NEWWAVE

电磁炉 Flash 单片机

NW2018

版本 : V1.00　日期 : 2018-05-07

WWW.NEWWAVE-SZ.COM

目录

特性

CPU 特性

- 工作电压：
  - ♦ f_{SYS} = 16MHz：3.15V~5.5V
- V_{DD} = 5V，系统时钟为 16MHz 时，指令周期为 0.25μs
- 提供暂停和唤醒功能，以降低功耗
- 振荡器类型：
  - ♦ 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
- 所有指令都可在 1~3 个指令周期内完成
- 查表指令
- 115 条指令
- 8 层堆栈
- 位操作指令

周边特性

- Flash 程序存储：4K×16
- RAM 数据存储：256×8
- True EEPROM 存储器：64×8
- 看门狗定时器功能
- 17 个双向 I/O 口
- 2 个引脚与外部中断口共用
- 多个定时器模块用于时间测量、捕捉输入、比较匹配输出、PWM 输出及单脉冲输出
- 双时基功能可提供固定时间的中断信号
- 9 个外部通道 12-bit 分辨精度的 A/D 转换器
- 5 个比较器功能，其中 4 个带有 8-bit D/A 转换器
- 1 个运算放大器功能 – OPA
- 9-bit 可编程脉冲发生器 – PPG
  - ♦ 支持脉宽限制
  - ♦ 2 个 PPG 预载寄存器
  - ♦ 支持不可重复触发控制 – 来自 8-bit 定时 / 计数器
  - ♦ 支持输出高电平或低电平有效
  - ♦ 自动记录同步信号次数
- 1 个 8-bit 定时 / 计数器，可用于 PPG 不可重复触发控制
- 3 组恒定电流输出
- 外围时钟输出
- 单个 I^2C 接口

- 全双工异步通信接口模块 – UART
- 低电压复位功能
- 低电压检测功能
- Flash 程序存储器烧录可达 100,000 次
- Flash 程序存储器数据可保存 10 年以上
- True EEPROM 数据存储器烧录可达 1,000,000 次
- True EEPROM 数据存储器数据可保存 10 年以上
- 封装类型：16-pin NSOP、20-pin DIP/SOP

概述

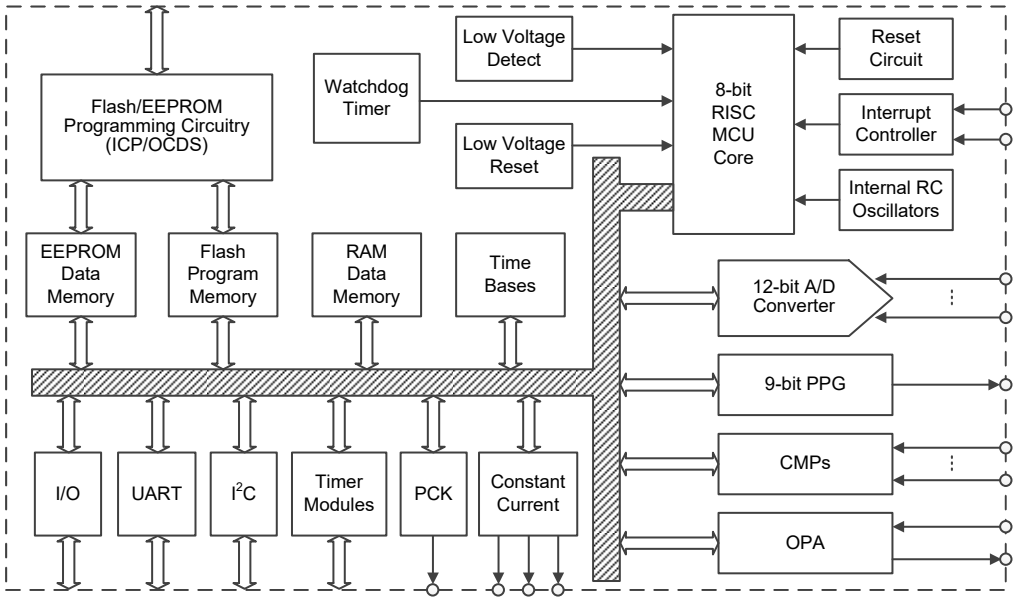
NW2018 是一款 8 位具有高性能精简指令集的 Flash 单片机。该单片机具有一系列功能和特性，其 Flash 存储器可多次编程的特性给用户提供了极大的方便。存储器方面，还包含了一个 RAM 数据存储器和一个可用于存储序列号、校准数据等非易失性数据的 True EEPROM 存储器。

在模拟特性方面，该单片机包含一个多通道 12 位 A/D 转换器、一个 OPA 和多个比较器功能。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 I^2C 和 UART 接口功能，为设计者提供了易与外部硬件通信的接口。内部看门狗定时器、低电压复位和低电压检测等内部保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地运行。

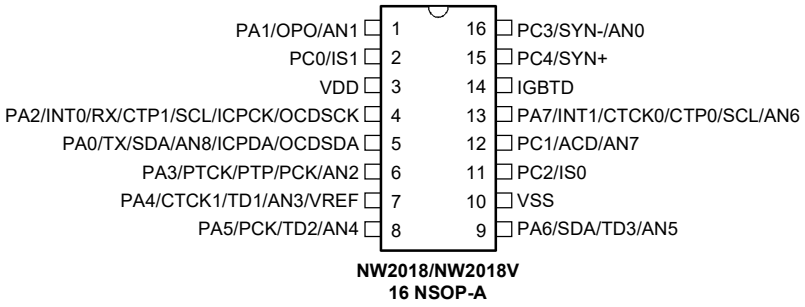
该单片机提供了内部高速和低速振荡器功能选项，且内建完整的系统振荡器，无需外接元件。其在不同工作模式之间动态切换的能力，为用户提供了一个优化单片机操作和减少功耗的手段。

外加时基功能、I/O 使用灵活、一个可编程脉冲发生器 PPG、恒定电流输出以及提供外围时钟输出等其它特性，使这款单片机可以广泛应用于各种产品中，例如单片机控制的电源管理产品等方面。

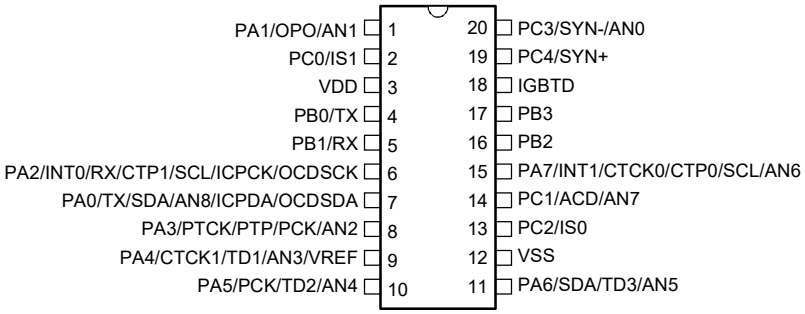
方框图



引脚图



NW2018/NW2018V
16 NSOP-A



NW2018/NW2018V
20 DIP-A/NSOP-A

注：1. 若共用脚同时有多种输出，所需引脚共用功能通过相应的软件控制位决定。
2. NW2018V 是 NW2018 的 EV 芯片，OCDSCK 和 OCDSDA 引脚仅存在于 OCDS EV 芯片。
3. 在较小封装中可能含有未引出的引脚，需合理设置其状态以避免输入浮空造成额外耗电，详见“待机电流注意事项”和“输入 / 输出端口”章节。

引脚说明

除了电源引脚和 PPG 输出引脚外，该单片机的所有引脚都以它们的端口名称进行标注，例如 PA0、PA1 等，用于描述这些引脚的数字输入 / 输出功能。然而，这些引脚也与其它功能共用，如模数转换器、定时器模块等。每个引脚的功能如下表所述，而引脚配置的详细内容见规格书其它章节。

引脚说明表格所列情况是针对最多引脚的封装，并非所有引脚都适用于小封装。

引脚名称	功能	OPT	I/T	O/T	说明
PA0/TX/SDA/AN8/ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	TX	PAS0	—	CMOS	UART TX 串行数据输出
	SDA	PAS0 CTRL	ST	NMOS	I^2C 数据线
	AN8	PAS0	AN	—	ADC 外部模拟输入通道
	ICPDA	—	ST	CMOS	ICP 数据 / 地址
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址，仅用于 EV 芯片
PA1/OPO/AN1	PA1	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	OPO	PAS0	—	AN	运算放大器输出脚
	AN1	PAS0	AN	—	ADC 外部模拟输入通道
PA2/INT0/RX CTP1/SCL/ICPCK/OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	INT0	PAS0 INTEG INTC0	ST	—	外部中断输入 0
	RX	PAS0 CTRL	ST	—	UART RX 串行数据输入
	CTP1	PAS0	—	CMOS	CTM1 输出
	SCL	PAS0 CTRL	ST	NMOS	I^2C 时钟线
	ICPCK	—	ST	—	ICP 时钟引脚
	OCDSCK	—	ST	—	OCDS 时钟引脚，仅用于 EV 芯片
PA3/PTCK/PTP/PCK/AN2	PA3	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	PTCK	PAS0	ST	—	PTM 时钟输入
	PTP	PAS0	—	CMOS	PTM 输出
	PCK	PAS0	—	CMOS	外围时钟输出
	AN2	PAS0	AN	—	ADC 外部模拟输入通道

引脚名称	功能	OPT	I/T	O/T	说明
PA4/ CTCK1/TD1/ AN3/VREF	PA4	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	CTCK1	PAS1	ST	—	CTM1 时钟输入
	TD1	PAS1	—	AN	恒定电流输出
	AN3	PAS1	AN	—	ADC 外部模拟输入通道
	VREF	PAS1	AN	—	ADC & DAC 参考电压输入
PA5/PCK/TD2/ AN4	PA5	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	PCK	PAS1	—	CMOS	外围时钟输出
	TD2	PAS1	—	AN	恒定电流输出
	AN4	PAS1	AN	—	ADC 外部模拟输入通道
PA6/SDA/TD3/ AN5	PA6	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SDA	PAS1 CTRL	ST	NMOS	I^2C 数据线
	TD3	PAS1	—	AN	恒定电流输出
	AN5	PAS1	AN	—	ADC 外部模拟输入通道
PA7/INT1/ CTCK0/CTP0/ SCL/AN6	PA7	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	INT1	PAS1 INTEG MFI3	ST	—	外部中断输入 1
	CTCK0	PAS1	ST	—	CTM0 时钟输入
	CTP0	PAS1	—	CMOS	CTM0 输出
	SCL	PAS1 CTRL	ST	NMOS	I^2C 时钟线
	AN6	PAS1	AN	—	ADC 外部模拟输入通道
PB0/TX	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	TX	PBS0	—	CMOS	UART TX 串行数据输出
PB1/RX	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	RX	PBS0 CTRL	ST	—	UART RX 串行数据输入
PB2~PB3	PB2~PB3	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻

引脚名称	功能	OPT	I/T	O/T	说明
PC0/IS1	PC0	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	IS1	PCS0	AN	—	运算放大器输入
PC1/ACD/AN7	PC1	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	ACD	PCS0	AN	—	比较器 2 和比较器 3 同相输入
	AN7	PCS0	AN	—	ADC 外部模拟输入通道
PC2/IS0	PC2	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	IS0	PCS0	AN	—	比较器 4 同相输入，由软件选择
PC3/SYN-/AN0	PC3	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SYN-	PCS0	AN	—	比较器 0 反相输入
	AN0	PCS0	AN	—	ADC 外部模拟输入通道
PC4/SYN+	PC4	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SYN+	PCS1	AN	—	比较器 0 和比较器 1 同相输入
IGBTD	IGBTD	—	—	PMOS/ NMOS	可编程脉冲发生器输出脚 PPG 输出脚可通过软件选项选择输出为高电平有效 (PMOS 输出) 或低电平有效 (NMOS 输出)。
VDD	VDD	—	PWR	—	正电源电压
VSS	VSS	—	PWR	—	负电源电压，接地

注：I/T：输入类型；
OPT：通过寄存器选项来配置；
ST：施密特触发输入；
NMOS：NMOS 输出；
AN：模拟信号
O/T：输出类型；
PWR：电源；
CMOS：CMOS 输出；
PMOS：PMOS 输出；

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°C ~150°C
工作温度 ..... -40°C ~85°C
I_{OH} 总电流 ..... -80mA
I_{OL} 总电流..... 80mA
总功耗 500mW

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

直流电气特性

Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压 (HIRC)	—	$f_{SYS} = f_{HIRC} = 16MHz$	3.15	—	5.5	V
	工作电压 (LIRC)	—	$f_{SYS} = f_{LIRC} = 32kHz$	3.15	—	5.5	
I_{DD}	工作电流 (HIRC)	5V	无负载，所有外围功能 off，$f_{SYS} = f_{HIRC}/2 = 8MHz$	—	2.0	3.0	mA
		5V	无负载，所有外围功能 off，$f_{SYS} = f_{HIRC} = 16MHz$	—	3.2	4.8	mA
	工作电流 (LIRC)	5V	无负载，所有外围功能 off，$f_{SYS} = f_{LIRC} = 32kHz$	—	12	24	μA
I_{STB}	待机电流 (休眠模式)	5V	无负载，所有外围功能 off，WDT on	—	3	5	μA
	待机电流 (空闲模式 0)	5V	无负载，所有外围功能 off，f_{SUB} on	—	5	10	μA
	待机电流 (空闲模式 1)	5V	无负载，所有外围功能 off，f_{SUB} on，$f_{SYS} = f_{HIRC} = 16MHz$	—	1.4	2	mA
V_{IL}	I/O 口低电平输入电压	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	I/O 口高电平输入电压	5V	—	3.5	—	5	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	I/O 口灌电流	5V	$V_{OL} = 0.1V_{DD}$	33	66	—	mA
I_{OH}	I/O 口源电流	5V	$V_{OH} = 0.9V_{DD}$	-7	-14	—	mA
R_{PH}	I/O 口上拉电阻	5V	—	10	30	50	kΩ
I_{LEAK}	输入漏电流	5V	$V_{IN} = V_{DD}$ 或 $V_{IN} = V_{SS}$	—	—	±1	μA
V_{DR}	RAM 数据保持电压	—	—	1	—	—	V

交流电气特性

Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{SYS}	系统时钟 (HIRC)	3.15V~5.5V	$f_{SYS} = f_{HIRC} = 16MHz$	—	16	—	MHz
	系统时钟 (LIRC)	3.15V~5.5V	$f_{SYS} = f_{LIRC} = 32kHz$	—	32	—	kHz
f_{HIRC}	内部高速 RC 振荡器时钟 (HIRC)	5V	Ta = 25°C	-1.0%	16	+1.0%	MHz
		3.3V~5.5V	Ta = 25°C	-2.5%	16	+2.5%	
		5V	Ta = -40°C ~ 85°C	-2.0%	16	+2.0%	
		3.3V~5.5V	Ta = -40°C ~ 85°C	-3.0%	16	+3.0%	
f_{LIRC}	内部低速 RC 振荡器时钟 (LIRC)	5V	Ta = 25°C	-10%	32	+10%	kHz
		5V±0.5V	Ta = -40°C ~ 85°C	-40%	32	+40%	
		3.15V~5.5V	Ta = -40°C ~ 85°C	-50%	32	+60%	
t_{INT}	外部中断最小脉宽	—	—	10	—	—	μs

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
t_{TCK}	xTMn xTCKn 最小输入脉宽	—	—	0.3	—	—	μs
t_{SRESET}	软件复位最小脉宽	—	—	45	90	120	μs
t_{EERD}	EEPROM 读周期	—	—	—	—	4	t_{SYS}
t_{EEWR}	EEPROM 写周期	—	—	—	4	6	ms
t_{RSTD}	系统复位延迟时间 (上电复位，LVR 硬件复位，LVR 软件复位，WDT 软件复位)	—	—	25	50	100	ms
	系统复位延迟时间 (WDT 溢出硬件复位)	—	—	8.3	16.7	33.3	ms
t_{SST}	系统启动时间 (从 HALT 中唤醒，HALT 状态下 f_{SYS} off)	—	$f_{SYS} = f_{HIRC}$~$f_{HIRC}/64$	16	—	—	t_{HIRC}
		—	$f_{SYS} = f_{LIRC}$	2	—	—	t_{SYS}
	系统启动时间 (低速模式 ↔ 正常模式)	—	f_{HIRC} off → on (HIRCF = 1)	16	—	—	t_{HIRC}
	系统启动时间 (从 HALT 中唤醒，HALT 状态下 f_{SYS} on)	—	$f_{SYS} = f_H$~$f_H/64$, $f_H = f_{HIRC}$	2	—	—	t_{HIRC}
		—	$f_{SYS} = f_{LIRC}$	2	—	—	t_{SYS}
	系统启动时间 (WDT 溢出硬件复位)	—	—	0	—	—	t_{SYS}

A/D 转换器电气特性

Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	3.15	—	5.5	V
V_{ADI}	输入电压	—	—	0	—	V_{REF}	V
V_{REF}	参考电压	—	—	—	V_{DD}	—	V
DNL	非线性微分误差	5V	$V_{REF} = V_{DD}$，t_{ADCK} = 0.5μs	—	—	±3	LSB
		5V	$V_{REF} = V_{DD}$，t_{ADCK} = 8μs				
INL	非线性积分误差	5V	$V_{REF} = V_{DD}$，t_{ADCK} = 0.5μs	—	—	±4	LSB
		5V	$V_{REF} = V_{DD}$，t_{ADCK} = 8μs				
I_{ADC}	A/D 转换器使能的额外电流	5V	无负载，t_{ADCK} = 0.5μs	—	1.5	3	mA
t_{ADCK}	A/D 转换时钟周期	—	—	0.5	—	10	μs
t_{ON2ST}	A/D 转换器 On-to-Start 时间	—	—	4	—	—	μs
t_{ADS}	A/D 转换采样时间	—	—	—	4	—	t_{ADCK}
t_{ADC}	A/D 转换时间 (包括采样和保持时间)	—	—	—	16	—	t_{ADCK}

LVD & LVR 电气特性

Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	3.15	—	5.5	V
V_{LVR}	低电压复位电压	—	LVR 使能，电压选择 3.15V	-5%	3.15	+5%	V
V_{LVD}	低电压检测电压	—	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%	
		—	LVD 使能，电压选择 4.0V	-5%	4.0	+5%	
$I_{LVRLVDBG}$	工作电流	5V	LVD 使能，LVR 使能，VBGEN = 0	—	20	25	μA
		5V	LVD 使能，LVR 使能，VBGEN = 1	—	180	200	μA
I_{LVR}	LVR 使能的额外电流	—	LVD 除能，VBGEN = 0	—	—	20	μA
I_{LVD}	LVD 使能的额外电流	—	LVR 除能，VBGEN = 0	—	—	20	μA
t_{LVDS}	LVDO 稳定时间	—	LVR 使能时，VBGEN = 0，LVD off → on	—	—	15	μs
t_{LVR}	最小低电压复位时间	—	—	120	240	480	μs
t_{LVD}	最小低电压中断时间	—	—	60	120	240	μs

参考电压电气特性

Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{BG}	Bandgap 参考电压	—	—	-5%	1.04	+5%	V
I_{BG}	使用带缓冲器 Bandgap 参考源的额外电流	—	LVR 除能，LVD 除能	—	—	200	μA
t_{BGS}	V_{BG} 开启稳定时间	—	无负载	—	—	150	μs

注：V_{BG} 电压用于 A/D 转换器 PGA 输入。

运算放大器电气特性

Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	3.15	—	5.5	V
I_Q	静态电流	5V	无负载，V_{IN} = 1/2 V_{CM}	—	200	320	μA
I_{OPA}	OPA 使能的额外电流	5V	无负载	—	300	450	μA
V_{OS}	输入失调电压	5V	未校准 (AOF[5:0]=100000B)	-15	—	15	mV
		5V	校准后	-2	—	2	mV
V_{CM}	共模电压范围	5V	—	V_{SS}	—	V_{DD}-1.4	V
Gain	OPA 增益误差	5V	所有增益	-5	—	5	%
A_{OL}	开环增益	5V	—	60	80	—	dB
SR	转换速率	5V	无负载	0.6	1.8	—	V/μs
GBW	增益带宽	5V	R_{LOAD} = 1MΩ, C_{LOAD} = 100pF	1500	2200	—	kHz
PSRR	电源抑制比	5V	—	60	80	—	dB
CMRR	共模抑制比	5V	—	60	80	—	dB

注：IS1 引脚输入到 OPA 的电压不能小于 -300mV，以避免 PAD 上的 ESD 保护二极管影响 OPA 的增益。

比较器电气特性

Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	3.15	—	5.5	V
I_{CMP}	比较器使能的额外电流	5V	—	—	200	300	μA
V_{OS}	输入失调电压	5V	未校准 (CnOF[4:0] = 10000B)	-10	—	10	mV
		5V	校准后	-4	—	4	mV
V_{CM}	共模电压范围	—	—	V_{SS}	—	V_{DD}-1.4	V
A_{OL}	开环增益	5V	—	60	80	—	dB
V_{HYS}	迟滞宽度	5V	—	20	40	60	mV
t_{RP}	响应时间	5V	10mV 偏置，C_{LOAD} = 3pF	—	—	2	μs
		5V	60mV 偏置，C_{LOAD} = 3pF	—	—	1.5	μs

8-bit DAC 电气特性

Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
DNL	非线性微分误差	5V	DAC $V_{REF} = V_{DD}$	—	—	±1	LSB
INL	非线性积分误差	5V	DAC $V_{REF} = V_{DD}$	—	—	±1.5	LSB
I_{DAC}	DAC 使能的额外功耗	5V	8-bit DAC 输出 0V	—	40	80	μA
		5V	8-bit DAC 输出 1.8V	—	400	800	μA
		5V	8-bit DAC 输出 3.6V	—	360	720	μA

恒定电流电气特性

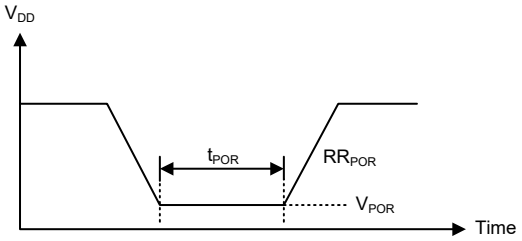
Ta = 25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	2.2	—	5.5	V
I_{CC}	输出电流	5V	TDCCS[2:0] = 000	-15%	2.1	+15%	μA
		5V	TDCCS[2:0] = 100	-15%	9.0	+15%	μA
		5V	TDCCS[2:0] = 001	-5%	20	+5%	μA
		5V	TDCCS[2:0] = 101	-5%	80	+5%	μA
		5V	TDCCS[2:0] = 010	-5%	200	+5%	μA
		5V	TDCCS[2:0] = 110	-5%	800	+5%	μA
		5V	TDCCS[2:0] = 011 或 111	-5%	2	+5%	mA
R_{TD}	恒定电流内部模拟开关启动电阻	5V	—	80	160	320	Ω
t_{ST}	建立时间	5V	TDCCS[2:0] = 011 或 111	—	2	4	μs
		5V	TDCCS[2:0] = 110	—	10	20	μs
		5V	TDCCS[2:0] = 010	—	30	60	μs
		5V	TDCCS[2:0] = 101	—	100	200	μs
		5V	TDCCS[2:0] = 001	—	300	600	μs
		5V	TDCCS[2:0] = 100	—	1000	2000	μs
		5V	TDCCS[2:0] = 000	—	3000	6000	μs

上电复位特性

Ta = 25°C

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

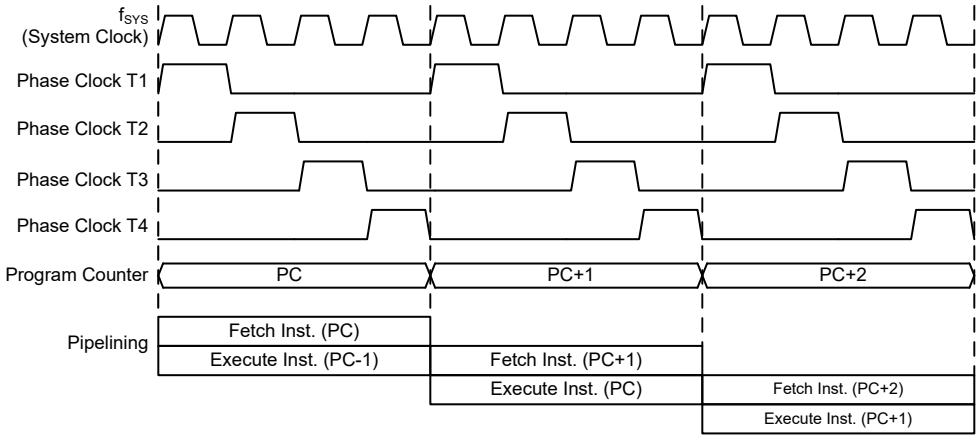


系统结构

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

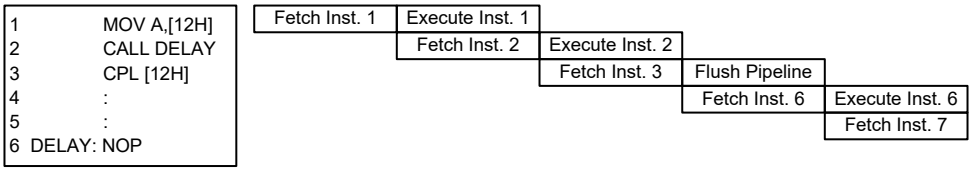
时序和流水线结构

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



系统时序和流水线

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



指令捕捉

程序计数器

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

当执行的指令要求跳转到不连续的地址时，如跳转指令、子程序调用、中断或复位等，单片机通过加载所需要的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。

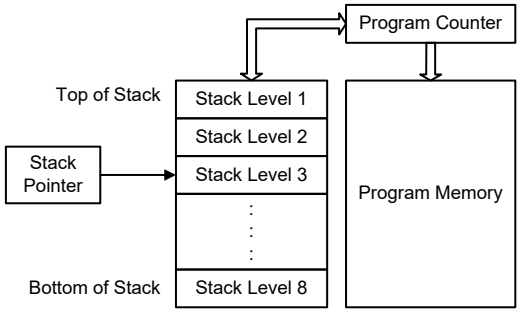
程序计数器	
程序计数器高字节	PCL 寄存器
PC11~PC8	PCL7~PCL0

程序计数器

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



如果堆栈已满，且有非屏蔽的中断发生，中断请求标志会被置位，但中断响应将被禁止。当堆栈指针减少 (执行 RET 或 RETI)，中断将被响应。这个特性提供程序设计者简单的方法来预防堆栈溢出。然而即使堆栈已满，CALL 指令仍

然可以被执行，而造成堆栈溢出。使用时应避免堆栈溢出的情况发生，因为这可能导致不可预期的程序分支指令执行错误。

若堆栈溢出，则首个存入堆栈的程序计数器数据将会丢失。

算术逻辑单元 – ALU

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

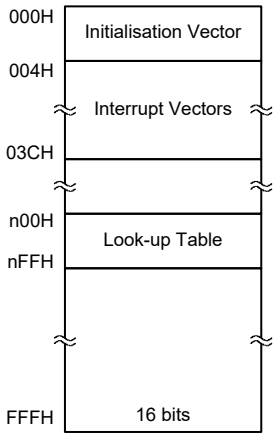
- 算术运算：
  ADD，ADDM，ADC，ADCM，SUB，SUBM，SBC，SBCM，DAA
  LADD，LADDM，LADC，LADCM，LSUB，LSUBM，LSBC，LSBCM，LDAA
- 逻辑运算：
  AND，OR，XOR，ANDM，ORM，XORM，CPL，CPLA
  LAND，LANDM，LOR，LXOR， LORM，LXORM，LCPL，LCPLA
- 移位运算：
  RRA，RR，RRCA，RRC，RLA，RL，RLCA，RLC
  LRRA，LRR，LRRCA，LRRC，LRLA，LRL，LRLCA，LRLC
- 递增和递减：
  INCA，INC，DECA，DEC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

Flash 程序存储器

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

结构

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



程序存储器结构

特殊向量

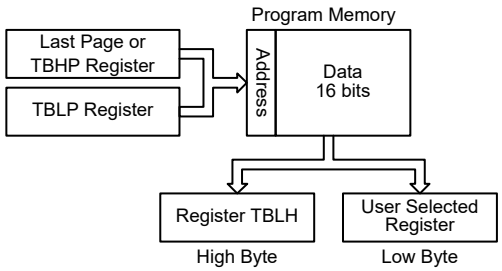
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

查表

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

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

下图是查表中寻址 / 数据流程：



查表范例

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

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

表格读取程序范例

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

在线烧录 – ICP

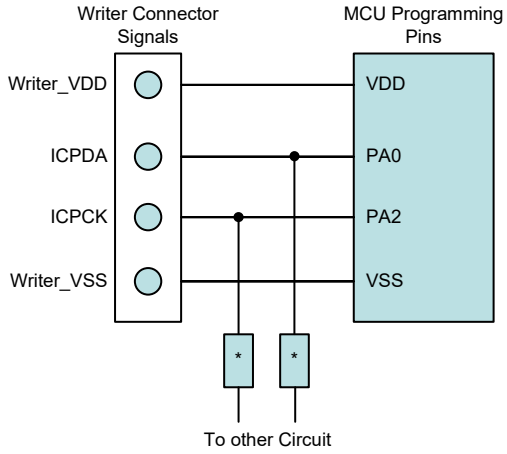
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。另外，该单片机提供 4 线接口的在线烧录方式。用户可将进行过烧录或未经过烧录的单片机芯片连同电路板一起制成，最后阶段进行程序的更新和程序的烧录，在无需去除或重新插入芯片的情况下方便地保持程序为最新版。

Flash MCU 与烧录器引脚对应表如下：

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

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

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



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

片上调试 – OCDS

EV 芯片 NW2018V 用于 NW2018 单片机仿真。此 EV 芯片提供片上调试功能 (OCDS—On-Chip Debug Support) 用于开发过程中的单片机调试。除了片上调试功能方面，EV 芯片和实际单片机在功能上几乎是兼容的。用户可将 OCDSDA 和 OCDSCK 引脚连接至开发工具，从而实现 EV 芯片对实际单片机的仿真。OCDSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户用 EV 芯片进行调试时，实际单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。

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

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。

数据存储器分为两种类型，第一种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。第二种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 可通过设置正确的间接寻址指针值实现。

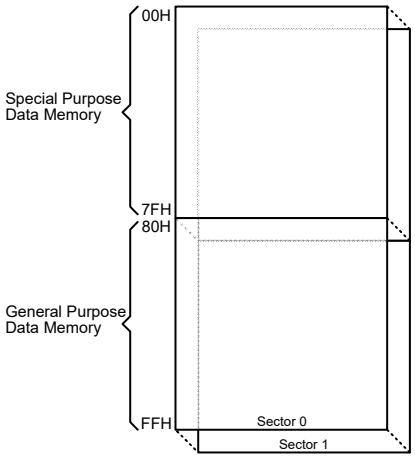
结构

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

特殊功能数据存储器地址范围为 00H~7FH，而通用数据存储器地址范围为 80H~FFH。

特殊功能数据存储器	通用数据存储器	
有效 Sectors	容量	Sector：地址
0~1	256×8	0: 80H~FFH 1: 80H~FFH

数据存储器概要



数据存储器结构

数据存储器寻址

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

直接寻址可用于所有 Sector，通过扩展指令可以寻址所有可用的数据存储器空间。当所访问的数据存储器位于除 Sector 0 外的任何数据存储器 Sector 时，扩展指令可代替间接寻址方式用来访问数据存储器。标准指令和扩展指令的主要区别在于扩展指令中的数据存储器地址“m”有 9 个有效位，高字节表示 Sector，低字节表示指定的地址。

通用数据存储器

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

特殊功能数据存储器

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

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH		
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	SCC	
10H	INTC0	
11H	INTC1	
12H	INTC2	
13H	INTC3	
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	MFI0	
19H		
1AH		
1BH		
1CH	MFI1	
1DH	MFI2	
1EH	MFI3	
1FH	WDTC	
20H	RSTFC	
21H	LVRC	
22H	LVDC	
23H	INTEG	
24H	TBC	
25H	EEA	
26H	EED	
27H	PC	
28H	PCC	
29H	PCPU	
2AH	CTM0C0	
2BH	CTM0C1	
2CH	CTM0DL	
2DH	CTM0DH	
2EH	CTM0AL	
2FH	CTM0AH	
30H	PTMC0	
31H	PTMC1	
32H	PTMDL	
33H	PTMDH	
34H	PTMAL	
35H	PTMAH	
36H	PTMRPL	
37H	PTMRPH	
38H	IICC0	
39H	IICC1	
3AH	IICD	
3BH	IICA	
3CH	IICTOC	
3DH		
3EH		
3FH		

	Sector 0	Sector 1
40H	SADOL	EEC
41H	SADOH	
42H	SADC0	
43H	SADC1	
44H	SADC2	
45H	PB	
46H	PBC	
47H	PBPU	
48H	PBS0	
49H	PPGC	
4AH	PPGPC	
4BH	PPGTA	
4CH	PPGTB	
4DH	PPGTEX	
4EH	PWLT	
4FH	TDC	
50H	CTM1C0	
51H	CTM1C1	
52H	CTM1DL	
53H	CTM1DH	
54H	CTM1AL	
55H	CTM1AH	
56H		
57H		
58H	USR	
59H	UCR1	
5AH	UCR2	
5BH	TXR_RXR	
5CH	BRG	
5DH		
5EH		
5FH		
60H	PPGCTC	
61H	SYNCBUF	
62H	PCKC	
63H	CTRL	
64H	PCS0	
65H	PCS1	
66H	CMP1C2	
67H	CMP3C2	
68H	CMP0C0	
69H	CMP0C1	
6AH	CMP0C2	
6BH	CMP0C3	
6CH	HIRCC	
6DH	CMP1C0	
6EH	CMP1C1	
6FH	CMP1DA	
70H	CMP2C0	
71H	CMP2C1	
72H	CMP2DA	
73H	CMP3C0	
74H	CMP3C1	
75H	CMP3DA	
76H	CMP4C0	
77H	CMP4C1	
78H	CMP4C2	
79H	CMP4DA	
7AH	OPAC0	
7BH	OPAC1	
7CH	PAS0	
7DH	PAS1	
7EH	TMR	
7FH	TMRC	

: Unused, read as 00H

特殊功能数据存储器

特殊功能寄存器

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

间接寻址寄存器 – IAR0，IAR1，IAR2

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

存储器指针 – MP0，MP1L/MP1H，MP2L/MP2H

该单片机提供五个存储器指针，即 MP0、MP1L、MP1H、MP2L 和 MP2H。由于这些指针在数据存储器中能像普通的寄存器一般被操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由存储器指针所指定的地址。MP0、IAR0 用于访问 Sector 0，而 MP1L/MP1H 和 IAR1、MP2L/MP2H 和 IAR2 可根据 MP1H 或 MP2H 寄存器访问所有的 Sector。使用扩展指令可对所有的数据 Sector 进行直接寻址。

