及端口控制寄存器都将被设为逻辑高。所有输入 / 输出引脚默认为输入状态，而其电平则取决于其它相连接电路以及是否选择了上拉电阻。如果端口控制寄存器 PAC~PCC，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~PC 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标准型定时器章节。

简介

该单片机包含 3 个 TM，分别命名为 TM0、TM1 和 TM2。根据 TM 实现功能的不同可将其划分为特定的类型。TM0 和 TM1 为 10-bit 简易型 TM，TM2 都为 16-bit 标准型 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	STM
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 CTM	16-bit STM

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。

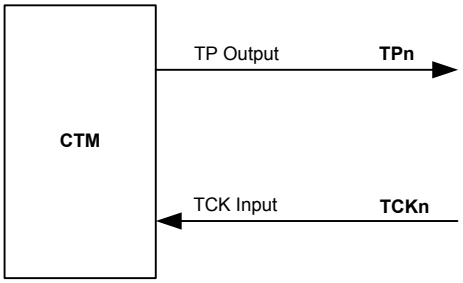
TM 外部引脚

无论哪种类型的 TM，都有一个 TM 输入引脚 TCKn。通过设置 TMnC0 寄存器中的 TnCK2~TnCK0 位，选择 TM 功能并将该引脚作为 TM 时钟源输入脚。外部时钟源可通过该引脚来驱动内部 TM。外部 TM 输入脚也与其它功能共用，但是，如果设置适当值给 TnCK2~TnCK0，该引脚会连接到内部 TM。TM 引脚可选择上升沿有效或下降沿有效。

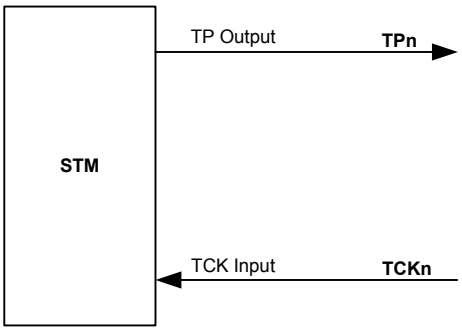
每个 TM 有一个输出引脚 TPn。当 TM 工作在比较匹配输出模式且比较匹配发生时，此引脚会由 TM 控制切换到高电平或低电平或翻转。外部 TPn 输出引脚也被 TM 用来产生 PWM 输出波形。当 TM 输出引脚与其它功能共用时，TM 输出功能需要通过寄存器先被设置。寄存器中的一个单独位用于决定其相关引脚用于外部 TM 输出还是用于其它功能。

TM 输入 / 输出引脚控制寄存器

通过设置与 TM 输入 / 输出引脚相关的引脚共用功能控制寄存器位，选择作为 TM 输入 / 输出功能或其它共用功能引脚。在使用 TM 功能前应正确的设置相关的控制位将相应的引脚用作 TM 输入 / 输出。更多引脚共用功能控制详见引脚共用功能章节。



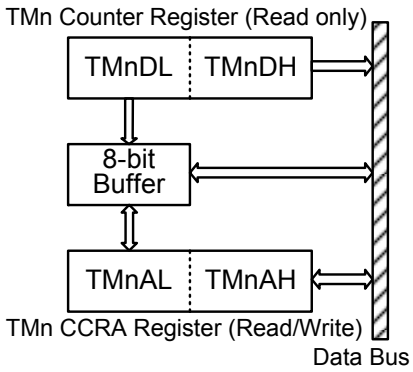
TMn 功能引脚控制框图（n=0 或 1）



TM2 功能引脚控制框图（n=2）

编程注意事项

TM 计数寄存器和捕捉 / 比较寄存器 CCRA 为 10-bit 或 16-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 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 低字节寄存器，TMnAL，否则可能导致无法预期的结果。



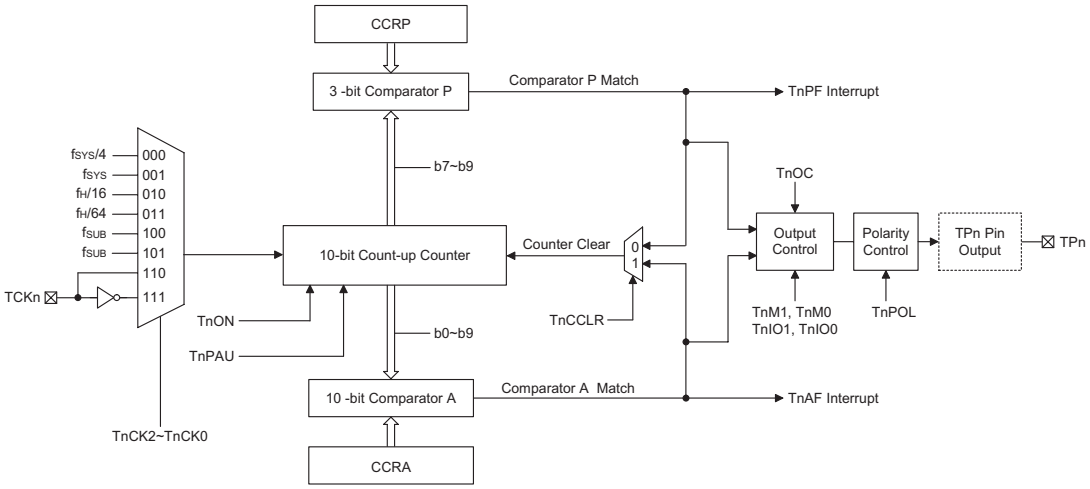
读写流程如下步骤所示：

- 写数据至 CCRA
  - ♦ 步骤 1. 写数据至低字节寄存器 TMnAL
    - – 注意，此时数据仅写入 8-bit 缓存器。
  - ♦ 步骤 2. 写数据至高字节寄存器 TMnAH
    - – 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 中读取数据
  - ♦ 步骤 1. 由高字节寄存器 TMnDH 或 TMnAH 读取数据
    - – 注意，此时高字节寄存器中的数据直接读取，同时由低字节寄存器读取的数据锁存至 8-bit 缓存器中。
  - ♦ 步骤 2. 由低字节寄存器 TMnDL 或 TMnAL 读取数据
    - – 注意，此时读取 8-bit 缓存器中的数据。

简易型 TM

虽然简易型 TM 是这几种 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 由一个外部输入脚控制并驱动一个外部输出脚。

名称	TM 编号	TM 输入引脚	TM 输出引脚
10-bit CTM	0，1	TCK0/TCK1	TP0/TP1



简易型 TM 方框图（n=0 或 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 寄存器介绍

简易型 TM 的所有操作由一组（六个）寄存器控制。每组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 TM 寄存器列表（n=0 或 1）

TMnC0 寄存器

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	TnPAU	TnCK2	TnCK1	TnCK0	TnON	TnRP2	TnRP1	TnRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TnPAU**：TMn 计数器暂停控制位

0：运行

1：暂停

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

Bit 6~4 **TnCK2~TnCK0**：选择 TMn 计数时钟位

000：$f_{SYS}/4$

001：f_{SYS}

010：$f_H/16$

011：$f_H/64$

100：f_{SUB}

101：f_{SUB}

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换时，内部计数器将保持其剩余值。若 TM 处于比较匹配输出模式时，当 TnON 位经由低到高的转换时，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 进行比较比较器 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，将选择比较器 P 比较匹配清除内部计数器。TnCCLR 位设为低，内部计数器在比较器 P 比较匹配发生时被重置；由于 CCRP 只与计数器高三位比较，比较结果是 128 时钟周期的倍数。CCRP 被清零时，实际上会使得计数器在最大值溢出。

TMnC1 寄存器

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

Bit 7~6 **TnM1~TnM0**：选择 TMn 工作模式位

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

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

Bit 5~4 **TnIO1~TnIO0**：选择 TPn 输出功能位

比较匹配输出模式
00：无变化
01：输出低
10：输出高
11：输出翻转

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

定时 / 计数器模式
未使用

此两位用于决定在一定条件达到时 TMn 输出脚如何改变状态。这两位值的设置取决于 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 TMn 输出脚如何改变状态。当比较器 A 比较匹配输出发生时 TMn 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。TMn 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，TMn 输出脚将不会发生变化。在 TMn 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 TMn 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。仅在 TMn 关闭时改变 TnIO1 和 TnIO0 位的值是很有必要的。若在 TM 运行时改变 TnIO1 和 TnIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **TnOC**：TPn 输出控制位

比较匹配输出模式

0：初始低

1：初始高

PWM 模式

0：低有效

1：高有效

这是 TMn 输出脚输出控制位。它取决于 TMn 此时正运行于比较匹配输出模式还是 PWM 模式。若 TMn 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 TMn 输出脚的逻辑电平值。在 PWM 模式时，其决定 PWM 信号是高有效还是低有效。

Bit 2 **TnPOL**：TPn 输出极性控制位

0：同相

1：反相

此位控制 TPn 输出脚的极性。此位为高时 TMn 输出脚反相，为低时 TMn 输出脚同相。若 TMn 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 TMn 包括两个比较器 - 比较器 A 和比较器 P。这两个比较器每个都可以用作清除内部计数器。TnCCLR 位设为高，计数器在比较器 A 比较匹配发生时被清除；此位设为低，计数器在比较器 P 比较匹配发生或计数器溢出时被清除。计数器溢出清除的方法仅在 CCRP 被清除为 0 时才能生效。TnCCLR 位在 PWM 模式时未使用。

TMnDL 寄存器

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

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

TMn 10-bit 计数器 bit 7~bit 0

TMnDH 寄存器

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

Bit 7~2 未定义，读为“0”

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

TMn 10-bit 计数器 bit 9~bit 8

TMnAL 寄存器

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

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

TMnAH 寄存器

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

Bit 7~2 未定义，读为“0”

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

简易型 TM 工作模式

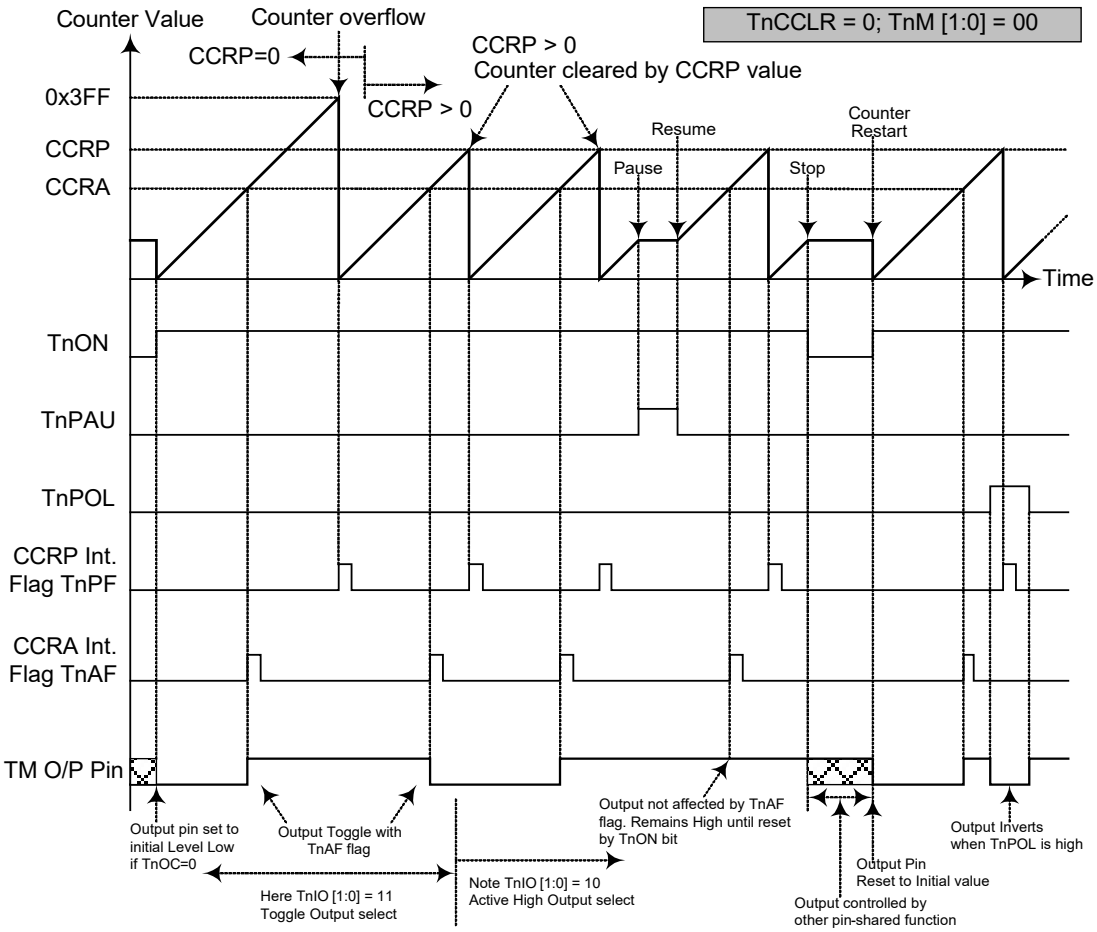
简易型 TM 有三种工作模式，即比较匹配输出模式，PWM 模式或定时 / 计数器模式。通过设置 TMnC1 寄存器的 TnM1 和 TnM0 位选择任意工作模式。

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。

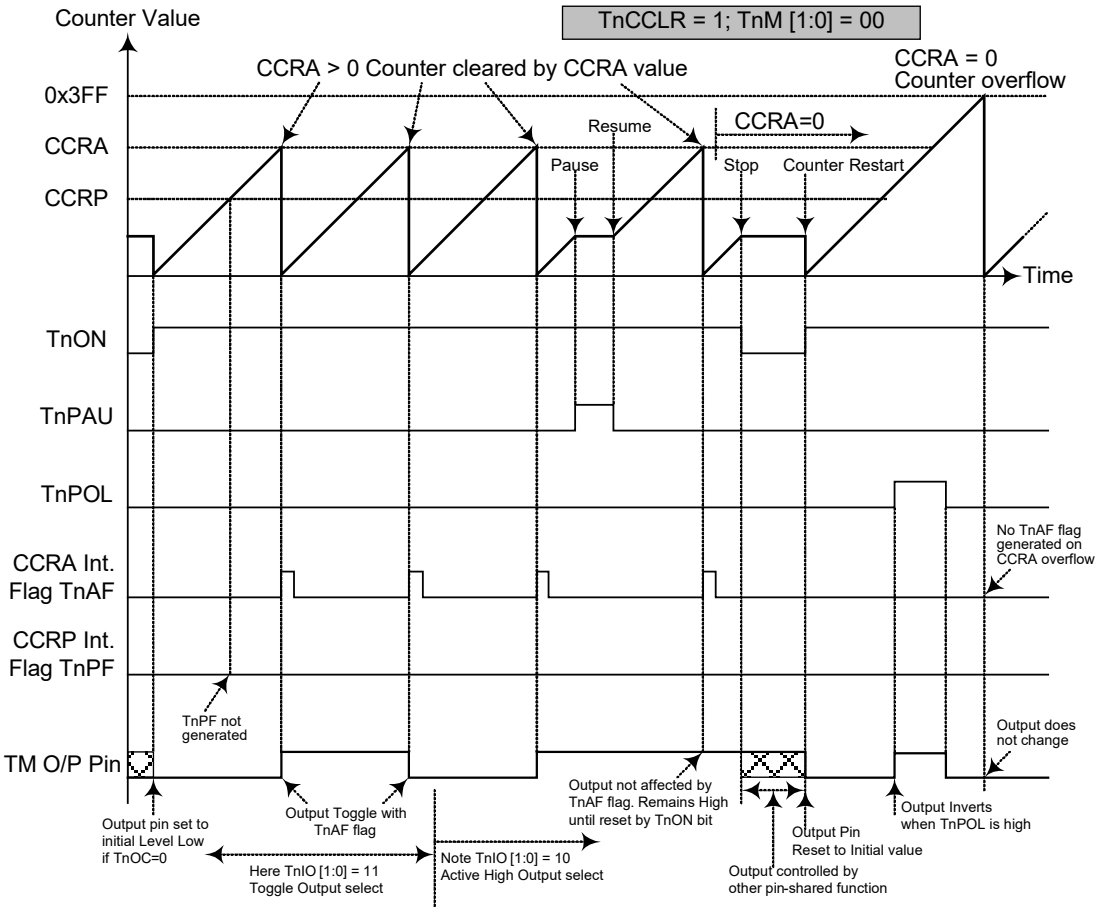
如果 TMnC1 寄存器的 TnCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 TnAF 中断请求标志产生。所以当 TnCCLR 为高时，不产生 TnPF 中断请求标志。如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 TnAF 请求标志。

正如该模式名所言，当比较匹配发生后，TMn 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，TMn 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 TMn 输出脚。TMn 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。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 或 1）

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



比较匹配输出模式 – TnCCLR=1（n=0 或 1）

注：1. TnCCLR=1，比较器 A 匹配将清除计数器
2. TMn 输出脚仅由 TnAF 标志位控制
3. 在 TnON 上升沿 TMn 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 TMn 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 TMn 工作在此模式，TMnC1 寄存器中的 TnM1 和 TnM0 位需要设置为“10”。TMn 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 TMn 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。

- **10-bit 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，

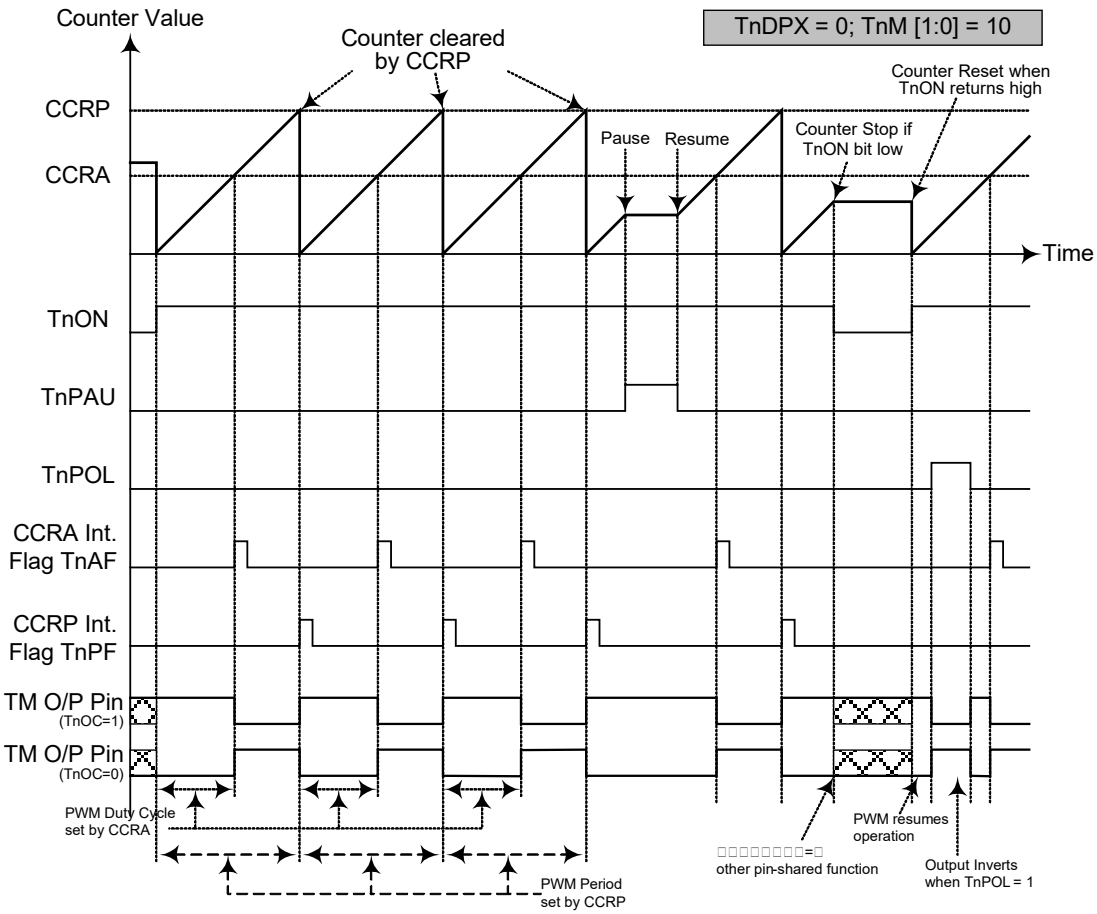
CTM PWM 输出频率 =(f_{SYS}/4)/512= f_{SYS}/2048=7.8125kHz，duty=128/512=25%

