半桥电磁炉 Flash 单片机

HT45F0075

版本：V1.20　日期：2024-03-15

www.holtek.com

目录

特性

CPU 特性

- 工作电压：
  - ♦ f_{SYS}=16MHz：4.5V~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：
  - ♦ 内部高速 32MHz/16MHz RC – HIRC
  - ♦ 内部低速 32kHz RC – LIRC
- 多种工作模式：快速、低速、空闲、休眠
- 内部集成完整振荡器电路，无需外接元器件
- 所有指令可在 1~3 个指令周期完成
- 查表指令
- 115 条指令
- 8 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器：16K×16
- RAM 数据存储器：1024×8
- True EEPROM 存储器：1024×8
- 在线应用编程功能 – IAP
- 看门狗定时器功能
- 多达 24 个双向 I/O 口
- 2 条与 I/O 口共用引脚的带数字去抖功能的外部中断输入
- PCK 外部引脚用于外围时钟输出
- 多个定时器模块用于时间测量、输入捕捉、比较匹配输出、PWM 输出及单脉冲输出
- 通用串行接口模块 – USIM，用于 SPI、I^2C 或 UART 通信
- 双时基功能，可产生固定时间的中断信号
- 循环冗余校验单元 – CRC
- 乘除法单元 – MDU
- 12-bit PWM 产生器，包含抖频功能
  - ♦ 2 组互补式 PWM 输出
  - ♦ 死区时间可分别设置
  - ♦ 10-bit TMC 用于时间测量和相位信号捕捉测量
  - ♦ 相位侦测及保护机制
  - ♦ PWM 自动调变及自动关闭电路，提供错误信号保护
  - ♦ 硬件和软件抖频功能

- 带内部参考电压 V_{BG} 的 12-bit 分辨率的 A/D 转换器
  - ♦ 11 个外部模拟信号输入通道 – AN0~AN10
  - ♦ 内部模拟信号输入通道可实现对 OPA 输出信号及内部参考电压测量
  - ♦ 支持 1 通道 /2 通道自动转换模式
  - ♦ 多种信号可用来触发开始自动转换，可编程前沿消隐及延迟启动时间
- 9 组过电压保护电路用于实现相位侦测、IGBT 过流保护、IGBT 短路电流保护、AC 过流保护、AC 电压过零检测
- 可编程增益的运算放大器功能
- 低电压复位功能
- 低电压检测功能
- 封装类型：16-pin NSOP，20/24/28-pin SOP

概述

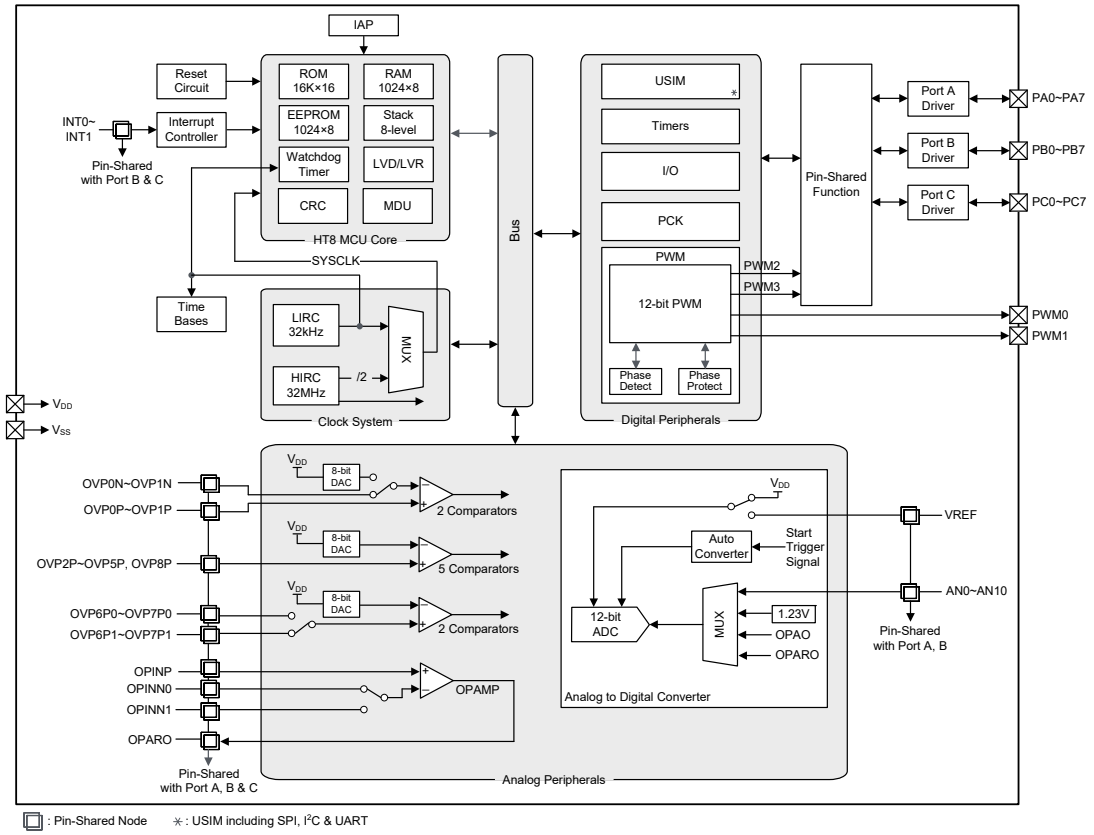
HT45F0075 是一款具有 8 位高性能精简指令集的 Flash 单片机，专门为半桥电磁炉应用而设计。针对半桥电磁炉产品所需要的功率控制，该单片机提供了完善的保护机制。内建硬件过电流、浪涌、相位保护，拥有 12-bit A/D 转换器搭配 2 通道 A/D 自动转换，可测量电磁炉电压、电流等重要参数，除了可实现半桥电磁炉必备功能外，也可节省外围零件，节省产品 PCB 尺寸。

该产品内建 PWM 产生器，输出互补式 PWM 信号，实现工作频率点相位侦测。因操作工作频率影响电流大小，搭配逻辑控制可实现半桥电磁炉功率控制需求。互补式 PWM 控制信号的频率可操作区间为 7.8kHz~16MHz，操作频率以每 1/32MHz 为单位可进行调节，占空比分辨率为 12-bit，插入的死区时间 (12-bit) 可调节范围为 31.25ns~128μs，另外可各别调整互补式控制信号周期时间，非常适合开发半桥电磁炉机种，当操作频率进入谐振电容区时，硬件会依用户设置自动调节 PWM 周期和占空比，使操作频率进入谐振电感区。

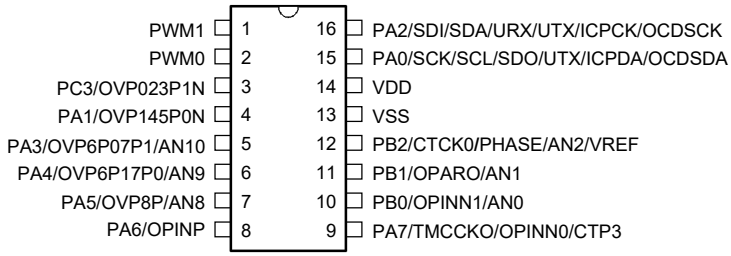
此款单片机除包含上述优点外，内建一组运算放大器来测量谐振电流做功率计算，和九组 OVP 来达到以下功能：两组 OVP 来检测相位；一组 OVP 用做 AC 过电流保护；两组做 IGBT 过电流保护；两组做 IGBT 短路电流保护；另外两组 OVP 分别检测 AC 电压过零 (取电压峰值) 与其它应用电路侦测使用。

该单片机应用在半桥电磁炉，符合市面上所需的主流规格，更详细应用开发信息可参考应用说明章节或查阅 Holtek 网站中的电磁炉应用方案。

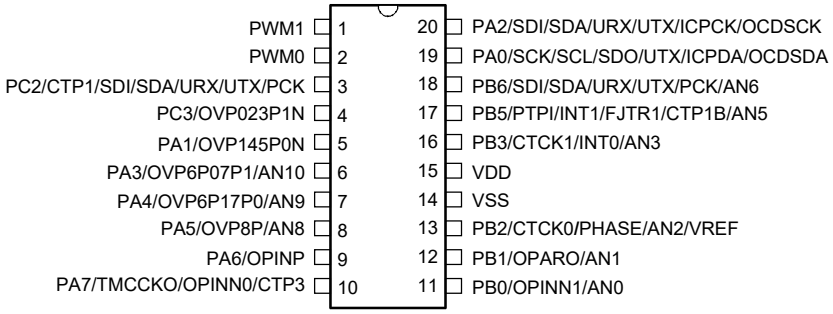
方框图



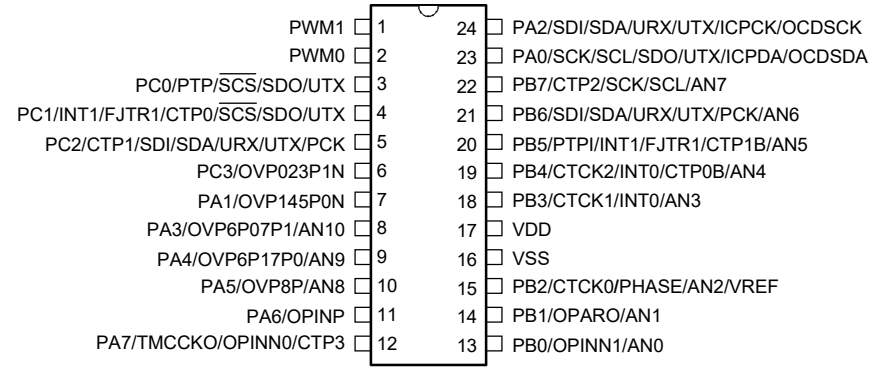
引脚图



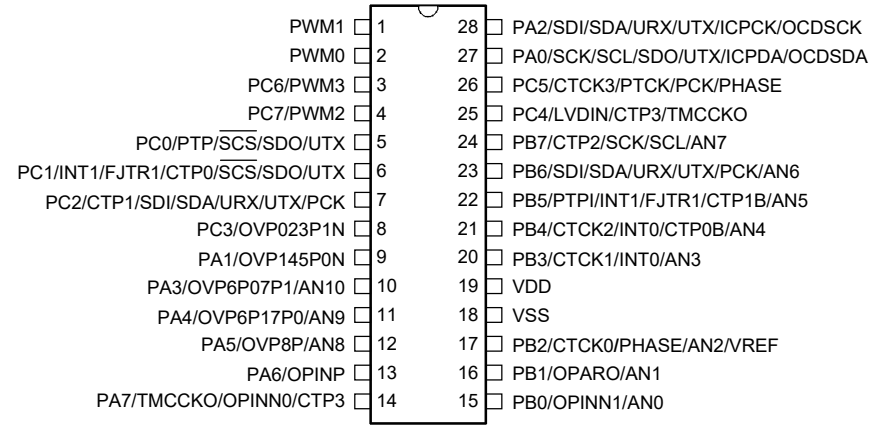
HT45F0075/HT45V0075
16 NSOP-A



HT45F0075/HT45V0075
20 SOP-A



HT45F0075/HT45V0075
24 SOP-A



HT45F0075/HT45V0075
28 SOP-A

注：1. 若共用脚有多种输出，所需引脚功能通过相应引脚共用寄存器中的软件控制位进行选择。
2. 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。
3. 与 PA2 和 PA0 引脚共用的 OCDSCK 和 OCDSDA 引脚功能仅存在于 OCDS EV 芯片 HT45V0075。

引脚说明

每个引脚的功能如下表所述，而引脚配置的详细内容见规格书其它章节。下方表格针对最大封装引脚进行说明，有部分引脚在较小封装类型中未出现。

引脚名称	功能	OPT	I/T	O/T	描述
PA0/SCK/SCL/SDO/UTX/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
	SCK	PAS0 IFS	ST	CMOS	SPI 时钟线
	SCL	PAS0 IFS	ST	NMOS	I^2C 时钟线
	SDO	PAS0	—	CMOS	SPI 串行数据输出
	UTX	PAS0	—	CMOS	UART 串行数据输出
	ICPDA	—	ST	CMOS	ICP 数据 / 地址引脚
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址引脚，仅用于 EV 芯片
PA1/OVP145P0N	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
	OVP145P0N	PAS0	AN	—	OVP1/OVP4/OVP5 正端输入和 OVP0 负端输入引脚
PA2/SDI/SDA/URX/UTX/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
	SDI	PAS0 IFS	ST	—	SPI 串行数据输入
	SDA	PAS0 IFS	ST	NMOS	I^2C 数据线
	URX/UTX	PAS0 IFS	ST	CMOS	UART 串行数据输入 (全双工通信), UART 串行数据输入 / 输出 (单线通信模式)
	ICPCK	—	ST	—	ICP 时钟引脚
	OCDSCK	—	ST	—	OCDS 时钟引脚，仅用于 EV 芯片
PA3/OVP6P07P1/AN10	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
	OVP6P07P1	PAS0	AN	—	OVP6/OVP7 正端输入引脚
	AN10	PAS0	AN	—	A/D 转换器外部输入通道 10
PA4/OVP6P17P0/AN9	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
	OVP6P17P0	PAS1	AN	—	OVP6/OVP7 正端输入引脚
	AN9	PAS1	AN	—	A/D 转换器外部输入通道 9

引脚名称	功能	OPT	I/T	O/T	描述
PA5/OVP8P/AN8	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
	OVP8P	PAS1	AN	—	OVP8 正端输入引脚
	AN8	PAS1	AN	—	A/D 转换器外部输入通道 8
PA6/OPINP	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
	OPINP	PAS1	AN	—	OPA 同相输入引脚
PA7/TMCCKO/ OPINN0/CTP3	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
	TMCCKO	PAS1	—	CMOS	10-bit TMC TMCCKO 时钟信号输出
	OPINN0	PAS1	AN	—	OPA 反相输入引脚 0
	CTP3	PAS1	—	CMOS	CTM3 输出
PB0/OPINN1/AN0	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OPINN1	PBS0	AN	—	OPA 反相输入引脚 1
	AN0	PBS0	AN	—	A/D 转换器外部输入通道 0
PB1/OPARO/AN1	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OPARO	PBS0	—	AN	OPA 输出引脚
	AN1	PBS0	AN	—	A/D 转换器外部输入通道 1
PB2/CTCK0/ PHASE/AN2/VREF	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	CTCK0	PBS0	ST	—	CTM0 时钟输入
	PHASE	PBS0	—	CMOS	PHASE 信号 (相位侦测电路) 输出
	AN2	PBS0	AN	—	A/D 转换器外部输入通道 2
	VREF	PBS0	AN	—	A/D 转换器外部参考电压输入
PB3/CTCK1/INT0/ AN3	PB3	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	CTCK1	PBS0	ST	—	CTM1 时钟输入
	INT0	PBS0 IFS	ST	—	外部中断 0
	AN3	PBS0	AN	—	A/D 转换器外部输入通道 3
PB4/CTCK2/INT0/ CTP0B/AN4	PB4	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	CTCK2	PBS1	ST	—	CTM2 时钟输入
	INT0	PBS1 IFS	ST	—	外部中断 0
	CTP0B	PBS1	—	CMOS	CTM0 反相输出
	AN4	PBS1	AN	—	A/D 转换器外部输入通道 4

引脚名称	功能	OPT	I/T	O/T	描述
PB5/PTPI/INT1/FJTR1/CTP1B/AN5	PB5	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTPI	PBS1	ST	—	PTM 捕捉输入
	INT1	PBS1 IFS	ST	—	外部中断 1
	FJTR1	PBS1 IFS	ST	—	FJTIMER 输入 1
	CTP1B	PBS1	—	CMOS	CTM1 反相输出
	AN5	PBS1	AN	—	A/D 转换器外部输入通道 5
PB6/SDI/SDA/URX/UTX/PCK/AN6	PB6	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SDI	PBS1 IFS	ST	—	SPI 串行数据输入
	SDA	PBS1 IFS	ST	NMOS	I^2C 数据线
	URX/UTX	PBS1 IFS	ST	CMOS	UART 串行数据输入 (全双工通信), UART 串行数据输入 / 输出 (单线通信模式)
	PCK	PBS1	—	CMOS	外围时钟输出
	AN6	PBS1	AN	—	A/D 转换器外部输入通道 6
PB7/CTP2/SCK/SCL/AN7	PB7	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	CTP2	PBS1	—	CMOS	CTM2 输出
	SCK	PBS1 IFS	ST	CMOS	SPI 时钟线
	SCL	PBS1 IFS	ST	NMOS	I^2C 时钟线
	AN7	PBS1	AN	—	A/D 转换器外部输入通道 7
PC0/PTP/$\overline{SCS}$/SDO/UTX	PC0	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTP	PCS0	—	CMOS	PTM 输出
	$\overline{SCS}$	PCS0 IFS	ST	CMOS	SPI 从机选择
	SDO	PCS0	—	CMOS	SPI 串行数据输出
	UTX	PCS0	—	CMOS	UART 串行数据输出

引脚名称	功能	OPT	I/T	O/T	描述
PC1/INT1/FJTR1/CTP0/$\overline{SCS}$/SDO/UTX	PC1	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	INT1	PCS0 IFS	ST	—	外部中断 1
	FJTR1	PCS0 IFS	ST	—	FJTIMER 输入 1
	CTP0	PCS0	—	CMOS	CTM0 输出
	$\overline{SCS}$	PCS0 IFS	ST	CMOS	SPI 从机选择
	SDO	PCS0	—	CMOS	SPI 串行数据输出
	UTX	PCS0	—	CMOS	UART 串行数据输出
PC2/CTP1/SDI/SDA/URX/UTX/PCK	PC2	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	CTP1	PCS0	—	CMOS	CTM1 输出
	SDI	PCS0 IFS	ST	—	SPI 串行数据输入
	SDA	PCS0 IFS	ST	NMOS	I^2C 数据线
	URX/UTX	PCS0 IFS	ST	CMOS	UART 串行数据输入 (全双工通信), UART 串行数据输入 / 输出 (单线通信模式)
	PCK	PCS0	—	CMOS	外围时钟输出
PC3/OVP023P1N	PC3	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OVP023P1N	PCS0	AN	—	OVP0/OVP2/OVP3 正端输入和 OVP1 负端输入引脚
PC4/LVDIN/CTP3/TMCCKO	PC4	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	LVDIN	PCS1	AN	—	LVD 监测的外部电压输入
	CTP3	PCS1	—	CMOS	CTM3 输出
	TMCCKO	PCS1	—	CMOS	10-bit TMC TMCCKO 时钟信号输出
PC5/CTCK3/PTCK/PCK/PHASE	PC5	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	CTCK3	PCS1	ST	—	CTM3 时钟输入
	PTCK	PCS1	ST	—	PTM 时钟输入
	PCK	PCS1	—	CMOS	外围时钟输出
	PHASE	PCS1	—	CMOS	PHASE 信号 (相位侦测电路) 输出
PC6/PWM3	PC6	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PWM3	PCS1	—	CMOS	PWM3 输出
PC7/PWM2	PC7	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PWM2	PCS1	—	CMOS	PWM2 输出
PWM0	PWM0	—	—	CMOS	PWM0 输出

引脚名称	功能	OPT	I/T	O/T	描述
PWM1	PWM1	—	—	CMOS	PWM1 输出
VDD	VDD	—	PWR	—	正电源电压
VSS	VSS	—	PWR	—	负电源电压

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；
AN：模拟信号；
O/T：输出类型；
PWR：电源；
CMOS：CMOS 输出；
NMOS：NMOS 输出；

极限参数

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

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

直流电气特性

以下表格中参数测量结果可能受多个因素影响，如振荡器类型、工作电压、工作频率、引脚负载状况、温度和程序指令等。

工作电压特性

Ta=25°C

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 (HIRC)	f_{SYS}=f_{HIRC}=16MHz	4.5	—	5.5	V
	工作电压 (LIRC)	f_{SYS}=f_{LIRC}=32kHz	4.5	—	5.5	V

工作电流特性

Ta=25°C

符号	正常工作	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{DD}	低速模式 (LIRC)	5V	f_{SYS}=f_{LIRC}=32kHz	—	30	50	μA
	快速模式 (HIRC)	5V	f_{SYS}=f_{HIRC}=16MHz	—	2.5	5.0	mA

注：当使用该表格电气特性数据时，以下几点需注意：
1. 任何数字输入都设置为非浮空的状态。
2. 所有测量都在无负载且所有外围功能关闭的条件下进行。
3. 无直流电流路径。
4. 所有工作电流数值是在执行连续的 NOP 指令循环程序下测得。

待机电流特性

Ta=25°C

符号	待机模式	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{STB}	休眠模式	5V	WDT on	—	3	5	μA
	空闲模式 0 (LIRC)	5V	f_{SUB} on	—	5	10	μA
	空闲模式 1 (HIRC)	5V	f_{SUB} on, f_{SYS}=16MHz	—	1.4	2.0	mA

注：当使用该表格电气特性数据时，以下几点需注意：
1. 任何数字输入都设置为非浮空的状态。
2. 所有测量都在无负载且所有外围功能关闭的条件下进行。
3. 无直流电流路径。
4. 所有待机电流数值都是在 HALT 指令执行后即停止执行所有指令后测得。

交流电气特性

以下表格中参数测量结果可能受多个因素影响，如振荡器类型、工作电压、工作频率和温度等等。

内部高速 RC 振荡器 HIRC 频率精准度

程序烧录时，烧录器依据用户选择的 HIRC 频率和工作电压 (5V) 对 HIRC 进行频率精准度调整。

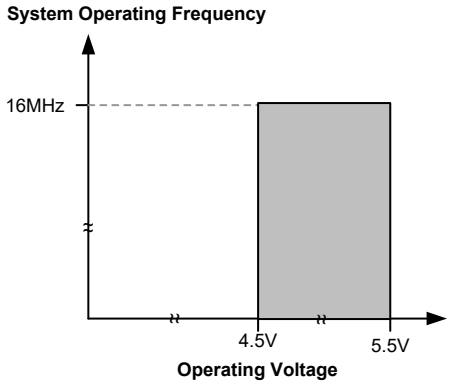
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{HIRC}	通过烧录器调整后的 16MHz HIRC 频率	5V	25°C	-1%	16	+1%	MHz
			-40°C~85°C	-2%	16	+2%	
		4.5V~5.5V	25°C	-2.5%	16	+2.5%	
			-40°C~85°C	-3%	16	+3%	
f_{PWM}	通过烧录器调整后的 32MHz HIRC 频率	5V	25°C	-1%	32	+1%	MHz
			-40°C~85°C	-2%	32	+2%	
		4.5V~5.5V	25°C	-2.5%	32	+2.5%	
			-40°C~85°C	-3%	32	+3%	

注：1. 选择烧录器在 5V 电压下调整 HIRC 频率，在此提供 V_{DD} 固定为 5V 时的 HIRC 精准度值。
2. 表格中也提供 V_{DD} 电压在某一范围内时的 HIRC 精准度值。

内部低速振荡器 LIRC 电气特性

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{LIRC}	LIRC 振荡器频率	5V	25°C	-2%	32	+2%	kHz
		4.5V~5.5V	-40°C~85°C	-7%	32	+7%	

工作频率特性曲线



系统上电时间特性

Ta=25°C

符号	参数	测试条件	最小	典型	最大	单位
t_{SST}	系统启动时间 (从 f_{SYS} off 的状态下唤醒)	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{SYS}
		$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	系统启动时间 (从 f_{SYS} on 的状态下唤醒)	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_{SYS}
		$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	系统速度切换时间 (快速模式 → 低速模式或 低速模式 → 快速模式)	f_{HIRC} off → on	—	16	—	t_{HIRC}
t_{RSTD}	系统复位延迟时间 (上电复位或 LVR 硬件复位)	RR_{POR}=5V/ms	14	16	18	ms
	系统复位延迟时间 (LVRC/WDTC 软件复位)	—				
	系统复位延迟时间 (WDT 溢出复位)	—	14	16	18	ms
t_{SRESET}	最小软件复位时间	—	45	90	120	μs

注：1. 系统启动时间里提到的 f_{SYS} on/off 状态取决于工作模式类型以及所选的系统时钟振荡器。更多相关细节请参考系统工作模式章节。

2. t_{HIRC} 等符号所表示的时间单位，是对应频率值的倒数，相关频率值在前面表格有说明。例如，$t_{HIRC}=1/f_{HIRC}$，$t_{SYS}=1/f_{SYS}$ 等等。

3. 系统速度切换时间实际上是指新使能的振荡器的启动时间。

输入 / 输出口电气特性

Ta=25°C，除非另有说明

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{IL}	I/O 口或输入引脚低电平输入电压	5V	—	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	
V_{IH}	I/O 口或输入引脚高电平输入电压	5V	—	3.5	—	5.0	V
		—		$0.8V_{DD}$	—	V_{DD}	
I_{OL}	I/O 口灌电流	5V	$V_{OL}=0.1V_{DD}$	32	65	—	mA
	PWM0/PWM1 引脚灌电流	5V		32	65	—	
I_{OH}	I/O 口源电流	5V	$V_{OH}=0.9V_{DD}$	-8	-16	—	mA
	PWM0/PWM1 引脚源电流	5V		-8	-16	—	
R_{PH}	I/O 口上拉电阻 (1)	5V	Ta=-40°C~85°C	10	30	50	kΩ
I_{LEAK}	输入漏电流	5V	$V_{IN}=V_{DD}$ 或 $V_{IN}=V_{SS}$	—	—	±1	μA
t_{INT}	外部中断最小输入脉宽	—	—	0.3	—	—	μs
t_{EIRT}	外部中断去抖时间	—	INTnDB[1:0]=00B (n=0~1)	—	0	—	t_H
			INTnDB[1:0]=01B (n=0~1)	7	—	8	
			INTnDB[1:0]=10B (n=0~1)	15	—	16	
			INTnDB[1:0]=11B (n=0~1)	23	—	24	
t_{FJT}	FJTR1 最小输入脉宽	—	—	0.3	—	—	μs
t_{TCK}	CTCKn/PTCK 引脚输入最小脉宽	—	—	0.3	—	—	μs
t_{TPI}	PTM PTPI 引脚输入最小脉宽	—	—	0.3	—	—	μs
f_{TMCLK}	PTM 定时器时钟源最大频率	5V	—	—	—	1	f_{SYS}
t_{CPW}	PTM 最小捕捉脉宽	—	—	t_{CPW}(2)	—	—	μs

注：1. R_{PH} 内部上拉电阻值的计算方法是：将引脚接地并设置为输入且使能上拉电阻功能，然后在特定电源电压下测量该引脚上的电流，最后电压除以测量的电流值从而得到此上拉电阻值。

2. 若 PTCAPTS=0，则 $t_{CPW}=max(2\times t_{TMCLK}, t_{TPI})$

若 PTCAPTS=1，则 $t_{CPW}=max(2\times t_{TMCLK}, t_{TCK})$

$t_{TMCLK}=1/f_{TMCLK}$

例 1：若 PTCAPTS=0，f_{TMCLK}=16MHz，t_{TPI}=0.3μs，则 $t_{CPW}=max(0.125\mu s, 0.3\mu s)=0.3\mu s$

例 2：若 PTCAPTS=1，f_{TMCLK}=16MHz，t_{TCK}=0.3μs，则 $t_{CPW}=max(0.125\mu s, 0.3\mu s)=0.3\mu s$

例 3：若 PTCAPTS=0，f_{TMCLK}=8MHz，t_{TPI}=0.3μs，则 $t_{CPW}=max(0.25\mu s, 0.3\mu s)=0.3\mu s$

例 4：若 PTCAPTS=0，f_{TMCLK}=4MHz，t_{TPI}=0.3μs，则 $t_{CPW}=max(0.5\mu s, 0.3\mu s)=0.5\mu s$

存储器电气特性

Ta=-40°C~85°C，除非另有说明

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
Flash 程序存储器							
t_{FER}	IAP 擦除时间	—	FWERTS=0	—	3.2	3.9	ms
		—	FWERTS=1	—	3.7	4.5	ms
t_{FWR}	IAP 写时间	—	FWERTS=0	—	2.2	2.7	ms
		—	FWERTS=1	—	3.0	3.6	ms
E_P	存储单元耐受性	—	—	100K	—	—	E/W(2)
t_{RETD}	ROM 数据保存时间	—	Ta=25°C	—	40	—	Year
t_{ACTV}	ROM 激活时间 – 从暂停模式唤醒 (1)	—	—	32	—	64	μs
数据 EEPROM 存储器							
t_{EEER}	擦除时间	—	EWERTS=0	—	3.2	3.9	ms
		—	EWERTS=1	—	3.7	4.5	ms
t_{EEWR}	写时间 (字节模式)	—	EWERTS=0	—	5.4	6.6	ms
		—	EWERTS=1	—	6.7	8.1	ms
	写时间 (页模式)	—	EWERTS=0	—	2.2	2.7	ms
		—	EWERTS=1	—	3.0	3.6	ms
E_P	存储单元耐受性	—	—	100K	—	—	E/W(2)
t_{RETD}	ROM 数据保存时间	—	Ta=25°C	—	40	—	Year
RAM 数据存储器							
V_{DR}	RAM 数据保存电压	—	—	1.0	—	—	V

注：1. 在计算从暂停模式唤醒的系统总启动时间时，还需加上 ROM 激活时间 t_{ACTV}。
2. “E/W”表示擦 / 写次数。

LVD/LVR 电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{LVR}	低电压复位电压	—	LVR 使能，电压选择 2.1V	-5%	2.1	+5%	V
			LVR 使能，电压选择 2.55V		2.55		
			LVR 使能，电压选择 3.15V		3.15		
			LVR 使能，电压选择 3.8		3.8		
V_{LVD}	低电压检测电压	—	LVD 使能，电压选择 LVDIN 引脚 = 1.23V	-10%	1.23	+10%	V
			LVD 使能，电压选择 2.2V	-5%	2.2	+5%	
			LVD 使能，电压选择 2.4V	-5%	2.4	+5%	
			LVD 使能，电压选择 2.7V	-5%	2.7	+5%	
			LVD 使能，电压选择 3.0V	-5%	3.0	+5%	
			LVD 使能，电压选择 3.3V	-5%	3.3	+5%	
			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
			LVD 使能，LVR 使能 VBGEN=1	—	180	200	
t_{LVDS}	LVDO 稳定时间	—	LVR 使能，VBGEN=0 LVD off → on	—	—	18	μs
t_{LVR}	产生 LVR 复位的低电压最短保持时间	—	TLVR[1:0]=00B	120	240	480	μs
			TLVR[1:0]=01B	0.5	1.0	2.0	ms
			TLVR[1:0]=10B	1	2	4	
			TLVR[1:0]=11B	2	4	8	
t_{LVD}	产生 LVD 中断的低电压最短保持时间	—	—	60	120	240	μs

注：当选择 V_{LVD}=1.23V 时，用于监测 LVDIN 引脚输入电压是否低于 1.23V。其它 V_{LVD} 选项用于监测电源电压 V_{DD}。

内部参考电压电气特性

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%	1.23	+5%	V

注：V_{BG} 电压可作为 A/D 转换器内部信号输入。

A/D 转换器电气特性

Ta=25°C

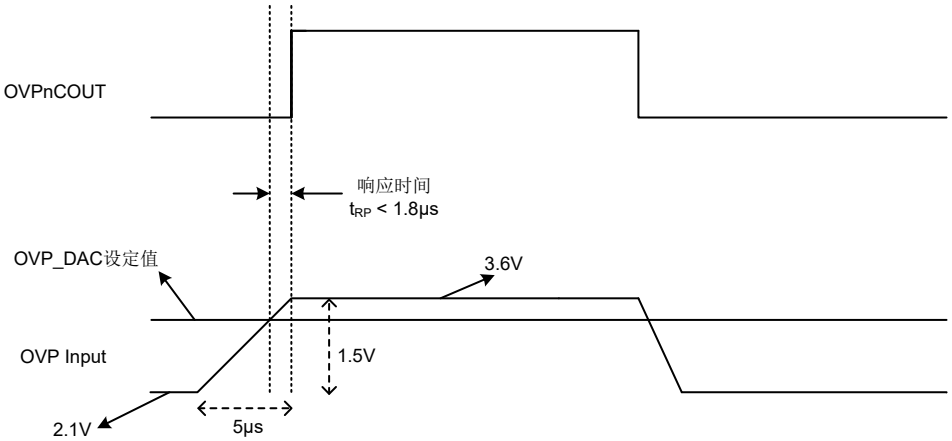
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{ADI}	A/D 转换器输入电压	—	—	0	—	V_{REF}	V
V_{REF}	A/D 转换参考电压	—	—	2	—	V_{DD}	V
N_R	分辨率	—	—	—	—	12	Bit
DNL	非线性微分误差	5V	V_{REF}=V_{DD}，t_{ADCK}=0.5μs	-3	—	+3	LSB
			V_{REF}=V_{DD}，t_{ADCK}=10μs				
INL	非线性积分误差	5V	V_{REF}=V_{DD}，t_{ADCK}=0.5μs	-4	—	+4	LSB
			V_{REF}=V_{DD}，t_{ADCK}=10μs				
I_{ADC}	A/D 转换器使能的额外电流	5V	无负载，t_{ADCK}=0.5μs	—	850	1000	μA
t_{ADCK}	A/D 转换器时钟周期	—	4.5V≤V_{DD}≤5.5V	0.5	—	10.0	μs
t_{ON2ST}	A/D 转换器 On-to-Start 时间	—	—	4	—	—	μs
t_{ADS}	A/D 转换器采样时间	—	—	—	4	—	t_{ADCK}
t_{ADC}	单次 A/D 转换时间 (包括采样和保持时间)	—	—	—	16	—	t_{ADCK}

过电压保护电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{OVP}	OVP 工作电流	5V	OVPnEN=1 (n=2~8)	—	500	750	μA
			OVPnEN=1，DACnEN=0 (n=0, 1)	—	100	140	μA
			OVPnEN=1，DACnEN=1 (n=0, 1)	—	500	750	μA
V_{OS}	输入失调电压	5V	已校准	-2	—	2	mV
V_{HYS}	迟滞	5V	HYSn[1:0]=00B	0	0	5	mV
			HYSn[1:0]=01B	15	30	50	
			HYSn[1:0]=10B	30	60	90	
			HYSn[1:0]=11B	40	90	130	
V_{CM}	共模电压范围	5V	—	V_{SS}	—	V_{DD}-1	V
R_O	R2R 输出电阻值	5V	—	—	10	—	kΩ
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
t_{RP}	OVP 响应时间	5V	OVPnDA=10110011b OVPnDEB[2:0]=000 DAC V_{REF}=V_{DD} OVP Input=2.1V~3.6V（注）	—	1.0	1.8	μs
		5V	60mV 过驱动电压	—	—	1	

注：OVP 响应时间 (t_{RP}) 验证波形 (5V)：



运算放大器电气特性

Ta=25°C，除非另有指定

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{OPA}	运算放大器使能的额外电流	5V	无负载	—	300	600	μA
V_{OS}	输入失调电压	5V	未校准 (OPOOF [5:0]=100000B)	-15	—	15	mV
			已校准	-2	—	2	
V_{CM}	共模电压范围	5V	—	V_{SS}	—	V_{DD}-1.4	V
V_{OR}	最大输出电压范围	5V	—	V_{SS}+0.1	—	V_{DD}-0.1	V
R_{OPAR1}	OPAR1 电阻值	5V	—	7.5	10	12.5	kΩ
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	—	2200	—	kHz
PSRR	电源电压抑制比	5V	—	60	80	—	dB
CMRR	共模抑制比	5V	—	60	80	—	dB
Ga	PGA 增益精准度（注）	5V	相对增益, Ta=-40°C~85°C	-5	—	5	%

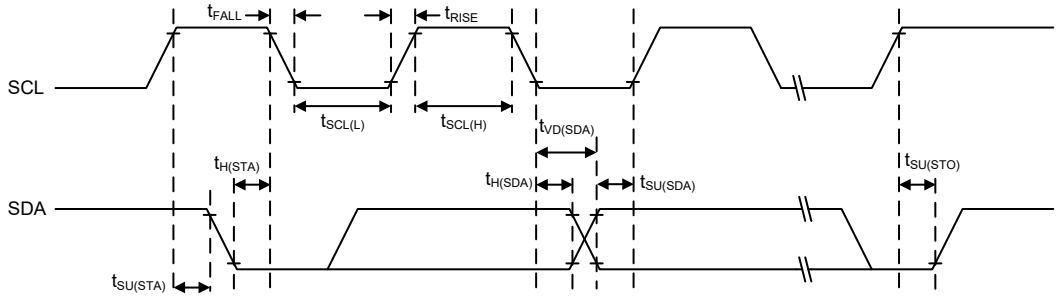
注：使 PGA 输出电压满足 V_{OR} 规格要求，可保证此处的 PGA 增益精准度。

I²C 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_{I2C}	I²C 标准模式 (100kHz) 时的 f_{SYS} 频率 (注)	—	无去抖时间	2	—	—	MHz
			2 个系统时钟时间去抖	4	—	—	
			4 个系统时钟时间去抖	4	—	—	
	I²C 快速模式 (400kHz) 时的 f_{SYS} 频率 (注)	—	无去抖时间	4	—	—	MHz
			2 个系统时钟时间去抖	8	—	—	
			4 个系统时钟时间去抖	8	—	—	
f_{SCL}	SCL 时钟频率	5V	标准模式	—	—	100	kHz
			快速模式	—	—	400	
$t_{SCL(H)}$	SCL 时钟高电平时间	5V	标准模式	3.5	—	—	μs
			快速模式	0.9	—	—	
$t_{SCL(L)}$	SCL 时钟低电平时间	5V	标准模式	3.5	—	—	μs
			快速模式	0.9	—	—	
t_{FALL}	SCL 和 SDA 下降沿时间	5V	标准模式	—	—	1.3	μs
			快速模式	—	—	0.34	
t_{RISE}	SCL 和 SDA 上升沿时间	5V	标准模式	—	—	1.3	μs
			快速模式	—	—	0.34	
$t_{SU(SDA)}$	SDA 数据建立时间	5V	标准模式	0.25	—	—	μs
			快速模式	0.1	—	—	
$t_{H(SDA)}$	SDA 数据保持时间	5V	—	0.1	—	—	μs
$t_{VD(SDA)}$	SDA 数据有效时间	5V	—	—	—	0.6	μs
$t_{SU(STA)}$	START 条件建立时间	5V	标准模式	3.5	—	—	μs
			快速模式	0.6	—	—	
$t_{H(STA)}$	START 条件保持时间	5V	标准模式	4.0	—	—	μs
			快速模式	0.6	—	—	
$t_{SU(STO)}$	STOP 条件建立时间	5V	标准模式	3.5	—	—	μs
			快速模式	0.6	—	—	

注：使用去抖功能可使传输更稳定，降低受干扰影响通信失败的机率。

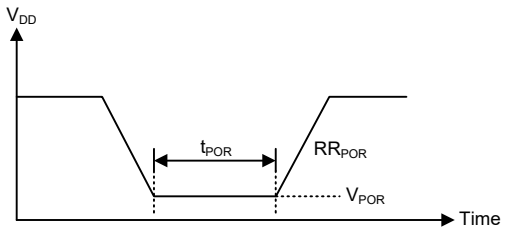


I²C 时序图

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



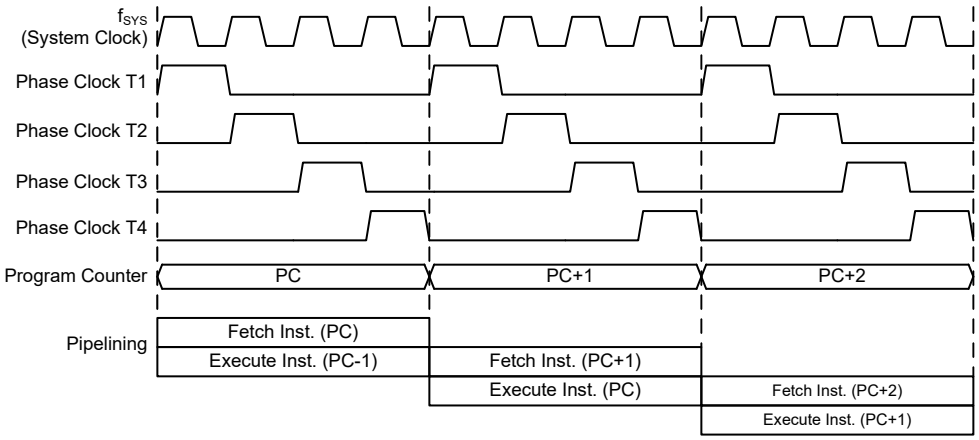
系统结构

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

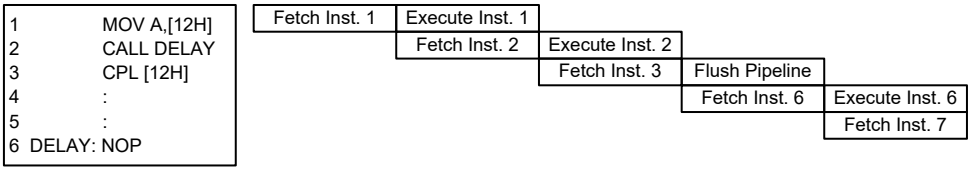
时序和流水线结构

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。

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



系统时序和流水线



指令捕捉

程序计数器

在程序执行期间，程序计数器用来指向下一个要执行的指令地址。除了“JMP”和“CALL”指令需要跳转到一个非连续的程序存储器地址之外，它会在每条指令执行完成以后自动加一。对于存储器容量大于 8K 字的单片机，其程序存储器地址可能位于某一程序存储区中，可通过程序存储区指针的 PBP0 位来选择。只有较低的 8 位，即所谓的程序计数器低字节寄存器 PCL，可以被用户直接读写。

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

程序计数器	
高字节	低字节 (PCL)
PBP0, PC12~PC8	PCL7~PCL0

程序计数器

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

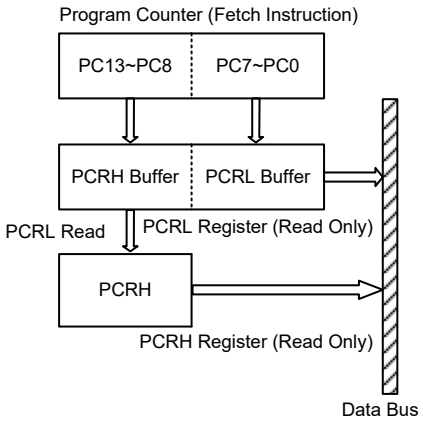
程序计数器读寄存器

程序计数器读寄存器为只读寄存器，用于读取目前程序执行地址的计数器值。先读低字节寄存器再读高字节寄存器，即读取目前程序执行地址的低字节，同

时将程序计数器的高字节数据放置在 8-bit PCRH 缓存器。然后读取 PCRH 寄存器，从 8-bit PCRH 缓存器中读取数据。

下面举例说明如何读取目前程序执行地址。当目前程序执行地址为 123H 时，执行指令步骤如下：

1. 执行 MOV A, PCRL 指令之后 → ACC 值为 23H，并且 PCRH 值为 01H；
   执行 MOV A, PCRH 指令之后 → ACC 值为 01H。
2. 执行 LMOV A, PCRL 指令之后 → ACC 值为 23H，并且 PCRH 值为 01H；
   执行 LMOV A, PCRH 指令之后 → ACC 值为 01H。



- **PCRL 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**：程序计数器读低字节寄存器 bit 7 ~ bit 0

- **PCRH 寄存器**

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

Bit 7~6 未定义，读为“0”