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

间接寻址程序举例

范例 1

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

范例 2

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
code .section at 0 'code'
org 00h
start:
mov a,04h                  ; setup size of block
mov block,a
mov a,01h                  ; setup the memory sector
mov mp1h,a
mov a,offset adres1        ; Accumulator loaded with first RAM address
mov mp1l,a                 ; setup memory pointer with first RAM address
loop:
clr IAR1                   ; clear the data at address defined by 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 必须在每次进行加加法、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。

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

另外，当进入一个中断程序或执行子程序调用时，状态寄存器不会自动压入到堆栈保存。假如状态寄存器的内容是重要的且子程序可能改变状态寄存器的话，则需谨慎的去做正确的储存。注意，STATUS 寄存器的 0~3 位是可读写位。

- **STATUS 寄存器**

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

“x”：未知

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

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

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

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

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

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

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

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

EEPROM 数据存储器

此单片机的一个特性是内建 EEPROM 数据存储器。“Electrically Erasable Programmable Read Only Memory”为电可擦可编程只读存储器，由于其非易失的存储结构，即使在电源掉电的情况下存储器内的数据仍然保存完好。这种存储区扩展了 ROM 空间，对设计者来说增加了许多新的应用机会。EEPROM 可以用来存储产品编号、校准值、用户特定数据、系统配置参数或其它产品信息等。EEPROM 的数据读取和写入过程也会变的更简单。

EEPROM 数据存储器结构

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

EEPROM 寄存器

有三个寄存器控制内部 EEPROM 数据存储器总的操作，地址寄存器 EEA、数据寄存器 EED 及控制寄存器 EEC。EEA 和 EED 位于 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，可以通过 MP1L/MP1H 和 IAR1 或 MP2L/MP2H 和 IAR2 进行间接读取或写入。由于 EEC 控制寄存器位于 Sector 1 中的“40H”，在 EEC 寄存器上的任何操作被执行前，MP1L 或 MP2L 必须先设为“40H”，MP1H 或 MP2H 被设为“01H”。

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

EEPROM 寄存器列表

- **EEA 寄存器**

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

Bit 7~6　未定义，读为“0”

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：**数据 EEPROM 数据
数据 EEPROM 数据 Bit 7~Bit 0

- **EEC 寄存器**

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

Bit 7~4　未定义，读为“0”

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

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

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

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

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

从 EEPROM 中读取数据

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

写数据到 EEPROM

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

写保护

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

EEPROM 中断

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

编程注意事项

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

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

程序举例

从 EEPROM 中读取数据 — 轮询法

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

写数据到 EEPROM — 轮询法

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

振荡器

不同的振荡器选择可以让使用者在不同的应用需求中实现更大范围的功能。振荡器的灵活性使得在速度和功耗方面可以达到最优化。振荡器控制是通过应用程序和相关的控制寄存器共同完成的。

振荡器概述

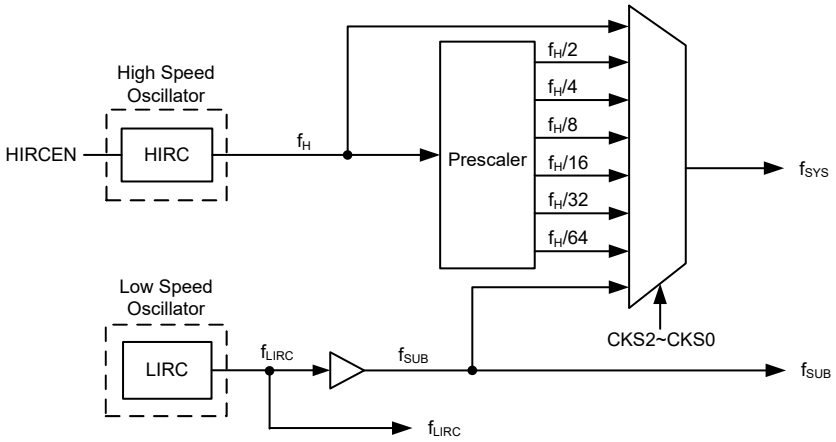
振荡器除了作为系统时钟源，还作为看门狗定时器和时基中断的时钟源。集成的内部振荡器不需要任何外围器件。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

类型	名称	频率
内部高速 RC	HIRC	16MHz
内部低速 RC	LIRC	32kHz

振荡器类型

系统时钟配置

此单片机有两个系统振荡器，包括一个高速振荡器和一个低速振荡器。高速振荡器为内部 16MHz RC 振荡器 – HIRC。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

内部 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，不需其它外部器件。内部 RC 振荡器具有一个固定频率：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。

内部 32kHz 振荡器 – LIRC

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

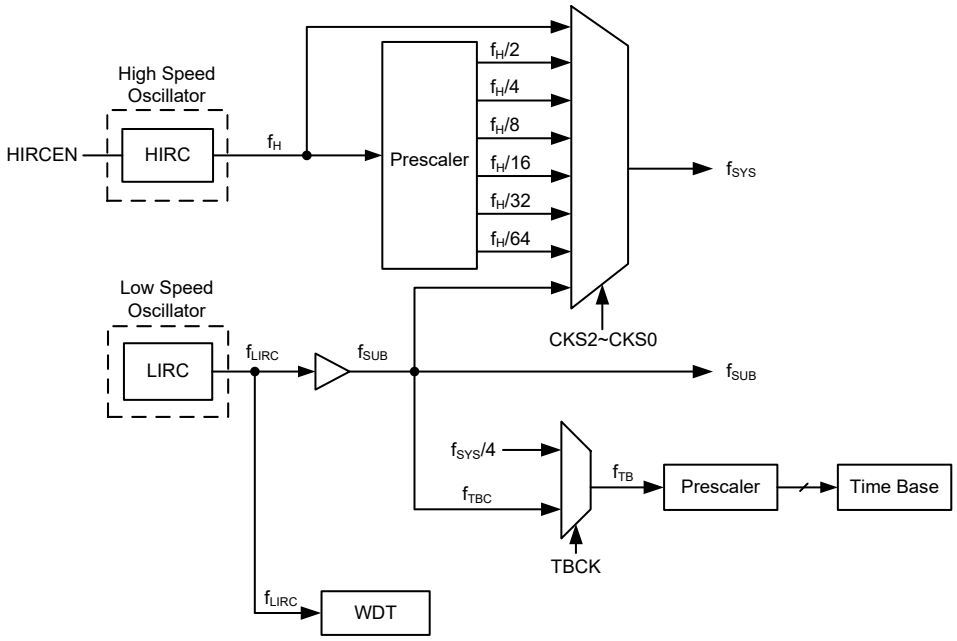
工作模式和系统时钟

现今的应用要求单片机具有较高的性能及尽可能低的功耗，这种矛盾的要求在便携式电池供电的应用领域尤为明显。高性能所需要的高速时钟将增加功耗，反之亦然。此单片机提供高、低速两种时钟源，它们之间可以动态切换，用户可通过优化单片机操作来获得最佳性能 / 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 HIRC 振荡器，低频时钟源来自 LIRC 振荡器。其它系统时钟还有高速系统振荡器的分频 $f_H/2$~$f_H/64$。



单片机时钟配置

注：当系统时钟源 f_{SYS} 由 f_H 到 f_{SUB} 转换时，可以通过设置相应的使能控制位将高速振荡器停止以节省耗电，或者使其继续振荡为外围电路提供 f_H~$f_H/64$ 频率的时钟源。

系统工作模式

单片机有 6 种不同的工作模式，每种有它自身的特性，根据应用中不同的性能和功耗要求可选择不同的工作模式。单片机正常工作有两种模式：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	xxx	On	On	On	On
空闲模式 2	Off	1	0	000~110	On	On	Off	On
				111	Off			
休眠模式	Off	0	0	xxx	Off	Off	Off	On(2)

“x”：无关

注：1. 在低速模式下，f_H 开启或关闭由相应的振荡器使能位控制。
2. 在休眠模式下，由于看门狗定时器仍然使能，f_{LIRC} 也将使能。

正常模式

顾名思义，这是主要的工作模式之一，单片机的所有功能均可在此模式中实现且系统时钟由一个高速振荡器提供。该模式下单片机正常工作的时钟源来自 HIRC 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。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。在低速模式下，f_H 需要通过设置相应的寄存器来关闭。

休眠模式

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为低时，系统进入休眠模式。在休眠模式中，CPU 及 f_{SUB} 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 寄存器中的 FHIDEN 位为低、FSIDEN 位为高时，系统进入空闲模式 0。在空闲模式 0 中，CPU 停止，但低速振荡器会开启以驱动一些外围功能。

空闲模式 1

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，但高速和低速振荡器都会开启以保持一些外围功能继续工作。

空闲模式 2

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

控制寄存器

寄存器 SCC、HIRCC 用于控制系统时钟和相应的振荡器配置。

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

系统工作模式控制寄存器列表

- **SCC 寄存器**

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

Bit 7~5 **CKS2~CKS0**：系统时钟选择位
000：f_H
001：$f_H/2$
010：$f_H/4$
011：$f_H/8$
100：$f_H/16$
101：$f_H/32$
110：$f_H/64$
111：f_{SUB}
这三位用于选择系统时钟源。除了 f_H 或 f_{SUB} 提供的系统时钟源外，也可使用高频振荡器的分频作为系统时钟。

Bit 4~2 未定义，读为“0”

Bit 1 **FHIDEN**：CPU 关闭时高频振荡器控制位
0：除能
1：使能
此位用来控制在 CPU 执行 HALT 指令关闭后高速振荡器是被激活还是停止。

Bit 0 **FSIDEN**：CPU 关闭时低频振荡器控制位
0：除能
1：使能
此位用来控制在 CPU 执行 HALT 指令关闭后低速振荡器是被激活还是停止。LIRC 振荡器是由该位与 WDT 功能控制位共同控制的。如果该位被清零，但 WDT 功能使能，LIRC 振荡器也将使能。

- **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”

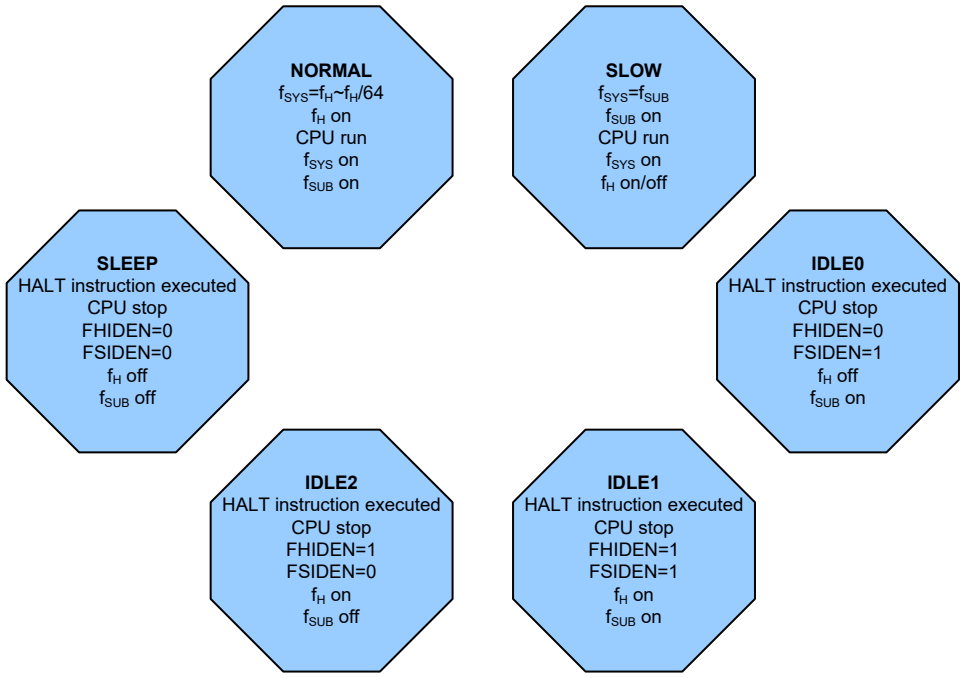
Bit 1　**HIRCF**：HIRC 振荡器稳定标志位
0：HIRC 不稳定
1：HIRC 稳定
此位用于表明 HIRC 振荡器是否稳定。HIRCEN 位置高使能 HIRC 振荡器时，HIRCF 位会先被清零，在 HIRC 稳定后会被置高。

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

工作模式切换

单片机可在各个工作模式间自由切换，使得用户可根据所需选择最佳的性能 / 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。

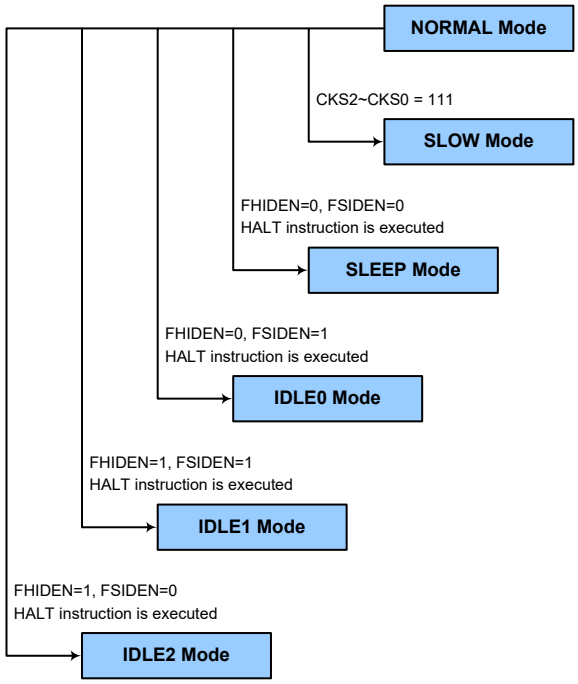
简单来说，正常模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而正常模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



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。

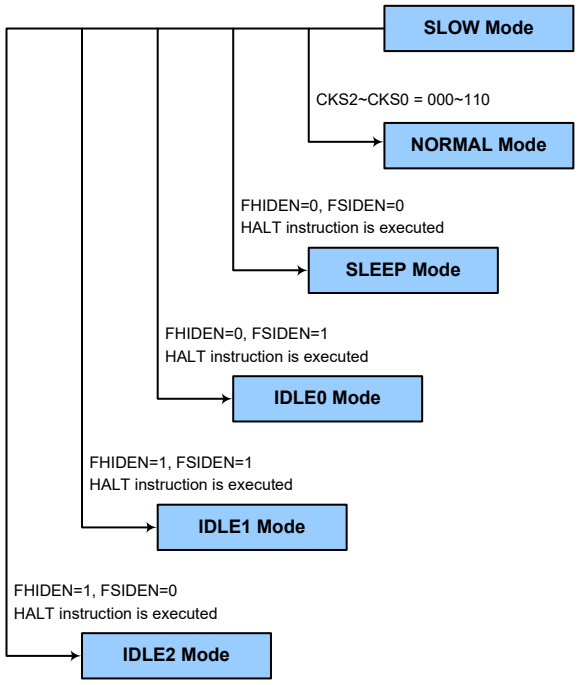
低速模式的时钟源来自 LIRC 振荡器，因此要求这个振荡器在所有模式切换动作发生前稳定下来。



低速模式切换到正常模式

在低速模式时系统时钟来自 f_{SUB}。切换回正常模式时，需设置 CKS2~CKS0 位为“000”~“110”使系统时钟从 f_{SUB} 切换到 f_H~$f_H/64$。

然而，如果在低速模式下 f_H 因未使用而关闭，那么从低速模式切换到正常模式时，它需要一定的时间来重新起振和稳定，可通过检测 HIRCC 寄存器中的 HIRCF 位进行判断，所需的高速系统振荡器稳定时间指定在交流电气特性中。



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 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为“0”。在这种模式下，除了 WDT 以外的所有时钟和功能都将关闭。在上述条件下执行该指令后，将发生的情况如下：

- 系统时钟停止运行，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 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。
- WDT 将被清零并重新开始计数。

进入空闲模式 0

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

- f_H 时钟停止运行，应用程序停止在“HALT”指令处，但 f_{SUB} 时钟将继续运行。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 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。
- WDT 将被清零并重新开始计数。

进入空闲模式 1

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

- f_H 和 f_{SUB} 时钟开启，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 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。
- WDT 将被清零并重新开始计数。

进入空闲模式 2

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

- f_H 时钟开启，f_{SUB} 时钟关闭，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 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。
- WDT 将被清零并重新开始计数。

静态电流的注意事项

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，如果选择 LIRC 振荡器，会导致耗电增加。在空闲模式 1 和空闲模式 2 中，高速振荡器开启。若外围功能时钟源来自高速振荡器，额外的静态电流也可能会有几百微安。

唤醒

单片机进入休眠模式或空闲模式后，系统时钟将停止以降低功耗。然而单片机再次唤醒，原来的系统时钟重新起振、稳定且恢复正常工作需要一定的时间。

系统进入休眠或空闲模式之后，可以通过以下几种方式唤醒：

- PA 口下降沿
- 系统中断
- WDT 溢出

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

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

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源由内部 RC 振荡器 LIRC 提供。电压为 5V 时内部振荡器 LIRC 的频率大约为 32kHz，这个特殊的内部时钟周期随 V_{DD}、温度和制成的不同而变化。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。

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 软件控制位
01010 或 10101：使能
其它值：MCU 复位
若因外部环境噪声使 WE[4:0] 值发生改变，则在 2~3 个 LIRC 周期后产生复位，复位后 RSTFC 寄存器中的 WRF 标志位会被置位。

Bit 2~0 **WS2~WS0**：WDT 溢出周期选择位
000：$2^8/f_{LIRC}$
001：$2^{10}/f_{LIRC}$
010：$2^{12}/f_{LIRC}$
011：$2^{14}/f_{LIRC}$
100：$2^{15}/f_{LIRC}$
101：$2^{16}/f_{LIRC}$
110：$2^{17}/f_{LIRC}$
111：$2^{18}/f_{LIRC}$
这三位控制 WDT 时钟源的分频比，从而实现对 WDT 溢出周期的控制。

- **RSTFC 寄存器**

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

“x”：未知

Bit 7~3 未定义，读为“0”

Bit 2 **LVRF**：LVR 复位标志位
具体描述见其它章节

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

Bit 0 **WRF**：WDTC 控制寄存器软件复位标志位
0：未发生
1：发生
当发生 WDTC 控制寄存器软件复位时，此位被置为“1”。只能通过应用程序清零。

看门狗定时器操作

当 WDT 溢出时，它产生一个芯片复位的动作。这也就意味着正常工作期间，用户需在应用程序中看门狗溢出前有策略地清看门狗定时器以防止其产生复位，可使用清除看门狗指令实现。无论什么原因，程序失常跳转到一个未知的地址或进入一个死循环，这些清除指令都不能被正确执行，此种情况下，看门狗将溢出以使单片机复位。WDTC 寄存器中的 WE4~WE0 位可用来控制看门狗定时器的使能 / 复位。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 LIRC 时钟周期后单片机复位。上电后这些位初始化为“01010B”。

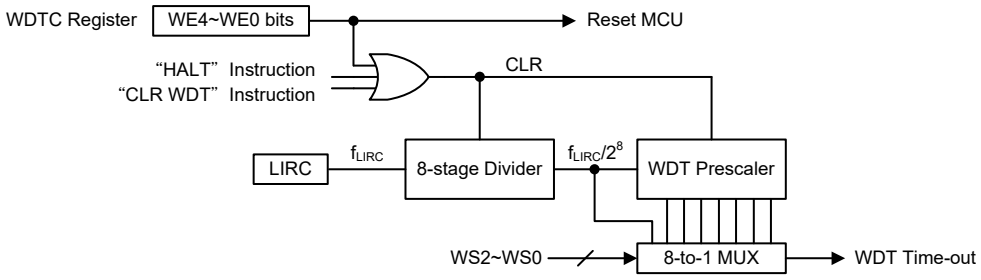
WE4~WE0 位	WDT 功能
10101B 或 01010B	使能
其它值	MCU 复位

看门狗定时器功能控制

程序正常运行时，WDT 溢出将导致芯片复位，并置位状态标志位 TO。若系统处于休眠或空闲模式，当 WDT 发生溢出时，状态寄存器中的 TO 应置位，仅 PC 和堆栈指针复位。有三种方法可以用来清除 WDT 的内容。第一种是通过 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。

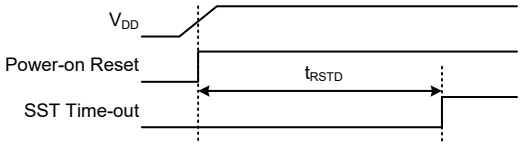
另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。还有一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。

复位功能

单片机的几种内部复位方式将在此处做具体介绍。

上电复位

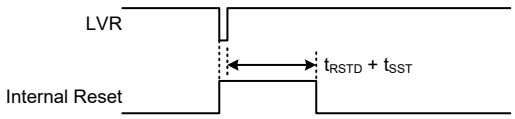
这是最基本且不可避免的复位，发生在单片机上电后。除了保证程序存储器从开始地址执行，上电复位也使得其它寄存器被设定在预设条件。所有的输入 / 输出端口控制寄存器在上电复位时会保持高电平，以确保上电后所有引脚被设定为输入状态。



上电复位时序图

低电压复位 – LVR

单片机具有低电压复位电路，用来监测它的电源电压。LVR 始终使能于特定的电压值，V_{LVR}。例如在更换电池的情况下，单片机供应的电压可能会在 0.9V~V_{LVR} 之间，这时 LVR 将会自动复位单片机且 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 LVD & LVR 电气特性中 t_{LVR} 参数的值。如果低电压存在不超过 t_{LVR} 参数的值，则 LVR 将会忽略它且不会执行复位功能。V_{LVR} 参数值通过 LVRC 寄存器设置。若因为环境噪声或软件设置修改了 LVRC 寄存器的值，LVR 将在 2~3 个 LIRC 时钟周期后复位单片机。同时 RSTFC 寄存器 LRF 位将被置“1”。上电复位后 LVRC 的初始值是 01010101B。需要注意的是，当单片机进入空闲或休眠模式，LVR 功能将自动关闭。



低电压复位时序图

- **LVRC 寄存器**

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

Bit 7~0 **LVS7~LVS0**：LVR 电压选择
01010101：3.15V
00110011：3.15V
10011001：3.15V
10101010：3.15V
其它值：MCU 复位 – 寄存器复位为 POR 值
若低电压情况发生且满足以上定义的低电压复位值，则单片机复位。此时复位后的寄存器内容保持不变。
除了以上定义的低电压复位值外，其它值也能导致单片机复位。需要经过 2~3 个 LIRC 时钟周期响应复位。但此时寄存器内容将复位为 POR 值。

- **RSTFC 寄存器**

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

“x”：未知

Bit 7~3 未定义，读为“0”

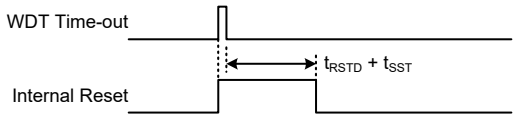
Bit 2 **LVRF**：LVR 复位标志位
0：未发生
1：发生
当特定的低电压复位条件发生时，此位被置为“1”，且只能通过应用程序清零。

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

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

正常运行时看门狗溢出复位

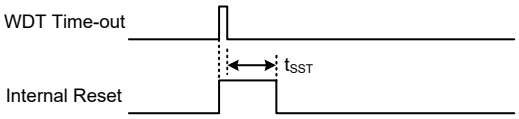
除了看门狗溢出标志位 TO 将被设为“1”之外，正常运行时看门狗溢出复位和 LVR 复位相同。



正常运行时看门狗溢出时序图

休眠或空闲时看门狗溢出复位

休眠或空闲时看门狗溢出复位和其它种类的复位有些不同。除了程序计数器与堆栈指针将被清“0”及 TO 位被设为“1”外，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。



休眠或空闲时看门狗溢出复位时序图

复位初始状态

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

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

“u”代表不改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

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

不同的复位形式对单片机内部寄存器的影响是不同的。为保证复位后程序能正常执行，了解寄存器在特定条件复位后的设置是非常重要的。下表即为不同方式复位后内部寄存器的状况。此芯片有多种封装类型，表格反应较大的封装的情况。

寄存器	上电复位	WDT 溢出 (正常模式)	WDT 溢出 (空闲 / 休眠模式)
MP0	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	---- xxxx	---- xxxx	---- uuuu
STATUS	xx00 xxxx	xx1u uuuu	uu11 uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
SCC	001- --00	001- --00	uuu- --uu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
INTC3	0000 0000	0000 0000	uuuu uuuu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
MFI0	-000 -000	-000 -000	-uuu -uuu
MFI1	-000 -000	-000 -000	-uuu -uuu
MFI2	-000 -000	-000 -000	-uuu -uuu
MFI3	--00 --00	--00 --00	--uu --uu
WDTC	0101 0011	0101 0011	uuuu uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu
LVRC	0101 0101	0101 0101	uuuu uuuu
LVDC	--00 0000	--00 0000	--uu uuuu
INTEG	---- 0000	---- 0000	---- uuuu
TBC	0011 -111	0011 -111	uuuu -uuu
EEA	--00 0000	--00 0000	--uu uuuu
EED	0000 0000	0000 0000	uuuu uuuu
PC	---1 1111	---1 1111	---u uuuu
PCC	---1 1111	---1 1111	---u uuuu
PCPU	---0 0000	---0 0000	---u uuuu
CTM0C0	0000 0000	0000 0000	uuuu uuuu
CTM0C1	0000 0000	0000 0000	uuuu uuuu
CTM0DL	0000 0000	0000 0000	uuuu uuuu
CTM0DH	---- --00	---- --00	---- --uu
CTM0AL	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常模式)	WDT 溢出 (空闲 / 休眠模式)
CTM0AH	---- --00	---- --00	---- --uu
PTMC0	0000 0---	0000 0---	uuuu u---
PTMC1	0000 0000	0000 0000	uuuu uuuu
PTMDL	0000 0000	0000 0000	uuuu uuuu
PTMDH	---- --00	---- --00	---- --uu
PTMAL	0000 0000	0000 0000	uuuu uuuu
PTMAH	---- --00	---- --00	---- --uu
PTMRPL	0000 0000	0000 0000	uuuu uuuu
PTMRPH	---- --00	---- --00	---- --uu
IICC0	---- 000-	---- 000-	---- uuu-
IICC1	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA	0000 000-	0000 000-	uuuu uuu-
IICTOC	0000 0000	0000 0000	uuuu uuuu
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
	---- xxxx	---- xxxx	---- uuuu (ADRFS=1)
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 -000	0000 -000	uuuu -uuu
SADC2	0--0 0000	0--0 0000	u--u uuuu
PB	---- 1111	---- 1111	---- uuuu
PBC	---- 1111	---- 1111	---- uuuu
PBPU	---- 0000	---- 0000	---- uuuu
PBS0	---- 0000	---- 0000	---- uuuu
PPGC	00-0 0-00	00-0 0-00	uu-u u-uu
PPGPC	0000 0000	0000 0000	uuuu uuuu
PPGTA	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGTB	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGTEX	---x ---x	---x ---x	---u ---u
PWLT	xxxx xxxx	xxxx xxxx	uuuu uuuu
TDC	0000 0000	0000 0000	uuuu uuuu
CTM1C0	0000 0000	0000 0000	uuuu uuuu
CTM1C1	0000 0000	0000 0000	uuuu uuuu
CTM1DL	0000 0000	0000 0000	uuuu uuuu
CTM1DH	---- --00	---- --00	---- --uu
CTM1AL	0000 0000	0000 0000	uuuu uuuu
CTM1AH	---- --00	---- --00	---- --uu
USR	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常模式)	WDT 溢出 (空闲 / 休眠模式)
UCR2	0000 0000	0000 0000	uuuu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGCTC	---- 0000	---- 0000	---- uuuu
SYNCBUF	0000 0000	0000 0000	uuuu uuuu
PCKC	-000 0000	-000 0000	-uuu uuuu
CTRL	-000 0---	-000 0---	-uuu u---
PCS0	0000 0000	0000 0000	uuuu uuuu
PCS1	---- --00	---- --00	---- --uu
CMP1C2	00-- ----	00-- ----	uu-- ----
CMP3C2	00-- ----	00-- ----	uu-- ----
CMP0C0	0000 000-	0000 000-	uuuu uuu-
CMP0C1	x001 0000	x001 0000	uuuu uuuu
CMP0C2	--00 0000	--00 0000	--uu uuuu
CMP0C3	--00 0000	--00 0000	--uu uuuu
HIRCC	---- --01	---- --01	---- --uu
CMP1C0	0000 0000	0000 0000	uuuu uuuu
CMP1C1	x001 0000	x001 0000	uuuu uuuu
CMP1DA	0000 0000	0000 0000	uuuu uuuu
CMP2C0	0000 0000	0000 0000	uuuu uuuu
CMP2C1	x001 0000	x001 0000	uuuu uuuu
CMP2DA	0000 0000	0000 0000	uuuu uuuu
CMP3C0	0-00 0000	0-00 0000	u-uu uuuu
CMP3C1	x001 0000	x001 0000	uuuu uuuu
CMP3DA	0000 0000	0000 0000	uuuu uuuu
CMP4C0	0000 0000	0000 0000	uuuu uuuu
CMP4C1	x001 0000	x001 0000	uuuu uuuu
CMP4C2	00-- --00	00-- --00	uu-- --uu
CMP4DA	0000 0000	0000 0000	uuuu uuuu
OPAC0	0000 000x	0000 000x	uuuu uuuu
OPAC1	0010 0000	0010 0000	uuuu uuuu
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
TMR	0000 0000	0000 0000	uuuu uuuu
TMRC	00-0 -000	00-0 -000	uu-u -uuu
EEC	---- 0000	---- 0000	---- uuuu

注：“u”表示不改变
“x”表示未知
“-”表示未定义

输入 / 输出端口

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	PA1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	—	—	PB3	PB2	PB1	PB0
PBC	—	—	—	—	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	—	—	PBPU3	PBPU2	PBPU1	PBPU0
PC	—	—	—	PC4	PC3	PC2	PC1	PC0
PCC	—	—	—	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	—	—	—	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0

“—”：未定义，读为“0”

I/O 逻辑功能寄存器列表

上拉电阻

许多产品应用在端口处于输入状态时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 I/O 引脚设为输入或设为 NMOS 输出时，上拉电阻功能才会受 PxPU 控制开启，其他状态上拉功能不可用。

- **PxPU 寄存器**

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

PxPUn：I/O Px 口上拉电阻控制位
0：除能
1：使能

PxPUn 位用于控制上拉电阻功能。这里的 x 可以是端口 A、B 和 C。但是，每个 I/O 端口实际有效位可能不同。

PA 口唤醒

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

需要注意的是，只有当引脚功能为通用 I/O 功能且单片机处于省电模式时，唤醒功能才会受 PAWU 控制开启，其它状态下此唤醒功能不可用。

- **PAWU 寄存器**

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

Bit 7~0 **PAWU7~PAWU0**：PA 口 bit 7 ~ bit 0 唤醒功能控制位
0：除能
1：使能

输入 / 输出端口控制寄存器

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

- **PxC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PxC7	PxC6	PxC5	PxC4	PxC3	PxC2	PxC1	PxC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

PxCn：I/O Px 口类型选择位
0：输出
1：输入

PxCn 位用于控制引脚类型。这里的 x 可以是端口 A、B 和 C。但是，每个 I/O 端口实际有效位可能不同。

引脚共用功能

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

引脚共用寄存器

封装中有限的引脚个数会对某些单片机功能造成影响。然而，引脚功能共用和引脚功能选择，使得小封装单片机具有更多不同的功能。单片机包含端口“x”输出功能选择寄存器“n”，记为 PxSn，和输入功能选择寄存器记为 CTRL，这些寄存器可以用来选择共用引脚的特定功能。

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，然后再配置相应的功能设置最后再使能此功能。要正确地取消引脚共用功能，首先应除能该功能，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	—	—	—	—	PBS03	PBS02	PBS01	PBS00
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	—	—	—	—	—	—	PCS11	PCS10
CTRL	—	TDCCS2	RXPC	SCLPC	SDAPC	—	—	—

引脚共用功能选择寄存器列表

- **PAS0 寄存器**

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

Bit 7~6 **PAS07~PAS06**：PA3 引脚共用功能选择
00：PA3/PTCK
01：PTP
10：PCK
11：AN2

Bit 5~4 **PAS05~PAS04**：PA2 引脚共用功能选择
00：PA2/INT0
01：RX
10：CTP1
11：SCL

Bit 3~2 **PAS03~PAS02**：PA1 引脚共用功能选择
00：PA1
01：OPO
10：AN1
11：OPO/AN1

Bit 1~0 **PAS01~PAS00**：PA0 引脚共用功能选择
00：PA0
01：TX
10：SDA
11：AN8

- **PAS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PAS17~PAS16**：PA7 引脚共用功能选择
00：PA7/INT1/CTCK0
01：CTP0
10：SCL
11：AN6

Bit 5~4 **PAS15~PAS14**：PA6 引脚共用功能选择
00：PA6
01：SDA
10：TD3
11：AN5

Bit 3~2 **PAS13~PAS12**：PA5 引脚共用功能选择
00：PA5
01：PCK
10：TD2
11：AN4

Bit 1~0 **PAS11~PAS10**：PA4 引脚共用功能选择
00：PA4/ CTCK1
01：TD1
10：VREF
11：AN3

- **PBS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PBS03	PBS02	PBS01	PBS00
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 **PBS03~PBS02**：PB1 引脚共用功能选择
00：PB1
01：PB1
10：PB1
11：RX

Bit 1~0 **PBS01~PBS00**：PB0 引脚共用功能选择
00：PB0
01：PB0
10：PB0
11：TX

- **PCS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
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 **PCS07~PCS06**：PC3 引脚共用功能选择
00：PC3
01：PC3
10：SYN-
11：AN0

Bit 5~4 **PCS05~PCS04**：PC2 引脚共用功能选择
00：PC2
01：PC2
10：PC2
11：IS0

Bit 3~2 **PCS03~PCS02**：PC1 引脚共用功能选择
00：PC1
01：AN7
10：ACD
11：ACD/AN7

Bit 1~0 **PCS01~PCS00**：PC0 引脚共用功能选择
00：PC0
01：PC0
10：PC0
11：IS1

- **PCS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PCS11	PCS10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

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

- **CTRL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	TDCCS2	RXPC	SCLPC	SDAPC	—	—	—
R/W	—	R/W	R/W	R/W	R/W	—	—	—
POR	—	0	0	0	0	—	—	—

Bit 7　未定义，读为“0”