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%

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

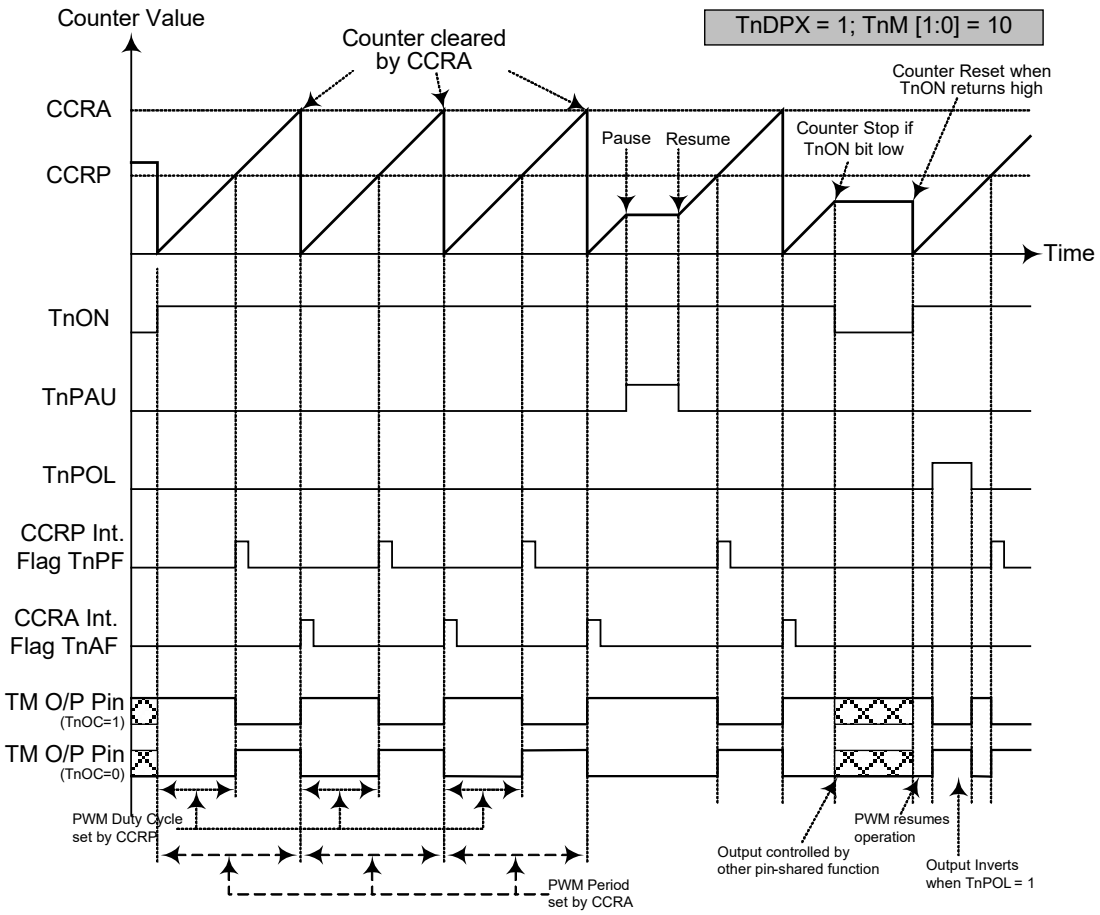
CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

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 或 1）

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



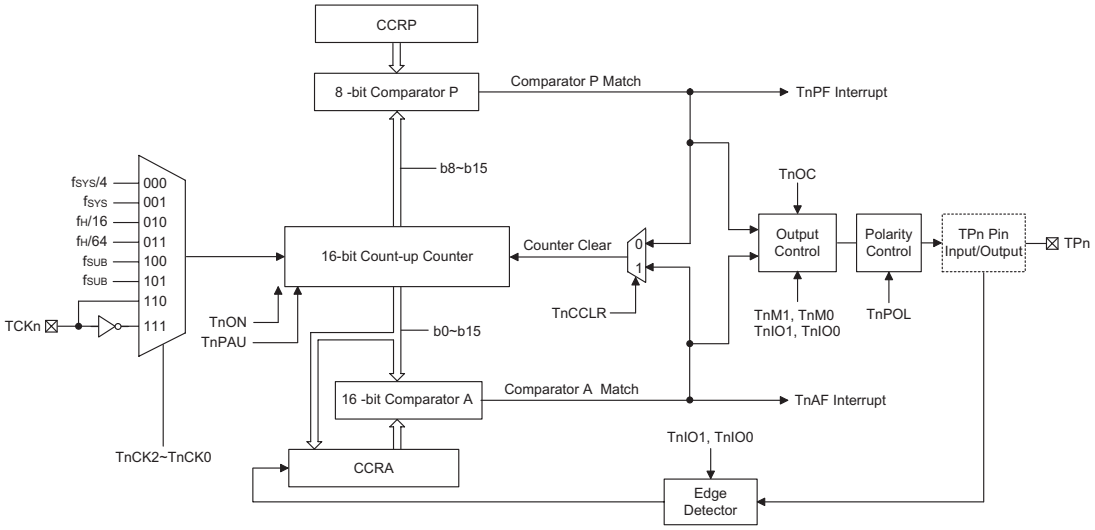
PWM 模式 – TnDPX=1（n=0 或 1）

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

标准型 TM – STM

标准型 TM 包括 5 种工作模式，即比较匹配输出、定时 / 事件计数器、捕捉输入、单脉冲输出和 PWM 输出模式。标准型 TM 由一个外部输入脚控制并驱动一个外部输出脚。

名称	TM 编号	TM 输入引脚	TM 输出引脚
16-bit STM	2	TCK2	TP2



标准型 TM 框图（n=2）

标准型 TM 操作

标准型 TM 的核心是一个由用户选择的内部或外部时钟源驱动的 16 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 8 位宽度，与计数器的高 8 位比较；而 CCRA 是 16 位的，与计数器的所有位比较。

通过应用程序改变 16 位计数器值的唯一方法是使 T2ON 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。标准型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，也可以控制输出脚。所有工作模式的设定都是通过设置相关寄存器来实现的。

标准型 TM 寄存器介绍

标准型 TM 的所有工作模式由一系列寄存器控制。一对只读寄存器用来存放 16 位计数器的值，一对读 / 写寄存器存放 16 位 CCRA 的值。剩下两个控制寄存器用来设置不同的操作和工作模式，以及 CCRP 的 8 位。

寄存器名称	位							
	7	6	5	4	3	2	1	0
TM2C0	T2PAU	T2CK2	T2CK1	T2CK0	T2ON	—	—	—
TM2C1	T2M1	T2M0	T2IO1	T2IO0	T2OC	T2POL	T2DPX	T2CCLR
TM2DL	D7	D6	D5	D4	D3	D2	D1	D0
TM2DH	D15	D14	D13	D12	D11	D10	D9	D8
TM2AL	D7	D6	D5	D4	D3	D2	D1	D0
TM2AH	D15	D14	D13	D12	D11	D10	D9	D8
TM2RP	D7	D6	D5	D4	D3	D2	D1	D0

16-bit 标准型 TM 寄存器列表

TM2C0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	T2PAU	T2CK2	T2CK1	T2CK0	T2ON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7　**T2PAU**：TM2 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，STM 保持上电状态并继续耗电。当此位由低到高转换时，计数器将保留其剩余值，直到此位再次改变为低电平，并从此值开始继续计数。

Bit 6 ~ 4　**T2CK2 ~ T2CK0**：选择 TM2 计数时钟位

000：$f_{SYS}/4$

001：f_{SYS}

010：$f_H/16$

011：$f_H/64$

100：f_{SUB}

101：f_{SUB}

110：TCK2 上升沿时钟

111：TCK2 下降沿时钟

此三位用于选择 TM2 的时钟源。外部引脚时钟源能被选择在上升沿或下降沿有效。f_{SYS} 是系统时钟，f_H 和 f_{SUB} 是其它的内部时钟源，细节方面请参考振荡器章节。

Bit 3　**T2ON**：TM2 计数器 On/Off 控制

0：Off

1：On

此位控制 TM2 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 TM2。清零此位将停止计数器并关闭 TM2 减少耗电。当此位经由低到高转换时，内部计数器将复位清零；当此位经由高到低转换时，内部计数器将保持其剩余值，直到此位再次改变为高电平。

若 TM2 处于比较匹配输出模式时，当 T2ON 位经由低到高的转换时，TM2 输出脚将复位至 T2OC 位指定的初始值。

Bit 2 ~ 0　未定义，读为“0”

TM2C1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	T2M1	T2M0	T2IO1	T2IO0	T2OC	T2POL	T2DPX	T2CCLR
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 **T2M1~T2M0**：选择 TM2 工作模式

00：比较匹配输出模式

01：捕捉输入模式

10：PWM 模式或单脉冲输出模式

11：定时 / 计数器模式

这两位设置 TM2 需要的工作模式。为了确保操作可靠，TM2 应在 T2M1 和 T2M0 位有任何改变前先关掉。在定时 / 计数器模式，TM2 输出脚控制必须除能。

Bit 5 ~ 4 **T2IO1~T2IO0**：选择 TM2 外部引脚功能

比较匹配输出模式

00：无变化

01：输出低

10：输出高

11：输出翻转

PWM 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 TM2 捕捉输入引脚上升沿输入捕捉

01：在 TM2 捕捉输入引脚下降沿输入捕捉

10：在 TM2 捕捉输入引脚双沿输入捕捉

11：输入捕捉除能

定时 / 计数器模式

未使用

此两位用于决定在一定条件达到时 TM2 外部引脚如何改变状态。这两位值的选择取决于 TM2 运行在哪种模式下。

在比较匹配输出模式下，T2IO1 和 T2IO0 位决定当从比较器 A 比较匹配输出发生时 TM2 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 TM2 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。TM2 输出脚的初始值通过 TM2C1 寄存器的 T2OC 位设置取得。注意，由 T2IO1 和 T2IO0 位得到的输出电平必须与通过 T2OC 位设置的初始值不同，否则当比较匹配发生时，TM2 输出脚将不会发生变化。在 TM2 输出脚改变状态后，通过 T2ON 位由低到高电平的转换复位至初始值。

在 PWM 模式，T2IO1 和 T2IO0 用于决定比较匹配条件发生时怎样改变 TM2 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 TM2 关闭时改变 T2IO1 和 T2IO0 位的值是很有必要的。若在 TM2 运行时改变 T2IO1 和 T2IO0 的值，PWM 输出的值是无法预料的。

Bit 3 **T2OC**：TM2 输出脚 TP2 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 TM2 输出脚输出控制位。它取决于 TM2 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 TM2 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 TM2 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 **T2POL**：TP2 输出极性控制位
0：同相
1：反相
此位控制 TP2 输出脚的极性。此位为高时 TM2 输出脚反相，为低时 TM2 输出脚同相。若 TM2 处于定时 / 计数器模式时其不受影响。

Bit 1 **T2DPX**：TM2 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 **T2CCLR**：选择 TM2 计数器清零条件位
0：TM2 比较器 P 匹配
1：TM2 比较器 A 匹配
此位用于选择清除计数器的方法。标准型 TM 包括两个比较器 - 比较器 A 和比较器 P。这两个比较器每个都可以用作清除内部计数器。T2CCLR 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。T2CCLR 位在 PWM，单脉冲或输入捕捉模式时未使用。

TM2DL 寄存器

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

Bit 7 ~ 0 **D7~D0**：TM2 计数器低字节寄存器 bit 7 ~ bit 0
TM2 16-bit 计数器 bit 7 ~ bit 0

TM2DH 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	0	0	0	0	0	0	0	0

Bit 7 ~ 0 **D15~D8**：TM2 计数器高字节寄存器 bit7 ~ bit 0
TM2 16-bit 计数器 bit 15 ~ bit 8

TM2AL 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**：TM2 CCRA 低字节寄存器 bit 7 ~ bit 0
TM2 16-bit CCRA bit 7 ~ bit 0

TM2AH 寄存器

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

Bit 7 ~ 0 **D15~D8**：TM2 CCRA 高字节寄存器 bit 7 ~ bit 0
TM2 16-bit CCRA bit 15 ~ bit 8

TM2RP 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**： TM2 CCRP 寄存器 bit 7~bit 0
TM2 CCRP 8 位寄存器，与 TM2 计数器 bit 15~bit 8 比较。
比较器 P 匹配周期
0：65536 个 TM2 时钟周期
1~255：256×(1~255) 个 TM2 时钟周期
此八位设定内部 CCRP 8-bit 寄存器的值，然后与内部计数器的高八位进行比较。如果 T2CCLR 位设为 0 时，比较结果为 0 并清除内部计数器。T2CCLR 位设为低，CCRP 比较匹配结果将重置内部计数器。由于 CCRP 只与计数器高八位比较，比较结果是 256 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。

标准型 TM 工作模式

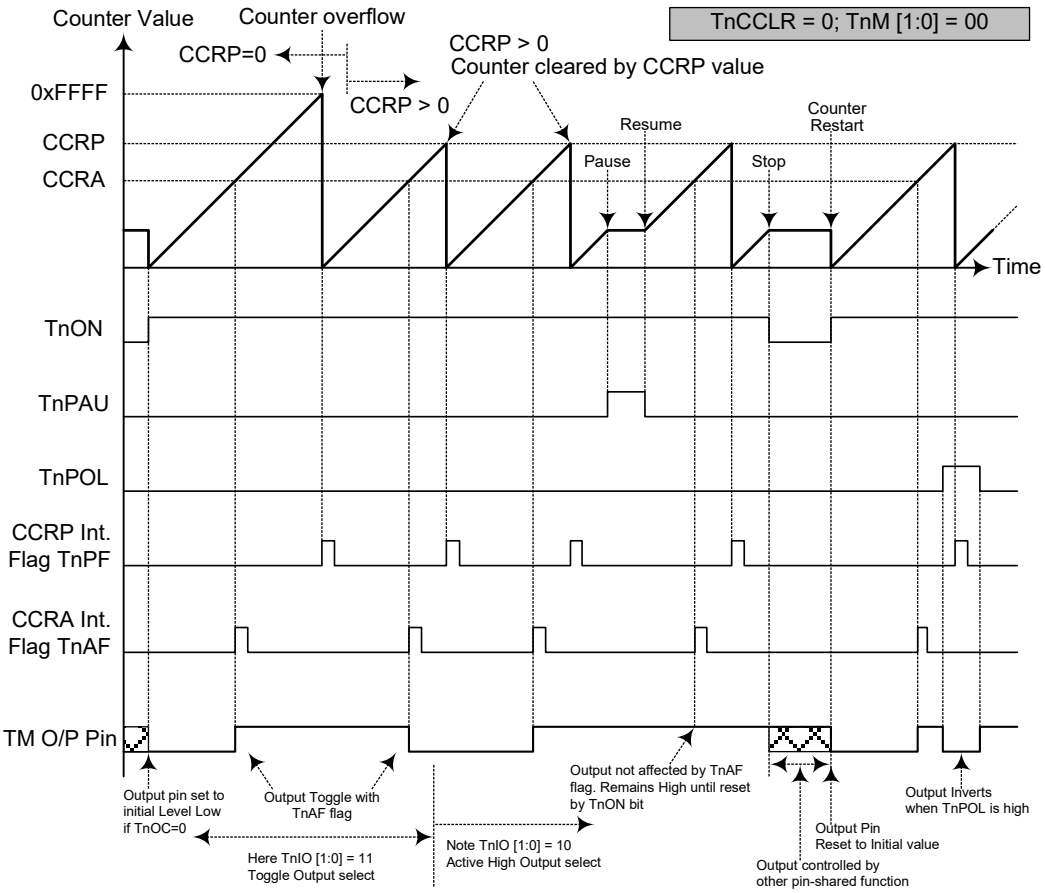
标准型 TM 有五种工作模式，即比较匹配输出模式、PWM 输出模式、单脉冲输出模式、捕捉输入模式或定时 / 计数器模式。通过设置 TM2C1 寄存器的 T2M1 和 T2M0 位选择任意模式。

比较匹配输出模式

为使 TM2 工作在此模式，TM2C1 寄存器中的 T2M1 和 T2M0 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 T2CCLR 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 T2AF 和 T2PF 将分别置位。

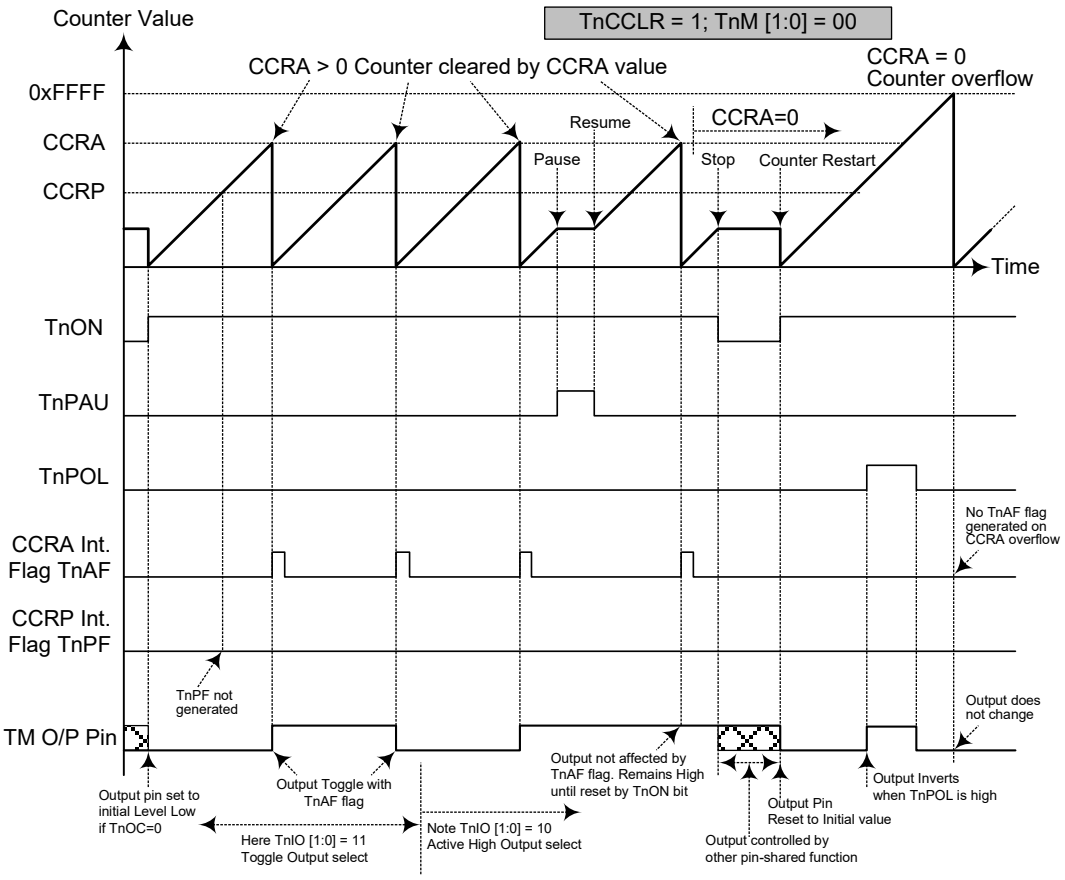
如果 TM2C1 寄存器的 T2CCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅产生 T2AF 中断请求标志。所以当 T2CCLR 为高时，不会产生 T2PF 中断请求标志。在比较匹配输出模式下，CCRA 不能设为“0”。

正如该模式名所言，当比较匹配发生后，TM2 输出脚状态改变。当比较器 A 比较匹配发生后 T2AF 标志产生时，TM2 输出脚状态改变。比较器 P 比较匹配发生时产生的 T2PF 标志不影响 TM2 输出脚。TM2 输出脚状态改变方式由 TM2C1 寄存器中 T2IO1 和 T2IO0 位决定。当比较器 A 比较匹配发生时，T2IO1 和 T2IO0 位决定 TM2 输出脚输出高，低或翻转当前状态。TM2 输出脚初始值，在 T2ON 位由低到高电平的变化后通过 T2OC 位设置。注意，若 T2IO1 和 T2IO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 – TnCCLR=0(n=2)

注： 1. T2CCLR=0，比较器 P 匹配将清除计数器
2. TM2 输出脚仅由 T2AF 标志位控制
3. 在 T2ON 上升沿 TM 输出脚复位至初始值



比较匹配输出模式 – TnCCLR=1(n=2)

注：1. T2CCLR=1，比较器 A 匹配将清除计数器
2. TM2 输出脚仅由 T2AF 标志位控制
3. 在 T2ON 上升沿 TM2 输出脚复位至初始值
4. 当 T2CCLR=1 时，不会产生 T2PF 标志