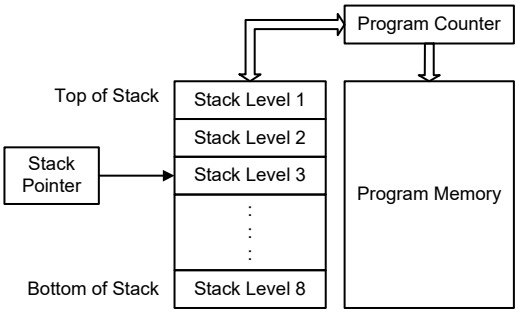
Bit 5~0 **D13~D8**：程序计数器读高字节寄存器 bit 5 ~ bit 0

堆栈

堆栈是一个特殊的存储空间，用来存储程序计数器中的内容。该单片机有 8 层堆栈，堆栈既不是数据部分也不是程序空间部分，而且它既不是可读取也不是可写入的。当前层由堆栈指针 (STKPTR) 加以指示，它是一个只读寄存器。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。



● **STKPTR 寄存器**

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

Bit 7 **OSF**：堆栈溢出标志位
0：未发生堆栈溢出
1：发生堆栈溢出
当堆栈已满，再次执行 CALL 指令或当堆栈为空，再次执行 RET 指令，OSF 位会被置为 1。该位只能通过软件清零，硬件不会自动复位。

Bit 6~3 未定义，读为“0”

Bit 2~0 **D2~D0**：堆栈指针寄存器 bit 2 ~ bit 0

下面举例说明连续执行 CALL 指令或连续执行 RET 指令，相关位是如何变化的。

1. 连续执行 9 次 CALL 指令，期间未执行 RET 指令，STKPTR[2:0] 及 OSF 位的变化如下：

CALL 执行次数	0	1	2	3	4	5	6	7	8	9
STKPTR[2:0] 值	0	1	2	3	4	5	6	7	0	1
OSF 位	0	0	0	0	0	0	0	0	0	1

2. 当 OSF 为 1 时，若不清除 OSF 位，则不管执行几次 RET 指令，OSF 位会一直为 1。
3. 当堆栈为空时，连续执行 8 次 RET 指令，STKPTR[2:0] 及 OSF 位的变化如下：

RET 执行次数	0	1	2	3	4	5	6	7	8
STKPTR[2:0] 值	0	7	6	5	4	3	2	1	0
OSF 位	0	1	1	1	1	1	1	1	1

算术逻辑单元 – ALU

算术逻辑单元是单片机中很重要的部分，执行指令集中的算术和逻辑运算。ALU 连接到单片机的数据总线，在接收相关的指令码后执行需要的算术与逻辑操作，并将结果存储在指定的寄存器，当 ALU 计算或操作时，可能导致进位、

借位或其它状态的改变，而相关的状态寄存器会因此更新内容以显示这些改变，ALU 所提供的功能如下：

- 算术运算

  ADD，ADDM，ADC，ADCM，SUB，SUBM，SBC，SBCM，DAA，
  LADD，LADDM，LADC，LADCM，LSUB，LSUBM，LSBC，LSBCM，
  LDAA
- 逻辑运算

  AND，OR，XOR，ANDM，ORM，XORM，CPL，CPLA，
  LAND，LANDM，LOR，LORM，LXOR，LXORM，LCPL，LCPLA
- 移位运算

  RRA，RR，RRCA，RRC，RLA，RL，RLCA，RLC，
  LRR，LRRA，LRRCA，LRRC，LRLA，LRL，LRLCA，LRLC
- 递增和递减

  INCA，INC，DECA，DEC，
  LINCA，LINC，LDECA，LDEC
- 分支判断

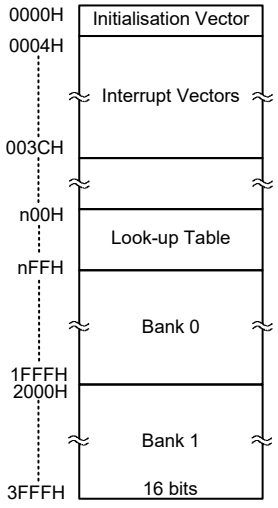
  JMP，SZ，SZA，SNZ，SIZ，SDZ，SIZA，SDZA，CALL，RET，RETI，
  LSNZ，LSZ，LSZA，LSIZ，LSIZA，LSDZ，LSDZA

Flash 程序存储器

程序存储器用来存放用户代码即储存程序。程序存储器为 Flash 类型意味着可以多次重复编程，方便用户使用同一芯片进行程序的修改。使用适当的单片机编程工具，此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 16K×16 位，程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口。数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。



程序存储器结构

特殊向量

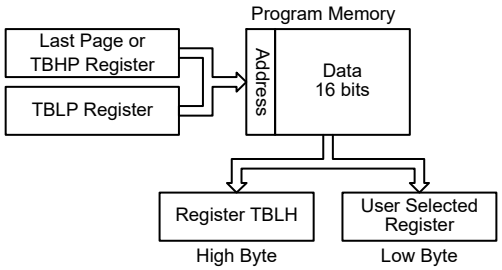
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 0000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

查表

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

在设定完表格指针后，当数据存储器 [m] 位于 Sector 0，表格数据可以使用如“TABRD [m]”或“TABRDL [m]”等指令分别从程序存储器查表读取。如果存储器 [m] 位于其它 Sector，表格数据可以使用如“LTABRD [m]”或“LTABRDL [m]”等扩展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，而高字节中未使用的位将被读取为“0”。

下图是查表中寻址 / 数据流程：



查表范例

以下范例说明表格指针和表格数据如何被定义和执行。这个例子使用的表格数据用 ORG 伪指令储存在存储器中。ORG 指令的值“1F00H”位于 Bank1，指向的地址是 16K 程序存储器中最后一页的起始地址。表格指针的初始值设为“06H”，这可保证从数据表格读取的第一笔数据位于程序存储器地址“3F06H”，即最后一页起始地址后的第六个地址。值得注意的是，假如“TABRD [m]”或“LTABRD[m]”指令被使用，则表格指针指向 TBLP 和 TBHP 指定的地址。在这个例子中，表格数据的高字节等于零，而当“TABRD [m]”或“LTABRD[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 / 可写寄存器，且能重复储存，若主程序和中断服务程序都使用表格读取指令，应注意对 TBLH 中数据的保护。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

```
rombank1 code1
ds .section 'data'
tempreg1 db ?       ; temporary register #1
tempreg2 db ?       ; temporary register #2
:
code0 .section 'code'
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,3fh           ; initialise high table pointer
mov tbhp,a          ; it is not necessary to set tbhp if executing tabrdl or
                    ; ltabrdl
:
:
tabrd tempreg1      ; transfers value in table referenced by table pointer
                    ; data at program memory address "3F06H" 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 "3F05H" transferred to
                    ; tempreg2 and TBLH in this example the data "1AH" is
                    ; transferred to tempreg1 and data "0FH" to register
                    ; tempreg2
:
code1 .section 'code'
org 1F00h           ; sets initial address of program memory last page
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

在线烧录 – ICP

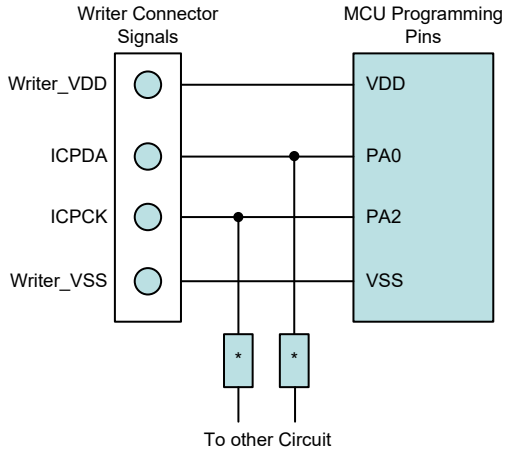
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。另外，Holtek 单片机提供 4 线接口的在线烧录方式。用户可将进行过烧录或未经过烧录的单片机芯片连同电路板一起制成，最后阶段进行程序的更新和程序的烧录，在无需去除或重新插入芯片的情况下方便地保持程序为最新版。

Holtek Flash 单片机与烧录器引脚对应表格如下：

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

程序存储器可以通过 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 HT45V0075 用于单片机仿真。此 EV 芯片提供片上调试功能 (OCDS) 用于开发过程中的单片机调试。除了片上调试功能，单片机和 EV 芯片在功能上几乎是兼容的。用户可将 OCDSDA 和 OCDSCK 引脚连接至 Holtek HT-IDE 开发工具，从而实现 EV 芯片对单片机的仿真。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，OCDSDA 和 OCDSCK 引脚上的其它共用功能对 EV 芯片无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。关于 OCDS 功能的详细描述，请参考“Holtek e-Link for 8-bit MCU OCDS 使用手册”文件。

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

在线应用编程 – IAP

Flash 型程序存储器便于用户在同一芯片上对程序进行更新和修改。单片机提供的 IAP 功能使用户可以方便地对 Flash 程序存储器进行多次编程。IAP 功能可以通过内部固件进行程序的更新，而无需外接烧录器或 PC。此外，IAP 接口通过 I/O 引脚可以设置为任何类型的通信协议，例如 UART，使用 I/O 引脚。关于内部固件，用户可以选择 Holtek 提供的版本或创建自己的内部固件。以下章节说明了如何实现 IAP 固件程序。

Flash 存储器读取 / 写入容量

Flash 存储器以页为单位进行擦 / 写操作，以字为单位进行读出操作。页的大小和写入缓冲器的大小都为 32 字。注意，在执行写入操作之前必须先执行擦除操作。

Flash 存储器擦 / 写功能成功使能时 CFWEN 位会被硬件置高，当该位被置高，便可写入数据到“写入缓冲器”。FWT 位用于启动写入程序，并指示写入操作的状态。当该位由应用程序置高时将开始一个写入程序，当写入操作结束后该位将由硬件清零。

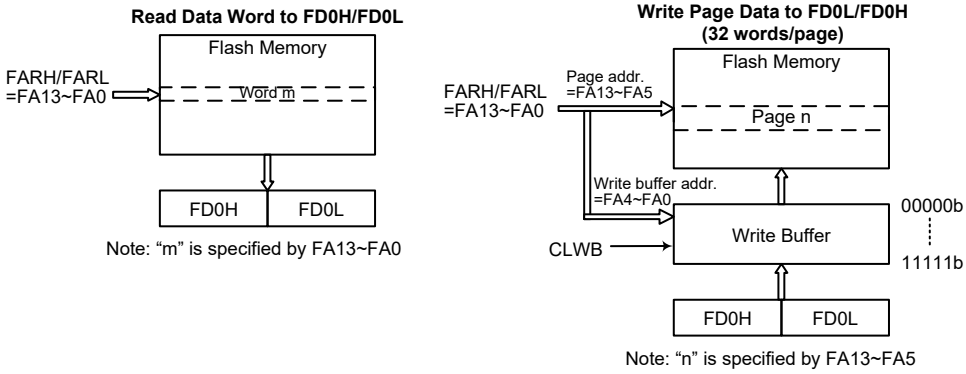
读出操作是通过一个特定的读出程序来执行的。FRDEN 位用于使能读出功能，由应用程序设置 FRD 位来启动读出程序，并指示读出操作的状态。当读出操作结束后该位将由硬件清零。

操作	格式
擦除	32 字 / 次
写入	32 字 / 次
读出	1 字 / 次
注：页大小 = 写入缓冲器大小 = 32 字	

IAP 操作格式

页	FARH	FARL[7:5]	FARL [4:0]
0	0000 0000	000	标记地址
1	0000 0000	001	
2	0000 0000	010	
3	0000 0000	011	
4	0000 0000	100	
5	0000 0000	101	
6	0000 0000	110	
7	0000 0000	111	
8	0000 0001	000	
9	0000 0001	001	
: :	: :	: :	
510	0011 1111	110	
511	0011 1111	111	

页序号及地址选择



Flash 存储器 IAP 读 / 写结构

写入缓冲器

执行写入操作时写入缓冲器用于临时存储写入的数据。通过执行 Flash 存储器擦 / 写使能程序成功使能 Flash 存储器擦 / 写功能后，才可将要写入的数据填入到写入缓冲器。通过配置 FC2 寄存器中的 CLWB 位可以清除写入缓冲器。置高 CLWB 位可以使能清除写入缓冲器程序，完成后该位会被硬件自动清零。建议第一次使用写入缓冲器或更新写入缓冲器内的数据时，应先置高 CLWB 位将写

入缓冲器清零。

写入缓冲器的大小为 32 字，与页的大小一致。写入缓冲器的地址与存储器地址位 FA13~FA5 指定的 Flash 存储器页的地址相对应。写入到 FD0L 和 FD0H 寄存器的数据会被加载到写入缓冲器。当写入数据到高字节数据寄存器 FD0H 时，会将存储在 FD0L 和 FD0H 数据寄存器内的数据都加载到写入缓冲器，并使 Flash 存储器地址自动加一，之后新的地址会被加载到 FARH 和 FARL 地址寄存器。当 Flash 存储器地址到达当前页的最大地址，即 32 字的页为 11111b，地址将不再增加，并停在该页的最后一个地址，此时需要再设定一个新的页地址才可进行其它擦 / 写操作。

写入程序结束后，硬件会自动清除写入缓冲器。注意，如果在比对步骤时发现写入到 Flash 存储器的数据不正确，则需通过应用程序手动清除写入缓冲器，在写入缓冲器被清零之后再重新对其写入数据。

IAP Flash 程序存储器寄存器

与 IAP 相关的 Flash 存取寄存器有两个地址寄存器、四个 16-bit 数据寄存器和三个控制寄存器。使用地址、数据和控制寄存器可以对 Flash 存储器执行 16 位数据读 / 写操作。内部 Flash 程序存储器所有操作由一系列寄存器控制。由于与 IAP 功能相关的寄存器都位于 Sector 1 中，它们只能通过扩展指令直接访问，或者通过存储器指针对 MP1H/MP1L 或 MP2H/MP2L 和间接寻址寄存器 IAR1 或 IAR2 进行间接读取或写入。

寄存器名称	位							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	FWERTS	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	—	—	FA13	FA12	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP 寄存器列表

● **FC0 寄存器**

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

Bit 7 **CFWEN**：Flash 存储器擦 / 写功能使能控制
0：Flash 存储器擦 / 写功能除能
1：Flash 存储器擦 / 写功能已成功使能
当此位由应用程序清零后，Flash 存储器擦 / 写功能除能。注意，对此位直接写“1”不会使能擦 / 写功能。此位用于指示 Flash 存储器擦 / 写功能状态。当此位由硬件置为“1”时，表明 Flash 存储器擦 / 写功能已经成功使能，若为“0”，表明 Flash 存储器擦 / 写功能除能。

Bit 6~4 **FMOD2~FMOD0**：Flash 存储器模式选择
000：写入模式
001：页擦除模式
011：读出模式
110：Flash 存储器擦 / 写功能使能模式
其它值：保留
这几位用于选择 Flash 存储器的操作模式。注意在执行擦 / 写 Flash 存储器操作之前必须先成功使能“Flash 存储器擦 / 写使能模式”。

Bit 3 **FWPEN**：Flash 存储器擦 / 写功能使能程序触发控制位
0：擦 / 写功能使能程序未被触发或程序定时器发生溢出
1：擦 / 写功能使能程序被触发且程序定时器开始计时
该位用于启动 Flash 存储器擦 / 写使能程序和内部定时器。此位由应用程序置高，当内部定时器计时溢出后由硬件清零。需在 FWPEN 置高后尽快写入正确数据序列到 FD1L/FD1H、FD2L/FD2H 和 FD3L/FD3H 寄存器。

Bit 2 **FWT**：Flash 存储器写入控制位
0：未开始 Flash 存储器写入程序或 Flash 存储器写入程序已完成
1：开始 Flash 存储器写入程序
此位由软件置“1”，当 Flash 存储器写入程序完成后由硬件清零。

Bit 1 **FRDEN**：Flash 存储器读出使能位
0：Flash 存储器读出除能
1：Flash 存储器读出使能
此位为 Flash 存储器读出使能位，在执行 Flash 存储器读出操作之前需将此位置高。将此位清零则禁止 Flash 存储器读出操作。

Bit 0 **FRD**：Flash 存储器读出控制位
0：未开始 Flash 存储器读出程序或 Flash 存储器读出程序已完成
1：开始 Flash 存储器读出程序
此位由软件置“1”，当 Flash 存储器读出程序完成后由硬件清零。

注：1. 在同一条指令中 FWT、FRDEN 和 FRD 位不可同时设置为“1”。
2. 需确保 f_{SUB} 时钟运行稳定后才可执行擦 / 写操作。
3. 应注意，当读 / 写 / 擦操作成功启动后，CPU 将停止运行。
4. 需确保读 / 写 / 擦操作已执行完毕后才可执行其它操作。

● **FC1 寄存器**

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

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

● **FC2 寄存器**

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

Bit 7~2 未定义，读为“0”
Bit 1 **FWERTS**：擦除和写入时间选择
0：擦除时间为 3.2ms (t_{FER}) / 写入时间为 2.2ms (t_{FWR})
1：擦除时间为 3.7ms (t_{FER}) / 写入时间为 3.0ms (t_{FWR})
Bit 0 **CLWB**：Flash 存储器写入缓冲器清除控制位
0：未开始写入缓冲器清除或写入缓冲器清除程序已完成
1：开始写入缓冲器清除程序
此位由软件置“1”，当写缓冲区清除过程完成后由硬件清零。

● **FARL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
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 **FA7~FA0**：Flash 存储器地址 bit 7 ~ bit 0

● **FARH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	FA13	FA12	FA11	FA10	FA9	FA8
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 **FA13~FA8**：Flash 存储器地址 bit 13 ~ bit 8

● **FD0L 寄存器**

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

Bit 7~0 **D7~D0**：第一个 Flash 存储器数据 bit 7 ~ bit 0
注意写入低字节数据寄存器 FD0L 的数据只能存储在 FD0L 寄存器，不会加载到低 8 位写入缓冲器。

● **FD0H 寄存器**

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

Bit 7~0 **D15~D8**：第一个 Flash 存储器数据 bit 15 ~ bit 8

注意当写入 8 位数据到高字节数据寄存器 FD0H 时，存储在 FD0H 和 FD0L 寄存器内的 16 位数据将同时加载到 16 位写入缓冲器中，此时 Flash 存储器地址寄存器 FARH 和 FARL 的内容将自动加一。

● **FD1L 寄存器**

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

Bit 7~0 **D7~D0**：第二个 Flash 存储器数据 bit 7 ~ bit 0

● **FD1H 寄存器**

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

Bit 7~0 **D15~D8**：第二个 Flash 存储器数据 bit 15 ~ bit 8

● **FD2L 寄存器**

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

Bit 7~0 **D7~D0**：第三个 Flash 存储器数据 bit 7 ~ bit 0

● **FD2H 寄存器**

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

Bit 7~0 **D15~D8**：第三个 Flash 存储器数据 bit 15 ~ bit 8

● **FD3L 寄存器**

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

Bit 7~0 **D7~D0**：第四个 Flash 存储器数据 bit 7 ~ bit 0

● **FD3H 寄存器**

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

Bit 7~0 **D15~D8**：第四个 Flash 存储器数据 bit 15 ~ bit 8

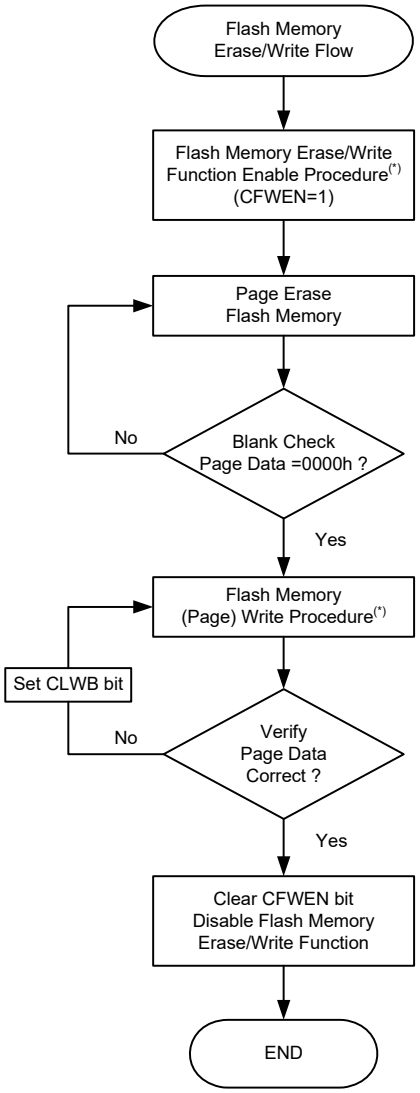
Flash 存储器擦 / 写流程

在开始更新 Flash 存储器之前，先了解 Flash 存储器擦 / 写流程操作是很重要的，用户可参考下列步骤进行 IAP 程序开发，以确保 Flash 存储器内容更新正确。

Flash 存储器擦 / 写流程说明

1. 先启动“Flash 存储器擦 / 写使能程序”。当 Flash 存储器擦 / 写功能成功使能后，FC0 寄存器中的 CFWEN 位会由硬件自动置高，此时才可执行 Flash 存储器擦或写操作。详细内容请参考“Flash 存储器擦 / 写使能程序”。
2. 配置 Flash 存储器地址以指定要擦除的页，标记地址，然后擦除此页。

   对于页擦除操作，首先设置 FARL 和 FARH 寄存器来指定要擦除页的起始地址，然后写入任意数据到 FD0H 寄存器来标记地址。每写入一个任意数据到 FD0H 寄存器，当前地址将自动加一。当地址自动递增到当前页的最大地址，即 11111b，地址将不再增加，并停在该页的最后一个地址。注意写数据到 FD0H 是为了标记地址，这一操作必须执行以确定要擦除哪些地址。
3. 查空确认是否擦除成功，可采用 TABRD 指令进行读取并比对是否为“0000h”，如果擦除不成功返回步骤 2 再执行页擦除。
4. 写入数据至该页，详细内容请参考“Flash 存储器写入程序”。
5. 采用 TABRD 指令进行读取并比对写入数据是否正确，如果读出的数据与写入数据不符，即写入不成功，设置 CLWB 位为“1”清除“写入缓冲器”再返回步骤 4，再写入相同数据。
6. 完成当前页擦 / 写后，如果无需擦 / 写其它页，可清除 CFWEN 位来除能“Flash 存储器擦 / 写使能模式”。



Flash 存储器擦 / 写流程

注：“Flash 存储器擦 / 写功能使能程序”和“Flash 存储器写程序”将在后面介绍。

Flash 存储器擦 / 写使能步骤

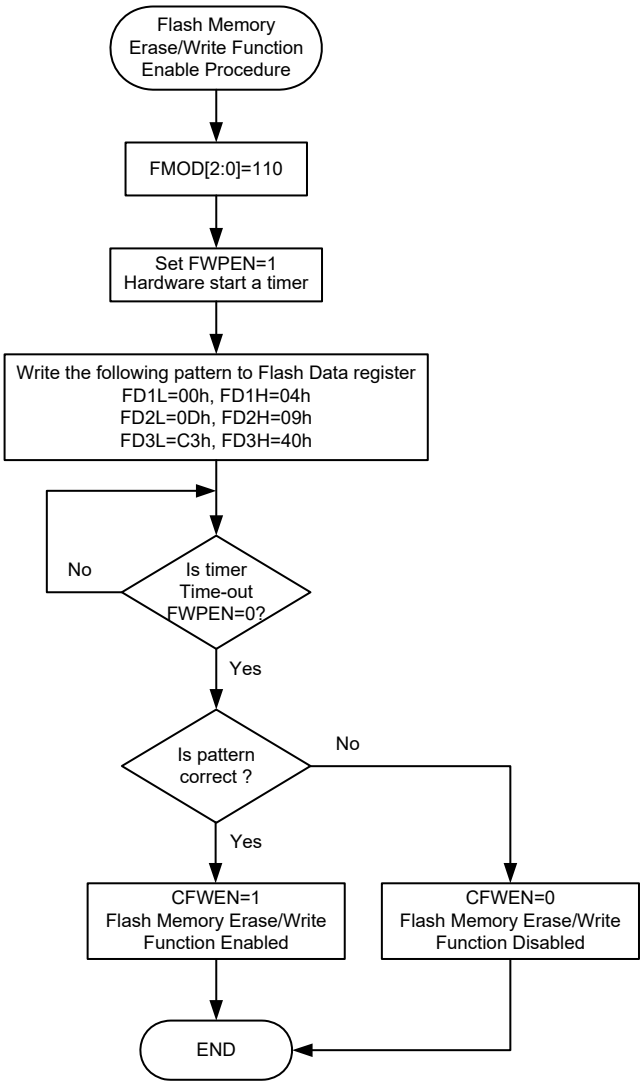
Flash 存储器擦 / 写使能模式是为防止 Flash 存储器内容被误修改而专门设计的。用户必须先使能 Flash 存储器擦 / 写功能，才能通过 IAP 控制寄存器来更改 Flash 存储器数据。

Flash 存储器擦 / 写使能步骤说明

1. 写入数值“110”至 FC0 寄存器中的 FMOD[2:0] 位，选择 Flash 存储器擦 / 写使能模式。
2. 设置 FC0 寄存器中的 FWPEN 位为“1”，启动 Flash 存储器擦 / 写使能程序，此时内部硬件线路会启动一个内部定时器。
3. 用户必须在 FWPEN 位置高后尽快填入正确数据序列至 FD1L~FD3L 和 FD1H~FD3H 寄存器中，数据序列为 FD1L=00h、FD1H=04h、FD2L=0Dh、FD2H=09h、FD3L=C3h、FD3H=40h。

4. 一旦定时器计时结束，无论写入的数据序列是否正确，FWPEN 位将由硬件自动清零。
5. 如果写入的数据序列不正确，表示 Flash 存储器擦 / 写功能没有成功使能，需重复以上步骤。如果写入的数据序列正确，表示 Flash 存储器擦 / 写功能成功使能。
6. 一旦 Flash 存储器擦 / 写功能成功使能，即可通过 IAP 控制寄存器进行页擦 / 写操作来更新 Flash 存储器内容。

将 FC0 寄存器中的 CFWEN 位清零，可除能 Flash 存储器擦 / 写功能，此时不必再执行以上步骤。



Flash 存储器擦 / 写功能使能步骤

Flash 存储器写入步骤

当 Flash 擦 / 写功能成功使能后，CFWEN 位会被硬件置高，此时要写入 Flash 存储器的数据才能加载到写入缓冲器。在开始写入程序之前，应先正确配置 IAP 控制寄存器，将所选的 Flash 存储器页的数据擦除。

写入缓冲器的大小为每页 32 字，其地址与 FA13~FA5 指定的 Flash 存储器页的地址为相对应关系。注意，写入缓冲器的地址与对应存储器的地址必须在相同页。

Flash 存储器连续地址写入步骤说明

对于写入操作每次写入的数据最多为 32 字。多笔连续地址的数据写入时，写入缓冲器的地址将自动加“1”。用户只需将第一笔数据的地址填入 FARL 和 FARH，并将第一笔数据依序填入 FD0L 和 FD0H 寄存器。先写 FD0L 再写 FD0H，才会将 FD0L 和 FD0H 数据一起填入写入缓冲器。写入缓冲器的地址将自动加“1”，因此，要填入第二笔数据时，可不用修改 FARL 和 FARH 重新指定地址。当连续地址到达当前页的最后一个地址时，写入缓冲器的地址将不会再自动加“1”，保持在最后一个地址。

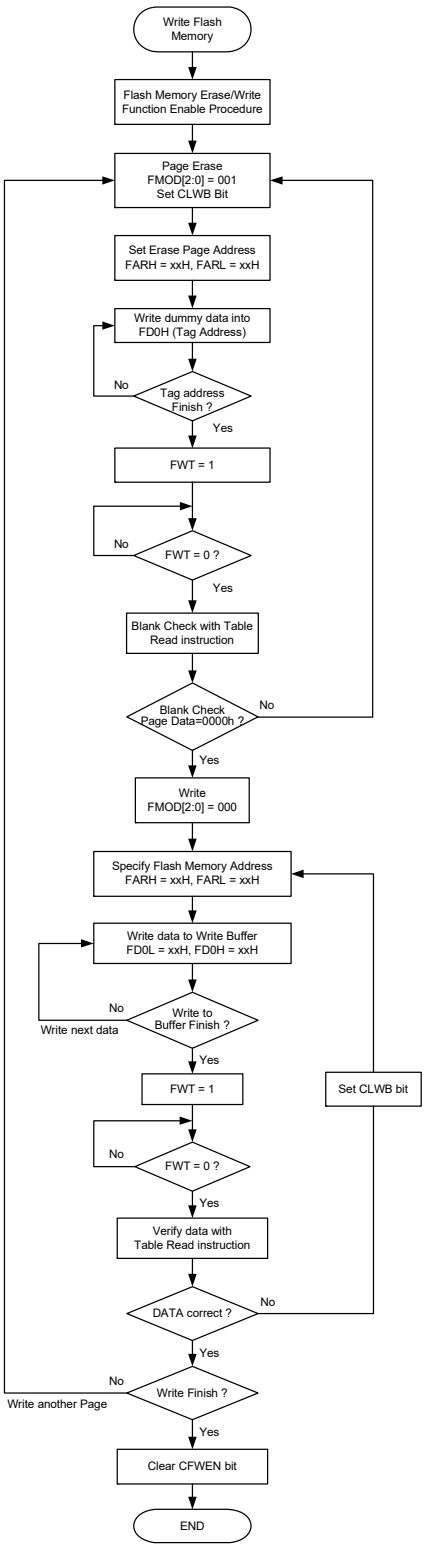
1. 启动“Flash 存储器擦 / 写使能程序”，确认 CFWEN 的值，如果 CFWEN 被硬件置高，表示可进行 IAP 擦 / 写操作。详细内容请参考“Flash 存储器擦 / 写使能程序”。
2. 设定 FMOD[2:0] 为“001”，选择擦除模式，并且设定 CLWB 位为“1”清除“写入缓存器”。设定 FWT 位为“1”，擦除目标页，该页由 FARH 和 FARL 指定且需标记地址，直到 FWT 变为“0”。
3. 通过查表指令读出方式进行查空，以确保擦除操作已成功完成。

   如果擦除操作不成功则返回步骤 2。

   如果擦除操作成功则接着执行步骤 4。
4. 设定 FMOD[2:0] 为“000”，选择写入模式。
5. 先将目标起始地址写入 FARL 和 FARH 寄存器中，将要往连续地址所在页写入的数据依序写入 FD0L 和 FD0H 寄存器。最多可写入 32 字。
6. 设定 FWT 位为“1”，将写入缓冲器的数据写入到对应的 Flash 存储器中，直到 FWT 变为“0”。
7. 通过查表指令读出方式进行数据比对，以确保写入操作已成功完成。

   如果写入操作不成功，设置 CLWB 位为“1”清除写入缓冲器，再返回步骤 5。

   如果写入操作成功则接着执行步骤 8。
8. 将 CFWEN 位清零以除能 Flash 存储器擦 / 写功能。



Flash 存储器连续地址写入步骤

注：1. 当擦 / 写操作成功启动时，所有 CPU 相关的操作将暂停。
2. 在擦除或写入操作中，FWT 位由高变低所需时间可以通过 FC2 寄存器中的 FWERTS 位选择。

Flash 存储器非连续地址写入步骤说明

连续地址写入操作与非连续地址写入操作的主要差别在于要写入的数据是否位于连续地址。如果要写入的数据不是位于连续的地址，当一笔数据成功写入到 Flash 存储器后需重新配置另一个目标地址。

以两笔非连续的数据写入操作为例，说明如下：

1. 启动“Flash 存储器擦 / 写使能程序”，确认 CFWEN 位的值，如果 CFWEN 被硬件置高，表示可进行 IAP 擦 / 写操作。详细内容请参考“Flash 存储器擦写使能程序”。
2. 设定 FMOD[2:0] 为“001”，选择擦除模式，并且设定 CLWB 位为“1”清除“写入缓存器”。设定 FWT 位为“1”，擦除目标页，该页由 FARH 和 FARL 指定且需标记地址，直到 FWT 变为“0”。
3. 通过查表指令读出方式进行查空，以确保擦除操作已成功完成。

   如果擦除操作不成功则返回步骤 2。

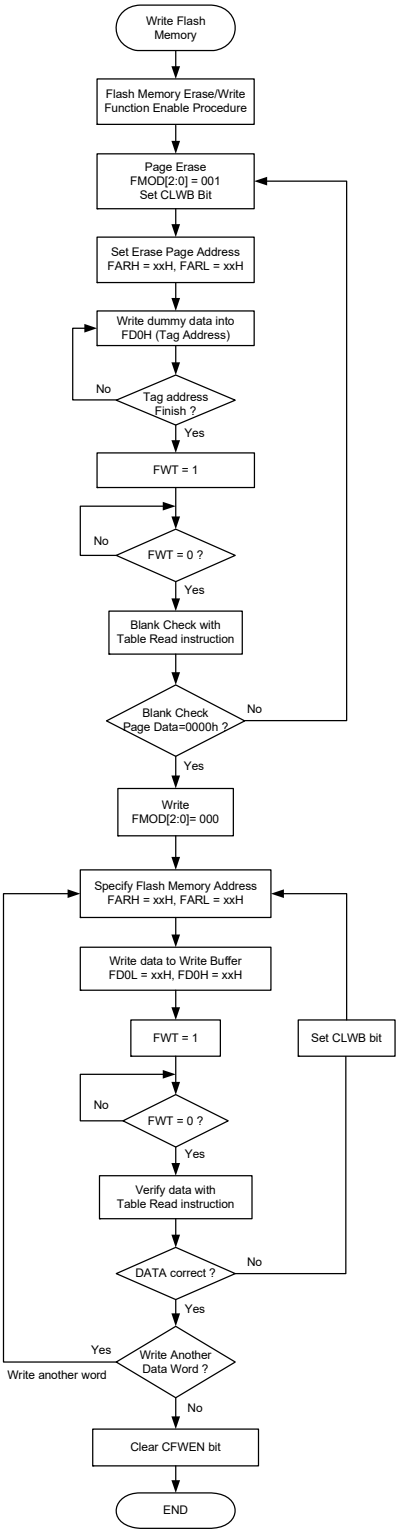
   如果擦除操作成功则接着执行步骤 4。
4. 设定 FMOD[2:0] 为“000”，选择写入模式。
5. 先将目标地址 ADDR1 写入 FARL 和 FARH 寄存器中，将要写入的数据 DATA1 先写入 FD0L 寄存器再写入 FD0H 寄存器。
6. 设定 FWT 位为“1”，将写入缓冲器的数据写入到对应的 Flash 存储器中，直到 FWT 变为“0”。
7. 通过查表指令读出方式进行数据比对，以确保写入操作已成功完成。

   如果写入操作不成功，设置 CLWB 位为“1”清除写入缓冲器，再返回步骤 5。

   如果写入操作成功则接着执行步骤 8。
8. 再将目标地址 ADDR2 写入 FARL 和 FARH 寄存器中，将要写入的数据 DATA2 先写入 FD0L 寄存器再写入 FD0H 寄存器。
9. 设定 FWT 位为“1”，将写入缓冲器的数据写入到对应的 Flash 存储器中，直到 FWT 变为“0”。
10. 通过查表指令读出方式进行数据比对，以确保写入操作已成功完成。

    如果写入操作不成功，设置 CLWB 位为“1”清除写入缓冲器，再返回步骤 8。

    如果写入操作成功则接着执行步骤 11。
11. 将 CFWEN 位清零以除能 Flash 存储器擦 / 写功能。



Flash 存储器非连续地址写入步骤

注：1. 当擦 / 写操作成功启动时，所有 CPU 相关的操作将暂停。

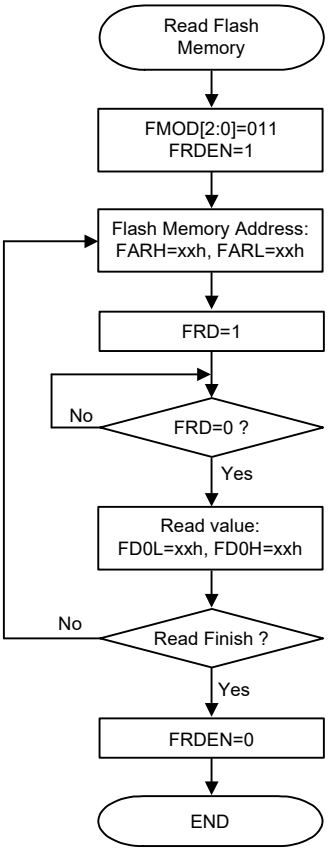
2. 在擦除或写入操作中，FWT 位由高变低所需时间可以通过 FC2 寄存器中的 FWERTS 位选择。

Flash 存储器写入操作注意事项

1. 要开始对 Flash 存储器进行 IAP 擦 / 写操作之前，必须先完成“Flash 存储器擦 / 写使能程序”。
2. Flash 存储器擦除操作以页为单位进行擦除。
3. 写入缓冲器中的数据填入 Flash 存储器是以页为单位进行的，且写入时不可跨页填写。
4. 数据写入 Flash 存储器后，必须以查表指令“TABRD”读出方式比对所写数据是否正确，若比对发现写入数据不正确时，通过置高 CLWB 位将写入缓冲器清除，然后重新写入数据。无需清除对应的 Flash 存储器页，直接再写入，然后再比对，直到写入正确。
5. IAP 写入与数据比对时需与最高应用频率相同。

Flash 存储器读出步骤

要启动 Flash 存储器读出程序，需将 FMOD[2:0] 位设为“011”选择 Flash 存储器读出模式，将 FRDEN 位设为“1”使能读出功能。将要读出的地址填入 FARH 和 FARL 地址寄存器中，并将 FRD 位设为“1”，然后便可开始 Flash 存储器读出操作。当 FRD 被硬件清为“0”时，则可从 FD0H 和 FD0L 寄存器中取得 Flash 存储器中该地址数据。进行 Flash 存储器读出操作前，无需执行 Flash 存储器擦 / 写使能程序。



Flash 存储器读出步骤

注：1. 当读操作成功启动时，所有 CPU 相关的操作将暂停。
2. FRD 位状态由高变低所需的典型时间为三个指令周期。

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。

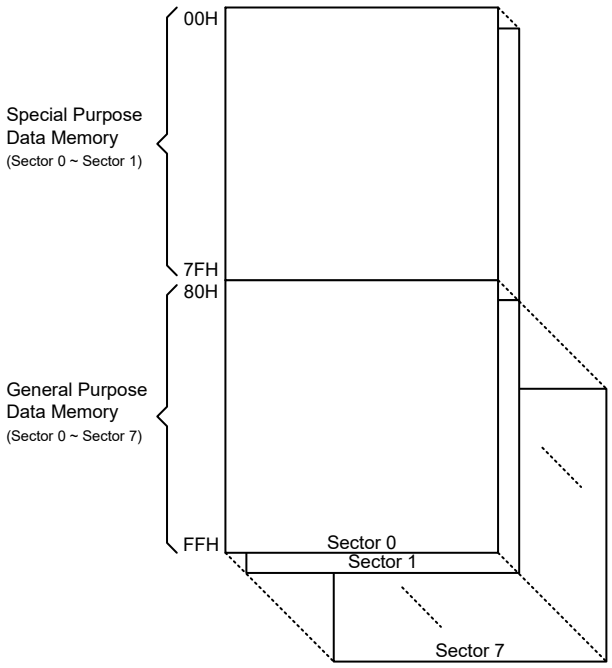
结构

数据存储器被分为若干个 Sector，都可在 8-bit 的存储器中实现。特殊功能数据寄存器地址范围为 00H~7FH，而通用数据存储器地址范围为 80H~FFH。

切换不同的数据存储器 Sector 可通过设置正确的存储器指针值实现。

特殊功能数据存储器	通用数据存储器	
所在 **Sector**	容量	**Sector**：地址
0, 1	1024×8	0：80H~FFH 1：80H~FFH : 7：80H~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，因此它并没有可用于数据存储器 sector 选择的存储区指针。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，通过扩展指令可以直接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 11 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，该数据存储器区域就是通用数据存储器。这个数据存储区可让使用者进行读取和写入的操作。使用位操作指令可对个别的位做置位或复位的操作，较大地方便了用户在数据存储器内进行位操作。

特殊功能数据存储器

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

	Sector 0	Sector 1
00H	IAR0	PAS0
01H	MP0	PAS1
02H	IAR1	PBS0
03H	MP1L	PBS1
04H	MP1H	PCS0
05H	ACC	PCS1
06H	PCL	IFS
07H	TBLP	LVRC
08H	TBLH	TLVRC
09H	TBHP	LVDC
0AH	STATUS	WDTC
0BH	PBP	INTEG
0CH	IAR2	INTDB
0DH	MP2L	PSCR
0EH	MP2H	TB0C
0FH	RSTFC	TB1C
10H	SCC	PCKC
11H	HIRCC	CTM0C0
12H	ORMC	CTM0C1
13H	IECC	CTM0DL
14H	PA	CTM0DH
15H	PAC	CTM0AL
16H	PAPU	CTM0AH
17H	PAWU	CTM1C0
18H	PB	CTM1C1
19H	PBC	CTM1DL
1AH	PBPU	CTM1DH
1BH	PC	CTM1AL
1CH	PCC	CTM1AH
1DH	PCPU	CTM2C0
1EH	INTC0	CTM2C1
1FH	INTC1	CTM2DL
20H	INTC2	CTM2DH
21H	INTC3	CTM2AL
22H	MFI0	CTM2AH
23H	MFI1	CTM3C0
24H	MFI2	CTM3C1
25H	MFI3	CTM3DL
26H	MFI4	CTM3DH
27H	MFI5	CTM3AL
28H	MFI6	CTM3AH
29H	MFI7	PTMC0
2AH	MFI8	PTMC1
2BH	SADC0	PTMDL
2CH	SADC1	PTMDH
2DH	SADC2	PTMAL
2EH	SADOL	PTMAH
2FH	SADOH	PTMRPL
30H	SADO1BL	PTMRPH
31H	SADO1BH	FC0
32H	SADO2BL	FC1
33H	SADO2BH	FC2
34H	LEBC	FARL
35H	ATAC1C	FARH
36H	ATAC2C	FD0L
37H	ATADT	FD0H
38H	TCRL	FD1L
39H	TCRH	FD1H
3AH	TCRC	FD2L
3BH	CRCCR	FD2H
3CH	CRCIN	FD3L
3DH	CRCDL	FD3H
3EH	CRCDH	STKPTR
3FH	EEAL	

	Sector 0	Sector 1
40H	EEAH	EEC
41H	EED	FJTMC
42H	SIMC0	FJTMR1
43H	SIMC1/UUCR1	FJTMR2
44H	SIMD/UTXR_RXR	FJTMR3
45H	SIMA/SIMC2/UUCR2	FJTMR4
46H	UUCR3	FJTMD
47H	SIMTOC/UBRG	FJDT0AL
48H	UUSR	FJDT0AH
49H	PWMC0	FJDT0BL
4AH	PWMC1	FJDT0BH
4BH	PWMC2	FJDT1AL
4CH	ERSGC	FJDT1AH
4DH	PWMPC0	FJDT1BL
4EH	PWMPC1	FJDT1BH
4FH	PWMPC2	DTMINL
50H	PWMPL	DTMINH
51H	PWMPH	OPC0
52H	PWMDL	OPC1
53H	PWMDH	OPOCAL
54H	PWMRL	OVP0C0
55H	PWMRH	OVP0C1
56H	DT0L	OVP0C2
57H	DT0H	OVP0DA
58H	PLC	OVP1C0
59H	MDUWR0	OVP1C1
5AH	MDUWR1	OVP1C2
5BH	MDUWR2	OVP1DA
5CH	MDUWR3	OVP2C0
5DH	MDUWR4	OVP2C1
5EH	MDUWR5	OVP2C2
5FH	MDUWCTRL	OVP2DA
60H	DT1L	OVP3C0
61H	DT1H	OVP3C1
62H	PWMMAXPL	OVP3C2
63H	PWMMAXPH	OVP3DA
64H	PWMMINPL	OVP4C0
65H	PWMMINPH	OVP4C1
66H	DECPWMP	OVP4C2
67H	TMCC0	OVP4DA
68H	TMCC1	OVP5C0
69H	TMCDL	OVP5C1
6AH	TMCDH	OVP5C2
6BH	TMCCCRAL	OVP5DA
6CH	TMCCCRAH	OVP6C0
6DH	PPDSITA	OVP6C1
6EH	DETL	OVP6C2
6FH	DETH	OVP6DA
70H	FJC0	OVP7C0
71H	FJC1	OVP7C1
72H	FJF0	OVP7C2
73H	FJM1C0	OVP7DA
74H	FJM1C1	OVP8C0
75H	FJM1C2	OVP8C1
76H	FJM1C3	OVP8C2
77H	FJM0C0	OVP8DA
78H	FJM0C1	PCRL
79H	FJM0C2	PCRH
7AH	FJPAL	DETMAXL
7BH	FJPAH	DETMAXH
7CH	FJPBL	FJC2
7DH	FJPBH	
7EH	FJPHL	
7FH	FJPHH	

: Unused, read as 00H

特殊功能数据存储器结构

特殊功能寄存器

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

间接寻址寄存器 – IAR0, IAR1, IAR2

间接寻址寄存器 IAR0、IAR1 和 IAR2 的地址虽位于数据存储区，但不同于普通寄存器，它们没有实际的物理地址。与定义实际存储器地址的直接寻址方法不同，间接寻址是使用间接寻址寄存器和存储器指针来执行存储器数据操作。在间接寻址寄存器 IAR0、IAR1 和 IAR2 上的任何动作，将对间接寻址指针 MP0、MP1L/MP1H 或 MP2L/MP2H 所指定的存储器地址产生对应的读 / 写操作。它们总是成对出现，IAR0 和 MP0 可以访问 Sector 0，而 IAR1 和 MP1L/MP1H、IAR2 和 MP2L/MP2H 可以访问任何 Sector。因为这些间接寻址寄存器不是实际存在的，直接读取将返回“00H”的结果，而直接写入此寄存器则不做任何操作。

存储区指针 – MP0, MP1L, MP1H, MP2L, MP2H

该单片机提供五个存储区指针，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样被操作，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区指针所指定的地址。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 个 RAM 地址的区块，它们已事先定义成地址 adres1 到 adres4。

间接寻址程序示例 1

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

间接寻址程序示例 2

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

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。

使用扩展指令直接寻址程序举例

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

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

程序存储区指针 – PBP

该单片机程序存储器被分为两个 Bank，可以通过设置程序存储区指针 PBP 来访问不同的程序存储区。PBP 寄存器应在单片机使用“JMP”或“CALL”指令执行“分支”操作前正确地配置。在分支指令执行后会跳转到一个非连续的程序存储器地址，此地址位于程序存储区指针所选 Bank 内。

● **PBP 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	PBP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1　未定义，读为“0”

Bit 0　**PBP0**：程序存储器 Bank 选择
0：Bank 0
1：Bank 1

累加器 – ACC

对任何单片机来说，累加器是相当重要的，且与 ALU 所完成的运算有密切关系，所有 ALU 得到的运算结果都会暂时存在 ACC 累加器里。若没有累加器，ALU 必须在每次进行如加法、减法和移位的运算时，将结果写入到数据存储器，这样会造成程序编写和时间的负担。另外数据传送也常常牵涉到累加器的临时储存功能，例如在用户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

为了提供额外的过程控制功能，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。

查表寄存器 – 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用户指定的地址。

Option 存储器映射寄存器 – ORMC

ORMC 寄存器用于使能 Option 存储器映射功能。Option 存储器的容量为 64 个字。当连续写入特定数据序列 55H 和 AAH 到该寄存器，Option 存储器映射功能将使能，通过使用查表指令即可读到 Option 存储器的内容，Option 存储器的 00H~3FH 地址会一一对应到程序存储器最后一页的 C0H~FFH 地址。

要成功使能 Option 存储器映射功能，该特定的数据序列 55H 和 AAH 必须在两个指令周期内连续写入。建议在写入该特定数据序列前应当先将总中断位 EMI 清零，在数据序列成功写入后，根据用户的需求在适当的时间再将其置高。当数据序列成功写入时会启动内部定时器，$4\times t_{LIRC}$ 时间之后会自动结束映射。因此，用户需及时读出数据，否则需要重新启动 Option 存储器映射功能。每次 ORMC 寄存器被连续写入后，定时器都会重新计数。

当使用查表指令来读取 Option 存储器内容时，“TABRD [m]”和“TABRDL [m]”指令皆可使用。然而，若使用“TABRD [m]”指令来读取，必须配置 TBHP 寄存器将表格指针设定在最后一页。更多查表的描述请参考相关章节。

- **ORMC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ORMC7	ORMC6	ORMC5	ORMC4	ORMC3	ORMC2	ORMC1	ORMC0
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 **ORMC7~ORMC0**：Option 存储器映射特定序列

当将特定序列 55H 和 AAH 连续写入该寄存器，会使能 Option 存储器映射功能。需注意，单片机从空闲 / 休眠模式唤醒后，该寄存器的内容将被清除。

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。

● **STATUS 寄存器**

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

"x"：未知

Bit 7 **SC**：OV 标志位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运算结果
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 Z 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 标志位也受循环移位指令的影响。

EEPROM 数据存储器

此单片机内建 EEPROM 数据存储器，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。这种存储区扩展了存储器空间，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 1024×8 位。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。通过 EEC 控制寄存器中的 MODE 模式选择位，可以确定对 EEPROM 进行字节模式或页模式读写操作。

EEPROM 寄存器

有四个寄存器控制内部 EEPROM 数据存储器总的操作，地址寄存器 EEAL 和 EEAH、数据寄存器 EED 及控制寄存器 EEC。EEAL、EEAH 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，只能通过 MP1L/MP1H 和 IAR1 或 MP2L/MP2H 和 IAR2 进行间接读取或写入。由于 EEC 控制寄存器位于 Sector 1 中的“40H”，在 EEC 寄存器上的任何操作被执行前，MP1L 或 MP2L 必须先设为“40H”，MP1H 或 MP2H 被设为“01H”。

寄存器名称	位							
	7	6	5	4	3	2	1	0
EEAL	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
EEAH	—	—	—	—	—	—	EEAH1	EEAH0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD

EEPROM 寄存器列表

- **EEAL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
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 **EEAL7~EEAL0：**数据 EEPROM 地址低字节 bit 7 ~ bit 0

- **EEAH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	EEAH1	EEAH0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

Bit 1~0 **EEAH1~EEAH0：**数据 EEPROM 地址高字节 bit 1 ~ bit 0

● **EED 寄存器**

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

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

● **EEC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD
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 **EWERTS**：EEPROM 擦 / 写时间选择位
0：擦除时间为 3.2ms (t_{EEER}) / 写入时间为 2.2ms (t_{EEWR})
1：擦除时间为 3.7ms (t_{EEER}) / 写入时间为 3.0ms (t_{EEWR})

Bit 6 **EREN**：数据 EEPROM 擦使能位
0：除能
1：使能
此位为数据 EEPROM 擦使能位，向数据 EEPROM 擦操作之前需将此位置高。擦周期结束后，硬件自动将此位清零。将此位清零时，则禁止向数据 EEPROM 擦操作。

Bit 5 **ER**：EEPROM 擦控制位
0：擦周期结束
1：开始擦周期
此位为 EEPROM 擦控制位，由应用程序将此位置高将激活擦周期。擦周期结束后，硬件自动将此位清零。当 EREN 未先置高时，此位置高无效。

Bit 4 **MODE**：EEPROM 操作模式选择位
0：字节操作模式
1：页操作模式
此位为 EEPROM 页操作选择位，当此位置高将选择页写入、擦除或读取功能。否则，EEPROM 为字节写入或读取功能。EEPROM 页缓冲区大小为 16 字节。

Bit 3 **WREN**：数据 EEPROM 写使能位
0：除能
1：使能
此位为 EEPROM 写使能位，向数据 EEPROM 写操作之前需将此位置高。写周期结束后，硬件自动将此位清零。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：开始写周期
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：开始读周期
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：1. 在同一条指令中 EREN、ER、WREN、WR、RDEN 和 RD 不能同时置为“1”。
2. 需确保 f_{SUB} 时钟运行稳定后才可执行擦 / 写操作。
3. 需确保擦 / 写操作已执行完毕后才可改写 EEPROM 相关寄存器或启动 IAP 功能。

从 EEPROM 中读取数据

此单片机有两种模式可实现从 EEPROM 中读取数据，即字节读模式和页读模式，可通过 EEC 寄存器中的 EEPROM 操作模式选择位 MODE 选择。

字节读模式

当模式选择位 MODE 为 0 时，可以执行 EEPROM 字节读操作。对于字节读操作，应首先将要读取数据的地址放入 EEAH 和 EEAL 寄存器中，并将 EEC 寄存器中的读取使能位 RDEN 置高以使能读取功能。然后，将 RD 位置高，以开始 EEPROM 字节读操作。请注意，若 RD 位已置为高而 RDEN 位还未被置高则不能开始读操作。若读取周期结束，RD 位将自动清除为“0”，数据可以从 EED 寄存器中读取。读到的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。

页读取模式

当模式选择位 MODE 置高时，可以执行 EEPROM 页读操作。对于页读取操作，页大小最多为 16 个字节。对于页读操作，应首先将要读取页的起始地址放入 EEAH 和 EEAL 寄存器中，并将 EEC 寄存器中的读取使能位 RDEN 置高以使能读取功能。然后，将 RD 位置高，以开始 EEPROM 字节读操作。请注意，若 RD 位已置为高而 RDEN 位还未被设置则不能开始读操作。当前字节读周期结束时，RD 位将自动清零，此时可以从 EED 寄存器读取 EEPROM 数据，而且当前地址由硬件自动加一。只要再置高 RD 位无需重新配置 EEPROM 地址和 RDEN 控制位，就可以连续读取下一个 EEPROM 地址的数据。应用程序可轮询 RD 位以确定数据可以有效地被读取。

EEPROM 的高 6 位地址，用于指定要读取的页位置，而低 4 位用于指向实际地址。在页读取操作模式下，低 4 位的地址值将自动加 1。但是，高 6 位的地址值不会自动增加。当 EEPROM 地址低 4 位自动递增到页的上限，即 0FH，EEPROM 地址低 4 位的值会停止在 0FH，EEPROM 地址将不会再增加。

EEPROM 页擦操作

当模式选择位 MODE 为 1 时，可执行 EEPROM 页擦操作。EEPROM 一页可擦除 16 个字节。上电复位后内部页缓存器将由硬件清零。当 EEPROM 擦使能控制位 EREN 由 1 变为 0 时，内部页缓存器也会被清零。注意当 EREN 位由 0 变为 1 时，内部页缓存器不会清零。EEPROM 地址高 6 位用来指定要擦除页的位置，而低 4 位用来指向实际的地址。在页擦操作模式每写入一字节 Dummy 数据到 EED 寄存器，低 4 位地址将自动加一，而高 6 位地址不会自动增加。当 EEPROM 地址低 4 位自动递增到页的上限，即 0FH，EEPROM 地址低 4 位的值会停止在 0FH，EEPROM 地址将不会再增加。

为了实现页擦操作，EEPROM 中要擦除页的起始地址需先放入 EEAH 和 EEAL 寄存器中，要写入的 Dummy 数据也需存入 EED 寄存器中。一页的最大数据长度为 16 字节。注意写 EED 是为了标记地址，这一操作必须执行以确定要擦除哪些地址。当一整页的 Dummy 数据都写入 EED 寄存器后，EEC 寄存器中的 EREN 位先置高以使能擦功能，然后 EEC 寄存器中的 ER 位需立即置高以开始擦操作。这两条指令必须在两个指令周期内连续执行才可成功启动一个擦除操作。进行擦除操作之前应先将总中断使能位 EMI 清零，在一个有效的擦启动步骤完成之后再将其使能。

由于控制 EEPROM 擦除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 ER 位或判断 EEPROM 中断以检测擦除周期是否完成。若擦除周期完成，ER 位将自动清零，通知用户页数据已擦除。因此，应用程序将轮询 RD 位以确定擦周期是否结束。擦操作结束后，EREN 位将由硬件置零。执行完一个页擦操作后，EEPROM 被擦除页的内容将全为零。

EEPROM 写操作

从 EEPROM 中写入数据可以通过此设备的两种模式实现，即字节写模式或页写模式，由 EEC 寄存器中的 EEPROM 操作模式选择位 MODE 控制。

字节写模式

当模式选择位 MODE 为 0 时，可执行 EEPROM 字节写操作。为了实现字节写操作，EEPROM 中要写入数据的地址需先放入 EEAH 和 EEAL 寄存器中，写入的数据也需存入 EED 寄存器中。EEC 寄存器中的写使能位 WREN 先置高以使能写功能，然后 EEC 寄存器中的 WR 位需立即置高以开始写操作。这两条指令必须在两个指令周期内连续执行才可成功启动一个写操作。进行写操作之前应先将总中断使能位 EMI 清零，在一个有效的写启动步骤完成之后再将其使能。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。写入操作完成后，WREN 位由硬件置零。注意，字节写操作被成功启动前会自动执行字节擦除操作。

页写模式

在执行页写操作之前，确保已成功执行相关页擦操作。当模式选择位 MODE 设置为高时，执行 EEPROM 页写操作。EEPROM 能够写入 16 字节的页面。上电复位后内部页缓存器将由硬件清零。当 EEPROM 写使能控制位 WREN 由 1 变为 0 时，内部页缓存器也会被清零。注意当 WREN 位由 0 变为 1 时，内部页缓存器不会清零。除了最多可以写入 16 字节 EEPROM 数据以外，页写操作启动的方法与字节写操作相同。EEPROM 地址高 6 位用来指定要写入页的位置，而低 4 位用来指向实际的地址。在页写操作模式每写入一字节数据到 EED 寄存器，低 4 位地址将自动加一，而高 6 位地址不会自动增加。当 EEPROM 地址低 4 位自动递增到页的上限，即 0FH，EEPROM 地址低 4 位的值会停止在 0FH，EEPROM 地址将不会再增加。此时再对 EED 寄存器写入数据也将无效。

为了实现页写操作，EEPROM 中要写入页的起始地址需先放入 EEAH 和 EEAL 寄存器中，要写入的数据也需存入 EED 寄存器中。一页的最大数据长度为 16 字节。注意当写入一字节数据到 EED 寄存器，EED 中的数据会加载到内部页缓存器中，然后当前地址值会自动加一。当一页数据被全部写入页缓存器，EEC 寄存器中的写使能位 WREN 先置高以使能写功能，然后 EEC 寄存器中的 WR 位需立即置高以开始写操作。这两条指令必须在两个指令周期内连续执行才可成功启动一个写操作。进行写操作之前应先将总中断使能位 EMI 清零，在一个有效的写启动步骤完成之后再将其使能。若 WR 位已置为高而 WREN 位还未被设置则不能开始写操作。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。写入操作完成后，WREN 位由硬件置零。

写保护

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

EEPROM 中断

EEPROM 擦 / 写周期结束后将产生 EEPROM 擦 / 写中断。EEPROM 中断需先通过设置相关中断寄存器的 DEE 位使能 EEPROM 中断。但由于 EEPROM 中断属于多功能中断，其相关的多功能中断使能位也需被置位。当 EEPROM 擦 / 写周期结束，DEF 请求标志位将被置位。若总中断，EEPROM 中断和相关的多功能中断使能且堆栈未满的情况下将跳转到相应的 EEPROM 中断向量中执行。当中断被响应，仅多功能中断标志位会被自动复位，EEPROM 中断标志位需通过应用程序手动复位。详见中断章节。

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 MP1H 或 MP2H 也可以正常清零以阻止进入 EEPROM 控制寄存器存在的 Sector 1。尽管没有必要，写一个简单的读回程序以检查新写入的数据是否正确还是应该考虑的。

WREN 位置位后，EEC 寄存器中的 WR 位需立即置位，以确保写周期正确地执行。写或擦周期开始前总中断位 EMI 应先清零，在一个有效的写或擦启动步骤完成之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 中读取一个字节数据 – 轮询法