Bit 6　**TDCCS2**：恒定电流输出电流选择
详见别处描述

Bit 5　**RXPC**：RX 输入源引脚选择
0：PA2
1：PB1

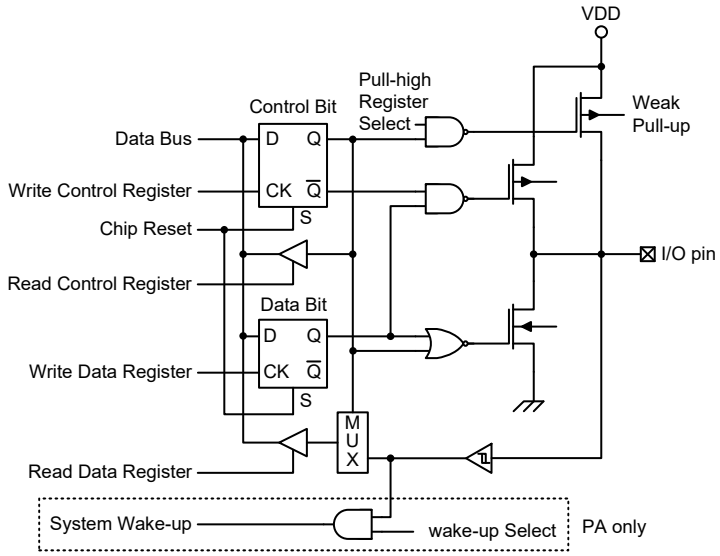
Bit 4　**SCLPC**：SCL 输入源引脚选择
0：PA2
1：PA7

Bit 3　**SDAPC**：SDA 输入源引脚选择
0：PA0
1：PA6

Bit 2~0　未定义，读为“0”

输入 / 输出引脚结构

下图为输入 / 输出引脚逻辑功能的内部结构图。输入 / 输出引脚的准确逻辑结构图可能与此图不同，这里只是为了方便对 I/O 引脚逻辑功能的理解提供一个参考。由于存在诸多的引脚共用结构，在此不方便提供所有类型引脚功能结构图。



逻辑功能输入 / 输出端口

编程注意事项

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

定时 / 计数器在任何单片机中都是一个很重要的部分，提供程序设计者一种实现和时间有关功能的方法。该单片机具有一个 8 位的向上计数器。该定时 / 计数器有两种不同的工作模式，可以当作一个普通定时器或用于 PPG 不可重复触发功能。

有两种和定时 / 计数器相关的寄存器。第一种类型的寄存器是用来存储实际的计数值，赋值给此寄存器可以设定初始值，读取此寄存器可获得定时 / 计数器的内容；第二种类型的寄存器为定时器控制寄存器，用来定义定时 / 计数器工作模式和定时设置。

配置定时 / 计数器输入时钟源

该定时 / 计数器只有一个内部时钟来源，即系统时钟 f_{SYS}。内部时钟可以用来产生精确的时基信号。内部时钟首先由分频器分频，分频比由定时器控制寄存器中的 TPSC[2:0] 位来确定。

定时 / 计数寄存器 – TMR

定时 / 计数寄存器 TMR 位于特殊数据存储器内的特殊功能寄存器，用于储存定时器的当前值。在用作内部定时且收到一个内部计数脉冲时，此寄存器的值将会加一。定时器将从预置寄存器所载入的值开始计数，到 FFH 时定时器溢出且会产生一个内部中断信号。定时器随后重新载入预置寄存器的值并继续计数。

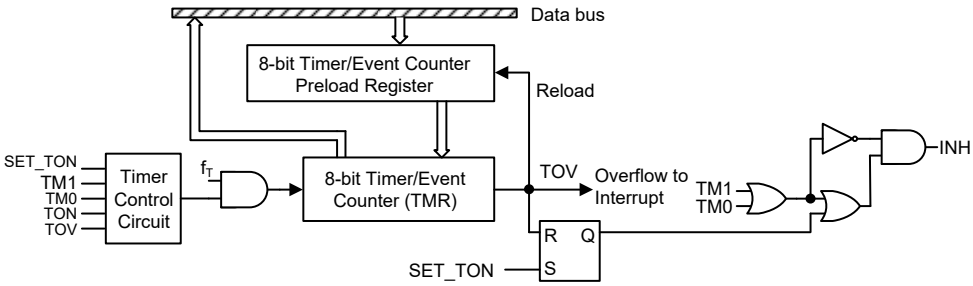
注意，上电后预置寄存器处于未知状态。为了得到定时器的最大计数范围，预置寄存器需要先清为零。定时 / 计数器在关闭条件下，写数据到预置寄存器，会立即写入实际的定时器。而如果定时 / 计数器已经打开且正在计数，在此期间内写入到预置寄存器的任何新数据将保留在预置寄存器，直到溢出发生时才被写入实际定时器。

定时 / 计数器控制寄存器 – TMRC

该单片机灵活的特性也表现在定时器的多功能上，定时 / 计数器能提供几种不同的工作模式，由相应的控制寄存器来选择定时 / 计数器的工作方式。

定时 / 计数控制寄存器为 TMRC，配合相应的 TMR 寄存器控制定时 / 计数器的全部操作。在使用定时器之前，需要先正确地设定定时 / 计数控制寄存器，以便保证定时器能正确操作，而这个过程通常在程序初始化期间完成。

定时 / 计数控制寄存器的第 7 位和第 6 位，即 TM1/TM0，用来设定定时器的工作模式，定时器模式或是用于 PPG 不可重复触发功能的模式 0。定时 / 计数控制寄存器的第 4 位即 TON，用于定时器开关控制，设定为逻辑高时，计数器开始计数，而清零时则停止计数。定时 / 计数控制寄存器的第 0~2 位用来控制输入时钟预分频器。



注：INH 用来抑制 PPG 进一步触发。

8-bit 定时 / 计数器结构图

- **TMRC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TM1	TM0	—	TON	—	TPSC2	TPSC1	TPSC0
R/W	R/W	R/W	—	R/W	—	R/W	R/W	R/W
POR	0	0	—	0	—	0	0	0

Bit 7~6 **TM1~TM0**：选择定时 / 计数器工作模式
00：模式 0 (用于 PPG 不可重复触发功能)
01：未使用
10：定时器模式
11：未使用

Bit 5 未定义，读为“0”

Bit 4 **TON**：定时 / 计数器使能
0：除能
1：使能

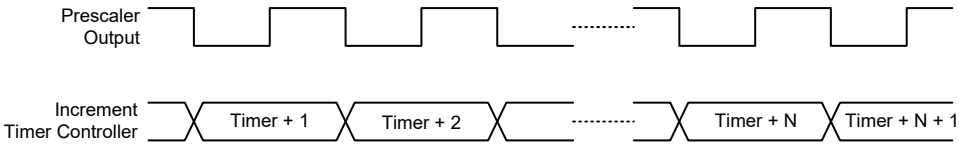
Bit 3 未定义，读为“0”

Bit 2~0 **TPSC2~TPSC0**：选择定时器预分频比
定时器内部时钟 f_T=
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：$f_{SYS}/128$

定时器模式

在这个模式下，定时器可以用来测量固定时间间隔，当定时器发生溢出时，就会产生一个内部中断信号。为使定时 / 计数器工作在定时器模式，位 TM1/TM0 必须分别设置为 1 和 0。

在定时器模式中，f_{SYS} 被用来当定时器的输入时钟源。然而，该定时器时钟源可以被预分频器进一步分频，分频比是由定时器控制寄存器的 TPSC2~TPSC0 位来确定。定时器控制寄存器第 4 位，即 TON 位需要设为逻辑高，才能令定时器工作。每次内部时钟由高到低的电平转换都会使定时器值增加一。当定时器计数已满即溢出时，会产生中断信号且定时器会重新载入预置寄存器的值，然后继续计数。定时器溢出以及相应的内部中断产生也是唤醒暂停模式的一种方法。



定时器模式时序图

模式 0

该定时 / 计数器具有模式 0，可以实现 PPG 不可重复触发功能。为使定时 / 计数器工作在模式 0，工作模式选择位 TM1/TM0 必须都设置为 0。

在模式 0 中，定时 / 计数器在 PPG 停止时开始计数，计数溢出时停止计数。也就是说一旦 PPG 停止时 TON 将被置位，定时 / 计数器溢出时 TON 将被清零。同其它模式一样，当定时 / 计数器计满，即溢出时会产生中断信号且定时 / 计数器会重新加载预置寄存器的值，然后继续向上计数。

编程注意事项

当定时 / 计数器工作在定时器模式时，内部的系统时钟作为定时器的时钟源，因此与单片机所有运算都能同步。在这个模式下，当定时器寄存器溢出时，单片机将产生一个内部中断信号，使程序进入相应的内部中断向量。

当读取定时 / 计数器值或写数据到预置寄存器时，计数时钟会被禁止以避免发生错误，但这样做可能会导致计数错误，所以程序设计者应该考虑到这点。在第一次使用定时 / 计数器之前，要仔细确认有没有正确地设定初始值。中断控制寄存器中的定时器中断使能位需要正确的设置，否则相应定时 / 计数器内部中断仍然无效。定时 / 计数器控制寄存器中的定时 / 计数器工作模式和定时器预分频比也需要正确的设定，以确保定时 / 计数器按照应用需求而正确的配置。在定时 / 计数器打开之前，需要确保先载入定时 / 计数器寄存器的初始值，这是因为在上电后，定时 / 计数器寄存器中的初始值是未知的。定时 / 计数器初始化后，可以使用定时 / 计数器控制寄存器中的使能位来打开或关闭定时器。

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相关定时器章节。

简介

该单片机包含 3 个 TM，每个 TM 可被划分为一个特定的类型，即简易型 TM 或周期型 TM。虽然性质相似，但不同 TM 特性复杂度不同。本章介绍简易型和周期型 TM 的共性，更多详细资料分别见后面各章。两种类型 TM 的特性和区别见下表。

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

TM 功能概要

CTM0	CTM1	PTM
10-bit CTM	10-bit CTM	10-bit PTM

TM 名称 / 类型参考

TM 操作

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

TM 时钟源

驱动 TM 计数器的时钟源很多。通过设置 xTMn 控制寄存器的 xTnCK2~xTnCK0 位，选择所需的时钟源，其中 x 代表 C 或 P 类型，n 代表具体 TM 编号。由于该单片机只包含一个 PTM，PTM 相关的引脚、寄存器和控制位等都不带编号。该时钟源来自系统时钟 f_{SYS} 的分频比或内部高速时钟 f_H 或 f_{SUB} 时钟源或外部 xTCKn 引脚。xTCKn 引脚时钟源用于允许外部信号作为 TM 时钟源或用于事件计数。

TM 中断

每个类型 TM 拥有两个内部中断，分别是内部比较器 A 或比较器 P，当比较匹配发生时产生 TM 中断。当 TM 中断产生时，计数器清零并改变 TM 输出引脚的状态。

TM 外部引脚

每种类型的 TM 都有一个 TM 输入引脚 xTCKn。xTMn 输入引脚 xTCKn 作为 xTMn 时钟源输入脚，通过设置 xTMnC0 寄存器中的 xTnCK2~xTnCK0 位进行选择。外部时钟源可通过该引脚来驱动内部 TM。TM 引脚可选择上升沿有效或下降沿有效。PTCK 引脚还可用作 PTM 单脉冲输出模式的外部触发引脚。

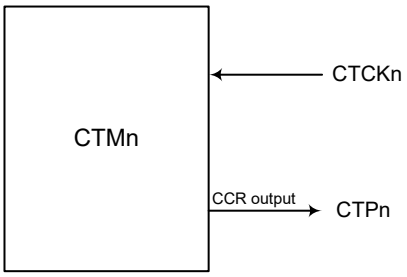
每个 TM 都有一个输出引脚 xTPn。当 TM 工作在比较匹配输出模式且比较匹配发生时，这些引脚会由 TM 控制切换到高电平或低电平或翻转。外部 xTPn 输出引脚也被 TM 用来产生 PWM 输出波形。此外，PTP 引脚还可用作 PTM 捕捉输入模式的外部触发源，其有效边沿有上升沿、下降沿和双沿，通过设置 PTMC1 寄存器中的 PTIO1~PTIO0 位来选择有效边沿类型。当 TM 输入和输出引脚与其它功能共用时，TM 输入和输出功能需要通过相关引脚共用功能选择寄存器先被设置。

CTM0		CTM1		PTM	
输入	输出	输入	输出	输入	输出
CTCK0	CTP0	CTCK1	CTP1	PTCK	PTP

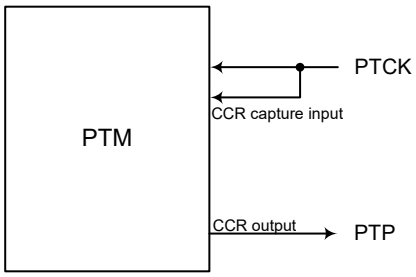
TM 外部引脚

TM 输入 / 输出引脚选择

选择作为 TM 输入 / 输出引脚还是其它共用引脚功能是通过设置相关引脚共用寄存器来实现的。每个 TM 输入 / 输出引脚都有对应的引脚共用选择位。正确设置选择位将相应的引脚用作 TM 输入 / 输出。更多引脚共用功能选择详见引脚共用功能章节。



CTM 功能引脚控制框图 (n=0 或 1)

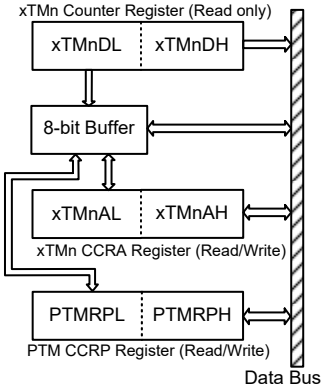


PTM 功能引脚控制框图

编程注意事项

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

CCRA 和 CCRP 寄存器访问方式如下图所示，读写这些成对的寄存器需通过特殊的方式。建议使用“MOV”指令按照以下步骤访问 CCRA 或 CCRP 低字节寄存器，即 xTMnAL 和 PTMRPL，否则可能导致无法预期的结果。

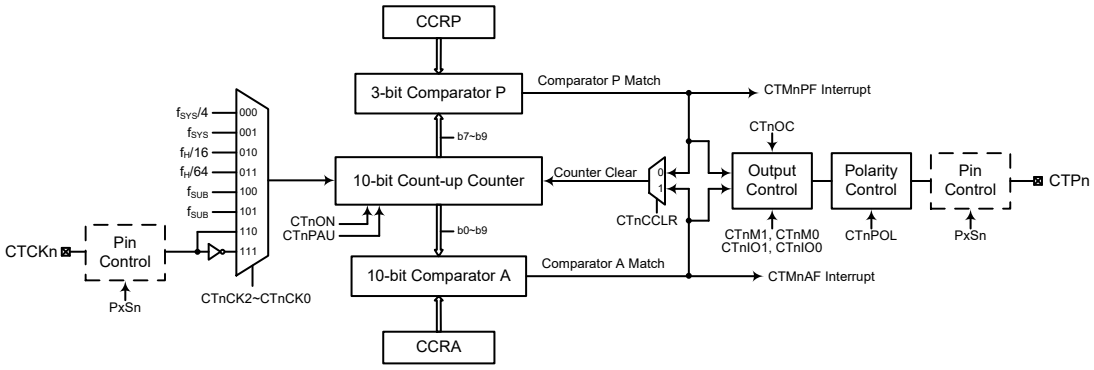


读写流程如下步骤所示：

- 写数据至 CCRA 或 CCRP
  - ♦ 步骤 1. 写数据至低字节寄存器 xTMnAL 或 PTMRPL
    - – 注意，此时数据仅写入 8-bit 缓存器。
  - ♦ 步骤 2. 写数据至高字节寄存器 xTMnAH 或 PTMRPH
    - – 注意，此时数据直接写入高字节寄存器，同时锁存在 8-bit 缓存器中的数据写入低字节寄存器。
- 由计数器寄存器和 CCRA 或 CCRP 中读取数据
  - ♦ 步骤 1. 由高字节寄存器 xTMnDH、xTMnAH 或 PTMRPH 读取数据
    - – 注意，此时高字节寄存器中的数据直接读取，同时由低字节寄存器读取的数据锁存至 8-bit 缓存器中。
  - ♦ 步骤 2. 由低字节寄存器 xTMnDL、xTMnAL 或 PTMRPL 读取数据
    - – 注意，此时读取 8-bit 缓存器中的数据。

简易型 TM – CTM

虽然简易型 TM 是几种 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。



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 CTnON 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 CTMn 中断信号。简易型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，也可以控制输出脚。所有工作模式的设定都是通过设置相关寄存器来实现的。

简易型 TM 寄存器介绍

简易型 TM 的所有操作由一系列寄存器控制。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
CTMnC0	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
CTMnC1	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
CTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnDH	—	—	—	—	—	—	D9	D8
CTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnAH	—	—	—	—	—	—	D9	D8

10-bit 简易型 TM 寄存器列表 (n=0 或 1)

- **CTMnC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
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 **CTnPAU**：CTMn 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，CTMn 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 **CTnCK2~CTnCK0**：选择 CTMn 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：CTCKn 上升沿时钟
111：CTCKn 下降沿时钟
此三位用于选择 CTMn 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 **CTnON**：CTMn 计数器 On/Off 控制位
0：Off
1：On
此位控制 CTMn 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 CTMn。清零此位将停止计数器并关闭 CTMn 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 CTMn 处于比较匹配输出模式或 PWM 输出模式，当 CTnON 位经由低到高转换时，CTMn 输出脚将复位至 CTnOC 位指定的初始值。

Bit 2~0 **CTnRP2~CTnRP0**：CTMn CCRP 3-bit 寄存器，与 CTMn 计数器 bit 9～bit 7 进行比较
比较器 P 匹配周期
000：1024 个 CTMn 时钟周期
001：128 个 CTMn 时钟周期
010：256 个 CTMn 时钟周期
011：384 个 CTMn 时钟周期
100：512 个 CTMn 时钟周期
101：640 个 CTMn 时钟周期
110：768 个 CTMn 时钟周期
111：896 个 CTMn 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 CTnCCLR 位设定为 0 时，此比较结果可用于清除内部计数器。CTnCCLR 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。

- **CTMnC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
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 **CTnM1~CTnM0**：选择 CTMn 工作模式位
00：比较匹配输出模式
01：未定义
10：PWM 输出模式
11：定时 / 计数器模式
这两位设置 CTMn 需要的工作模式。为了确保操作可靠，CTMn 应在 CTnM1 和 CTnM0 位有任何改变前先关掉。在定时 / 计数器模式，CTMn 输出脚控制必须除能。

Bit 5~4 **CTnIO1~CTnIO0**：选择 CTMn 外部引脚 CTPn 功能位
比较匹配输出模式
00：无变化
01：输出低
10：输出高
11：输出翻转
PWM 输出模式
00：强制无效状态
01：强制有效状态
10：PWM 输出
11：未定义
定时 / 计数器模式
未使用
此两位用于决定在一定条件达到时 CTMn 输出脚如何改变状态。这两位值的选择决定 CTMn 运行在哪种模式下。
在比较匹配输出模式下，CTnIO1 和 CTnIO0 位决定当比较器 A 比较匹配输出发生时 CTMn 输出脚如何改变状态。当比较器 A 比较匹配输出发生时 CTMn 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。CTMn 输出脚的初始值通过 CTMnC1 寄存器的 CTnOC 位设置取得。注意，由 CTnIO1 和 CTnIO0 位得到的输出电平必须与通过 CTnOC 位设置的初始值不同，否则当比较匹配发生时，CTMn 输出脚将不会发生变化。在 CTMn 输出脚改变状态后，通过 CTnON 位由低到高电平的转换复位至初始值。
在 PWM 输出模式，CTnIO1 和 CTnIO0 用于决定比较匹配条件发生时怎样改变 CTMn 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 CTMn 关闭时改变 CTnIO1 和 CTnIO0 位的值是很有必要的。若在 CTMn 运行时改变 CTnIO1 和 CTnIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **CTnOC**：CTMn CTPn 输出控制位
比较匹配输出模式
0：初始低
1：初始高
PWM 输出模式
0：低有效
1：高有效
这是 CTMn 输出脚输出控制位。它取决于 CTMn 此时正运行于比较匹配输出模式还是 PWM 输出模式。若 CTMn 处于定时 / 计数器模式，则其不受影响。在比较匹配输出模式时，其决定比较匹配发生前 CTMn 输出脚的逻辑电平值。在 PWM 输出模式时，其决定 PWM 信号是高有效还是低有效。

Bit 2 **CTnPOL**：CTMn CTPn 输出极性控制位
0：同相
1：反相
此位控制 CTPn 输出脚的极性。此位为高时 CTMn 输出脚反相，为低时 CTMn 输出脚同相。若 CTMn 处于定时 / 计数器模式时其不受影响。

Bit 1 **CTnDPX**：CTMn 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 **CTnCCLR**：选择 CTMn 计数器清零条件位
0：CTMn 比较器 P 匹配
1：CTMn 比较器 A 匹配
此位用于选择清除计数器的方法。简易型 TM 包括两个比较器 -- 比较器 A 和比较器 P。这两个比较器每个都可以用作清除内部计数器。CTnCCLR 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。CTnCCLR 位在 PWM 输出模式时未使用。

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

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

- **CTMnAL 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**：CTMn CCRA 低字节寄存器 bit 7 ~ bit 0
CTMn 10-bit CCRA bit 7 ~ bit 0

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

简易型 TM 工作模式

简易型 TM 有三种工作模式，即比较匹配输出模式，PWM 输出模式或定时 / 计数器模式。通过设置 CTMnC1 寄存器的 CTnM1 和 CTnM0 位选择任意工作模式。

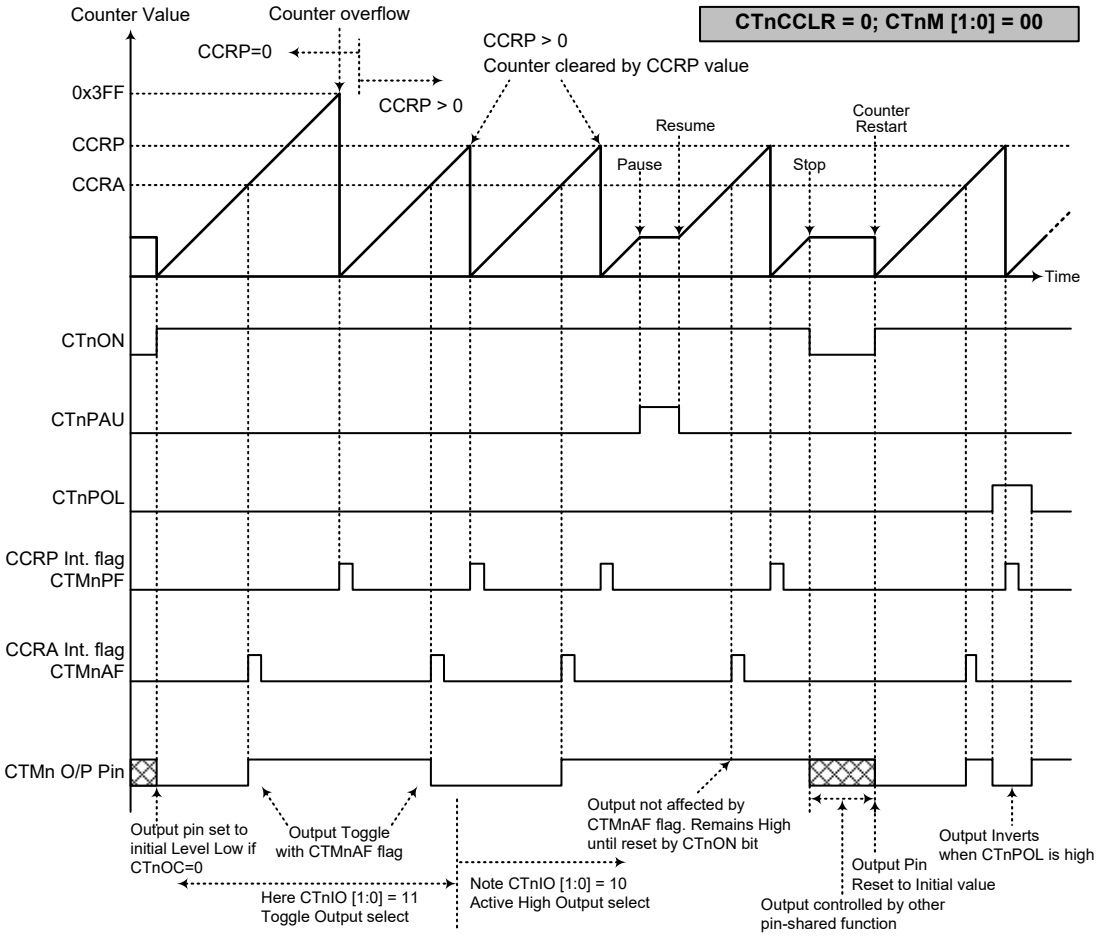
比较匹配输出模式

为使 CTMn 工作在此模式，CTMnC1 寄存器中的 CTnM1 和 CTnM0 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 CTnCCLR 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 CTMnAF 和 CTMnPF 将分别置起。

如果 CTMnC1 寄存器的 CTnCCLR 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 CTMnAF 中断请求标志产生。所以当 CTnCCLR 为高时，不产生 CTMnPF 中断请求标志。

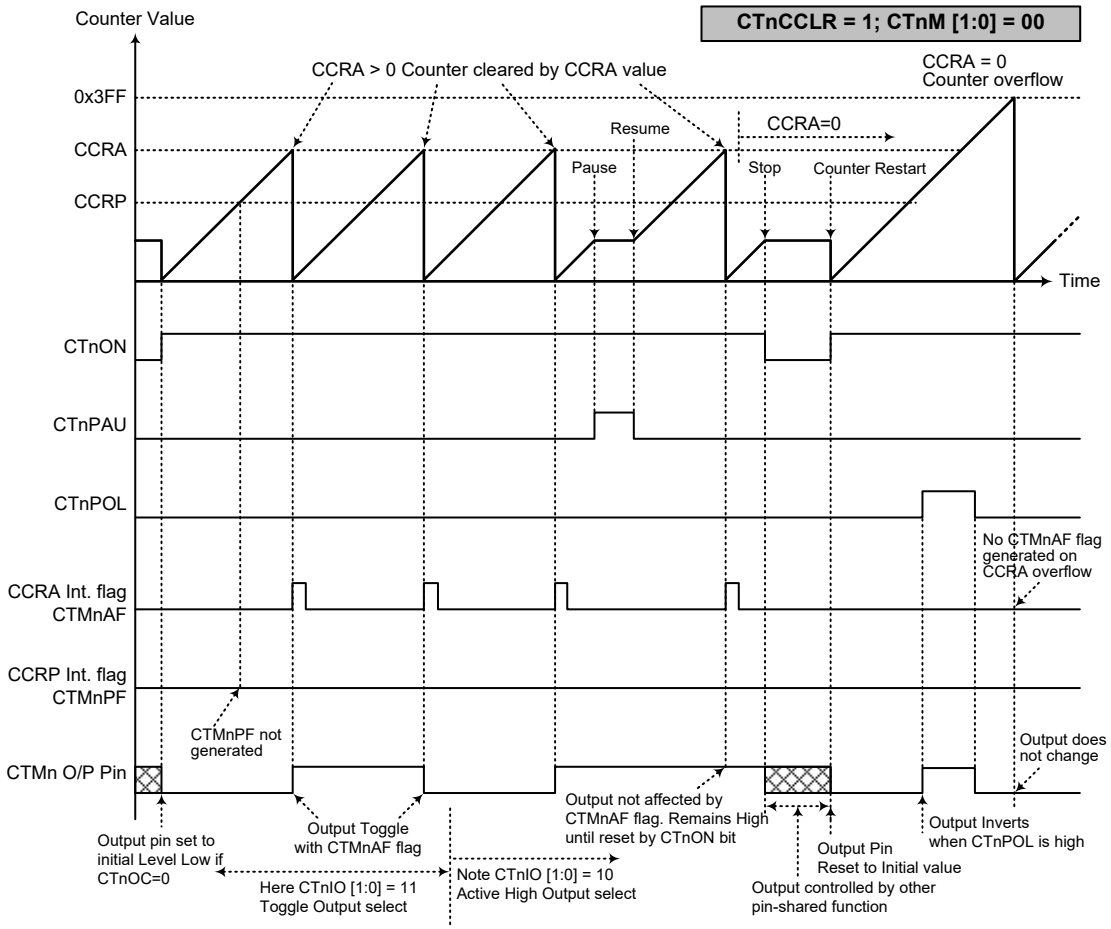
如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 CTMnAF 请求标志。

正如该模式名所言，当比较匹配发生后，CTMn 输出脚状态改变。当比较器 A 比较匹配发生后 CTMnAF 标志产生时，CTMn 输出脚状态改变。比较器 P 比较匹配发生时产生的 CTMnPF 标志不影响 CTMn 输出脚。CTMn 输出脚状态改变方式由 CTMnC1 寄存器中 CTnIO1 和 CTnIO0 位决定。当比较器 A 比较匹配发生时，CTnIO1 和 CTnIO0 位决定 TM 输出脚输出高，低或翻转当前状态。CTMn 输出脚初始值，在 CTnON 位由低到高电平的变化后通过 CTnOC 位设置。注意，若 CTnIO1 和 CTnIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 -- CTnCCLR=0

注：1. CTnCCLR=0，比较器 P 匹配将清除计数器
2. CTMn 输出脚仅由 CTMnAF 标志位控制
3. 在 CTnON 上升沿 CTMn 输出脚复位至初始值
4. n=0 或 1



比较匹配输出模式 -- CTnCCLR=1

注：1. CTnCCLR=1，比较器 A 匹配将清除计数器
2. CTMn 输出脚仅由 CTMnAF 标志位控制
3. 在 CTnON 上升沿 CTMn 输出脚复位至初始值
4. 当 CTnCCLR=1 时，CTMnPF 标志位不会产生
5. n=0 或 1

定时 / 计数器模式

为使 CTMn 工作在此模式，CTMnC1 寄存器中的 CTnM1 和 CTnM0 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 CTMn 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 CTMn 输出脚可用作普通 I/O 引脚或其它功能。

PWM 输出模式

为使 CTMn 工作在此模式，CTMnC1 寄存器中的 CTnM1 和 CTnM0 位需要设置为“10”。CTMn 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 CTMn 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就极其灵活。在 PWM 输出模式中，CTnCCLR 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 CTMnC1 寄存器的 CTnDPX 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。CTMnC1 寄存器中的 CTnOC 位决定 PWM 波形的极性，CTnIO1 和 CTnIO0 位使能 PWM 输出或将 CTMn 输出脚置为逻辑高或逻辑低。CTnPOL 位对 PWM 输出波形的极性取反。

- **10-bit CTMn，PWM 输出模式，边沿对齐模式，CTnDPX=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，CTMn 时钟源选择 f_{SYS}/4，CCRP=100b，CCRA=128，

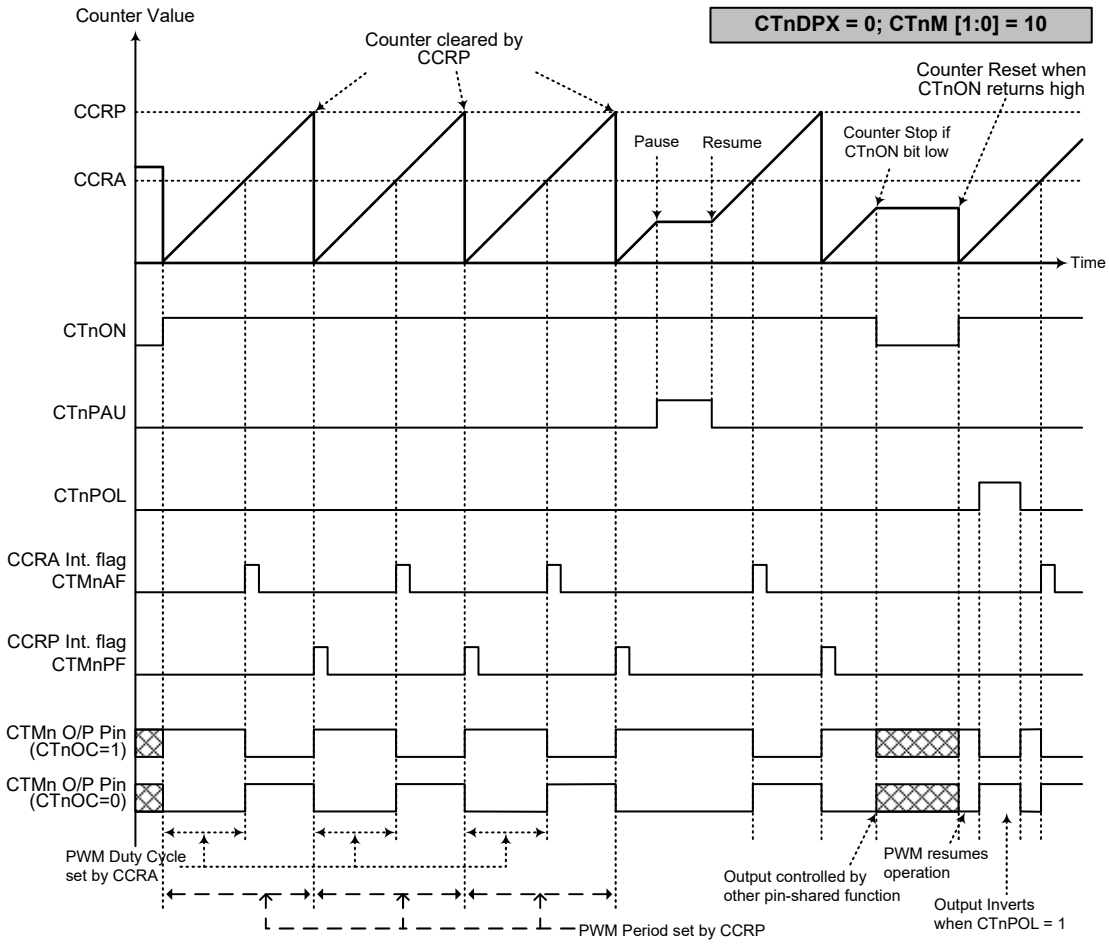
CTMn PWM 输出频率 = (f_{SYS}/4)/512=f_{SYS}/2048=7.8125kHz，duty=128/512=25%

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 CTMn，PWM 输出模式，边沿对齐模式，CTnDPX=1**