定时 / 计数器模式

为使 TM2 工作在此模式，TM2C1 寄存器中的 T2M1 和 T2M0 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 TM2 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 TM2 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 TM2 工作在此模式，TM2C1 寄存器中的 T2M1 和 T2M0 位需要设置为“10”，且 T2IO1 和 T2IO0 位也需要设置为“10”。TM2 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 TM2 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，T2CCLR 位不影响 PWM 周期。CCRA 和 CCRP 寄存器决定 PWM 波形，一个用来清除内部计数器并控制 PWM 波形的频率，另一个用来控制占空比。所以 PWM 波形由 CCRA 和 CCRP 寄存器共同决定。

当比较器 A 或比较器 P 比较匹配发生时，将产生 CCRA 或 CCRP 中断标志。TM2C1 寄存器中的 T2OC 位决定 PWM 波形的极性，T2IO1 和 T2IO0 位使能 PWM 输出或将 TM2 输出脚置为逻辑高或逻辑低。T2POL 位对 PWM 输出波形的极性取反。

- **16-bit STM, PWM 模式 , 边沿对齐模式，T2DPX=0**

CCRP	周期	占空比
1~255	CCRP×256	CCRA
0	65536	

若 f_{SYS}=16MHz，TM2 时钟源选择 f_{SYS}/4，CCRP=010b，CCRA=128，

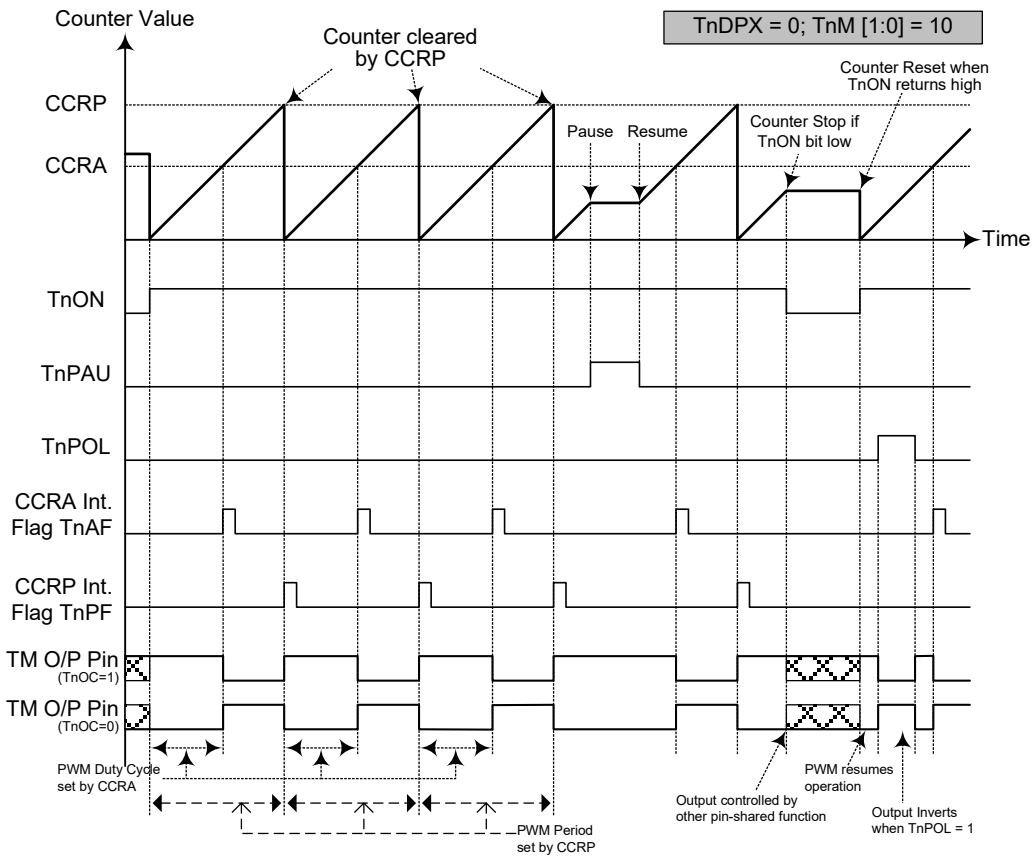
TM2 PWM 输出频率 =(f_{SYS}/4)/512= f_{SYS}/2048=7.8125kHz，duty=128/512=25%

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%

- **16-bit STM, PWM 模式 , 边沿对齐模式，T2DPX=1**

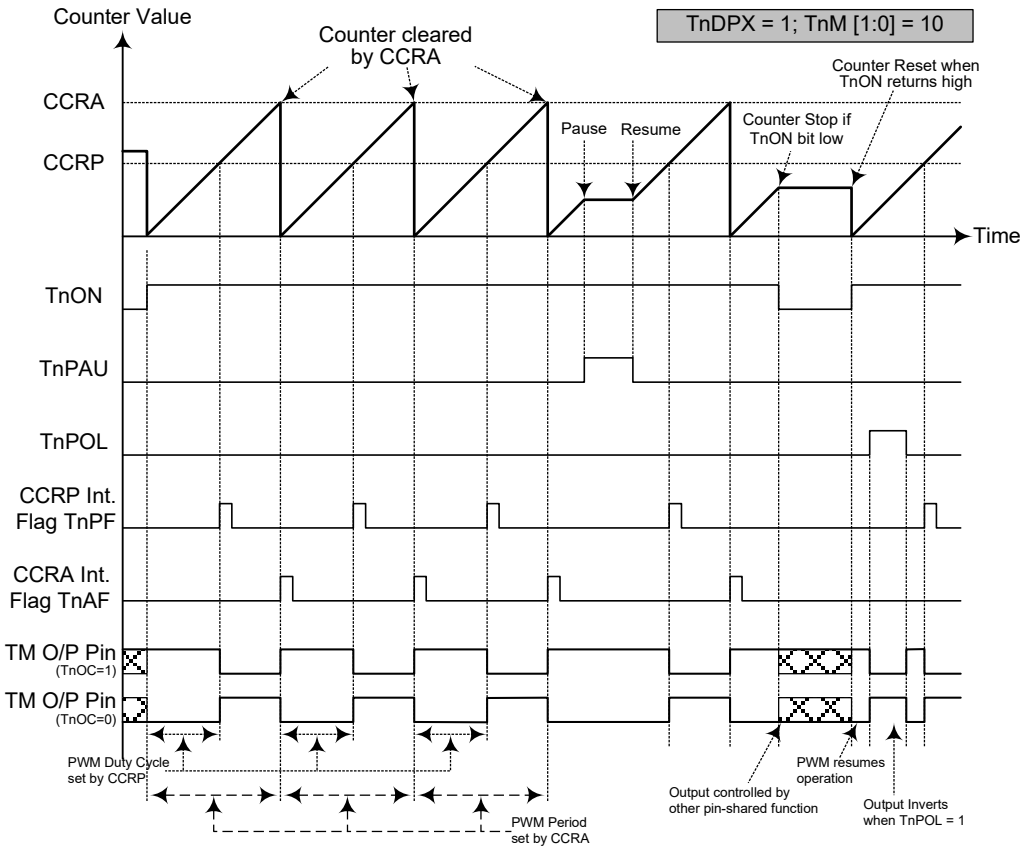
CCRP	周期	占空比
1~255	CCRA	CCRP×256
0		65536

PWM 的输出周期由 CCRA 寄存器的值与 TM2 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=2)

注： 1. T2DPX=0, CCRP 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 T2IO[1:0]=00 或 01，PWM 功能不变
4. T2CCLR 位不影响 PWM 操作



PWM 模式 – TnDPX=1(n=2)

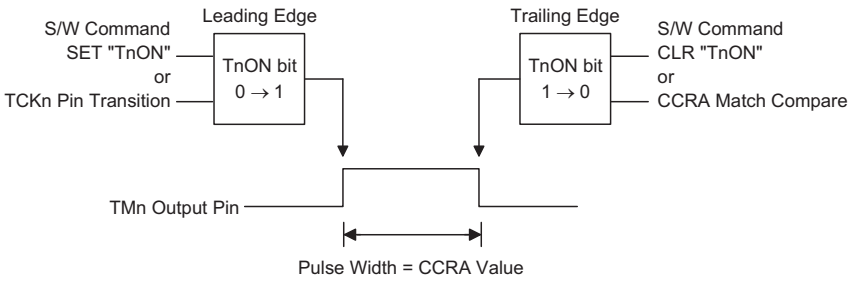
注： 1. T2DPX=1, CCRA 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 T2IO[1:0]=00 或 01，PWM 功能不变
4. T2CCLR 位不影响 PWM 操作

单脉冲模式

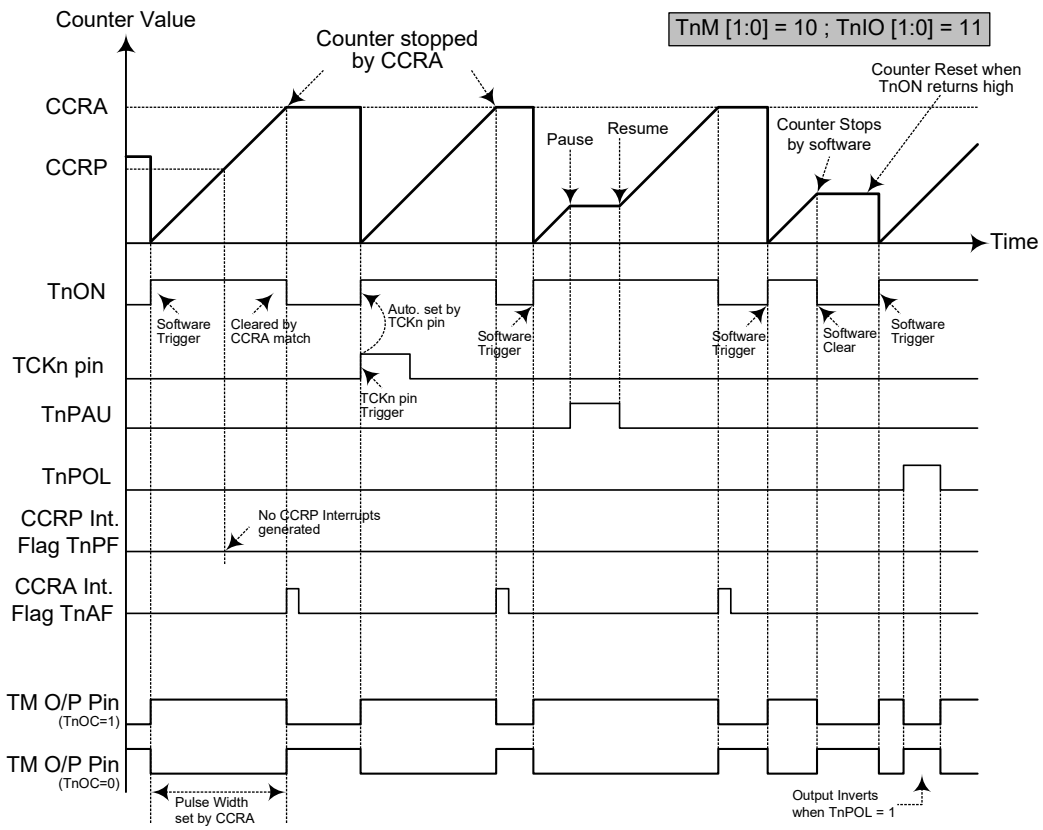
为使 TM2 工作在此模式，TM2C1 寄存器中的 T2M1 和 T2M0 位需要设置为“10”，同时 T2IO1 和 T2IO0 位需要设置为“11”。正如模式名所言，单脉冲输出模式，在 TM2 输出脚将产生一个脉冲输出。

脉冲输出可以通过应用程序控制 T2ON 位由低到高的转变来触发。而处于单脉冲模式时，T2ON 位可在 TCKn 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 T2ON 位转变为高电平时，计数器将开始运行，并产生脉冲前沿。当脉冲有效时 T2ON 位保持高电平。通过应用程序使 T2ON 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，会自动清除 T2ON 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 TM2 中断。T2ON 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 T2CCLR 位未使用。



单脉冲产生示意图 (n=2)



单脉冲模式 (n=2)

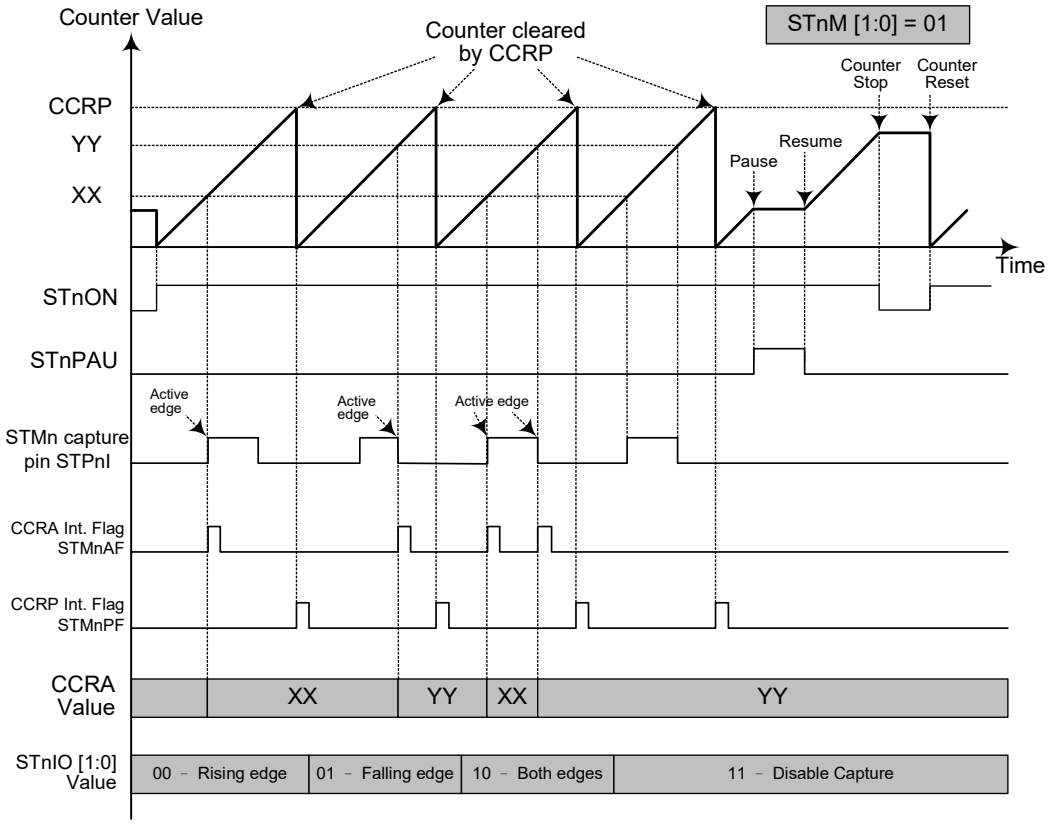
注： 1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 TCK2 脚或设置 T2ON 位为高来触发脉冲
4. 单脉冲模式中，T2IO[1:0] 需置位“11”，且不能更改。

捕捉输入模式

为使 TM2 工作在此模式，TM2C1 寄存器中的 T2M1 和 T2M0 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。TP2 脚上的外部信号，通过设置 TM2C1 寄存器的 T2IO1 和 T2IO0 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 T2ON 位由低到高转变时，计数器启动。

当 TP2 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 TM2 中断。无论 TP2 引脚发生哪种边沿转换，计数器将继续工作直到 T2ON 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 TM2 中断。记录 CCRP 溢出中断信号的值可以测量脉宽。通过设置 T2IO1 和 T2IO0 位选择 TP2 引脚为上升沿、下降沿或双沿有效。如果 T2IO1 和 T2IO0 位都设置为高，无论 TP2 引脚发生哪种边沿转换都不会产生捕捉操作，但计数器将会继续运行。

T2CCLR 和 T2DPX 位在此模式中未使用。



捕捉输入模式 (n=2)

注：
1. T2M[1:0]=01 并通过 T2IO1 和 T2IO0 位设置有效边沿
2. TM2 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中
3. T2CCLR 位未使用
4. 无输出功能 -- T2OC 和 T2POL 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

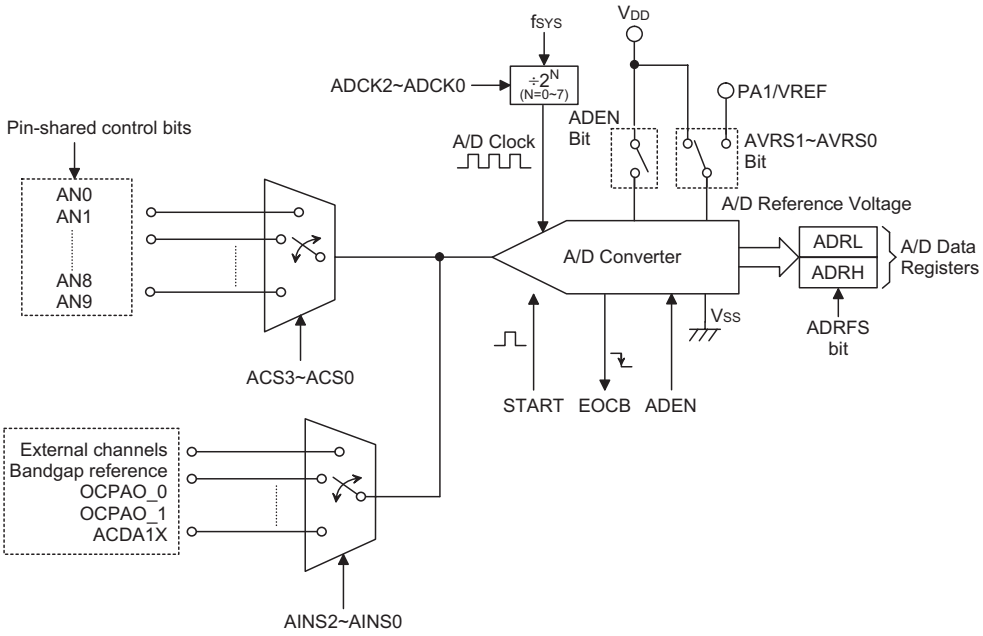
A/D 转换器

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

A/D 简介

该单片机都包含一个多通道的 A/D 转换器，它们可以直接接入外部模拟信号（来自传感器或其它控制信号）并直接将这些信号转换成 12 位的数字量。选择转换外部或内部模拟信号由 AINS2~AINS0 和 ACS3~ACS0 位控制。若选择内部模拟信号，除了设置 AINS2~AINS0 和 ACS3~ACS0 位外还需合理设置相应的引脚共用功能控制位。关于 A/D 输入信号的详细描述请参考“A/D 转换寄存器介绍”和“A/D 输入信号”两节内容。

下图显示了 A/D 转换器内部结构和相关的寄存器。



A/D 转换器结构

A/D 转换寄存器介绍

A/D 转换器的所有工作主要由四个寄存器控制。一对只读寄存器来存放 12 位 A/D 转换数据的值。剩下两个控制寄存器设置 A/D 转换器的操作和控制功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
ADRL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
ADRL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADRH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADRH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
ADCR0	START	EOCB	ADEN	ADRFS	ACS3	ACS2	ACS1	ACS0
ADCR1	AINS2	AINS1	AINS0	AVRS1	AVRS0	ADCK2	ADCK1	ADCK0

A/D 转换寄存器列表

A/D 转换器数据寄存器 – ADRL, ADRH

对于具有 12 位 A/D 转换器的单片机，需要两个数据寄存器存放转换结果，一个高字节寄存器 ADRH 和一个低字节寄存器 ADRL。在 A/D 转换完毕后，单片机可以直接读取这些寄存器以获得转换结果。由于寄存器只使用了 16 位中的 12 位，其数据存储格式由 ADCR0 寄存器的 ADRFS 位控制，如下表所示。D0~D11 是 A/D 换转数据结果位。未使用的位读为“0”。