```
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 IAR1.4                 ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H      ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L      ; user defined low byte address
MOV EEAL, A
SET IAR1.1                 ; set RDEN bit, enable read operations
SET IAR1.0                 ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0                 ; check for read cycle end
JMP BACK
CLR IAR1                   ; disable EEPROM read function
```

```
CLR MP1H
MOV A, EED                ; move read data to register
MOV READ_DATA, A
```

从 EEPROM 中读取一页数据 – 轮询法

```
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 IAR1.4                ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H     ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L     ; user defined low byte address
MOV EEAL, A
SET IAR1.1                ; set RDEN bit, enable read operations
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL READ
CALL READ
:
:
JMP PAGE_READ_FINISH
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
READ:
SET IAR1.0                ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0                ; check for read cycle end
JMP BACK
MOV A, EED                ; move read data to register
MOV READ_DATA, A
RET
:
PAGE_READ_FINISH:
CLR IAR1                  ; disable EEPROM read function
CLR MP1H
```

擦除 EEPROM 的一页数据 – 轮询法

```
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 IAR1.4                ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H     ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L     ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP Erase_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA        ; user defined data, erase mode don't care data
                          ; value
MOV EED, A
RET
```

```
:
Erase_START:
CLR EMI
SET IAR1.6                ; set EREN bit, enable erase operations
SET IAR1.5                ; start Erase Cycle - set ER bit - executed
                          ; immediately after setting EREN bit
SET EMI
BACK:
SZ  IAR1.5                ; check for erase cycle end
JMP BACK
CLR MP1H
```

写入一个字节数据到 EEPROM – 轮询法

```
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 IAR1.4                ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H     ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L     ; user defined low byte address
MOV EEAL, A
MOV A, EEPROM_DATA        ; user defined data
MOV EED, A
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit - executed
                          ; immediately after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2                ; check for write cycle end
JMP BACK
CLR MP1H
```

写入一页数据到 EEPROM – 轮询法

```
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 IAR1.4                ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H     ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L     ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP WRITE_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA        ; user defined data
MOV EED, A
RET
:
WRITE_START:
```