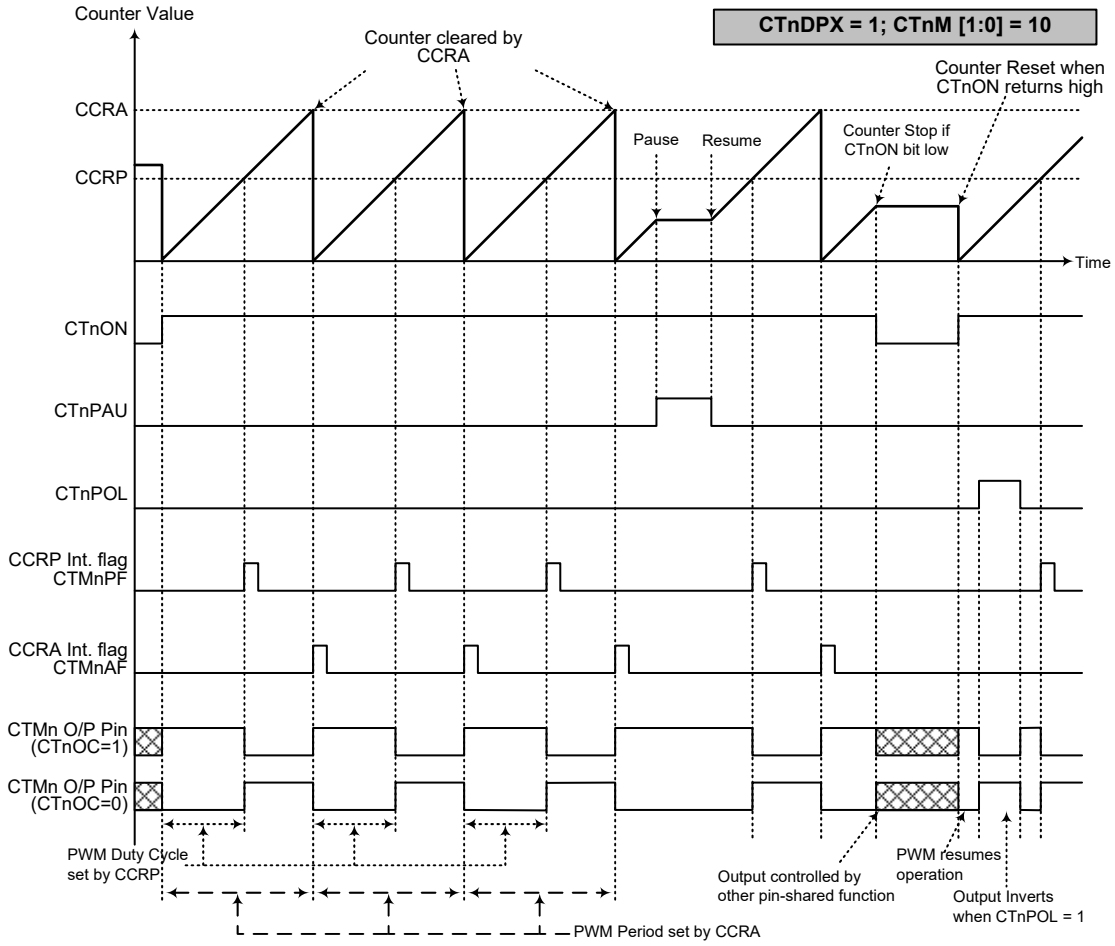
CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

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



PWM 输出模式 -- CTnDPX=0

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

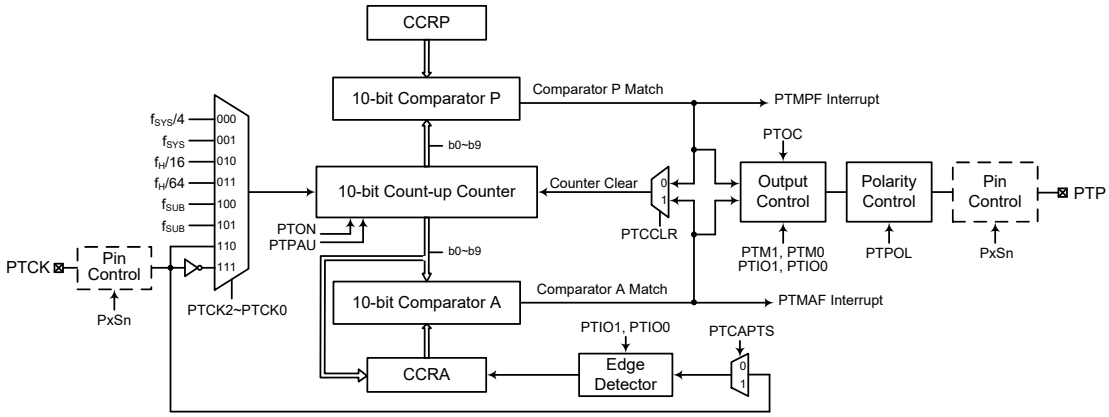


PWM 输出模式 -- CTnDPX=1

注：1. CTnDPX=1，CCRA 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 CTnIO1，CTnIO0=00 或 01，PWM 功能不变
4. CTnCCLR 位不影响 PWM 操作
5. n=0 或 1

周期型 TM – PTM

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



注：当 PTM 工作在捕捉输入模式时，PTCAPTS 位必须设为 1 选择 PTCK 捕捉输入。

周期型 TM 方框图

周期型 TM 操作

周期型 TM 是 10 位宽度。周期型 TM 核心是一个由用户选择的内部或外部时钟源驱动的 10 位向上计数器，它还包括两个内部比较器即比较器 A 和比较器 P。这两个比较器将计数器的值与 CCRA 和 CCRP 寄存器中的值进行比较。CCRP 和 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 PTON 位发生上升沿跳变清除计数器。此外，计数器溢出或比较匹配也会自动清除计数器。上述条件发生时，通常情况会产生 PTM 中断信号。周期型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，也可以控制输出脚。所有工作模式的设定都是通过设置相关寄存器来实现的。

周期型 TM 寄存器介绍

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	—	—	—	—	—	—	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	—	—	—	—	—	—	D9	D8
PTMRPL	PTRP7	PTRP6	PTRP5	PTRP4	PTRP3	PTRP2	PTRP1	PTRP0
PTMRPH	—	—	—	—	—	—	PTRP9	PTRP8

10-bit 周期型 TM 寄存器列表

- **PTMC0 寄存器**

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

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

Bit 6~4 **PTCK2~PTCK0**：选择 PTM 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：PTCK 上升沿
111：PTCK 下降沿
此三位用于选择 PTM 的时钟源。外部引脚时钟源能被选择在上升沿或下降沿有效。f_{SYS} 是系统时钟，f_H 和 f_{SUB} 是其它的内部时钟源，细节方面请参考振荡器章节

Bit 3 **PTON**：PTM 计数器 On/Off 控制位
0：Off
1：On
此位控制 PTM 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 PTM。清零此位将停止计数器并关闭 PTM 减少耗电。当此位经由低到高转变时，内部计数器将复位清零；当此位经由高到低转换时，内部计数器将保持其剩余值，直到此位再次改变为高电平。
若 PTM 处于比较匹配输出模式、PWM 输出模式或单脉冲输出模式，当 PTON 位经由低到高转换时，PTM 输出脚将复位至 PTOC 位指定的初始值。

Bit 2~0 未定义，读为“0”

- **PTMC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PTM1~PTM0**：选择 PTM 工作模式位
00：比较匹配输出模式
01：捕捉输入模式
10：PWM 输出模式或单脉冲输出模式
11：定时 / 计数器模式
这两位设置 PTM 需要的工作模式。为了确保操作可靠，PTM 应在 PTM1 和 PTM0 位有任何改变前先关掉。在定时 / 计数器模式，PTM 输出脚控制必须除能。

Bit 5~4 **PTIO1~PTIO0**：选择 PTM 功能位
比较匹配输出模式
00：无变化
01：输出低
10：输出高
11：输出翻转
PWM 输出模式 / 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 PTCK 上升沿输入捕捉
01：在 PTCK 下降沿输入捕捉
10：在 PTCK 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 PTM 输出脚如何改变状态。这两位值的选择决定 PTM 运行在哪种模式下。
在比较匹配输出模式下，PTIO1 和 PTIO0 位决定当从比较器 A 比较匹配输出发生时 PTM 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 PTM 输出脚能设为切换高、切换低或翻转当前状态。若此两位同时为 0 时，这个输出将不会改变。PTM 输出脚的初始值通过 PTMC1 寄存器的 PTOC 位设置取得。注意，由 PTIO1 和 PTIO0 位得到的输出电平必须与通过 PTOC 位设置的初始值不同，否则当比较匹配发生时，PTM 输出脚将不会发生变化。在 PTM 输出脚改变状态后，通过 PTON 位由低到高电平的转换复位至初始值。
在 PWM 输出模式，PTIO1 和 PTIO0 用于决定比较匹配条件发生时怎样改变 PTM 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。仅在 PTM 关闭时改变 PTIO1 和 PTIO0 位的值是很有必要的。若在 PTM 运行时改变 PTIO1 和 PTIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **PTOC**：PTM PTP 输出控制位
比较匹配输出模式
0：初始低
1：初始高
PWM 输出模式 / 单脉冲输出模式
0：低有效
1：高有效

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

Bit 2 **PTPOL**：PTM PTP 输出极性控制位
0：同相
1：反相
此位控制 PTP 输出脚的极性。此位为高时 PTM 输出脚反相，为低时 PTM 输出脚同相。若 PTM 处于定时 / 计数器模式时其不受影响。

Bit 1 **PTCAPTS**：选择 PTM 捕捉触发源
0：未定义
1：来自 PTCK 引脚
当 PTM 工作于捕捉输入模式时，此位必须设置为 1。

Bit 0 **PTCCLR**：选择 PTM 计数器清零条件位
0：PTM 比较器 P 匹配
1：PTM 比较器 A 匹配
此位用于选择清除计数器的方法。周期型 PTM 包括两个比较器 -- 比较器 A 和比较器 P，两者都可以用作清除内部计数器。PTCCLR 位设为高，计数器在比较器 A 比较匹配发生时被清除；此位设为低，计数器在比较器 P 比较匹配发生或计数器溢出时被清除。计数器溢出清除的方法仅在 CCRP 被清除为 0 时才能生效。PTCCLR 位在 PWM 输出模式、单脉冲输出模式或捕捉输入模式时未使用。

- **PTMDL 寄存器**

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

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

- **PTMDH 寄存器**

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

Bit 7~2 未定义，读为“0”

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

- **PTMAL 寄存器**

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

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

- **PTMAH 寄存器**

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

Bit 7~2　未定义，读为“0”

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

- **PTMRPL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PTRP7	PTRP6	PTRP5	PTRP4	PTRP3	PTRP2	PTRP1	PTRP0
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　**PTRP7~PTRP0**：PTM CCRP 低字节寄存器 bit 7 ~ bit 0
PTM 10-bit CCRP bit 7 ~ bit 0

- **PTMRPH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PTRP9	PTRP8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未定义，读为“0”

Bit 1~0　**PTRP9~PTRP8**：PTM CCRP 高字节寄存器 bit 1 ~ bit 0
PTM 10-bit CCRP bit 9 ~ bit 8

周期型 TM 工作模式

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

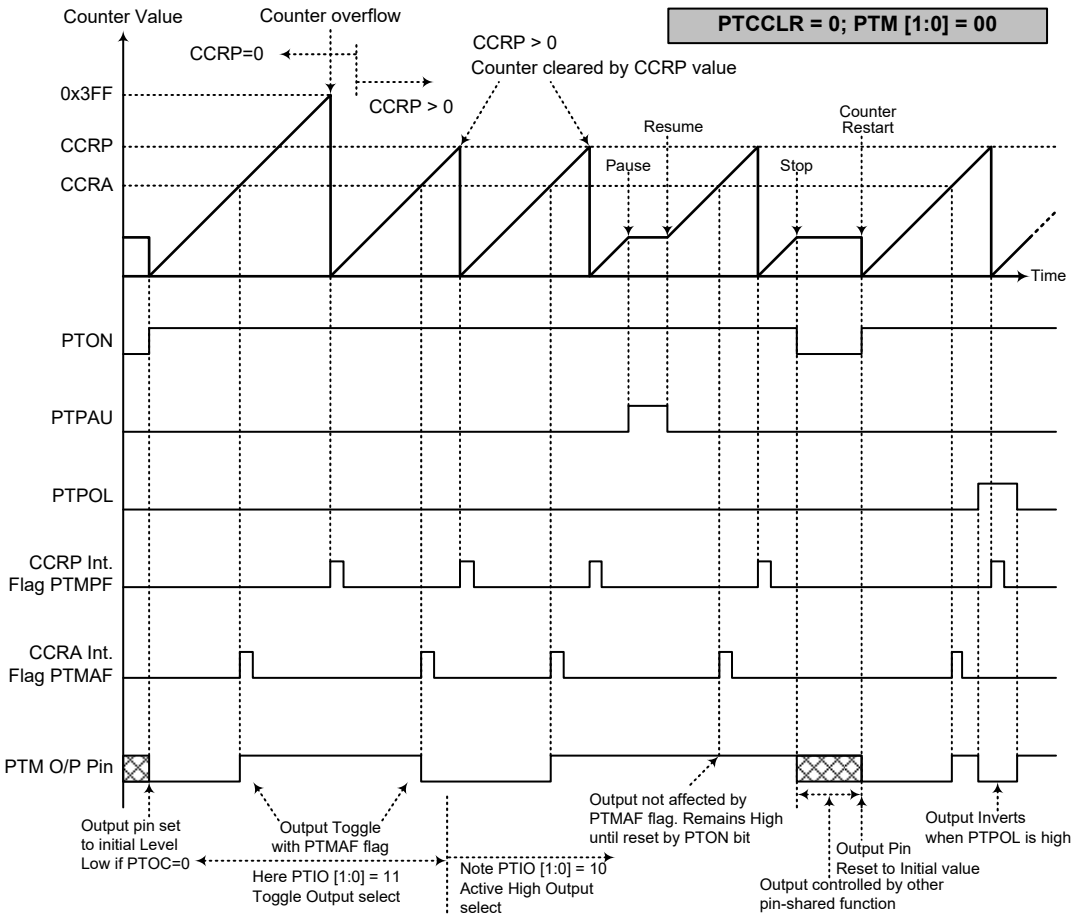
比较匹配输出模式

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

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

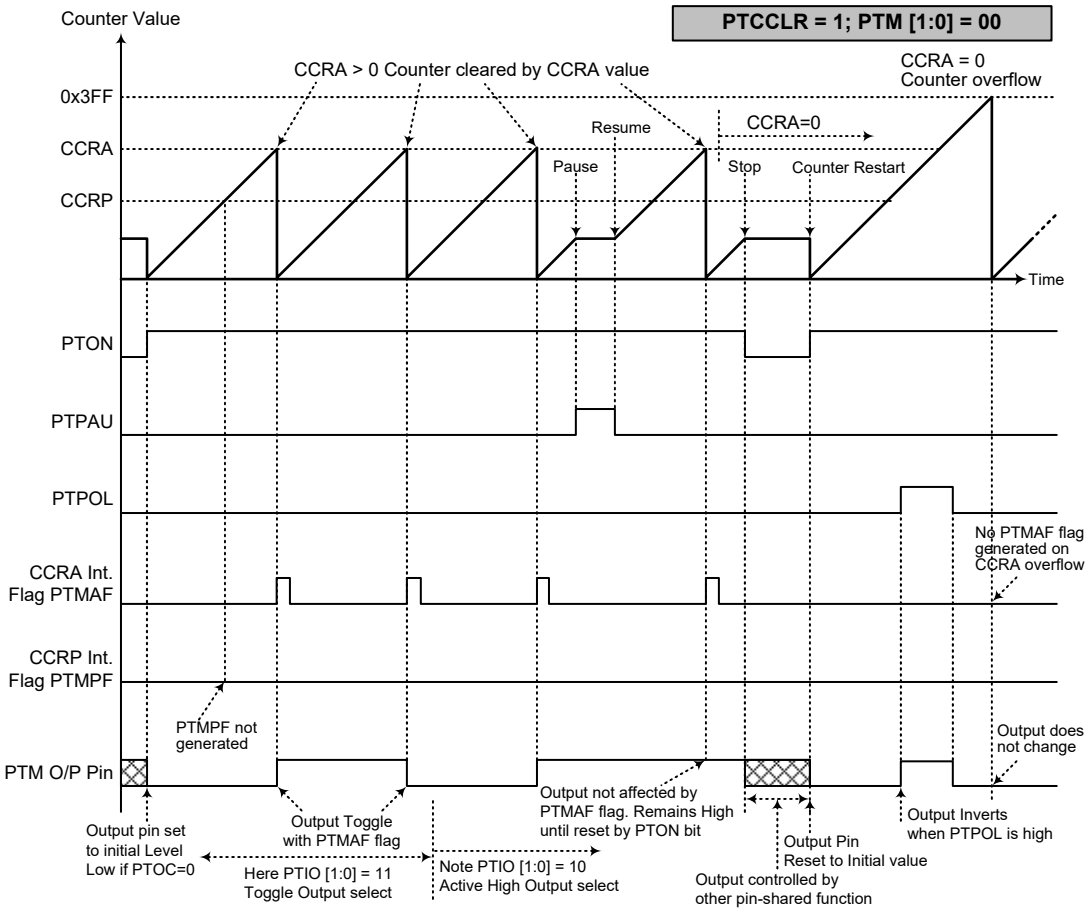
如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 PTMAF 请求标志。

正如该模式名所言，当比较匹配发生后，PTM 输出脚状态改变。当比较器 A 比较匹配发生后 PTMAF 中断请求标志产生时，PTM 输出脚状态改变。比较器 P 比较匹配发生时产生的 PTMPF 标志不影响 PTM 输出脚。PTM 输出脚状态改变方式由 PTMC1 寄存器中 PTIO1 和 PTIO0 位决定。当比较器 A 比较匹配发生时，PTIO1 和 PTIO0 位决定 PTM 输出脚输出高，低或翻转当前状态。PTM 输出脚初始值，在 PTON 位由低到高电平的变化后通过 PTOC 位设置。注意，若 PTIO1 和 PTIO0 位同时为 0 时，引脚输出不变。



比较器匹配输出模式 -- PTCCLR = 0

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



比较器匹配输出模式 -- **PTCCLR = 1**

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

定时 / 计数器模式

为使 PTM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 位需要设置为“11”。定时 / 计数器模式与比较输出模式操作方式相同，并产生同样的中断请求标志。不同的是，在定时 / 计数器模式下 PTM 输出脚未使用。因此，比较匹配输出模式中的描述和时序图可以适用于此功能。该模式中未使用的 PTM 输出脚可用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 PTM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 位需要设置为“10”，且 PTIO1 和 PTIO0 位也需要设置为“10”。PTM 的 PWM 功能在马达控制，加热控制，照明控制等方面十分有用。给 PTM 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就极其灵活。在 PWM 输出模式中，PTCCLR 位对 PWM 周期无影响。CCRP 和 CCRA 寄存器都用于控制 PWM 方波。CCRP 寄存器通过清除内部计数从而控制 PWM 周期，CCRA 寄存器设置 PWM 的占空比。PWM 波形的周期和占空比由 CCRP 和 CCRA 寄存器的值控制。

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

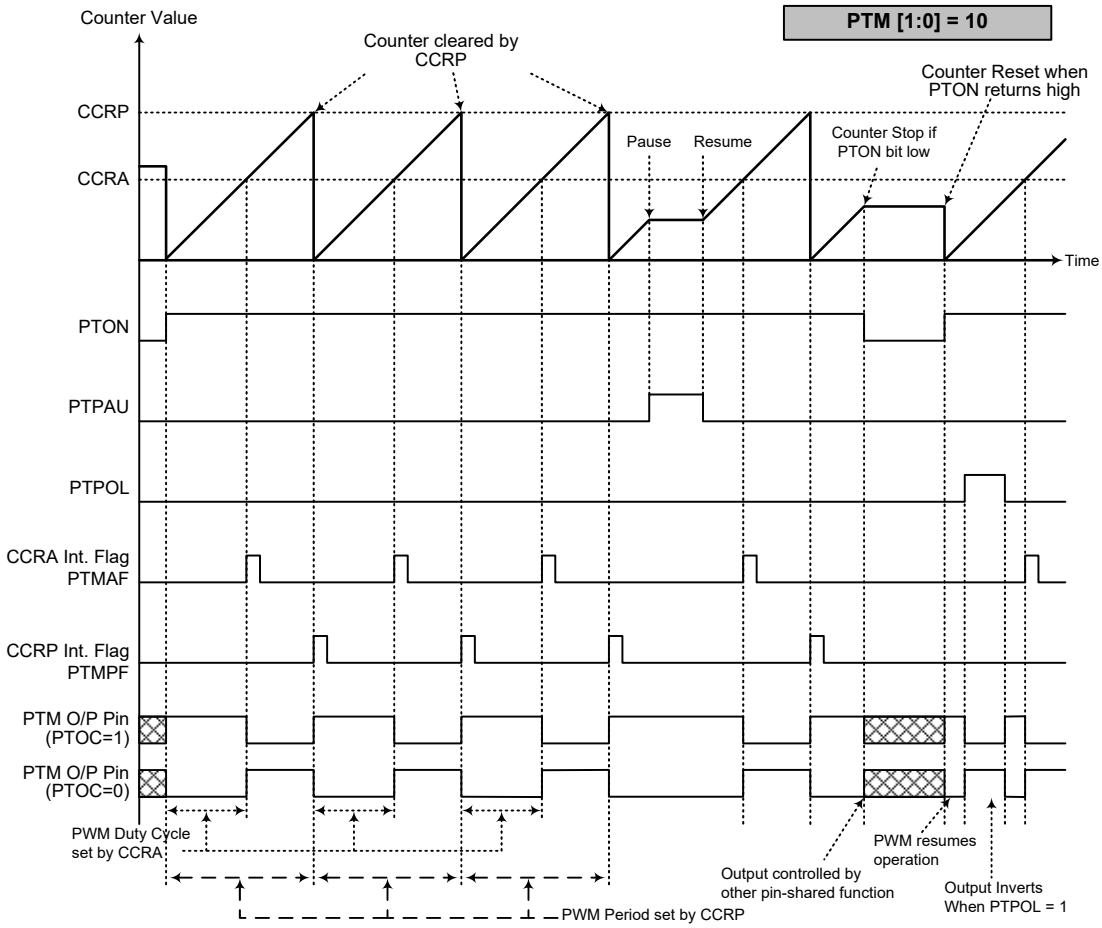
- **10-bit PTM，PWM 输出模式，边沿对齐模式**

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

若 f_{SYS}=16MHz，PTM 时钟源选择 f_{SYS}/4，CCRP=512 且 CCRA=128，

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

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



PWM 输出模式

注：1. CCRP 清除计数器

2. 计数器清除并决定 PWM 周期

3. 当 PTIO[1:0]=00 或 01，PWM 功能不变

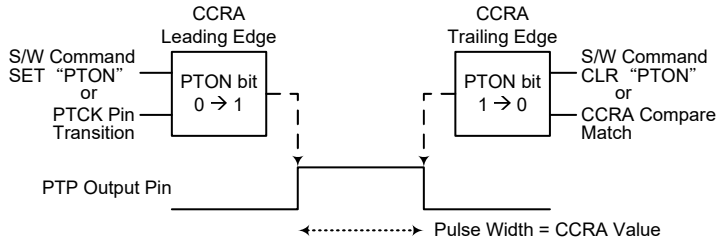
4. PTCCLR 位对 PWM 功能无影响

单脉冲输出模式

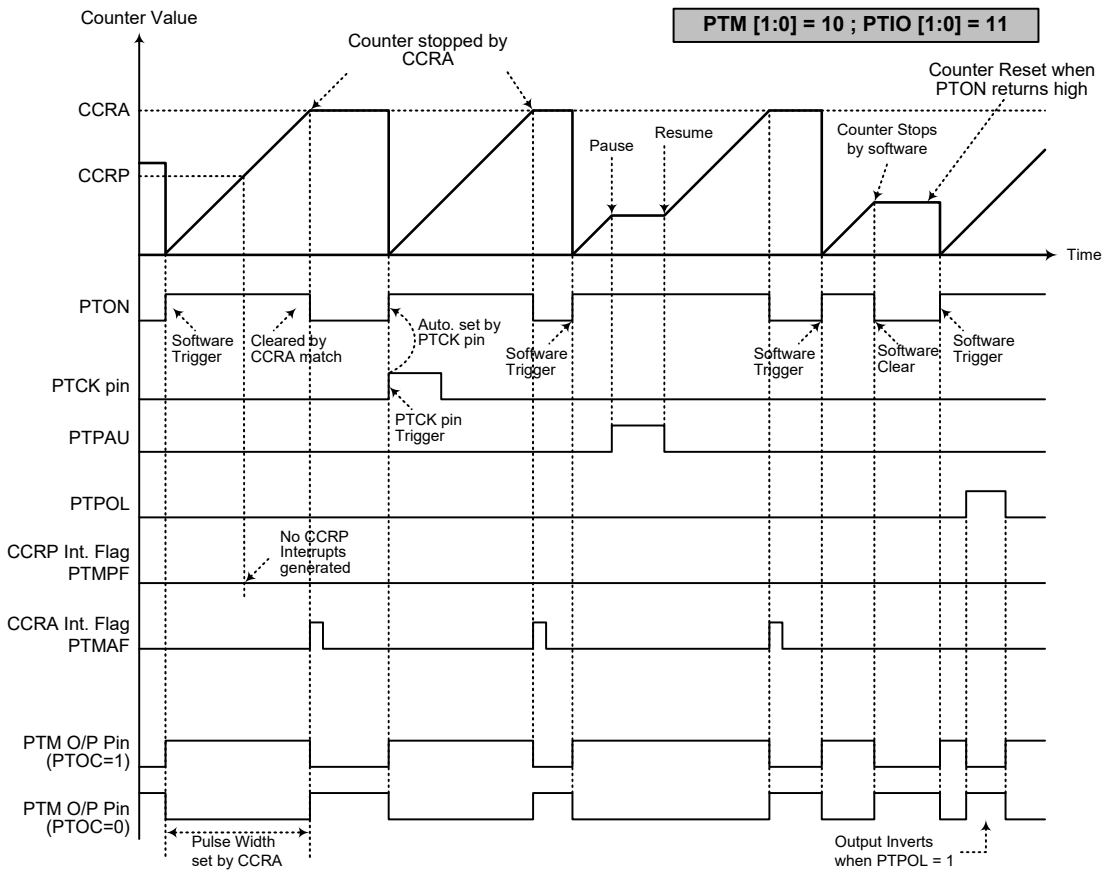
为使 PTM 工作在此模式，PTMC1 寄存器中的 PTM1 和 PTM0 位需要设置为“10”，并且相应的 PTIO1 和 PTIO0 需要设置为“11”。正如模式名所言，单脉冲输出模式，在 PTM 输出脚将产生一个脉冲输出。

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

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



单脉冲产生示意图



单脉冲输出模式

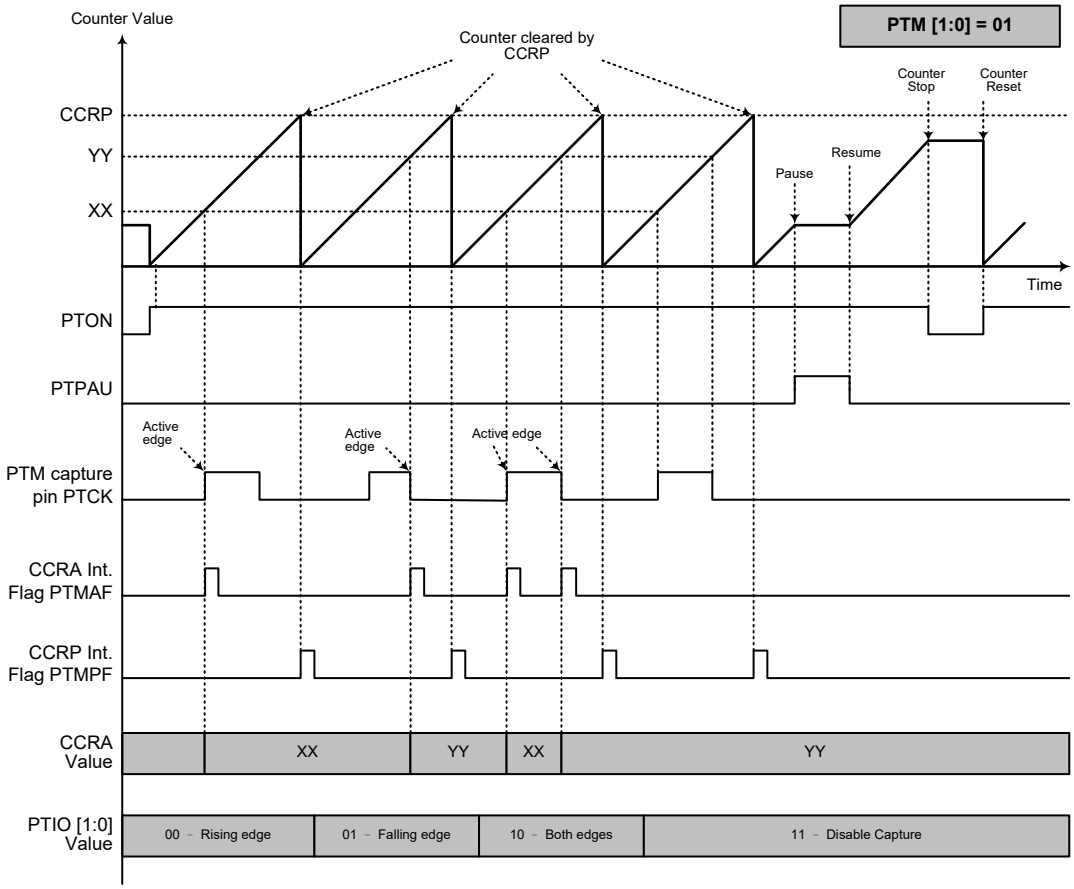
注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 PTCK 脚或设置 PTON 位为高来触发脉冲
4. PTCK 脚有效沿会自动置位 PTON
5. 单脉冲输出模式中，PTIO[1:0] 需置位“11”，且不能更改。

捕捉输入模式

为使 PTM 工作在此模式，PTMC1 寄存器的 PTM1 和 PTM0 位需要设置为“01”。此模式使能外部信号捕捉并保存内部计数器当前值，因此被用于诸如脉冲宽度测量的应用中。通过设置 PTMC1 寄存器的 PTCAPTS 位为 1 选择 PTCK 引脚上的外部信号。可通过设置 PTMC1 寄存器的 PTIO1 和 PTIO0 位选择有效边沿类型，即上升沿，下降沿或双沿有效。通过应用程序将 PTON 位由低到高转变时，计数器启动。

当 PTCK 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 PTM 中断。无论 PTCK 引脚发生哪种边沿转换，计数器将继续工作直到 PTON 位发生下降沿跳变。当 CCRP 比较匹配发生时计数器复位至零；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 PTM 中断。记录 CCRP 溢出中断信号的值可以测量长脉宽。通过设置 PTIO1 和 PTIO0 位选择 PTCKn 引脚为上升沿，下降沿或双沿有效。如果 PTIO1 和 PTIO0 位都设置为高，无论 PTCK 引脚发生哪种边沿转换都不会产生捕捉操作，但计数器仍会继续运行。

当 PTCK 引脚与其它功能共用，PTM 工作在输入捕捉模式时需多加注意。这是因为如果引脚被设为输出，那么该引脚上的任何电平转变都可能执行输入捕捉操作。PTCCLR，PTOC 和 PTPOL 位在此模式中未使用。



捕捉输入模式

注：1. PTM[1:0]=01 并通过 PTIO[1:0] 位设置有效边沿
2. PTM 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中
3. PTCCLR 位未使用
4. 无输出功能 – PTOC 和 PTPOL 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

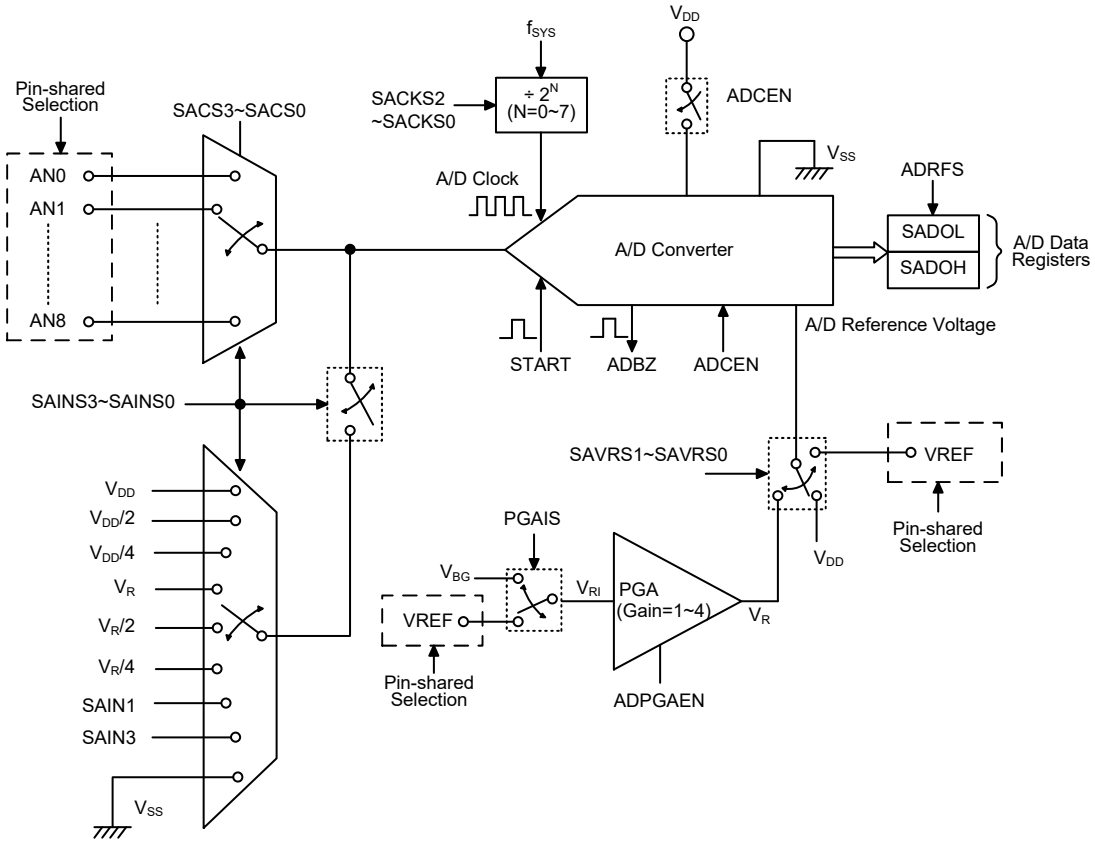
对于大多数电子系统而言，处理现实世界的模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先需要通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

A/D 转换器简介

此单片机包含一个多通道的 A/D 转换器，它们可以直接接入外部模拟信号 (来自传感器或其它控制信号) 或内部模拟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 SAINS3~SAINS0 位和 SACS3~SACS0 位共同控制。注意，若要转换内部模拟信号时，已选的外部输入通道会自动断开以避免故障。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 输入引脚”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。

外部输入通道	内部输入信号	A/D 通道选择位
9：AN0~AN8	8：V_{DD}，$V_{DD}/2$，$V_{DD}/4$，V_R，$V_R/2$，$V_R/4$，SAIN1，SAIN3	SAINS3~SAINS0，SACS3~SACS0



注：内部信号 SAIN1 和 SAIN3 的具体描述请参考 SADC1 寄存器。

A/D 转换器结构

A/D 转换寄存器介绍

A/D 转换器的所有工作由一系列寄存器控制。一对只读寄存器来存放 12 位 A/D 转换数据的值。剩下三个控制寄存器设置 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
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
SADC2	ADPGAEN	—	—	PGAIS	SAVRS1	SAVRS0	PGAGS1	PGAGS0

A/D 转换寄存器列表

A/D 转换器数据寄存器 – SADOL，SADOH

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

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

A/D 数据寄存器

A/D 转换器控制寄存器 – SADC0，SADC1，SADC2

寄存器 SADC0、SADC1 和 SADC2 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 转换器的忙碌状态。由于每个单片机只包含一个实际的模数转换电路，因此这些外部和内部模拟信号中的每一个都需要分别被发送到转换器。SADC0 寄存器中的 SACS3~SACS0 位用于选择哪个外部模拟输入通道被连接到内部 A/D 转换器。SADC1 寄存器中的 SAINS3~SAINS0 位用于选择外部模拟输入通道或内部模拟信号被连接到内部 A/D 转换器。