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

A/D 转换数据寄存器对

A/D 转换控制寄存器 – ADCR0, ADCR1, PAS0, PAS1, PBS0

寄存器 ADCR0 和 ADCR1 用来控制 A/D 转换器的功能和操作。这些 8 位的寄存器定义包括选择连接至内部 A/D 转换器的模拟通道，数字化数据格式，A/D 时钟源，并控制和监视 A/D 转换器的开始和转换结束状态。寄存器 ADCR0 的 ACS3~ACS0 位定义 A/D 转换器外部输入通道编号。寄存器 ADCR1 的 AINS2~AINS0 位的功能决定选择外部模拟输入通道或某个内部信号被连接到内部 A/D 转换器。 若 AINS2~AINS0 为 000 或 101~111，则选择转换外部模拟输入信号；若 AINS2~AINS0 为除 000 或 101~111 以外的其它值，则选择转换对应的内部信号。需要注意的是，当选择内部模拟信号时，应将 ACS3~ACS0 设置为 1010~1111 中的任意值使外部信号输入脚浮空。否则，已选择的外部输入通道会和内部模拟信号一起连接至内部 A/D 转换器，这将导致无法预期的损害。

引脚共用功能控制寄存器 PAS0, PAS1 和 PBS0 包含对应的引脚共用功能选择位，用来定义 PA 或 PB 口中的哪些引脚为 A/D 转换器的模拟输入，哪些引脚不作为 A/D 转换输入。当引脚作为 A/D 输入时，其原来的 I/O 或其它引脚共用功能消失，此外，其内部上拉电阻也将自动断开。

AINS [2:0]	ACS [3:0]	输入信号	描述
000,101~111	0000~1001	AN0~AN9	外部通道输入
	1010~1111	—	外部通道浮空，无外部信号输入
001	1010~1111	V_{BG}	内部 V_{BG} 参考电压
010	1010~1111	OCPAO_0	过流保护 0 输入电压信号
011	1010~1111	OCPAO_1	过流保护 1 输入电压信号
100	1010~1111	ACDA1X	交流过零检测结果信号

A/D 转换器输入信号选择

- **ADCR0 寄存器**

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

Bit 7 **START**：启动 A/D 转换位
0→1→0 ：启动
0→1：重置 A/D 转换，并且设置 EOCB 为“1”
此位用于初始化 A/D 转换过程。通常此位为低，但如果设为高再被清零，将初始化 A/D 转换过程。当此位为高，将重置 A/D 转换器。

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

Bit 5 **ADEN**：ADC 模块电源开 / 关控制位
0：ADC 模块电源关
1：ADC 模块电源开
此位控制 A/D 内部功能的电源。该位设为高将使能 A/D 转换器。如果该位被清零将关闭 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 ADEN=0 以减少功耗。
2. ADEN=0 将关闭 ADC 模块的电源。

Bit 4 **ADRFS**：ADC 数据格式控制位
0：ADC 数据高字节是 ADRH 的 bit 7~bit 0，低字节是 ADRL 的 bit 7~bit 4
1：ADC 数据高字节是 ADRH 的 bit 3~bit 0，低字节是 ADRL 的 bit 7~bit 0

Bit 3 ~ 0 **ACS3 ~ ACS0**：A/D 转换器外部模拟通道输入选择位
0000：AN0
0001：AN1
0010：AN2
0011：AN3
0100：AN4
0101：AN5
0110：AN6
0111：AN7
1000：AN8
1001：AN9
1010~1111：所有外部通道浮空

- **ADCR1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	AINS2	AINS1	AINS0	AVRS1	AVRS0	ADCK2	ADCK1	ADCK0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~5 **AINS2~AINS0**：ADC 输入信号选择位
000：来自外部模拟通道输入
001：来自 V_{BG} 参考电压
010：OCPAO_0
011：OCPAO_1
100：ACDA1X
101~111：来自外部模拟通道输入

Bit 4~3 **AVRS1~AVRS0**：选择 ADC 参考电压
00：VREF 引脚
01：内部 ADC 电源
1x：VREF 引脚

Bit 2 ~ 0 **ADCK2 ~ADCK0**：选择 ADC 时钟源
000：f_{SYS}
001：$f_{SYS}/2$
010：$f_{SYS}/4$
011：$f_{SYS}/8$
100：$f_{SYS}/16$
101：$f_{SYS}/32$
110：$f_{SYS}/64$
111：$f_{SYS}/128$
这三位用于选择 A/D 转换器的时钟源。

A/D 操作

ADCR0 寄存器中的 START 位，用于打开和复位 A/D 转换器。当单片机设定此位从逻辑低到逻辑高，然后再到逻辑低，就会开始一个模数转换周期。当 START 位从逻辑低到逻辑高，但不再回到逻辑低时，ADCR0 寄存器中的 EOCB 位为“1”，复位模数转换器。START 位用于控制内部模数转换器的整个开启动作。

ADCR0 寄存器中的 EOCB 位用于表明模数转换过程的完成。在转换周期结束后，EOCB 位会被单片机自动清零。此外，也会置位中断控制寄存器内相应的 A/D 中断请求标志位，如果中断使能，就会产生对应的内部中断信号。A/D 内部中断信号将引导程序到相应的 A/D 内部中断入口。如果 A/D 内部中断被禁止，可以让单片机轮询 ADCR0 寄存器中的 EOCB 位，检查此位是否被清除，以作为另一种侦测 A/D 转换周期结束的方法。

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

虽然 A/D 时钟源是由系统时钟 f_{SYS} 和 ADCK2~ADCK0 位决定，但可选择的最大 A/D 时钟源则有一些限制。由于允许的 A/D 时钟周期 t_{AD} 的范围为 0.5μs ~10μs，所以选择系统时钟速度时就必须小心。如果系统时钟速度为 4MHz 时，ADCK2~ADCK0 位不能设为“000”或“11x”。必须保证设定的 A/D 转换时钟周期不小于时钟周期的最小值，不大于时钟周期的最大值，否则将会产生不准确的 A/D 转换值。

使用者可以参考下面的表格，被标上星号 * 的数值是不允许的，因超出了规定的范围。

f_{SYS}	A/D 时钟周期 (t_{AD})							
	ADCK [2:0]= 000 (f_{SYS})	ADCK [2:0]= 001 (f_{SYS}/2)	ADCK [2:0]= 010 (f_{SYS}/4)	ADCK [2:0]= 011 (f_{SYS}/8)	ADCK [2:0]= 100 (f_{SYS}/16)	ADCK [2:0]= 101 (f_{SYS}/32)	ADCK [2:0]= 110 (f_{SYS}/64)	ADCK [2:0]= 111 (f_{SYS}/128)
1MHz	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *	128μs *
2MHz	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *
4MHz	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *
8MHz	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *
12MHz	83ns *	167ns *	333ns *	667ns	1.33μs	2.67μs	5.33μs	10.67μs *
16MHz	62.5ns *	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs
20MHz	50ns*	100ns*	200ns*	400ns*	800ns	1.6μs	3.2μs	6.4μs

A/D 时钟周期范例

ADCR0 寄存器的 ADEN 位用于控制 A/D 转换电路电源的开 / 关。该位必须置高以开启 A/D 转换器电源。当设置 ADEN 位为高开启 A/D 转换器内部电路时，在 A/D 转换成功初始化前需一段延时。即使通过设置相关引脚共用控制位，选择无引脚作为 A/D 输入，如果 ADEN 设为“1”，那么仍然会产生功耗。因此当未使用 A/D 转换器功能时，在功耗敏感的应用中建议设置 ADEN 为低以减少功耗。

A/D 转换器参考电压来自 A/D 电源电压或外部参考源引脚 VREF，可通过 AVRS1~AVRS0 位来选择。若 AVRS1~AVRS0 位为“01”，参考电压来自 A/D 电源电压。若 AVRS1~AVRS0 位设为除“01”外的其它任意值，则参考电压来自 VREF 引脚。由于 VREF 引脚与其它功能共用，当选择 VREF 参考电压时，需合理设置相关引脚共用控制位选择 VREF 引脚功能且除能其它引脚功能。

AVRS [1:0]	参考电压	描述
00，10, 11	VREF	A/D 转换器参考电压来自 VREF 引脚
01	内部 ADC 电源	A/D 转换器参考电压来自内部 ADC 电源

A/D 转换器参考电压选择

A/D 输入信号

所有的 A/D 模拟输入引脚都与 PA 和 PB 端口的 I/O 引脚及其它功能共用。使用 PAS0, PAS1 及 PBS0 寄存器中的相关控制位可以将它们设置为 A/D 转换器模拟输入脚或具有其它功能。如果引脚的对应控制位选择 A/D 输入功能，那么该引脚作为 A/D 转换输入且原引脚功能除能。通过这种方式，引脚的功能可由程序来控制，灵活地切换引脚功能。如果将引脚设为 A/D 输入，则通过寄存器编程设置的所有上拉电阻会自动断开。请注意，PAC 或 PBC 端口控制寄存器不需要为使能 A/D 输入而先设定为输入模式，当引脚共用控制位使能 A/D 输入时，端口控制寄存器的状态将被重置。

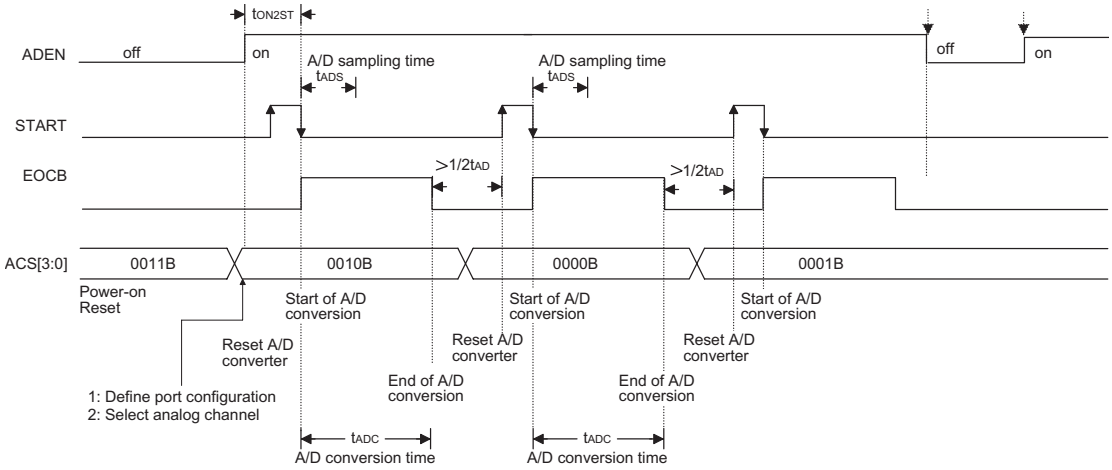
通过设置 ADCR1 寄存器的 AINS2~AINS0 位也可选择内部模拟信号作为连接到 A/D 转换器的模拟输入信号。当选择内部模拟信号时，应将 ACS3~ACS0 设置为 1010~1111 中的任意值以关断外部模拟通道输入。否则会发生内部模拟信号与外部通道输入信号同时接入 A/D 转换器的情况，从而导致不可预期的错误。

A/D 转换率及时序图

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

最大 A/D 转换率 = A/D 时钟周期 /16

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



A/D 转换时序图

A/D 转换步骤

下面概述实现 A/D 转换过程的各个步骤。

- 步骤 1
  通过 ADCR1 寄存器中的 ADCK2~ADCK0 位，选择所需的 A/D 转换时钟。
- 步骤 2
  将 ADCR0 寄存器中的 ADEN 位置高以使能 A/D 转换器。
- 步骤 3
  通过 ADCR1 寄存器中的 AINS2~AINS0 位，选择连接至内部 A/D 转换器的信号。
  若选择外部通道输入，接着执行步骤 4。
  若选择内部模拟信号，接着执行步骤 5。
- 步骤 4
  若已通过 AINS2~AINS0 位选择 A/D 输入信号来自外部通道，接着应设置 ACS3~ACS0 位选择哪个外部通道接至 A/D 转换器。通过设置相关的引脚共用控制位将该引脚规划为 A/D 输入引脚。接着执行步骤 6。
- 步骤 5
  选择内部模拟信号前，应将 ACS3~ACS0 设置为 1010~1111 中的任意值以断开外部通道。接着设置 AINS2~AINS0 位选择所需的内部模拟信号。接着执行步骤 6。
- 步骤 6
  通过 AVRS1~AVRS0 位选择参考电压。

- 步骤 7
  设置 ADRFS 位选择 A/D 转换器输出数据格式。
- 步骤 8
  如果要使用中断，则中断控制寄存器需要正确地设置，以确保 A/D 中断功能是激活的。总中断控制位 EMI 需要置位为“1”, 以及 A/D 转换器中断位 ADE 也需要置位为“1”。
- 步骤 9
  现在可以通过设定 ADCR0 寄存器中的 START 位从“0”到“1”再回到“0”，开始模数转换的过程。
- 步骤 10
  可以轮询 ADCR0 寄存器中的 EOCB 位，检查模数转换过程是否完成。当此位为逻辑高时，表示转换正在进行中。当此位为逻辑低时，表示转换过程已经完成。转换完成后，可读取 A/D 数据寄存器 ADRL 和 ADRH 获得转换后的值。另一种方法是，若中断使能且堆栈未满，则程序等待 A/D 中断发生。

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

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 ADEN 为低，关闭 A/D 内部电路以减少电源功耗。此时，不考虑输入脚的模拟电压，内部 A/D 转换器电路不产生功耗。如果 A/D 转换器输入脚用作普通 I/O 脚，必须特别注意，输入电压为无效逻辑电平也可能增加功耗。

A/D 转换功能

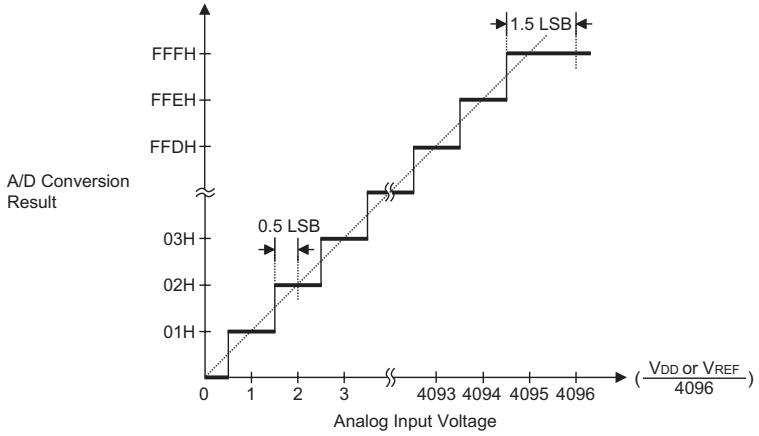
单片机含有一组 12 位的 A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于 V_{DD} 或 V_{REF} 的电压值，因此每一位可表示 V_{DD} 或 $V_{REF}/4096$ 的模拟输入值。

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

通过下面的等式可估算 A/D 转换器输入电压值：

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

下图显示 A/D 转换器模拟输入值和数字输出值之间理想的转换功能。除了数字化数值 0，其后的数字化数值会在精确点之前的 0.5 LSB 处改变，而数字化数值的最大值将在 V_{DD} 或 V_{REF} 之前的 1.5 LSB 处改变。



理想的 A/D 转换功能

A/D 转换应用范例

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

范例：使用查询 EOCB 的方式来检测转换结束

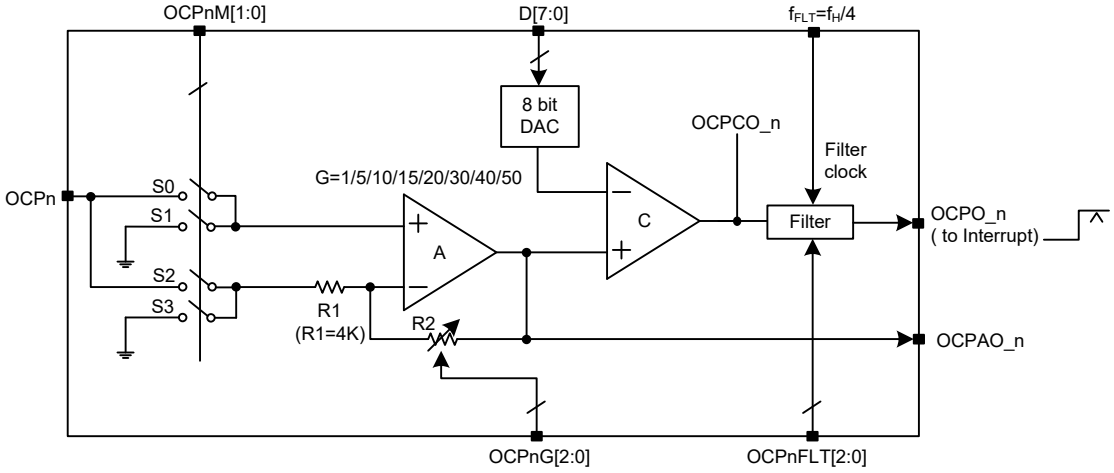
```
clr  ADE                    ; disable ADC interrupt
mov  a,0BH
mov  ADCR1,a                ; select fSYS/8 as A/D clock and select the external
                            ; input signals voltage
set  ADEN
mov  a,03h                  ; setup PAS0 to configure pin AN0
mov  PAS0,a
mov  a,20h
mov  ADCR0,a                ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr  START                  ; high pulse on start bit to initiate conversion
set  START                  ; reset A/D
clr  START                  ; start A/D
polling_EOC:
sz   EOCB                   ; poll the ADCR0 register EOCB bit to detect end
                            ; of A/D conversion
jmp  polling_EOC            ; continue polling
mov  a,ADRL                 ; read low byte conversion result value
mov  ADRL_buffer,a          ; save result to user defined register
mov  a,ADRH                 ; read high byte conversion result value
mov  ADRH_buffer,a          ; save result to user defined register
:
:
jmp  start_conversion       ; start next a/d conversion
```

范例：使用中断的方式来检测转换结束

```
clr  ADE                    ; disable ADC interrupt
mov  a,0BH
mov  ADCR1,a                ; select fSYS/8 as A/D clock and select the
                            ; external input signals voltage
set  ADEN
mov  a,03h                  ; setup PAS0 to configure pins AN0
mov  PAS0,a
mov  a,20h
mov  ADCR0,a                ; enable and connect AN0 channel to A/D converter
Start_conversion:
clr  START                  ; high pulse on START bit to initiate conversion
set  START                  ; reset A/D
clr  START                  ; start A/D
clr  ADF                    ; clear ADC interrupt request flag
set  ADE                    ; enable ADC interrupt
set  EMI                    ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov  acc_stack,a            ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a         ; save STATUS to user defined memory
:
:
mov  a,ADRL                 ; read low byte conversion result value
mov  ADRL_buffer,a          ; save result to user defined register
mov  a,ADRH                 ; read high byte conversion result value
mov  ADRH_buffer,a          ; save result to user defined register
:
:
EXIT_INT_ISR:
mov  a,status_stack
mov  STATUS,a               ; restore STATUS from user defined memory
mov  a,acc_stack            ; restore ACC from user defined memory
reti
```

过电流保护功能 – OCP

OCP 是过电流保护（Over Current Protect）的缩写。OCP 检测到的输入电压与要监测的源电流成正比，如果输入电压大于 DAC 设置的参考电压，OCP 将会产生一个输出信号表示源电流超过了规格。



过流保护电路（**n=0 或 1**）

过流保护电路操作

源电压从 OCPn 引脚输入，四个模拟开关 S0~S3 用于模式选择。一个运算放大器和两个电阻组成了一个 PGA 功能。选择 PGA 的电压从正极或是负极输入，相应的可以产生正的或负的增益。8-bit DAC 主要是用于产生一个参考电压，比较器可以将放大的输出电压 OCPAO_n 与此参考电压进行比较，比较结果 OCPCO_n 经滤波后产生比较器输出 OCPO_n 产生 OCP 中断。OCPO_n 是 OCPCO_n 去抖之后的输出信号，通过 OCPO_n 输出位判断源电流是否超过了设定的参考值。

滤波器时钟 f_{FLT} 是由 HXT 时钟提供。如果将 OPA 输出 OCPAO_n 作为 ADC 的输入，则通过 A/D 转换功能可以读出放大后的电压的值。DAC 输出电压的值由 OCPnREF 寄存器设置，DAC 的输出值被定义为：

$$\text{DAC } V_{OUT} = (\text{DAC } V_{REF}/256) \times \text{OCPnREF } [7:0] \quad （1）$$

OCP 寄存器

OCPnC0 和 OCPnC1 寄存器是 OCPn 控制寄存器，用来控制 OCPn 工作模式、PGA 和滤波器功能。OCPnREF 寄存器用来为过电流保护提供参考电压。OCPnACAL 和 OCPnCCAL 用于运算放大器和比较器的输入失调校准。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OCPnC0	OCPnM1	OCPnM0	—	—	—	—	—	—
OCPnC1	OCPnO	—	OCPnG2	OCPnG1	OCPnG0	OCPnFLT2	OCPnFLT1	OCPnFLT0
OCPnREF	D7	D6	D5	D4	D3	D2	D1	D0
OCPnACAL	OCPnAOFM	OCPnARS	OCPnAOF5	OCPnAOF4	OCPnAOF3	OCPnAOF2	OCPnAOF1	OCPnAOF0
OCPnCCAL	OCPnAXCX	OCPnCOFM	OCPnCRS	OCPnCOF4	OCPnCOF3	OCPnCOF2	OCPnCOF1	OCPnCOF0