```
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit - executed
                          ; immediately after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2                ; check for write cycle end
JMP BACK
CLR MP1H
```

振荡器

不同的振荡器选择可以让使用者在不同的应用需求中实现更大范围的功能。振荡器的灵活性使得在速度和功耗方面可以达到较佳的优化。振荡器选择是通过应用程序使用相关的控制寄存器完成的。

振荡器概述

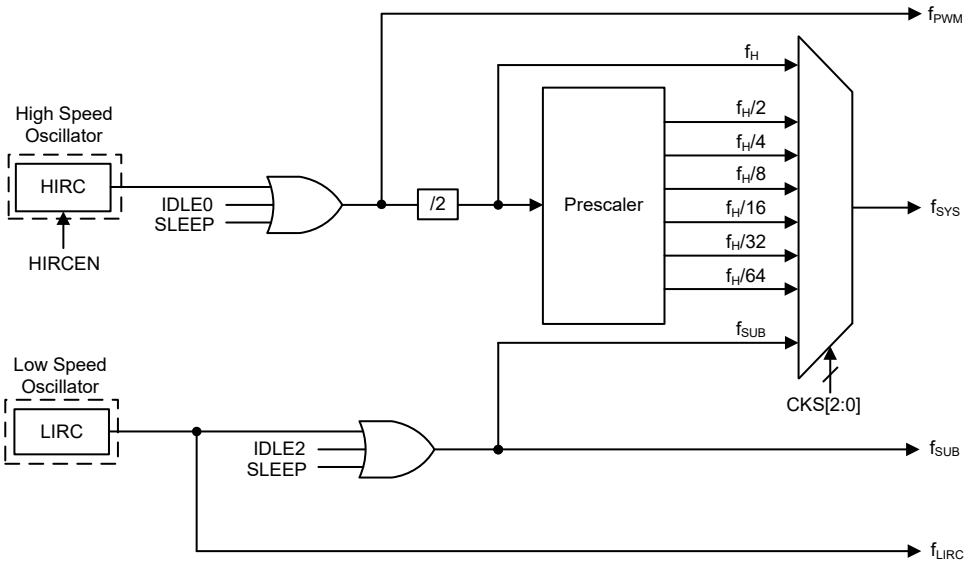
振荡器除了作为系统时钟源，还作为看门狗定时器和时基中断的时钟源。集成的两个内部振荡器不需要任何外围器件，它们提供的高速和低速系统振荡器具有较宽的频率范围。较高频率的振荡器拥有更好的性能，但会产生更高的功耗，反之亦然。动态切换不同系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

类型	名称	频率
内部高速 RC 振荡器	HIRC	f_{PWM}=32MHz， f_{HIRC}=16MHz
内部低速 RC 振荡器	LIRC	32kHz

振荡器类型

系统时钟配置

该单片机有两个振荡器，包括一个高速振荡器和一个低速振荡器。高速振荡器 HIRC 提供 32MHz 和 16MHz 频率。16MHz 可作为高速系统时钟来源。低速振荡器为内部 32kHz 低速振荡器 LIRC。使用高速或低速时钟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。



系统时钟配置

内部高速 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 振荡器可提供 32MHz 时钟 (f_{PWM}) 给 PWM 电路，也可输出 16MHz 频率 (f_{HIRC}) 作为高速内部系统时钟 f_H。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因 V_{DD}、温度以及芯片制成工艺不同的影响较大程度地降低。

内部 32kHz 振荡器 – LIRC

内部 32kHz 系统振荡器是一个低频振荡器。它是一个完全集成 RC 振荡器，典型频率值为 32kHz 且无需外部组件。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响较大程度地降低。

低速振荡器除了提供一个系统时钟源外，也用来为看门狗计时器和时基中断提供时钟来源。

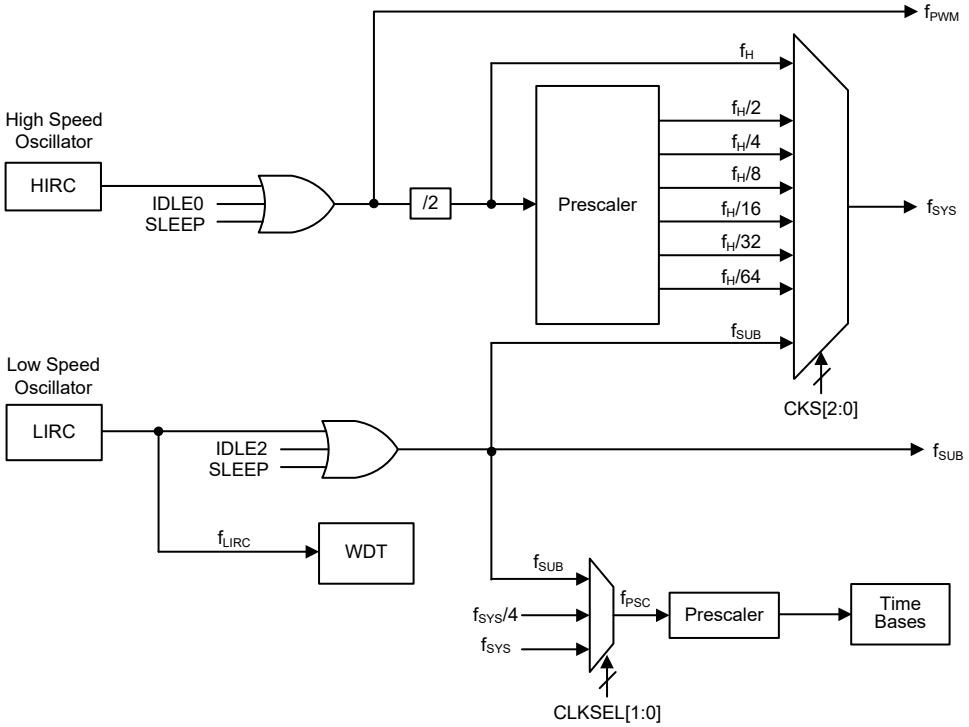
工作模式和系统时钟

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

系统时钟

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

主系统时钟可来自高频时钟源 f_H 或或低频时钟源 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} 从 f_H 切换到 f_{SUB} 时，可以通过设置相应的高速振荡器使能控制位，选择停止高速振荡器以节省耗电，但也将无法提供 f_H~$f_H/64$ 或 f_{PWM} 时钟供其它电路使用。

系统工作模式

单片机有 6 种不同的工作模式，每种有它自身的特性，根据应用中不同的性能和功耗要求可选择不同的工作模式。单片机正常工作有两种模式：快速模式和低速模式。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 关闭时以节省功耗。

工作模式	CPU	寄存器设置			f_{SYS}	f_H	f_{SUB}	f_{LIRC}
		FHIDEN	FSIDEN	CKS2~CKS0				
快速模式	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. 在低速模式中，f_H 开启或关闭由相应的振荡器使能位控制。
2. 在休眠模式中，因 WDT 功能使能，f_{LIRC} 时钟开启。

快速模式

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。该低速时钟源可来自 f_{SUB}，而 f_{SUB} 来自于 LIRC 振荡器。

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 停止运行，f_{SUB} 停止为外围功能提供时钟。因看门狗定时器功能使能，f_{LIRC} 时钟将继续运行。

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，若系统时钟来自低速振荡器，则会开启以确保一些外围功能继续工作。

空闲模式 1

执行 HALT 指令后且 SCC 寄存器中的 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，但系统时钟以及高速和低速振荡器都会开启以确保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，若系统时钟来自高速振荡器，则会开启以确保一些外围功能继续工作。

控制寄存器

寄存器 SCC 和 HIRCC 用于控制系统时钟和 HIRC 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 振荡器的 ON/OFF 也受 WDT 功能控制。若 WDT 功能使能，则即使该位为 0，LIRC 还会继续运行以提供时钟给 WDT。

注：使用 CKS2~CKS0 位进行时钟切换设置之后，在当前时钟成功切换至目标时钟源之前需要一定的延时。因此，若接下来执行的操作需要目标时钟源立即响应，则在此之前必须规划适当的延迟时间。
时钟切换延迟时间 = $4\times t_{SYS}+[0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$，其中 $t_{Curr.}$ 指代当前的时钟周期，$t_{Tar.}$ 指代目标时钟周期，t_{SYS} 指代当前系统时钟周期。

● **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”

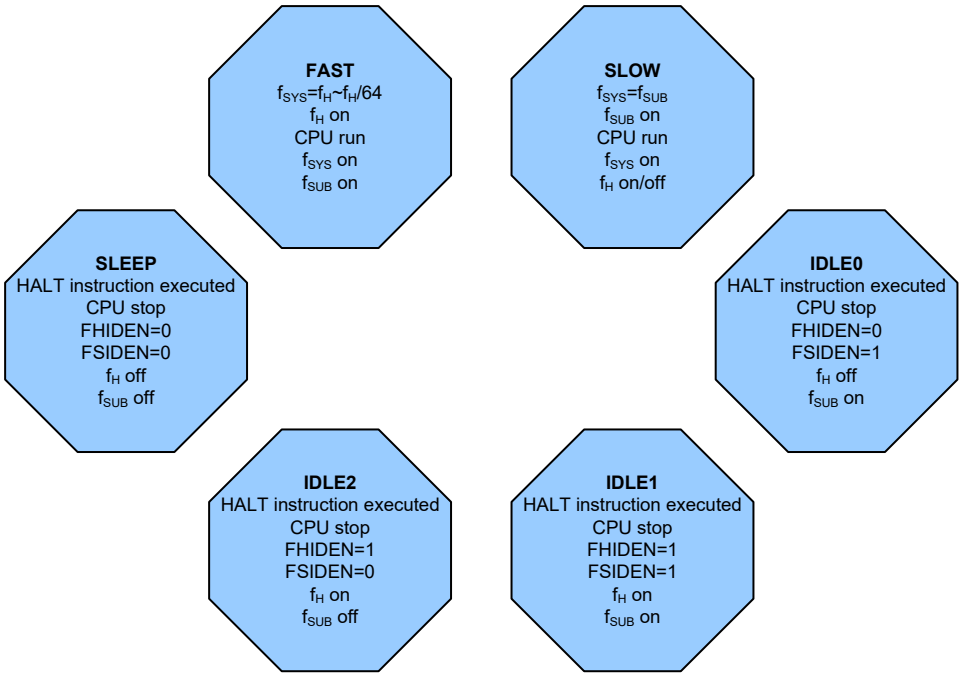
Bit 1　**HIRCF**：HIRC 振荡器稳定标志位
0：HIRC 未稳定
1：HIRC 稳定
此位用于表明 HIRC 振荡器是否稳定。HIRCEN 位置高使能 HIRC 振荡器时，HIRCF 位会先被清零，在 HIRC 稳定后会被置高。HIRC 稳定时间为 16 个时钟周期。

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

工作模式切换

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

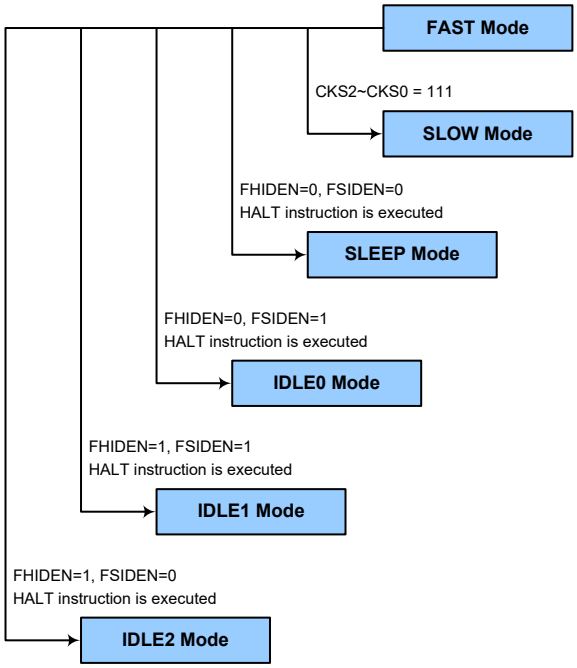
简单来说，快速模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



快速模式切换到低速模式

系统运行在快速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。

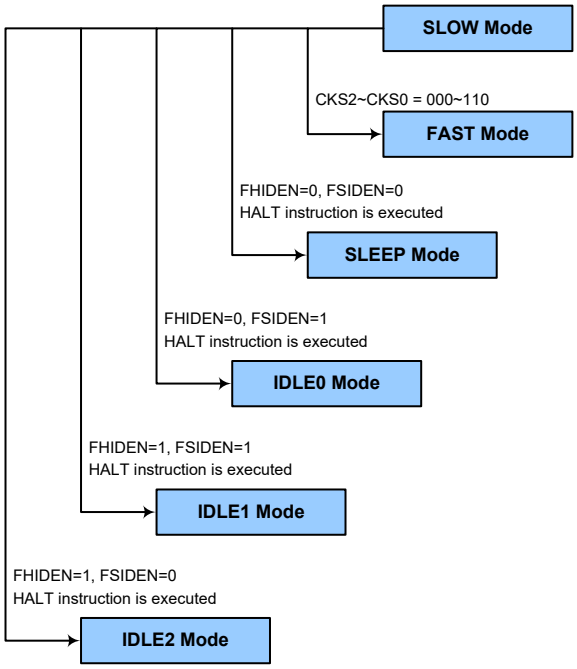
低速模式的时钟源来自 LIRC 振荡器，此振荡器需在所有模式切换动作发生前稳定下来。



低速模式切换到快速模式

在低速模式时系统时钟来自 f_{SUB}。切换回快速模式时，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快速模式时，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在系统上电时间电气特性中有说明。



进入休眠模式

进入休眠模式的方法仅有一种，即应用程序中执行“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。
- 因 WDT 功能一直使能，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，即应用程序中执行“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。
- 因 WDT 功能一直使能，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，即应用程序中执行“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。
- 因 WDT 功能一直使能，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，即应用程序中执行“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。
- 因 WDT 功能一直使能，WDT 将被清零并重新开始计数。

待机电流注意事项

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进一步降低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入 / 输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。

另外还需注意单片机设为输出的 I/O 引脚上的负载。应将它们设置在有最小拉电流的状态或将它们和其它的 CMOS 输入一样接到没有拉电流的外部电路上。还应注意的是，如果使用 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，额外的待机电流也可能会有几百微安。

唤醒

为降低功耗，可关闭 CPU 使单片机进入休眠模式或空闲模式。然而当单片机再次唤醒时，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

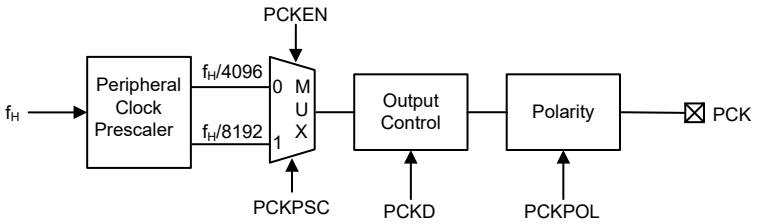
单片机执行 HALT 指令，系统进入空闲或休眠模式，PDF 将被置位；系统上电或执行清除看门狗的指令，PDF 将被清零。若系统由看门狗定时器溢出唤醒，会发生看门狗定时器复位，TO 将被置位。看门狗定时器溢出将会置位 TO 标志

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。

外围时钟输出

外围时钟输出功能允许时钟从专门的外部输出引脚 PCK 输出，使单片机能够为外部硬件提供与单片机时钟同步的时钟信号。



外围时钟输出方框图

外围时钟输出操作

外围时钟输出引脚 PCK 与其它功能共用引脚，应该合理配置相关引脚共用功能选择寄存器使能该引脚功能以输出选择的时钟。外围时钟输出功能由 PCKC 寄存器控制。设置完 PCKEN 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_H 分频，分频比由 PCKC 寄存器中的 PCKPSC 位选择。

PCK 输出真值表如下：

PCKEN	PCKD	PCKPOL	PCK 输出
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	PCK
1	1	1	PCK

外围时钟输出寄存器

外围时钟输出功能由 PCKC 寄存器控制。

- **PCKC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PCKD	PCKPOL	PCKEN	—	—	—	PCKPSC
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　**PCKD**：PCK 输出控制
0：无效
1：有效
此位用于控制 PCK 的输出是否有效。如果该位清零，PCK 引脚输出取决于 PCKPOL 位的设置。

Bit 5　**PCKPOL**：PCK 输出极性控制
0：同相
1：反相
当 PCKD=0 或 PCKEN=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 功能控制
0：除能
1：使能

Bit 3~1　未定义，读为“0”

Bit 0　**PCKPSC**：外围时钟 f_{PCK} 输出选择
0：f_H/4096
1：f_H/8192

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源来自内部时钟 f_{LIRC}，由 LIRC 振荡器提供。内部振荡器 LIRC 的频率大约为 32kHz，这个特殊的内部时钟周期会随 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 功能的使能以及溢出周期选择。空闲或休眠模式下 WDT 溢出复位后，WDTC 值不发生改变，其它任何单片机复位后，WDTC 都为 01010011B。

● **WDTC 寄存器**

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

Bit 7~3 **WE4~WE0**：WDT 功能控制位
01010 或 10101：WDT 使能
其它值：MCU 复位
若因外部环境噪声使 WE4~WE0 值发生改变，则单片机会在 t_{SRESET} 时间后产生复位，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
具体描述见低电压复位章节

Bit 1 **LRF**：LVR 控制寄存器软件复位标志位
具体描述见低电压复位章节。

Bit 0 **WRF**：WDT 控制寄存器软件复位标志位
0：未发生
1：发生
当 WDT 控制寄存器软件复位发生时，此位被置为“1”。该位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，清除指令不能被正确执行，此种情况下，看门狗将溢出以使单片机复位。WDTC 寄存器中的 WE4~WE0 位可使能看门狗定时器。如果 WE4~WE0 为 10101B 或 01010B，则 WDT 总是使能。如果 WE4~WE0 为 10101B 和 01010B 以外的其它任意值，单片机将在 t_{SRESET} 延迟时间后复位。上电后这些位的值为“01010B”。

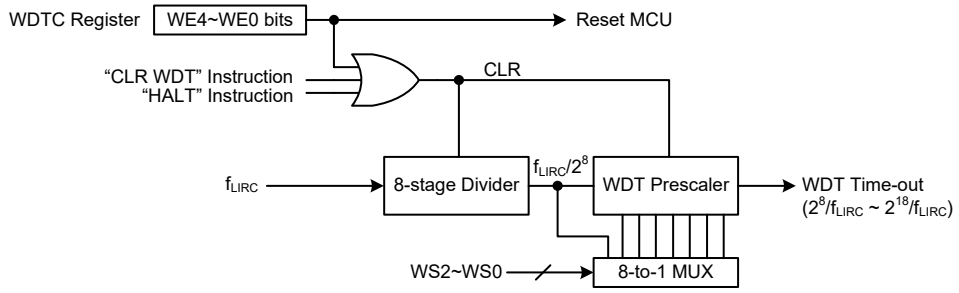
WE4~WE0 位	WDT 功能
01010B 或 10101B	使能
其它值	MCU 复位

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，PDF 位保持为高，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 WDTC 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 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。

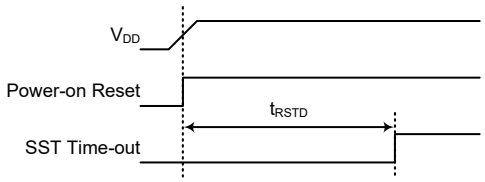
另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。此外还有一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。

复位功能

此单片机提供多种复位方式，通过内部事件触发。

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。



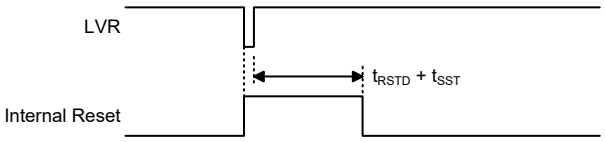
上电复位时序图

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 V_{DD}。当电源电压低于某一预设值时，它将复位单片机。在快速和低速模式下，低电压复位功能始终使能于特定的电压值，V_{LVR}。例如在更换电池的情况下，单片机供应的电压可能会在 0.9V~V_{LVR} 之间，这时 LVR 将会自动复位单片机且 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 t_{LVR} 值。如果低电压存在时间没超过 t_{LVR}，则 LVR 将会忽略它且不会执行复位功能。t_{LVR} 值可通过 TLVRC 寄存器中的 TLVR1~TLVR0 位设置。

实际的 V_{LVR} 参数值通过 LVRC 寄存器中的 LVS7~LVS0 位设置。若由于受到干扰 LVS7~LVS0 变为其它值时，单片机将在 t_{SRESET} 时间后发生复位。此时 RSTFC 寄存器的 LRF 位被置位。上电复位后寄存器的值为 01010101B。

注意，当单片机进入空闲或休眠模式时，LVR 功能将自动除能。



低电压复位时序图

低电压复位寄存器

LVRC 和 TLVRC 寄存器用于控制低电压复位功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
LVRC	LVS7	LVS6	LVS5	LVS4	LVS3	LVS2	LVS1	LVS0
TLVRC	—	—	—	—	—	—	TLVR1	TLVR0

低电压复位寄存器列表

- **LVRC 寄存器**

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

Bit 7~0 **LVS7~LVS0**：LVR 电压选择
01010101：2.1V
00110011：2.55V
10011001：3.15V
10101010：3.8V
其它值：MCU 复位 – 寄存器复位为 POR 值

当上述定义的相应的低电压出现，且低电压保持时间超过 t_{LVR} 值，则单片机复位发生。实际的 t_{LVR} 值可通过 TLVRC 寄存器中的 TLVR1~TLVR0 位设置。此种复位后，该寄存器内容保持不变。

若 LVRC 寄存器值改变为上述定义的“其它值”，将会产生单片机复位。需要经过一段 t_{SRESET} 延迟时间来响应复位。此种复位后，该寄存器的值将恢复到上电复位值。

- **TLVRC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	TLVR1	TLVR0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	1

Bit 7~2 未定义，读为“0”

Bit 1~0 **TLVR1~TLVR0**：产生 LVR 复位的低电压最短保持时间 t_{LVR} 选择
00：(7~8)×t_{LIRC}
01：(31~32)×t_{LIRC}
10：(63~64)×t_{LIRC}
11：(127~128)×t_{LIRC}
有关 t_{LVR} 具体信息可参考 LVD/LVR 电气特性表格。

- **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，产生软件复位，此位被置为“1”，且只能通过应用程序清零。

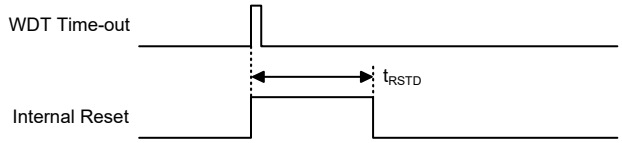
Bit 0 **WRF**：WDT 控制寄存器软件复位标志位
具体描述见看门狗定时器控制寄存器章节。

IAP 复位

当写值“55H”至 IAP 功能相关的 FC1 寄存器时，将产生一个复位信号将整个单片机复位。详见在线应用编程章节。

正常运行时看门狗溢出复位

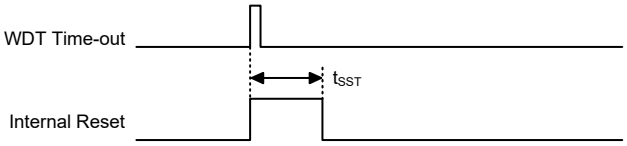
在快速或低速模式下正常运行时发生看门狗溢出复位，看门狗溢出标志位 TO 将被设为“1”。



正常运行时看门狗溢出复位时序图

休眠或空闲时看门狗溢出复位

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



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	快速模式或低速模式时的 LVR 复位
1	u	快速模式或低速模式时的 WDT 溢出复位
1	1	空闲模式或休眠模式时的 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	清除，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 口设为输入类型
堆栈指针	堆栈指针指向堆栈顶端

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

寄存器	上电复位	WDT 溢出 (正常工作)	WDT 溢出 (空闲 / 休眠)
IAR0	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	uuuu uuuu
IAR1	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	--xx xxxx	--uu uuuu	--uu uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
PBP	---- ---0	---- ---0	---- ---u
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu
SCC	001- --00	001- --00	uuu- --uu
HIRCC	---- --01	---- --01	---- --uu
ORMC	0000 0000	0000 0000	0000 0000
IECC	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
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
PC	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	uuuu uuuu
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
MFI0	-000 -000	-000 -000	-uuu -uuu
MFI1	0000 0000	0000 0000	uuuu uuuu
MFI2	--00 --00	--00 --00	--uu --uu
MFI3	--00 --00	--00 --00	--uu --uu
MFI4	-000 -000	-000 -000	-uuu -uuu
MFI5	0000 0000	0000 0000	uuuu uuuu
MFI6	0000 0000	0000 0000	uuuu uuuu
MFI7	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常工作)	WDT 溢出 (空闲 / 休眠)
MFI8	-000 -000	-000 -000	-uuu -uuu
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 0000	0000 0000	uuuu uuuu
SADC2	00-0 -000	00-0 -000	uu-u -uuu
SADOL	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
SADO1BL	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADO1BH	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADO2BL	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADO2BH	xxxx xxxx	xxxx xxxx	uuuu uuuu
LEBC	0--0 0000	0--0 0000	u--u uuuu
ATAC1C	--00 0000	--00 0000	--uu uuuu
ATAC2C	---0 0000	---0 0000	---u uuuu
ATADT	---- 0000	---- 0000	---- uuuu
TCRL	xxxx xxxx	xxxx xxxx	uuuu uuuu
TCRH	xxxx xxxx	xxxx xxxx	uuuu uuuu
TCRC	---0 --00	---0 --00	---u --uu
CRCCR	---- ---0	---- ---0	---- ---u
CRCIN	0000 0000	0000 0000	uuuu uuuu
CRCDL	0000 0000	0000 0000	uuuu uuuu
CRCDH	0000 0000	0000 0000	uuuu uuuu
EEAL	0000 0000	0000 0000	uuuu uuuu
EEAH	---- --00	---- --00	---- --uu
EED	0000 0000	0000 0000	uuuu uuuu
SIMC0	1110 0000	1110 0000	uuuu uuuu
SIMC1 (UMD=0)	1000 0001	1000 0001	uuuu uuuu
UUCR1* (UMD=1)	0000 00x0	0000 00x0	uuuu uuuu
SIMD/UTXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/SIMC2/UUCR2	0000 0000	0000 0000	uuuu uuuu
UUCR3	---- ---0	---- ---0	---- ---u
SIMTOC (UMD=0)	0000 0000	0000 0000	uuuu uuuu
UBRG* (UMD=1)	xxxx xxxx	xxxx xxxx	uuuu uuuu
UUSR	0000 1011	0000 1011	uuuu uuuu
PWMC0	0000 0000	0000 0000	uuuu uuuu
PWMC1	0000 0000	0000 0000	uuuu uuuu
PWMC2	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常工作)	WDT 溢出 (空闲 / 休眠)
ERSGC	0000 0000	0000 0000	uuuu uuuu
PWMPC0	0000 0000	0000 0000	uuuu uuuu
PWMPC1	0000 0000	0000 0000	uuuu uuuu
PWMPC2	0-00 --00	0-00 --00	u-uu --uu
PWMPL	0000 0000	0000 0000	uuuu uuuu
PWMPH	---- 0000	---- 0000	---- uuuu
PWMDL	0000 0000	0000 0000	uuuu uuuu
PWMDH	---- 0000	---- 0000	---- uuuu
PWMRL	0000 0000	0000 0000	uuuu uuuu
PWMRH	---- 0000	---- 0000	---- uuuu
DT0L	0000 0000	0000 0000	uuuu uuuu
DT0H	---- 0000	---- 0000	---- uuuu
PLC	---- --00	---- --00	---- --uu
MDUWR0	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR1	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR2	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR3	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR4	xxxx xxxx	0000 0000	uuuu uuuu
MDUWR5	xxxx xxxx	0000 0000	uuuu uuuu
MDUWCTRL	00-- ----	00-- ----	uu-- ----
DT1L	0000 0000	0000 0000	uuuu uuuu
DT1H	---- 0000	---- 0000	---- uuuu
PWMMAXPL	0000 0000	0000 0000	uuuu uuuu
PWMMAXPH	---- 0000	---- 0000	---- uuuu
PWMMINPL	0000 0000	0000 0000	uuuu uuuu
PWMMINPH	---- 0000	---- 0000	---- uuuu
DECPWMP	0000 0000	0000 0000	uuuu uuuu
TMCC0	-000 0000	-000 0000	-uuu uuuu
TMCC1	0000 ----	0000 ----	uuuu ----
TMCDL	0000 0000	0000 0000	uuuu uuuu
TMCDH	---- --00	---- --00	---- --uu
TMCCCRAL	0000 0000	0000 0000	uuuu uuuu
TMCCCRAH	---- --00	---- --00	---- --uu
PPDSITA	0000 0000	0000 0000	uuuu uuuu
DETL	0000 0000	0000 0000	uuuu uuuu
DETH	---- -000	---- -000	---- -uuu
FJC0	0000 0000	0000 0000	uuuu uuu
FJC1	0000 --00	0000 --00	uuuu --uu
FJF0	0000 --00	0000 --00	uuuu --uu
FJM1C0	-000 -000	-000 -000	-uuu -uuu
FJM1C1	-000 0000	-000 0000	-uuu uuuu
FJM1C2	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常工作)	WDT 溢出 (空闲 / 休眠)
FJM1C3	0000 0000	0000 0000	uuuu uuuu
FJM0C0	-000 0000	-000 0000	-uuu uuuu
FJM0C1	0-00 0-00	0-00 0-00	u-uu u-uu
FJM0C2	0-00 0-00	0-00 0-00	u-uu u-uu
FJPAL	0000 0000	0000 0000	uuuu uuuu
FJPAH	---- 0000	---- 0000	---- uuuu
FJPBL	0000 0000	0000 0000	uuuu uuuu
FJPBH	---- 0000	---- 0000	---- uuuu
FJPHL	0000 0000	0000 0000	uuuu uuuu
FJPHH	---- 0000	---- 0000	---- uuuu
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	0000 0000	0000 0000	uuuu uuuu
PBS1	0000 0000	0000 0000	uuuu uuuu
PCS0	0000 0000	0000 0000	uuuu uuuu
PCS1	0000 0000	0000 0000	uuuu uuuu
IFS	-000 0000	-000 0000	-uuu uuuu
LVRC	0101 0101	0101 0101	uuuu uuuu
TLVRC	---- --01	---- --01	---- --uu
LVDC	--00 0000	--00 0000	--uu uuuu
WDTC	0101 0011	0101 0011	uuuu uuuu
INTEG	---- 0000	---- 0000	---- uuuu
INTDB	---- 0000	---- 0000	---- uuuu
PSCR	---- --00	---- --00	---- --uu
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu
PCKC	-000 ---0	-000 ---0	-uuu ---u
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
CTM0AH	---- --00	---- --00	---- --uu
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
CTM2C0	0000 0000	0000 0000	uuuu uuuu
CTM2C1	0000 0000	0000 0000	uuuu uuuu
CTM2DL	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常工作)	WDT 溢出 (空闲 / 休眠)
CTM2DH	---- --00	---- --00	---- --uu
CTM2AL	0000 0000	0000 0000	uuuu uuuu
CTM2AH	---- --00	---- --00	---- --uu
CTM3C0	0000 0000	0000 0000	uuuu uuuu
CTM3C1	0000 0000	0000 0000	uuuu uuuu
CTM3DL	0000 0000	0000 0000	uuuu uuuu
CTM3DH	---- --00	---- --00	---- --uu
CTM3AL	0000 0000	0000 0000	uuuu uuuu
CTM3AH	---- --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
FC0	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	uuuu uuuu
FC2	---- --00	---- --00	---- --uu
FARL	0000 0000	0000 0000	uuuu uuuu
FARH	--00 0000	--00 0000	--uu uuuu
FD0L	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	uuuu uuuu
STKPTR	0--- -000	0--- -000	u--- -000
EEC	0000 0000	0000 0000	uuuu uuuu
FJTMC	-000 0-00	-000 0-00	-uuu u-uu
FJTMR1	0000 0000	0000 0000	uuuu uuuu
FJTMR2	0000 0000	0000 0000	uuuu uuuu
FJTMR3	0000 0000	0000 0000	uuuu uuuu
FJTMR4	0000 0000	0000 0000	uuuu uuuu
FJTMD	0000 0000	0000 0000	uuuu uuuu
FJDT0AL	0000 0000	0000 0000	uuuu uuuu
FJDT0AH	---- 0000	---- 0000	---- uuuu
FJDT0BL	0000 0000	0000 0000	uuuu uuuu
FJDT0BH	---- 0000	---- 0000	---- uuuu

寄存器	上电复位	WDT 溢出 (正常工作)	WDT 溢出 (空闲 / 休眠)
FJDT1AL	0000 0000	0000 0000	uuuu uuuu
FJDT1AH	---- 0000	---- 0000	---- uuuu
FJDT1BL	0000 0000	0000 0000	uuuu uuuu
FJDT1BH	---- 0000	---- 0000	---- uuuu
DTMINL	0000 0000	0000 0000	uuuu uuuu
DTMINH	---- 0000	---- 0000	---- uuuu
OPC0	---- 0000	---- 0000	---- uuuu
OPC1	0--0 0000	0--0 0000	u--u uuuu
OPOCAL	0010 0000	0010 0000	uuuu uuuu
OVP0C0	000- -000	000- -000	uuu- -uuu
OVP0C1	0001 0000	0001 0000	uuuu uuuu
OVP0C2	0--- 0000	0--- 0000	u--- uuuu
OVP0DA	0000 0000	0000 0000	uuuu uuuu
OVP1C0	000- -000	000- -000	uuu- -uuu
OVP1C1	0001 0000	0001 0000	uuuu uuuu
OVP1C2	0--- 0000	0--- 0000	u--- uuuu
OVP1DA	0000 0000	0000 0000	uuuu uuuu
OVP2C0	000- -000	000- -000	uuu- -uuu
OVP2C1	0001 0000	0001 0000	uuuu uuuu
OVP2C2	---- 00--	---- 00--	---- uu--
OVP2DA	0000 0000	0000 0000	uuuu uuuu
OVP3C0	000- -000	000- -000	uuu- -uuu
OVP3C1	0001 0000	0001 0000	uuuu uuuu
OVP3C2	---- 00--	---- 00--	---- uu--
OVP3DA	0000 0000	0000 0000	uuuu uuuu
OVP4C0	000- -000	000- -000	uuu- -uuu
OVP4C1	0001 0000	0001 0000	uuuu uuuu
OVP4C2	---- 00--	---- 00--	---- uu--
OVP4DA	0000 0000	0000 0000	uuuu uuuu
OVP5C0	000- -000	000- -000	uuu- -uuu
OVP5C1	0001 0000	0001 0000	uuuu uuuu
OVP5C2	---- 00--	---- 00--	---- uu--
OVP5DA	0000 0000	0000 0000	uuuu uuuu
OVP6C0	000- -000	000- -000	uuu- -uuu
OVP6C1	0001 0000	0001 0000	uuuu uuuu
OVP6C2	---- 0000	---- 0000	---- uuuu
OVP6DA	0000 0000	0000 0000	uuuu uuuu
OVP7C0	000- -000	000- -000	uuu- -uuu
OVP7C1	0001 0000	0001 0000	uuuu uuuu
OVP7C2	---- 0000	---- 0000	---- uuuu
OVP7DA	0000 0000	0000 0000	uuuu uuuu
OVP8C0	000- -000	000- -000	uuu- -uuu

寄存器	上电复位	WDT 溢出 (正常工作)	WDT 溢出 (空闲 / 休眠)
OVP8C1	0001 0000	0001 0000	uuuu uuuu
OVP8C2	---- 00--	---- 00--	---- uu--
OVP8DA	0000 0000	0000 0000	uuuu uuuu
PCRL	0000 0000	0000 0000	uuuu uuuu
PCRH	--00 0000	--00 0000	--uu uuuu
DETMAXL	0000 0000	0000 0000	uuuu uuuu
DETMAXH	---- -000	---- -000	----- -uuu
FJC2	0000 -000	0000 -000	uuuu -uuu

注：“u”表示不改变
“x”表示未知
“-”表示未定义
“*”：UUCR1 和 SIMC1 寄存器共用同一个存储器地址，UBRG 和 SIMTOC 寄存器共用同一个存储器地址。复位发生后，通过应用程序手动设置 UMD 位为“1”后可获得 UUCR1 和 UBRG 寄存器的默认值。

输入 / 输出端口

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

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
IECC	IECS7	IECS6	IECS5	IECS4	IECS3	IECS2	IECS1	IECS0

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，当引脚规划为数字输入时，可由内部连接到一个上拉电阻。这些上拉电阻可通过相应的上拉控制寄存器 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，仅当 I/O 引脚设为数字输入或 NMOS 输出时，上拉功能才会受 PxPU 控制开启，其它输出状态下上拉功能不可用。

- **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.n 端口上拉电阻控制位
0：除能
1：使能

PxPUn 位用于控制 Px.n 端口上拉电阻功能。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 口的某一个引脚发生从高电平到低电平的转换。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，只有当引脚被设置为通用 I/O 功能输入类型且单片机处于休眠 / 空闲模式时，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**：PA7~PA0 引脚唤醒功能控制位
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，当 IECM 为低时，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.n 端口输入 / 输出类型选择位
0：输出
1：输入
PxCn 位用于控制 Px.n 端口类型。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。

读端口功能

读端口功能用于管理数据读取路径，即读取的数据来自数据锁存还是 I/O 引脚。该功能专为 I/O 功能和 A/D 通道上的 IEC 60730 自诊断测试而设计。寄存器 IECC 用于使能 / 除能读端口功能。若向寄存器 IECC 写入一个特定的数据模式“11001010”，内部信号 IECM 将被置高以使能读端口功能。读端口功能使能后执行读端口指令“mov acc, Px”，相应引脚上的值将被传至累加器 ACC，其中“x”代表相应的 I/O 端口名称。读端口功能只控制读取路径，不会影响到引脚功能配置以及当前单片机的操作。若向 IECC 寄存器写入除 11001010 以外的其它值，内部信号 IECM 将被清零，除能读端口功能且数据读取路径来自锁存器。若此功能除能，引脚将作为所选中的共用引脚功能进行正常操作。

● **IECC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	IECS7	IECS6	IECS5	IECS4	IECS3	IECS2	IECS1	IECS0
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 **IECS7~IECS0**：读端口功能使能控制
11001010：IECM=1 – 读端口功能使能
其它值：IECM=0 – 读端口功能除能

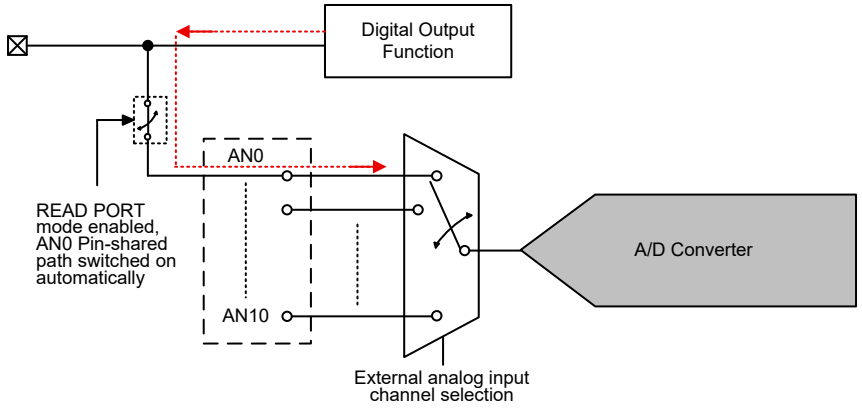
读端口功能	除能		使能	
端口控制寄存器位 – PxC.n	1	0	1	0
I/O 功能	引脚值	数据锁存值	引脚值	
数字输入功能				
数字输出功能 (USIM 引脚除外)	0			
USIM：SCK/SCL, SDI/SDA/URX/UTX	引脚值			
模拟功能	0			

注：上方表格灰色阴影内容为执行了“mov a, Px”指令后 ACC 寄存器中的内容，其中“x”表示相关的端口名称。

读端口的另一个功能是检查 A/D 信号通道。当读端口功能除能，若相应的选择位没有选中 A/D 输入引脚功能，则从外部引脚到内部模拟输入的 A/D 通道将会被关闭。对于带 A/D 转换通道的单片机，如 AN10~AN0，通过适当配置 A/D 控制寄存器中的外部模拟输入通道选择位并选中相应的模拟输入引脚功能，所需的 A/D 转换通道将被开启。而读端口模式的功能则是强制开启 A/D 通道。例如，当 AN0 选择作为 A/D 转换器模拟输入通道且读端口功能使能，则无论相应外部引脚是否选择作为 AN0 模拟输入引脚功能，AN0 模拟输入通道都将

开启。通过这种方式，AN0 模拟输入路径可与其共用引脚上的数字输出内部连接，然后在无外接模拟输入电压的情况下对相应的数字数据进行转换，从而实现 AN0 模拟输入通道检测。

注意，当使用读端口功能检查 A/D 路径时，A/D 转换器参考电压需等于 I/O 电源电压。



A/D 通道输入路径内部连接

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。

引脚共用功能选择寄存器

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”引脚共用功能选择寄存器“n”，记为 PxSn，和输入源引脚选择寄存器记为 IFS，这些寄存器可以用来选择多功能共用引脚上所需的功能以及选择一些输入功能引脚位置。

要注意的最重要一点是，确保所需的引脚共用功能被正确地选择和取消。对于大部分共用功能，要选择所需的引脚共用功能，首先应通过相应的引脚共用控制寄存器正确地选择该功能，然后再配置相应的外围功能设置以使能外围功能。但是，在设置相关引脚控制位时，一些数字输入引脚如 INTn、xTCKn 等，与对应的通用 I/O 口共用同一个引脚共用设置选项。要选择这个引脚功能，除了上述的必要的引脚共用控制和外围功能设置外，还必须将其对应的端口控制寄存器位设置为输入。要正确地取消引脚共用功能，首先应除能外围功能，然后再修改相应的引脚共用控制寄存器以选择其它的共用功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10

寄存器名称	位							
	7	6	5	4	3	2	1	0
IFS	—	IFS6	IFS5	IFS4	IFS3	IFS2	IFS1	IFS0

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 引脚共用功能选择位
00：PA3
01：OVP6P07P1
10：AN10
11：OVP6P07P1 & AN10

Bit 5~4 **PAS05~PAS04**：PA2 引脚共用功能选择位
00：PA2
01：SDI/SDA/URX/UTX
10：PA2
11：PA2

Bit 3~2 **PAS03~PAS02**：PA1 引脚共用功能选择位
00：PA1
01：OVP145P0N
10：保留
11：保留

Bit 1~0 **PAS01~PAS00**：PA0 引脚共用功能选择位
00：PA0
01：SDO/UTX
10：SCK/SCL
11：PA0

- **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 引脚共用功能选择位
00：PA7
01：TMCCKO
10：OPINN0
11：CTP3

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

Bit 3~2 **PAS13~PAS12**：PA5 引脚共用功能选择位
00：PA5
01：OVP8P
10：AN8
11：OVP8P & AN8

Bit 1~0 **PAS11~PAS10**：PA4 引脚共用功能选择位
00：PA4
01：OVP6P17P0
10：AN9
11：OVP6P17P0 & AN9

- **PBS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
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 **PBS07~PBS06**：PB3 引脚共用功能选择位
00：PB3/CTCK1/INT0
01：AN3
10：PB3/CTCK1/INT0
11：PB3/CTCK1/INT0

Bit 5~4 **PBS05~PBS04**：PB2 引脚共用功能选择位
00：PB2/CTCK0
01：PHASE
10：AN2
11：VREF

Bit 3~2 **PBS03~PBS02**：PB1 引脚共用功能选择位
00：PB1
01：AN1
10：OPARO
11：OPARO & AN1

Bit 1~0 **PBS01~PBS00**：PB0 引脚共用功能选择位
00：PB0
01：OPINN1
10：AN0
11：PB0

- **PBS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
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 **PBS17~PBS16**：PB7 引脚共用功能选择位
00：PB7
01：CTP2
10：SCK/SCL
11：AN7

Bit 5~4 **PBS15~PBS14**：PB6 引脚共用功能选择位
00：PB6
01：SDI/SDA/URX/UTX
10：PCK
11：AN6

Bit 3~2 **PBS13~PBS12**：PB5 引脚共用功能选择位
00：PB5/PTPI/INT1/FJTR1
01：CTP1B
10：AN5
11：PB5/PTPI/INT1/FJTR1

Bit 1~0 **PBS11~PBS10**：PB4 引脚共用功能选择位
00：PB4/CTCK2/INT0
01：CTP0B
10：AN4
11：PB4/CTCK2/INT0

● **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 引脚共用功能选择位
00：PC3
01：OVP023P1N
10：保留
11：保留

Bit 5~4 **PCS05~PCS04**：PC2 引脚共用功能选择位
00：PC2
01：CTP1
10：SDI/SDA/URX/UTX
11：PCK

Bit 3~2 **PCS03~PCS02**：PC1 引脚共用功能选择位
00：PC1/INT1/FJTR1
01：CTP0
10：$\overline{SCS}$
11：SDO/UTX

Bit 1~0 **PCS01~PCS00**：PC0 引脚共用功能选择位
00：PC0
01：PTP
10：$\overline{SCS}$
11：SDO/UTX

● **PCS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
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 **PCS17~PCS16**：PC7 引脚共用功能选择位
00：PC7
01：PWM2
10：PC7
11：PC7

Bit 5~4 **PCS15~PCS14**：PC6 引脚共用功能选择位
00：PC6
01：PWM3
10：PC6
11：PC6

Bit 3~2 **PCS13~PCS12**：PC5 引脚共用功能选择位
00：PC5/CTCK3/PTCK
01：PCK
10：PHASE
11：PC5/CTCK3/PTCK

Bit 1~0 **PCS11~PCS10**：PC4 引脚共用功能选择位
00：PC4
01：LVDIN
10：CTP3
11：TMCCKO

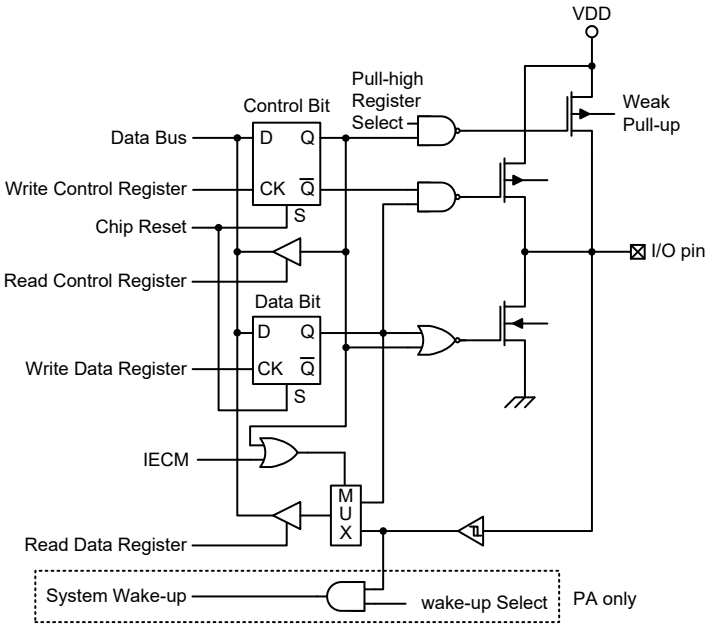
- **IFS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	IFS6	IFS5	IFS4	IFS3	IFS2	IFS1	IFS0