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

- **SADC0 寄存器**

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

Bit 7 **START**：启动 A/D 转换位
0→1→0：启动
此位用于初始化 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。

Bit 6 **ADBZ**：A/D 转换忙碌标志位
0：A/D 转换结束或未开始转换
1：A/D 转换中
此位用于表明 A/D 转换过程是否完成。当 START 位由低变为高再变为低时，ADBZ 位为高，表明 A/D 转换已初始化。A/D 转换结束后，此位被清零。

Bit 5 **ADCEN**：A/D 转换器使能 / 除能控制位
0：除能
1：使能
此位控制 A/D 内部功能。该位被置高将使能 A/D 转换器。如果该位设为低将关闭 A/D 转换器以降低功耗。当 A/D 转换器除能时，A/D 数据寄存器 SADOH 和 SADOL 的内容将保持不变。

Bit 4 **ADRFS**：A/D 转换数据格式选择位
0：A/D 转换数据格式 →SADOH=D[11:4]; SADOL=D[3:0]
1：A/D 转换数据格式 →SADOH=D[11:8]; SADOL=D[7:0]
此位控制存放在两个 A/D 数据寄存器中的 12 位 A/D 转换结果的格式。细节方面请参考 A/D 数据寄存器章节。

Bit 3~0 **SACS3~SACS0**：A/D 外部模拟通道输入选择位
0000：AN0
0001：AN1
0010：AN2
0011：AN3
0100：AN4
0101：AN5
0110：AN6
0111：AN7
1000：AN8
1001~1111：浮空

- **SADC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
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 7~4 **SAINS3~SAINS0**：A/D 输入信号选择位
0000：外部信号 – 外部模拟通道输入
0001：内部信号 – 内部 A/D 转换器电源电压 V_{DD}
0010：内部信号 – 内部 A/D 转换器电源电压 $V_{DD}/2$
0011：内部信号 – 内部 A/D 转换器电源电压 $V_{DD}/4$
0100：外部信号 – 外部模拟通道输入
0101：内部信号 – PGA 输出电压 V_R
0110：内部信号 – PGA 输出电压 $V_R/2$
0111：内部信号 – PGA 输出电压 $V_R/4$
1000：内部信号 – SAIN1：OPA 输出
1001：接地
1010：内部信号 – SAIN3：恒定电流单位增益缓冲输出
1011：接地
1100~1111：外部信号 – 外部模拟通道输入
当选择内部模拟信号时，无论 SACS3~SACS0 为何值，外部通道输入信号都会自动关闭。内部参考电压可通过 SADC2 寄存器中的 SAVRS1~SAVRS0 位选择不同的电压源。

Bit 3 未定义，读为“0”

Bit 2~0 **SACKS2~SACKS0**：A/D 时钟源选择位
000：f_{SYS}
001：$f_{SYS}/2$
010：$f_{SYS}/4$
011：$f_{SYS}/8$
100：$f_{SYS}/16$
101：$f_{SYS}/32$
110：$f_{SYS}/64$
111：$f_{SYS}/128$
这三位用于选择 A/D 转换器的时钟源。

- **SADC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ADPGAEN	—	—	PGAIS	SAVRS1	SAVRS0	PGAGS1	PGAGS0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7 **ADPGAEN**：A/D 转换器 PGA 功能使能 / 除能控制位
0：除能
1：使能
该位用于控制 A/D 转换器的内部 PGA 功能以提供不同的参考电压。当该位被置高，内部参考电压 V_R 可作为 A/D 转换器的内部转换信号或参考电压。如果内部参考电压不用于 A/D 转换器，应正确配置 PGA 功能以减小功耗。

Bit 6~5 未定义，读为“0”

Bit 4 **PGAIS**：PGA 输入 (V_{RI}) 选择位
0：VREF 引脚
1：V_{BG}
当选择内部信号输入时，外部 VREF 引脚的参考电压输入会被自动断开。另外，需通过设置 LVDC 寄存器中的 VBGEN 位为高使能内部参考电压 V_{BG}。

Bit 3~2 **SAVRS1~SAVRS0**：A/D 转换器参考电压选择位
00：V_{DD}
01：VREF 引脚
1x：V_R
当选择内部参考电压时，外部 VREF 引脚的参考电压输入会被自动断开。

Bit 1~0 **PGAGS1~PGAGS0**：PGA 增益选择位
00：增益 = 1
01：增益 = 2
10：增益 = 3
11：增益 = 4

A/D 转换器操作

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

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

A/D 转换器的时钟源为系统时钟 f_{SYS} 或其分频，而分频系数由 SADC1 寄存器中的 SACKS2~SACKS0 位决定。虽然 A/D 时钟源是由系统时钟 f_{SYS} 和 SACKS2~SACKS0 位决定，但可选择的最大 A/D 时钟源则有一些限制。由于允许的 A/D 时钟周期 t_{ADCK} 的范围为 0.5μs~10μs，所以选择系统时钟速度时就必须小心。必须保证设置的 A/D 转换时钟周期不小于时钟周期的最小值或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。

f_{SYS}	A/D 时钟周期 (t_{ADCK})							
	SACKS[2:0] = 000 (f_{SYS})	SACKS[2:0] = 001 ($f_{SYS}/2$)	SACKS[2:0] = 010 ($f_{SYS}/4$)	SACKS[2:0] = 011 ($f_{SYS}/8$)	SACKS[2:0] = 100 ($f_{SYS}/16$)	SACKS[2:0] = 101 ($f_{SYS}/62$)	SACKS[2:0] = 110 ($f_{SYS}/64$)	SACKS[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

A/D 时钟周期范例

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

A/D 转换器参考电压

A/D 转换器参考电压可以来自正电源电压 V_{DD}、外部参考源引脚 VREF 或 PGA 输出电压 V_R，可通过 SADC2 寄存器中的 SAVRS1~SAVRS0 位来选择。当 SAVRS1~SAVRS0 位设为“00”，参考电压来自 V_{DD}。当 SAVRS1~SAVRS0 位设为“01”，参考电压来自 VREF 引脚。当 SAVRS1~SAVRS0 位设为“1x”，参考电压来自 PGA 输出，V_R。由于 VREF 引脚都与其它功能共用，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共用引脚功能。另外，当外部参考电压引脚 VREF 和内部参考电压，V_{DD} 或 V_R，同时接入 A/D 转换器时，硬件自动选择内部参考电压且断开外部参考电压输入。

当选择 A/D 参考电压来自 PGA 输出时，PGA 的输入需通过设置 SADC2 寄存器的 PGAIS 位选择来自外部 VREF 引脚或内部参考电压 V_{BG}。当外部 VREF 引脚和内部 V_{BG} 信号同时接入 PGA 输入端时，硬件会自动选择内部信号作为 PGA 输入并断开外部 VREF 引脚。

模拟输入值一定不能超过所选的参考电压值。

SAVRS[1:0]	参考电压源	说明
00	V_{DD}	内部 A/D 转换器电源电压
01	VREF 引脚	外部 A/D 转换器参考电压引脚 VREF
10 或 11	V_R	内部 A/D 转换器 PGA 输出电压

A/D 转换器参考电压选择

A/D 转换器输入信号

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

若 SAINS3~SAINS0 位为“0000”、“0100”或“1100~1111”，则选择转换外部模拟输入信号，具体通道编号由 SACS3~SACS0 位决定。若 SAINS3~SAINS0 位为“0001~1000”或“1010”，则选择转换内部模拟信号，这些内部信号可来自于 A/D 转换器电源 V_{DD} 分压、PGA 输出电压 V_R 分压、OPA 输出或恒定电流单位增益缓冲输出。当选择内部模拟信号时，外部输入通道会自动关闭以避免信号冲突。

SAINS[3:0]	SACS[3:0]	输入信号	描述
0000, 0100, 1100~1111	0000~1000	AN0~AN8	外部模拟通道输入
	1001~1111	–	浮空
0001	xxxx	V_{DD}	A/D 转换器电源电压
0010	xxxx	$V_{DD}/2$	A/D 转换器电源电压 /2
0011	xxxx	$V_{DD}/4$	A/D 转换器电源电压 /4
0101	xxxx	V_R	A/D 转换器 PGA 输出电压
0110	xxxx	$V_R/2$	A/D 转换器 PGA 输出电压 /2
0111	xxxx	$V_R/4$	A/D 转换器 PGA 输出电压 /4
1000	xxxx	SAIN1	OPA 输出
1001, 1011	xxxx	GND	接地
1010	xxxx	SAIN3	恒定电流单位增益缓冲输出

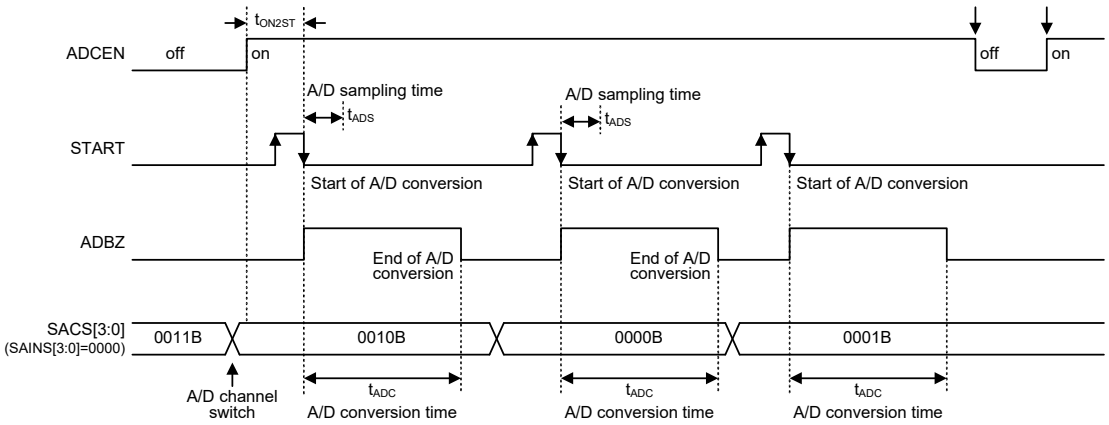
A/D 转换器输入信号选择

A/D 转换率及时序图

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

最大 A/D 转换率 = A/D 时钟周期 ÷ 16

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



A/D 转换时序图 – 外部模拟通道输入

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 SADC1 寄存器中的 SACKS2~SACKS0 位，选择所需的 A/D 转换时钟。
- 步骤 2
  将 SADC0 寄存器中的 ADCEN 位置高使能 A/D 转换器。
- 步骤 3
  通过 SADC1 寄存器中的 SAINS3~SAINS0 位，选择连接至内部 A/D 转换器的信号。
  若选择外部通道输入，接着执行步骤 4。
  若选择内部模拟信号，接着执行步骤 5。
- 步骤 4
  若已通过 SAINS3~SAINS0 位选择 A/D 输入信号来自外部通道输入，接着应设置相关的引脚共用控制位将该引脚规划为 A/D 输入引脚。通过设置 SACS3~SACS0 位选择哪个外部通道接至 A/D 转换器。接着执行步骤 6。
- 步骤 5
  若已通过 SAINS3~SAINS0 位选择 A/D 输入信号来自内部模拟信号，无论 SACS3~SACS0 为何值，外部通道输入都会自动断开。接着执行步骤 6。
- 步骤 6
  通过 SADC2 寄存器的 SAVRS1~SAVRS0 位选择参考电压。若选择参考电压来自 PGA 输出，必须使能 PGA 并且通过 SADC2 寄存器中的 PGAIA 位选择 PGA 输入源。
- 步骤 7
  设置 SADC0 寄存器的 ADRFS 位选择 A/D 转换器输出数据格式。
- 步骤 8
  如果要使用中断，则中断控制寄存器需要正确地设置，以确保 A/D 中断功能是激活的。总中断控制位 EMI 需要置位为“1”，A/D 转换器中断位 ADE 以及相应的多功能中断使能位也需要置位为“1”。
- 步骤 9
  现在可以通过设置 SADC0 寄存器中的 START 位从“0”到“1”再回到“0”，开始模数转换的过程。

- 步骤 10

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

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

编程注意事项

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

A/D 转换功能

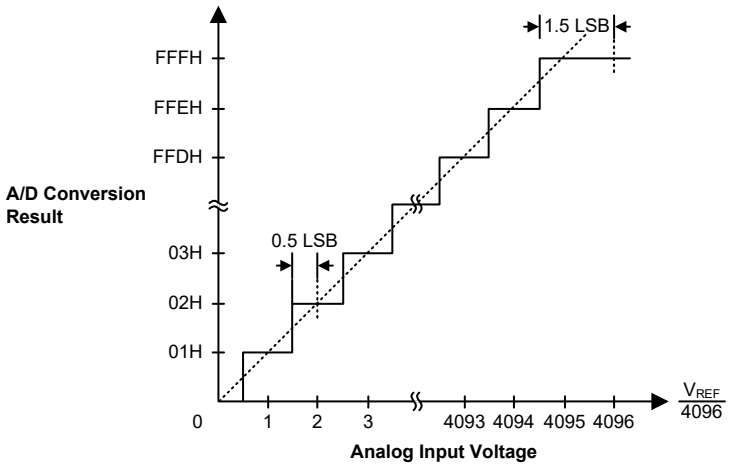
单片机含有一组 12 位的 A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于实际 A/D 转换器参考电压 V_{REF} 的值，因此每一位可表示 $V_{REF}/4096$ 的模拟输入值。

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

通过下面的等式可估算 A/D 转换器输入电压值：

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

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



理想的 A/D 转换功能

A/D 转换应用范例

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

范例 1：使用查询 ADBZ 的方式来检测转换结束

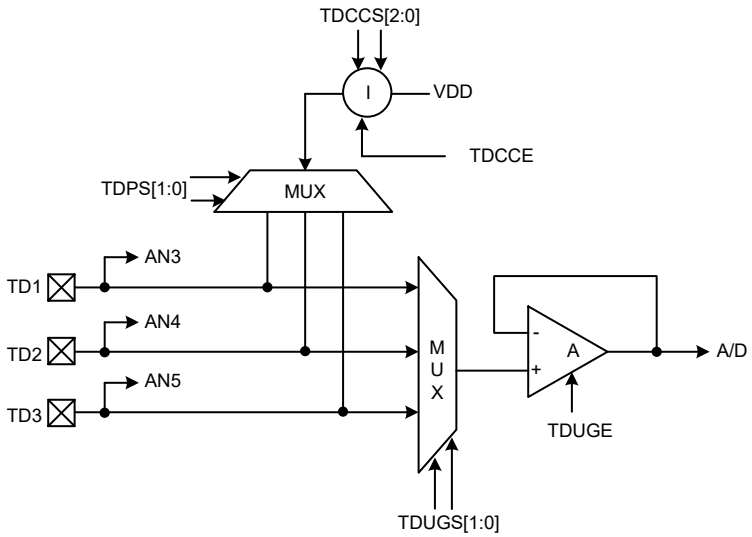
```
clr ADE                    ; disable ADC interrupt
mov a,03H
mov SADC1,a                ; select fSYS/8 as A/D clock
set ADCEN
mov a,c0H                  ; setup PCS0 to configure pin AN0
mov PCS0,a
mov a,20H
mov SADC0,a                ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr START                  ; high pulse on start bit to initiate conversion
set START                  ; reset A/D
clr START                  ; start A/D
:
polling_EOC:
sz  ADBZ                   ; poll the SADC0 register ADBZ bit to detect end
                           ; of A/D conversion
jmp polling_EOC            ; continue polling
:
mov a,SADOL                ; read low byte conversion result value
mov sadol_buffer,a         ; save result to user defined register
mov a,SADOH                ; read high byte conversion result value
mov sadoh_buffer,a         ; save result to user defined register
:
jmp start_conversion       ; start next A/D conversion
```

范例 2：使用中断的方式来检测转换结束

```
clr ADE                    ; disable ADC interrupt
mov a,03H
mov SADC1,a                ; select fSYS/8 as A/D clock
set ADCEN
mov a,c0H                  ; setup PCS0 to configure pin AN0
mov PCS0,a
mov a,20H
mov SADC0,a                ; enable and connect AN0 channel to A/D converter
:
Start_conversion:
clr START                  ; high pulse on START bit to initiate conversion
set START                  ; reset A/D
clr START                  ; start A/D
clr ADF                    ; clear ADC interrupt request flag
clr MF1F                   ; clear Multi-function interrupt 1 flag
set ADE                    ; enable ADC interrupt
set MF1E                   ; enable Multi-funciton interrupt 1
set EMI                    ; enable global interrupt
:
:
ADC_ISR:                   ; ADC interrupt service routine
mov acc_stack,a            ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a         ; save STATUS to user defined memory
:
mov a, SADOL               ; read low byte conversion result value
mov sadol_buffer,a         ; save result to user defined register
mov a, SADOH               ; read high byte conversion result value
mov sadoh_buffer,a         ; save result to user defined register
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a               ; restore STATUS from user defined memory
mov a,acc_stack            ; restore ACC from user defined memory
reti
```

恒定电流

该单片机内建三个通道的恒定电流，对应的 TD1~TD3 引脚连接电阻到地。当设定好电流值的恒定电流流经外部电阻到地时，会产生一个压降，当此电压降经由内部单位增益缓冲器连接到内部 A/D 转换器，通过 A/D 转换器量得此电压，由此可以计算出外部的电阻值。



恒定电流方框图

- **TDC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TDUGE	TDCCE	TDCCS1	TDCCS0	TDUGS1	TDUGS0	TDPS1	TDPS0
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 **TDUGE**：单位增益缓冲器电源控制
0：除能
1：使能
当选择除能时，单位增益缓冲器不耗电。

Bit 6 **TDCCE**：恒定电流电源控制
0：除能
1：使能
当选择除能时，恒定电流不耗电。当选择使能时，必须将 VBGEN 位设置成 1，恒定电流才会动作。

Bit 5~4 **TDCCS1~TDCCS0**：恒定电流输出电流选择
TDCCS[2:0] =
000：2.2μA
001：20μA
010：200μA
011：2mA
100：9.2μA
101：80μA
110：800μA
111：2mA
注意，TDCCS2 位位于 CTRL 寄存器。

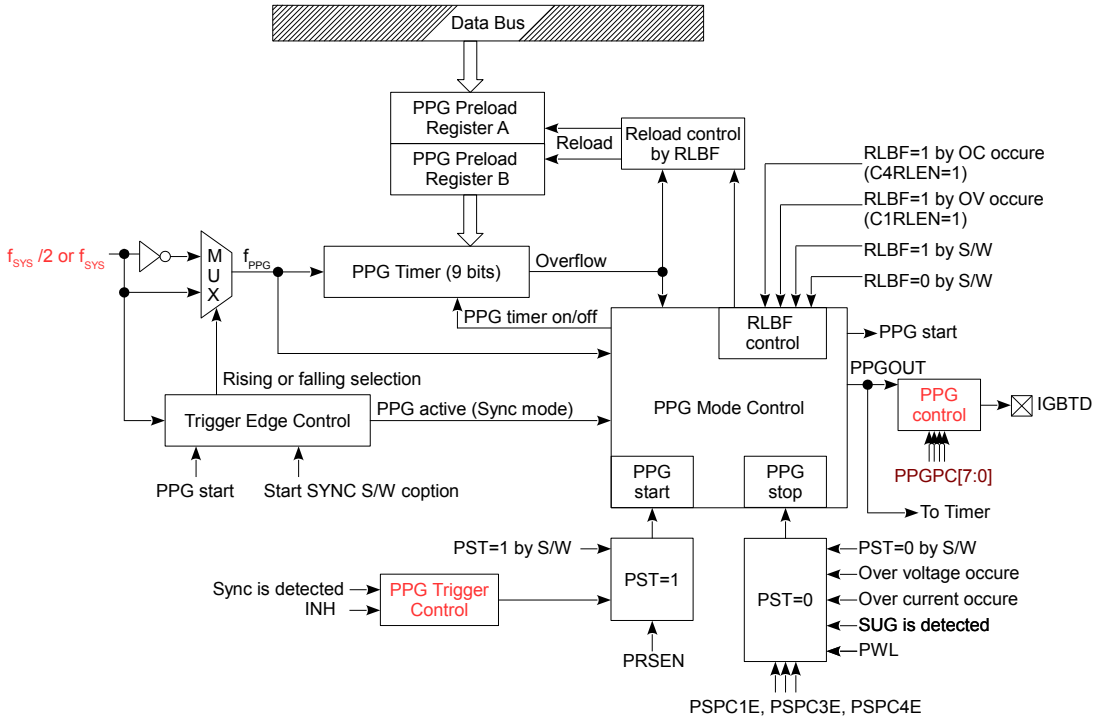
Bit 3~2 **TDUGS1~TDUGS0**：单位增益缓冲器输入选择
00：连接到 TD1
01：连接到 TD2
10：连接到 TD3
11：连接到 TD3

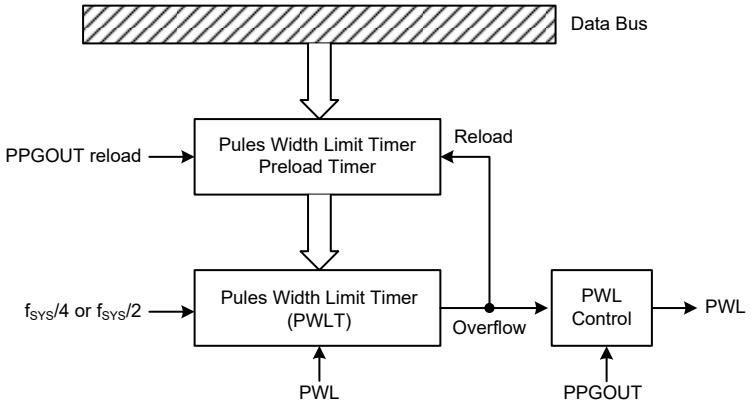
Bit 1 ~ 0 **TDPS1~TDPS0**：恒定电流输出选择
00：连接到 TD1
01：连接到 TD2
10：连接到 TD3
11：连接到 TD3

可编程脉冲发生器 – PPG

该单片机提供一组 9 位的 PPG 输出通道。PPG 的可编程周期为 512×T，对于一个输出脉宽，T 为 $1/f_{SYS}$ 或 $2/f_{SYS}$。使用脉宽限制计数器可以限制 PPG 的脉宽。PPG 功能只有在正常模式下可使用。

当 PPG 检测到有一个触发信号输入，就会输出一个脉冲。触发源可以通过软件配置来自 SYNC 被检测到或软件触发位。通过设置极性控制寄存器 PPGPC，IGBTD 可以输出一个低电平有效脉冲或高电平有效脉冲。当 $0.6V<V_{DD}<1.2V$ 时，IGBTD 输出引脚浮空，会产生复位，且 PPG 输出无效，此时需要外接一个上拉或下拉电阻 (取决于极性软件选项)。





PPG 方框图

PPG 模块由一个 PPG 计数器、一个 PPG 模式控制器和四个比较器组成。而 PPG 计数器由一个 9 位向上计数器和两个 9 位预载数据寄存器组成。可编程脉冲发生器 PPG 从预载寄存器的当前值开始计数，直到内部数据由“1FFH→000H”时停止计数。写入“000H”到 PPGTA 和 PPGTB 寄存器会产生一个 512×T 的脉宽输出。一旦计数器溢出，预载寄存器里的值会自动装载到计数器中，同时产生一个信号去停止 PPG 计数器。PPG 计数器溢出的同时软件触发位 PST 也会被清零。

PPG 计数器重新载入的时机：

1. PPG 计数器溢出；
2. PPG 关闭；
3. 任何导致 PPG 停止的动作。

PPG 寄存器

PPG 的功能和操作由一系列寄存器控制。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PPGC	PST	PRSEN	—	RLBF	PTSYN	—	TRGMOD	PCSD
PPGPC	PPGPC7	PPGPC6	PPGPC5	PPGPC4	PPGPC3	PPGPC2	PPGPC1	PPGPC0
PPGTA	PPGTA7	PPGTA6	PPGTA5	PPGTA4	PPGTA3	PPGTA2	PPGTA1	PPGTA0
PPGTB	PPGTB7	PPGTB6	PPGTB5	PPGTB4	PPGTB3	PPGTB2	PPGTB1	PPGTB0
PPGTEX	—	—	—	PPGTB8	—	—	—	PPGTA8
PWLT	D7	D6	D5	D4	D3	D2	D1	D0
PPGCTC	—	—	—	—	OVF	CTON	CTS1	CTS0
SYNCBUF	D7	D6	D5	D4	D3	D2	D1	D0

PPG 寄存器列表

- **PPGC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PST	PRSEN	—	RLBF	PTSYN	—	TRGMOD	PCSD
R/W	R/W	R/W	—	R/W	R/W	—	R/W	R/W
POR	0	0	—	0	0	—	0	0

Bit 7 **PST**：PPG 软件触发位
0：停止 PPG
1：启动 PPG

Bit 6 **PRSEN**：SYNC 被检测到时重新启动 PPG 模块输出使能 / 除能
0：除能
1：使能
该位为 0 时，PPG 模块输出只能通过设置软件控制位 PST 为 1 重新启动。该位为 1 时，PPG 输出可以通过设置软件控制位 PST 为 1 重新启动，也可以在 SYNC 被检测到时重新启动，这时 PST 会被设为 1。

Bit 5 未定义，读为“0”

Bit 4 **RLBF**：PPG 重新载入控制位
0：PPG 计数器重载值来自预载寄存器 A，PPGTA
1：PPG 计数器重载值来自预载寄存器 B，PPGTB

Bit 3 **PTSYN**：PPG 计数器是否与时钟同步
0：与时钟同步
1：与时钟异步

Bit 2 未定义，读为“0”

Bit 1 **TRGMOD**：选择触发 PPG 重新启动是比较器 0 输出的 1 个边沿或 2 个边沿
0：1 个边沿
1：2 个边沿
该位仅适用于上升沿或下降沿。(C0EG[1:0]=01 或 10)

Bit 1 **PCSD**：PPG 时钟是否除以 2
0：不除 2
1：除 2
最后的时钟 (除 2 或不除 2) 用于 PPG 计数器和脉宽限制计数器。

- **PPGPC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPGPC7	PPGPC6	PPGPC5	PPGPC4	PPGPC3	PPGPC2	PPGPC1	PPGPC0
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 **PPGPC7~PPGPC0**：IGBTD 输出有效电平控制位
01010101：IGBTD 输出低电平有效
10101010：IGBTD 输出高电平有效
其它值：IGBTD 输出浮空

- **PPGTA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPGTA7	PPGTA6	PPGTA5	PPGTA4	PPGTA3	PPGTA2	PPGTA1	PPGTA0
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 **PPGTA7~PPGTA0**：PPG 计数器预载寄存器 A bit 7 ~ bit 0

- **PPGTB 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPGTB7	PPGTB6	PPGTB5	PPGTB4	PPGTB3	PPGTB2	PPGTB1	PPGTB0
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 **PPGTB7~PPGTB0**：PPG 计数器预载寄存器 B bit 7 ~ bit 0

- **PPGTEX 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	PPGTB8	—	—	—	PPGTA8
R/W	—	—	—	R/W	—	—	—	R/W
POR	—	—	—	x	—	—	—	x

“x”：未知

Bit 7~5 未定义，读为“0”

Bit 4 **PPGTB8**：PPG 计数器预载寄存器 B bit 8

Bit 3~1 未定义，读为“0”

Bit 0 **PPGTA8**：PPG 计数器预载寄存器 A bit 8

- **PWLT 寄存器**

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

“x”：未知

Bit 7~0 **D7~D0**：PPG 脉宽限制计数器寄存器 bit 7 ~ bit 0

- **PPGCTC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	OVF	CTON	CTS1	CTS0
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 **OVF**：PPG 计数功能计数器溢出标志位
0：未溢出
1：溢出
此位用于指示 8-bit PPG 计数器是否发生溢出。该位由硬件置位后通过软件清零。

Bit 2 **CTON**：PPG 计数功能计数器控制位
0：停止计数
1：从 0 开始计数

Bit 1~0 **CTS1~CTS0**：PPG 计数值存入 SYNCBUF 的时间间隔选择位
00：f_{HIRC}/8192，约 0.5ms
01：f_{HIRC}/16384，约 1ms
10：f_{HIRC}/32768，约 2ms
11：f_{HIRC}/65536，约 4ms

- **SYNCBUF 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**：PPG 计数器值存储寄存器 bit 7 ~ bit 0
每隔 0.5ms、1ms、2ms 或 4ms (由 CTS1~CTS0 位选择) 硬件自动将 PPG 计数器的值存入此寄存器后，PPG 计数器重新从 0 开始计数。

通常情况下，如果 RLBF=0，PPG 计数器会重新载入 PPGTA 寄存器的值。如果 C1RLEN=1 或 C4RLEN=1，来自比较器输出的 C1O 或 C4O 上升沿会置高 RLBF 位，使得 PPG 计数器重新载入 PPGTB 寄存器的值，直到 RLBF 被软件清零。

PRSEN 位用于决定 PPG 的启动信号是否来自 C0O 的触发输入，如果是，那么一旦有一个 C0O 的下降沿、上升沿或双沿产生，PPG 计数器就会开始计数。

PSPC1E、PSPC3E 和 PSPC4E 位分别用于决定 PPG 的停止信号是否来自 C1O、C3O 和 C4O 的上升沿，如果是，那么一旦有一个 C1O、C3O 或 C4O 的上升沿产生，PPG 计数器就会停止计数。无论 PPG 是否处于有效周期，C1O、C3O 或 C4O 的上升沿都将清除 PRSEN 位，这样可以防止 PPG 再由 C0O 的下降沿、上升沿或双沿触发启动，PPG 的启动只能通过软件控制，直到 PRSEN 位再次由软件置位。

PST 是一个软件触发位，如果该位被置“1”，PPG 计数器会开始计数，当 PPG 计数器溢出时，该位会被清零同时 PPG 计数器停止计数。如果该位被置“0”，PPG 计数器会停止计数。

在 PPG 计数过程中，如果有一个 C0O 的下降沿、上升沿或双沿产生或 PST 被置位，PPG 计数器将不受影响，也就是说此时来自 C0O 或 PST 的再启动信号无效。PST 也可用作 PPG 计数器输出的状态标志位。

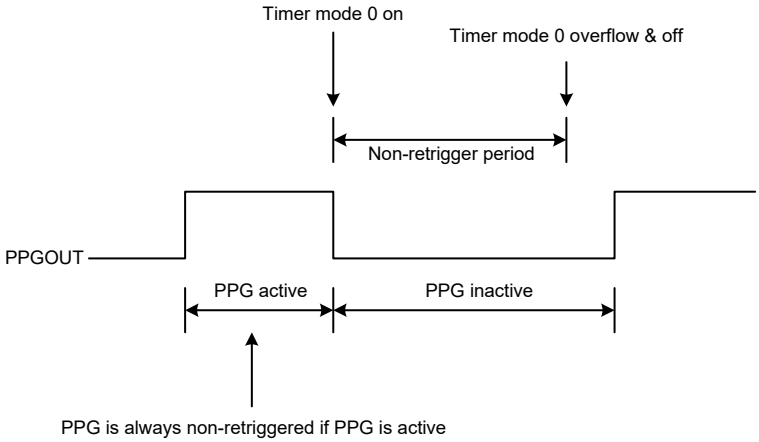
IGBTD 输出脉冲的有效电平由 PPGPC 寄存器决定。如果 PPGPC[7:0] 位被设为 01010101，PPG 输出低电平有效；如果 PPGPC[7:0] 位被设为 10101010，PPG 输出高电平有效。

另外一个功能，即 PPG 计数器是否与时钟同步，由 PTSYN 位决定。

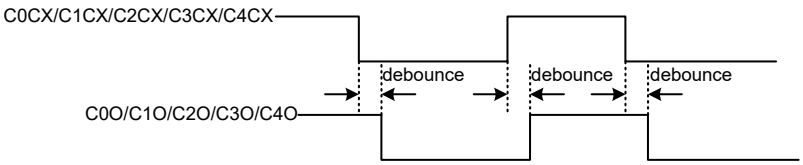
不可重复触发功能

PPG 模块具有不可重复触发功能，可以禁止 PPG 被再次触发。只要满足下列条件之一，PPG 将不会被重复触发。

1. PPG 有效；
2. 处于不可重复触发期间，定时 / 计数器在 PPG 停止时开始计数。(仅当定时 / 计数器工作于模式 0 有效，不可重复触发周期取决于定时 / 计数器)



- 对 C0CX/C1CX/C2CX/C3CX/C4CX 去抖



注：C0O/C1O/C2O/C3O/C4O 是比较器输出 C0CX/C1CX/C2CX/C3CX/C4CX 去抖后的信号。

脉宽限制功能

PPG 模块具有脉宽限制功能，可以停止 PPG 输出。当脉宽达到限值时，PPG 将停止输出。该功能是通过脉宽限制计数器实现的，该计数器在 PPG 触发时开始计数，溢出或 PPG 停止时停止计数。脉宽限值为 (256-PWLT)/(f_{SYS}/2)，其中 PWLT 是脉宽限制计数器寄存器的值。

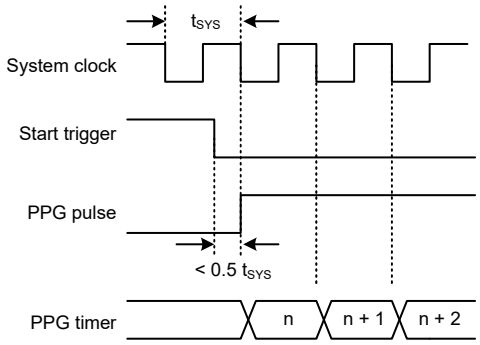
脉宽限制功能重新载入的时机：

1. 脉宽限制计数器溢出；

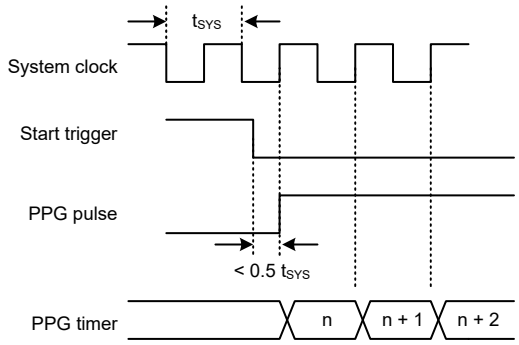
2. PPG 被触发。

控制 PPG 的启动延时 ≤0.5×(1/f_{SYS})，当选择时钟同步时，下一个系统周期的上升沿或下降沿均可触发 PPG 脉冲的输出。PPG 功能启动之后，PPG 输出使能，一旦第一个边沿 (系统频率的上升或下降沿) 来到，它便开始计数。第一个触发完成之后，接下来的触发时钟沿同第一次的触发时钟沿。例如一旦 PPG 的第一个启动是由一个下降沿触发，那么接下来的 PPG 都是由下降沿触发直到 PPG 停止输出，反之亦然。