OCP 寄存器列表（n=0 或 1）

OCPnC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	OCPnM1	OCPnM0	—	—	—	—	—	—
R/W	R/W	R/W	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

Bit 7~6 **OCPnM1~ OCPnM0**：模式选择位
00：OCPn 功能除能，S1 和 S3 闭合，S0 和 S2 断开
01：OCPn 功能在同相模式中使能，S0 和 S3 闭合，S1 和 S2 断开
10：OCPn 功能在反相模式中使能，S1 和 S2 闭合，S0 和 S3 断开
11：校准模式，S1 和 S3 闭合，S0 和 S2 断开
注：若选择 OCPn 功能除能则 OPA、CMP、DAC 和滤波器功能全部关闭，比较器输出和 OCPAO_n 都为低电平

Bit 5~0 未定义，读为“0”

OCPnC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	OCPnO	—	OCPnG2	OCPnG1	OCPnG0	OCPnFLT2	OCPnFLT1	OCPnFLT0
R/W	R	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	0

Bit 7 **OCPnO**：OCPn 输出（只读位）
Bit 6 未定义，读为“0”
Bit 5~3 **OCPnG2~OCPnG0**：OCPn OPA 增益
000：1
001：5
010：10
011：15
100：20
101：30
110：40
111：50

Bit 2~0 **OCPnFLT2~ OCPnFLT0**：OCPn 解调器滤波器选择位
000：0 t_{FLT}（无滤波）
001：1~2×t_{FLT}
010：3~4×t_{FLT}
011：7~8×t_{FLT}
100：15~16×t_{FLT}
101：31~32×t_{FLT}
110：63~64×t_{FLT}
111：127~128×t_{FLT}
注：$t_{FLT}=f_H/4$，$f_H=f_{HXT}$，$t_{FLT}=1/f_{FLT}$

OCPnREF 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 **OCPnREF**：过流保护功能参考电压选择位
参考电压 =（DAC 参考电压 /256）×（N），N=OCPnREF[7:0]

OCPnACAL 寄存器 – 过电流 OPA 校准寄存器

Bit	7	6	5	4	3	2	1	0
Name	OCPnAOFM	OCPnARS	OCPnAOF5	OCPnAOF4	OCPnAOF3	OCPnAOF2	OCPnAOF1	OCPnAOF0
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 **OCPnAOFM**：输入失调电压校准模式或正常工作模式选择位
0：正常工作模式
1：输入失调电压校准模式

Bit 6 **OCPnARS**：输入失调电压校准参考选择位
0：选择负相输入作为参考输入
1：选择正相输入作为参考输入

Bit 5~0 **OCPnAOF5~OCPnAOF0**：输入失调电压校准控制位

OCPnCCAL 寄存器 – 过电流比较器校准寄存器

Bit	7	6	5	4	3	2	1	0
Name	OCPnAXCX	OCPnCOFM	OCPnCRS	OCPnCOF4	OCPnCOF3	OCPnCOF2	OCPnCOF1	OCPnCOF0
R/W	R	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 **OCPnAXCX**：OPAn 和比较器校准输出，正逻辑（只读位）
若 OCPnAOFM=1，此位是 OPA 校准输出
若 OCPnCOFM=1，此位是比较器校准输出
注：OCPnAOFM 和 OCPnCOFM 位不能同时为“1”

Bit 6 **OCPnCOFM**：输入失调电压校准模式或正常工作模式选择位
0：正常工作模式
1：输入失调电压校准模式

Bit 5 **OCPnCRS**：输入失调电压校准参考选择位
0：选择负相输入作为参考输入
1：选择正相输入作为参考输入

Bit 4~0 **OCPnCOF4~OCPnCOF0**：输入失调电压校准控制位

输入电压范围

输入电压可以为正或负，结合 PGA 的工作模式，可以产生更灵活的应用。

（1）$V_{IN} > 0$，PGA 同相模式，PGA 输出电压为：

$$VO_{PGA}=(1+R_2/R_1)\times V_{IN} \quad （2）$$

（2）设置 PGA 工作于同相模式，可以产生单位增益缓冲器功能。

如果 OCPnM [1:0]=01，OCPnG [2:0]=000，PGA 的增益选择为 1，则可配置为单位增益缓冲器。开关 S2，S3 内部断开，PGA 的输出电压为：

$$VO_{PGA}=V_{IN} \quad （3）$$

（3）$0 > V_{IN} > -0.7V$，PGA 反相模式，PGA 输出电压为：

$$VO_{PGA}=-(R_2/R_1)\times V_{IN} \quad （4）$$

注：如果 V_{IN} 为负，其值不能再低于 -0.7V 以避免漏电流产生。

失调电压校准

过流保护电路有四种工作模式，通过 OCPnM1~OCPnM0 位进行选择。其中之一为校准模式。在此模式可以对运算放大器和比较器进行失调电压校准。

OPAMP 校准：

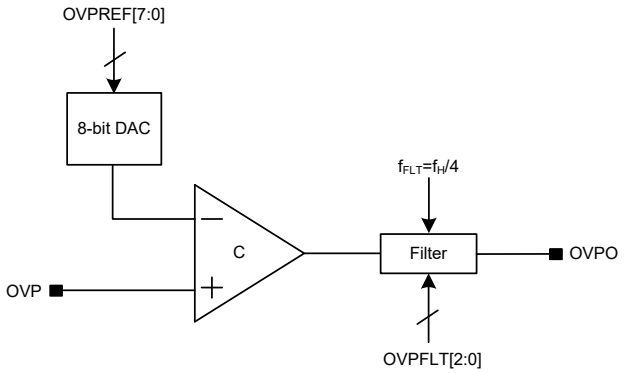
- 步骤 1：设置 OCPnM1~OCPnM0 = “11”，OCPnAOFM=1，OCP 处于 OPAMP 校准模式。
- 步骤 2：设置 OCPnAOF5~OCPnAOF0 = “000000”，然后读取 OCPnAXCX 位的状态。
- 步骤 3：设置 OCPnAOF[5:0]= OCPnAOF[5:0]+1，然后读取 OCPnAXCX 位的状态，如果 OCPnAXCX 位发生变化，则记录 OCPnAOF[5:0] 数据作为 VOS1。
- 步骤 4：设置 OCPnAOF[5:0]= “111111”，然后读取 OCPnAXCX 位的状态。
- 步骤 5：设置 OCPnAOF[5:0]= OCPnAOF[5:0]-1，然后读取 OCPnAXCX 位的状态，如果 OCPnAXCX 位发生变化，则记录 OCPnAOF[5:0] 数据作为 VOS2。
- 步骤 6：存储数据 VOS = (VOS1+VOS2)/2 到寄存器 OCPnAOF[5:0] 位中。校准完成。

比较器校准：

- 步骤 1：设置 OCPnM1~OCPnM0 = “11”，OCPnCOFM=1，OCPn 处于比较器校准模式。
- 步骤 2：设置 OCPnCOF[4:0] = “00000”，然后读取 OCPnAXCX 位的状态。
- 步骤 3：设置 OCPnCOF[4:0] = OCPnCOF[4:0] +1，然后读取 OCPnAXCX 位的状态，如果 OCPnAXCX 位发生变化，则记录寄存器数据作为 VOS1。
- 步骤 4：设置 OCPnCOF[4:0] = “11111”，然后读取 OCPnAXCX 位的状态。
- 步骤 5：设置 OCPnCOF[4:0] = OCPnCOF[4:0] -1，然后读取 OCPnAXCX 位的状态，如果 OCPnAXCX 位发生变化，则记录寄存器数据作为 VOS2。
- 步骤 6：存储数据 VOS = (VOS1+VOS2)/2 到寄存器 OCPnCOF[4:0] 位中。校准完成。

过电压保护功能 – OVP

OVP 是过电压保护（Over Voltage Protect）的缩写。OVP 检测到的输入电压与要监测的源电压成正比，如果输入电压大于 DAC 设置的参考电压，OVP 将会产生一个输出信号表示源电压超过了规格。



过压保护电路

过压保护电路操作

过压保护电路中的比较器将 OVP 引脚输入的电压与 8-bit DAC 产生的的参考电压相比较。此 DAC 参考电压可通过 OVPREF 寄存器设置。可通过 OVPO 输出信号判断源电压是否超过设定的规格。

OVP 寄存器

过压保护的所有工作由多个寄存器控制。OVPC 寄存器为控制寄存器，用来使能 / 除能 OVP 功能及设置滤波功能。OVPREF 寄存器用于提供过压保护电路的参考电压，OVPCCAL 用于比较器输入失调校准功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
OVPC	OVPEN	—	—	OVPO	—	OVPFLT2	OVPFLT1	OVPFLT0
OVPREF	D7	D6	D5	D4	D3	D2	D1	D0
OVPCCAL	OVPCX	OVPCOFM	OVPCRS	OVPCOF4	OVPCOF3	OVPCOF2	OVPCOF1	OVPCOF0

OVP 寄存器列表

OVPC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	OVPEN	—	—	OVPO	—	OVPFLT2	OVPFLT1	OVPFLT0
R/W	R/W	—	—	R	—	R/W	R/W	R/W
POR	0	—	—	0	—	0	0	0

Bit 7 **OVPEN**：过压保护功能控制位
0：OVP 功能（包括比较器和 DAC）除能
1：OVP 功能（包括比较器和 DAC）使能

Bit 6~5 未定义，读为“0”

Bit 4 **OVPO**：OVP 输出位，为只读位

Bit 3 未定义，读为“0”

Bit 2~0 **OVPFLT2~ OVPFLT0**：OVP 滤波器选择位

000：0 t_{FLT}（无滤波）
001：1~2×t_{FLT}
010：3~4×t_{FLT}
011：7~8×t_{FLT}
100：15~16×t_{FLT}
101：31~32×t_{FLT}
110：63~64×t_{FLT}
111：127~128×t_{FLT}

注：$t_{FLT}=f_H/4$，$f_H=f_{HXT}$，$t_{FLT}=1/f_{FLT}$

OVPREF 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 **OVPREF**：过压保护功能参考电压选择位
参考电压 =（DAC 参考电压 /256）×（N），N=OVPREF[7:0]

OVPCCAL 寄存器 – 比较器校准寄存器

Bit	7	6	5	4	3	2	1	0
Name	OVPCX	OVPCOFM	OVPCRS	OVPCOF4	OVPCOF3	OVPCOF2	OVPCOF1	OVPCOF0
R/W	R	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 **OVPCX**：比较器校准输出，正逻辑（只读位）

Bit 6 **OVPCOFM**：输入失调电压校准模式或正常工作模式选择位
0：正常工作模式
1：输入失调电压校准模式

Bit 5 **OVPCRS**：输入失调电压校准参考选择位
0：选择负相输入作为参考输入
1：选择正相输入作为参考输入

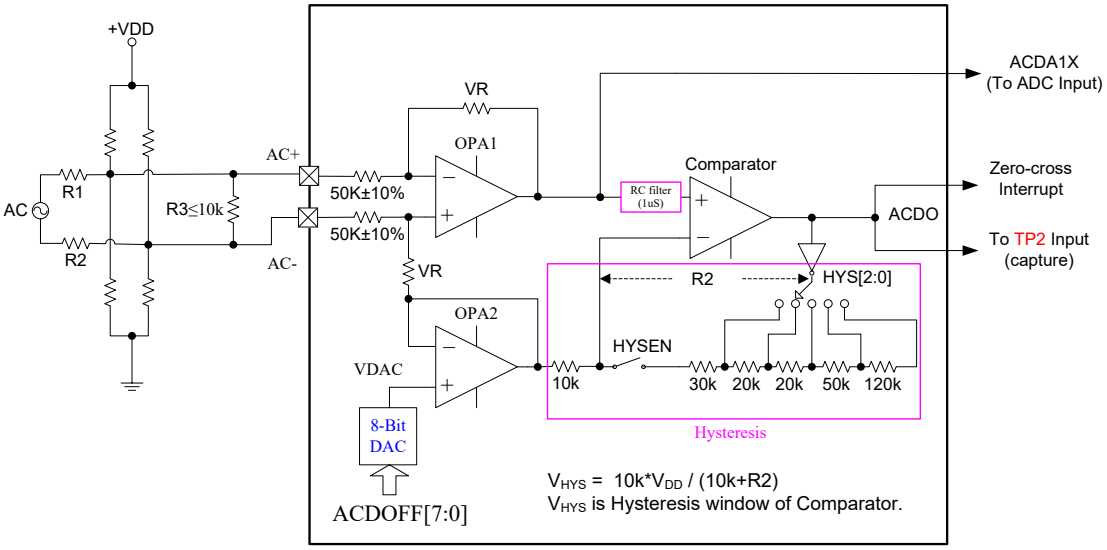
Bit 4~0 **OVPCOF4 ~ OVPCOF0**：输入失调电压校准控制位

比较器校准功能

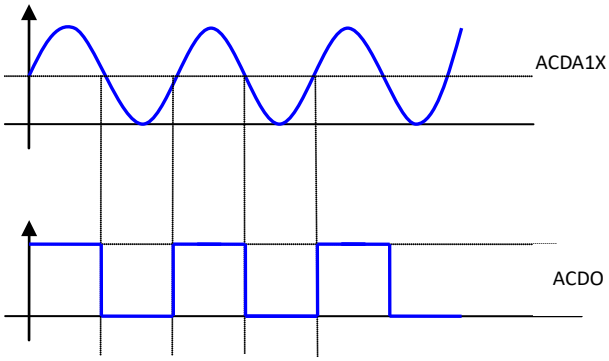
- 步骤 1：设置 OVPCOFM=1，OVP 处于比较器校准模式。
- 步骤 2：设置 OVPCOF[4:0] = “00000”，然后读取 OVPCX 位的状态。
- 步骤 3：设置 OVPCOF[4:0] = OVPCOF[4:0] +1，然后读取 OVPCX 位的状态，如果 OVPCX 位发生变化，则记录寄存器数据作为 VOS1。
- 步骤 4：设置 OVPCOF[4:0] = “11111”，然后读取 OVPCX 位的状态。
- 步骤 5：设置 OVPCOF[4:0]= OVPCOF[4:0] -1，然后读取 OVPCX 位的状态，如果 OVPCX 位发生变化，则记录寄存器数据作为 VOS2。
- 步骤 6：存储数据 VOS = (VOS1+VOS2)/2 到寄存器 OVPCOF[4:0] 位中。校准完成。

交流检测电路

交流检测电路用于检测应用程序被供给的交流电源是否正常。220V 的交流电压通过 AC+ 和 AC- 引脚连接到单片机进行检测。检测结果 ACDA1X 可输入到 A/D 转换器测量出实际的电压值。交流检测电路中包含了一个比较器功能可将检测结果 ACDA1X 与通过寄存器设置的参考电压进行比较，从而获得比较输出信号 ACDO。ACDO 信号可输入到 Zero-cross 中断电路或作为标准型 TM 捕捉输入信号，从 TP2 引脚输入。ACDA1X 和 ACDO 信号也可连接到外部引脚作为其它应用程序的参考或测量信号。需注意的是 AC 检测电路中的运算放大器和比较器使用 P 型输入架构，迟滞电压范围为 50mV~100mV。



交流检测电路图



交流检测输出信号

交流检测寄存器

交流检测功能由 ACDC 控制寄存器和 ACDOFF 寄存器进行设置，具体操作参考寄存器位描述。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACDC	ACDEN	HYSEN	HYS2	HYS1	HYS0	—	—	VR
ACDOFF	D7	D6	D5	D4	D3	D2	D1	D0

交流检测寄存器列表

ACDC 寄存器 – 交流检测控制寄存器

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	ACDEN	HYSEN	HYS2	HYS1	HYS0	—	—	VR
R/W	R/W	R/W	R/W	R/W	R/W	—	—	R/W
POR	0	0	0	0	0	—	—	0

Bit 7 **ACDEN**：交流检测使能控制位
0：除能
1：使能

Bit 6 **HYSEN**：迟滞电压使能控制位
0：除能
1：使能

Bit 5~3 **HYS2~HYS0**：比较器输入端迟滞功能电阻选择位
000：30kΩ
001：50kΩ
010：70kΩ
011：120kΩ
1xx：240kΩ

Bit 2~1 未定义，读为“0”

Bit 0 **VR**：交流检测电路中 VR 电阻值选择位
0：50kΩ
1：100kΩ

ACDOFF 寄存器 – 交流检测失调电压寄存器

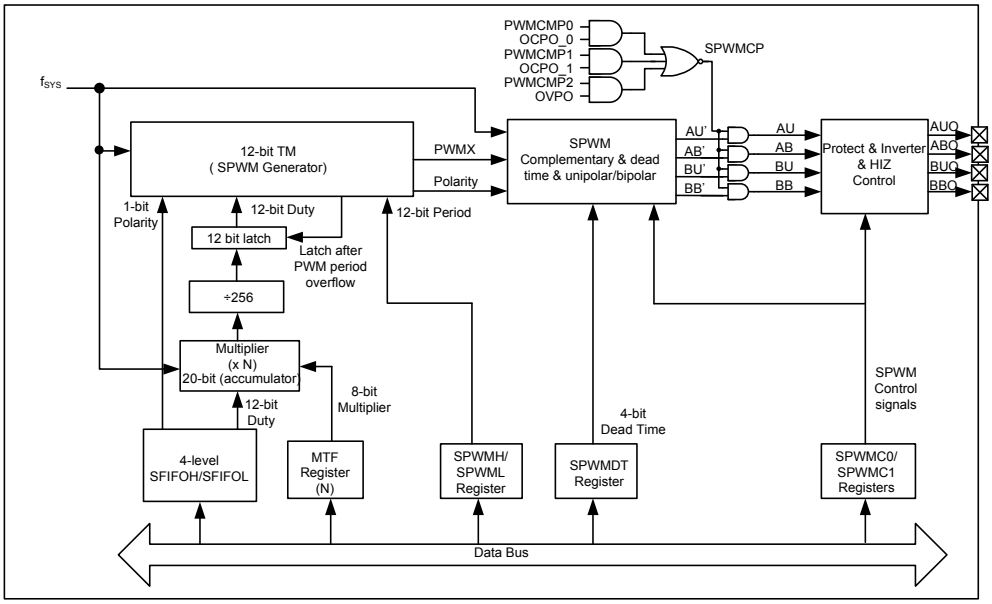
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：AC 失调电压校准 DAC 数据位

DAC 输出 $= \frac{(D7 \sim D0) \times V_{DD}}{256}$

正弦波 PWM 发生器

正弦波 PWM 发生电路，包括一个 12-bit 定时器模块用于产生正弦 PWM 信号，一个带死区时间插入功能的 PWM 互补输出控制电路及一个对输出信号进行保护和反相控制电路。通过控制输出信号 AUO, ABO, BUO 及 BBO 产生不同方向电流切换外部 MOSFET 产生正弦波信号。



正弦波 PWM 发生电路方框图

正弦波 PWM 寄存器

整个 SPWM 产生功能操作由八个寄存器控制。一对 PWM 周期寄存器，SPWMH 和 SPWML，用来存放 12-bit PWM 周期值，一对 PWM 占空比寄存器，SFIFOH 和 SFIFOL，用于存放 4-level PWM 占空比值。MTF 寄存器用于设置乘法器中 8-bit 乘数值，从而调整 PWM 信号的占空比周期。SPWMDT 寄存器设置死区时间。剩余的两个寄存器 SPWMC0 和 SPWMC1 为整个电路控制寄存器，可进行 PWM 设置，互补输出控制，保护和反相控制等。

需注意的是，因周期和占空比寄存器对为 12-bit，含有低字节和高字节结构。在写数据时，应先写入低字节数据，再写入高字节数据。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SPWMH	—	—	—	—	D11	D10	D9	D8
SPWML	D7	D6	D5	D4	D3	D2	D1	D0
SFIFOH	SPWMPOL	—	—	—	SFIFO11	SFIFO10	SFIFO9	SFIFO8
SFIFOL	SFIFO7	SFIFO6	SFIFO5	SFIFO4	SFIFO3	SFIFO2	SFIFO1	SFIFO0
SPWMC0	SPWMT	SPWMON	SEN	HIZ	—	PRTEN	DTYVAL	PRST
SPWMC1	PWMCMP2	PWMCMP1	PWMCMP0	—	BBINV	BUINV	ABINV	AUINV
MTF	D7	D6	D5	D4	D3	D2	D1	D0
SPWMDT	—	—	—	—	DT3	DT2	DT1	DT0

SPWM 发生器寄存器列表

SPWMDT 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	DT3	DT2	DT1	DT0
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 3~0　　**DT3~DT0**：SPWM 死区时间控制位

死区时间 $= \frac{(DT3 \sim DT0) \times 2}{f_{SYS}}$，其中 f_{SYS} 为系统时钟频率。

SPWMH, SPWML 寄存器对

Bit	SPWMH								SPWML							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	—	—	—	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0	0	0	0	0	0	0	0	0

“—“：未定义，读为“0”

D11~D0：SPWM 12-bit 周期控制位

TM 输出周期 $= \frac{SPWM[11:0]+1}{f_{SYS}}$，其中 f_{SYS} 为系统时钟频率， SPWM[11:0] 为 SPWM 周期寄存器值，其值必须大于 7。

SPWM 12-bit TM 由系统时钟驱动， 输出周期可通过 SPWMH / SPWML 寄存器对进行设置。在对此寄存器写值时，应先写入低字节寄存器再写入高字节寄存器。.