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 **IFS6**：FJTR1 输入源引脚选择位
0：PB5
1：PC1
Bit 5 **IFS5**：INT1 输入源引脚选择位
0：PB5
1：PC1
Bit 4 **IFS4**：INT0 输入源引脚选择位
0：PB4
1：PB3
Bit 3~2 **IFS3~IFS2**：SDI/SDA/URX/UTX 输入源引脚选择位
00：PA2
01：PA2
10：PC2
11：PB6
Bit 1 **IFS1**：SCK/SCL 输入源引脚选择位
0：PB7
1：PA0
Bit 0 **IFS0**：$\overline{SCS}$ 输入源引脚选择位
0：PC0
1：PC1

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示意图。具体输入 / 输出引脚的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 / 输出结构

编程注意事项

在编程中，最先要考虑的是端口的初始化。复位之后，所有的输入 / 输出数据及端口控制寄存器都将被设为逻辑高。所有输入 / 输出引脚默认为输入状态，而其电平则取决于其它相连接电路以及是否选择了上拉电阻。如果端口控制寄存器将某些引脚设定为输出状态，这些输出引脚会有初始高电平输出，除非端口数据寄存器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对应的端口控制寄存器，或使用指令“SET [m].i”及“CLR [m].i”来设定端口控制寄存器中个别的位。注意，当使用这些位控制指令时，系统即将产生一个读 - 修改 - 写的操作。单片机需要先读入整个端口上的数据，修改个别的位，然后重新把这些数据写入到输出端口。

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

定时器模块 – TM

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

这里只介绍各种 TM 的共性，更多详细资料请参考简易型和周期型定时器章节。

简介

该单片机包含多个 TM，每个 TM 都可被归为一个特定的类型，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。此两种类型 TM 的特性和区别见下表。

TM 功能	CTM	PTM
定时 / 计数器	√	√
输入捕捉	—	√
比较匹配输出	√	√
PWM 输出	√	√
单脉冲输出	—	√
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

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 型或 P 型 TM，n 代表具体 TM 编号。该时钟源来自系统时钟 f_{SYS} 或内部高速时钟 f_H 或 f_{SUB} 时钟源或外部 xTCKn 引脚输入信号作为时钟源。xTCKn 引脚的时钟源用于允许外部信号作为 TM 时钟源或者用于事件计数。

TM 中断

每个简易型或周期型 TM 都有两个内部中断，分别是内部比较器 A 或比较器 P，当比较匹配发生时产生 TM 中断。当 TM 中断产生时，计数器清零并改变 TM 输出引脚的状态。

TM 外部引脚

无论哪种类型的 TM 都有一个 TM 输入引脚 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

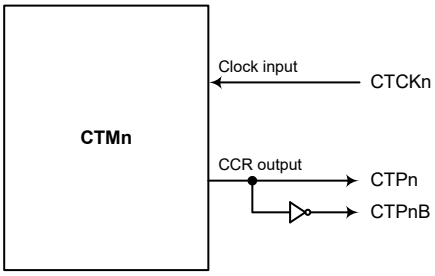
行选择。外部时钟源可通过该引脚来驱动内部 TM。xTCKn 引脚可选择上升沿有效或下降沿有效。PTM 还有一个输入引脚 PTPI。PTPI 或 PTCK 可被选择作为捕捉输入脚，其有效边沿有上升沿、下降沿和双沿，通过设置 PTMC1 寄存器中的 PTIO1~PTIO0 位来选择有效边沿类型。

每个 TM 都有一个或两个输出引脚 xTPn 和 xTPnB，xTPnB 为 xTPn 输出的反相。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 和 xTPnB 输出引脚也被 TM 用来产生 PWM 输出波形。

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事先通过相关引脚共用功能选择寄存器先被设置。更多引脚共用功能选择详见引脚共用功能章节。各 TM 引脚的数量不同，详见下表。

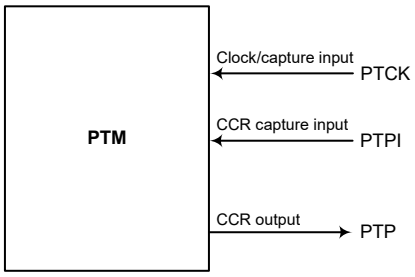
CTMn (n=0~3)		PTM	
输入	输出	输入	输出
CTCK0	CTP0, CTP0B	PTCK, PTPI	PTP
CTCK1	CTP1, CTP1B		
CTCK2	CTP2		
CTCK3	CTP3		

TM 外部引脚



注：CTM2 & CTM3 无 CTPnB 输出。

CTM 功能引脚方框图 (n=0~3)

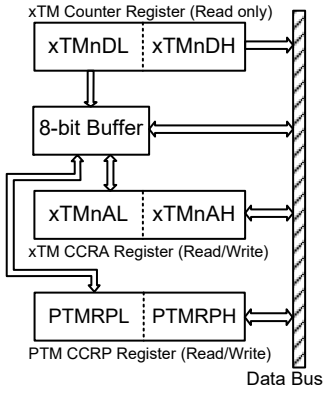


PTM 功能引脚方框图

编程注意事项

TM 计数器寄存器和捕捉 / 比较寄存器 CCRA 和 CCRP，都为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 和 CCRP 低字节寄存器，否则可能导致无法预期的结果。

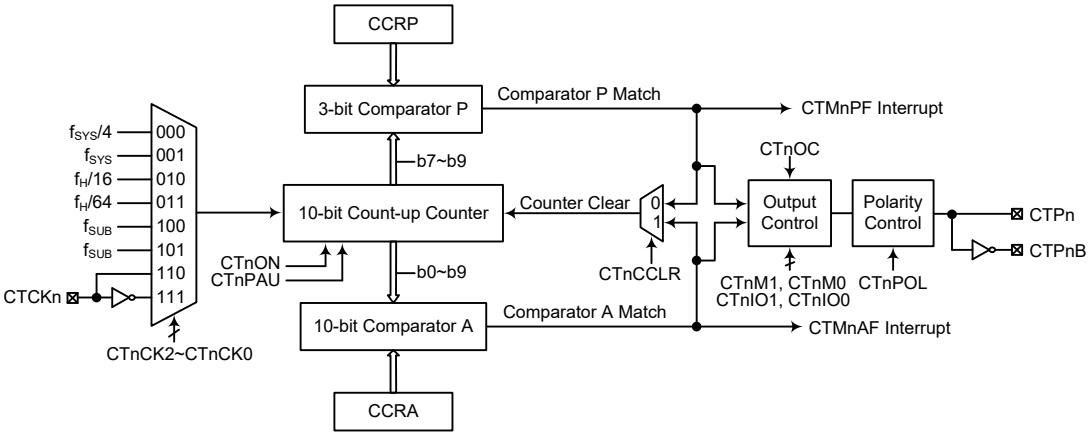


读写流程如下步骤所示：

- 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

简易型 TM 包括三种工作模式，即比较匹配输出，定时 / 事件计数器和 PWM 输出模式。简易型 TM 由一个外部输入脚控制并驱动一个或两个外部输出脚。



注：1. CTMn 外部引脚与其它功能共用引脚，因此在使用 CTMn 之前应该合理配置相关引脚共用功能选择寄存器以确保使能 CTMn 引脚功能。对于 CTCKn 引脚还需设置相应的端口控制寄存器，将该引脚设置为输入口。

2. 对于 CTM2 和 CTM3，不存在 CTPnB 外部输出引脚。

10-bit 简易型 TM 方框图 (n=0~3)

简易型 TM 操作

简易型 TM 核心是一个由用户选择的内部或外部时钟源驱动的 10 位向上计数器，它还包括两个内部比较器即比较器 A 和比较器 P。这两个比较器将计数器的值与 CCRP 和 CCRA 寄存器中的值进行比较。CCRP 是 3 位的，与计数器的高 3 位比较；而 CCRA 是 10 位的，与计数器的所有位比较。

通过应用程序改变 10 位计数器值的唯一方法是使 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，通常情况下会产生 CTM 中断信号。简易型 TM 可工作在不同的模式，可由包括来自外部输入脚的不同时钟源驱动，也可以控制输出脚。所有工作模式的设定都是通过设置相关内部寄存器来实现的。

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。包含一对只读寄存器用来存放 10 位计数器的值，一对读 / 写寄存器存放 10 位 CCRA 的值，剩下两个控制寄存器设置不同的操作和控制模式以及 3 位的 CCRP 值。

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=0~3)

● **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暂停计数器。此位清零可恢复正常计数器操作。在暂停情况下 CTMn 将保持上电状态并继续产生功耗。当此位由低到高转换，计数器将保持其当前值，直到此位再变为低时从此值开始继续计数。

Bit 6~4 **CTnCK2~CTnCK0**：CTMn 计数器时钟选择位
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，细节方面请参考工作模式和系统时钟章节。

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。设置此位为高可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 减少功耗。当此位经由低到高转换时，内部计数器将复位清零，当此位由高到低转换时，内部计数器将保持其当前值，直到此位再次转变为高。
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 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 输出状态未定义。

Bit 5~4 **CTnIO1~CTnIO0**：CTMn 外部引脚 (CTPn) 功能选择位
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：PWM 输出无效状态
01：PWM 输出有效状态
10：PWM 输出
11：未定义
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。这两位值的选择取决于 CTMn 运行在哪种模式下。若 CTMn 处于定时 / 计数器模式，此位无效。
在比较匹配输出模式下，CTnIO1 和 CTnIO0 位决定当从比较器 A 比较匹配输出发生时 CTMn 输出脚如何改变状态。当从比较器 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。只有在 CTMn 关闭时才可改变 CTnIO1 和 CTnIO0 位的值。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，此位无效。在比较匹配输出模式时，比较匹配发生前其决定 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**：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 / 计数器模式时此位无效。

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 CTMn 包括两个比较器即比较器 A 和比较器 P，两者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

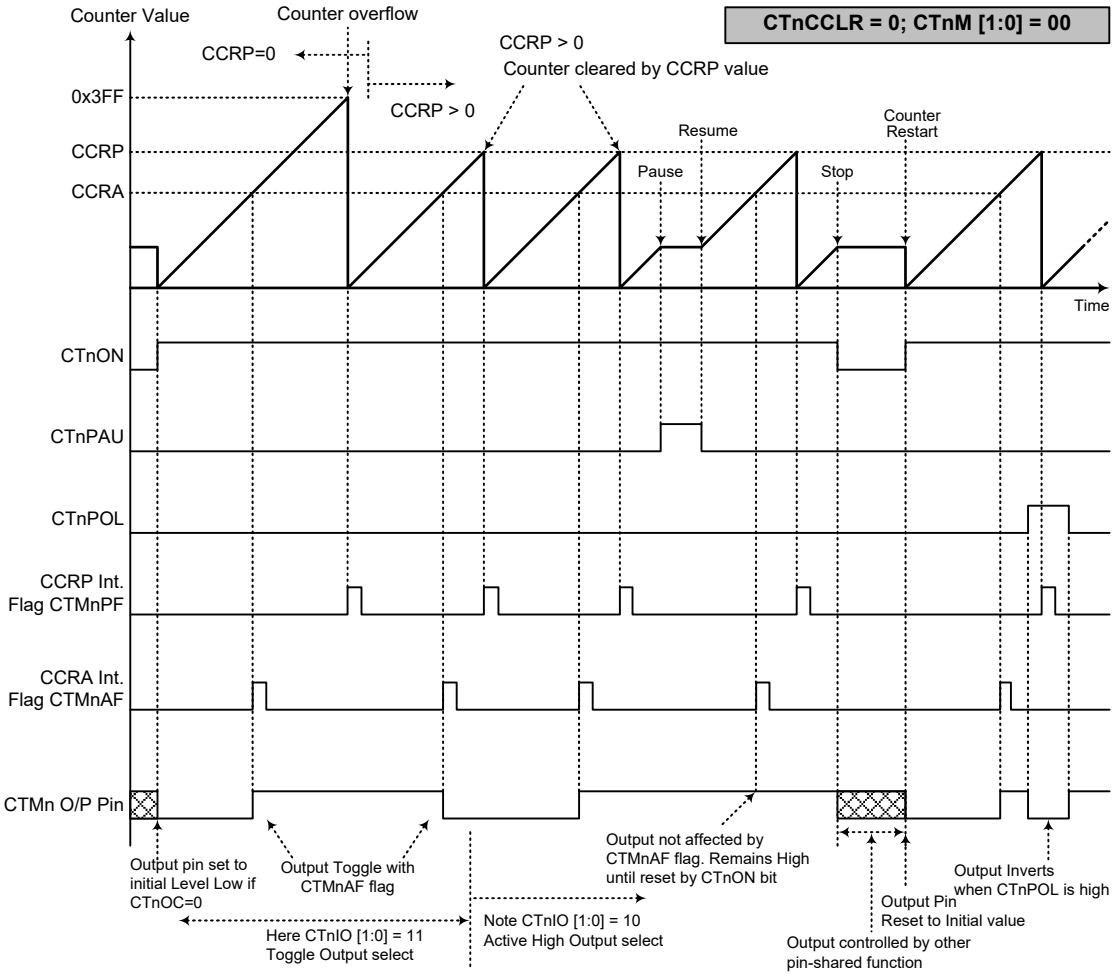
简易型 TM 有三种工作模式，即比较匹配输出模式、PWM 输出模式或定时 / 计数器模式。通过设置 CTMnC1 寄存器的 CTnM1 和 CTnM0 位选择任意模式。

比较匹配输出模式

为使 CTMn 工作在此模式，CTMnC1 寄存器的 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。

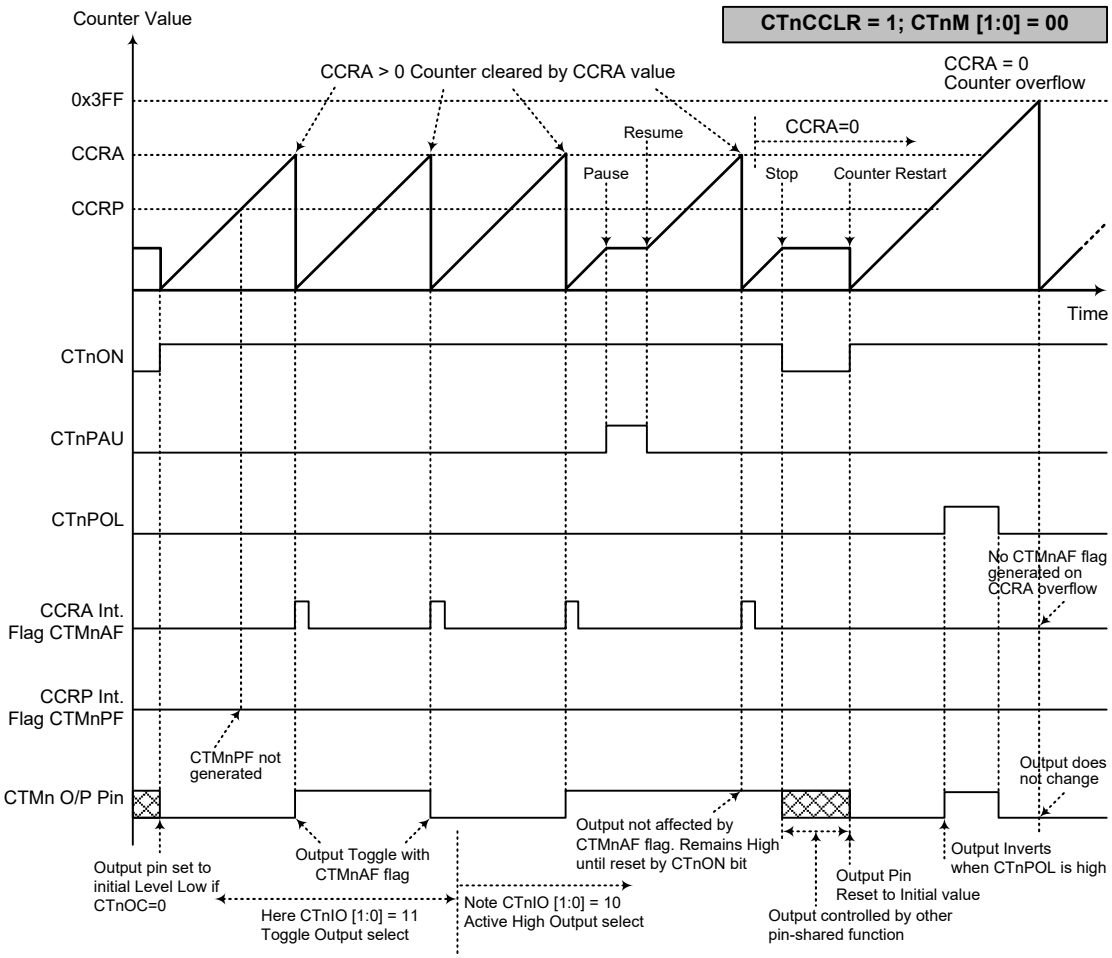
如果 CTMnC1 寄存器的 CTnCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 CTMnAF 中断请求标志产生。所以当 CTnCCLR 为高时，不会产生 CTMnPF 中断请求标志。如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 CTMnAF 请求标志。

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 中断请求标志产生时，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 CTMn 输出脚输出高，低或翻转当前状态。在 CTnON 位由低到高电平的变化后，CTMn 输出脚初始状态为 CTnOC 位所指定的电平。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 (n=0~3)

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



比较匹配输出模式 – CTnCCLR=1 (n=0~3)

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，不会产生 CTMnPF 标志位

定时 / 计数器模式

为使 CTMn 工作在此模式，CTMnC1 寄存器的 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 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 CTMn 工作在此模式，CTMnC1 寄存器的 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，其波形的选择就较为灵活。在 PWM 输出模式中，CTnCCLR 位对 PWM 周期无影响。CCRP 和 CCRA 寄存器都用于控制 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。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，会将对应的 CCRA 或 CCRP 中断标志位置位。CTMnC1 寄存器的 CTnOC 位选择 PWM 波形的极性，CTnIO1 和 CTnIO0 位使能 PWM 输出或强制 CTMn 输出脚为高电平或低电平。CTnPOL 位用于 PWM 输出波形的极性反相控制。

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

CCRP	1~7	0
周期	CCRP×128	1024
占空比	CCRA	

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

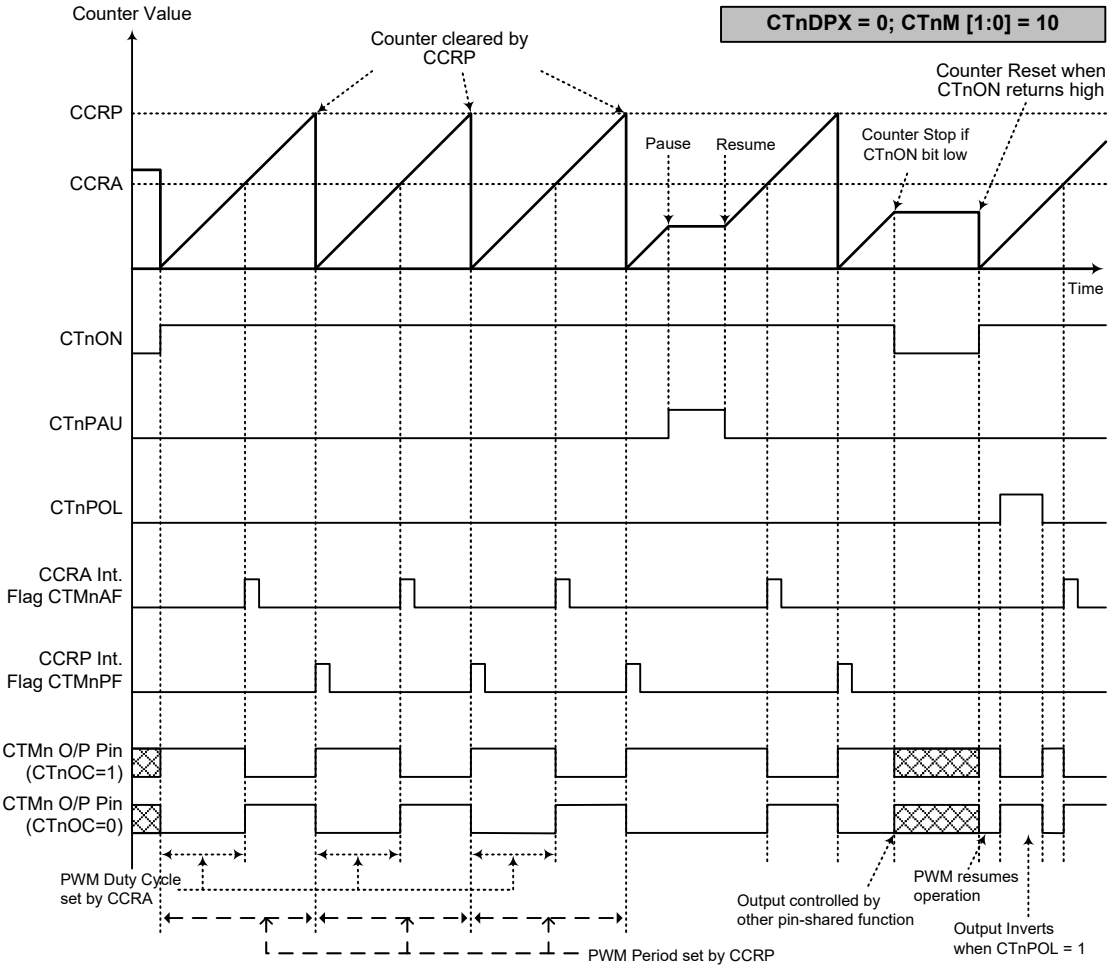
CTMn PWM 输出频率 = (f_{SYS}/4)/(4×128)=f_{SYS}/2048=8kHz，duty=128/(4×128)=25%。

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

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

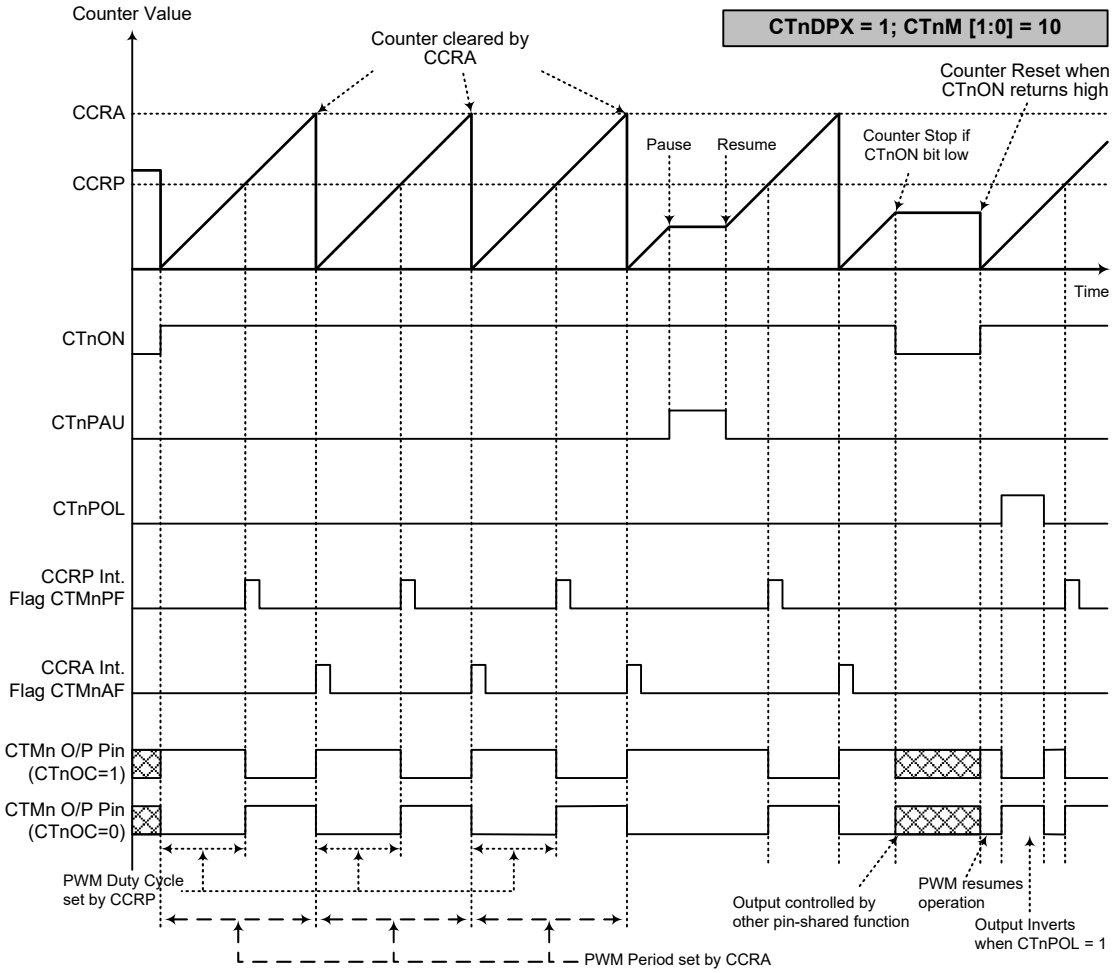
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

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 (n=0~3)

注：1. 这里的 CTnDPX=0 – 计数器由 CCRP 清除
2. 计数器清零并设置 PWM 周期
3. 即使在 CTnIO[1:0]=00 或 01 时，内部 PWM 功能继续运行
4. CTnCCLR 位对 PWM 操作没有影响

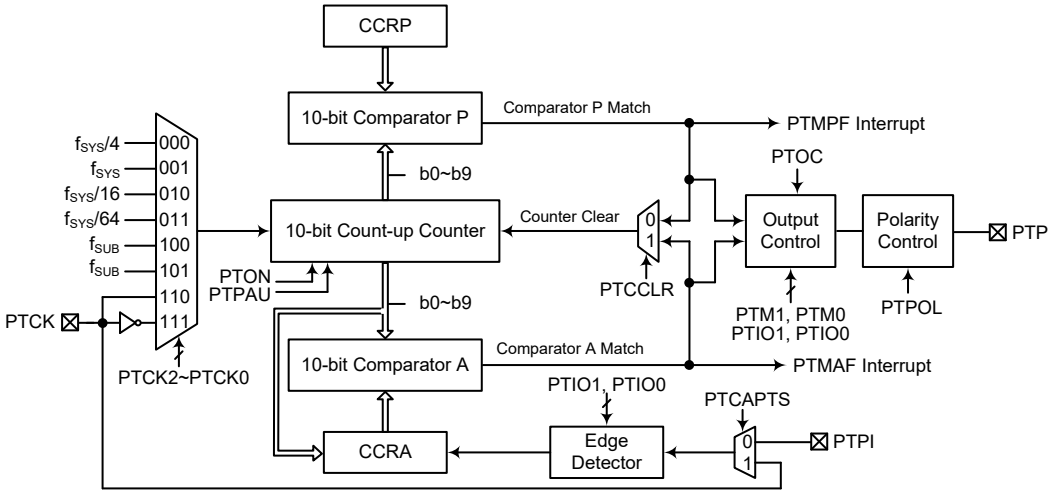


PWM 输出模式 – CTnDPX=1 (n=0~3)

注：1. 这里的 CTnDPX=1 – 计数器由 CCRA 清除
2. 计数器清零并设置 PWM 周期
3. 即使在 CTnIO[1:0]=00 或 01 时，内部 PWM 功能继续运行
4. CTnCCLR 位对 PWM 操作无影响

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。



注：PTM 外部引脚与其它功能共用引脚，因此在使用 PTM 功能前，需确保已通过相关的引脚共用功能选择寄存器选择了 PTM 引脚功能。对于 PTCK 和 PTPI 引脚还需设置相应的端口控制寄存器，将该引脚设置为输入口。

10-bit 周期型 TM 方框图

周期型 TM 操作

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

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

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。在暂停情况下 PTM 保持上电状态并继续产生功耗。当此位由低到高转变时，计数器将保留其当前值，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**：PTM 计数器时钟选择位
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，细节方面请参考工作模式和系统时钟章节。

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**：PTM 工作模式选择位
00：比较匹配输出模式
01：捕捉输入模式
10：PWM 输出模式或单脉冲输出模式
11：定时 / 计数器模式

这两位设置 PTM 需要的工作模式。为了确保操作可靠，在对 PTM1 和 PTM0 位写值之前，应先把 PTM 关掉。在定时 / 计数器模式，PTM 输出状态未定义。

Bit 5~4 **PTIO1~PTIO0**：PTM 外部引脚功能选择位

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：PWM 输出无效状态
- 01：PWM 输出有效状态
- 10：PWM 输出
- 11：单脉冲输出

捕捉输入模式
- 00：在 PTPI 或 PTCK 上升沿输入捕捉
- 01：在 PTPI 或 PTCK 下降沿输入捕捉
- 10：在 PTPI 或 PTCK 双沿输入捕捉
- 11：输入捕捉除能

此两位用于决定在一定条件达到时 PTM 外部引脚如何改变状态。这两位值的选择取决于 PTM 运行在哪种模式下。若 PTM 处于定时 / 计数器模式时，该位无效。

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。只有在 CTMn 关闭时才可改变 CTnIO1 和 CTnIO0 位的值。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，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：来自 PTPI 引脚
- 1：来自 PTCK 引脚

如果 PTCK 用作捕捉输入源，则不能将其再选作 PTM 的时钟源。

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 包括两个比较器即比较器 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	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 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	—	—	—	—	—	—	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 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

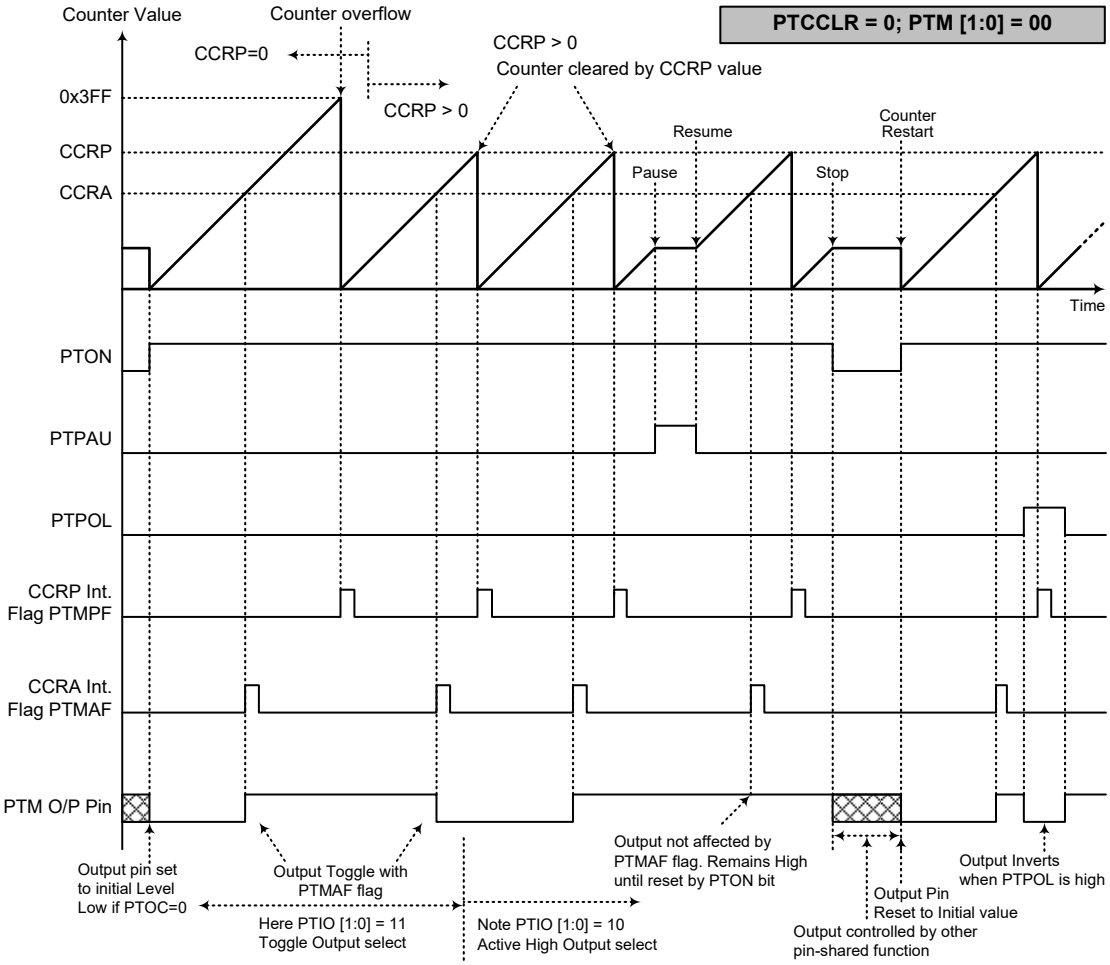
周期型 TM 有五种工作模式，即比较匹配输出模式、PWM 输出模式、单脉冲输出模式、捕捉输入模式或定时 / 计数器模式。通过设置 PTMC1 寄存器的 PTM1 和 PTM0 位选择任意模式。

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。

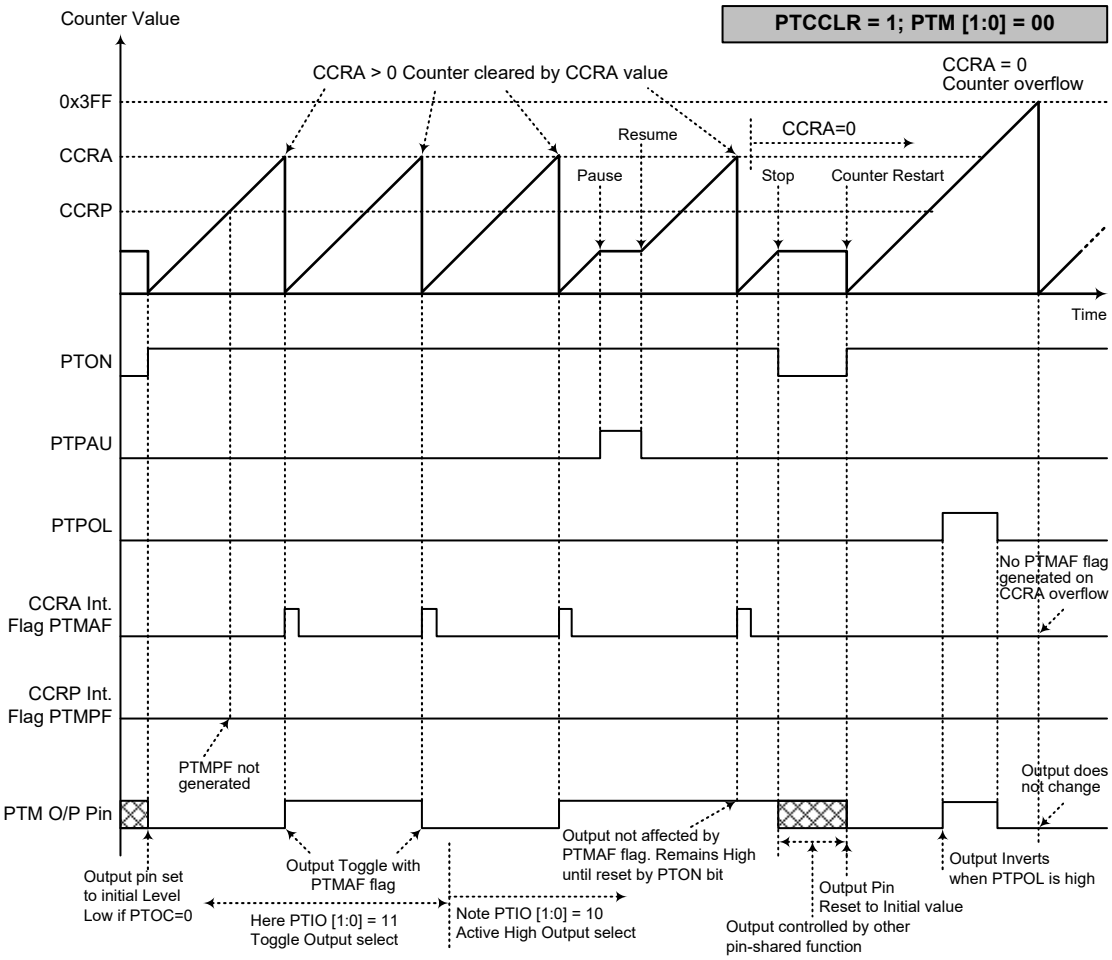
如果 PTMC1 寄存器的 PTCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 PTMAF 中断请求标志产生。所以当 PTCCLR 为高时，不会产生 PTMPF 中断请求标志。在比较匹配输出模式中，CCRA 寄存器值不能设为“0”。如果 CCRA 被清零，当计数值达到最大 3FFH 时，计数器溢出，而此时不产生 PTMAF 请求标志。

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。在 PTON 位由低到高后，PTM 输出脚初始状态为 PTOC 位所指定的电平。注意，若 PTIO1 和 PTIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 – 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



比较匹配输出模式 – PTCCLR=1

注：1. PTCCLR=1，比较器 A 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”。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，其波形的选择就较为灵活。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，会将对应的 CCRA 或 CCRP 中断标志位置起。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。

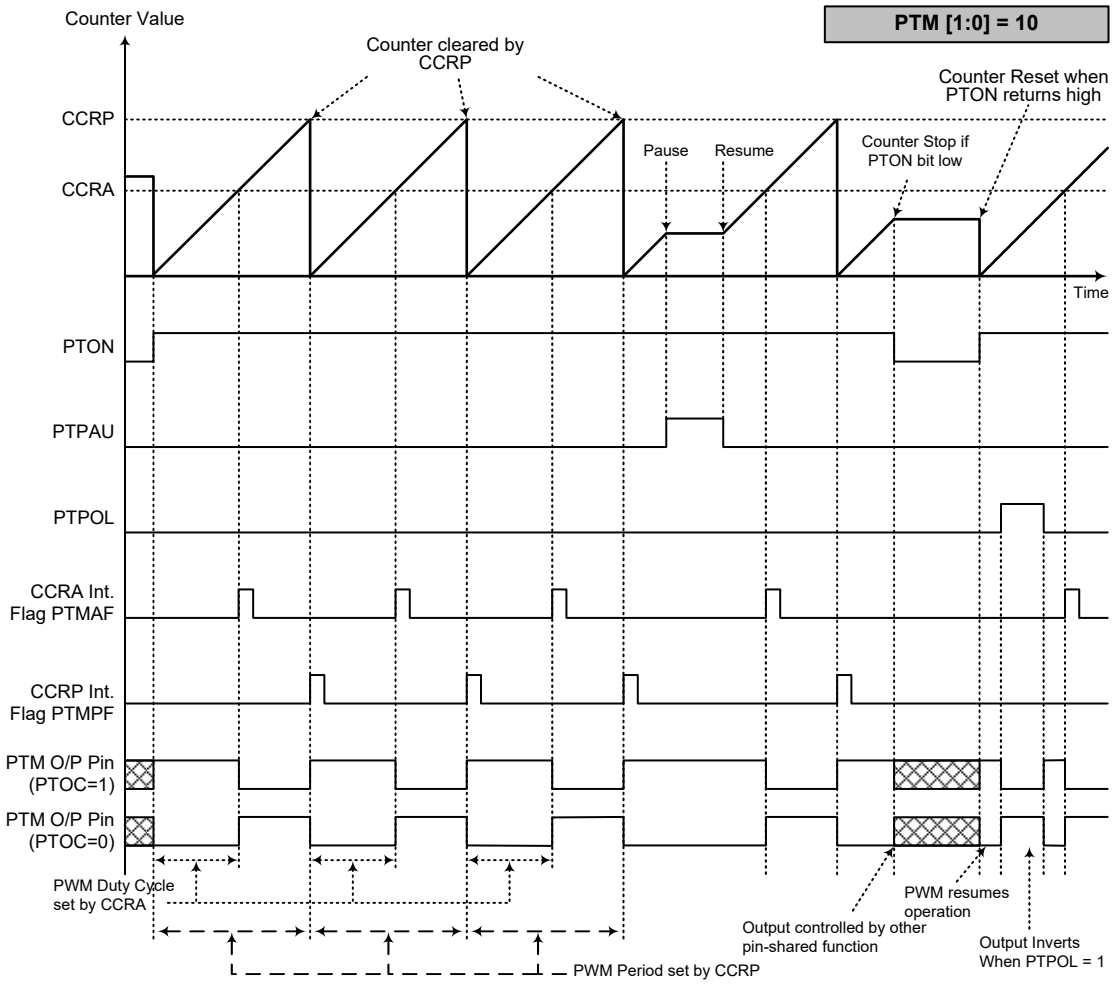
- **10-bit PTM，PWM 输出模式，边沿对齐模式**

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=8kHz，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

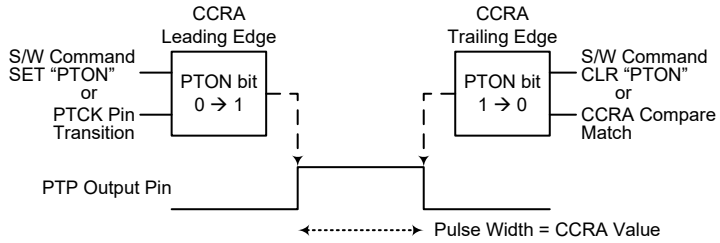
注：1. 计数器由 CCRP 清除
2. 计数器清零并设置 PWM 周期
3. 即使在 PTIO[1:0]=00 或 01 时，内部 PWM 功能继续运行
4. PTCCLR 位对 PWM 操作没有影响

单脉冲输出模式

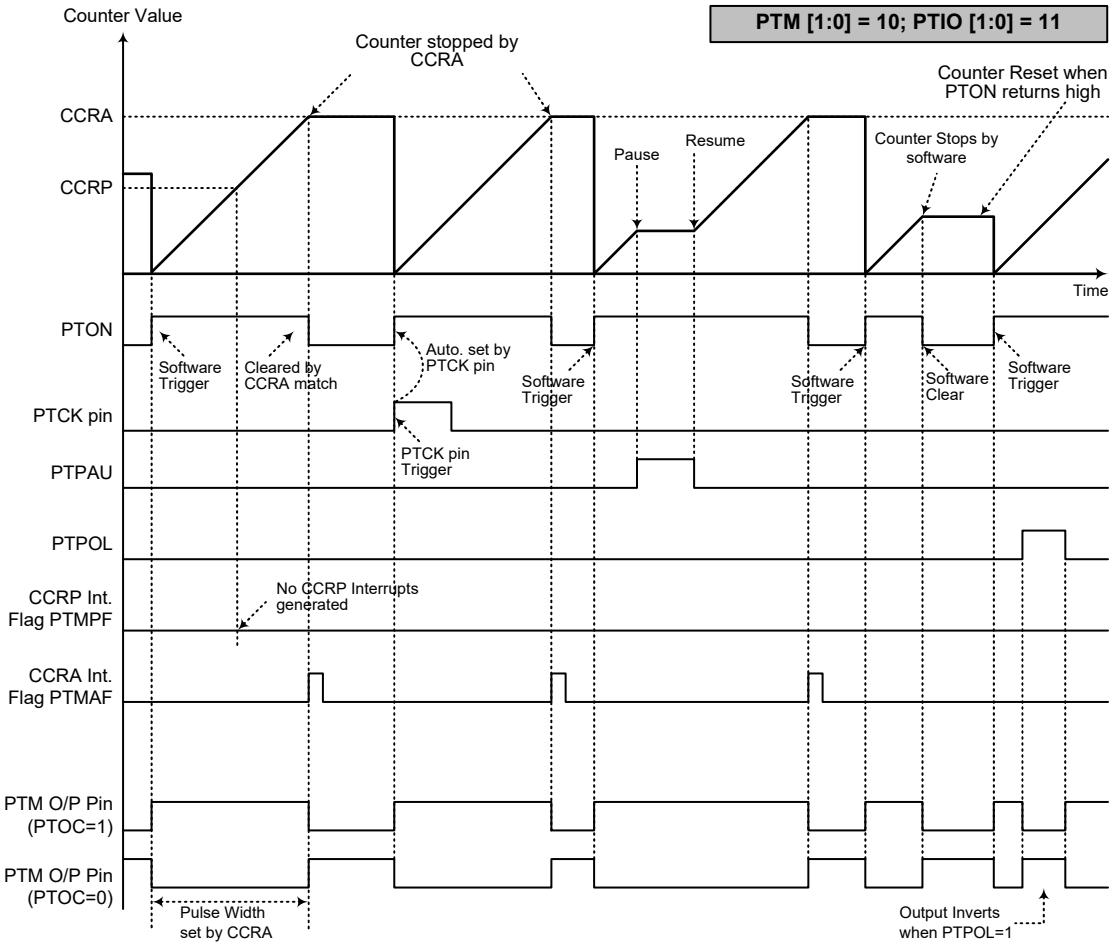
为使 PTM 工作在此模式，PTMC1 寄存器中的 PTM1 和 PTM0 位需要设置为“10”，并且相应的 PTIO1 和 PTIO0 需要设置为“11”。正如模式名所言，单脉冲输出模式，在 PTM 输出脚将产生一个脉冲输出。

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。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

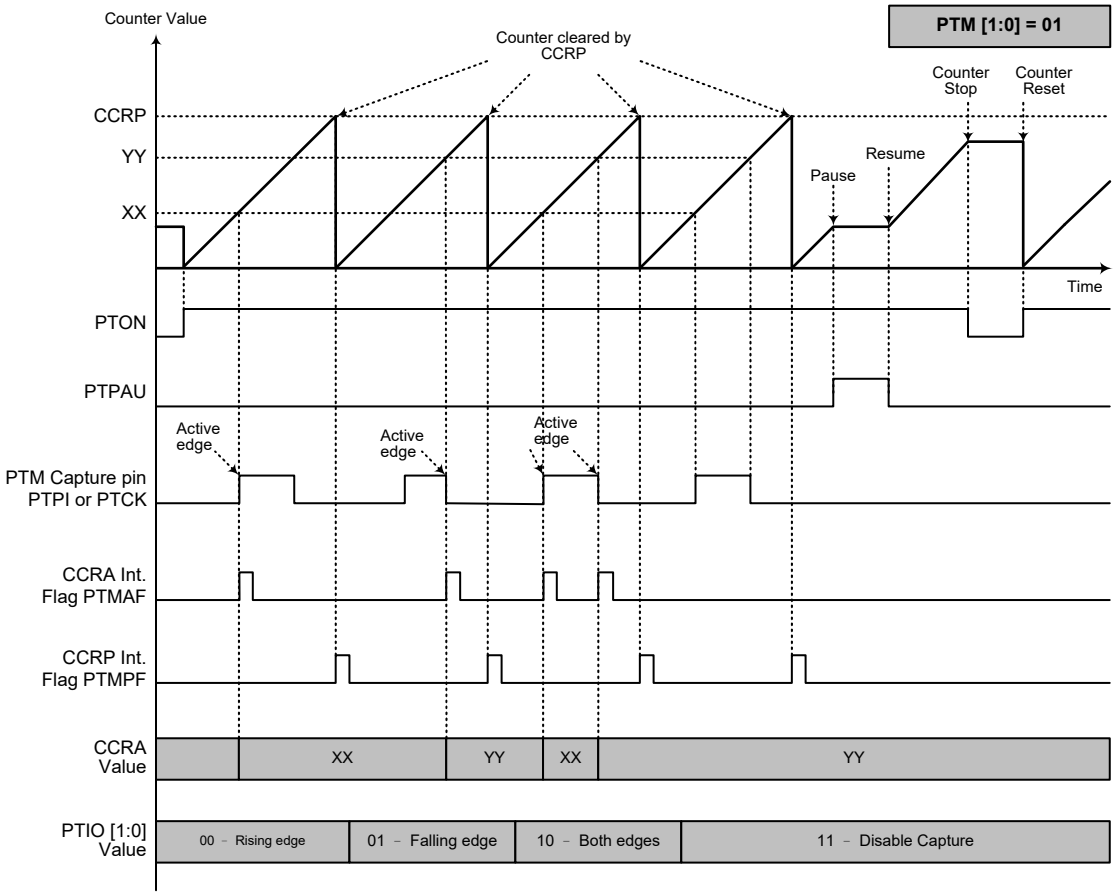
注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 PTCK 脚或设置 PTON 位为高来触发脉冲
4. PTCK 脚有效边沿会自动置位 PTON 位
5. 单脉冲输出模式中，PTIO[1:0] 需设置为“11”，且不能更改。

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。捕捉输入源可以来自 PTPI 或 PTCK 引脚输入信号，通过设置 PTMC1 寄存器的 PTCAPTS 位选择。PTMC1 寄存器的 PTIO1 和 PTIO0 位用于选择输入有效边沿类型，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 PTPI 或 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。之后无论 PTPI 或 PTCK 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；通过这种方式 CCRP 的值可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 PTPI 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 PTPI 或 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。PTCCLR、PTOC 和 PTPOL 位在此模式中未使用。

有几点注意事项须留意。如果 PTCK 用作捕捉输入源，则不能将其再选作 PTM 的时钟源。如果捕捉脉宽小于 2 个定时器时钟周期，则可能会被硬件忽略。当计数器的值被有效捕捉边沿锁存到 CCRA 寄存器后，再过 0.5 个定时器时钟周期，PTMAF 标志位将被置高。从接收到有效捕捉边沿，到开始将计数器值锁存到 CCRA 寄存器的动作，这之间的延迟时间小于 1.5 个定时器时钟周期。



捕捉输入模式

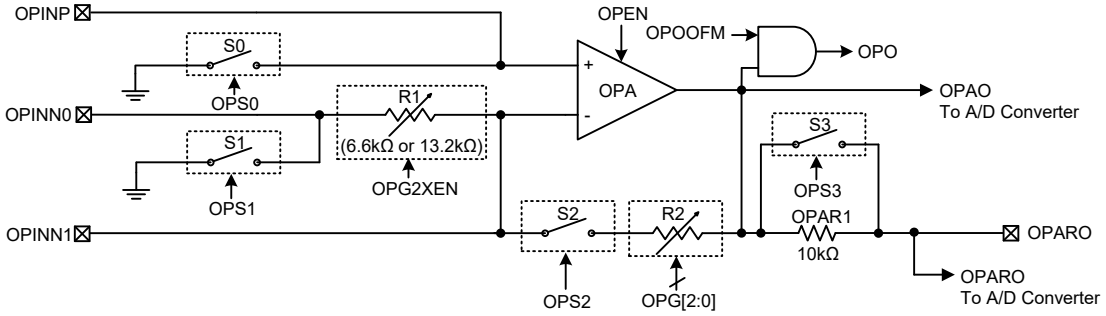
注：1. PTM[1:0]=01 并通过 PTIO[1:0] 位设置有效边沿
2. PTM 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中
3. PTCCLR 位未使用
4. 无输出功能 – PTOC 和 PTPOL 位未使用
5. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大
6. 捕捉输入模式需在有 PTM 计数时钟的情况下才可使用

运算放大器

此款单片机内部集成了运算放大器功能，可对输入端间微小的信号进行多倍放大。运算放大器完全内部集成，故大大减少了对外部元器件的需求。

运算放大器操作

运算放大器可根据用户需求通过内部寄存器设置，对信号进行放大。放大增益由 OPG2~OPG0 位进行选择。OPG2XEN 位可对 OPG2~OPG0 位所选择的增益值再进行翻倍。放大后的信号可通过 OPARO 引脚输出，也可连接到 A/D 转换器的内部通道进行测量。运算放大器的功能框图如下所示。



运算放大器电路

OPA 寄存器描述

运算放大器可工作在正常模式或输入失调校准模式，都可通过相关寄存器进行设置。OPC0 寄存器用于控制电路中各开关的 On/Off。OPC1 寄存器用于设置 OPA 功能的使能 / 除能以及选择增益值。OPC1 寄存器中的 OPO 位搭配 OPOCAL 寄存器用于实现输入失调校准功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
OPC0	—	—	—	—	OPS3	OPS2	OPS1	OPS0
OPC1	OPEN	—	—	OPO	OPG2XEN	OPG2	OPG1	OPG0
OPOCAL	OPOOFM	OPORSP	OPOOF5	OPOOF4	OPOOF3	OPOOF2	OPOOF1	OPOOF0

OPA 寄存器列表

- **OPC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	OPS3	OPS2	OPS1	OPS0
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　**OPS3**：OPA 开关 S3 on/off 控制
0：Off
1：On

Bit 2　**OPS2**：OPA 开关 S2 on/off 控制
0：Off
1：On

Bit 1 **OPS1**：OPA 开关 S1 on/off 控制
0：Off
1：On

Bit 0 **OPS0**：OPA 开关 S0 on/off 控制
0：Off
1：On

● OPC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	OPEN	—	—	OPO	OPG2XEN	OPG2	OPG1	OPG0
R/W	R/W	—	—	R	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7 **OPEN**：OPA 功能使能控制位
0：除能
1：使能

Bit 6~5 未定义，读为“0”

Bit 4 **OPO**：OPA 输出状态位 (正逻辑)
当设置 OPOCAL 寄存器中的 OPOOFM 为 1 时，OPO 位用于指示 OPA 输出状态，具体使用参考失调电压校准章节。此位为只读位。

Bit 3 **OPG2XEN**：R2/R1 比值翻倍控制位
0：除能 (R1=13.2kΩ)
1：使能 (R1=6.6kΩ)

Bit 2~0 **OPG2~OPG0**：R2/R1 比值选择位
000：R2/R1=2 (OPG2XEN=0); R2/R1=4 (OPG2XEN=1)
001：R2/R1=5 (OPG2XEN=0); R2/R1=10 (OPG2XEN=1)
010：R2/R1=6 (OPG2XEN=0); R2/R1=12 (OPG2XEN=1)
011：R2/R1=7 (OPG2XEN=0); R2/R1=14 (OPG2XEN=1)
100：R2/R1=20 (OPG2XEN=0); R2/R1=40 (OPG2XEN=1)
101：R2/R1=25 (OPG2XEN=0); R2/R1=50 (OPG2XEN=1)
110：R2/R1=30 (OPG2XEN=0); R2/R1=60 (OPG2XEN=1)
111：R2/R1=35 (OPG2XEN=0); R2/R1=70 (OPG2XEN=1)
因这些位对增益的选择是通过电阻 R1 和 R2 来实现。因此在使用这些位选择增益时，需确保开关 S1 为 ON，或者选择 OPINN0 输入同时 S2 开关应为 ON。这样才可保证增益精准度。

● OPOCAL 寄存器

Bit	7	6	5	4	3	2	1	0
Name	OPOOFM	OPORSP	OPOOF5	OPOOF4	OPOOF3	OPOOF2	OPOOF1	OPOOF0
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 **OPOOFM**：输入失调电压校准模式 / 运算放大器模式选择位
0：运算放大器模式
1：输入失调电压校准模式

Bit 6 **OPORSP**：运算放大器输入失调电压校准参考选择位
0：反相输入端作为参考输入
1：同相输入端作为参考输入

Bit 5~0 **OPOOF5~OPOOF0**：运算放大器输入失调电压校准控制位

输入电压范围

为了操作的灵活性，在不同的PGA工作模式下，输入电压可以为正也可以为负。同相输入或反相输入电压的 PGA 输出基于不同的公式计算如下：

- 输入电压 $V_{IN} > 0$，PGA 工作在同相比例放大模式，PGA 输出电压可由以下公式获得：

$$V_{OPGA}=(1+R_2/R_1)\times V_{IN}$$

- 当 S0 和 S2 开关为 ON，S1 和 S3 开关为 OFF，输入结点为 OPINN0。输入电压 $0 > V_{IN} > -0.2V$ 时，PGA 工作在反相比例放大模式，PGA 输出电压可由以下公式获得：

$$V_{OPGA}=-(R_2/R_1)\times V_{IN}$$

注意：反相放大时，输入电压不能低于 -0.2V，否则会产生漏电流。

运算放大器输入失调校准

该运算放大器提供输入失调校准功能。校准后的数据储存在 OPOOF[5:0] 位。OPOOFM 为校准模式选择位。OPORSP 可选择校准时参考电压来自正端或负端输入，OPINP 和 OPINN1 为 OPA 的两个输入端，OPAO 为 OPA 模拟输出电压。OPA 数字输出位 OPO，用于 OPA 校准模式。

输入失调校准

需注意的是因为 OPA 输入引脚与其它功能共用引脚，因此在开始 OPA 校准之前，需确保已通过相关的引脚共用功能选择寄存器选择了 OPA 引脚功能。运算放大器输入失调校准步骤如下。

步骤 1. 设置 OPOOFM=1 以及 OPORSP=1，运算放大器工作于失调校准模式。为了确保校准后的 V_{OS} 尽可能小，校准模式下的输入参考电压应该跟正常模式下的输入直流工作电压相同。

步骤 2. 设置 OPOOF[5:0]=000000，请延迟一段时间后再读取 OPO 位。

步骤 3. 使 OPOOF[5:0]=OPOOF[5:0]+1，请延迟一段时间后再读取 OPO 位。

如果 OPO 位状态不改变，然后重复步骤 3 直到 OPO 位状态改变。

如果 OPO 位状态改变，记录此时的 OPOOF[5:0] 值为 V_{OS1} 然后转到步骤 4。

步骤 4. 设置 OPOOF[5:0]=111111，请延迟一段时间后再读取 OPO 位。

步骤 5. 使 OPOOF[5:0]=OPOOF[5:0]-1，请延迟一段时间后再读取 OPO 位。

如果 OPO 位状态不改变，然后重复步骤 5，直到 OPO 位状态改变。

如果 OPO 位状态改变，记录此时的 OPOOF[5:0] 值为 V_{OS2} 然后转到步骤 6。

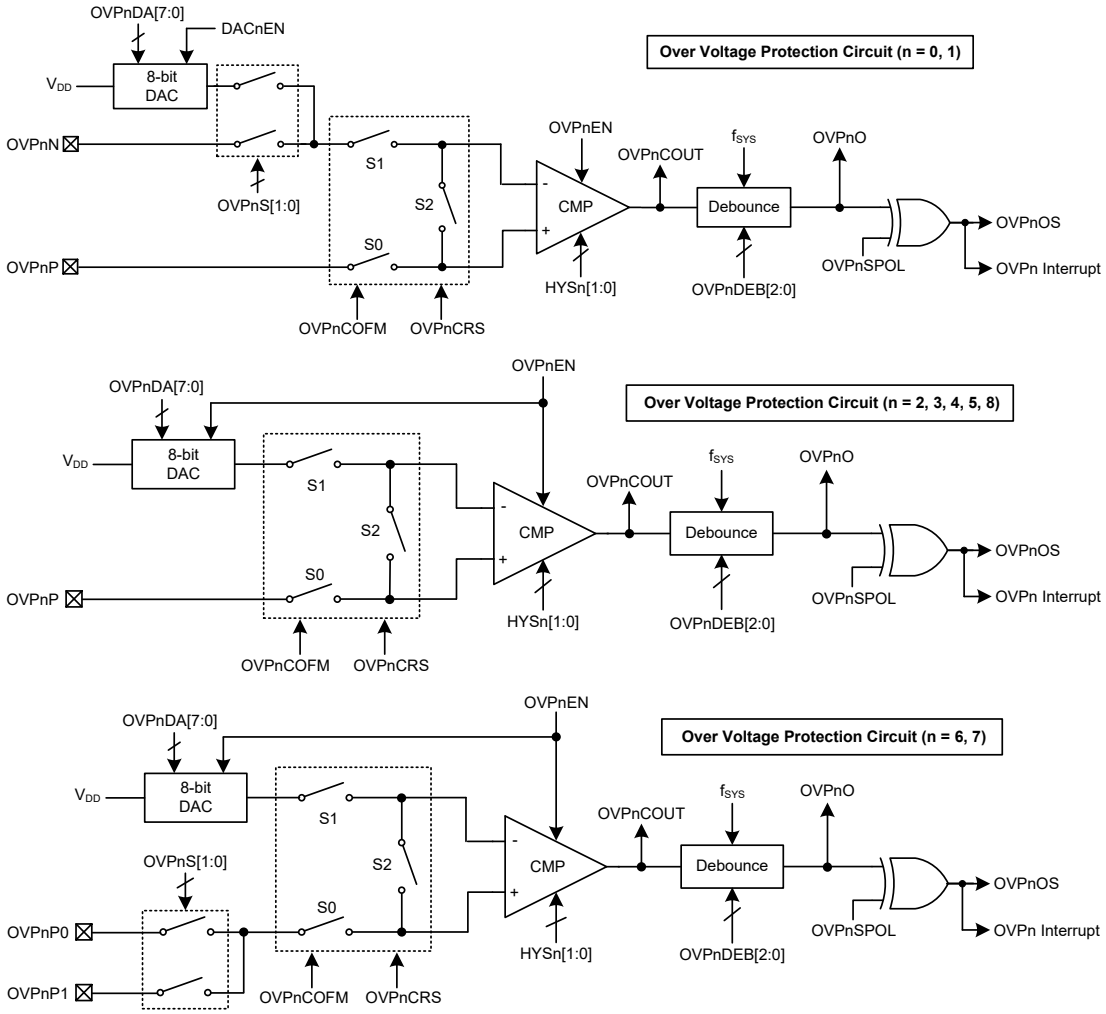
步骤 6. 将运算放大器输入失调校准值 V_{OS} 存入 OPOOF[5:0] 位中，校准结束。

$V_{OS}=(V_{OS1}+V_{OS2})/2$。

如果 $(V_{OS1}+V_{OS2})/2$ 不是整数，舍掉小数。

过电压保护 – OVP

该单片机内建九组过电压保护电路，OVP0~OVP8，可针对不同应用提供保护并产生输出信号。为防止电压超过一指定值，可将此电压输入到 OVPn 一端，然后与 8-bit DAC 产生的参考电压进行比较，或者与另一端输入的电压进行比较。当有超过预设电压的情况发生时，OVPn 的输出会发生翻转，依此产生 OVPnO 信号，若中断使能，也可产生 OVPn 中断。通过 OVPnSPOL 控制位可设置 OVPnOS 信号为 OVPnO 的同相或者反相信号。



注：1. 因 OVPn 功能引脚与其它功能共用引脚，因此在使能 OVPn 功能前，需确保已通过引脚共用功能选择位选择了 OVPn 引脚功能。

2. 上述方框图中的 OVP1P、OVP4P、OVP5P 和 OVP0N 连接到同一个外部引脚，引脚名为 OVP145P0N。OVP0P、OVP2P、OVP3P 以及 OVP1N 连接到同一个外部引脚，引脚名 OVP023P1N。OVP6P0 和 OVP7P1 连接到同一个外部引脚，引脚名为 OVP6P07P1。OVP6P1 和 OVP7P0 连接到同一个外部引脚，引脚名为 OVP6P17P0。具体可参考引脚说明章节。

3. 上图中 S0、S1 和 S2 开关的 ON/OFF 由 OVPnCOFM 和 OVPnCRS 位控制，控制逻辑如下表所示：

OVPnCOFM	OVPnCRS	S0	S1	S2
0	x	ON	ON	OFF
1	0	OFF	ON	ON

OVPnCOFM	OVPnCRS	S0	S1	S2
1	1	ON	OFF	ON

“x”代表不相关

过电压保护寄存器

OVPn 功能的所有操作是通过一系列寄存器控制的。因 OVP0~OVP8 控制电路结构并非完全一样，一些寄存器的位也不同。如 OVPnEN 位的控制，DACnEN 位和 OVPnS[1:0] 位等。详细资料请参考寄存器定义部分。

寄存器名称	位							
	7	6	5	4	3	2	1	0
OVPnC0	OVPnO	OVPnSPOL	OVPnEN	—	—	OVPnDEB2	OVPnDEB1	OVPnDEB0
OVPnC1	OVPnCOUT	OVPnCOFM	OVPnCRS	OVPnCOF4	OVPnCOF3	OVPnCOF2	OVPnCOF1	OVPnCOF0
OVPnC2	DACnEN	—	—	—	HYSn1	HYSn0	OVPnS1	OVPnS0
OVPnDA	D7	D6	D5	D4	D3	D2	D1	D0

OVPn 寄存器列表 (n=0, 1)

寄存器名称	位							
	7	6	5	4	3	2	1	0
OVPnC0	OVPnO	OVPnSPOL	OVPnEN	—	—	OVPnDEB2	OVPnDEB1	OVPnDEB0
OVPnC1	OVPnCOUT	OVPnCOFM	OVPnCRS	OVPnCOF4	OVPnCOF3	OVPnCOF2	OVPnCOF1	OVPnCOF0
OVPnC2	—	—	—	—	HYSn1	HYSn0	—	—
OVPnDA	D7	D6	D5	D4	D3	D2	D1	D0

OVPn 寄存器列表 (n=2, 3, 4, 5, 8)

寄存器名称	位							
	7	6	5	4	3	2	1	0
OVPnC0	OVPnO	OVPnSPOL	OVPnEN	—	—	OVPnDEB2	OVPnDEB1	OVPnDEB0
OVPnC1	OVPnCOUT	OVPnCOFM	OVPnCRS	OVPnCOF4	OVPnCOF3	OVPnCOF2	OVPnCOF1	OVPnCOF0
OVPnC2	—	—	—	—	HYSn1	HYSn0	OVPnS1	OVPnS0
OVPnDA	D7	D6	D5	D4	D3	D2	D1	D0

OVPn 寄存器列表 (n=6, 7)

- **OVPnC0 寄存器 (n=0~8)**

Bit	7	6	5	4	3	2	1	0
Name	OVPnO	OVPnSPOL	OVPnEN	—	—	OVPnDEB2	OVPnDEB1	OVPnDEB0
R/W	R	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7 **OVPnO**：OVPn 比较器去抖后输出
0：正端输入电压 < 负端输入电压
1：正端输入电压 > 负端输入电压

Bit 6 **OVPnSPOL**：OVPn 去抖输出信号极性控制位
0：同相
1：反相
该位决定当 OVPnO 位从低变高或从高变低时，相应的 OVPnOS 输出信号为从低变高还是从高变低。
注意，无论 OVPnEN 位为何值，该位都有效。

Bit 5 **OVPnEN**：OVPn 功能使能控制位
0：除能
1：使能

若设置该位为零，除能 OVPn 功能，不会再产生耗电。对于 OVP2~OVP8，清零 OVPnEN 位，D/A 转换器及比较器都将被关闭。对于 OVP0 和 OVP1，清零 OVPnEN 位，只将电路中的比较器关闭， 而 D/A 转换器的开 / 关则由专门的 DACnEN 位进行控制。

当 OVPnEN 位为零时，OVPnCOUT 输出为下拉状态，因 OVPnOS 会受 OVPnSPOL 位影响。若 OVPnSPOL=0，OVPnOS 输出也为下拉状态。若 OVPnSPOL=1，则 OVPnOS 输出信号为高。

Bit 4~3　未定义，读为“0”

Bit 2~0　**OVPnDEB2~OVPnDEB0**：OVPn 比较器去抖动时间设置

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

注：$t_{DEB}=1/f_{SYS}$

● OVPnC1 寄存器 (n=0~8)

Bit	7	6	5	4	3	2	1	0
Name	OVPnCOUT	OVPnCOFM	OVPnCRS	OVPnCOF4	OVPnCOF3	OVPnCOF2	OVPnCOF1	OVPnCOF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7　**OVPnCOUT**：OVPn 比较器输出 (去抖前)

0：正端输入端电压 < 负端输入端电压
1：正端输入端电压 > 负端输入端电压

当 OVPnEN 位为 0 时，OVPnCOUT 输出为下拉状态。

Bit 6　**OVPnCOFM**：OVPn 比较器模式选择位

0：正常比较器模式
1：输入失调电压校准模式

Bit 5　**OVPnCRS**：OVPn 比较器输入失调电压校准参考选择位

0：负输入端作为参考输入
1：正输入端作为参考输入

Bit 4~0　**OVPnCOF4~OVPnCOF0**：OVPn 比较器输入失调电压校准控制位

这 5 位用来执行比较器的输入失调校准操作，并重新储存 OVPn 比较器的输入失调校准值。具体操作请参考“比较器输入失调校准”章节。

● OVPnC2 寄存器 (n=0, 1)

Bit	7	6	5	4	3	2	1	0
Name	DACnEN	—	—	—	HYSn1	HYSn0	OVPnS1	OVPnS0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7　**DACnEN**：OVPn D/A 转换器使能控制位

0：除能
1：使能

Bit 6~4　未定义，读为“0”

Bit 3~2　**HYSn1~HYSn0**：OVPn 比较器迟滞电压窗口控制位

具体参考“过电压保护电气特性”表章节。

Bit 1~0　**OVPnS1~OVPnS0**：OVPn 负端输入选择位

00：不选择
01：OVPnN

10：DAC 输出
11：不选择

● **OVPnC2 寄存器 (n=2, 3, 4, 5, 8)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HYSn1	HYSn0	—	—
R/W	—	—	—	—	R/W	R/W	—	—
POR	—	—	—	—	0	0	—	—

Bit 7~4　未定义，读为“0”
Bit 3~2　**HYSn1~HYSn0**：OVPn 比较器迟滞电压窗口控制位
具体参考“过电压保护电气特性”表章节
Bit 1~0　未定义，读为“0”

● **OVP6C2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HYS61	HYS60	OVP6S1	OVP6S0
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　**HYS61~HYS60**：OVP6 比较器迟滞电压窗口控制位
具体参考“过电压保护电气特性”表章节。
Bit 1~0　**OVP6S1~OVP6S0**：OVP6 输入选择位
00：不选择
01：OVP6P07P1
10：OVP6P17P0
11：不选择

● **OVP7C2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HYS71	HYS70	OVP7S1	OVP7S0
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　**HYS71~HYS70**：OVP7 比较器迟滞电压窗口控制位
具体参考“过电压保护电气特性”表章节。
Bit 1~0　**OVP7S1~OVP7S0**：OVP7 输入选择位
00：不选择
01：OVP6P17P0
10：OVP6P07P1
11：不选择

● **OVPnDA 寄存器 (n=0~8)**

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**：OVPn D/A 转换器输出电压控制位
DAC $V_{OUT}=(V_{DD}/256)\times OVPnDA[7:0]$

比较器输入失调校准

需注意的是因为 OVPn 输入引脚与其它功能共用引脚，因此在开始 OVPn 比较器校准之前，需确保已通过相关的引脚共用功能选择寄存器选择了 OVPn 引脚功能。在失调校准前，需将 HYSn[1:0] 设置为 00，以确保迟滞电压为 0。OVPn 比较器失调电压校准步骤如下。

步骤 1. 设置 OVPnCOFM=1 以及 OVPnCRS=0，OVPn 则工作于比较器失调校准模式。S1 和 S2 为 ON。

步骤 2. 设置 OVPnDA[7:0]，为了确保校准后的 V_{OS} 尽可能小，校准模式下的输入参考电压应该跟正常模式下的输入直流工作电压相同。

步骤 3. 设置 OVPnCOF[4:0]=00000，请延迟一段时间后再读取 OVPnCOUT 位。

步骤 4. 使 OVPnCOF[4:0]=OVPnCOF[4:0]+1，请延迟一段时间后再读取 OVPnCOUT 位。

如果 OVPnCOUT 位状态不改变，然后重复步骤 4 直到 OVPnCOUT 位状态改变。

如果 OVPnCOUT 位状态改变，记录此时的 OVPnCOF[4:0] 值为 V_{OS1} 然后转到步骤 5。

步骤 5. 设置 OVPnCOF[4:0]=11111，请延迟一段时间后再读取 OVPnCOUT 位。

步骤 6. 使 OVPnCOF[4:0]=OVPnCOF[4:0]-1，请延迟一段时间后再读取 OVPnCOUT 位。

如果 OVPnCOUT 位状态不改变，然后重复步骤 6，直到 OVPnCOUT 位状态改变。

如果 OVPnCOUT 位状态改变，记录此时的 OVPnCOF[4:0] 值为 V_{OS2} 然后转到步骤 7。

步骤 7. 将 OVPn 比较器输入失调校准值 $V_{OS}=(V_{OS1}+V_{OS2})/2$ 存入 OVPnCOF[4:0] 位中，校准结束。

如果 $(V_{OS1}+V_{OS2})/2$ 不是整数，舍弃小数。

注：n=0, 1 时，若要选择 DAC 进行校准，须将 DACnEN 设置为 1，且 OVPnS[1:0] 需设置为 10。

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

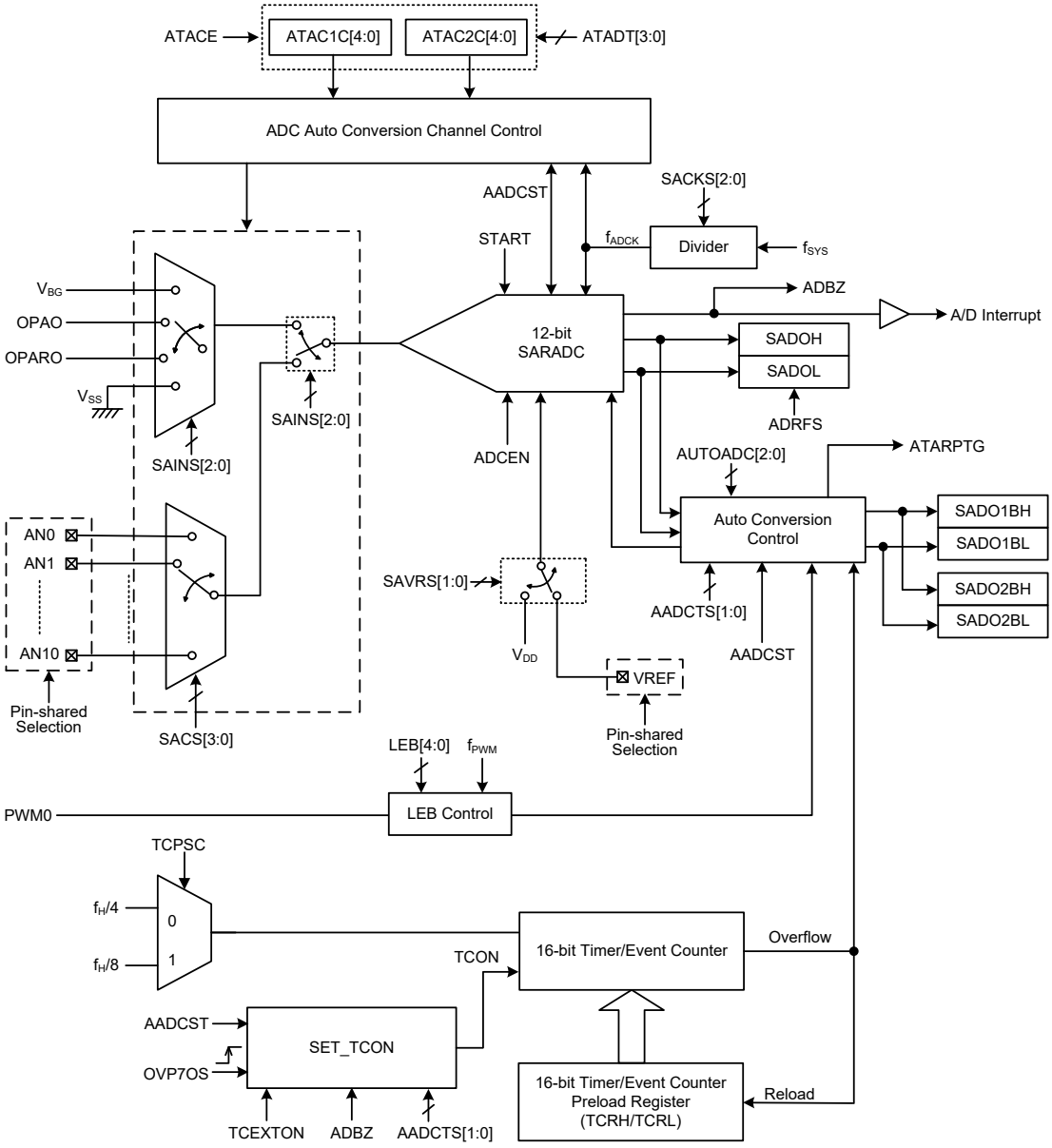
此单片机包含一个多通道的 A/D 转换器，它可以直接接入外部模拟信号 (来自传感器或其它控制信号) 或内部模拟信号 (如内部 Bandgap 参考电压 V_{BG}，运算放大器输出 OPAO，运算放大器输出 OPARO 或 A/D 转换器负电源电压 V_{SS}) 并直接将这些信号转换成 12 位的数字量。选择转换外部或内部模拟信号由 SAINS2~SAINS0 和 SACS3~SACS0 位共同控制。应注意的是，若通过 SAINS2~SAINS0 位选择内部模拟信号，则外部通道模拟输入将被自动关闭。关于 A/D 输入信号的更多详细信息请分别参见“A/D 转换器控制寄存器”和“A/D 转换器输入信号”章节。

外部输入通道	内部通道信号
AN0~AN10	V_{BG}, OPAO, OPARO, V_{SS}

A/D 转换器可以工作在三种模式，分别为手动触发转换模式，1-CH 自动转换模式和 2-CH 自动转换模式。手动触发转换模式为一般 ADC 功能，即为通过用户程序控制 START 位启动一次转换。自动转换模式是在接收到有效触发信号后自动开始转换。触发自动转换信号是通过 AADCTS1~AADCTS0 位选择。一组自动转换可包含 1、2、4、8 或 16 次转换由 AUTOADC2~AUTOADC0 位定义。

AUTOADC[2:0] 位	ATACE 位	A/D 转换模式	通道输入信号选择位
000	x	手动触发转换模式	SAINS[2:0]&SACS[3:0]
001~111	0	1-CH 自动转换模式	SAINS[2:0]&SACS[3:0]
001~111	1	2-CH 自动转换模式	CH1: ATAC1C[4:0] CH2: ATAC2C[4:0]

下图显示了 A/D 转换器的内部结构，以及相关的寄存器与控制位。



A/D 转换器结构

A/D 转换器寄存器介绍

A/D 转换器的所有操作由多个寄存器控制。一对 12-bit 只读寄存器用来存放 12-bit ADC 数据的值。两组 16-bit 寄存器 SADO1BH/SADO1BL 和 SADO2BH/SADO2BL 用于自动转换功能，储存 CH1 和 CH2 转换后累加的结果。剩下的寄存器为控制寄存器，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
SADO1BH	D15	D14	D13	D12	D11	D10	D9	D8
SADO1BL	D7	D6	D5	D4	D3	D2	D1	D0
SADO2BH	D15	D14	D13	D12	D11	D10	D9	D8
SADO2BL	D7	D6	D5	D4	D3	D2	D1	D0
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
SADC2	AADCTS1	AADCTS0	—	ATARPTG	—	AUTOADC2	AUTOADC1	AUTOADC0
LEBC	AADCST	—	—	LEB4	LEB3	LEB2	LEB1	LEB0
ATAC1C	—	—	ATACE	ATAC1SS	ATAC1S3	ATAC1S2	ATAC1S1	ATAC1S0
ATAC2C	—	—	—	ATAC2SS	ATAC2S3	ATAC2S2	ATAC2S1	ATAC2S0
ATADT	—	—	—	—	ATADT3	ATADT2	ATADT1	ATADT0
TCRC	—	—	—	TCON	—	—	TCEXTON	TCPSC
TCRH	D15	D14	D13	D12	D11	D10	D9	D8
TCRL	D7	D6	D5	D4	D3	D2	D1	D0

A/D 转换器寄存器列表

A/D 转换器数据寄存器

对于具有 12-bit A/D 转换器的单片机，需要两个数据寄存器存放转换结果，一个高字节寄存器 SADOH 和一个低字节寄存器 SADOL。在完成一次 A/D 转换后，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 转换数据结果位。未使用的位读为“0”。应注意若 A/D 转换器除能则 A/D 数据寄存器的内容将保持不变。

若 A/D 转换器工作在自动转换模式下 (AUTOADC[2:0]≠000)，ADRFS 位由硬件固定为 1。转换结果数据储存格式只有一种，即：D[11:8]=SADOH[3:0]；D[7:0]=SADOL[7:0]，SADOH 中未使用的位为“0”。

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 转换器数据寄存器

该单片机还提供两对缓冲寄存器，SADO1BH&SADO1BL 和 SADO2BH&SADO2BL。这两个 16 位的寄存器用于存放完成指定自动转换次数后，CH1 和 CH2 转换结果累加值。

寄存器	SADO1BH								SADO1BL							
位	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

A/D 自动转换数据缓冲器 1 寄存器

寄存器	SADO2BH								SADO2BL							
位	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

A/D 自动转换数据缓冲器 2 寄存器

A/D 转换器控制寄存器

A/D 转换器的工作由一系列寄存器进行控制。这些 8-bit 寄存器定义包括选择连接至内部 A/D 转换器的模拟通道，数字化数据储存格式，A/D 时钟源，A/D 转换工作模式，并控制和监测 A/D 转换器的开始和转换忙碌状态。由于单片机只包含一个实际的模数转换硬件电路，因此外部或内部模拟输入中的每一个都需要分别被发送到转换器。若 A/D 转换器工作为手动触发转换模式或 1-CH 自动转换模式，寄存器 SADC1 中的 SAINS2~SAINS0 位和寄存器 SADC0 的 SACS3~SACS0 位可用来决定哪个模拟信号将被连接到内部 A/D 转换器。但若 A/D 转换器工作在 2-CH 自动转换模式，CH1 和 CH2 信号需分别通过 ATAC1C 和 ATAC2C 寄存器中相应位进行选择。LEBC 寄存器用于使能 A/D 自动转换，设置 PWM0 触发信号前沿消隐时间。SADC2 寄存器用于选择自动转换开始触发信号及自动转换次数。ATAC1C, ATAC2C 和 ATADT 寄存器用于使能 2-CH 自动转换模式，选择 2-CH 转换模式下 CH1 和 CH2 的输入信号，以及两次 A/D 转换间的间隔时间等。

相关引脚共用功能选择位用来定义 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 (用于手动触发转换)
0 → 1 → 0：启动 A/D 转换
注：当 AUTOADC[2:0]=000 时，此位的设置才有效。
在手动触发 A/D 转换模式，此位用于启动一次 A/D 转换过程。通常此位为低，但如果设为高再被清零，将启动 A/D 转换过程。