例 1：在 PPG 启动之后第一个触发信号是下降沿，PPG 计数器此后都是下降沿触发直到 PPG 停止。



例 2：在 PPG 启动之后第一个触发信号是上升沿，PPG 计数器此后都是上升沿触发直到 PPG 停止。



任何停止 PPG 的动作，如 PPG 计数器溢出、C1O/C3O/C4O 上升沿 (PSPC1E=1 / PSPC3E=1 / PSPC4E=1)、软件指令停止 (PST=1→0) 或达到脉冲限制等，均可导致以下情况的发生：

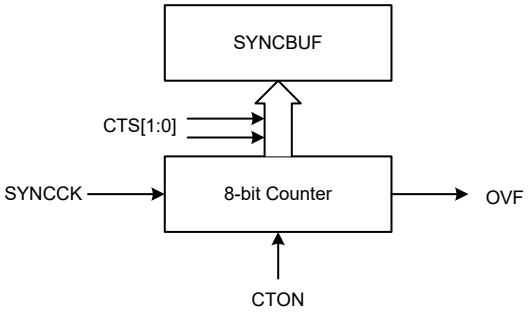
- PPG 计数器数据重新载入
- PST 会被清 0
- PPG 无效

启动 PPG 功能的操作：

- 通过软件选项使能 PPG 功能
- 设置 IGBTD 输出的有效电平
- 确定 PPG 计数时钟是否与系统时钟 f_{SYS} 同步
- PPG 输入模式选择
- 设置 PPG 输出脉宽，写入数据到 PPGTA、PPGTB 和 PPGTEX 寄存器
- 确定是否使用 C1O/C4O 的上升沿来使能 PPGTB 的重新载入功能
- 确定是否使用不可重复触发周期功能 (使用定时 / 计数器的模式 0)
- 通过 PWLT 寄存器设置脉宽限制计数器来限制脉宽
- 当 PPG 由 SYNC 被检测到或软件触发位 PST 被设为 1 触发时，PPG 将从预载寄存器的当前值开始计数。当 PPG 软件触发位 PST 被设为 0、C1O/C3O/C4O 的上升沿发生、PPG 计数器溢出或达到脉冲限制时，PPG 将停止计数。

自动记录同步信号次数功能

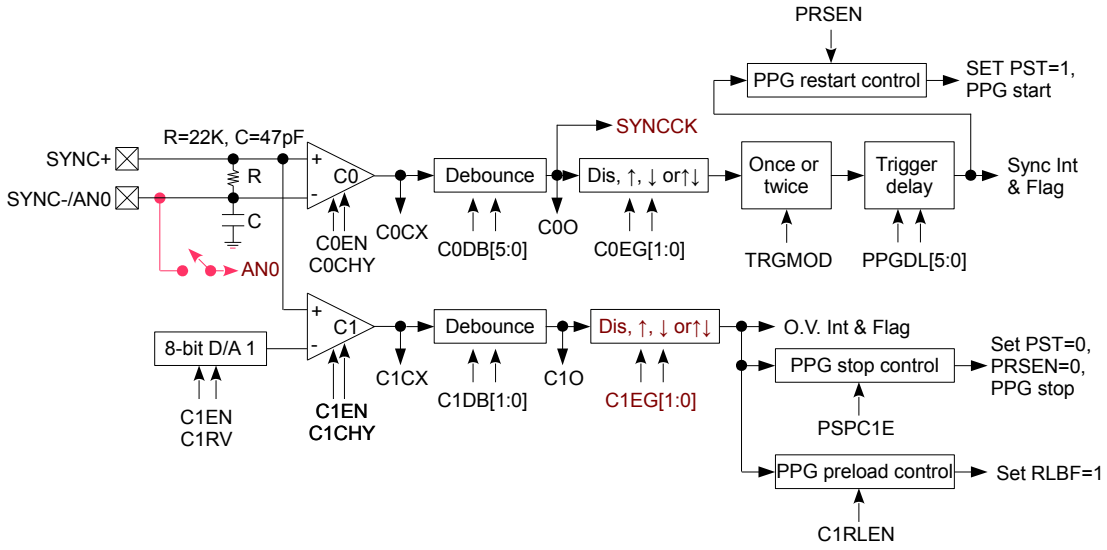
PPG 模块内含一个 8 位计数功能计数器，每来一个 SYNCCK 信号，即 C0O 上升沿，此计数器加一。通过 PPGCTC 寄存器中的 CTS1~CTS0 位选择每隔 0.5ms、1ms、2ms 或 4ms 将此计数器的值存入 SYNCBUF 寄存器。OVF 标志位可用于检查此计数器是否溢出。当到达所选的时间时，会产生一个中断。

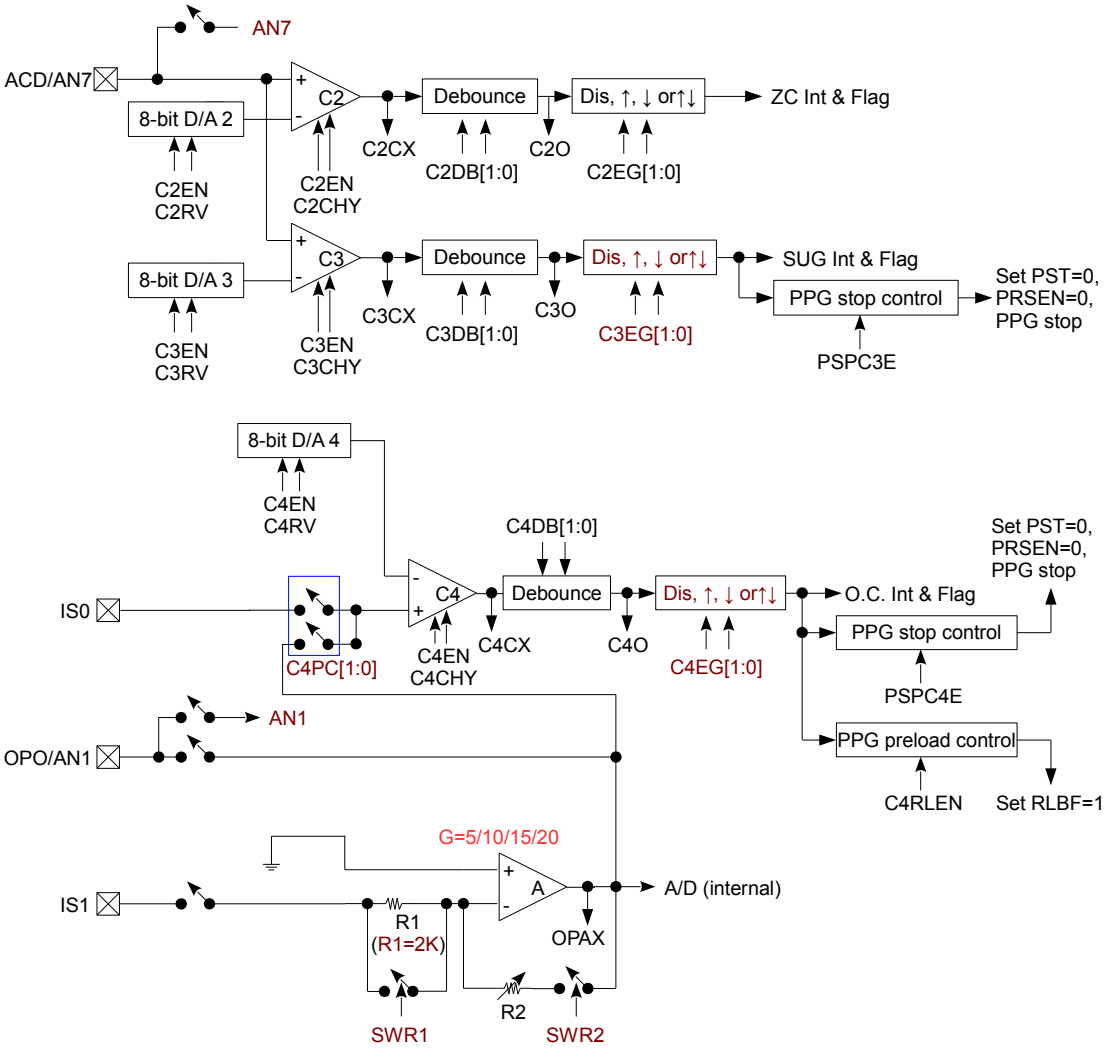


自动记录同步信号次数示意图

比较器 & 运算放大器

该单片机内建五个比较器和一个运算放大器。其中四个比较器带有 D/A 转换器模块，分别用于过压检测、过流检测、浪涌电压检测和过零检测应用。另一个比较器用于 SYNC 峰值检测。运算放大器用于放大外部电流。




AN7
ACD/AN7
8-bit D/A 2
C2
C2EN
C2RV
C2EN
C2CHY
C2CX
Debounce
C2DB[1:0]
C2O
Dis, ↑, ↓ or↑↓
C2EG[1:0]
ZC Int & Flag
8-bit D/A 3
C3
C3EN
C3RV
C3EN
C3CHY
C3CX
Debounce
C3DB[1:0]
C3O
Dis, ↑, ↓ or↑↓
C3EG[1:0]
SUG Int & Flag
PPG stop control
PSPC3E
Set PST=0,
PRSEN=0,
PPG stop
8-bit D/A 4
C4EN
C4RV
C4DB[1:0]
Set PST=0,
PRSEN=0,
PPG stop
IS0
C4
C4PC[1:0]
C4EN
C4CHY
C4CX
Debounce
C4O
Dis, ↑, ↓ or↑↓
C4EG[1:0]
O.C. Int & Flag
PPG stop control
PSPC4E
AN1
OPO/AN1
PPG preload control
C4RLEN
Set RLBF=1
G=5/10/15/20
IS1
A
A/D (internal)
R1
(R1=2K)
OPAX
R2
SWR1
SWR2

比较器 & OPA 方框图

比较器寄存器

比较器的功能和操作由一系列寄存器控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
CMP0C0	C0EG1	C0EG0	C0EN	C0CHY	C0O	C0RSW	C0CSW	—
CMP0C2	—	—	C0DB5	C0DB4	C0DB3	C0DB2	C0DB1	C0DB0
CMP0C3	—	—	PPGDL5	PPGDL4	PPGDL3	PPGDL2	PPGDL1	PPGDL0
CMP1C0	PSPC1E	C1RLEN	C1EN	C1CHY	C1O	C1RV	C1DB1	C1DB0
CMP1C2	C1EG1	C1EG0	—	—	—	—	—	—
CMP2C0	C2EG1	C2EG0	C2EN	C2CHY	C2O	C2RV	C2DB1	C2DB0
CMP3C0	PSPC3E	—	C3EN	C3CHY	C3O	C3RV	C3DB1	C3DB0
CMP3C2	C3EG1	C3EG0	—	—	—	—	—	—
CMP4C0	PSPC4E	C4RLEN	C4EN	C4CHY	C4O	C4RV	C4DB1	C4DB0
CMP4C2	C4EG1	C4EG0	—	—	—	—	C4PC1	C4PC0
CMPnC1 (n=0~4)	CnCX	CnOFM	CnRS	CnOF4	CnOF3	CnOF2	CnOF1	CnOF0
CMPnDA (n=1~4)	D7	D6	D5	D4	D3	D2	D1	D0

比较器寄存器列表

- **CMP0C0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	C0EG1	C0EG0	C0EN	C0CHY	C0O	C0RSW	C0CSW	—
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	—
POR	0	0	0	0	0	0	0	—

Bit 7~6 **C0EG1~C0EG0**：比较器 0 输出的 SYNC 中断边沿控制位
00：除能
01：上升沿触发
10：下降沿触发
11：双沿触发

Bit 5 **C0EN**：比较器 0 使能控制位
0：除能
1：使能，比较器 0 电源开启
如果该位为 0，比较器 0 将除能并且不会产生功耗。这将导致比较器 0 被关闭。

Bit 4 **C0CHY**：比较器 0 迟滞使能控制位
0：除能
1：使能

Bit 3 **C0O**：比较器 0 去抖数字输出位

Bit 2 **C0RSW**：比较器 0 内部电阻模拟开关控制位
0：Off
1：On

Bit 1 **C0CSW**：比较器 0 内部电容模拟开关控制位
0：Off
1：On

Bit 0 未定义，读为“0”

- **CMPnC1 寄存器 (n=0)**

Bit	7	6	5	4	3	2	1	0
Name	CnCX	CnOFM	CnRS	CnOF4	CnOF3	CnOF2	CnOF1	CnOF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	0	0	1	0	0	0	0

“x”：未知

Bit 7 **CnCX**：比较器数字输出
0：正相输入电压 < 反相输入电压
1：正相输入电压 > 反相输入电压

Bit 6 **CnOFM**：比较器模式 / 输入失调电压校准模式选择位
0：比较器模式
1：输入失调电压校准模式

Bit 5 **CnRS**：比较器输入失调电压校准参考输入选择位
0：选择内部 0V 输入
1：选择比较器正相输入端

Bit 4~0 **CnOF4~CnOF0**：比较器输入失调电压校准设置

- **CMPnC1 寄存器 (n=1~4)**

Bit	7	6	5	4	3	2	1	0
Name	CnCX	CnOFM	CnRS	CnOF4	CnOF3	CnOF2	CnOF1	CnOF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	0	0	1	0	0	0	0

“x”：未知

Bit 7 **CnCX**：比较器数字输出
0：正相输入电压 < 反相输入电压
1：正相输入电压 > 反相输入电压

Bit 6 **CnOFM**：比较器模式 / 输入失调电压校准模式选择位
0：比较器模式
1：输入失调电压校准模式

Bit 5 **CnRS**：比较器输入失调电压校准参考输入选择位
0：选择比较器反相输入端
1：选择比较器正相输入端

Bit 4~0 **CnOF4~CnOF0**：比较器输入失调电压校准设置

- **CMP0C2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	C0DB5	C0DB4	C0DB3	C0DB2	C0DB1	C0DB0
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 **C0DB5~C0DB0**：选择比较器 0 输出去抖时间
000000：无去抖
000001：去抖时间 = (0~1) × $1/f_{SYS}$
000010：去抖时间 = (1~2) × $1/f_{SYS}$
000011：去抖时间 = (2~3) × $1/f_{SYS}$
:
:
101111：去抖时间 = (46~47) × $1/f_{SYS}$
11xxxx：去抖时间 = (47~48) × $1/f_{SYS}$

- **CMP0C3 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PPGDL5	PPGDL4	PPGDL3	PPGDL2	PPGDL1	PPGDL0
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　**PPGDL5~PPGDL0**：选择 PPG 触发延迟
000000：无延迟
000001：延迟时间 = 1 × 1/f_{SYS}
000010：延迟时间 = 2 × 1/f_{SYS}
000011：延迟时间 = 3 × 1/f_{SYS}
:
:
101111：延迟时间 = 47 × 1/f_{SYS}
11xxxx：延迟时间 = 48 × 1/f_{SYS}
注：1. 触发延迟意味着被发送的 PPG 触发信号有一定的延迟时间。
2. 在触发延迟期间的 PPG 触发将被忽略。

- **CMP1C0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PSPC1E	C1RLEN	C1EN	C1CHY	C1O	C1RV	C1DB1	C1DB0
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7　**PSPC1E**：过压发生时 PPG 模块停止输出使能 / 除能位
0：除能
1：使能
若使能，当过压情况发生时，PPG 模块输出将无效，PST 位被置 0，PRSEN 位被置 0。

Bit 6　**C1RLEN**：过压发生时 RLBF 被置位使能 / 除能位
0：除能
1：使能
若使能，当过压情况发生时，PPG 计数器将从预载寄存器 B 重新载入计数值，RLBF 被置 1。

Bit 5　**C1EN**：比较器 1 使能控制位
0：除能
1：使能，比较器 1 及其 DAC 的电源开启
如果该位为 0，比较器 1 及其 DAC 将除能并且不会产生功耗。这将导致比较器 1 及其 DAC 都被关闭。

Bit 4　**C1CHY**：比较器 1 迟滞使能控制位
0：除能
1：使能

Bit 3　**C1O**：比较器 1 去抖数字输出位

Bit 2　**C1RV**：比较器 1 DAC 的参考电压选择
0：VDD 引脚
1：VREF 引脚

Bit 1~0　**C1DB1~C1DB0**：比较器 1 去抖时间选择
00：无去抖
01：去抖时间 = (3~4) × 1/f_{SYS}
10：去抖时间 = (7~8) × 1/f_{SYS}
11：去抖时间 = (15~16) × 1/f_{SYS}

- **CMP1C2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	C1EG1	C1EG0	—	—	—	—	—	—
R/W	R/W	R/W	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

Bit 7~6 **C1EG1~C1EG0**：比较器 1 输出的过压保护中断边沿控制位
00：除能
01：上升沿触发
10：下降沿触发
11：双沿触发

Bit 5~0 未定义，读为“0”

- **CMP2C0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	C2EG1	C2EG0	C2EN	C2CHY	C2O	C2RV	C2DB1	C2DB0
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **C2EG1~C2EG0**：比较器 2 输出的过零中断边沿控制位
00：除能
01：上升沿触发
10：下降沿触发
11：双沿触发

Bit 5 **C2EN**：比较器 2 使能控制位
0：除能
1：使能，比较器 2 及其 DAC 的电源开启
如果该位为 0，比较器 2 及其 DAC 将除能并且不会产生功耗。这将导致比较器 2 及其 DAC 都被关闭。

Bit 4 **C2CHY**：比较器 2 迟滞使能控制位
0：除能
1：使能

Bit 3 **C2O**：比较器 2 去抖数字输出位

Bit 2 **C2RV**：选择比较器 2 DAC 的参考电压
0：VDD 引脚
1：VREF 引脚

Bit 1~0 **C2DB1~C2DB0**：比较器 2 去抖时间选择
00：无去抖
01：去抖时间 = (3~4) × $1/f_{SYS}$
10：去抖时间 = (7~8) × $1/f_{SYS}$
11：去抖时间 = (15~16) × $1/f_{SYS}$

- **CMP3C0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PSPC3E	—	C3EN	C3CHY	C3O	C3RV	C3DB1	C3DB0
R/W	R/W	—	R/W	R/W	R	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	0

Bit 7 **PSPC3E**：浪涌电压发生时 PPG 模块停止输出使能 / 除能位
0：除能
1：使能
若使能，当浪涌电压情况发生时，PPG 模块输出将无效，PST 位被置 0，PRSEN 位被置 0。

Bit 6　　未定义，读为“0”

Bit 5　　**C3EN**：比较器 3 使能控制位
0：除能
1：使能，比较器 3 及其 DAC 的电源开启
如果该位为 0，比较器 3 及其 DAC 将除能并且不会产生功耗。这将导致比较器 3 及其 DAC 都被关闭。

Bit 4　　**C3CHY**：比较器 3 迟滞使能控制位
0：除能
1：使能

Bit 3　　**C3O**：比较器 3 去抖数字输出位

Bit 2　　**C3RV**：选择比较器 3 DAC 的参考电压
0：VDD 引脚
1：VREF 引脚

Bit 1~0　　**C3DB1~C3DB0**：比较器 3 去抖时间选择
00：无去抖
01：去抖时间 = (3~4) × 1/f_{SYS}
10：去抖时间 = (7~8) × 1/f_{SYS}
11：去抖时间 = (15~16) × 1/f_{SYS}

- **CMP3C2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	C3EG1	C3EG0	—	—	—	—	—	—
R/W	R/W	R/W	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

Bit 7~6　　**C3EG1~C3EG0**：比较器 3 输出的浪涌电压保护中断边沿控制位
00：除能
01：上升沿触发
10：下降沿触发
11：双沿触发

Bit 5~0　　未定义，读为“0”

- **CMP4C0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PSPC4E	C4RLEN	C4EN	C4CHY	C4O	C4RV	C4DB1	C4DB0
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7　　**PSPC4E**：过流发生时 PPG 模块停止输出使能 / 除能位
0：除能
1：使能
若使能，当过流情况发生时，PPG 模块输出将无效，PST 位被置 0，PRSEN 位被置 0。

Bit 6　　**C4RLEN**：过流发生时 RLBF 被置位使能 / 除能位
0：除能
1：使能
若使能，当过流情况发生时，PPG 计数器将从预载寄存器 B 重新载入计数值，RLBF 被置 1。

Bit 5 **C4EN**：比较器 4 使能控制位
0：除能
1：使能，比较器 1 及其 DAC 的电源开启
如果该位为 0，比较器 4 及其 DAC 将除能并且不会产生功耗。这将导致比较器 4 及其 DAC 都被关闭。

Bit 4 **C4CHY**：比较器 4 迟滞使能控制位
0：除能
1：使能

Bit 3 **C4O**：比较器 4 去抖数字输出位

Bit 2 **C4RV**：选择比较器 4 DAC 的参考电压
0：VDD 引脚
1：VREF 引脚

Bit 1~0 **C4DB1~C4DB0**：比较器 4 去抖时间选择
00：无去抖
01：去抖时间 = (3~4) × $1/f_{SYS}$
10：去抖时间 = (7~8) × $1/f_{SYS}$
11：去抖时间 = (15~16) × $1/f_{SYS}$

- **CMP4C2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	C4EG1	C4EG0	—	—	—	—	C4PC1	C4PC0
R/W	R/W	R/W	—	—	—	—	R/W	R/W
POR	0	0	—	—	—	—	0	0

Bit 7~6 **C4EG1~C4EG0**：比较器 4 输出的过流保护中断边沿控制位
00：除能
01：上升沿触发
10：下降沿触发
11：双沿触发

Bit 5~2 未定义，读为“0”

Bit 1~0 **C4PC1~C4PC0**：比较器 4 同相输入端选择位
0x：IS0 引脚
1x：OPA 输出

- **CMPnDA 寄存器 (n=1~4)**

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**：CMPn DAC 数据寄存器 bit 7 ~ bit 0
8-bit DAC 数据位
CMPn DAC 输出电压 = (DAC 参考电压) × (D[7:0])/256

运算放大器寄存器

运算放大器的功能和操作由一系列寄存器控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
OPAC0	OPAM1	OPAM0	OPAG2	OPAG1	OPAG0	SWR2	SWR1	OPAX
OPAC1	AOFM	ARS	AOF5	AOF4	AOF3	AOF2	AOF1	AOF0

运算放大器寄存器列表

- **OPAC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OPAM1	OPAM0	OPAG2	OPAG1	OPAG0	SWR2	SWR1	OPAX
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
POR	0	0	0	0	0	0	0	x

“x”：未知

Bit 7~6 **OPAM1~OPAM0**：OPA 使能 / 除能控制位
00：OPA 除能
01：OPA 使能
10：OPA 使能
11：OPA 除能

Bit 5~3 **OPAG2~OPAG0**：OPA 增益选择位
000：×5
001：×5
010：×10
011：×15
100：×20
101：×20
110：×20
111：×20

Bit 2 **SWR2**：OPA R2 开关控制位
0：Off
1：On

Bit 1 **SWR1**：OPA R1 开关控制位
0：Off
1：On

Bit 0 **OPAX**：输入失调电压校准模式下运算放大器数字输出
0：正相输入电压 < 反相输入电压
1：正相输入电压 > 反相输入电压

- **OPAC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	AOFM	ARS	AOF5	AOF4	AOF3	AOF2	AOF1	AOF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **AOFM**：运算放大器模式 / 输入失调电压校准模式选择位
0：运算放大器模式
1：输入失调电压校准模式

Bit 6 **ARS**：运算放大器输入失调电压校准参考输入选择位
0：选择运算放大器反相输入端
1：选择运算放大器正相输入端

Bit 5~0 **AOF5~AOF0**：运算放大器输入失调电压校准设置

运算放大器输入失调校准

校准步骤如下所示：

1. 设置 AOFM=1 且 OPAM[1:0]=01 or 10，OPA 进入失调电压校准模式
2. 设置 AOF[5:0]=000000，读取 OPAX 值
3. 将 AOF[5:0] 值加一再读取 OPAX 值
   若 OPAX 值无变化，重复步骤 3，直到 OPAX 值发生改变。记录此时 AOF[5:0] 值为 V_{OS1}，接着执行步骤 4。
4. 设置 AOF[5:0]=111111，读取 OPAX 值
5. 将 AOF[5:0] 值减一再读取 OPAX 值
   若 OPAX 值无变化，重复步骤 5，直到 OPAX 值发生改变，记录此时 AOF[5:0] 值为 V_{OS2}，接着执行步骤 6。
6. 重新设置 AOF[5:0]=V_{OS}=(V_{OS1}+V_{OS2})/2，校准完成。

比较器输入失调校准

校准步骤如下所示：

1. 设置 CnEN=1 和 CnOFM=1，比较器进入失调电压校准模式
2. 设置 CnOF[4:0]=00000，读取 CnCX 值
3. 将 CnOF[4:0] 值加一再读取 CnCX 值
   若 CnCX 值无变化，重复步骤 3，直到 CnCX 值发生改变，记录此时 CnOF[4:0] 值为 V_{OS1}，接着执行步骤 4。
4. 设置 CnOF[4:0]=11111，读取 CnCX 值
5. 将 CnOF[4:0] 值减一再读取 CnCX 值，
   若 CnCX 值无变化，重复步骤 5，直到 CnCX 值发生改变，记录此时 CnOF[4:0] 值为 V_{OS2}，接着执行步骤 6。
6. 重新设置 CnOF[4:0]=V_{OS}=(V_{OS1}+V_{OS2})/2，校准完成。

过压保护功能 – OVP

为了防止 SYNC+ 输入电压大于某一电压，可将 SYNC+ 输入电压与 8 位的参考电压进行比较。一旦 SYNC+ 输入电压大于参考电压，将强制 PPG 无效，并产生中断告知 MCU。此外 PPG 的预载寄存器会改为预载寄存器 B。这些都可以通过软件使能或除能。

过流保护功能 – OCP

为了防止 IS0 输入电压或 OPA 输出电压大于某一电压，可将比较器的正相输入电压与 8 位的参考电压进行比较。一旦 IS0 输入电压或 OPA 输出电压大于参考电压，将强制 PPG 无效，并产生中断告知 MCU。此外 PPG 的预载寄存器会改为预载寄存器 B。这些都可以通过软件使能或除能。

浪涌电压保护功能

为了防止 ACD 输入电压大于某一电压，可将 ACD 输入电压与 8 位的参考电压进行比较。一旦 ACD 输入电压大于参考电压，将强制 PPG 无效，并产生中断告知 MCU。这可以通过软件使能或除能。

过零 (ZC) 检测功能

为了检测 ACD 输入电压是否过零，可将 ACD 输入电压与 8 位的参考电压进行比较。一旦 ACD 输入电压过零，将会产生过零中断。

SYNC 检测功能

为了检测 SYNC 峰值电压，将 SYNC+ 和 SYNC- 与一个 RC 延迟进行比较，R 或 C 可以通过软件除能。一旦 SYNC 被检测到，将会触发 PPG 启动，这可以通过软件使能或除能。

运算放大器功能 – OPA

OPA 用来放大外部电流，OPA 内建 5、10、15 和 20 倍的放大倍率。除了输出到 OPA 输出引脚 OPO 之外，同时也可经由内部，通过软件控制位选择直接连到内部 A/D 输入端还是比较器 4 的同相输入端。

外围时钟输出

外围时钟输出功能使单片机能够为外部硬件提供和单片机时钟同步的时钟信号。

外围时钟操作

外围时钟输出引脚 PCK 与 I/O 口共用，可以通过相关引脚共用控制位选择该引脚功能。外围时钟功能由 PCKC 寄存器控制。设置完 PCKEN 和 PCKPOL 位后，将 PCKD 位置 1 会使能 PCK 输出功能。将 PCKD 位清零会除能 PCK 输出功能，且根据 PCKPOL 位的设置将 PCK 引脚输出强制拉低或拉高。外围时钟输出的时钟源来自 f_{HIRC} 分频，分频比由 PCKPSC3~PCKPSC0 位选择。

- **PCKC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PCKD	PCKPOL	PCKEN	PCKPSC3	PCKPSC2	PCKPSC1	PCKPSC0
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　**PCKD**：PCK 输出控制位
0：无效
1：有效
此位用于控制 PCK 的输出是否有效。

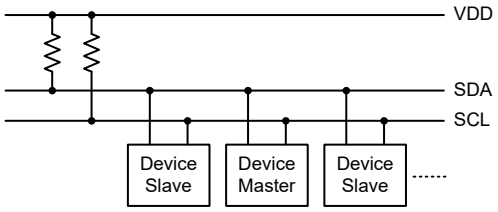
Bit 5　**PCKPOL**：PCK 输出极性控制位
0：同相
1：反相
当 PCKD=0 时，若此位为 0，则 PCK 输出低；若此位为 1，则 PCK 输出高。

Bit 4　**PCKEN**：PCK 总控制位
0：除能
1：使能

Bit 3~0　**PCKPSC3~PCKPSC0**：外围时钟分频比选择位
0000：f_{HIRC}/8
0001：f_{HIRC}/16
0010：f_{HIRC}/24
0011：f_{HIRC}/32
0100：f_{HIRC}/40
0101：f_{HIRC}/48
0110：f_{HIRC}/2048
0111：f_{HIRC}/4096
1xxx：f_{HIRC}/8192

I²C 模块串行接口

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



I²C 主从总线连接图

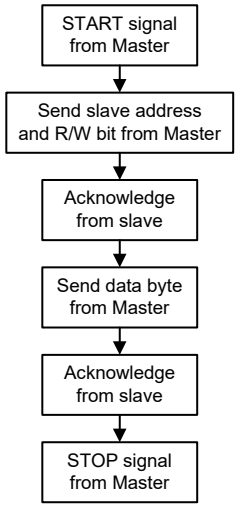
I²C 接口操作

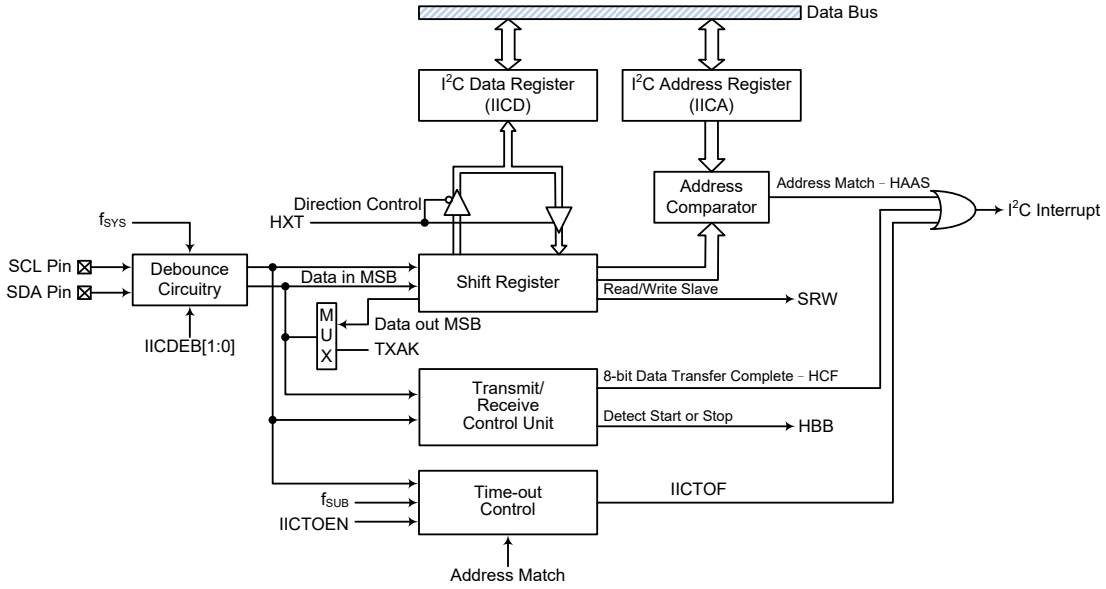
I²C 串行接口是一个双线的接口，有一条串行数据线 SDA 和一条串行时钟线 SCL。由于可能有多个设备在同一条总线上相互连接，所以这些设备的输出都是开漏型输出。因此应在这些输出口上都应加上拉电阻。应注意的是：I²C 总线上的每个设备都没有选择线，但分别与唯一的地址一一对应，用于 I²C 通信。

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

在 I²C 通信期间，建议用户不要通过应用程序使处理器进入 HALT 状态。

如果引脚被配置为 I²C 接口中 SDA 或 SCL 功能，该引脚必须配置为开集输入 / 输出口，且通过相应的上拉电阻控制寄存器使能该引脚的上拉电阻功能。





I²C 方框图

IICDEB1 和 IICDEB0 位决定 I²C 接口的去抖时间。这个功能可以使用内部时钟在外部时钟上增加一个去抖间隔，减小时钟线上毛刺发生的可能性，以避免单片机发生误动作。如果选择了这个功能，去抖时间可以选择 2 个或 4 个系统时钟。为了达到需要的 I²C 数据传输速度，系统时钟 f_{SYS} 和 I²C 去抖时间之间存在一定的关系。I²C 标准模式或者快速模式下，用户需注意所选的系统时钟频率与标准匹配去抖时间的设置，其具体关系如下表所示。

I²C 去抖时间选择	I²C 标准模式 (100kHz)	I²C 快速模式 (400kHz)
无去抖时间	f_{SYS} > 2 MHz	f_{SYS} > 5 MHz
2 个系统时钟去抖时间	f_{SYS} > 4 MHz	f_{SYS} > 10 MHz
4 个系统时钟去抖时间	f_{SYS} > 8 MHz	f_{SYS} > 20 MHz

I²C 最小 f_{SYS} 频率

I²C 寄存器

I²C 总线的四个控制寄存器是 IICC0、IICC1、IICA 和 IICTOC 及一个数据寄存器 IICD。

寄存器名称	位							
	7	6	5	4	3	2	1	0
IICC0	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
IICC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
IICD	D7	D6	D5	D4	D3	D2	D1	D0
IICA	A6	A5	A4	A3	A2	A1	A0	—
IICTOC	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0

I²C 寄存器列表

- **IICD 寄存器**

IICD 用于存储发送和接收的数据。在单片机尚未将数据写入到 I²C 总线中时，要传输的数据应存在 IICD 中。I²C 总线接收到数据之后，单片机就可以从 IICD 数据寄存器中读取。所有通过 I²C 传输或接收的数据都必须通过 IICD 实现。

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

“x”：未知

Bit 7~0 **D7~D0**：I²C 数据缓存位 bit 7~bit 0

- **IICA 寄存器**

IICA 寄存器用于存放 7 位从机地址，寄存器 IICA 中的第 7~1 位是单片机的从机地址，位 0 未定义。如果接至 I²C 总线的主机发送出的地址和寄存器 IICA 中存储的地址相符，那么就选中了这个从机。

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

Bit 0 未定义，读为“0”

I²C 控制寄存器

单片机中有两个控制 I²C 接口功能的寄存器，IICC0 和 IICC1。寄存器 IICC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 IICC1 包括多个用于指示 I²C 传输状态的相关标志位。另外还有一个 IICTOC 寄存器用于控制 I²C 超时功能，此寄存器在 I²C 超时一节介绍。

- **IICC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4 未定义，读为“0”