MTF 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

D7~D0：乘法器乘数 N 控制位

N =D[7:0]

SFIFOH, SFIFOL 寄存器

Bit	SFIFOH								SFIFOL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	SPWMPOL	—	—	—	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	W	—	—	—	W	W	W	W	W	W	W	W	W	W	W	W
POR	0	—	—	—	0	0	0	0	0	0	0	0	0	0	0	0

SPWMPOL：用于指示当前采样所处周期

0：负半周弦波

1：正半周弦波

“—“：未定义，读为“0”

D11~D0：SPWM 12-bit 占空比控制位

TM 输出占空比 $= \frac{SFIFO[1:0] \times (MTF[7:0]/256)}{f_{SYS}}$，其中 f_{SYS} 为系统时钟频率，SFIFO[11:0] 为 SPWM 占空比寄存器值，MTF[7:0] 为 MTF 寄存器值。

SPWM 12-bit TM 由系统时钟驱动，输出占空比由 SFIFOH/SFIFOL 及 MTF 寄存器控制。在对 SFIFOH/SFIFOL 寄存器写值时，应先写入低字节寄存器再写入高字节寄存器。此 12 位的 SFIFO 寄存器对采用 4-level FIFO 结构。用户一次存放四笔数据，每 4 个取样周期存放一次占空比数据。当 4-level SFIFO 占空比数据读完后则发出中断，通知 MCU 写入下四笔占空比数据至 SFIFO 寄存器对中。

SPWMC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SPWMT	SPWMON	SEN	HIZ	—	PRTEN	DTYVAL	PRST
R/W	R/W	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	1	0	0	0	—	1	0	0

Bit 7 **SPWMT**：SPWM 驱动模式选择位
0：双极性模式
1：单极性模式

Bit 6 **SPWMON**：SPWM 功能使能控制位
0：除能
1：使能
若将此位清零，正弦波 PWM 产生电路中 12-bit TM, 互补及死区时间插入电路都将关闭，AU’/AB’BU’/BB’信号将被强制拉低。但此位不会影响到 SPWM 保护和反相电路。

Bit 5 **SEN**：SPWM 有效及输出使能控制位
0：除能
1：使能
若将此位清零，SPWM 互补信号 AU’/AB’BU’/BB’将停留在当前状态，同时互补输出对 AUO/ABO/BUO/BBO 由 HIZ, PWMCMP0~PWMCMP2, AUINV/ABINV/BUINV/BBINV 位控制。
若将此位置 1，SPWM 功能将被启动，12-bit 的 PWM 周期寄存器对以及占空比寄存器对中的值将依次加载到 12-bit TM 中，并与计数器中的值进行比较。

Bit 4 **HIZ**：AUO/ABO/BUO/BBO 高阻态控制位
0：当 SEN=0 时，AUO/ABO/BUO/BBO 输出由 AUINV/ABINV/BUINV/BBINV 位控制
1：当 SEN=0 时，AUO/ABO/BUO/BBO 处于高阻态

Bit 3 未定义，读为“0”

Bit 2 **PRTEN**：SPWM 保护功能控制位
0：除能
1：使能
若此位置 1，将开启保护功能，从而可避免 AUO&ABO 或 BUO&BBO 同时为有效电平的状况。有效电平由对应的 AUINV/ABINV/BUINV/BBINV 位选择，若 INV 位为 0，有效电平为 1，若 INV 位为 1，有效电平为 0。

Bit 1 **DTYVAL**：SPWM 占空比值选择位
0：占空比 = SFIFO[11:0] x N /256，其中 N 为 MTF 寄存器值
1：占空比 = SFIFO[11:0]

Bit 0 **PRST**：FIFO 读 / 写 Point 清零控制位
0：无动作
1：将读 / 写 Point(RP/WP) 清零

SPWMC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PWMCMP2	PWMCMP1	PWMCMP0	—	BBINV	BUINV	ABINV	AUINV
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 **PWMCMP2**：OVPO 为高时 AU, AB, BU, BB 状态控制位
0：不会立即切为低电平
1：立即切为低电平

Bit 6 **PWMCMP1**：OCPO_1 为高时 AU, AB, BU, BB 状态控制位
0：不会立即切为低电平
1：立即切为低电平

Bit 5 **PWMCMP0**：OCPO_0 为高时 AU, AB, BU, BB 状态控制位
0：不会立即切为低电平
1：立即切为低电平

Bit 4 未定义，读为“0”

Bit 3 **BBINV**：BBO 反相控制位
0：同相
1：反相

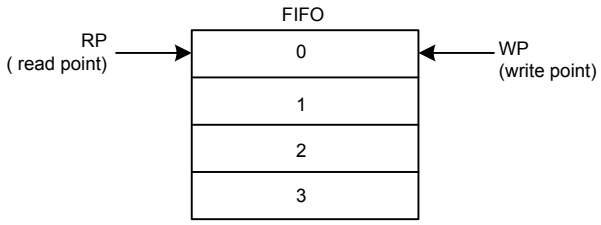
Bit 2 **BUINV**：BUO 反相控制位
0：同相
1：反相

Bit 1 **ABINV**：ABO 反相控制位
0：同相
1：反相

Bit 0 **AUINV**：AUO 反相控制位
0：同相
1：反相

FIFO 操作

SFIFO 寄存器对用于存放 12-bit 的 SPWM 波形的占空比值，用户可通过改写寄存器对中的值对 SPWM 的占空比进行调整。此 12-bit 数据采用 4-level 结构，需每一次写入时，存放四笔数据，并依序载入占空比寄存器中，经过相应的运算处理，与对应的周期寄存器中值一起控制生成的 SPWM 波形。当 4-level SFIFO 四笔 duty 数据读完时，则发出 SPWM 中断通知 MCU 存入下四笔 duty 值至 4-level SFIFO 寄存器对中。



4-Level FIFO

注：1. 当一个 SPWM 周期完成，FIFO 中的数据被读取后，RP 值会自动加 1。
（RP=0 → 1 → 2 → 3 → 0 → 1 → 2 → 3 …）
2. 无论 SPWMON 值为 0 或 1，只要对 FIFO 写入数据，WP 值会自动加 1。
（RP=0 → 1 → 2 → 3 → 0 → 1 → 2 → 3 …）
3. 当第四笔数据读出时，即 RP=3 时，会产生 SPWM 中断。
4. 设置 PRST=1，RP 及 WP 值会被清零。
5. 每次读取 FIFO 数据后，经过乘法器操作处理需要 8 个系统时钟。

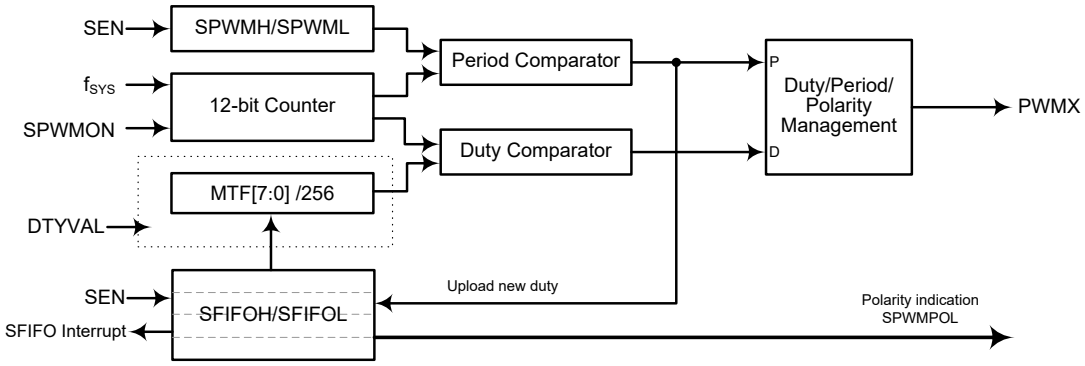
建议的 FIFO 操作步骤

0. SPWMON=0
1. 初始化 RP 及 WP 指针：设置 PRST=1, 即清零 PR 及 WP 值
2. 写入四笔数据到 FIFO（WP 初始值为 0，每写入一笔，WP 值加 1）
3. SPWMON=1
4. 等待 SPWM 中断发生，并于中断后再次写入四笔 duty 数据
5. 重复步骤 4，直到 SPWMON 位被清零。

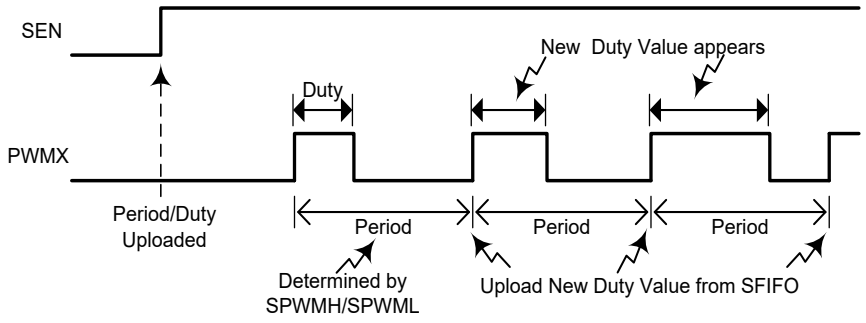
注：1. 在 duty 数据被读取后，另一周期开始时，下一笔 duty 数据会被自动加载到寄存器中。
2. SPWM duty 经乘法器处理至少需要 8 个时钟周期，所以设置的周期值不得小于 10 个系统时钟。

SPWM TM 操作

此 12-bit SPWM TM 由时钟系统时钟驱动产生 PWM 信号，PWMX。寄存器对 SFIFOH/SFIFOL 及 SPWMH/SPWML 分别用于控制产生的 PWM 波形占空比及周期。PWMX 信号周期由 SPWMH/SPWML 寄存器值及系统采样周期决定，占空比由 SFIFOH/SFIFOL 寄存器决定。此单片机同时提供了 8-bit MTF 寄存器设置乘法器中的乘数值 N，因此可进一步对占空比进行调整。通过 SPWMC0 寄存器 DTYVAL 位可选择占空比的值来自 SFIFO[11:0]´N/256 结果或是不经过乘法器直接为 SFIFO[11:0] 值。可通过查看指示位 SPWMPOL 位，得出当前采样为正半周还是负半周。



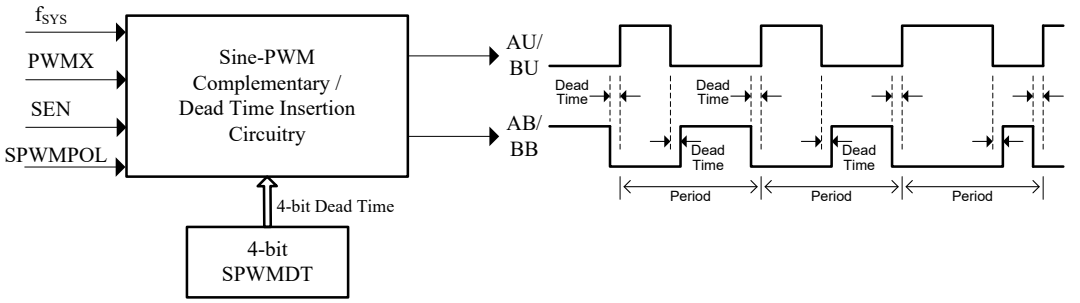
12-bit SPWM TM 方框图



12-bit SPWM 波形

互补式 PWM 信号及死区时间插入

通过 12-bit SPWM TM 产生的 PWMX 信号可通过互补控制电路，产生互补的信号对即 AU&BU 互补信号对和 BU&BB 互补信号对。死区时间插入电路可避免产生的互补信号出现同时有效的状况。插入的死区时间由 4-bit 寄存器 SPWMDT 进行设置。

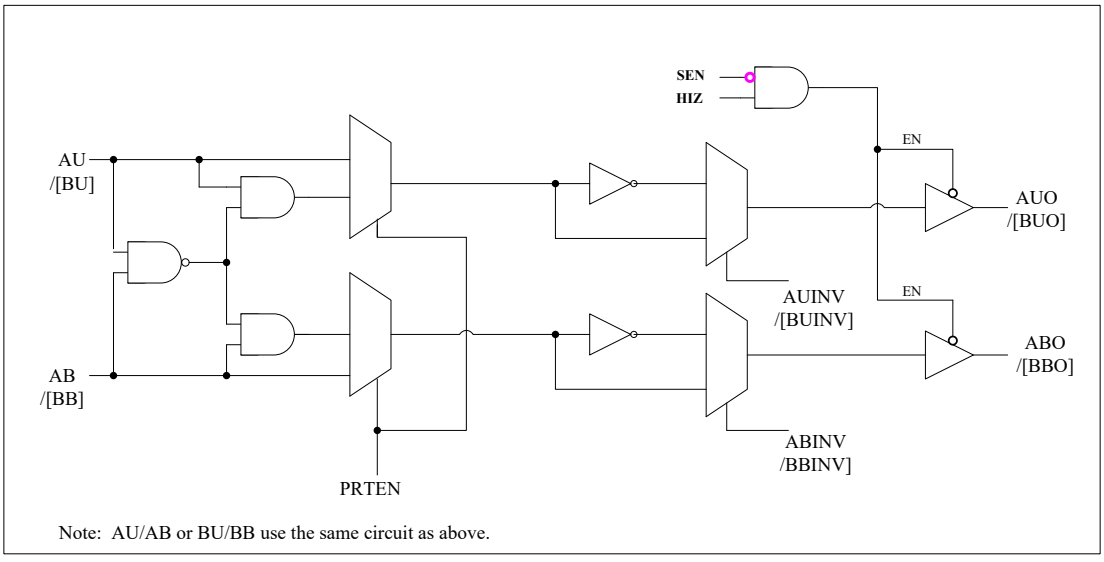


SPWM 互补 / 死区时间插入功能框图

保护和反相控制功能

虽然插入死区时间可以防止互补信号对出现同时有效的状况。但一些无法预期的错误，如软件误动作或电磁干扰等也会导致互补信号对同时为高状态。因些此单片机提供了另外一个保护功能，即当 AU/AB 互补信号或 BU/BB 互补信号对同时为有效状态时，将强制将其拉低。反相电路可用于将输出的信号进行反相后输出，可通过相应的反相控制位选择开启或不开启反相功能。

此单片机同时也提供过电流和过电压保护功能，在 OCP 及 OVP 章节中有具体描述。若有过电流或过电压状况发生，OVPO, OCPO_1 及 OCPO_0 位将被置高，同进 AU, AB, BU 和 BB 信号被强制拉低。



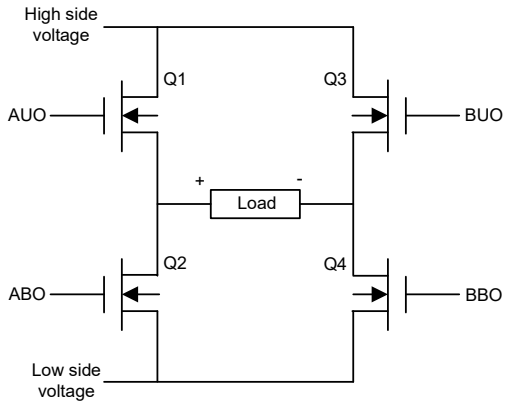
SPWM 保护及反相控制电路图

SPWM 驱动控制模式

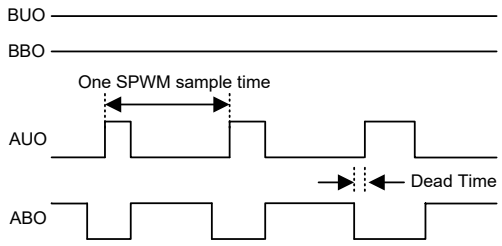
此单片机提供单极性和双极性驱动控制模式，通过设置 SPWMT 位可进行灵活切换。下述表格概括了在不同驱动模式下，输出信号的状态。

	单极性模式（SPWMT=1）		双极性模式（SPWMT=0）	
	正半周（SPWMPOL=1）	负半周（SPWMPOL=0）	正半周（SPWMPOL=1）	Negative half-cycle SPWMPOL=0
AU	PWM 信号	保持低	PWM 信号	AB 信号反相，加死区处理
AB	AU 信号反相，加死区处理	保持高	AU 信号反相，加死区处理	PWM 信号
BU	保持低	PWM 信号	除能	除能
BB	保持高	BU 信号反相，加死区处理	除能	除能

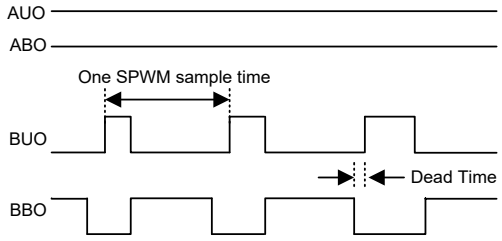
单极性 SPWM 驱动控制电路



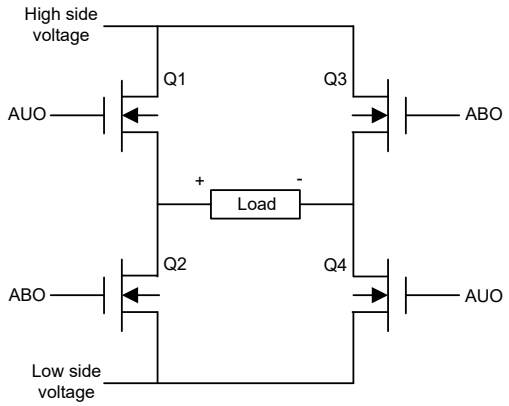
单极性模式，正半周



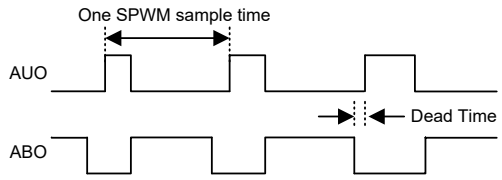
单极性模式，负半周



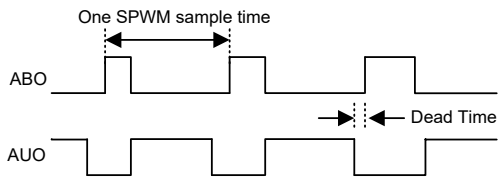
双极性 SPWM 驱动控制电路



双极性模式，正半周



双极性模式，负半周

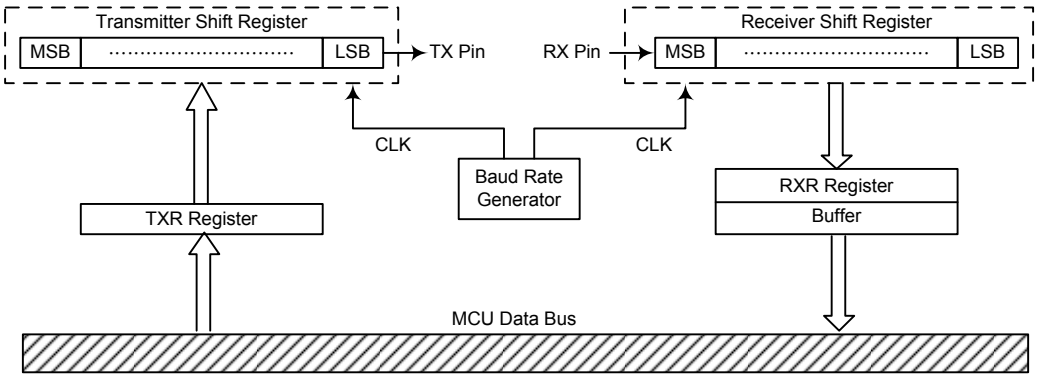


UART 模块串行接口

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

UART 模块特性如下：

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



UART 数据传输方框图

UART 外部引脚

内部 UART 有两个外部引脚 TX 和 RX，可与外部串行接口进行通信。TX 和 RX 引脚相应的为 UART 接口的发送和接收引脚。在使用 UART 功能之前，要先正确设置相关的引脚共用功能选择寄存器选择 RX 和 TX 引脚功能。若 UARTEN 位和 TXEN/RXEN 位均为“1”时，将会自动设置 TX 脚为输出，RX 为输入状态，同时断开 RX 和 TX 引脚上的上拉电阻。若 UARTEN 位或 TXEN/RXEN 位为“0”时，TX 和 RX 引脚功能除能，将会作为通用 I/O 或共用功能使用