Bit 6 **ADBZ**：A/D 转换忙碌标志位
0：无 A/D 转换正在进行中
1：A/D 转换正在进行中
该只读标志位用于表明 A/D 转换器是否正在执行 A/D 转换。当 ADBZ 位为高，表明 A/D 转换已启动。A/D 转换结束后，此位被清零。

Bit 5 **ADCEN**：A/D 转换器功能使能控制位
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。如果该位清零将关闭 A/D 转换器以降低功耗。当 A/D 转换器功能除能时，A/D 数据寄存器 SADOH 和 SADOL 的内容将保持不变。

Bit 4 **ADRFS**：A/D 转换器输出数据格式控制位
0：A/D 转换器输出数据格式 → SADOH=D[11:4]；SADOL=D[3:0]
1：A/D 转换器输出数据格式 → 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。
注：当 AUTOADC[2:0]=000 时，此位才可由软件改写；当 AUTOADC[2:0]≠000 时，ADRFS 位由硬件固定为 1。

Bit 3~0 **SACS3~SACS0**：A/D 转换器外部输入通道选择位
0000：AN0
0001：AN1
0010：AN2
0011：AN3
0100：AN4
0101：AN5
0110：AN6
0111：AN7
1000：AN8
1001：AN9
1010：AN10
1011~1111：未定义，输入浮空
注：该位仅用于手动触发或 1-CH 自动转换模式下通道输入信号选择。

● SADC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~5 **SAINS2~SAINS0**：A/D 转换输入信号选择位
000：外部信号 – 外部模拟通道输入，ANn
001：内部信号 – 内部 Bandgap 参考电压，V_{BG}
010：内部信号 – 运算放大器输出，OPAO
011：内部信号 – 运算放大器输出，OPARO
100：内部信号 – 内部负电源电压，V_{SS}
101~111：外部信号 – 外部模拟通道输入，ANn
注：该位仅用于手动触发或 1-CH 自动转换模式下通道输入信号选择。
当选择转换内部模拟信号，无论 SACS3~SACS0 为何值，外部通道信号输入将自动关闭。此举预防了外部通道输入与内部模拟信号连接从而导致的不可预期的后果。

Bit 4~3 **SAVRS1~SAVRS0**：A/D 转换器参考电压选择
00：外部 VREF 引脚
01：内部电源 V_{DD}
1x：外部 VREF 引脚
该位用于选择 A/D 转换器参考电压源。当选中内部参考电压源，来自外部 VREF 引脚的参考电压将被自动断开。

Bit 2~0 **SACKS2~SACKS0**：A/D 转换时钟 f_{ADCK} 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：$f_{SYS}/128$

● **SADC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	AADCTS1	AADCTS0	—	ATARPTG	—	AUTOADC2	AUTOADC1	AUTOADC0
R/W	R/W	R/W	—	R/W	—	R/W	R/W	R/W
POR	0	0	—	0	—	0	0	0

Bit 7~6 **AADCTS1~AADCTS0**：自动转换触发源选择位
00：PWM0 信号上升沿
01：OVP7OS 上升沿启动定时计数器，溢出后开始转换
10：AADCST 位从 0 变 1 时启动定时计数器，溢出后开始转换
11：软件位触发 – AADCST 位由 0 变 1 后开始转换
具体可参考“自动转换功能描述”章节内容。

Bit 5 未定义，读为“0”

Bit 4 **ATARPTG**：自动转换进行中，信号重复触发标志位
0：A/D 转换中无触发信号触发
1：A/D 转换中有触发信号触发
该位只能由硬件置位，通过软件清零，不能软件置位。当 AADCST 由 0 变 1 时，硬件会自动将此位清零。在自动转换过程中，若 A/D 转换还未完成，又有触发信号发生，则该位会被置位。

Bit 3 未定义，读为“0”

Bit 2~0 **AUTOADC2~AUTOADC0**：自动转换次数选择位
000：N=0，A/D 自动转换功能除能
001：N=1
010：N=2
011：N=4
100：N=8
101~111：N=16
这三位定义一组自动转换包含的 A/D 转换次数 (N)，也可用于除能自动转换功能。
若 N≠0 且 ATACE=0，转换 N 次后，N 次转换累加结果存于 SADO1BH 和 SADO1BL 并置位 ADF 标志位。若 N≠0 且 ATACE=1，转换 N 次后，N 次 CH1 转换累加结果存于 SADO1BH 和 SADO1BL；N 次 CH2 转换累加结果存于 SADO2BH 和 SADO2BL 并置位 ADF 标志位。

注：1. 若 AUTOADC[2:0]≠000，则 SADC0 寄存器中的“START”位失效，软件需通过设置 AADCST 位为高，使能一组 A/D 自动转换，待接收到相应触发源信号，即可开始 A/D 自动转换。
2. A/D 自动转换模式下，完成一次转换后，转换结果数据储存格式只有一种：D[11:8]=SADOH[3:0]; D[7:0]=SADOL[7:0]，SADOH 中未使用的位为“0”。

• LEBC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	AADCST	—	—	LEB4	LEB3	LEB2	LEB1	LEB0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7 **AADCST**：A/D 自动转换控制位
0：除能
1：使能
该位由软件置位，可通过硬件或软件清零。当 A/D 自动转换次数达到 AUTOADC2~AUTOADC0 的设置值时，会自动清除此位。当自动转换触发源信号突然停止或发生异常，可通过软件清零此位，相关应用请参考 A/D 自动转换软件停止说明章节。当此位由 0 变 1 时，会自动清除 ATARPTG 位。
注：当 AUTOADC2~AUTOADC0 为“000”时，此位无效。

Bit 6~5 未定义，读为“0”

Bit 4~0 **LEB4~LEB0**：前沿消隐时间控制位
LEB 时间 = (LEB[4:0]+1)×16/f_{PWM}，其中 f_{PWM}=32MHz
LEB[4:0] = 0~31，共 32 阶 LEB 时间可供设置。
例如：
LEB[4:0] = 0，则 LEB 时间 = (0+1)×16×0.03125μs=0.5μs
LEB[4:0] = 5，则 LEB 时间 = (5+1)×16×0.03125μs=3μs
注：LEB[4:0] 仅于 PWM0 上升沿触发时有效 (AADCTS[1:0]=00)。

• ATAC1C 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	ATACE	ATAC1SS	ATAC1S3	ATAC1S2	ATAC1S1	ATAC1S0
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 **ATACE**：2-CH A/D 自动转换功能控制位
0：除能
1：使能

Bit 4 **ATAC1SS**：CH1 输入信号来源选择 (2-CH A/D 自动转换模式)
0：来自内部通道信号
1：来自外部通道 ANn 输入
当 2-CH A/D 自动转换模式使能时，ATAC1SS 位的选择才有效。

Bit 3~0 **ATAC1S3~ATAC1S0**：CH1 输入信号选择 (2-CH A/D 自动转换模式)
若 ATAC1SS=0：
0000：V_{BG} 参考电压
0001：运算放大器输出，OPAO
0010：运算放大器输出，OPARO
0011：内部负电源电压，V_{SS}
0100~1111：V_{BG} 参考电压
若 ATAC1SS=1：
0000：AN0
0001：AN1
0010：AN2
0011：AN3
0100：AN4
0101：AN5
0110：AN6
0111：AN7

1000：AN8
1001：AN9
1010：AN10
1011~1111：未定义，输入浮空
当 2-CH 自动转换模式使能时，ATAC1S[3:0] 位的选择才有效。

● **ATAC2C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	ATAC2SS	ATAC2S3	ATAC2S2	ATAC2S1	ATAC2S0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5　未定义，读为“0”

Bit 4　**ATAC2SS**：CH2 输入信号来源选择 (2-CH A/D 自动转换模式)
0：来自内部通道信号
1：来自外部通道 ANn 输入
当 2-CH A/D 自动转换模式使能时，ATAC2SS 位的选择才有效。

Bit 3~0　**ATAC2S3~ATAC2S0**：CH2 输入信号选择 (2-CH A/D 自动转换模式)
若 ATAC2SS=0：
0000：V_{BG} 参考电压
0001：运算放大器输出，OPAO
0010：运算放大器输出，OPARO
0011：内部负电源电压，V_{SS}
0100~1111：V_{BG} 参考电压
若 ATAC2SS=1：
0000：AN0
0001：AN1
0010：AN2
0011：AN3
0100：AN4
0101：AN5
0110：AN6
0111：AN7
1000：AN8
1001：AN9
1010：AN10
1011~1111：未定义，输入浮空
当 2-CH 自动转换模式使能时，ATAC2S[3:0] 位的选择才有效。

● **ATADT 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	ATADT3	ATADT2	ATADT1	ATADT0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未定义，读为“0”

Bit 3~0　**ATADT3~ATADT0**：自动 A/D 转换间隔延迟时间
0000：$1\times t_{ADCK}$
0001：$2\times t_{ADCK}$
0010：$3\times t_{ADCK}$
…
…
1110：$15\times t_{ADCK}$
1111：$16\times t_{ADCK}$
注：$t_{ADCK}=1/f_{ADCK}$。

定时计数器寄存器

A/D 转换器电路包含了一个 16 位定时计数器功能，当定时计数器发生溢出时，就会触发 A/D 自动转换开始。有三个寄存器控制该定时计数器的操作，包括一个控制寄存器 TCRC 和 16 位的预载寄存器，TCRH&TCRL。TCRC 寄存器用于控制定时计数器使能，选择定时计数器启动方式以及计数时钟等。TCON 位置高后，定时计数器会从预载值开始向上计数，计满时间后，即计数值达到最大值 FFFFH 时溢出。

- **TCRC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	TCON	—	—	TCEXTON	TCPSC
R/W	—	—	—	R/W	—	—	R/W	R/W
POR	—	—	—	0	—	—	0	0

Bit 7~5　未定义，读为“0”

Bit 4　**TCON**：定时计数器使能控制位
0：除能
1：使能
注意该位只能软件清零，不能写 1。
当 AADCTS[1:0]=01 时：当 OVP7OS 产生一个上升沿，会把 TCON 位设为“1”，定时计数器溢出时，TCON 位会被自动清零。
当 AADCTS[1:0]=10 时：当使 AADCST 位从“0”变为“1”时，会把 TCON 位设为“1”。直到 AADCST 再次变为“0”时，才会把 TCON 位清零。

Bit 3~2　未定义，读为“0”

Bit 1　**TCEXTON**：外部上升沿触发 TCON 置位控制位
0：除能
1：使能
当此位为“0”时，OVP7OS 上升沿信号对 TCON 位无影响。当此位为“1”且 ADBZ 位为 0 时，OVP7OS 上升沿信号会把 TCON 位置为“1”。

Bit 0　**TCPSC**：定时计数器计数时钟选择
0：$f_H/4$
1：$f_H/8$

- **TCRH 寄存器**

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

Bit 7~0　**D15~D8**：16 位定时计数器预载寄存器高字节

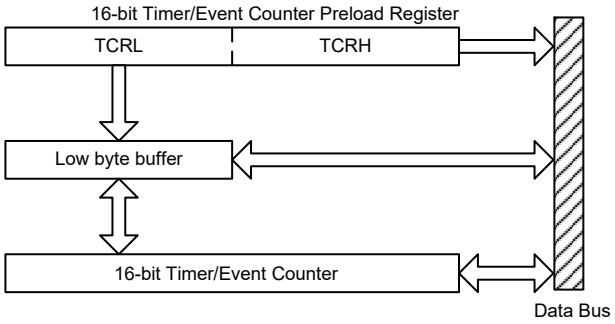
- **TCRL 寄存器**

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

Bit 7~0　**D7~D0**：16 位定时计数器预载寄存器低字节

写 TCRL 寄存器时，数据仅被写入到一个内部低字节缓冲器，而当写 TCRH 寄存器时，此高字节数据连同低字节缓冲器中的值将被一起写入到对应的 TCRH

和 TCRL 寄存器中。因此该定时计数器预载寄存器将在每次写入 TCRH 寄存器时被改写。读取 TCRH 寄存器将把 TCRH 和 TCRL 寄存器中的值分别锁存到目标地址和低字节缓冲器中，读取 TCRL 寄存器将读取到低字节缓冲器的值。



TCRH & TCRL 写入 / 读取

A/D 转换器参考电压

A/D 转换器的参考电压可以来自电源电压 V_{DD} 或来自 VREF 引脚上的外部参考源，通过配置 SADC1 寄存器中的 SAVRS1 和 SAVRS0 位进行选择。由于 VREF 引脚与其它功能共用引脚，当此引脚被选作参考电压输入引脚时，相应的引脚共用功能选择位应被预先设置以除能其它引脚共用功能。若是选择内部 V_{DD} 电源电压作为参考源，则来自 VREF 引脚的外部参考输入将由硬件自动关闭。

注意 A/D 转换器输入的模拟信号值一定不能超过所选的参考电压值。

SAVRS[1:0]	参考信号源	描述
00, 1x	VREF 引脚	来自外部 A/D 转换器参考电压引脚 VREF
01	V_{DD}	内部电源电压 V_{DD}

A/D 转换器参考电压选择

A/D 转换器输入信号

A/D 转换器的所有模拟输入引脚与 I/O 端口及其它功能共用。使用 PxS0 和 PxS1 寄存器中相应的引脚共用功能选择位，可以将它们设置为 A/D 转换器外部通道输入还是其它共用功能。如果相应引脚设置为 A/D 转换器模拟通道输入那么该引脚原引脚功能除能。通过这种方式，引脚的功能可由程序来控制，灵活地切换引脚功能。如果将引脚设为 A/D 输入，则通过寄存器编程设置的所有上拉电阻会自动断开。请注意，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，当相关 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。

由于该单片机只包含一个实际的模数转换硬件电路，因此外部或内部模拟信号中的每一个都需要分别被发送到转换器。对于手动触发转换模式，以及 1-CH A/D 自动转换模式，要进行 A/D 转换的信号由 SAINS2~SAINS0 和 SACS3~SACS0 位选择。SADC1 寄存器中的 SAINS2~SAINS0 位用于确定转换信号是来自外部通道输入或内部模拟信号。SADC0 寄存器中的 SACS3~SACS0 位用于确定所要转换的外部通道输入。若 SAINS2~SAINS0 位设为“000”，“101”~“111”则所要转换信号为外部模拟通道信号，SACS3~SACS0 位可决定选择哪个外部通道的信号进行转换。对于 2-CH A/D 自动转换模式，CH1 和 CH2 输入信号的选择是分别通过 ATAC1C 和 ATAC2C 寄存器进行选择。

ATAC1SS/ATAC2SS 位用于确定 CH1/CH2 转换信号是来自外部通道输入或内部模拟信号。ATAC1S[3:0]/ ATAC2S[3:0] 位用于确定 CH1/CH2 所要转换的外部通道输入或者所要转换的内部信号。

若选择内部模拟信号进行转换，外部通道信号输入将被自动关闭，此举将预防外部通道输入与内部模拟信号连接从而导致的不可预期的后果。

SAINS[2:0]	SACS[3:0]	输入信号	描述
000, 101~111	0000~1010	AN0~AN10	外部模拟通道输入 ANn
	1011~1111	—	输入浮空
001	xxxx	V_{BG}	内部参考电压 V_{BG}
010	xxxx	OPAO	内部 OPA 输出，OPAO
011	xxxx	OPARO	内部 OPA 输出，OPARO
100	xxxx	V_{SS}	内部负电源电压，V_{SS}

A/D 转换器输入信号选择 – 手动或 1-CH 自动转换模式

ATAC1SS/ ATAC2SS	ATAC1S [3:0]/ ATAC2S [3:0]	CH1/CH2 输入信号	描述
0	0000, 0100~1111	V_{BG}	内部参考电压 V_{BG}
	0001	OPAO	内部 OPA 输出，OPAO
	0010	OPARO	内部 OPA 输出，OPARO
	0011	V_{SS}	内部负电源电压，V_{SS}
1	0000~1010	AN0~AN10	外部模拟通道输入 ANn
	1011~1111	—	输入浮空

A/D 转换器输入信号选择 – 2-CH 自动转换模式

A/D 转换器时钟源及电源

A/D 转换器时钟源来自系统时钟 f_{SYS}，可选择为 f_{SYS} 或 f_{SYS} 的分频。分频比由 SADC1 寄存器中的 SACKS2~SACKS0 位确定。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 时钟源还有一些限制。由于允许的 A/D 时钟周期 t_{ADCK} 的范围为 0.5μs~10μs，所以选择系统时钟速度时就必须小心。如果系统时钟速度为 4MHz，SACKS2~SACKS0 位不能设为“000”、“110”或“111”。必须保证设置的 A/D 转换时钟周期不小于允许的时钟周期的最小值或不大于允许的时钟周期的最大值，否则将会产生不准确的 A/D 转换值。使用者可参考下面的表格，被标上星号 * 的数值需特别注意。

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}/32)	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*
16MHz	62.5ns*	125ns*	250ns*	500ns	1μs	2μs	4μs	8μs

A/D 时钟周期范例

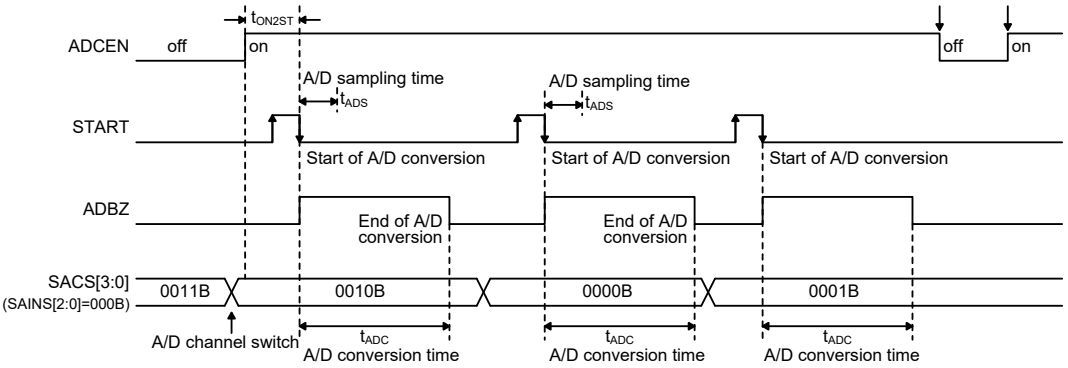
SADC0 寄存器的 ADCEN 位用于控制 A/D 转换电路电源的开启 / 关闭。该位必须置高以开启 A/D 转换器电源。当设置 ADCEN 位为高开启 A/D 转换器内部电路时，在 A/D 转换成功初始化前需一段延时，如时序图所示。即使选择无引脚作为 A/D 输入，若 ADCEN 设为“1”，仍然会产生功耗。因此当未使用 A/D 转换器功能时，在功耗敏感的应用中建议设置 ADCEN 为低以减少功耗。

A/D 转换率和时序图

一个完整的 A/D 转换包含两个阶段，数据采样和数据转换。数据采样时间定义为 t_{ADS}，需 4 个 A/D 时钟周期，而数据转换需 12 个 A/D 时钟周期。所以一个完整的 A/D 转换时间 t_{ADC} 一共需要 16 个 A/D 时钟周期。

最大 A/D 转换率 = 1/(A/D 时钟周期 × 16)

下列时序图表示一个外部通道输入信号模数转换过程中不同阶段的图形与时序。由应用过程控制开始 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 – AUTOADC[2:0]=000

手动触发 A/D 转换

此为一般 A/D 转换器功能 (AUTOADC[2:0]=000)。在此转换模式下，每次 A/D 转换都需通过设置 SADC0 寄存器中的 START 位进行手动触发开始。使用 SAINS[2:0] 和 SACS[3:0] 位选择 A/D 转换器输入信号。

SADC0 寄存器中的 START 位，用于复位和打开 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 位会自动清为“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。

下面概述执行手动触发开始的 A/D 转换需要的设置。

- 步骤 1

  通过 SADC1 寄存器中的 SACKS2~SACKS0 位，选择所需的 A/D 转换器转换时钟。

- 步骤 2

  将 SADC0 寄存器中的 ADCEN 位置高以使能 A/D 转换器。

- 步骤 3

  通过配置 SAINS2~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，前往步骤 4。

  若选择内部通道输入，前往步骤 5。

- 步骤 4

  若选择了外部通道输入，所需的外部输入通道可通过 SACS3~SACS0 位进行设置。当 A/D 输入信号来自外部通道输入，相应引脚应先通过配置相关的引脚共用功能控制位选择。完成此步骤后，前往步骤 6。

- 步骤 5

  若 SAINS2~SAINS0 位设置为 001~011 中任一值，将会选择对应的内部信号作为 A/D 转换器的输入。此时外部通道模拟输入将被自动断开。完成此步骤后，前往步骤 6。

- 步骤 6

  通过 SADC1 寄存器中的 SAVRS1~SAVRS0 位选择参考电压。

- 步骤 7

  设置 SADC0 寄存器中的 ADRFS 位选择 A/D 转换器输出数据格式。

- 步骤 8

  如果要使用 A/D 转换中断，则中断控制寄存器需要正确地设置，以确保 A/D 中断功能是开启的。总中断控制位 EMI 和 A/D 转换器中断位 ADE 都需要提前置位为“1”。

- 步骤 9

  现在可以通过设定 START 位从低到高再到低，开始 A/D 转换的过程。

- 步骤 10

  若 A/D 转换正在进行，ADBZ 标志将置高。A/D 转换结束后，ADBZ 标志置低，并可从 SADOH 和 SADOL 寄存器读取输出数据。

注：若使用轮询 SADC0 寄存器中的 ADBZ 位的方法以检测模数转换过程是否完成，上述中断使能步骤可以忽略。

1-CH 自动转换模式

在 1-CH 自动转换模式 (ATACE=0，AUTOADC[2:0]≠000)，使用 SAINS[2:0] 和 SACS[3:0] 位选择 A/D 转换器 CH1 输入信号。当其它 A/D 参数设置完成后，设置自动转换次数、选择转换开始触发信号并依所选的触发信号设置前沿消隐 (LEB) 时间或设置定时计数器相关参数。

之后设置 AADCST 位由低到高，使能自动转换。有触发信号的上升沿进来后，启动 LEB 时间或启动定时计数器计数或直接开始转换，每次转换后的数据会由 SADOH&SADOL 寄存器存入到 SADO1BH &SADO1BL。完成整组自动转换后，SADO1B 寄存器中的值为这几次转换的结果累加值。

下面概述执行 1-CH 自动触发开始的 A/D 转换需要的设置。

- 步骤 1

  通过 SADC1 寄存器中的 SACKS2~SACKS0 位，选择所需的 A/D 转换器转换时钟。

- 步骤 2

  将 SADC0 寄存器中的 ADCEN 位置高以使能 A/D 转换器。

- 步骤 3

  将 ATAC1C 寄存器中的 ATACE 位设置为“0”以使能 1-CH 自动转换功能。

- 步骤 4

  通过配置 SADC1 寄存器中的 SAINS2~SAINS0 位，选择 CH1 转换信号来自外部通道还是内部通道。

  若选择外部通道输入，前往步骤 5。

  若选择内部通道输入，前往步骤 6。

- 步骤 5

  若选择外部通道输入，需先设置相关的引脚共用功能控制位，将其规划为 A/D 输入引脚，接着设置 SACS3~SACS0 位选择该外部通道接到 A/D 转换器。接着前往步骤 7。

- 步骤 6

  若 SAINS2~SAINS0 位设置为 001~011 中任一值，将会选择对应的内部信号作为 A/D 转换器 CH1 的输入。此时外部通道模拟输入将被自动断开。接着前往步骤 7。

- 步骤 7

  通过 SADC2 寄存器中的 AUTOADC2~AUTOADC0 位，设置 ADC 自动转换包含的转换次数。

- 步骤 8

  通过 SADC2 寄存器中的 AADCTS1~AADCTS0 位，选择 ADC 自动转换的触发信号。

  若配置为“00”，选择 PWM0 上升沿输入，接着执行步骤 9。

  若配置为“01”，选择 OVP7OS 上升沿触发定时计数器开始计数，接着执行步骤 10。

  若配置为“10”，选择 AADCST 上升沿触发定时计数器开始计数，接着执行步骤 11。

  若配置为“11”，选择 AADCST 上升沿直接触发转换开始，接着执行步骤 13。

- 步骤 9

  通过 LEBC 寄存器中 LEB[4:0] 位，设置 ADC 开始转换前的前沿消隐时间，接着执行步骤 14。

- 步骤 10

  通过 TCRC 寄存器中的 TCEXTON 位设定是否使用 OVP7OS 上升沿来触发定时计数器，接着执行步骤 11。

- 步骤 11

  通过 TCRC 寄存器中 TCPSC 位，设置定时计数器时钟源，接着执行步骤 12。

- 步骤 12

  通过 TCRH 和 TCRL 寄存器，设置定时计数器计数值，接着执行步骤 14。

- 步骤 13

  通过 ATADT 寄存器的 ATADT[3:0] 位设置两次 A/D 转换之间的间隔时间，接着执行步骤 14。

- 步骤 14

  通过 SADC1 寄存器中的 SAVRS1~SAVRS0 位选择参考电压，接着执行步骤 15。

- 步骤 15

  如果要使用 A/D 转换中断，则中断控制寄存器需要正确地设置，以确保 A/D 中断功能是开启的。总中断控制位 EMI 以及 A/D 转换器中断控制位 ADE 都需要提前置位为“1”。接着执行步骤 16。

- 步骤 16

  设置 AADCST 位从低到高，当 AADCTS[1:0] 位所设定的触发信号到来时开始 A/D 自动转换。

注：当指定次数的 A/D 自动转换完成后，AADCST 位会被硬件自动清零且 A/D 转换中断标志位 ADF 会被设置为 1。

2-CH 自动转换模式

在 2-CH 自动转换模式 (ATACE=1，AUTOADC[2:0]≠000)，由 ATAC1C 寄存器中的 ATAC1SS 和 ATAC1S[3:0] 位决定自动转换 CH1 信号来源；由 ATAC2C 寄存器中的 ATAC2SS 和 ATAC2S[3:0] 位决定自动转换 CH2 信号来源。当其它 A/D 参数设置完成后，设置 ATACE 位、自动转换次数、选择转换开始触发信号并依所选的触发信号设置前沿消隐 (LEB) 时间或设置定时计数器相关参数。

之后设置 AADCST 位由低到高，使能自动转换。有触发信号的上升沿进来后，启动 LEB 时间或启动定时计数器计数或直接开始 CH1 转换。CH1 转换完成后，等待设定的延迟时间到达后，再开始 CH2 转换。每次 CH1 转换的数据会由 SADOH&SADOL 寄存器存入到 SADO1BH&SADO1BL。CH2 转换的数据会由 SADOH&SADOL 寄存器存入 SADO2BH&SADO2BL，待完成设置的转换次数后，即完成 2-CH 自动 A/D 转换，此时 SADO1B 寄存器中的值为 CH1 转换的结果累加值 , SADO2B 寄存器中的值为 CH2 转换的结果累加值。

若 AADCTS[1:0]=11 时，每个数据转换之间的延迟时间是根据 ATADT[3:0] 的设置来延迟。

下面概述执行 2-CH 自动触发开始的 A/D 转换需要的设置。

- 步骤 1

  通过 SADC1 寄存器中的 SACKS2~SACKS0 位，选择所需的 A/D 转换器转换时钟。

- 步骤 2

  将 SADC0 寄存器中的 ADCEN 位置高以使能 A/D 转换器。

- 步骤 3

  将 ATAC1C 寄存器中的 ATACE 位设置为“1”以使能 2-CH 自动转换功能。

- 步骤 4

  通过配置 ATACnC 寄存器中的 ATACnSS，选择连接至自动转换通道 CH1 和 CH2 的信号。

  若选择外部通道输入 (ATACnSS=1)，前往步骤 5。

  若选择内部通道输入 (ATACnSS=0)，前往步骤 6。

- 步骤 5

  若选择了外部通道输入，所需的外部输入通道可通过 ATACnS[3:0] 位指定。当 A/D 输入信号来自外部通道输入，相应引脚应先通过配置相关的引脚共用功能控制位选择。完成此步骤后，前往步骤 7。

- 步骤 6

  通过 ATACnS[3:0] 位选择需要的内部输入信号。完成此步骤后，前往步骤 7。

- 步骤 7

  通过 SADC2 寄存器中的 AUTOADC2~AUTOADC0 位，设置 ADC 自动转换包含的转换次数。

- 步骤 8

  通过 SADC2 寄存器中的 AADCTS1~AADCTS0 位，选择 ADC 自动转换的触发信号。

  若配置为“00”，选择 PWM0 上升沿输入，接着执行步骤 9。

  若配置为“01”，选择 OVP7OS 上升沿触发定时计数器开始计数，接着执行步骤 10。

  若配置为“10”，选择 AADCST 上升沿触发定时计数器开始计数，接着执行步骤 11。

  若配置为“11”，选择 AADCST 上升沿直接触发转换开始，接着执行步骤 13。

- 步骤 9

  通过 LEBC 寄存器中 LEB[4:0] 位，设置前沿消隐时间，接着执行步骤 13。

- 步骤 10

  通过 TCRC 寄存器中的 TCEXTON 位设定是否使用 OVP7OS 上升沿来触发定时计数器，接着执行步骤 11。

- 步骤 11

  通过 TCRC 寄存器中 TCPSC 位，设置定时计数器时钟源，接着执行步骤 12。

- 步骤 12

  通过 TCRH 和 TCRL 寄存器，设置定时计数器计数值，接着执行步骤 13。

- 步骤 13

  通过 ATADT 寄存器的 ATADT[3:0] 位设置两次 A/D 转换之间的延迟时间，接着执行步骤 14。

- 步骤 14

  通过 SADC1 寄存器中的 SAVRS1~SAVRS0 位选择参考电压。

- 步骤 15

  如果要使用 A/D 转换中断，则中断控制寄存器需要正确地设置，以确保 A/D 中断功能是开启的。总中断控制位 EMI 以及 A/D 转换器中断控制位 ADE 都需要提前置位为“1”。

- 步骤 16

  设置 AADCST 位从低到高，开始 A/D 自动转换过程。

注：当指定次数的 A/D 自动转换完成后，AADCST 位会被硬件自动清零且 A/D 转换中断标志位 ADF 会被设置为 1。

自动转换功能描述

一次自动触发转换过程可以由 PWM0 信号、OVP7OS 信号或 AADCST 软件位上升沿触发前沿消隐或定时计数器计数溢出后启动，也可以由 AADCST 软件位直接启动，触发源通过 AADCTS1~AADCTS0 位选择。下面将分别来作介绍。

AADCTS[1:0]=00B

若设置 AADCTS1~AADCTS0 位为 00，AUTOADC2~AUTOADC0 位不等于 000，则于软件设置 AADCST 位后使能 ADC 自动同步延时转换，转换开始触发源为 PWM0 信号上升沿。自 AADCST 为“1”后的每一个 PWM0 上升沿都会先启动前沿消隐，等LEB4~LEB0 设置的LEB时间到后，触发 ADC 转换开始。直到 A/D 转换次数与 AUTOADC2~AUTOADC0 位所设置的次数相同时，会自动清除 AADCST 位，并置位 ADC 中断标志位。若 PWM0 突然停止，AADCST 位可由软件清零。

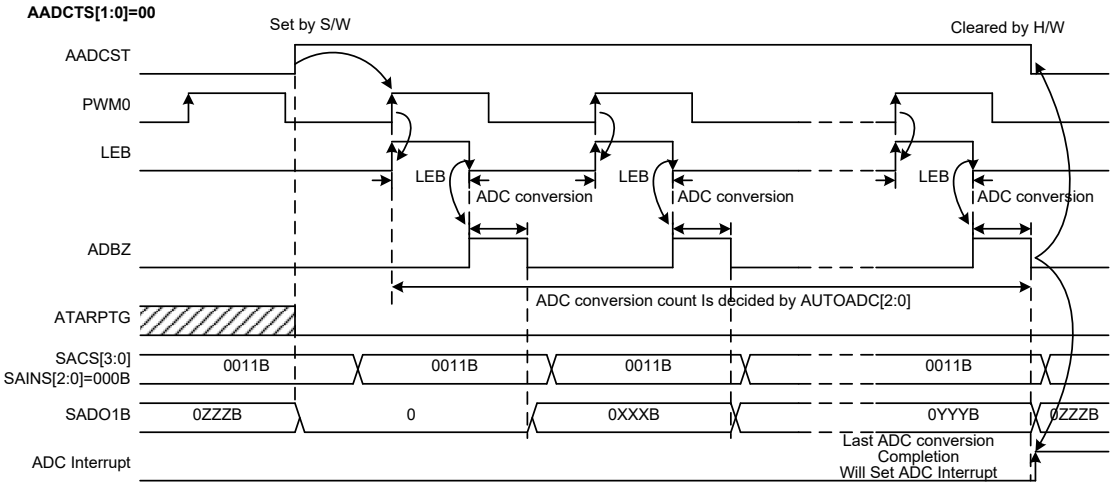
AADCST 为非重复触发，一旦设置后，将不可再次触发。也就是说，ADC 转换会依 AUTOADC2~AUTOADC0 设置的次数转换完毕才停止。程序设计时请注意，在 AADCST 清除后，请同时清除 ADF 标志位。

注：自动转换尚未完成，而 AADCST 被软件清零，若 ADBZ=1 则会待该次的 ADC 转换结束将 ADBZ 变为 0，若 ADBZ=0，在计数 LEB 时则会立即停止计数 LEB，之后才将自动转换这部份电路复位至初始状态，待复位完成后，方可重新置位 AADCST 位以启动下一次的 A/D 自动转换。若 AADCST 被软件清零，A/D 转换数据已达设定笔数时，也会触发中断。

1-CH 自动转换

当 AADCST 位由 0 变 1 时，会自动清除 ATARPTG 位，并等待 PWM0 有一个上升沿信号进入，启动 LEB 延迟，待延迟时间完成后，A/D 开始转换。当转换次数达到设定值，硬件会自动将 AADCST 清为“0”，并设置 A/D 转换中断标志位 ADF 为 1。

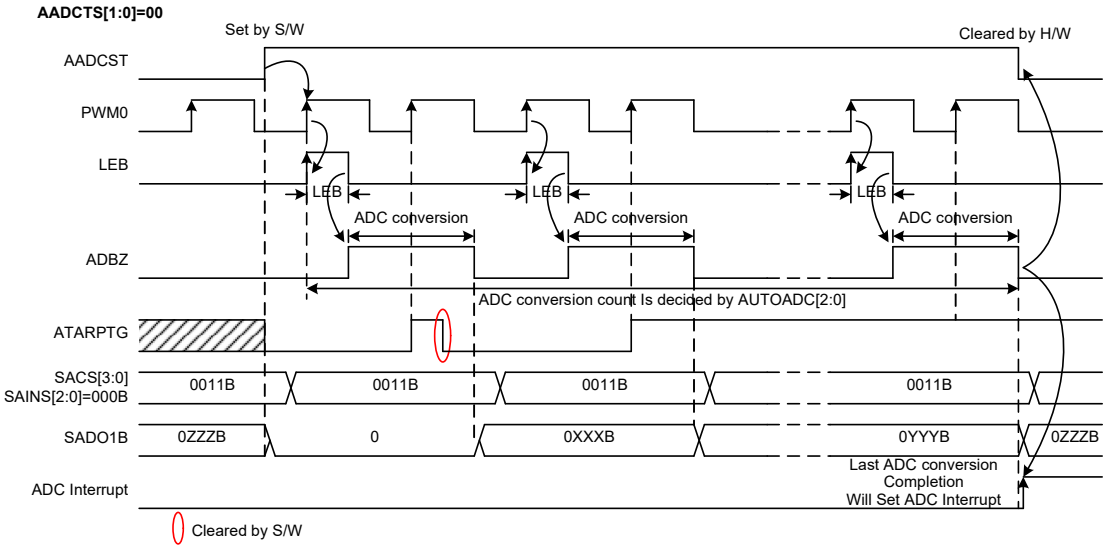
下图为在 PWM0 上升沿信号进入后，A/D 转换器有足够时间将数据转换完毕。然后下个 PWM0 上升沿信号进入，转换第二笔数据，直到完成整个自动转换设定。



1-CH 自动转换时序图 – 1 (AADCTS[1:0]=00)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

下图为在 PWM0 上升沿信号进入后，A/D 转换器没有足够时间将数据转换完毕，下个 PWM0 上升沿信号就进入，而 ATARPTG 会被置位为“1”。此时电路会继续将数据转换完后，且等到新的 PWM0 上升沿信号进入，才会开始转换第二笔数据，直到完成整个自动转换设定。



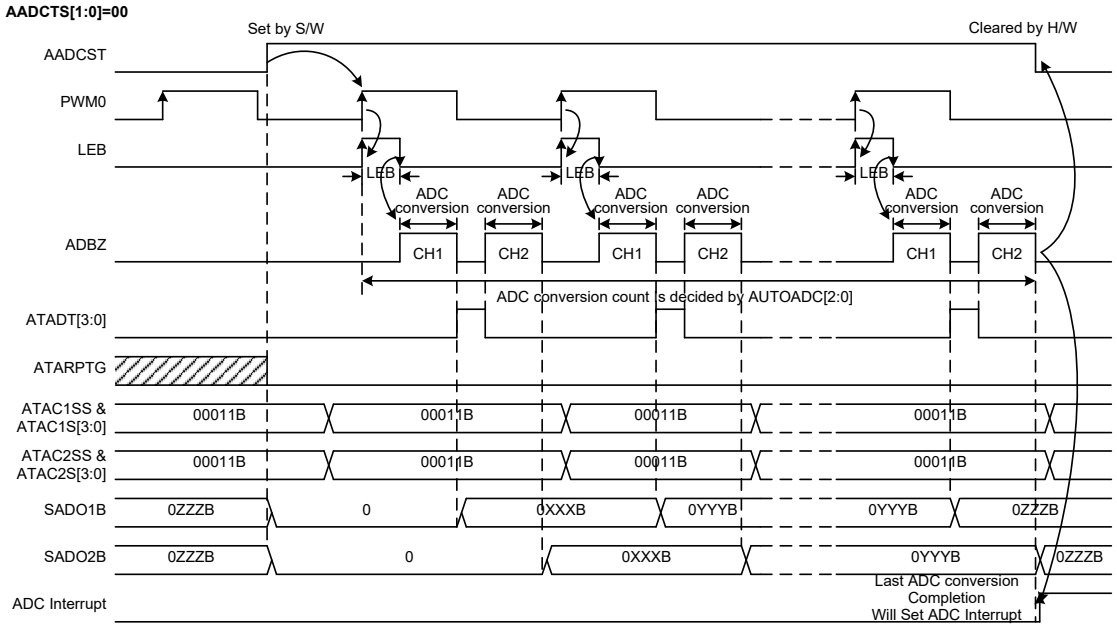
1-CH 自动转换时序图 – 2 (AADCTS[1:0]=00)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

2-CH 自动转换

当 AADCST 位由 0 变 1 时，会自动清除 ATARPTG 位，并等待 PWM0 有一个上升沿信号进入，启动 LEB 延迟，待延迟时间完成后，A/D 转换器开始 CH1 转换，待 CH1 转换完成，延迟一段时间 (延迟时间由 ATADT[3:0] 设置)，开始 CH2 转换，待 CH2 转换完成后，等待下一次的 PWM0 上升沿信号进入，再开始转换，直到转换次数达到设定值，硬件会自动将 AADCST 清为“0”，并设立 A/D 中断标志位 ADF 为高。

下图在 PWM0 上升沿信号进入后，A/D 转换器有足够时间将两通道的数据转换完毕，直到下个 PWM0 上升沿信号进入，转换第二笔数据，直到完成整个自动转换设定。



2-CH 自动转换时序图 – 1 (AADCTS[1:0]=00)

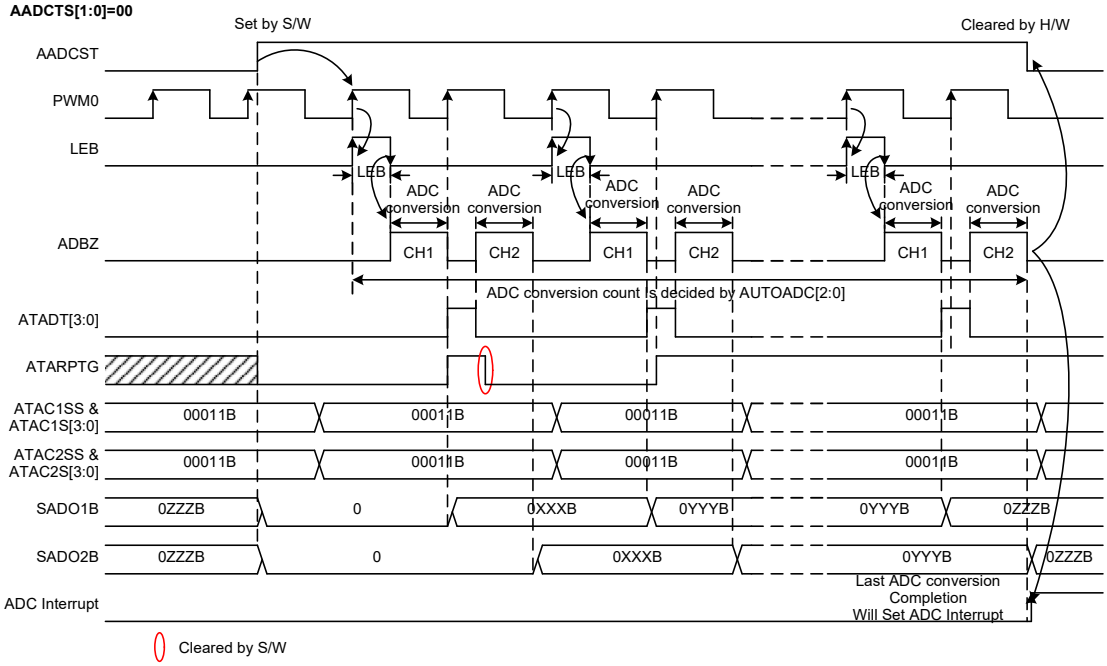
注：1. AUTOADC[2:0] ≠ 000

2. XXXB 为第一笔 A/D 转换值

3. YYYB 为累加后的 A/D 值

4. ZZZB 为自动 N 次转换后最终的累加值

下图在 PWM0 上升沿信号进入后，A/D 转换器没有足够时间将两通道的数据转换完毕，下个 PWM0 上升沿信号就进入，而 ATARPTG 会被置位为“1”，此时 A/D 转换器会持续将当前两通道的数据转换完后，等到新的 PWM0 上升沿信号进入，才会开始转换第二笔数据，直到完成整个自动转换设定。



2-CH 自动转换时序图 – 2 (AADCTS[1:0]=00)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

AADCTS[1:0]=01B

若设置 AADCTS1~AADCTS0 位为 01，AUTOADC2~AUTOADC0 位不等于 000，则于软件设置 AADCST 位后使能 ADC 自动同步延时转换，转换开始触发源为 OVP7OS 信号上升沿启动的 16 位定时计数器计数溢出事件。自 AADCST 为“1”后，等待 OVP7OS 产生一个上升沿信号，硬件会把 TCON 也设置为“1”，定时计数器从预载值开始向上计数。直到计数器数到最大值 FFFFH 溢出，会把 TCON 清零，触发 ADC 转换开始。定时计数器将等待下一个 OVP7OS 上升沿信号。直到 A/D 转换次数与 AUTOADC2~AUTOADC0 位所设置的次数相同时，会自动清除 AADCST 位，并置位 ADC 中断标志位。若触发信号发生异常没有产生上升沿信号，AADCST 位可由软件清零。

AADCST 为非重复触发，一旦设置后，将不可再次触发。也就是说，ADC 转换会依 AUTOADC2~AUTOADC0 设置的次数转换完毕才停止。程序设计时请注意，在 AADCST 清除后，请同时清除 ADF 标志位。

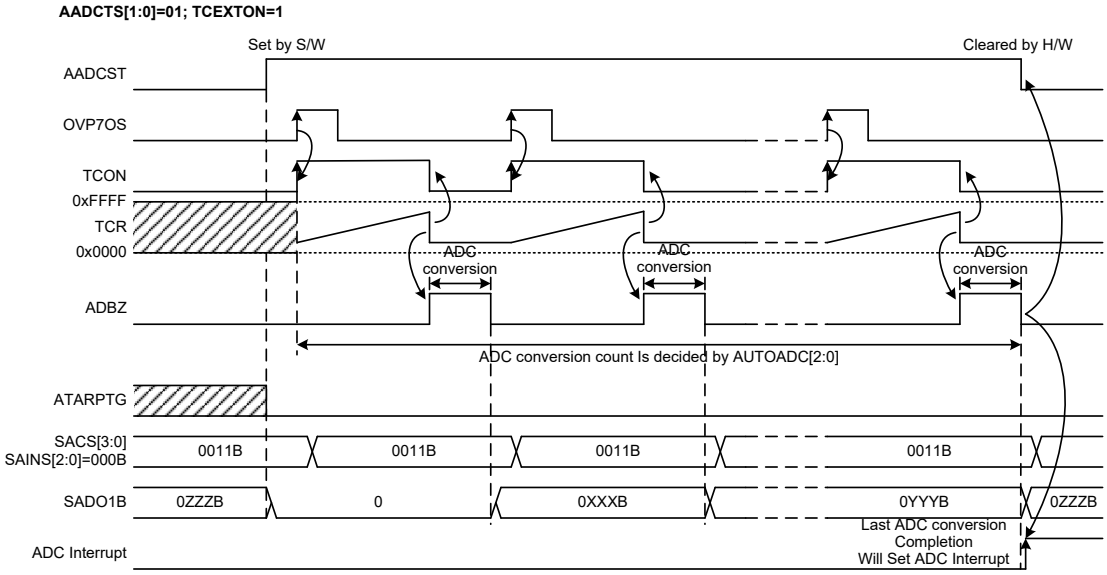
注：自动转换尚未完成，而 AADCST 被软件清零，若 ADBZ=1 请等待 ADBZ=0 后，将 ADC 切换为手动触发转换模式 (AUTOADC[2:0]=000)，并启动一次 ADC 转换 (START 位执行 0->1->0)，完成转换再切回自动转换模式，之后才将自动转换这部份电路复位至初始状态，待复位完成后，方可重新置位 AADCST 位以启动下一次的 A/D 自动转换。若 AADCST 被软件清零，A/D

转换数据已达设定笔数时，也会触发中断。

另外需注意当 AADCST=0 或 AADCST=1 但 TCEXTON=0 时，OVP7OS 产生一个上升沿信号，并不会把 TCON 位设置为“1”。

1-CH 自动转换

当 AADCST 位由 0 变 1 时，会自动清除 ATARPTG 位，并等待 OVP7OS 产生一个上升沿信号进入，启动定时计数器开始向上计数，待溢出会清除 TCON 位，同时触发 A/D 转换开始，当转换次数达到设定值，硬件会自动将 AADCST 清为“0”，并设立 A/D 转换中断标志位 ADF 为 1。

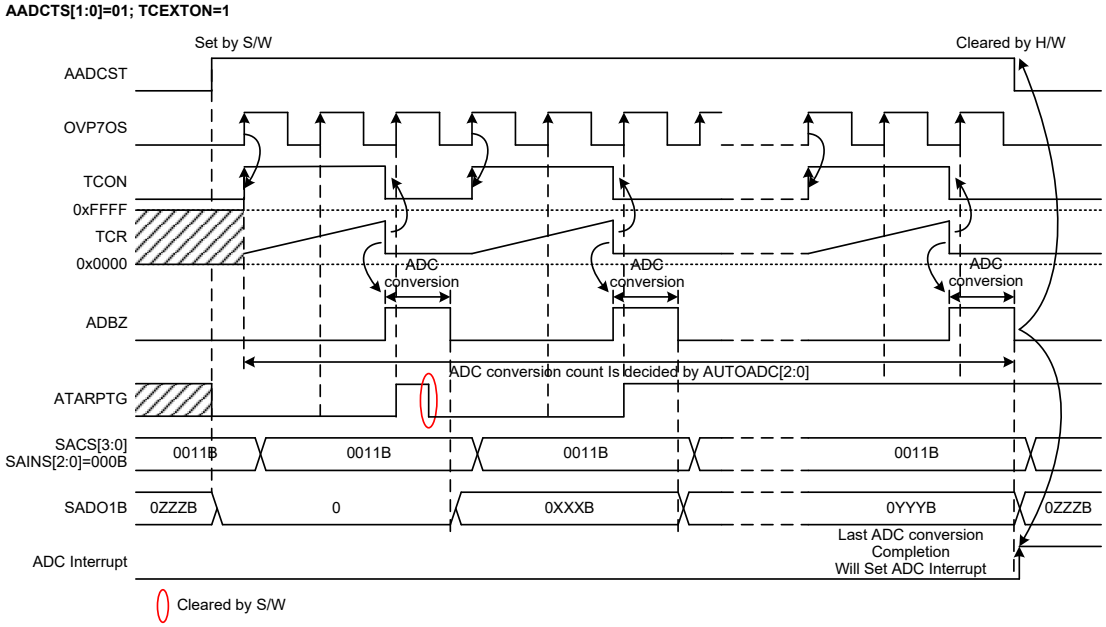


1-CH 自动转换时序图 – 1 (AADCTS[1:0]=01)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

下图在 AADCST 由 0 变 1 时，OVP7OS 产生一个上升沿信号会触发 TCON 变为“1”，启动定时计数器开始向上计数，待溢出会清除 TCON 位为零，并会触发 ADC 转换开始。若在 A/D 还未将数据转换完毕，下个 OVP7OS 上升沿信号就进入，则 ATARPTG 会被置位为“1”，此时电路会继续将该笔数据转换完后，且等到新的 OVP7OS 上升沿信号进入并使定时计数器开始计数，计数到溢出，才会开始转换第二笔数据，直到完成整个自动转换设定。

注：若是在计数期间尚未溢出时，OVP7OS 产生一个上升沿信号，并不影响 TCON 和 ATARPTG 位。



1-CH 自动转换时序图 – 2 (AADCTS[1:0]=01)

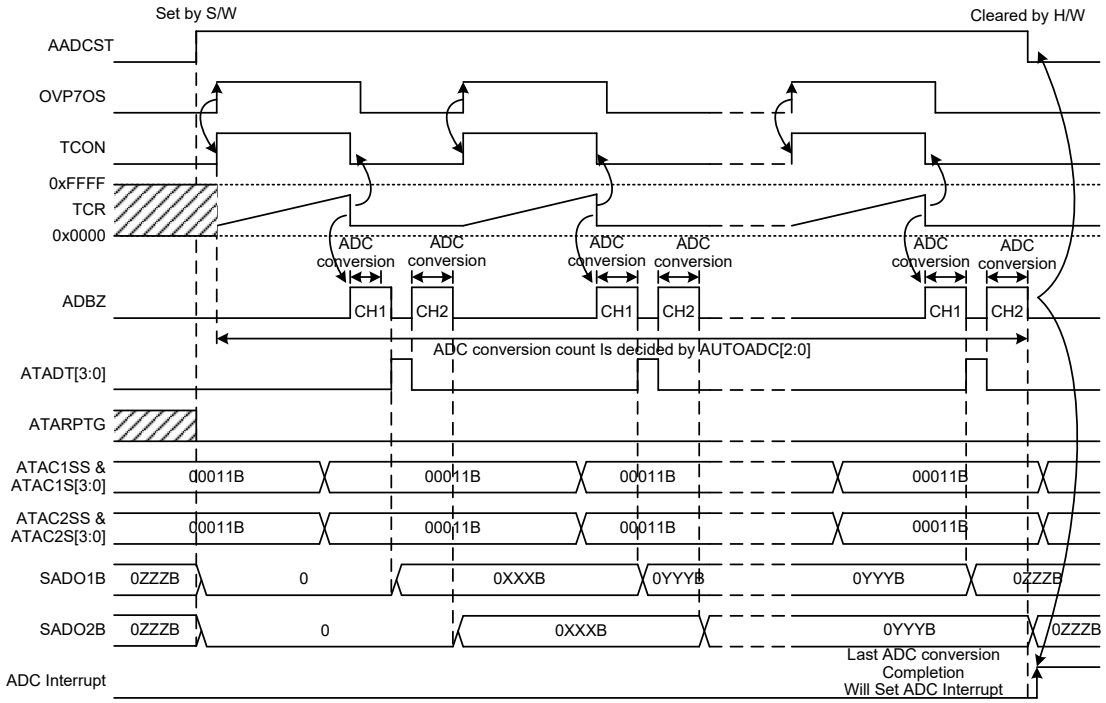
注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

2-CH 自动转换

当 AADCST 位由 0 变 1 时，会自动清除 ATARPTG 位，并等待 OVP7OS 产生一个上升沿信号进入，启动定时计数器开始向上计数，待溢出会清除 TCON 位，同时触发 ADC 转换开始。A/D 转换器开始 CH1 转换，待 CH1 转换完成，延迟一段时间 (延迟时间由 ATADT[3:0] 设置)，开始 CH2 转换，待 CH2 转换完成后，等待下一次的 OVP7OS 上升沿信号触发。当所有转换次数达到设定值，硬件会自动将 AADCST 清为“0”，并设立 A/D 中断标志位 ADF 为高。

下图在 AADCST 由 0 变 1 时，OVP7OS 产生一个上升沿信号触发 TCON 变为“1”，启动定时计数器开始向上计数，待溢出会清除 TCON 位为零，并会触发 ADC 转换开始，A/D 转换器有足够时间将两通道的数据转换完毕，直到下个 OVP7OS 产生一个上升沿信号进入，转换第二笔数据，直到完成整个自动转换设定。

AADCTS[1:0]=01; TCEXTON=1

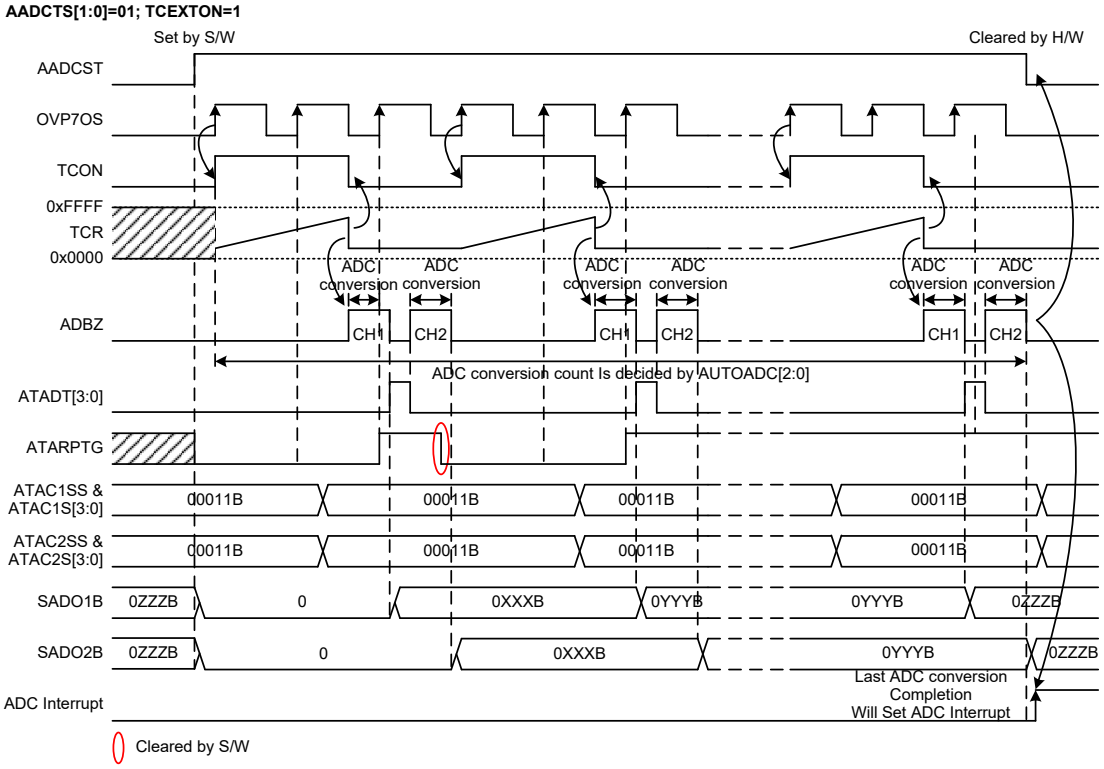


2-CH 自动转换时序图 – 1 (AADCTS[1:0]=01)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

下图在 AADCST 由 0 变 1 时，OVP7OS 产生一个上升沿信号触发 TCON 变为“1”，启动定时计数器开始向上计数，待溢出会清除 TCON 位为零，并会触发 ADC 转换开始，A/D 转换器没有足够时间将两通道的数据转换完毕，下个 OVP7OS 上升沿信号就进入，而 ATARPTG 会被置位为“1”，此时 A/D 转换器会持续将当前两通道的数据转换完后，等到新的 OVP7OS 上升沿信号触发的定时计数器溢出后，才会开始转换第二笔数据，直到完成整个自动转换设定。

注：若是在计数期间尚未溢出时，OVP7OS 产生一个上升沿信号，并不影响 TCON 和 ATARPTG 位。

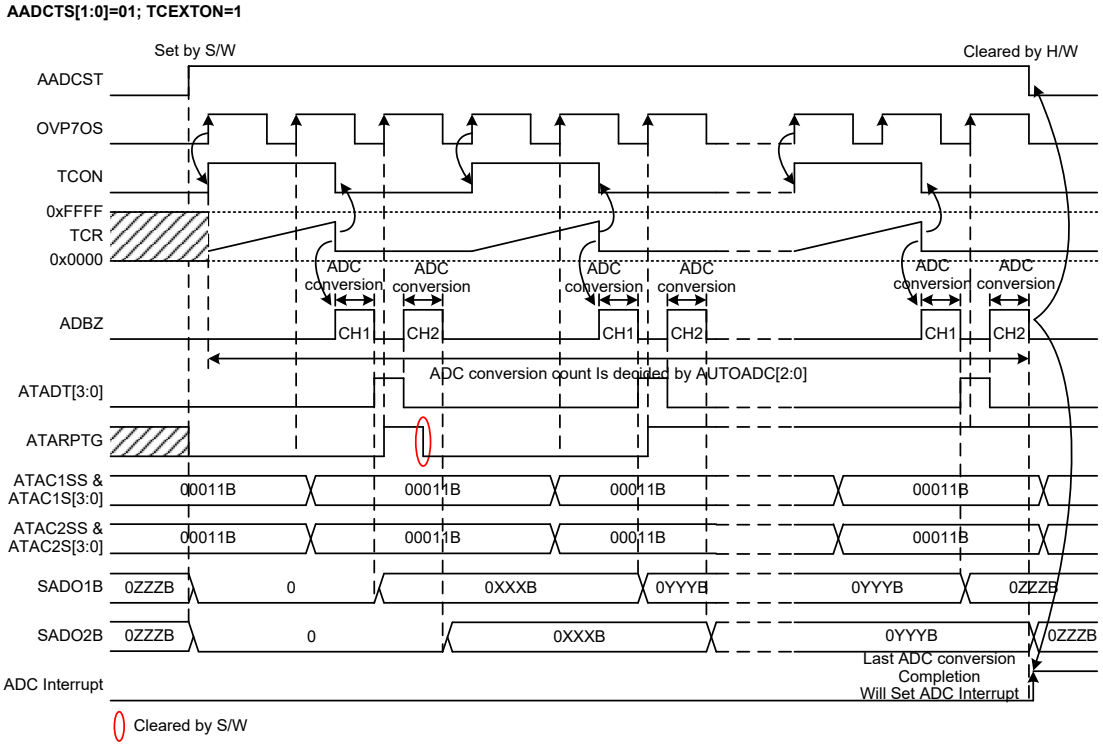


2-CH 自动转换时序图 – 2 (AADCTS[1:0]=01)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

下图在 AADCST 由 0 变 1 时，OVP7OS 产生一个上升沿信号触发 TCON 变为“1”，启动定时计数器开始向上计数，待溢出会清除 TCON 位为零，并会触发 ADC 转换开始。当 A/D 处在两通道转换之间的 ATADT[3:0] 时间时，下个 OVP7OS 上升沿信号就进入，而 ATARPTG 会被置位为“1”，此时 A/D 转换器会持续将当前两通道的数据转换完后，等到新的 OVP7OS 上升沿信号触发的定时计数器溢出后，才会开始转换下一笔数据，直到完成整个自动转换设定。

注：若是在计数期间尚未溢出时，OVP7OS 产生一个上升沿信号，并不影响 TCON 和 ATARPTG 位。



2-CH 自动转换时序图 – 3 (AADCTS[1:0]=01)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

AADCTS[1:0]=10B

若设置 AADCTS1~AADCTS0 位为 10，AUTOADC2~AUTOADC0 位不等于 000，则于软件设置 AADCST 位后使能 ADC 自动同步延时转换。设置 AADCST 为“1”，硬件会自动把 TCON 也设置为“1”，定时计数器从预载值开始向上计数，直到计数器数到最达值 FFFFH 溢出，会触发 ADC 转换开始。定时计数器将重新开始从预载值向上计数，溢出后触发下一次 ADC 转换开始。直到 A/D 转换次数与 AUTOADC2~AUTOADC0 位所设置的次数相同时，会自动清除 AADCST 位以及 TCON 位，并置位 ADC 中断标志位。若定时计数器发生异常或 TCON 位被清零，AADCST 位可由软件清零。

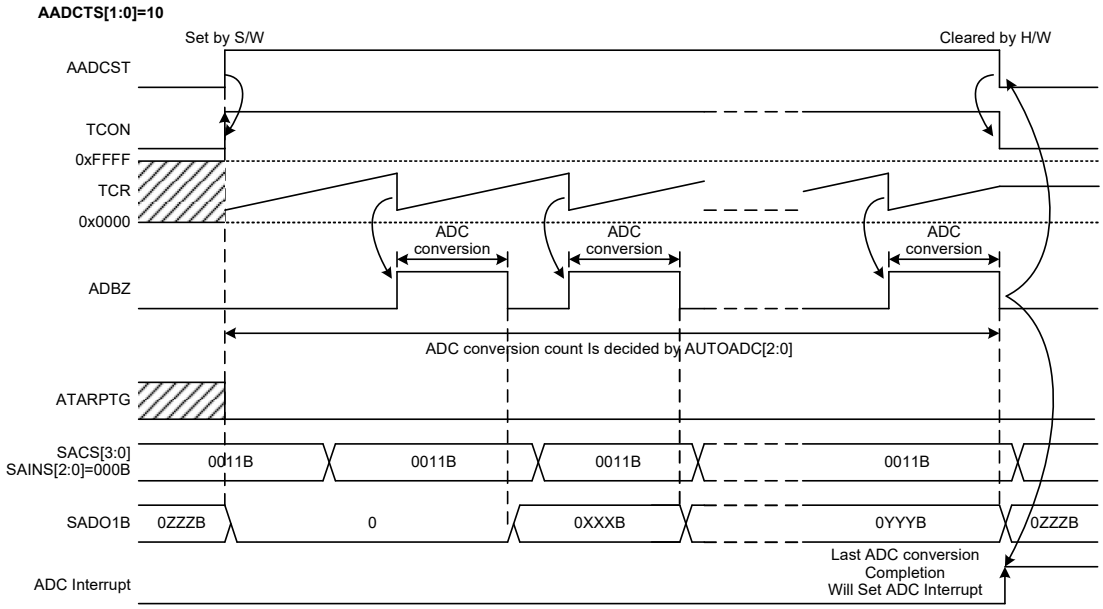
AADCST 为非重复触发，一旦设置后，将不可再次触发。也就是说，ADC 转换会依 AUTOADC2~AUTOADC0 设置的次数转换完毕才停止。程序设计时请注意，在 AADCST 清除后，请同时清除 ADF 标志位。

注：自动转换尚未完成，而 AADCST 被软件清零，若 ADBZ=1 请等待 ADBZ=0 后，将 ADC 切换为手动触发转换模式 (AUTOADC[2:0]=000)，并启动一次 ADC 转换 (START 位执行 0->1->0)，完成转换再切回自动转换模式，之后才将自动转换这部份电路复位至初始状态，待复位完成后，方可重新置位 AADCST 位以启动下一次的 A/D 自动转换。若 AADCST 被软件清零，A/D 转换数据已达设定笔数时，也会触发中断。

1-CH 自动转换

当 AADCST 位由 0 变 1 时，会自动清除 ATARPTG 位，并将 TCON 位设置为“1”，启动定时计数器开始向上计数，溢出后，触发 A/D 转换开始。当转换次数达到设定值，硬件会自动将 AADCST 和 TCON 位清为“0”，并设置 A/D 转换中断标志位 ADF 为 1。

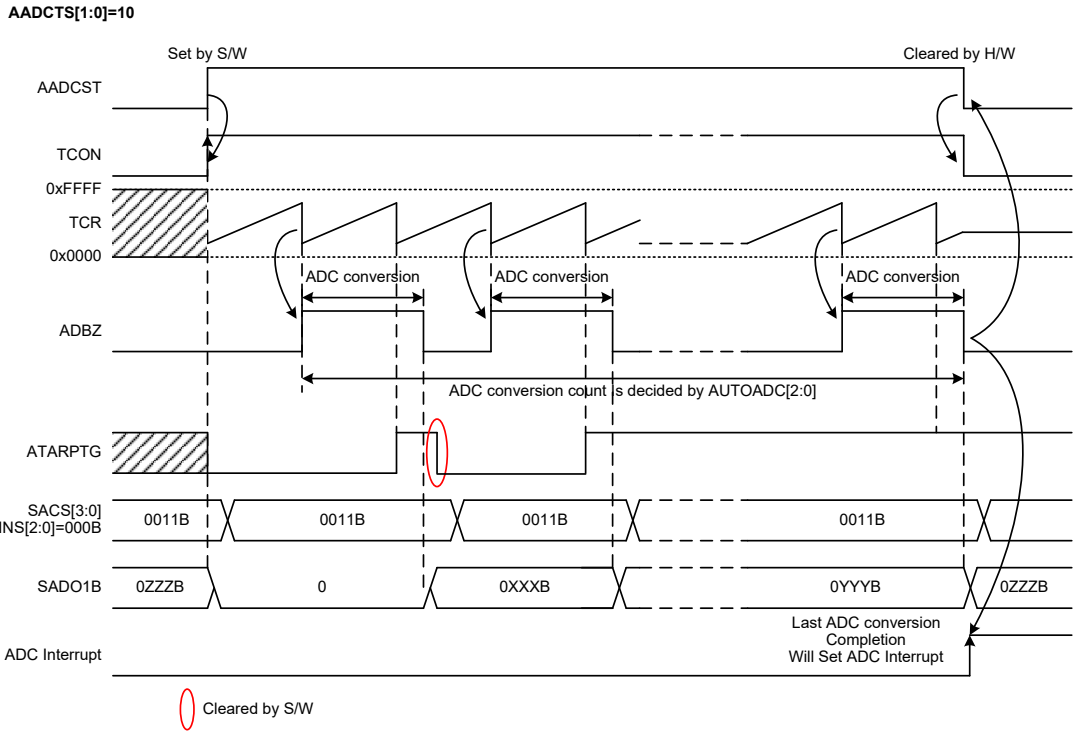
下图为 1-CH 自动转换时，在定时计数器下一次溢出时，A/D 转换器有足够的时间完成一笔数据转换。待下次定时计数器溢出时，开始转换下一笔数据，直到完成整个自动转换设定。



1-CH 自动转换时序图 – 1 (AADCTS[1:0]=10)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

下图在 1-CH 自动转换时，A/D 转换器没有足够时间将数据转换完毕，定时计数器又再次溢出，则 ATARPTG 会被置位为“1”，此时电路会继续将该笔数据转换完后，且等到又一次定时计数器溢出时，开始转换下一笔数据，直到完成整个自动转换设定。



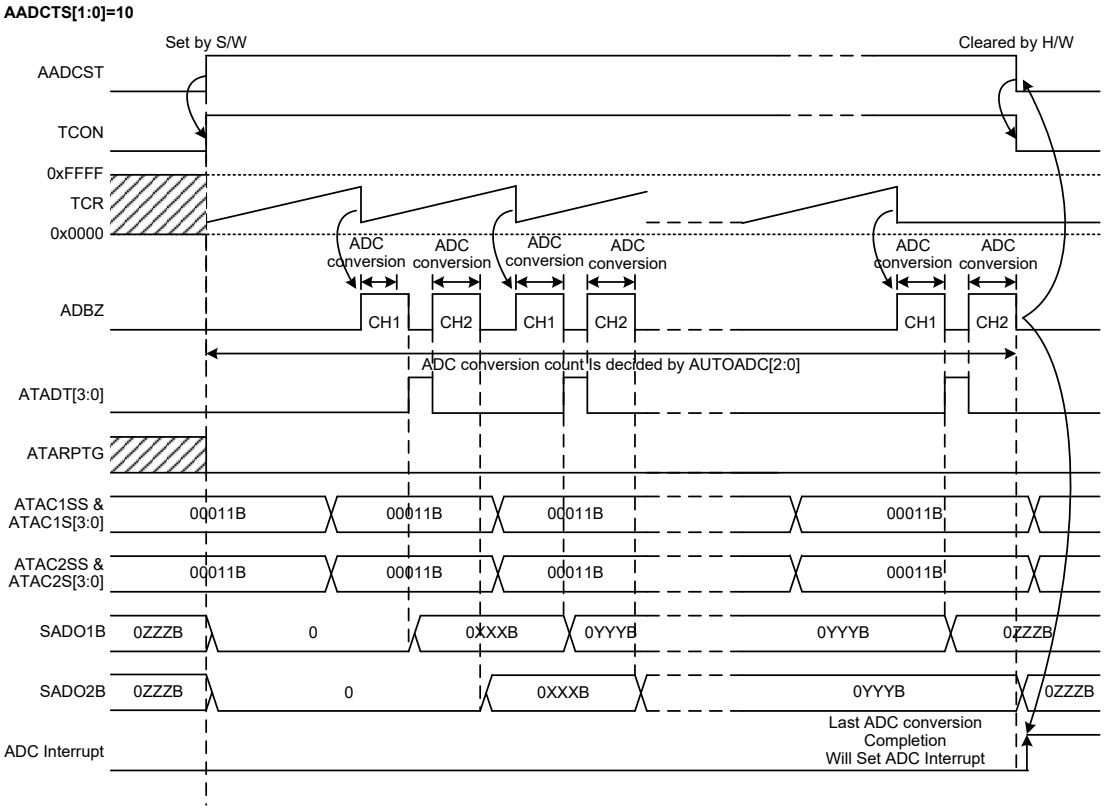
1-CH 自动转换时序图 – 2 (AADCTS[1:0]=10)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

2-CH 自动转换

当 AADCST 位由 0 变 1 时，会自动清除 ATARPTG 位，并将 TCON 位设置为“1”，使定时计数器从预载值开始计数，直到计数器数到最达值 FFFFH 溢出并从预载值开始重新向上计数，同时触发 ADC 转换开始。A/D 转换器先开始 CH1 转换，待 CH1 转换完成，延迟一段时间 (延迟时间由 ATADT[3:0] 设置)，开始 CH2 转换，待 CH2 转换完成后，等待下一次定时计数器溢出再开始转换。当所有转换次数达到设定值，硬件会自动将 AADCST 和 TCON 位清为“0”，并设置 A/D 中断标志位 ADF 为高。

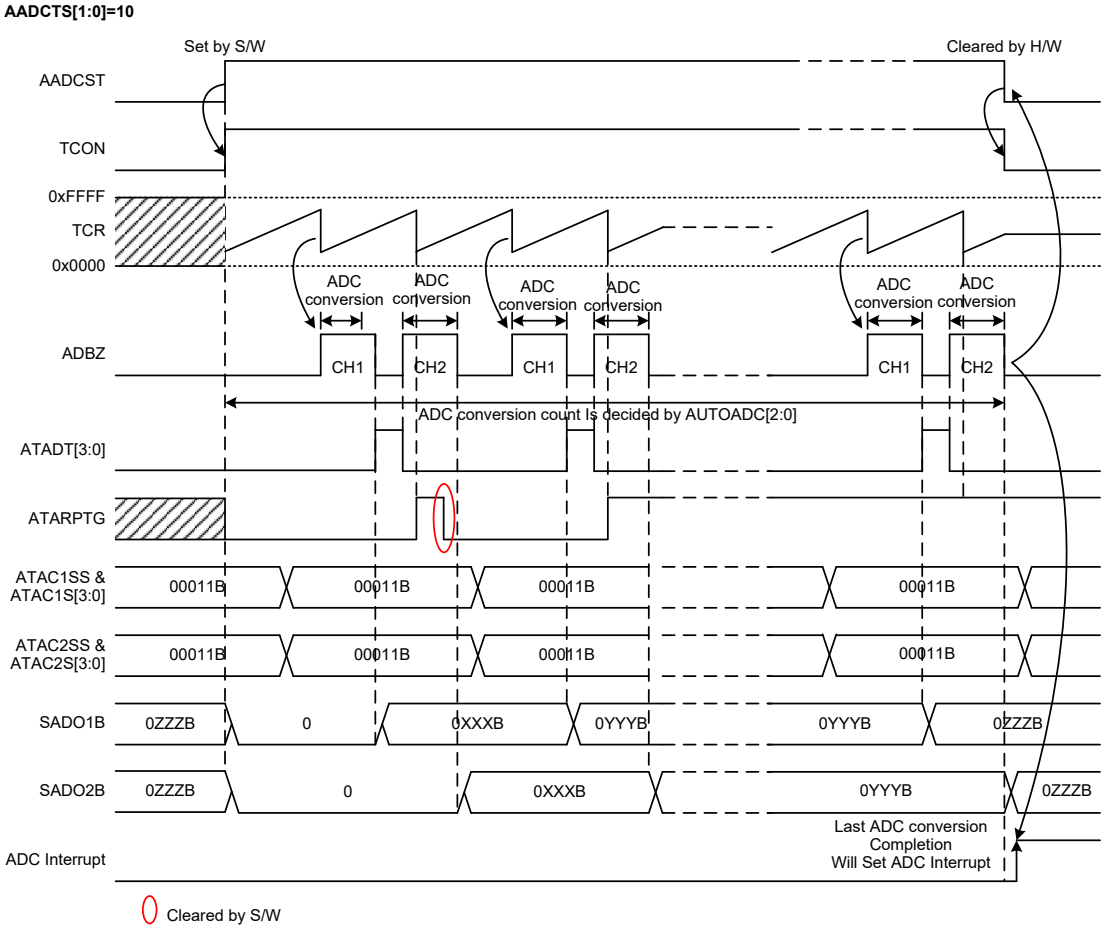
下图在定时计数器溢出时，A/D 转换器有足够的时间将两通道的数据转换完毕。待下次定时计数器溢出时，开始转换下一笔数据，直到完成整个自动转换设定。



2-CH 自动转换时序图 – 1 (AADCTS[1:0]=10)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

下图在 2-CH 自动转换时，A/D 转换器没有足够时间将数据转换完毕，定时计数器又再次溢出，则 ATARPTG 会被置位为“1”，此时电路会继续将该笔数据转换完后，且等到又一次定时计数器溢出时，开始转换下一笔数据，直到完成整个自动转换设定。



2-CH 自动转换时序图 – 2 (AADCTS[1:0]=10)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

AADCTS[1:0]=11B

若设置 AADCTS1~AADCTS0 位为 11，AUTOADC2~AUTOADC0 位不等于 000，则于软件设置 AADCST 位后使能 ADC 自动转换功能。自 AADCST 为“1”后，ADC 转换开始。直到 A/D 转换次数与 AUTOADC2~AUTOADC0 位所设置的次数相同时，会自动清除 AADCST 位，并置位 ADC 中断标志位。每笔 A/D 数据转换间隔为 ATADT3~ATADT0 位所设置的时间。

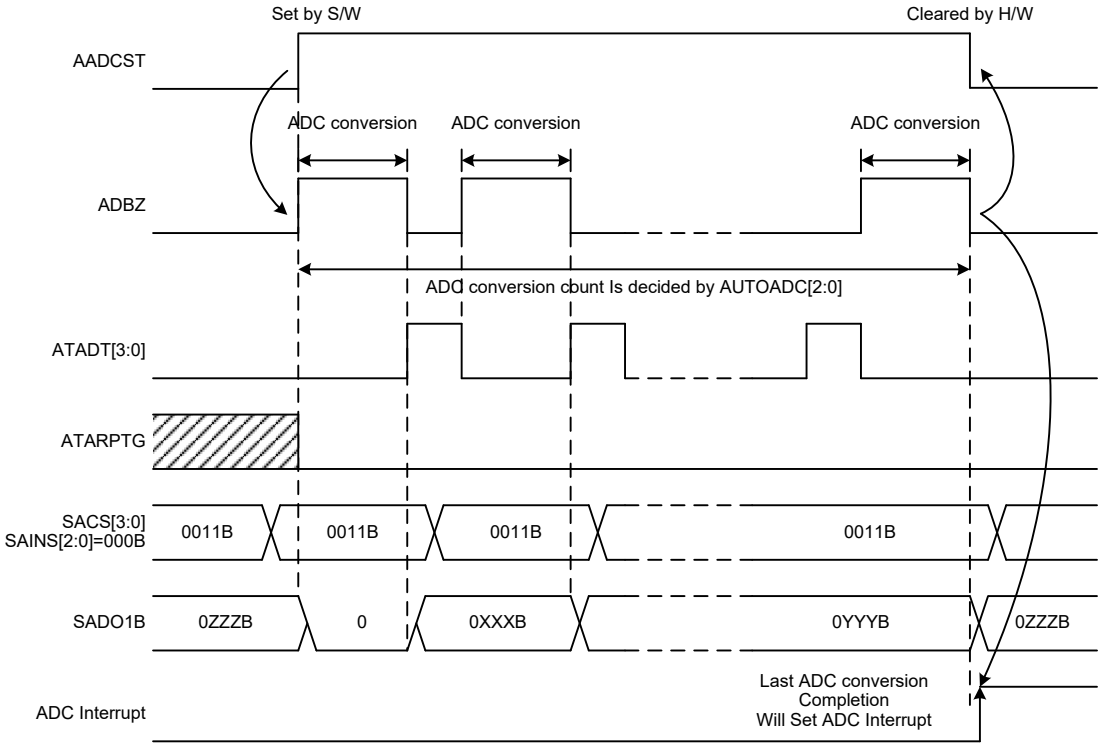
AADCST 为非重复触发，一旦设置后，将不可再次触发。也就是说，ADC 转换会依 AUTOADC2~AUTOADC0 设置的次数转换完毕才停止。程序设计时请注意，在 AADCST 清除后，请同时清除 ADF 标志位。

注：自动转换尚未完成，而 AADCST 被软件清零，若 ADBZ=1 则会待该次的 ADC 转换结束将 ADBZ 变为 0，若 ADBZ=0，则会将自动转换这部份电路复位至初始状态，待复位完成后，方可重新置位 AADCST 位以启动下一次的 A/D 自动转换。若 AADCST 被软件清零，A/D 转换数据已达设定笔数时，也会触发中断。

1-CH 自动转换

当 AADCST 位由 0 变 1 时，会自动清除 ATARPTG 位，A/D 开始转换。当转换次数达到设定值，硬件会自动将 AADCST 清为“0”，并设置 A/D 转换中断标志位 ADF 为 1。

AADCTS[1:0]=11



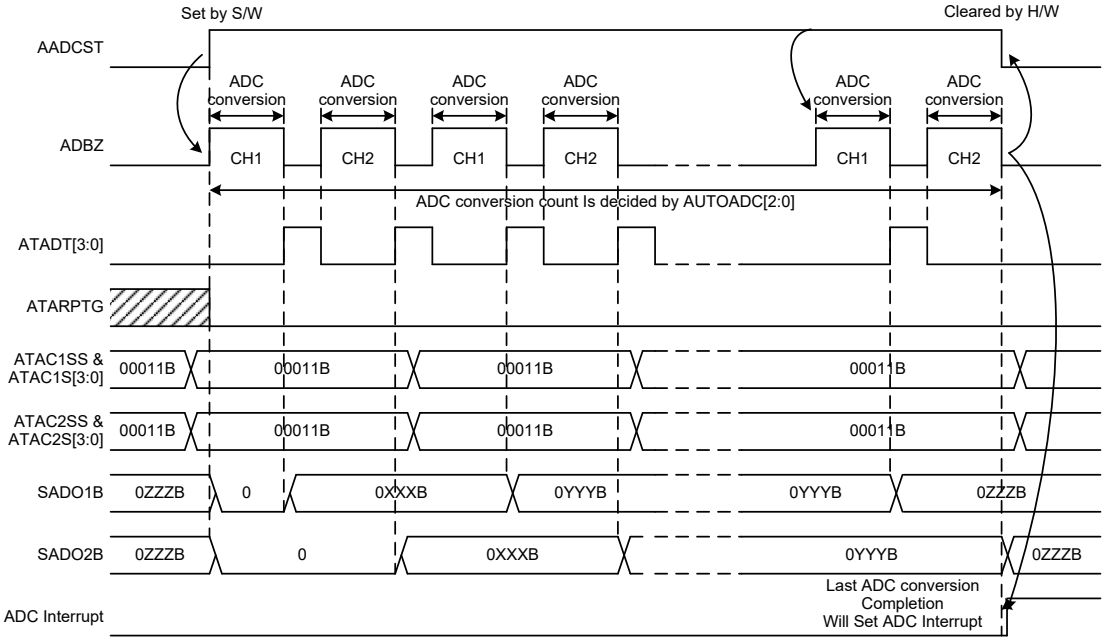
1-CH 自动转换时序图 (AADCTS[1:0]=11)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

2-CH 自动转换

当 AADCST 位由 0 变 1 时，会自动清除 ATARPTG 位，ADC 转换开始。先开始 CH1 转换，待 CH1 转换完成，延迟一段时间 (延迟时间由 ATADT[3:0] 设置)，开始 CH2 转换，待 CH2 转换完成，延迟一段时间 (延迟时间由 ATADT[3:0] 设置)，再开始 CH1 转换。当所有转换次数达到设定值，硬件会自动将 AADCST 清为“0”，并设立 A/D 中断标志位 ADF 为高。

AADCTS[1:0]=11



2-CH 自动转换时序图 (AADCTS[1:0]=11)

注：1. AUTOADC[2:0] ≠ 000
2. XXXB 为第一笔 A/D 转换值
3. YYYB 为累加后的 A/D 值
4. ZZZB 为自动 N 次转换后最终的累加值

A/D 自动转换软件停止说明

当 AADCTS[1:0]=00&11 时，自动转换尚未完成，AADCST 位被软件清零，若 ADBZ=1 则会待该次的 ADC 转换结束将 ADBZ 变为 0 之后才将自动转换这部份电路复位至初始状态，方可重新置位 AADCST 位以启动下一次的 A/D 自动转换。

当 AADCTS[1:0]=01&10 时，自动转换尚未完成，而 AADCST 被软件清零，若 ADBZ=1 请等待 ADBZ=0 后，将 ADC 切换为手动触发转换模式 (AUTOADC[2:0] =000)，并启动一次 ADC 转换 (START 位执行 0->1->0)，完成转换再切回自动转换模式，之后才将自动转换这部份电路复位至初始状态，待复位完成后，方可重新置位 AADCST 位以启动下一次的 A/D 自动转换。若 AADCST 被软件清零，A/D 转换数据已达设定笔数时，也会触发中断。

编程注意事项

在单片机工作时，如果 A/D 转换器未使用，通过设置 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

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

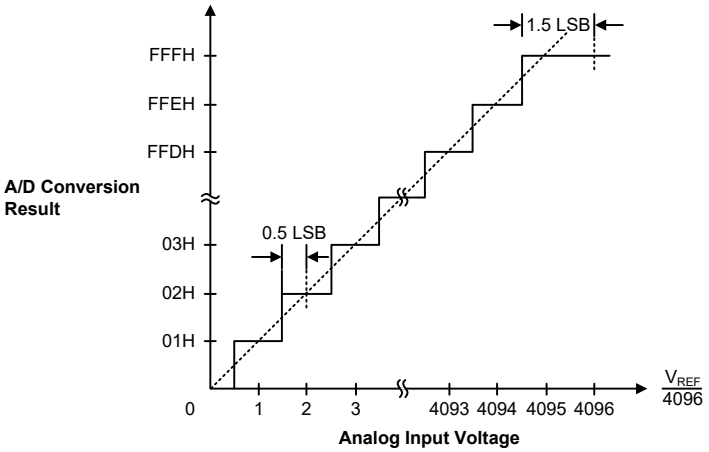
$$1\ LSB=V_{REF}/4096$$

通过下面的等式可估算 A/D 转换器输入电压值：

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

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

应注意的是此处的 V_{REF} 电压指的是 A/D 转换器参考电压源，由 SAVRS1~SAVRS0 位确定。



理想 A/D 转换功能

A/D 转换程序范例

下面两个范例程序用来说明怎样设置和实现手动触发启动 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 转换何时完成，而第二个范例则使用中断的方式判断。

范例 1：使用轮询 ADBZ 轮询的方式来检测转换结束