Bit 3~2 **IICDEB1~IICDEB0**：I²C 去抖时间选择位
00：未定义
01：2 个系统时钟去抖时间
1x：4 个系统时钟去抖时间
注：如果 f_{SYS} 讯号来自于 f_H 且已准备或 IAMWU=0，去抖电路为有效，否则，SCL 和 SDA 将绕过去抖电路。

Bit 1 **IICEN**：I²C 控制位
0：除能
1：使能
此位为 I²C 接口的开 / 关控制位。此位为“0”时，I²C 接口除能，SDA 和 SCL 脚将失去 I²C 功能，I²C 工作电流减小到最小值。此位为“1”时，I²C 接口使能。当 IICEN 位由低到高转变时，I²C 控制寄存器中的设置，如 HXT 和 TXAK，将

不会发生变化，其首先应在应用程序中初始化，此时相关 I²C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0　未定义，读为“0”

- **IICC1 寄存器**

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

Bit 7　**HCF**：I²C 总线数据传输结束标志位
0：数据正在被传输
1：8 位数据传输完成
数据正在传输时该位为低。当 8 位数据传输完成时，此位为高并产生一个中断。

Bit 6　**HAAS**：I²C 地址匹配标志位
0：地址不匹配
1：地址匹配
此标志位用于决定从机地址是否与主机发送地址相同。若地址匹配此位为高，否则此位为低。

Bit 5　**HBB**：I²C 总线忙标志位
0：I²C 总线闲
1：I²C 总线忙
当检测到 START 信号时 I²C 忙，此位变为高电平。当检测到 STOP 信号时 I²C 总线停止，该位变为低电平。

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

Bit 3　**TXAK**：I²C 总线发送应答标志位
0：从机发送应答标志
1：从机没有发送应答标志
单片机接收 8 位数据之后会将该位在第九个时钟时传到总线上。如果从机想要接收更多的数据，则应在接收数据之前将此位设置为“0”。

Bit 2　**SRW**：I²C 从机读 / 写位
0：从机应处于接收模式
1：从机应处于发送模式
SRW 位是从机读写位。决定主机是否希望传输或接收来自 I²C 总线的数据。当传输地址和从机的地址相同时，HAAS 位会被设置为高，从机将检测 SRW 位来决定进入发送模式还是接收模式。如果 SRW 位为高时，主机会请求从总线上读数据，此时从机处于传输模式。当 SRW 位为“0”时，主机往总线上写数据，从机处于接收模式以读取该数据。

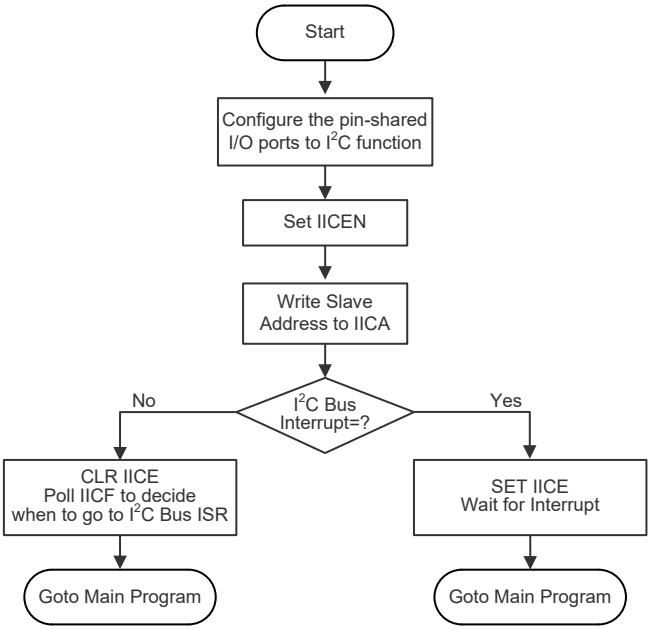
Bit 1　**IAMWU**：I²C 地址匹配唤醒控制位
0：除能
1：使能
此位应设置为“1”使能 I²C 地址匹配以使系统从休眠模式或空闲模式中唤醒。若进入休眠模式或空闲模式前 IAMWU 已经置高以使能 I²C 地址匹配唤醒功能，在系统唤醒后须软件清除此位以确保单片机正确地运行。

Bit 0　**RXAK**：I²C 总线接收应答标志位
0：从机接收到应答标志
1：从机没有接收到应答标志
RXAK 位是接收应答标志位。如果 RXAK 位被重设为“0”即 8 位数据传输之后，从机在第九个时钟有接受到一个应答信号。如果从机处于发送状态，从机作为发送方会检查 RXAK 位来判断主机接收方是否愿意继续接收下一个字节。因此发送方会一直发送数据，直到 RXAK 为“1”时才停止发送数据。这时，发送方将释放 SDA 线，主机方可发出停止信号从而释放 I²C 总线。

I²C 总线通信

I²C 总线上的通信需要四步完成，一个起始信号，一个从机地址发送，一个数据传输，还有一个停止信号。当起始信号被写入 I²C 总线时，总线上的所有从机都会接收到这个起始信号并且被通知总线上会即将有数据到达。数据的前 7 位是从机地址，高位在前，低位在后。如果发出的地址和从机地址匹配，IICC1 寄存器的 HAAS 位会被置位，同时产生 I²C 中断。进入中断服务程序后，系统要检测 HAAS 位和 IICTOF 位，以判断 I²C 总线中断是来自从机地址匹配，还是来自 8 位数据传递完毕，或是来自 I²C 超时。在数据传递中，注意的是，在 7 位从机地址被发送后，接下来的一位，即第 8 位，是读 / 写控制位，该位的值会反映到 SRW 位中。从机通过检测 SRW 位以确定自己是要进入发送模式还是接收模式。在 I²C 总线开始传送数据前，需要先初始化 I²C 总线，初始化 I²C 总线步骤如下：

- 步骤 1
  设置引脚共用 I/O 口为 I²C 引脚功能。设置 IICC0 寄存器中 IICEN 位为“1”，以使能 I²C 总线。
- 步骤 2
  向 I²C 总线地址寄存器 IICA 写入从机地址。
- 步骤 3
  设置 IICE 位，以使能 I²C 中断。



I²C 总线初始化流程图

I²C 总线起始信号

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

从机地址

总线上的所有从机都会侦测由主机发出的起始信号。发送起始信号后，紧接着主机会发送从机地址以选择要进行数据传输的从机。所有在 I²C 总线上的从机接收到 7 位地址数据后，都会将其与各自内部的地址进行比较。如果从机从主机上接收到的地址与自身内部的地址相匹配，则会产生一个 I²C 总线中断信号。地址位接下来的一位为读 / 写状态位 (即第 8 位)，将被保存到 IICC1 寄存器的 SRW 位，从机随后发出一个低电平应答信号 (即第 9 位)。当从机的地址匹配时，从机会将状态标志位 HAAS 置位。

I²C 总线有三个中断源，当程序运行至中断服务子程序时，通过检测 HAAS 位和 IICTOF 位，以确定 I²C 总线中断是来自从机地址匹配，还是来自 8 位数据传递完毕，或是来自 I²C 超时。当是从机地址匹配发生中断时，则从机或是用于发送模式并将数据写进 IICD 寄存器，或是用于接收模式并从 IICD 寄存器中读取空值以释放 SCL 线。

I²C 总线读 / 写信号

IICC1 寄存器的 SRW 位用来表示主机是要从 I²C 总线上读取数据还是要将数据写到 I²C 总线上。从机则通过检测该位以确定自己是作为发送方还是接收方。当 SRW 置“1”，表示主机要从 I²C 总线上读取数据，从机则作为发送方，将数据写到 I²C 总线；当 SRW 清“0”，表示主机要写数据到 I²C 总线上，从机则做为接收方，从 I²C 总线上读取数据。

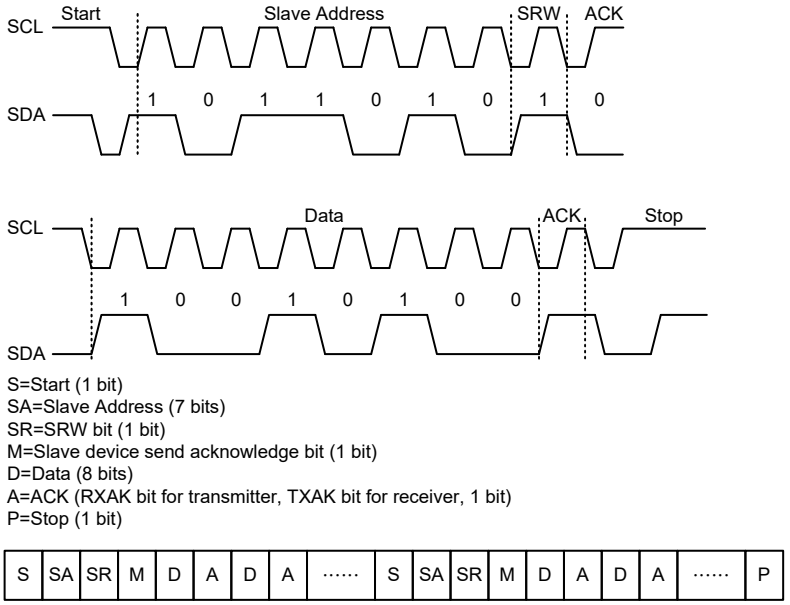
I²C 总线从机地址应答信号

主机发送呼叫地址后，当 I²C 总线上的任何从机内部地址与其匹配时，会发送一个应答信号。此应答信号会通知主机有从机已经接收到了呼叫地址。如果主机没有收到应答信号，则主机必须发送停止 (STOP) 信号以结束通信。当 HAAS 为高时，表示从机接收到的地址与自己内部地址匹配，则从机需检查 SRW 位，以确定自己是作为发送方还是作为接收方。如果 SRW 位为高，从机须设置成发送方，这样会置位 IICC1 寄存器的 HTX 位。如果 SRW 位为低，从机须设置成接收方，这样会清零 IICC1 寄存器的 HTX 位。

I²C 总线数据和应答信号

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

当接收器想要继续接收下一个数据时，必须在第 9 个时钟发出应答信号 (TXAK)。被设为发送方的从机将检测寄存器 IICC1 中的 RXAK 位以判断是否传输下一个字节的数据，如果单片机不传输下一个字节，那么它将释放 SDA 线并等待接收主机的停止信号。

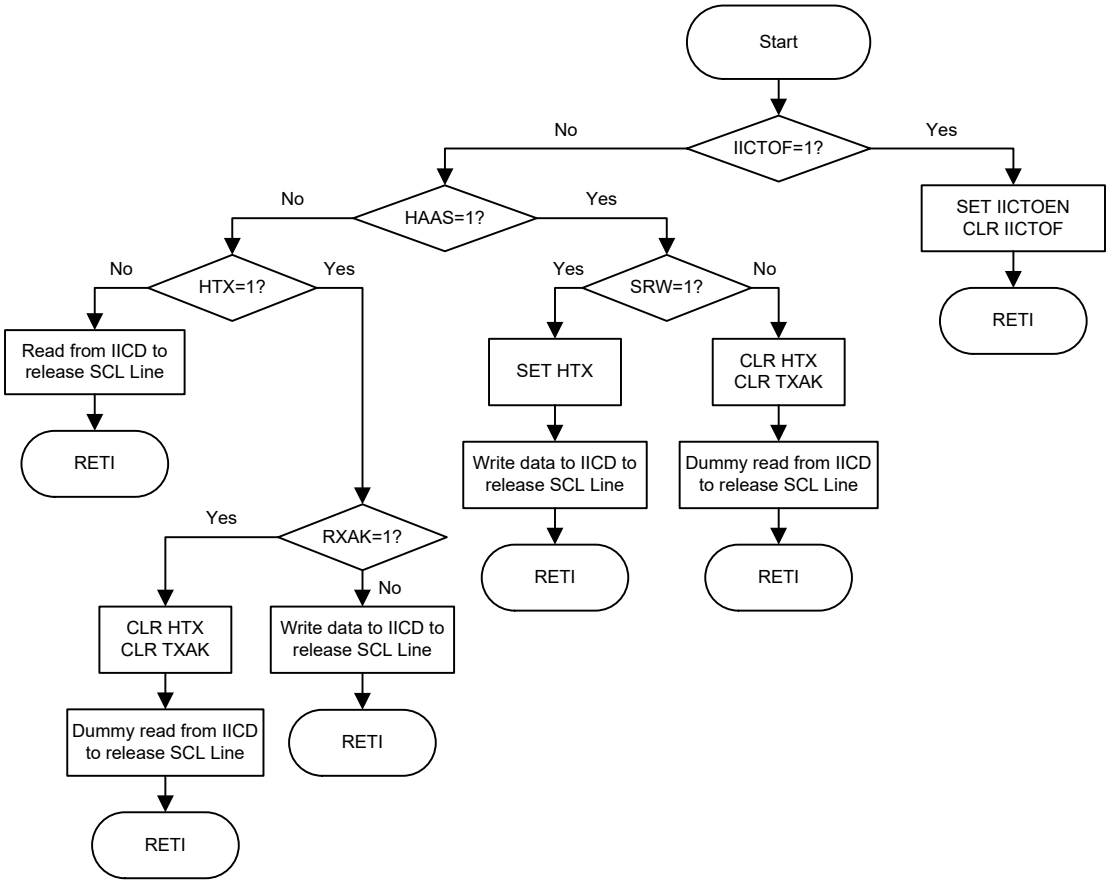


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	·····	S	SA	SR	M	D	A	D	A	·····	P

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

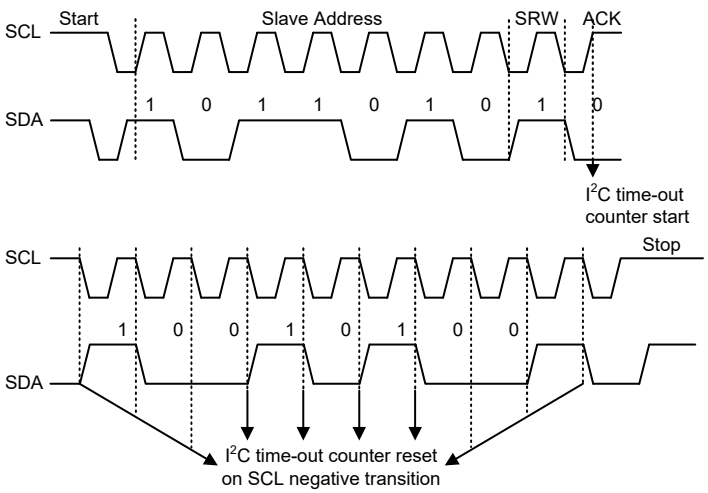
I^2C 通信时序图



I²C 总线 ISR 流程图

I²C 超时控制

超时功能可减少由于接收错误的时钟源而引起的 I²C 锁死问题。在一定时间内如果 I²C 总线未接收到时钟源，则在一定的超时周期后，I²C 电路和寄存器将会复位。超时计数器在 I²C 总线接收到“START”信号和“地址匹配”条件下开始计数，并在 SCL 下降沿处清零。在下一个 SCL 下降沿来临之前，如果等待时间大于 IICTOC 寄存器设定的超时时间，则会发生超时现象。当 I²C “STOP”条件发生时，超时计数器将停止计数。



I²C 超时

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

寄存器	I²C 超时发生后
IICD，IICA，IICC0	保持不变
IICC1	复位至 POR

IICTOF 标志位由应用程序清零。共有 64 个超时时钟周期选项可由 IICTOC 寄存器里的相应位来设置。超时时间的计算方法如下：

$$((1\sim64)\times32)/f_{SUB}$$

由此可得超时周期范围为 1ms~64ms。

- **IICTOC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0
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 **IICTOEN**：I²C 超时控制位
0：除能
1：使能

Bit 6 **IICTOF**：I²C 超时标志位
0：超时未发生
1：超时发生

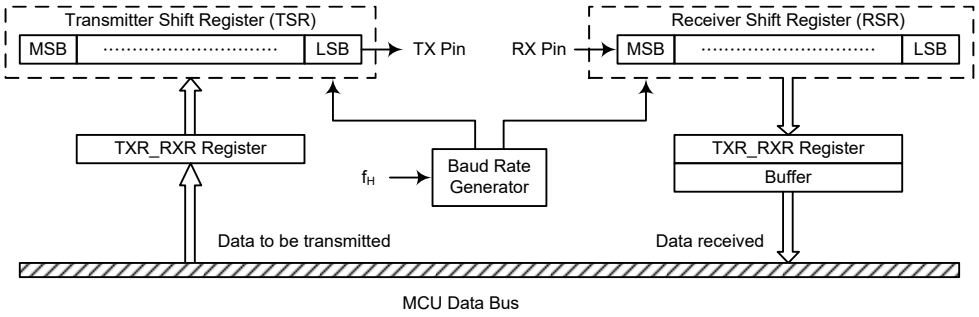
Bit 5~0 **IICTOS5~IICTOS0**：I²C 超时时间选择位
I²C 超时时钟源是 $f_{SUB}/32$
I²C 超时时间计算公式：(IICTOS[5:0]+1)×($32/f_{SUB}$)

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。

内置的 UART 功能包含以下特性：

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



UART 数据传输方框图

UART 外部引脚

内部 UART 有两个外部引脚 TX 和 RX，可与外部串行接口进行通信。TX 和 RX 分别为 UART 发送脚和接收脚，与 I/O 口或其它功能共用引脚。在使用 UART 功能前，应先通过相应的引脚共用功能选择寄存器，选择 TX 和 RX 引脚功能。当 UARTEN、TXEN 和 RXEN 位置高时，将自动设置这些 I/O 脚或其它共用功能脚作为 TX 输出和 RX 输入，并且除能 TX 和 RX 引脚上的上拉电阻功能。当 UARTEN、TXEN 或 RXEN 位清零除能 TX 或 RX 引脚功能后，TX 或 RX 引脚将处于浮空状态。这时 TX 或 RX 引脚是否连接内部上拉电阻是由相应的 I/O 上拉电阻控制位决定的。

UART 数据传输方案

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

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

需要注意的是，发送和接收都是共用同一个地址的数据寄存器，即 TXR_RXR 寄存器。

UART 状态和控制寄存器

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

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

UART 寄存器列表

- **USR 寄存器**

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

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

Bit 7 **PERR**：奇偶校验出错标志位
0：奇偶校验正确
1：奇偶校验出错
PERR 是奇偶校验出错标志位。若 PERR=0，奇偶校验正确；若 PERR=1，接收到的数据奇偶校验出错。只有使能了奇偶校验此位才有效。可使用软件清除该标志位，即先读取 USR 寄存器再读 TXR_RXR 寄存器来清除此位。

Bit 6 **NF**：噪声干扰标志位
0：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 TXR_RXR 寄存器将清除此标志位。

Bit 5 **FERR**：帧错误标志位
0：无帧错误发生
1：有帧错误发生
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 TXR_RXR 寄存器来清除此位。

Bit 4 **OERR**：溢出错误标志位
0：无溢出错误发生
1：有溢出错误发生
OERR 是溢出错误标志位，表示接收缓冲器是否溢出。若 OERR=0，没有溢出错误；若 OERR=1，发生了溢出错误，它将禁止下一组数据的接收。可通过软件清除该标志位，即先读取 USR 寄存器再读 TXR_RXR 寄存器将清除此标志位。

Bit 3 **RIDLE**：接收状态标志位
0：正在接收数据
1：接收器空闲
RIDLE 是接收状态标志位。若 RIDLE=0，正在接收数据；若 RIDLE=1，接收器空闲。在接收到停止位和下一个数据的起始位之间，RIDLE 被置位，表明 UART 空闲，RX 脚处于逻辑高状态。

Bit 2 **RXIF**：接收寄存器状态标志位
0：TXR_RXR 寄存器为空
1：TXR_RXR 寄存器含有有效数据
RXIF 是接收寄存器状态标志位。当 RXIF=0，TXR_RXR 寄存器为空；当 RXIF=1，TXR_RXR 寄存器接收到新数据。当数据从移位寄存器加载到 TXR_RXR 寄存器中，如果 UCR2 寄存器中的 RIE=1，则会触发中断。当接收数据时检测到一个或多个错误时，相应的标志位 NF、FERR 或 PERR 会在同一周期内置位。读取 USR 寄存器再读 TXR_RXR 寄存器，如果 TXR_RXR 寄存器中没有新的数据，那么将清除 RXIF 标志。

Bit 1 **TIDLE**：数据发送完成标志位
0：数据传输中
1：无数据传输
TIDLE 是数据发送完成标志位。若 TIDLE=0，数据传输中。当 TXIF=1 且数据发送完毕或者暂停字被发送时，TIDLE 置位。TIDLE=1，TX 引脚空闲且处于逻辑高状态。读取 USR 寄存器再写 TXR_RXR 寄存器将清除 TIDLE 位。数据字符或暂停字就绪时，不会产生该标志位。

Bit 0 **TXIF**：发送数据寄存器 TXR_RXR 状态位
0：数据还没有从缓冲器加载到移位寄存器中
1：数据已从缓冲器加载到移位寄存器中 (TXR_RXR 数据寄存器为空)
TXIF 是发送数据寄存器为空标志位。若 TXIF=0，数据还没有从缓冲器加载到移位寄存器中；若 TXIF=1，数据已从缓冲器中加载到移位寄存器中。读取 USR 存器再写 TXR_RXR 寄存器将清除 TXIF。当 TXEN 被置位，由于发送缓冲器未满，TXIF 也会被置位。

- **UCR1 寄存器**

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

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

“x”：未知

Bit 7 **UARTEN**：UART 功能使能位
0：UART 除能，RX 和 TX 可用作普通输入 / 输出口
1：UART 使能，TX 和 RX 脚作为 UART 功能引脚
此位为 UART 的使能位。UARTEN=0，UART 除能，RX 和 TX 可用作普通输入 / 输出口；UARTEN=1，UART 使能，TX 和 RX 将分别由 TXEN 和 RXEN 控制。当 UART 被除能将清除缓冲器，所有缓冲器中的数据将被忽略，另外波特率计数器、错误和状态标志位被复位，TXEN、RXEN、TXBRK、RXIF、OERR、FERR、PERR 和 NF 清零，而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2 和 BRG 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零，所有发送和接收将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

Bit 6 **BNO**：发送数据位数选择位
0：8-bit 传输数据
1：9-bit 传输数据
BNO 是发送数据位数选择位。BNO=1，传输数据为 9 位；BNO=0，传输数据为 8 位。若选择了 9 位数据传输格式，RX8 和 TX8 将分别存储接收和发送数据的第 9 位。
注：1. 若 BNOn =1(9-bit 数据传输格式)，奇偶校验使能时，数据的 9th data 为奇偶校验位，不会传送到 RX8.
2. 若 BNOn =0(8-bit 数据传输格式)，奇偶校验使能时，数据的 8th data 为奇偶校验位，不会传送到 RX7.

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

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

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

Bit 2 **TXBRK**：暂停字发送控制位
0：没有暂停字要发送
1：发送暂停字
TXBRK 是暂停字发送控制位。TXBRK=0，没有暂停字要发送，TX 引脚正常操作；TXBRK=1，将会发送暂停字，发送器将发送逻辑“0”。若 TXBRK 为高，缓冲器中数据发送完毕后，发送器输出将至少保持 13 位宽的低电平直至 TXBRK 复位。

Bit 1 **RX8**：接收 9-bit 数据传输格式中的第 9 位 (只读)
此位只有在传输数据为 9 位的格式中有效，用来存储接收数据的第 9 位。BNO 是用来控制传输位数是 8 位还是 9 位。

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

- **UCR2 寄存器**

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

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

Bit 7 **TXEN**：UART 发送使能位
0：UART 发送除能
1：UART 发送使能
此位为发送使能位。TXEN=0，发送将被除能，发送器立刻停止工作。另外发送缓冲器将被复位，此时 TX 可用作普通输入 / 输出口。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 可用作普通输入 / 输出口。

Bit 6 **RXEN**：UART 接收使能位
0：UART 接收除能
1：UART 接收使能
此位为接收使能位。RXEN=0，接收将被除能，接收器立刻停止工作。另外接收缓冲器将被复位，此时 RX 可用作普通输入 / 输出口。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 可用作普通输入 / 输出口。

Bit 5 **BRGH**：波特率发生器高低速选择位
0：低速波特率
1：高速波特率
此位为波特率发生器高低速选择位，它和 BRG 寄存器一起控制 UART 的波特率。BRGH=1，为高速模式；BRGH=0，为低速模式。

Bit 4 **ADDEN**：地址检测使能位
0：地址检测除能
1：地址检测使能
此位为地址检测使能和除能位。ADDEN=1，地址检测使能，此时数据的第 8 位 (BNO=0) 或第 9 位 (BNO=1) 为高，那么接到的是地址而非数据。若相应的中断使能且接收到的值最高位为 1，那么中断请求标志将会被置位，若地址检测功能使能且最高位为 0，那么将不会产生中断且收到的数据也会被忽略。

Bit 3 **WAKE**：RX 脚下降沿唤醒 UART 功能使能位
0：RX 脚下降沿唤醒 UART 功能除能
1：RX 脚下降沿唤醒 UART 功能使能
此位用于控制 RX 引脚下降沿时是否唤醒 UART 功能。此位仅当 UART 时钟源 f_H 关闭时有效。若 UART 时钟源 f_H 还开启，则无 RX 引脚唤醒 UART 功能无效。若此位置高且 UART 时钟 f_H 关闭，当 RX 引脚发生下降沿时会产生 UART 唤醒请求。若相应的中断使能，将产生 RX 引脚唤醒 UART 的中断，以告知单片机使其通过应用程序开启 UART 时钟源 f_H，从而唤醒 UART 功能。否则，若此位为低，即使 RX 引脚发生下降沿也无法恢复 UART 功能。

Bit 2 **RIE**：接收中断使能位
0：接收中断除能
1：接收中断使能
此位为接收中断使能或除能位。若 RIE=1，当 OERR 或 RXIF 置位时，UART 的中断请求标志置位；若 RIE=0，UART 中断请求标志不受 OERR 和 RXIF 影响。

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

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

- **TXR_RXR 寄存器**

TXR_RXR 是一个数据寄存器，用来存储 TX 引脚将要发送或 RX 引脚正在接收的数据。

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

“x”：未知

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

- **BRG 寄存器**

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

“x”：未知

Bit 7~0 **BRG7~BRG0**：波特率值
软件设置 BRGH 位 (设置波特率发生器的速度) 和 BRG 寄存器 (设置波特率的值)，一起控制 UART 的波特率。
注：若 BRGH=0，波特率 = $f_H/[64\times(N+1)]$；
若 BRGH=1，波特率 = $f_H/[16\times(N+1)]$。

波特率发生器

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

UCR2 的 BRGH 位	0	1
波特率 (BR)	f_H / [64 (N+1)]	f_H / [16 (N+1)]

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

下面举例怎样计算 BRG 寄存器中的值 N 和误差。

波特率和误差的计算

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

根据上表，波特率 BR = f_H / [64 (N+1)]

转换后的公式 N = [f_H / (BR×64)] - 1

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

取最接近的值，十进制 12 写入 BRG 寄存器，实际波特率如下

BR = 4000000 / [64 × (12+1)] = 4808

因此，误差 = (4808 - 4800) / 4800=0.16%

UART 模块的设置与控制

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

UART 的使能和除能

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

UARTEN 清零将除能 TX 和 RX，通过设置相关引脚共用控制位，这两个引脚可用作普通 I/O 口或其它引脚共用功能。当 UART 被除能时将清空缓冲器，所有缓冲器中的数据将被忽略，另外一些使能控制、错误标志和状态标志将被复位，如 TXEN、RXEN、TXBRK、RXIF、OERR、FERR、PERR 和 NF 清 零，而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2 和 BRG 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零，所有发送和接收将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

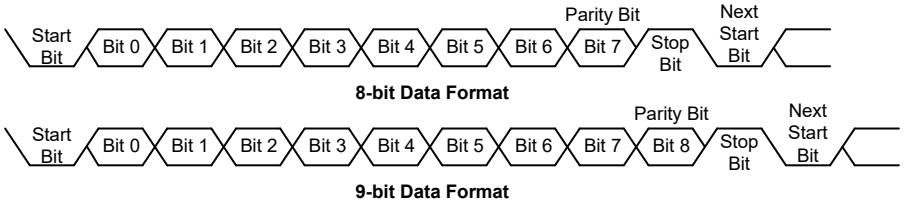
数据位、停止位位数以及奇偶校验的选择

数据传输格式由数据长度、是否校验、校验类型、地址位以及停止位长度组成。它们都是由 UCR1 寄存器的各个位控制的。BNO 决定数据传输是 8 位还是 9 位；PRT 决定校验类型；PREN 决定是否选择奇偶校验；而 STOPS 决定选用 1 位还是 2 位停止位。下表列出了各种数据传输格式。若地址检测功能使能，地址位，即数据字节的最高位，用来确定此帧是地址还是数据。停止位的长度和数据位的长度无关，且只有发送器需设置停止位长度。接收器只接收一个停止位。

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

发送和接收数据格式

下图是传输 8 位和 9 位数据的波形。



UART 发送器

UCR1 寄存器的 BNO 位是控制数据传输的长度。BNO=1 其长度为 9 位，第 9 位 MSB 存储在 UCR1 寄存器的 TX8 中。发送器的核心是发送移位寄存器 TSR，它的数据由发送寄存器 TXR_RXR 提供，应用程序只须将发送数据写入 TXR_RXR 寄存器。上组数据的停止位发出前，TSR 寄存器禁止写入。如果还有新的数据要发送，一旦停止位发出，待发数据将会从 TXR_RXR 寄存器加载到 TSR 寄存器。TSR 不像其它寄存器一样映射到数据存储器，所以应用程序不能对其进行读写操作。TXEN=1，发送使能，但若 TXR_RXR 寄存器没有数据或者波特率没有设置，发送器将不会工作。先写 TXR_RXR 寄存器再置高 TXEN 也会触发发送。当发送器使能，若 TSR 寄存器为空，数据写入 TXR_RXR 寄存器将会直接加载到 TSR 寄存器中。发送器工作时，TXEN 清零，发送器将立刻停止工作并且复位，此时通过设置相关引脚共用控制位，TX 引脚用作普通 I/O 口或其它引脚共用功能。

发送数据

当 UART 发送数据时，数据从移位寄存器中移到 TX 引脚上，其低位在前高位在后。在发送模式中，TXR_RXR 寄存器在内部总线和发送移位寄存器间形成一个缓冲。如果选择 9 位数据传输格式，最高位 MSB 取自 UCR1 寄存器的 TX8。

发送器初始化可由如下步骤完成：

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

如果要发送多个数据只需重复上一步骤。

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

1. 读取 USR 寄存器
2. 写 TXR_RXR 寄存器

只读标志位 TXIF 由 UART 硬件置位。若 TXIF=1，TXR_RXR 寄存器为空，其它数据可以写入而不会覆盖之前的数据。若 TEIE=1，TXIF 标志位会产生中断。在数据传输时，写 TXR_RXR 指令会将待发数据暂存在 TXR_RXR 寄存器中，当前数据发送完毕后，待发数据被加载到发送移位寄存器中。当发送器空闲时，写 TXR_RXR 指令会将数据直接加载到 TSR 寄存器中，数据传输立刻开始且 TXIF 置位。当发送完停止位或暂停帧后，表示一帧数据已发送完毕，此时 TIDLE 位将被置位。

可以通过以下步骤来清除 TIDLE：

1. 读取 USR 寄存器
2. 写 TXR_RXR 寄存器

清除 TXIF 和 TIDLE 软件执行次序相同。

发送暂停字

若 TXBRK=1，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。需要注意的是，暂停字至少 13 位宽。若 TXBRK 持续为高，那么发送器会一直发送暂停字；当应用程序将 TXBRK 清零后，发送器结束最后一帧暂停字的发送后接着发送一位或两位停止位。最后一帧暂停字的结尾自动为高电平，以确保下一帧数据起始位的检测。

UART 接收器

UART 接收器支持 8 位或者 9 位数据接收。若 BNO=1，数据长度为 9 位，而最高位 MSB 存放在 UCR1 寄存器的 TXR_RXR 中。接收器的核心是串行移位寄存器 RSR。RX 引脚上的数据送入数据恢复器中，它在 16 倍波特率的频率下工作，而串行移位器工作在正常波特率下。当在 RX 引脚上检测到停止位，若 TXR_RXR 寄存器为空，数据从 RSR 寄存器中加载到 TXR_RXR 寄存器。RX 引脚上的每一位数据会被采样三次以判断其逻辑状态。RSR 不像其它寄存器一样映射在数据存储器，所以应用程序不能对其进行读写操作。

接收数据

当 UART 接收数据时，数据低位在前高位在后，连续地从 RX 引脚进入移位寄存器。TXR_RXR 寄存器在内部总线和接收移位寄存器间形成一个缓冲。TXR_RXR 寄存器是一个两层的 FIFO 缓冲器，它能保存两帧数据的同时接收第三帧数据，应用程序必须保证在接收完第三帧前读取 TXR_RXR 寄存器，否则忽略第三帧数据并且发生溢出错误。

接收器的初始化可由如下步骤完成：

- 正确地设置 BNO、PRT 和 PREN 位以确定数据长度和校验类型。
- 设置 BRG 寄存器，选择期望的波特率。
- 置高 RXEN，使能 UART 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 TXR_RXR 寄存器中包含有效数据时，USR 寄存器中的 RXIF 位将会置位，溢出错误发生之前至多还有一帧数据可读。
- 若 RIE=1，数据从 RSR 寄存器加载到 TXR_RXR 寄存器中将产生中断。
- 若接收器检测到帧错误、噪声干扰错误、奇偶出错或溢出错误，那么相应的错误标志位置位。

可以通过如下步骤来清除 RXIF：

1. 读取 USR 寄存器
2. 读取 TXR_RXR 寄存器

接收暂停字

UART 接收任何暂停字都会当作帧错误处理。接收器只根据 BNO 位的设置外加一个停止位来确定一帧数据的长度。若暂停字位数大于 BNO 位指定的长度外加一个停止位，接收器认为接收已完毕，RXIF 和 FERR 置位，TXR_RXR 寄存器清 0，若相应的中断允许且 RIDLE 为高将会产生中断。暂停字只会被认为包含信息 0 且会置位 FERR 标志位。如果检测到较长的暂停信号，接收器会将此信号视为包含一个起始位、数据位和无效的停止位的数据帧并且置位 FERR 标志位。在下个开始位到来之前，接收器必须等待一个有效的停止位。接收器不会假定线上的暂停信号是下一个开始位。暂停字将会加载到缓冲器中，在接收到停止位前不会再接收数据，没有检测到停止位也会置位只读标志位 RIDLE。

UART 接收到暂停字会产生以下事件：

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

空闲状态

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

接收中断

USR 寄存器的只读标志位 RXIF 由接收器的边沿触发置位。若 RIE=1，数据从移位寄存器 RSR 加载到 TXR_RXR 寄存器时产生中断，同样地，溢出也会产生中断。