UART 数据传输方案

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

数据在波特率发生器的控制下，低位在前高位在后，从外部引脚 RX 进入接收移位寄存器 RSR。当数据接收完成，数据从接收移位寄存器移入可被用户程序操作的 RXR 寄存器中。RXR 寄存器被映射到单片机数据存储器中，而接收移位寄存器没有实际地址，所以接收移位寄存器不可直接操作。需要注意的是，上述发送寄存器 TXR 和接收寄存器 RXR，其实是共享一个地址的数据寄存器 TXR_RXR 寄存器。

UART 状态和控制寄存器

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

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

UART 寄存器列表

USR 寄存器

寄存器 USR 是 UART 的状态寄存器，可以通过程序读取从判断当前 UART 状态。USR 寄存器中所有位是只读的。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，表明正在接收数据；若 RIDLE=1，表明接收器空闲。在接收到停止位，还未接收到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，用来设置各种 UART 功能，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”表示未知

Bit 7 **UARTEN**：UART 功能使能位

0：UART 除能，TX 和 RX 脚作为通用 I/O 或其它共用功能

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 作为通用 I/O 或其它共用功能；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**：奇偶校验功能控制位

0: 除能

1: 使能

此位为奇偶校验功能控制位。PREN=1，使能奇偶校验；PREN=0，除能奇偶校验。

Bit 4 **PRT**：奇偶校验类型选择位

0：偶校验

1：奇校验

奇偶校验类型选择位。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：不发送暂停字

1：发送暂停字

TXBRK 是暂停字发送控制位。TXBRK=0，不发送暂停字，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 bit 8 数据 (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 bit 8 数据 (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 的另外一个控制寄存器，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 引脚处于浮空状态。若 TXEN=1 且 UARTEN=1，则发送将被使能，TX 引脚将由 UART 来控制。在数据传输时清除 TXEN 将中止数据发送且复位发送器，此时 TX 引脚处于浮空状态。

Bit 6 **RXEN**：UART 接收使能位

0：UART 接收除能

1：UART 接收使能

此位为接收使能位。RXEN=0，接收将被除能，接收器立刻停止工作。另外缓冲器将被复位，此时 RX 引脚处于浮空状态。若 RXEN=1 且 UARTEN=1，则接收将被使能，RX 引脚将由 UART 来控制。在数据传输时清除 RXEN 将中止数据接收且复位接收器，此时 RX 引脚处于浮空状态。

Bit 5 **BRGH**：波特率发生器高 / 低速选择位

0：低速波特率

1：高速波特率

此位为波特率发生器高低速选择位，它和 BRG 寄存器一起控制 UART 的波特率。BRGH=1，为高速模式；BRGH=0，为低速模式。

Bit 4 **ADDEN**：地址检测使能位

0：地址检测除能

1：地址检测使能

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

Bit 3 **WAKE**：RX 脚下降沿唤醒 UART 功能使能位

0：RX 脚下降沿唤醒 UART 功能除能

1：RX 脚下降沿唤醒 UART 功能使能

此位为 RX 引脚下降沿唤醒 UART 功能的使能和除能控制位。该位仅当 UART 时钟源关闭时有效。若 UART 时钟源开启，RX 引脚唤醒 UART 功能无效。当 UART 时钟关闭且 WAKE 位为 1，若 RX 引脚唤醒 UART 的中断使能，当 RX 引脚发生下降沿时会触发 UART 唤醒请求以告知单片机需唤醒 UART 功能，此唤醒动作是通过应用程序将 UART 时钟开启来实现的。相反的，若 WAKE 位为零，则即使 RX 引脚发生下降沿也无法恢复 UART 功能。

Bit 2 **RIE**：接收中断使能位

0：接收中断除能

1：接收中断使能

此位为接收中断使能或除能位。若 RIE=1，且 OERR 或 RXIF 置位时，UART 的中断请求标志置位；若 RIE=0，UART 中断请求标志不受 OERR 和 RXIF 影响。

Bit 1 **TIIE**：发送器空闲中断使能位
0：发送器空闲中断除能
1：发送器空闲中断使能
此位为发送器空闲中断的使能或除能位。若 TIIE=1，当 TIDLE 置位时，UART 的中断请求标志置位；若 TIIE=0，UART 中断请求标志不受 TIDLE 的影响。

Bit 0 **TEIE**：发送寄存器为空中断使能位
0：发送寄存器为空中断除能
1：发送寄存器为空中断使能
此位为发送寄存器为空中断的使能或除能位。若 TEIE=1，当 TXIF 置位时，UART 的中断请求标志置位；若 TEIE=0，UART 中断请求标志不受 TXIF 的影响。

TXR __ RXR 寄存器

TXR_RXR 寄存器为数据寄存器，用来存储发送到 TX 引脚或从 RX 引脚接收到的数据。

Bit	7	6	5	4	3	2	1	0
Name	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”表示未知

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

波特率发生器

UART 自身具有一个波特率发生器，通过它可以设置数据传输速率。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 位一起控制。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。下表的波特率计算公式中，除数取决于 BRG 寄存器的值 N，N 的范围是 0 到 255。

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

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

下面举例怎样计算 BRG 寄存器中的值 N 和误差。

BRG 寄存器

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

“x”未知

Bit 7~0 **BRG7~BRG0**：波特率值
软件设置 BRGH 位（设置波特率发生器的速度）和 BRG 寄存器（设置波特率的值），一起控制 UART 的波特率。

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

转换后的公式 $N=\frac{f_{SYS}}{(BR\times 64)}-1$

带入参数 $N=\frac{4000000}{(4800\times 64)}-1=12.0208$

取最接近的值，十进制 12 写入 BRG 寄存器，实际波特率如下

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

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

下面的表格给出了 BRGH 取不同值时的实际波特率和误差。

波特率 K/BPS	BRGH=0					
	f_{SYS} = 16 MHz			f_{SYS} = 20 MHz		
	BRG	Kbaud	误差 (%)	BRG	Kbaud	误差 (%)
0.3	—	—	—	—	—	—
1.2	207	1.202	0.16	259	1.202	0.16
2.4	103	2.404	0.16	129	2.404	0.16
4.8	51	4.808	0.16	64	4.808	0.16
9.6	25	9.615	0.16	32	9.470	-1.36
19.2	12	19.231	0.16	15	19.531	1.73
38.4	6	35.714	-6.99	7	39.063	1.73
57.6	3	62.5	8.51	4	62.5	8.51
115.2	1	125	8.51	2	104.17	-9.58
250	0	250	0	0	312.5	25

BRGH = 0 时的波特率和误差

波特率 K/BPS	BRGH=1					
	f_{SYS} = 16 MHz			f_{SYS} = 20 MHz		
	BRG	Kbaud	误差 (%)	BRG	Kbaud	误差 (%)
0.3	—	—	—	—	—	—
1.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
4.8	207	4.808	0.16	—	—	—
9.6	103	9.615	0.16	129	9.615	0.16
19.2	51	19.231	0.16	64	19.231	0.16
38.4	25	38.462	0.16	32	37.879	-1.36
57.6	16	58.824	2.12	21	56.818	-1.35
115.2	8	111.11	-3.55	10	113.636	-1.36
250	3	250	0	4	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，无校验，1 位停止位组成，用 8、N、1 表示，它是系统上电的默认格式。数据位数、停止位数和奇偶校验由 UCR1 寄存器的 BNO、PRT、PREN 和 STOPS 设置。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，这两个引脚作为通用 I/O 或其它共用功能。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，使能控制、错误和状态标志位被复位即 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、停止位以及奇偶校验的选择

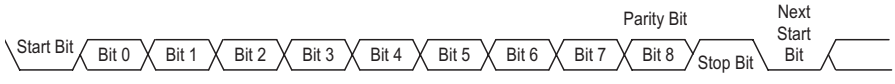
数据传输格式由数据长度、是否校验、校验类型、地址位以及停止位长度组成。它们都是由 UCR1 寄存器的各个位控制的。BNO 决定数据传输是 8 位还是 9 位；PRT 决定校验类型；PREN 决定是否选择奇偶校验；而 STOPS 决定选用 1 位还是 2 位停止位。下表列出了各种数据传输格式。地址位用来确定此帧是否为地址。停止位的长度和数据位的长度无关。

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 引脚返回 I/O 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述步骤。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 标志位置位时中断产生。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 引脚进入移位寄存器。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 寄存器中包含有效数据时，USR 寄存器中的 RXIF 位将会置位，且还有另外一帧数据可读。
- 若 RIE=1，数据从移位寄存器加载到 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，只侦测第一个停止位且此位必须为高，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

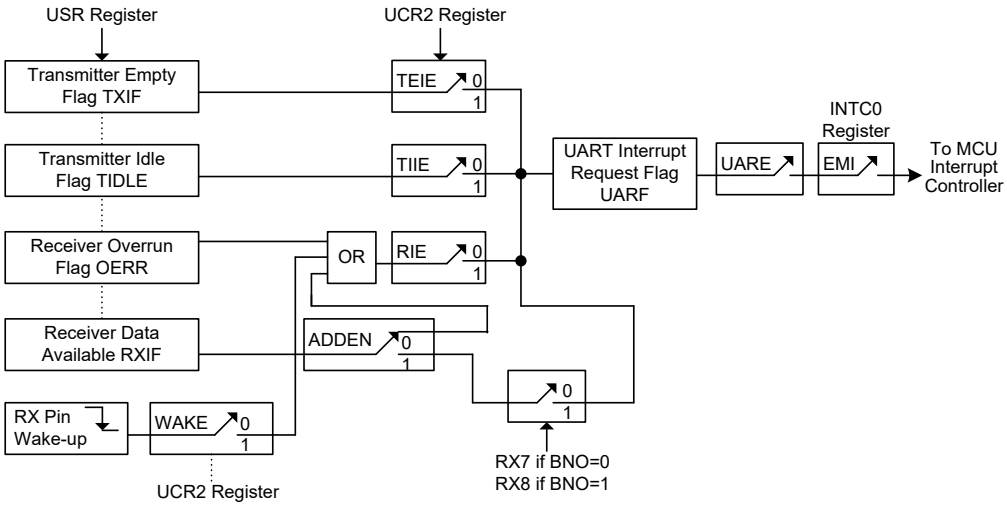
若接收到数据出现奇偶校验错误，USR 寄存器中只读标志 PERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。它同数据一起存储在缓冲器中，可被任何复位清除。注意，FERR 和 PERR 与相应的数据一起存储在对应的 USR 和 TXR_RXR 寄存器中，在读取数据之前必须先访问错误标志位。

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 UCR2 中的 WAKE 和 RIE 位被置位，RX 引脚上有下降沿可以唤醒单片机。应注意，RX 唤醒中断发生时，系统必须延时 t_{SST} 才能正常工作。

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



UART 中断结构

地址检测模式

置位 UCR2 寄存器中的 ADDEN 将启动地址检测模式。若此位为“1”，可产生接收数据有效中断，其请求标志位为 RXIF。若 ADDEN 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

当 f_{SYS} 关闭时，UART 模块将暂停运行，接至 UART 模块的时钟源将除能。若正在传送数据时关闭 f_{SYS}，发送将停止直到 UART 模块时钟源再次使能。同样地，当接收数据时 UART 进入暂停模式，数据接收也会停止。当 UART 电路进入暂停模式，USR、UCR1、UCR2 、接收 / 发送寄存器以及 BRG 寄存器都不会受到影响。建议在 MCU 进入暂停模式前先确保数据发送或接收已完成。

UART 功能中包括了 RX 引脚的唤醒功能，由 UCR2 寄存器中 WAKE 位控制。进入 IDLE0 或 SLEEP 模式前，若该标志位 WAKE 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 引脚的下降沿可将单片机从 IDLE0 或 SLEEP 模式唤醒。唤醒后系统需延时 t_{SST} 才能正常工作，在此期间，RX 引脚上的任何数据将被忽略。

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

中断

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

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~MFI2 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	—	—
INT 脚	INT0E	INT0F	—
AC 检测	ZXE	ZXF	—
SPWM 发生器	SFEE	SFEF	—
OCPn	OCPnF	OCPnE	n=0 或 1
OVP	OVPE	OVPF	—
A/D 转换器	ADE	ADF	—
UART	UARE	UARF	—
多功能	MFnE	MFnF	n=0~2
时基	TBnE	TBnF	n=0 或 1
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
TM	TnPE	TnPF	n=0~2
	TnAE	TnAF	

中断寄存器位命名规则

寄存器名称	位							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	ZXS1	ZXS0	INT0S1	INT0S0
INTC0	—	OCP0F	SFEF	ZXF	OCP0E	SFEE	ZXE	EMI
INTC1	ADF	INT0F	OVPF	OCP1F	ADE	INT0E	OVPE	OCP1E
INTC2	MF2F	MF1F	MF0F	UARF	MF2E	MF1E	MF0E	UARE
MFI0	T1AF	T1PF	T0AF	T0PF	T1AE	T1PE	T0AE	T0PE
MFI1	—	—	T2AF	T2PF	—	—	T2AE	T2PE
MFI2	TB1F	TB0F	DEF	LVF	TB1E	TB0E	DEE	LVE

中断寄存器列表

INTEG 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	ZXS1	ZXS0	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　**ZXS1, ZXS0**：交流过零检测中断边沿控制位
00：除能
01：上升沿
10：下降沿
11：双沿

Bit 1 ~ 0　**INT0S1, INT0S0**：外部 INT 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	—	OCP0F	SFEF	ZXF	OCP0E	SFEE	ZXE	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　**OCP0F**：过流保护 0 中断请求标志位
0：无请求
1：中断请求

Bit 5　**SFEF**：SPWM FIFO 为空中断请求标志位
0：无请求
1：中断请求

Bit 4　**ZXF**：交流过零检测中断请求标志位
0：无请求
1：中断请求

Bit 3　**OCP0E**：过流保护 0 中断控制位
0：除能
1：使能

Bit 2　**SFEE**：SPWM FIFO 为空中断控制位
0：除能
1：使能

Bit 1　**ZXE**：交流过零检测中断控制位
0：除能
1：使能

Bit 0　**EMI**：总中断控制位
0：除能
1：使能

INTC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ADF	INT0F	OVPF	OCP1F	ADE	INT0E	OVPE	OCP1E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **ADF**：A/D 转换器中断请求标志位
0：无请求
1：中断请求

Bit 6 **INT0F**：外部中断 0 请求标志位
0：无请求
1：中断请求

Bit 5 **OVPF**：过压保护中断请求标志位
0：无请求
1：中断请求

Bit 4 **OCP1F**：过流保护 1 中断请求标志位
0：无请求
1：中断请求

Bit 3 **ADE**：A/D 转换器中断控制位
0：除能
1：使能

Bit 2 **INT0E**：外部中断 0 控制位
0：除能
1：使能

Bit 1 **OVPE**：过压保护中断控制位
0：除能
1：使能

Bit 0 **OCP1E**：过流保护 1 中断控制位
0：除能
1：使能

INTC2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	MF2F	MF1F	MF0F	UARF	MF2E	MF1E	MF0E	UARE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **MF2F**：多功能中断 2 中断请求标志位
0：无请求
1：中断请求

Bit 6 **MF1F**：多功能中断 1 中断请求标志位
0：无请求
1：中断请求

Bit 5 **MF0F**：多功能中断 0 中断请求标志位
0：无请求
1：中断请求

Bit 4 **UARF**：UART 中断请求标志位
0：无请求
1：中断请求

Bit 3 **MF2E**：多功能中断 2 中断控制位
0：除能
1：使能

Bit 2 **MF1E**：多功能中断 1 中断控制位
0：除能
1：使能

Bit 1 **MF0E**：多功能中断 0 中断控制位
0：除能
1：使能

Bit 0 **UARE**：UART 中断控制位
0：除能
1：使能

MFIC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	T1AF	T1PF	T0AF	T0PF	T1AE	T1PE	T0AE	T0PE
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 **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 6 **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 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 **T1AE**：TM1 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 2 **T1PE**：TM1 比较器 P 匹配中断控制位
0：除能
1：使能

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	—	—	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

MFI2 寄存器

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

Bit 7　**TB1F**：时基 1 中断请求标志位
0：无请求
1：中断请求

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

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

Bit 4　**LVF**：低电压检测中断请求标志位
0：无请求
1：中断请求

Bit 3　**TB1E**：时基 1 中断控制位
0：除能
1：使能

Bit 2　**TB0E**：时基 0 中断控制位
0：除能
1：使能

Bit 1　**DEE**：EEPROM 写中断控制位
0：除能
1：使能

Bit 0　**LVE**：低电压检测中断控制位
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。

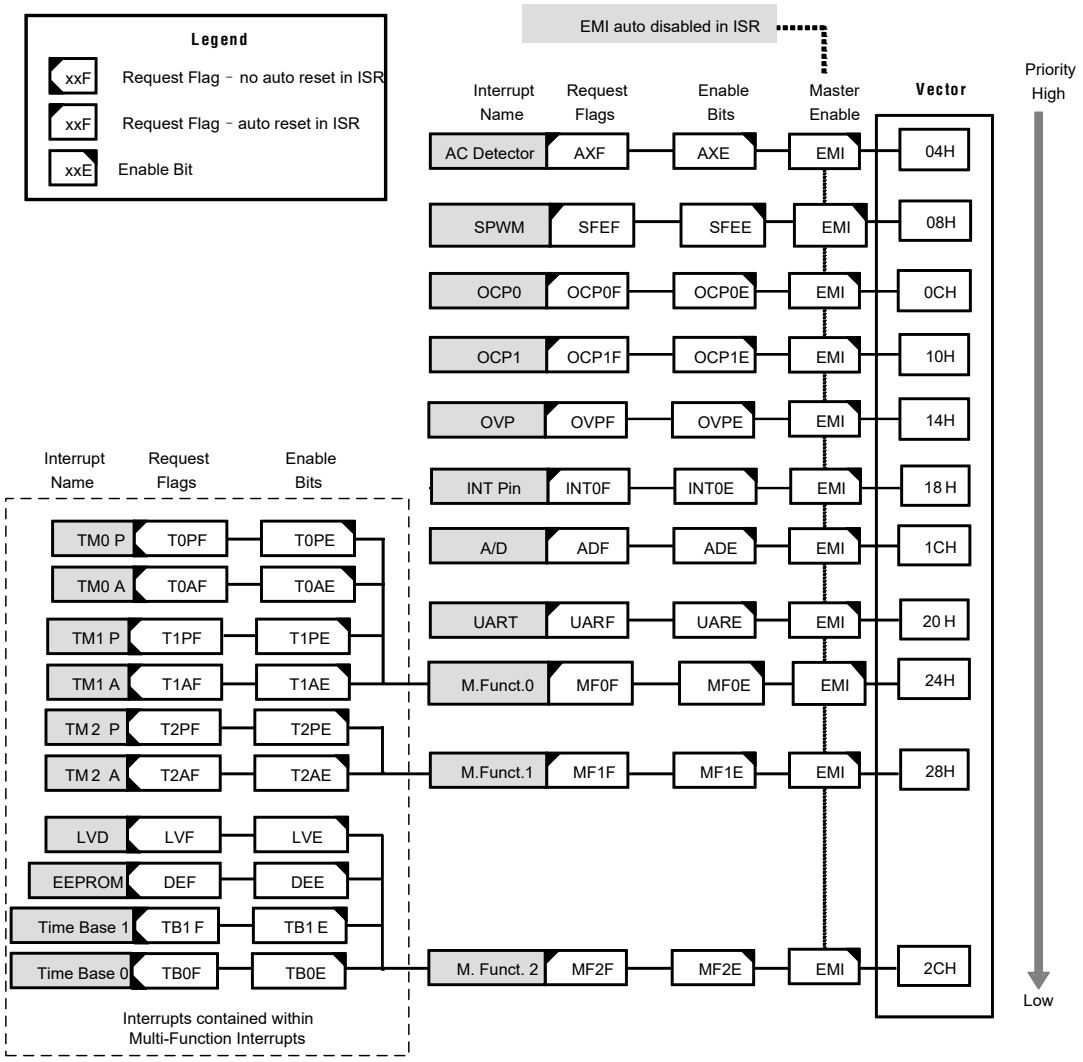
当中断发生时，下条指令的地址将被压入堆栈。相应的中断向量地址加载至 PC 中。系统将从此向量取下条指令。中断向量处通常为跳转指令，以跳转到相应的中断服务程序。中断服务程序必须以“RETI”指令返回至主程序，以继续执行原来的程序。