```
mov a,00h                ; disable A/D automatic conversion function
mov SADC2,a
clr ADE                  ; disable ADC interrupt
mov a,0Bh                ; select fSYS/8 as A/D clock and A/D input signal
                         ; comes from external channel
mov SADC1,a              ; select VDD as A/D reference voltage source
mov a,02h                ; setup PBS0 to configure pin AN0
mov PBS0,a
mov a,20h                ; enable A/D converter and select AN0 external
                         ; channel input
mov SADC0,a
:
start_conversion:
```

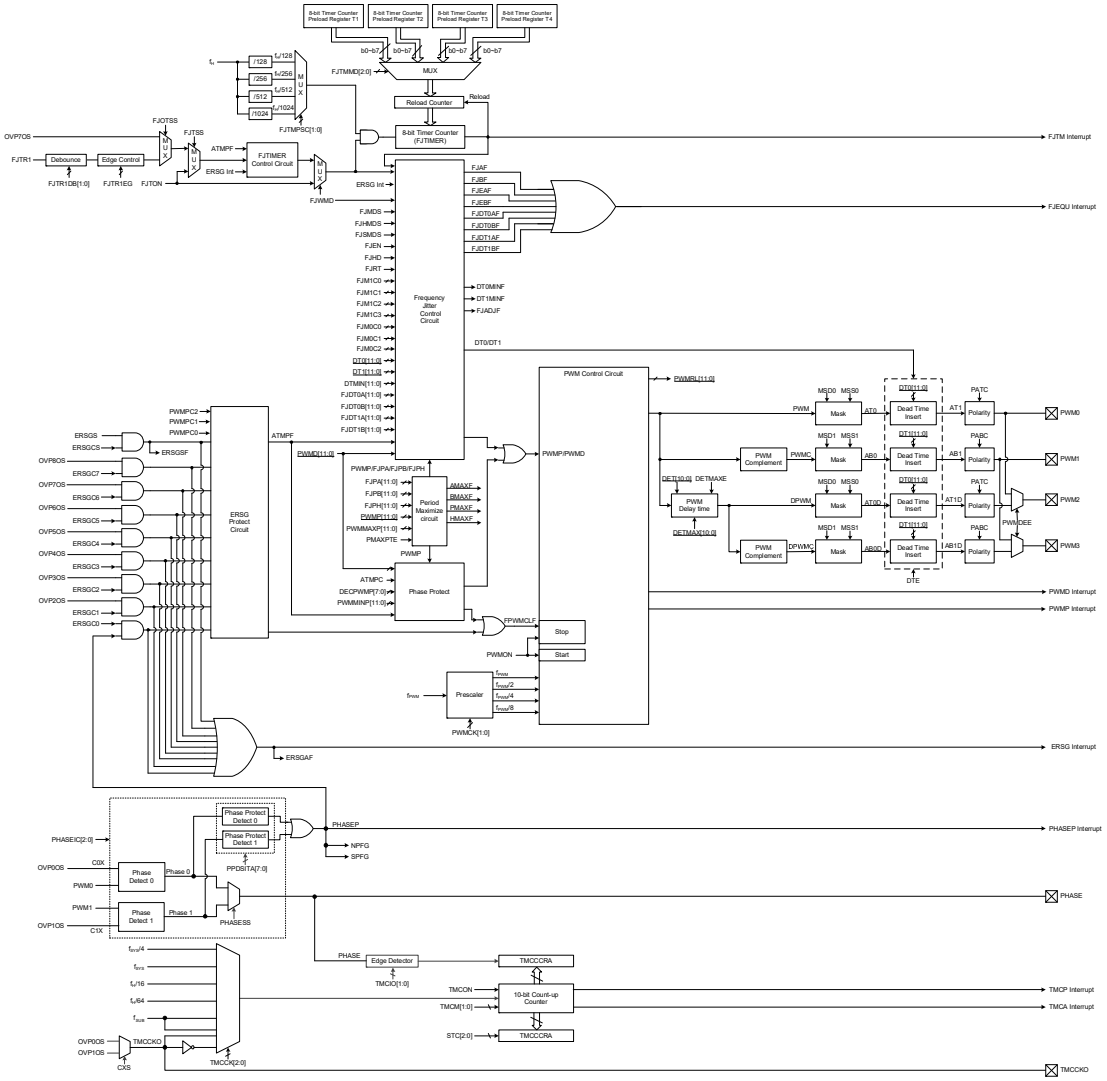
```
clr  START                  ; high pulse on start bit to initiate conversion
set  START                  ; reset A/D converter
clr  START                  ; start A/D conversion
polling_EOC:
sz   ADBZ                   ; poll the SADC0 register ADBZ bit to detect end
                            ; 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

```
mov  a,00h                  ; disable A/D automatic conversion function
mov  SADC2,a
clr  ADE                    ; disable ADC interrupt
mov  a,0Bh                  ; select fSYS/8 as A/D clock and A/D input signal
                            ; comes from external channel
mov  SADC1,a                ; select VDD as A/D reference voltage source
mov  a,02h                  ; setup PBS0 to configure pin AN0
mov  PBS0,a
mov  a,20h                  ; enable A/D converter and select AN0 external
                            ; channel input
mov  SADC0,a
:
Start_conversion:
clr  START                  ; high pulse on START bit to initiate conversion
set  START                  ; reset A/D converter
clr  START                  ; start A/D conversion
clr  ADF                    ; clear ADC interrupt request flag
set  ADE                    ; enable ADC interrupt
set  EMI                    ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov  acc_stack,a            ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a         ; save STATUS to user defined memory
:
:
mov  a,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
```