接收错误处理

UART 会产生几种接收错误，下面部分将描述各错误以及怎样处理。

溢出 – OERR 标志

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

产生溢出错误时将会发生以下事件：

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

先读取 USR 寄存器再读取 TXR_RXR 寄存器可将 OERR 清零。

噪声干扰 – NF 标志

数据恢复时多次采样可以有效的鉴别出噪声干扰。当检测到数据受到噪声干扰时将会发生以下事件：

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

先读取 USR 寄存器再读取 TXR_RXR 寄存器可将 NF 清零。

帧错误 – FERR 标志

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

奇偶校验错误 – PERR 标志

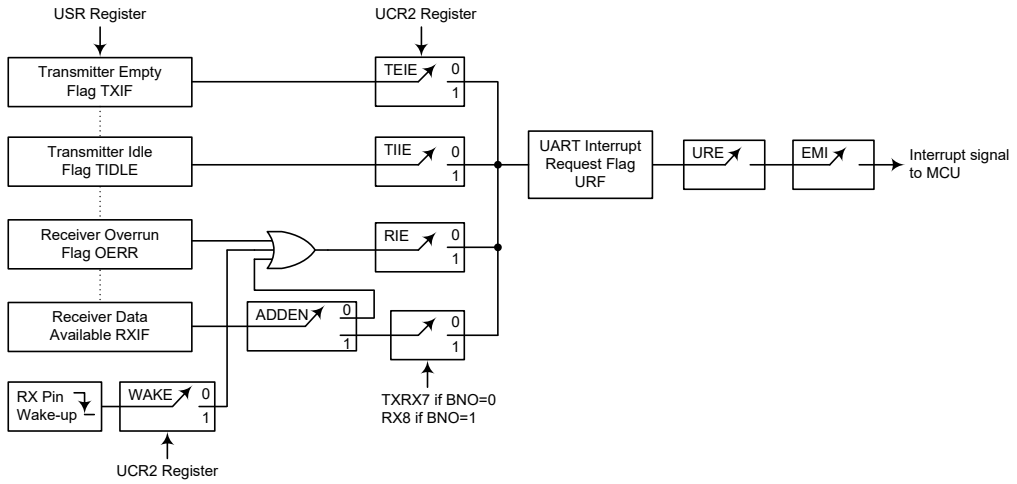
若接收到数据出现奇偶校验错误，USR 寄存器中只读标志 PERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。此标志位同接收的数据分别记录在 USR 寄存器和 TXR_RXR 寄存器中，此标志位可被任何复位清零。注意，在读取相应的数据之前必须先访问 USR 寄存器中的 FERR 和 PERR 错误标志位。

UART 模块中断结构

几个独立的 UART 条件可以产生一个 UART 中断。当条件满足时，会产生一个低脉冲信号。发送寄存器为空、发送器空闲、接收器数据有效、溢出和地址检测和 RX 引脚唤醒都会产生中断。若 UART 中断允许且堆栈未满，程序将会跳转到相应的中断向量执行中断服务程序，而后再返回主程序。其中四种情况，若其 UCR2 寄存器中相应中断允许位被置位，则 USR 寄存器中对应中断标志位将产生 UART 中断。发送器相关的两个中断情况有各自对应的中断允许位，而接收器相关的两个中断情况共用一个中断允许位。这些允许位可用于禁止个别的 UART 中断源。

地址检测也是 UART 的中断源，它没有相应的标志位，若 UCR2 寄存器中 ADDEN=1，当检测到地址将会产生 UART 中断。RX 引脚唤醒也可以产生 UART 中断，它没有相应的标志位，当 UART 时钟源 f_H 关闭且 UXR2 中的 WAKE 和 RIE 位被置位，RX 引脚上有下降沿时会产生 UART 中断。

注意，USR 寄存器标志位为只读状态，软件不能对其进行设置，和其它一些中断一样，在进入相应中断服务程序时也不能清除这些标志位。这些标志位仅在 UART 特定动作发生时才会自动被清除，详细解释见 UART 寄存器章节。整体 UART 中断的使能或除能可由中断控制寄存器中的相关中断使能控制位控制，其中断请求由 UART 模块决定。



UART 中断框图

地址检测模式

置位 UCR2 寄存器中的 ADDEN 将启动地址检测模式。若此位为“1”，可产生接收数据有效中断，其请求标志位为 RXIF。若 ADDEN 有效，只有在接收到数据最高位为 1 才会产生中断，中断允许位 URE 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，为了确保操作正确，必须将奇偶校验使能位清零以除能奇偶校验。

ADDEN	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

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

UART 功能中包括了 RX 引脚的唤醒功能，由 UCR2 寄存器中 WAKE 位控制。当 UART 时钟 f_H 关闭时，若 WAKE 位与 UART 允许位 UARTEN、接收器允许位 RXEN 和接收器中断允许位 RIE 都被置位，则 RX 引脚的下降沿可触发产生 RX 引脚唤醒 UART 的中断。唤醒后系统需延时一段时间才能正常工作，在此期间，RX 引脚上的任何数据将被忽略。

若要唤醒并产生 UART 中断，除了唤醒使能控制位和接收中断使能控制位需置位外，全局中断允许位 EMI 和 UART 中断使能控制位 URE 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。此单片机提供多个外部中断和内部中断功能，外部中断由 INT0~INT1 引脚动作产生，而内部中断由各种内部功能，如定时器模块、比较器、时基、LVD、EEPROM 和 A/D 转换器等产生。

中断寄存器

中断控制基本上是在一定单片机条件发生时设置请求标志位，应用程序中中断使能位的设置是通过位于专用数据存储器中的一系列寄存器控制的。寄存器总的分为三类。第一类是 INTC0~INTC3 寄存器，用于设置基本的中断；第二类是 MFI0~MFI3 寄存器，用于设置多功能中断；最后一种为 INTEG 寄存器，用于设置外部中断边沿触发类型。

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

功能		使能位	请求标志	注释
总中断		EMI	—	—
INTn 脚		INTnE	INTnF	n = 0 或 1
比较器	SYNC 中断	CPnE	CPnF	n = 0
	过压中断			n = 1
	过零中断			n = 2
	SUG 中断			n = 3
	过流中断			n = 4
PPG 自动记录同步信号次数中断		SYNCF	SYNCE	—
UART		URE	URF	—
I^2C		IICE	IICF	—
多功能		MFnE	MFnF	n = 0~3
A/D 转换器		ADE	ADF	—
时基		TBnE	TBnF	n = 0 或 1
EEPROM 写操作		DEE	DEF	—
LVD		LVE	LVF	—
定时 / 计数器溢出		TE	TF	—
CTM		CTMnPE	CTMnPF	n = 0 或 1
		CTMnAE	CTMnAF	
PTM		PTMPE	PTMPF	—
		PTMAE	PTMAF	

中断寄存器位命名模式

寄存器名称	位							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	CP1F	CP0F	INT0F	CP1E	CP0E	INT0E	EMI
INTC1	SYNCF	CP4F	CP3F	CP2F	SYNCE	CP4E	CP3E	CP2E
INTC2	TF	TB1F	MF0F	IICF	TE	TB1E	MF0E	IICE
INTC3	MF3F	MF2F	MF1F	URF	MF3E	MF2E	MF1E	URE
MFI0	—	TB0F	PTMAF	PTMPF	—	TB0E	PTMAE	PTMPE
MFI1	—	ADF	CTM0AF	CTM0PF	—	ADE	CTM0AE	CTM0PE
MFI2	—	DEF	CTM1AF	CTM1PF	—	DEE	CTM1AE	CTM1PE
MFI3	—	—	LVF	INT1F	—	—	LVE	INT1E

中断寄存器列表

- **INTEG 寄存器**

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

Bit 7~4 未定义，读为“0”

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

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

- **INTC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	CP1F	CP0F	INT0F	CP1E	CP0E	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 未定义，读为“0”

Bit 6 **CP1F**：比较器 1 中断请求标志位
0：无请求
1：中断请求

Bit 5 **CP0F**：比较器 0 中断请求标志位
0：无请求
1：中断请求

Bit 4 **INT0F**：INT0 中断请求标志位
0：无请求
1：中断请求

Bit 3 **CP1E**：比较器 1 中断控制位
0：除能
1：使能

Bit 2 **CP0E**：比较器 0 中断控制位
0：除能
1：使能

Bit 1 **INT0E**：INT0 中断控制位
0：除能
1：使能

Bit 0 **EMI**：总中断控制位
0：除能
1：使能

- **INTC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SYNCF	CP4F	CP3F	CP2F	SYNCE	CP4E	CP3E	CP2E
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 **SYNCF**：自动记录同步信号次数中断请求标志位
0：无请求
1：中断请求

Bit 6 **CP4F**：比较器 4 中断请求标志位
0：无请求
1：中断请求

Bit 5 **CP3F**：比较器 3 中断请求标志位
0：无请求
1：中断请求

Bit 4 **CP2F**：比较器 2 中断请求标志位
0：无请求
1：中断请求

Bit 3 **SYNCE**：自动记录同步信号次数中断控制位
0：除能
1：使能

Bit 2 **CP4E**：比较器 4 中断控制位
0：除能
1：使能

Bit 1 **CP3E**：比较器 3 中断控制位
0：除能
1：使能

Bit 0 **CP2E**：比较器 2 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	TF	TB1F	MF0F	IICF	TE	TB1E	MF0E	IICE
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 **TF**：定时 / 计数器溢出中断请求标志位
0：无请求
1：中断请求

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

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

Bit 4 **IICF**：I^2C 总线中断请求标志位
0：无请求
1：中断请求

Bit 3 **TE**：定时 / 计数器溢出中断控制位
0：除能
1：使能

Bit 2 **TB1E**：时基 1 控制位
0：除能
1：使能

Bit 1 **MF0E**：多功能中断 0 控制位
0：除能
1：使能

Bit 0 **IICE**：I^2C 总线中断控制位
0：除能
1：使能

- **INTC3 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MF3F	MF2F	MF1F	URF	MF3E	MF2E	MF1E	URE
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 **MF3F**：多功能中断 3 请求标志位
0：无请求
1：中断请求

Bit 6 **MF2F**：多功能中断 2 请求标志位
0：无请求
1：中断请求

Bit 5 **MF1F**：多功能中断 1 请求标志位
0：无请求
1：中断请求

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

Bit 3 **MF3E**：多功能中断 3 控制位
0：除能
1：使能

Bit 2 **MF2E**：多功能中断 2 控制位
0：除能
1：使能

Bit 1 **MF1E**：多功能中断 1 控制位
0：除能
1：使能

Bit 0 **URE**：UART 中断控制位
0：除能
1：使能

- **MFI0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	TB0F	PTMAF	PTMPF	—	TB0E	PTMAE	PTMPE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7　未定义，读为“0”

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

Bit 5　**PTMAF**：PTM 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4　**PTMPF**：PTM 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求

Bit 3　未定义，读为“0”

Bit 2　**TB0E**：时基 0 中断控制位
0：除能
1：使能

Bit 1　**PTMAE**：PTM 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0　**PTMPE**：PTM 比较器 P 匹配中断控制位
0：除能
1：使能

- **MFI1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	ADF	CTM0AF	CTM0PF	—	ADE	CTM0AE	CTM0PE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7　未定义，读为“0”

Bit 6　**ADF**：A/D 转换器中断请求标志位
0：无请求
1：中断请求

Bit 5　**CTM0AF**：CTM0 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4　**CTM0PF**：CTM0 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求

Bit 3　未定义，读为“0”

Bit 2　**ADE**：A/D 转换器中断控制位
0：除能
1：使能

Bit 1　**CTM0AE**：CTM0 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0　**CTM0PE**：CTM0 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	—	DEF	CTM1AF	CTM1PF	—	DEE	CTM1AE	CTM1PE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7　未定义，读为“0”

Bit 6　**DEF**：数据 EEPROM 中断请求标志位
0：无请求
1：中断请求

Bit 5　**CTM1AF**：CTM1 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4　**CTM1PF**：CTM1 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求

Bit 3　未定义，读为“0”

Bit 2　**DEE**：数据 EEPROM 中断控制位
0：除能
1：使能

Bit 1　**CTM1AE**：CTM1 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0　**CTM1PE**：CTM1 比较器 P 匹配中断控制位
0：除能
1：使能

- **MFI3 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVF	INT1F	—	—	LVE	INT1E
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　**LVF**：LVD 中断请求标志位
0：无请求
1：中断请求

Bit 4　**INT1F**：INT1 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　**LVE**：LVD 中断控制位
0：除能
1：使能

Bit 0　**INT1E**：INT1 中断控制位
0：除能
1：使能

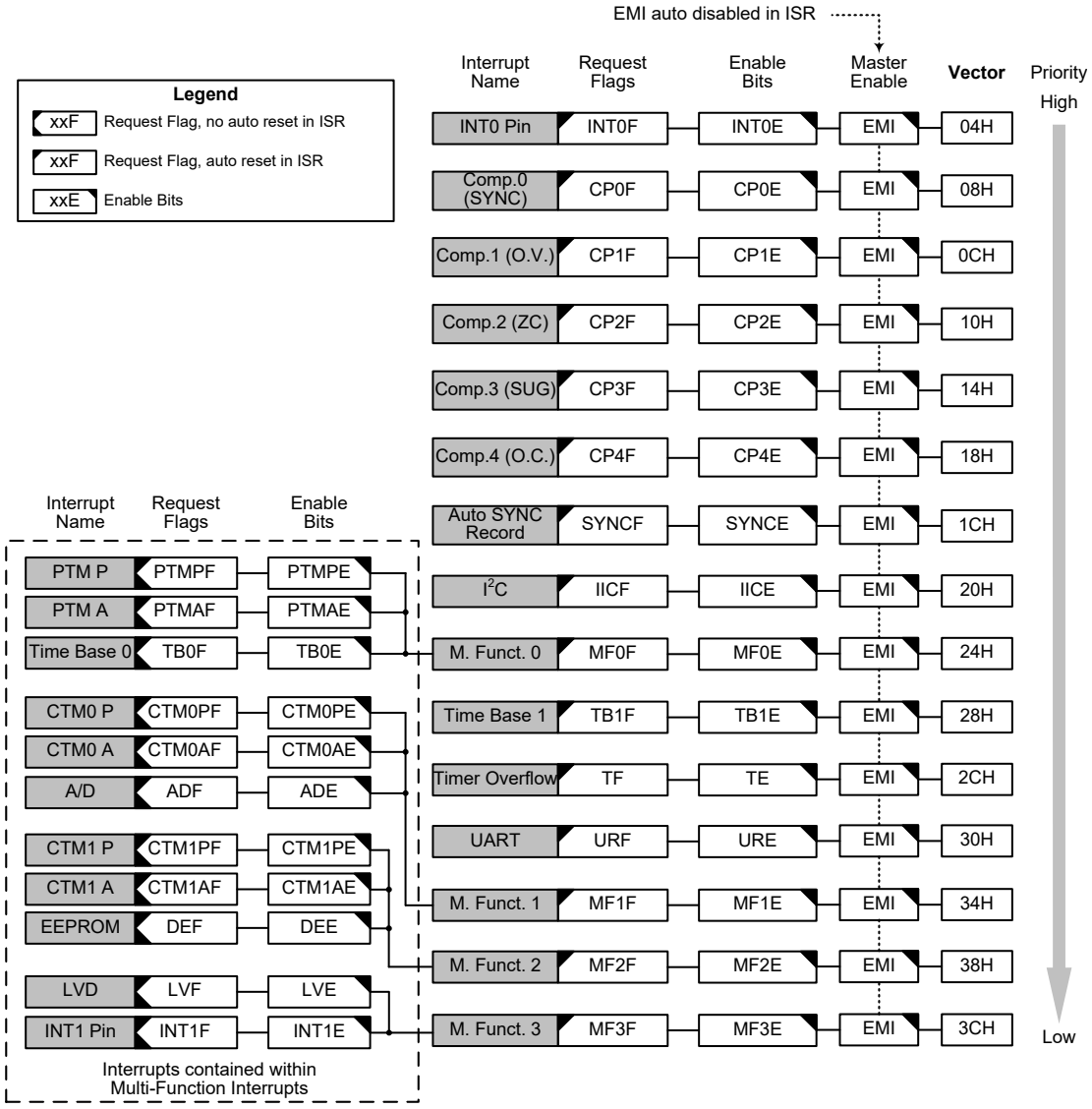
中断操作

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。


EMI auto disabled in ISR
Legend
xxF Request Flag, no auto reset in ISR
xxF Request Flag, auto reset in ISR
xxE Enable Bits
Interrupt Name
Request Flags
Enable Bits
Master Enable
Vector
Priority
High
Low
INT0 Pin INT0F INT0E EMI 04H
Comp.0 (SYNC) CP0F CP0E EMI 08H
Comp.1 (O.V.) CP1F CP1E EMI 0CH
Comp.2 (ZC) CP2F CP2E EMI 10H
Comp.3 (SUG) CP3F CP3E EMI 14H
Comp.4 (O.C.) CP4F CP4E EMI 18H
Auto SYNC Record SYNCF SYNCE EMI 1CH
I2C IICF IICE EMI 20H
M. Funct. 0 MF0F MF0E EMI 24H
Time Base 1 TB1F TB1E EMI 28H
Timer Overflow TF TE EMI 2CH
UART URF URE EMI 30H
M. Funct. 1 MF1F MF1E EMI 34H
M. Funct. 2 MF2F MF2E EMI 38H
M. Funct. 3 MF3F MF3E EMI 3CH
PTM P PTMPF PTMPE
PTM A PTMAF PTMAE
Time Base 0 TB0F TB0E
CTM0 P CTM0PF CTM0PE
CTM0 A CTM0AF CTM0AE
A/D ADF ADE
CTM1 P CTM1PF CTM1PE
CTM1 A CTM1AF CTM1AE
EEPROM DEF DEE
LVD LVF LVE
INT1 Pin INT1F INT1E
Interrupts contained within Multi-Function Interrupts

中断结构

外部中断

此单片机有两个外部中断，通过 INT0~INT1 引脚上的信号变化控制，INT0 中断为独立的中断，INT1 中断属于多功能中断。

要产生 INT0 中断，总中断使能位 EMI 和相应的外部中断使能位 INT0E 需先被置位。当触发沿选择位设置好触发类型，INT0 引脚的状态发生变化，相应的外部中断请求标志位 INT0F 将置位并触发 INT0 中断。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。当外部中断服务子程序响应时，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。

要产生 INT1 中断，总中断使能位 EMI、相应的外部中断使能位 INT1E 和相应的多功能中断使能位需先被置位。当触发沿选择位设置好触发类型，INT1 引脚的状态发生变化，相应的外部中断请求标志位 INT1F 将置位并触发 INT1 中断。若中断使能，堆栈未满并且外部中断脚状态改变，可跳转至相关多功能中断向量子程序中执行。当外部中断服务子程序响应时，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 INT1 中断请求标志需在应用程序中手动清除。

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，需通过引脚共用寄存器选择外部中断脚，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。注意，即使此引脚被用作外部中断输入，其上拉电阻仍保持有效。寄存器 INTEG 被用来选择有效的边沿类型，来触发外部中断。可以选择上升沿还是下降沿或双沿触发都产生外部中断。注意 INTEG 也可以用来除能外部中断功能。

比较器中断

比较器有 5 个中断，由内部比较器 0~4 控制，分别对应于 SYNC 中断、过压 (O.V.) 中断、过零 (ZC) 中断、SUG 中断和过流 (O.C.) 中断。当相应的比较器输出位的状态改变，即检测到 SYNC、过压情况发生、检测到过零、检测到浪涌电压或过流情况发生时，相应的比较器中断请求标志 CPnF 被置位，比较器中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和比较器中断使能位 CPnE 需先被置位。当中断使能，堆栈未满并且以上任何一种情况发生使得比较器输出位的状态发生改变时，将调用相应的比较器中断向量子程序。当中断服务子程序响应时，比较器中断请求标志位会自动复位且 EMI 位会被清零以除能其它中断。

自动记录同步信号次数中断

当达到软件选择的自动记录同步信号次数的时间间隔时，会产生中断请求。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和相应中断使能位 SYNCE 需先被置位。当中断使能，堆栈未满且达到记录时间时，将调用相应的中断向量子程序。当中断响应后，SYNCF 中断请求标志位会自动复位且 EMI 位会被清零以除能其它中断。

I²C 总线中断

当一个字节数据已由 I²C 接口接收或发送完，或 I²C 从机地址匹配，或 I²C 超时发生，中断请求标志 IICF 被置位，I²C 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和 I²C 中断使能位 IICE 需先被置位。当中断使能，堆栈未满且以上任一种情况发生时，将调用 I²C 中断向量子程序。当 I²C 中断响应时，I²C 中断请求标志位会自动复位且 EMI 位会被清零以除能其它中断。

UART 中断

UART 传输中断由几种 UART 传输条件来控制。当发送器为空、发送器空闲、接收器数据有效、接收器溢出、地址检测和 RX 引脚唤醒，UART 中断请求标志 URF 被置位，UART 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和 UART 中断使能位 URE 需先被置位。当中断使能，堆栈未满且以上任何一种情况发生时，将调用 UART 中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 URF 会自动复位且 EMI 位会被清零以除能其它中断。然而 USR 寄存器里的标志位只有在对 UART 执行特定动作时才会被清零，详细参考 UART 章节。

多功能中断

此单片机中有多达 4 个多功能中断，与其它中断不同，它没有独立源，但由其它现有的中断源构成，即 TM 中断，时基 0 中断，A/D 中断，EEPROM 中断，INT1 中断和 LVD 中断。

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

但必须注意的是，在中断响应时，虽然多功能中断标志会自动复位，但多功能中断源的请求标志位，即 TM 中断，时基 0 中断，A/D 中断，EEPROM 中断，INT1 中断和 LVD 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 转换器中断属于多功能中断，由 A/D 转换动作的结束来控制。当 A/D 转换器中断请求标志 ADF 被置位，即 A/D 转换过程完成时，中断请求发生。若要跳转到相应中断向量地址，总中断控制位 EMI、A/D 转换器中断使能位 ADE 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 A/D 转换动作结束时，可跳转至相关多功能中断向量子程序中执行。当 A/D 转换器中断服务子程序响应时，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，但相应的中断请求标志位 ADF 需在应用程序中手动清除。

时基中断

此单片机有两个时基中断，时基 0 中断属于多功能中断，时基 1 中断为独立的中断。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 TBnF 被置位时，中断请求发生。

若要跳转到时基 0 中断向量地址，总中断控制位 EMI、时基使能位 TB0E 和相应多功能中断使能位需先被置位。当中断使能，堆栈未满且时基溢出时，可跳转至相关多功能中断向量子程序中执行。当时基中断服务子程序响应时，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，但相应的中断请求标志位 TB0F 需在应用程序中手动清除。

若要跳转到时基 1 中断向量地址，总中断控制位 EMI 和时基使能位 TB1E 需先被置位。当中断使能，堆栈未满且时基溢出时，可跳转至时基 1 中断向量子程序中执行。当时基中断服务子程序响应时，中断请求位 TB1F 会自动复位且 EMI 将被自动清零以除能其它中断。

时基中断时钟源 f_{TB} 来自内部时钟源 f_{TBC} 或 $f_{SYS}/4$。f_{TB} 输入时钟首先经过分频器，分频率由程序设置 TBC 寄存器相关位获取合适的分频值以提供更长的时基中断周期。相应的控制时基中断周期的时钟源可通过 TBC 寄存器中的 TBCK 位选择。

- **TBC 寄存器**

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

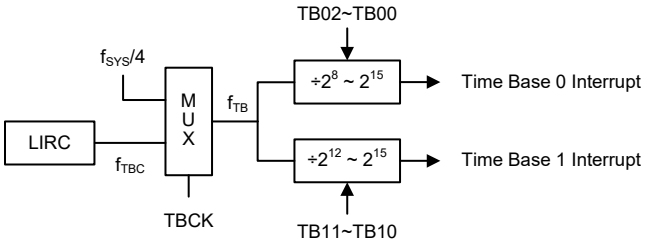
Bit 7 **TBON**：时基 0 和时基 1 控制位
0：除能
1：使能

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

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

Bit 3 未定义，读为“0”

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



时基中断

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 LVD 中断响应，EMI 将被自动清零以除能其它中断，多功能中断请求标志也可自动清除，但 LVF 标志需在应用程序中手动清除。

TM 中断

简易型和周期型 TM 各有两个中断，分别来自比较器 P、A 匹配，都属于多功能中断。每个 TM 各有两个中断请求标志位 xTMnPF、xTMnAF 及两个使能位 xTMnPE、xTMnAE。当 TM 比较器 P、A 匹配情况发生时，任意 TM 中断请求标志被置位，TM 中断请求产生。

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

定时 / 计数器溢出中断

要产生定时 / 计数器中断，总中断使能位 EMI 和定时器中断使能位 TE 需先被置位。当定时 / 计数器溢出，相应的中断请求标志位 TF 将置位并触发定时 / 计数器中断。若中断使能，堆栈未满且定时 / 计数器溢出发生时，可跳转至定时 / 计数器中断向量子程序中执行。当定时 / 计数器中断响应时，中断请求位 TF 会自动复位且 EMI 将被自动清零以除能其它中断。

中断唤醒功能

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

编程注意事项

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

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

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

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

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

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

低电压检测 – LVD

此单片机具有低电压检测功能，即 LVD。该功能使能用于监测电源电压 V_{DD}，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 3 个固定电压中的一个参考点。LVDO 位被置位时低电压情况发生，若 LVDO 位为低表明 V_{DD} 电压工作在当前所设置低电压水平值之上。LVDEN 位用于控制低电压检测功能的开启 / 关闭，设置此位为高使能此功能，反之，关闭内部低电压检测电路。VBGEN 位用于控制内部 Bandgap 参考缓冲电压的使能 / 除能。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

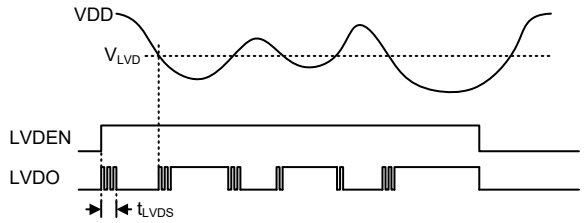
Bit 4　**LVDEN**：低电压检测控制位
0：除能
1：使能

Bit 3　**VBGEN**：内部 Bandgap 参考缓冲电压使能 / 除能控制位
0：除能
1：使能
如果 LVDEN 或 VBGEN 或 LVR 使能，Bandgap 会自动使能。

Bit 2~0　**VLVD2~VLVD0**：选择 LVD 电压位
000~100：未使用
101：3.3V
110：3.6V
111：4.0V

LVD 操作

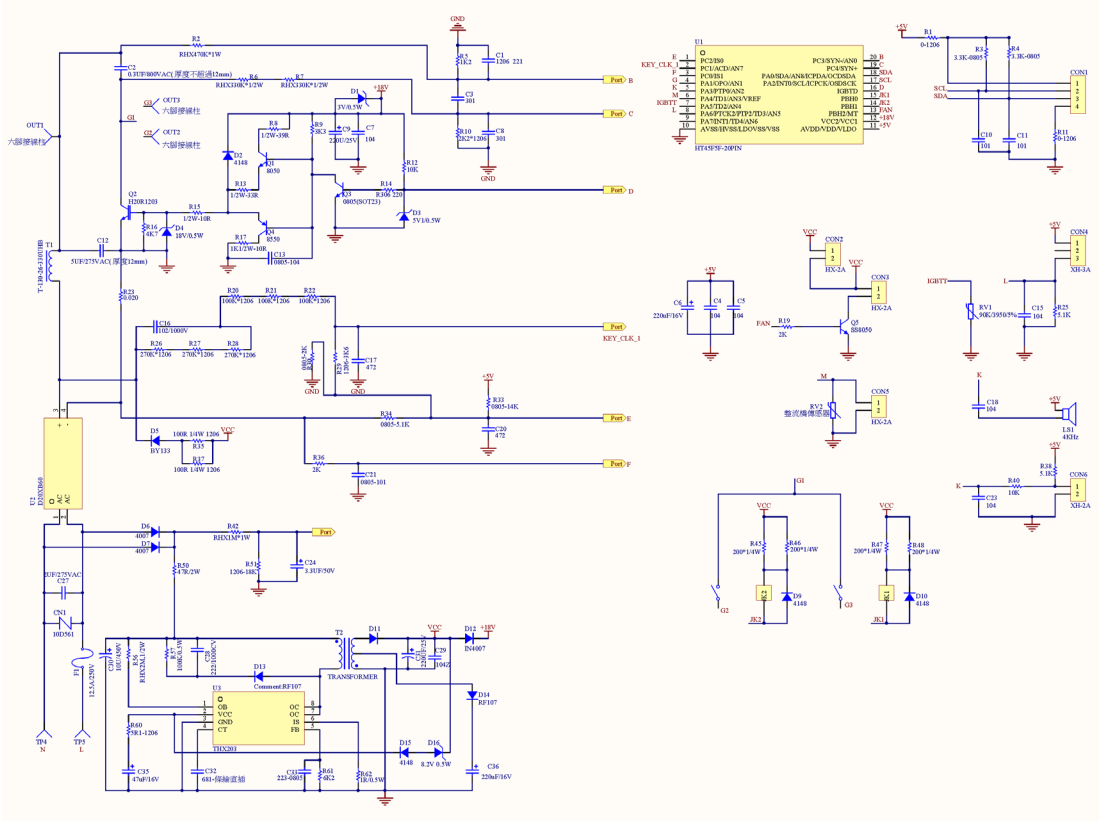
通过比较电源电压 V_{DD} 与存储在 LVDC 寄存器中的预置电压值的结果，低电压检测功能工作。其设置的范围为 3.3V~4.0V。当电源电压 V_{DD} 低于预置电压值时，LVDO 位被置为高，表明低电压产生。低电压检测功能由一个自动使能的参考电压提供。当单片机进入休眠模式时，即使 LVDEN 位被置高，低电压检测器也会被除能。低电压检测器使能后，读取 LVDO 位前，电路稳定需要一定的延时 t_{LVDS}。注意，V_{DD} 电压可能上升或下降比较缓慢，在 V_{LVD} 电压值附近时，LVDO 位可能有多种变化。



LVD 操作

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

应用电路



指令集

简介

任何单片机成功运作的核心在于它的指令集，此指令集为一组程序指令码，用来指导单片机如何去执行指定的工作。在该单片机中，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它们的应用。

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

指令周期

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

数据的传送

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

算术运算

算术运算和数据处理是大部分单片机应用所必需具备的能力，在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 的内容将需要多达 2 个周期来执行。

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

扩展指令集

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

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

助记符		说明	指令周期	影响标志位
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 $[m] \leftarrow ACC - [m] - \overline{C}$

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

20-pin DIP (300mil) 外形尺寸

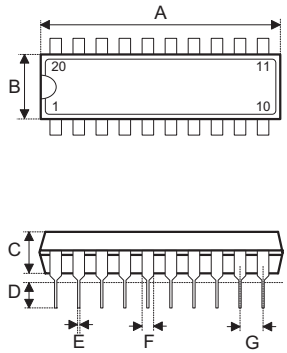


Fig1. Full Lead Packages

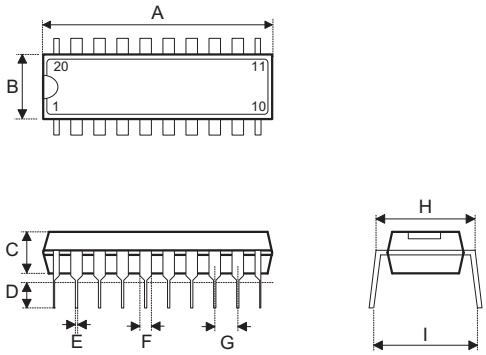


Fig2. 1/2 Lead Packages

见 **fig 1**

符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	0.980	1.030	1.060
B	0.240	0.250	0.280
C	0.115	0.130	0.195
D	0.115	0.130	0.150
E	0.014	0.018	0.022
F	0.045	0.060	0.070
G	—	0.1 BSC	—
H	0.300	0.310	0.325
I	—	—	0.430

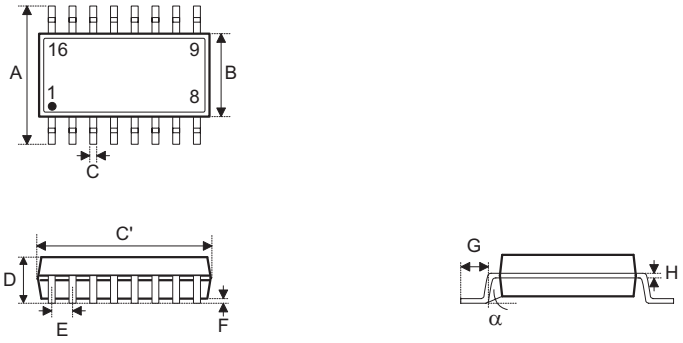
符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	24.89	26.16	26.92
B	6.10	6.35	7.11
C	2.92	3.30	4.95
D	2.92	3.30	3.81
E	0.36	0.46	0.56
F	1.14	1.52	1.78
G	—	2.54 BSC	—
H	7.62	7.87	8.26
I	—	—	10.92

见 fig 2

符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	0.945	0.965	0.985
B	0.275	0.285	0.295
C	0.120	0.135	0.150
D	0.110	0.130	0.150
E	0.014	0.018	0.022
F	0.045	0.050	0.060
G	—	0.1 BSC	—
H	0.300	0.310	0.325
I	—	—	0.430

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	24.00	24.51	25.02
B	6.99	7.24	7.49
C	3.05	3.43	3.81
D	2.79	3.30	3.81
E	0.36	0.46	0.56
F	1.14	1.27	1.52
G	—	2.54 BSC	—
H	7.62	7.87	8.26
I	—	—	10.92

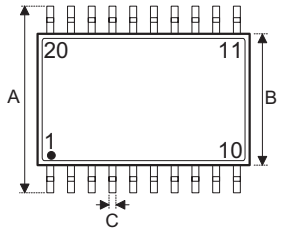
16-pin NSOP (150mil) 外形尺寸

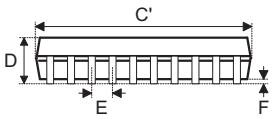


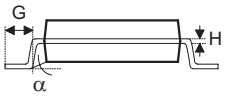
符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C’	—	0.390 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	—	6.0 BSC	—
B	—	3.9 BSC	—
C	0.31	—	0.51
C’	—	9.9 BSC	—
D	—	—	1.75
E	—	1.27 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

20-pin NSOP (150mil) 外形尺寸







符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	0.228	0.236	0.244
B	0.146	0.154	0.161
C	0.009	—	0.012
C’	0.382	0.390	0.398
D	—	—	0.069
E	—	0.032 BSC	—
F	0.002	—	0.009
G	0.020	—	0.031
H	0.008	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	5.80	6.00	6.20
B	3.70	3.90	4.10
C	0.23	—	0.30
C’	9.70	9.90	10.10
D	—	—	1.75
E	—	0.80 BSC	—
F	0.05	—	0.23
G	0.50	—	0.80
H	0.21	—	0.25
α	0°	—	8°

New Wave Electronics (Shenzhen) Ltd.

Room 503, Unit C, Productivity Building, Hi-tech Middle 3nd Road, Shenzhen Hi-Tech Industrial Park, Nanshan District, Shenzhen, China
Tel: 0755-8616-9157
Fax: 0755-8615-5429
Webs: http://www.newwave-sz.com