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 INT 引脚上的信号变化可控制外部中断。当触发沿选择位设置好触发类型，INT 引脚的状态发生变化，外部中断请求标志 INT0F 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 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 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。


Legend
xxF Request Flag – no auto reset in ISR
xxF Request Flag – auto reset in ISR
xxE Enable Bit
EMI auto disabled in ISR
Interrupt Name
Request Flags
Enable Bits
Master Enable
Vector
Priority
High
Low
AC Detector AXF AXE EMI 04H
SPWM SFEF SFEE EMI 08H
OCP0 OCP0F OCP0E EMI 0CH
OCP1 OCP1F OCP1E EMI 10H
OVP OVPF OVPE EMI 14H
INT Pin INT0F INT0E EMI 18 H
A/D ADF ADE EMI 1CH
UART UARF UARE EMI 20 H
M.Funct.0 MF0F MF0E EMI 24H
M.Funct.1 MF1F MF1E EMI 28H
M. Funct. 2 MF2F MF2E EMI 2CH
TM0 P T0PF T0PE
TM0 A T0AF T0AE
TM1 P T1PF T1PE
TM1 A T1AF T1AE
TM2 P T2PF T2PE
TM2 A T2AF T2AE
LVD LVF LVE
EEPROM DEF DEE
Time Base 1 TB1 F TB1 E
Time Base 0 TB0F TB0E
Interrupts contained within Multi-Function Interrupts

中断结构

多功能中断

该单片机中有多达三个多功能中断，与其它中断不同，它没有独立源，但由其它现有的中断源构成，即 TM 中断、LVD 中断、EEPROM 中断和时基中断。

当多功能中断中任何一种中断请求标志 MF0F~MF2F 被置位，多功能中断请求产生。当中断使能，堆栈未满，包括在多功能中断中的任意一个中断发生时，将调用多功能中断向量中的一个子程序。当响应中断服务子程序时，相关的多功能请求标志位会自动复位且 EMI 位会自动清零以除能其它中断。

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

A/D 转换器中断

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

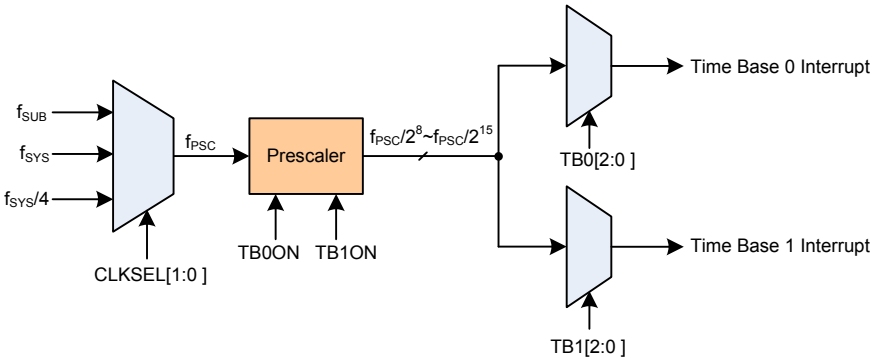
UART 中断

UART 模块中，发送器为空、发送器空闲、接收器数据有效、接收器溢出、地址检测匹配和 RX 引脚唤醒都会触发产生 UART 中断。当整体 UART 中断请求标志位被置位，即以上任何一种情况发生时，中断请求发生。当总中断使能位 EMI 和 UART 中断使能位 UARE 被置位，允许程序跳转到相应的中断向量地址。当中断使能，堆栈未满且以上任何一种情况发生时，将调用 UART 中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 UARF 会自动清零。EMI 位也会被清零以除能其它中断。而 UART 模块中的只读中断标志位仅在 UART 特定动作发生时才会自动被清除。更多关于 UART 中断的细节请参考 UART 章节。

时基中断

时基中断提供一个固定周期的中断信号，由各自的定时器功能产生溢出信号控制。当各自的中断请求标志 TB0F 或 TB1F 被置位时，中断请求发生。当总中断使能位 EMI、时基使能位 TB0E 或 TB1E 被置位，允许程序跳转到各自的中断向量地址。当中断使能，堆栈未满且时基溢出时，将调用它们各自的中断向量子程序。当响应中断服务子程序时，EMI 位会被清零以除能其它中断，相应的中断请求标志位 TB0F 或 TB1F 也会自动清零。

时基中断的目的是提供一个固定周期的中断信号，其时钟源 f_{PSC} 可通过 PSCR 寄存器的 CLKSEL1~CLKSEL0 位选择来自 $f_{SYS}/4$、f_{SYS} 或 f_{SUB}，之后再通过内置的分频器进一步分频，分频率由 TB0C 和 TB1C 寄存器的相关位进行选择以提供更少的时基中断周期。



时基中断

PSCR 寄存器

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

Bit 7 ~ 2　未定义，读为“0”

Bit 1 ~ 0　**CLKSEL1 ~ CLKSEL0**：分频器时钟源 f_{PSC} 选择位

00：f_{SYS}

01：$f_{SYS}/4$

1x：f_{SUB}

TB0C 寄存器

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

Bit 7　**TB0ON**：时基 0 使能控制位

0：除能

1：使能

Bit 6 ~ 3　未定义，读为“0”

Bit 2 ~ 0　**TB02 ~ TB00**：时基 0 溢出周期选择位

000：$2^8/f_{PSC}$

001：$2^9/f_{PSC}$

010：$2^{10}/f_{PSC}$

011：$2^{11}/f_{PSC}$

100：$2^{12}/f_{PSC}$

101：$2^{13}/f_{PSC}$

110：$2^{14}/f_{PSC}$

111：$2^{15}/f_{PSC}$

TB1C 寄存器

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

Bit 7　**TB1ON**：时基 1 使能控制位
0：除能
1：使能

Bit 6 ~ 3　未定义，读为“0”

Bit 2 ~ 0　**TB12 ~ TB10**：时基 1 溢出周期选择位
000：$2^8/f_{PSC}$
001：$2^9/f_{PSC}$
010：$2^{10}/f_{PSC}$
011：$2^{11}/f_{PSC}$
100：$2^{12}/f_{PSC}$
101：$2^{13}/f_{PSC}$
110：$2^{14}/f_{PSC}$
111：$2^{15}/f_{PSC}$

EEPROM 中断

EEPROM 中断属于多功能中断。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。

SPWM SFIFO 中断

当 SFIFO 中的第四笔数据被读取后会发出 SFIFO 为空信号，产生中断以通知单片机写入下四笔数据 , 此时中断请求标志位 SFEF 会被置位。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和 SFIFO 空中断使能位 SFEE 需先被置位。当中断使能，堆栈未满且 SFIFO 为空情况发生时，可跳转至相关中断向量子程序中执行。当 SFIFO 为空中断被响应，相应的中断请求标志位 SFEF 会自动清零，EMI 也会被自动清零以除能其它中断。

交流过零检测中断

当交流检测电路检测到输入的信号电压值小于内部 8-bit DAC 设置的指定值时，即比较器的输出端状态发生由低到高的转换时，会触发 AC 过零中断。此时相应的中断请求标志位 ZXF 会置位。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和交流检测中断使能位 ZXE 需先被置位。当中断使能，堆栈未满且交流检测过零情况发生时，可跳转至相关中断向量子程序中执行。当交流检测过零中断被响应，相应的中断请求标志位 ZXF 会自动清零，EMI 也会被自动清零以除能其它中断。

过流保护中断

当检测到过电流情况时，过电流保护中断请求标志位 OCPnF 被置位，OCPn 中断请求发生。当总中断使能位 EMI 和 OCPn 中断使能位 OCPnE 被置位，允许程序跳转到相应的中断向量地址。当中断使能，堆栈未满且过电流情况发生时，将调用 OCPn 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 OCPnF 会自动清零。EMI 位也会被清零以除能其它中断。

过压保护中断

当检测到过电压情况时，过电压保护中断请求标志位 OVPF 被置位，OVP 中断请求发生。当总中断使能位 EMI 和过电压保护中断使能位被置位，允许程序跳转到相应的中断向量地址。当中断使能，堆栈未满且过电压情况发生时，将调用 OVP 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 OVPF 会自动清零。EMI 位也会被清零以除能其它中断。

TM 中断

每个简易型 TM 或标准型 TM 各有两个中断。所有的 TM 中断也属于多功能中断。每个 TM 都有两个中断请求标志位 TnPF、TnAF 及两个使能位 TnPE、TnAE。当 TMn 比较器 P 或 A 匹配情况发生时，任意 TMn 中断请求标志被置位，TMn 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 TMn 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 TMn 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 TMn 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 TMn 中断请求标志需在应用程序中手动清除。

中断唤醒功能

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

编程注意事项

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

多功能中断中所含中断相应程序执行时，多功能中断请求标志 MF0F~MF2F 可以自动清零，但各自的请求标志需在应用程序中手动清除。

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

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

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

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

低电压检测 – LVD

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

LVD 寄存器

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

LVDC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	VBGEN	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7 ~ 6　未定义，读为“0”

Bit 5　**LVDO**：LVD 输出标志位
0：未检测到低电压
1：检测到低电压

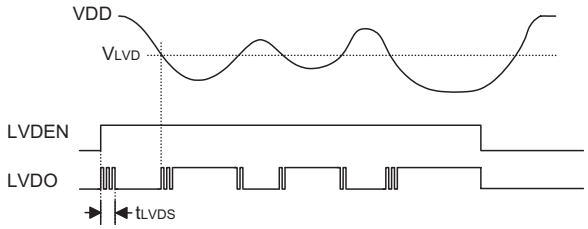
Bit 4　**LVDEN**：低电压检测控制位
0：除能
1：使能

Bit 3　**VBGEN**：V_{BG} 控制位
0：除能
1：使能

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

LVD 操作

低电压检测功能是通过比较电源电压 V_{DD} 与 LVDC 寄存器设置的预置电压值来进行检测工作。其可设置的电压范围为 2.0V~4.0V。当电源电压 V_{DD} 低于预置电压值时，LVDO 位被置为高，表明低电压产生。当单片机进入休眠模式时，即使 LVDEN 位为高，低电压检测器也会被除能。低电压检测器使能后，读取 LVDO 位前，电路稳定需要一定的延时 t_{LVDS}。注意，V_{DD} 电压可能上升或下降比较缓慢，在 V_{LVD} 电压值附近时，LVDO 位可能有多种变化。

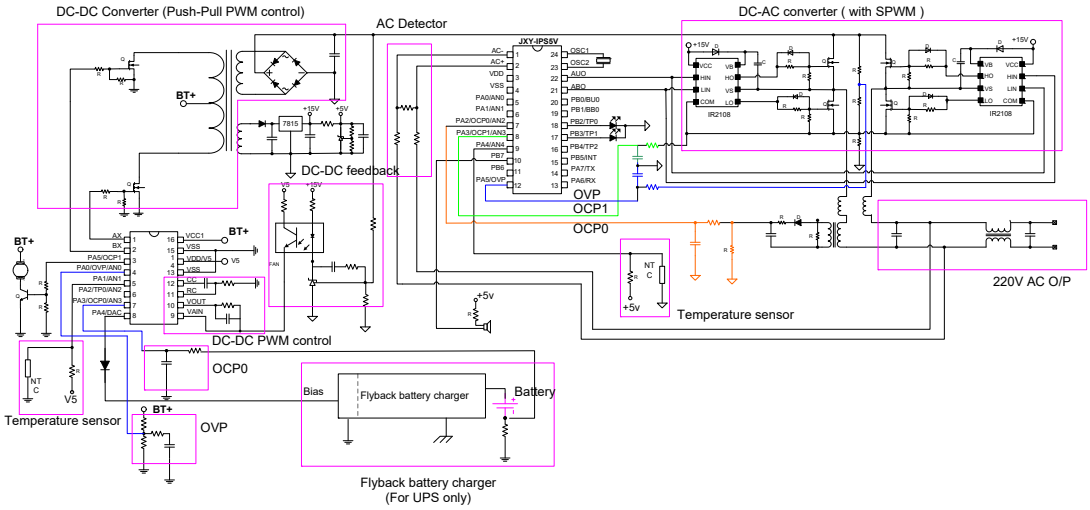


LVD 操作

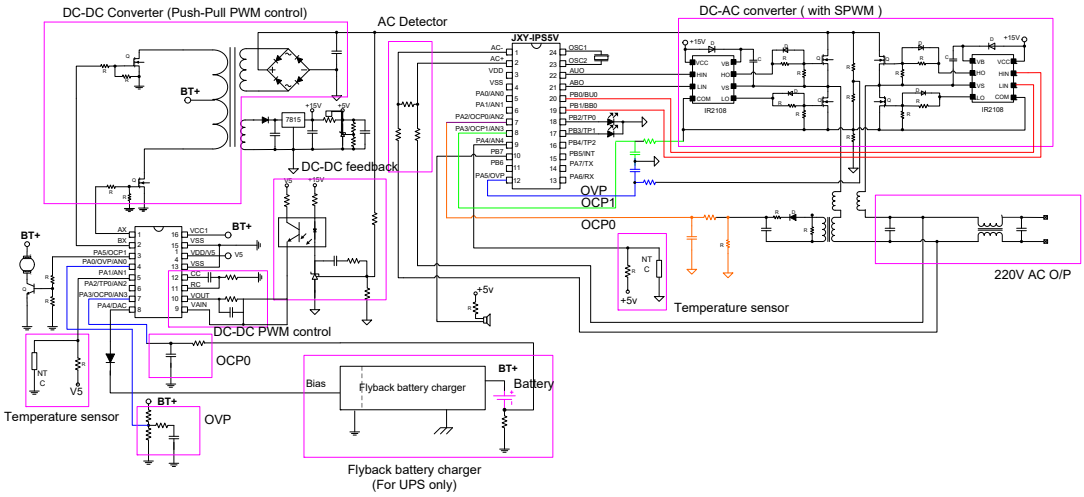
低电压检测器也有自己的中断功能，也是属于多功能中断的一种，它是除了轮询 LVDO 位之外的另一种检测低电压的方法。中断条件产生置位 LVDO 并延时 t_{LVD} 后，中断产生。若 LVDEN 位为高，当单片机掉电时低电压检测器保持有效状态。此种情况下，若 V_{DD} 降至小于 LVD 预置电压值时，中断请求标志位 LVF 将被置位，中断产生，单片机将从休眠或空闲模式中被唤醒。若不要求低电压检测的唤醒功能使能，在单片机进入休眠或空闲模式前应将 LVF 标志置为高。

应用电路

双极性控制电路图



单极性控制电路图



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。在该单片机中，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，使用几种 MOV 的指令，数据不但可以从寄存器转移至累加器 (反之亦然)，而且能够直接移动立即数到累加器。数据传送最重要的应用之一是从输入端口接收数据或传送数据到输出端口。

算术运算

算术运算和数据处理是大部分单片机应用所必需具备的能力，在该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 全都包含在该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该单片机的特性之一。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，该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

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

注：1. 对跳转指令而言，如果比较的结果牵涉到跳转即需多达 3 个周期，如果没有发生跳转，则只需一个周期。

2. 任何指令若要改变 PCL 的内容将需要 2 个周期来执行。

3. 对于“CLR WDT”指令而言，TO 和 PDF 标志位也许会受执行结果影响，“CLR WDT”被执行后，TO 和 PDF 标志位会被清除，否则 TO 和 PDF 标志位保持不变。

扩展指令集

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

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

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

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

2. 任何扩展指令若要改变 PCL 的内容将需要 3 个周期来执行。

指令定义

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

RR [m] Rotate Data Memory right

指令说明 将指定数据存储器的内容循环右移 1 位且第 0 位移到第 7 位。

功能表示 [m].i ← [m].(i+1) (i=0~6)
[m].7 ← [m].0

影响标志位 无

RRA [m] Rotate Data Memory right with result in ACC

指令说明 将指定数据存储器的内容循环右移 1 位，第 0 位移到第 7 位，移位结果存放到累加器，而指定数据存储器的内容保持不变。

功能表示 ACC.i ← [m].(i+1) (i=0~6)
ACC.7 ← [m].0

影响标志位 无

RRC [m] Rotate Data Memory right through Carry

指令说明 将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位。

功能表示 [m].i ← [m].(i+1) (i=0~6)
[m].7 ← C
C ← [m].0

影响标志位 C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

指令说明 将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位 , 移位结果送回累加器，但是指定数据寄存器的内容保持不变。

功能表示 ACC.i ← [m].(i+1) (i=0~6)
ACC.7 ← C
C ← [m].0

影响标志位 C

SBC A, [m] Subtract Data Memory from ACC with Carry

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

功能表示 ACC ← ACC – [m] – $\overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

SBC A, x	Subtract immediate data from ACC with Carry
指令说明	将累加器减去立即数以及进位标志，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	ACC ← ACC – [m] – $\overline{C}$
影响标志位	OV、Z、AC、C、SC、CZ
SBCM A, [m]	Subtract Data Memory from ACC with Carry and result in Data Memory
指令说明	将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	[m] ← ACC – [m] – $\overline{C}$
影响标志位	OV、Z、AC、C、SC、CZ
SDZ [m]	Skip if Decrement Data Memory is 0
指令说明	将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	[m] ← [m] -1，如果 [m]=0 跳过下一条指令执行
影响标志位	无
SDZA [m]	Decrement data memory and place result in ACC,skip if 0
指令说明	将指定数据存储器内容减 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果将存放到累加器，但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	ACC ← [m]-1，如果 ACC=0 跳过下一条指令执行
影响标志位	无
SET [m]	Set Data Memory
指令说明	将指定数据存储器的每一位设置为 1。
功能表示	[m] ← FFH
影响标志位	无

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

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

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

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

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

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

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

SUB A, x	Subtract immediate Data from ACC
指令说明	将累加器的内容减去立即数，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	ACC ← ACC – x
影响标志位	OV、Z、AC、C、SC、CZ

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

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

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

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

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

功能表示 ACC ← [m]，如果 [m]=0，跳过下一条指令执行

影响标志位 无

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

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

功能表示 如果 [m].i=0，跳过下一条指令执行

影响标志位 无

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

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

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

影响标志位 无

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

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

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

影响标志位 无

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

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

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

影响标志位 无

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

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

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

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

扩展指令定义

扩展指令被用来直接存取存储在任何数据存储器 Sector 中的数据。

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

功能表示 [m] ← 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

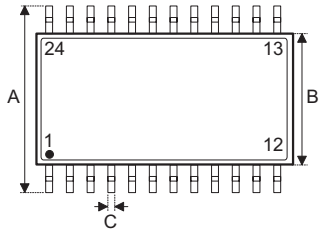
LITABRDL [m]	Increment table pointer low byte first and read table(last page) to TBLH and data memory
指令说明	将自加表格指针 TBLP 所指的程序代码低字节（最后一页）移至指定的数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无

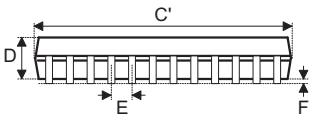
LXOR A, [m]	Logical XOR Data Memory to ACC
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。
功能表示	ACC ← ACC“XOR”[m]
影响标志位	Z

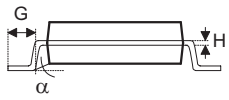
LXORM A, [m]	Logical XOR ACC to Data Memory
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果放到数据存储器。
功能表示	[m] ← ACC“XOR”[m]
影响标志位	Z

封装信息

24-pin SSOP (150mil) 外形尺寸







符号	尺寸 (单位：inch)		
	最小	正常	最大
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.341 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小	正常	最大
A	—	6.0 BSC	—
B	—	3.9 BSC	—
C	0.20	—	0.30
C’	0.20	—	0.30
D	—	—	1.75
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°