带死区时间的高精度 PWM

该单片机提供了一个完全内置的多功能互补式 PWM 输出产生器，可大大提高应用的灵活性。内含有两组互补式输出引脚，其中一组含有 10-bit 延迟时间设定。同时提供了完善的保护功能。当有错误发生时，可快速切断 PWM 输出。在此模块内也提供了一个 10-bit TMC 时间控制模块，可用于定时计数或对 Phase 信号进行输入捕捉。PWM 电路也包含了抖频功能，使能该功能可使 PWM 频率产生变化，从而降低 EMI 干扰。



注：OVPnOS 信号来自 OVPn 输出“OVPnCOUT”同相或反相去抖的信号，可通过软件选择。

PWM 产生器 / 保护电路 / 抖频功能方框图

寄存器描述

PWM 发生器功能、相位侦测、相位侦测保护、10-bit TMC、错误信号 (ERSG) 保护电路、抖频等功能的实施都是通过一系列寄存器来进行控制的。相关寄存器如下表所示。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PWMC0	MSS1	MSS0	PWMCK1	PWMCK0	PWMDEE	ATMPC	DTE	PWMON
PWMC1	FPWMCLF	ATMPF	PHASESS	CXS	PMAXPTE	ATMNPC	MSD1	MSD0
PWMC2	NPFG	PHASEIC2	PHASEIC1	PHASEIC0	SPFG	D2	ERSGS	DETMAXE
PLC	—	—	—	—	—	—	PABC	PATC
PWMPC0	PWMPC07	PWMPC06	PWMPC05	PWMPC04	PWMPC03	PWMPC02	PWMPC01	PWMPC00
PWMPC1	PWMPC17	PWMPC16	PWMPC15	PWMPC14	PWMPC13	PWMPC12	PWMPC11	PWMPC10
PWMPC2	ERSGAF	—	ERSGSF	ERSGCS	—	—	PWMPC21	PWMPC20
ERSGC	ERSGC7	ERSGC6	ERSGC5	ERSGC4	ERSGC3	ERSGC2	ERSGC1	ERSGC0
PWMPL	D7	D6	D5	D4	D3	D2	D1	D0
PWMPH	—	—	—	—	D11	D10	D9	D8
PWMDL	D7	D6	D5	D4	D3	D2	D1	D0
PWMDH	—	—	—	—	D11	D10	D9	D8
PWMRL	D7	D6	D5	D4	D3	D2	D1	D0
PWMRH	—	—	—	—	D11	D10	D9	D8
DT0L	D7	D6	D5	D4	D3	D2	D1	D0
DT0H	—	—	—	—	D11	D10	D9	D8
DT1L	D7	D6	D5	D4	D3	D2	D1	D0
DT1H	—	—	—	—	D11	D10	D9	D8
DETL	D7	D6	D5	D4	D3	D2	D1	D0
DETH	—	—	—	—	—	D10	D9	D8
PWMMAXPL	D7	D6	D5	D4	D3	D2	D1	D0
PWMMAXPH	—	—	—	—	D11	D10	D9	D8
PWMMINPL	D7	D6	D5	D4	D3	D2	D1	D0
PWMMINPH	—	—	—	—	D11	D10	D9	D8
DETMAXL	D7	D6	D5	D4	D3	D2	D1	D0
DETMAXH	—	—	—	—	—	D10	D9	D8
DECPWMP	D7	D6	D5	D4	D3	D2	D1	D0
TMCC0	—	TMCCK2	TMCCK1	TMCCK0	TMCON	STC2	STC1	STC0
TMCC1	TMCM1	TMCM0	TMCIO1	TMCIO0	—	—	—	—
TMCDL	D7	D6	D5	D4	D3	D2	D1	D0
TMCDH	—	—	—	—	—	—	D9	D8
TMCCCRAL	D7	D6	D5	D4	D3	D2	D1	D0
TMCCCRAH	—	—	—	—	—	—	D9	D8
PPDSITA	D7	D6	D5	D4	D3	D2	D1	D0
FJTMC	—	FJTMMD2	FJTMMD1	FJTMMD0	FJTON	—	FJTMPSC1	FJTMPSC0
FJTMR1	D7	D6	D5	D4	D3	D2	D1	D0
FJTMR2	D7	D6	D5	D4	D3	D2	D1	D0
FJTMR3	D7	D6	D5	D4	D3	D2	D1	D0
FJTMR4	D7	D6	D5	D4	D3	D2	D1	D0
FJTMD	D7	D6	D5	D4	D3	D2	D1	D0
FJPAL	D7	D6	D5	D4	D3	D2	D1	D0
FJPAH	—	—	—	—	D11	D10	D9	D8
FJPBL	D7	D6	D5	D4	D3	D2	D1	D0
FJPBH	—	—	—	—	D11	D10	D9	D8
FJPHL	D7	D6	D5	D4	D3	D2	D1	D0
FJPHH	—	—	—	—	D11	D10	D9	D8
DTMINL	D7	D6	D5	D4	D3	D2	D1	D0
DTMINH	—	—	—	—	D11	D10	D9	D8
FJDT0AL	D7	D6	D5	D4	D3	D2	D1	D0
FJDT0AH	—	—	—	—	D11	D10	D9	D8
FJDT0BL	D7	D6	D5	D4	D3	D2	D1	D0

寄存器名称	位							
	7	6	5	4	3	2	1	0
FJDT0BH	—	—	—	—	D11	D10	D9	D8
FJDT1AL	D7	D6	D5	D4	D3	D2	D1	D0
FJDT1AH	—	—	—	—	D11	D10	D9	D8
FJDT1BL	D7	D6	D5	D4	D3	D2	D1	D0
FJDT1BH	—	—	—	—	D11	D10	D9	D8
FJC0	FJWMD	FJEN	FJMDS	FJSMDS	FJHMDS	FJTSS	FJOTSS	FJADJF
FJC1	FJEAF	FJEBF	FJAF	FJBF	—	—	FJHD	FJRT
FJC2	FJDT0AF	FJDT0BF	FJDT1AF	FJDT1BF	—	FJTR1EG	FJTR1DB1	FJTR1DB0
FJF0	AMAXF	BMAXF	HMAXF	PMAXF	—	—	DT1MINF	DT0MINF
FJM1C0	—	FJCNT2	FJCNT1	FJCNT0	—	FJSA2	FJSA1	FJSA0
FJM1C1	—	FJTIMES2	FJTIMES1	FJTIMES0	FJACNT1	FJACNT0	FJASA1	FJASA0
FJM1C2	FJDT0EN	FJDT0ASA1	FJDT0ASA0	FJDT0CNT1	FJDT0CNT0	FJDT0SA2	FJDT0SA1	FJDT0SA0
FJM1C3	FJDT1EN	FJDT1ASA1	FJDT1ASA0	FJDT1CNT1	FJDT1CNT0	FJDT1SA2	FJDT1SA1	FJDT1SA0
FJM0C0	—	CFJCNT2	CFJCNT1	CFJCNT0	CFJS	CFJSA2	CFJSA1	CFJSA0
FJM0C1	CFJDT0EN	—	CFJDT0CNT1	CFJDT0CNT0	CFJDT0S	—	CFJDT0SA1	CFJDT0SA0
FJM0C2	CFJDT1EN	—	CFJDT1CNT1	CFJDT1CNT0	CFJDT1S	—	CFJDT1SA1	CFJDT1SA0

寄存器列表

- **PWMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MSS1	MSS0	PWMCK1	PWMCK0	PWMDEE	ATMPC	DTE	PWMON
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 **MSS1**：PWM1/PWM3 输出信号选择
0：输出 PWMC/DPWMC 信号 (PWM 信号)
1：输出 MSD1 (软件信号)
注：PWM0 和 PWM1 其中一个为 PWM 信号输出，则另外一个不可为软件信号，硬件会自动将软件信号改为 PWM 信号 (MSS1=0 或 MSS0=0)，PWM2 和 PWM3 同理。

Bit 6 **MSS0**：PWM0/PWM2 输出信号选择
0：输出 PWM/DPWM 信号 (PWM 信号)
1：输出 MSD0 (软件信号)
注：PWM0 和 PWM1 其中一个为 PWM 信号输出，则另外一个不可为软件信号，硬件会自动将软件信号改为 PWM 信号 (MSS1=0 或 MSS0=0)，PWM2 和 PWM3 同理。

Bit 5~4 **PWMCK1~PWMCK0**：选择 PWM 计数器时钟 f_{PWMPSC}
00：f_{PWM}
01：$f_{PWM}/2$
10：$f_{PWM}/4$
11：$f_{PWM}/8$

Bit 3 **PWMDEE**：PWM 移相角度控制位
0：除能
1：使能

Bit 2 **ATMPC**：停止自动调变周期控制位
0：无影响
1：停止自动调变周期
注：1. 当 PWMPCn[m+1:m] 位为 00 或 01 时，设置此位为无效动作，请勿将此位设 1。
2. 当 PWMPCn[m+1:m] 位为 10 或 11 时，若对应的保护信号消失时，设置 ATMPC 位为 1，PWM 会在完成当前周期后停止自动调变。

3. PWM 停止自动调变功能后，硬件自动将此位清零。
4. 详细说明请参考“ATMPC 设置说明”。

Bit 1 **DTE**：死区时间使能控制位
0：除能
1：使能
该位为高时，PWMON 位也要为高，才可使能死区时间插入功能。

Bit 0 **PWMON**：PWM 电路 on/off 控制位
0：Off
1：On
该位是 PWM 功能总开 / 关控制位。设置此位为高则使能 PWM 计数器使其运行，清零此位则除能 PWM。清零此位将会使正在计数的计数器停止计数并关闭 PWM 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。
当 PWMON 位为零以及 MSS1 或 MSS0 位其中一个为零，PWM0/PWM1/PWM2/PWM3 输出浮空。

PWM 输出与相关位设置的关系表如下所示。

相关位设置				PWM 输出			
PWMON	**MSS1**	**MSS0**	**PWMDEE**	**PWM3**	**PWM2**	**PWM1**	**PWM0**
0	0	0	x	浮空	浮空	浮空	浮空
0	0	1	x	浮空	浮空	浮空	浮空
0	1	0	x	浮空	浮空	浮空	浮空
0	1	1	x	MSD1	MSD0	MSD1	MSD0
1	0	0	0	PWMC	PWM	PWMC	PWM
1	0	1	0	PWMC	PWM	PWMC	PWM
1	1	0	0	PWMC	PWM	PWMC	PWM
1	0	0	1	DPWMC	DPWM	PWMC	PWM
1	0	1	1	DPWMC	DPWM	PWMC	PWM
1	1	0	1	DPWMC	DPWM	PWMC	PWM
1	1	1	x	MSD1	MSD0	MSD1	MSD0

“x”：无关

- **PWMC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	FPWMCLF	ATMPF	PHASESS	CXS	PMAXPTE	ATMNPC	MSD1	MSD0
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 **FPWMCLF**：PWM 自动关闭标志位
0：PWM 未自动关闭
1：PWM 已自动关闭
该位只能软件清零，不能写 1。

Bit 6 **ATMPF**：PWM 周期自动调变标志位
0：未启动自动调变功能
1：已启动自动调变功能
该位只能软件清零，不能写 1。当 PWMON 由 0 变 1 时，硬件会自动将此位清零。

Bit 5 **PHASESS**：选择要侦测的相位信号
0：Phase0
1：Phase1

Bit 4 **CXS**：选择 TMCCKO 信号来源
0：C0X (OVP0OS)
1：C1X (OVP1OS)
OVP0OS 和 OVP1OS 分别为过电压保护电路 0 和过电压保护电路 1 输出信号。

Bit 3 **PMAXPTE**：PWMP 最大值限制控制
0：无最大值限制，可写入任意值
1：写入最大值 = PWMMAXP
注：当 PMAXPTE 位为“0”时，写入 PWMP/FJPA/FJPB/FJPH 的值不受限于 PWMMAXP；当 PMAXPTE 位为“1”时，写入 PWMP/FJPA/FJPB/FJPH 的值会受限于 PWMMAXP。要写入的值若大于 PWMMAXP，则无法写入到对应的寄存器中。若在 PMAXPTE 位从 0 变为 1 之前，PWMP/FJPA/FJPB/FJPH 中的值大于等于 PWMMAXP，且 PWMP/FJPA/FJPB/FJPH 没有写入动作，硬件不会去更改这些寄存器的值。
下表以向 PWMP 写值作为范例说明：

步骤	PMAXPTE	PWMMAXP	PWMP (写入)	PWMP (结果)
1	0	1000	500	500
2	0	1000	1024	1024
3	1	1000	—	1024
4	1	1000	1010	1024
5	1	1000	980	980
6	1	1000	1010	980

Bit 2 **ATMNPC**：自动调变过程中对应 ERSG 信号变 0 则停止自动调变 (仅对 ERSG1 及 ERSG3 信号有效)
0：无操作
1：当 ERSG 信号由 1 变为 0，停止自动调变周期
注：当 ERSG1 或 ERSG3 信号为 1 且相对应的 PWMPCn[m+1:m] 位为 11 时，若 ATMNPC 位为 1 且 ERSG1 或 ERSG3 信号由 1 变 0 时，会依 ATMPC 置高时的动作停止自动调变功能。

Bit 1 **MSD1**：PWM1/PWM3 Mask 数据位
0：逻辑低
1：逻辑高
当 MSS0 与 MSS1 位都为 1 时，PWM0、PWM1、PWM2 和 PWM3 同时输出 MSD0 与 MSD1 软件信号。然而需注意 PWM0 和 PWM1 输出信号不可同时为高，若同时为高则硬件会强制其同时输出“0”。PWM2 和 PWM3 同理。

Bit 0 **MSD0**：PWM0/PWM2 Mask 数据位
0：逻辑低
1：逻辑高
当 MSS0 与 MSS1 位都为 1 时，PWM0、PWM1、PWM2 和 PWM3 同时输出 MSD0 与 MSD1 软件信号。然而需注意 PWM0 和 PWM1 输出信号不可同时为高，若同时为高则硬件会强制其同时输出“0”。PWM2 和 PWM3 同理。

- **PWMC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	NPFG	PHASEIC2	PHASEIC1	PHASEIC0	SPFG	D2	ERSGS	DETMAXE
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 **NPFG**：未检测到 Phase 宽度标志位
0：检测到 Phase 宽度
1：未检测到 Phase 宽度
该位只能软件清零，不能写 1。

Bit 6~4 **PHASEIC2~PHAEIC0**：相位侦测 & 相位保护侦测使能控制位
000：PWMON 为“1”后，立即启动相位侦测
001：PWMON 为“1”后，等待 1 个 PWM 周期后启动相位侦测
010：PWMON 为“1”后，等待 2 个 PWM 周期后启动相位侦测
011：PWMON 为“1”后，等待 3 个 PWM 周期后启动相位侦测
100：PWMON 为“1”后，等待 4 个 PWM 周期后启动相位侦测
101：PWMON 为“1”后，等待 5 个 PWM 周期后启动相位侦测
110：PWMON 为“1”后，等待 6 个 PWM 周期后启动相位侦测
111：PWMON 为“1”后，等待 7 个 PWM 周期后启动相位侦测
当 PWMON 由 0 变为 1 时，开始计数 PWM 周期。计数到此三位设置的 PWM 周期数量后，停止计数并启动相位侦测。当 PWMON 位再次变为 0 时，计数值清为零。

Bit 3 **SPFG**：Phase 宽度过小标志位
0：Phase 宽度 > PPDSITA 设定值
1：Phase 宽度 < PPDSITA 设定值
该位只能软件清零，不能写 1。

Bit 2 **D2**：保留，必须固定为“0”

Bit 1 **ERSGS**：错误信号软件产生位
0：逻辑低
1：逻辑高
注：当 ERSGCS 位为 1 时，该位才有效。通过设置该位为高，可启动对应的 ERSG 保护电路，根据 PWMPC2[1:0] 位的设置控制 PWM 调变或关闭。

Bit 0 **DETMAXE**：DET 最大值限制控制
0：无最大值限制，可写入任意值
1：写入最大值 = DETMAX
注：当 DETMAXE 位为“0”时，写入 DET 的值不受限于 DETMAX；当 DETMAXE 位为“1”时，写入 DET 的值会受限于 DETMAX。要写入的值若大于 DETMAX，则无法写入到对应的寄存器中。若在 DETMAXE 位从 0 变为 1 之前，DET 中的值大于等于 DETMAX，且 DET 没有写入动作，硬件不会去更改这些寄存器的值。
下表以向 DET 写值作为范例说明：

步骤	DETMAXE	DETMAX	DET (写入)	DET (结果)
1	0	1000	500	500
2	0	1000	1024	1024
3	1	1000	—	1024
4	1	1000	1010	1024
5	1	1000	980	980
6	1	1000	1010	980

● ERSGC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ERSGC7	ERSGC6	ERSGC5	ERSGC4	ERSGC3	ERSGC2	ERSGC1	ERSGC0
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 **ERSGC7**：错误信号 7 (ERSG7) 使能 ERSG 保护电路控制
0：除能
1：使能
ERSG7 信号为 OVP8 输出信号 (OVP8OS)。

Bit 6 **ERSGC6**：错误信号 6 (ERSG6) 使能 ERSG 保护电路控制
0：除能
1：使能
ERSG6 信号为 OVP7 输出信号 (OVP7OS)。

Bit 5 **ERSGC5**：错误信号 5 (ERSG5) 使能 ERSG 保护电路控制
0：除能
1：使能
ERSG5 信号为 OVP6 输出信号 (OVP6OS)。

Bit 4 **ERSGC4**：错误信号 4 (ERSG4) 使能 ERSG 保护电路控制
0：除能
1：使能
ERSG4 信号为 OVP5 输出信号 (OVP5OS)。

Bit 3 **ERSGC3**：错误信号 3 (ERSG3) 使能 ERSG 保护电路控制
0：除能
1：使能
ERSG3 信号为 OVP4 输出信号 (OVP4OS)。

Bit 2 **ERSGC2**：错误信号 2 (ERSG2) 使能 ERSG 保护电路控制
0：除能
1：使能
ERSG2 信号为 OVP3 输出信号 (OVP3OS)。

Bit1 **ERSGC1**：错误信号 1 (ERSG1) 使能 ERSG 保护电路控制
0：除能
1：使能
ERSG1 信号为 OVP2 输出信号 (OVP2OS)。

Bit 0 **ERSGC0**：错误信号 0 (ERSG0) 使能 ERSG 保护电路控制
0：除能
1：使能
ERSG0 信号为相位保护侦测电路输出的 PHASEP 信号。

- **PWMPC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PWMPC07	PWMPC06	PWMPC05	PWMPC04	PWMPC03	PWMPC02	PWMPC01	PWMPC00
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 **PWMPC07~PWMPC06**：ERSG3 (OVP4OS) PWM 保护控制选择位
00：PWM 立即关闭
01：PWM 不立即关闭，待执行完当前整个周期后再关闭
10：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，之后 PWM 关闭
11：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，PWM 不关闭
有关 PWM 保护动作的详细介绍，请参考“ERSG 保护电路”部分内容。

Bit 5~4 **PWMPC05~PWMPC04**：ERSG2 (OVP3OS) PWM 保护控制选择位
00：PWM 立即关闭
01：PWM 不立即关闭，待执行完当前整个周期后再关闭
10：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，之后 PWM 关闭
11：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，PWM 不关闭
有关 PWM 保护动作的详细介绍，请参考 “ERSG 保护电路”部分内容。

Bit 3~2 **PWMPC03~PWMPC02**：ERSG1 (OVP2OS) PWM 保护控制选择位
00：PWM 立即关闭
01：PWM 不立即关闭，待执行完当前整个周期后再关闭
10：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，之后 PWM 关闭
11：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，PWM 不关闭
有关 PWM 保护动作的详细介绍，请参考 “ERSG 保护电路”部分内容。

Bit 1~0 **PWMPC01~PWMPC00**：ERSG0 (PHASEP) PWM 保护控制选择位
00：PWM 立即关闭
01：PWM 不立即关闭，待执行完当前整个周期后再关闭
10：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，之后 PWM 关闭
11：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，PWM 不关闭
有关 PWM 保护动作的详细介绍，请参考“ERSG 保护电路”部分内容。

● **PWMPC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PWMPC17	PWMPC16	PWMPC15	PWMPC14	PWMPC13	PWMPC12	PWMPC11	PWMPC10
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 **PWMPC17~PWMPC16**：ERSG7 (OVP8OS) PWM 保护控制选择位
00：PWM 立即关闭
01：PWM 不立即关闭，待执行完当前整个周期后再关闭
10：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，之后 PWM 关闭
11：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，PWM 不关闭
有关 PWM 保护动作的详细介绍，请参考“ERSG 保护电路”部分内容。

Bit 5~4 **PWMPC15~PWMPC14**：ERSG6 (OVP7OS) PWM 保护控制选择位
00：PWM 立即关闭
01：PWM 不立即关闭，待执行完当前整个周期后再关闭
10：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，之后 PWM 关闭
11：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，PWM 不关闭
有关 PWM 保护动作的详细介绍，请参考“ERSG 保护电路”部分内容。

Bit 3~2 **PWMPC13~PWMPC12**：ERSG5 (OVP6OS) PWM 保护控制选择位
00：PWM 立即关闭
01：PWM 不立即关闭，待执行完当前整个周期后再关闭
10：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，之后 PWM 关闭
11：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，PWM 不关闭
有关 PWM 保护动作的详细介绍，请参考“ERSG 保护电路”部分内容。

Bit 1~0 **PWMPC11~PWMPC10**：ERSG4 (OVP5OS) PWM 保护控制选择位
00：PWM 立即关闭
01：PWM 不立即关闭，待执行完当前整个周期后再关闭
10：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，之后 PWM 关闭
11：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，PWM 不关闭
有关 PWM 保护动作的详细介绍，请参考“ERSG 保护电路”部分内容。

● **PWMPC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ERSGAF	—	ERSGSF	ERSGCS	—	—	PWMPC21	PWMPC20
R/W	R	—	R/W	R/W	—	—	R/W	R/W
POR	0	—	0	0	—	—	0	0

Bit 7 **ERSGAF**：保护发生标志位
0：无保护信号
1：有保护信号
注：1. ERSGAF 为 ERSG[7:1]、ERSGS 及 PHASEP INT 与其相应的使能位 AND 后的 OR 运算结果。当使能位为“1”的输入信号只要有一个信号有 0→1 的变化时，ERSGAF 就会被硬件设定为 1，当保护信号消失硬件会将 ERSGAF 清 0。
2. 当 ERSGAF=1 (有保护信号) 时，PWMPCn[m+1:m] 不可被软件修改 (n=0~2; m=0, 2, 4, 6)。

Bit 6 未定义，读为“0”

Bit 5 **ERSGSF**：通过 ERSGS 软件位使能 ERSG 保护电路标志位
0：没有发生通过软件位启动的 ERSG 保护
1：通过设置 ERSGS 位启动了相应的 ERSG 保护
该位只能软件清零，不能写 1。

Bit 4 **ERSGCS**：允许通过错误信号软件产生位 (ERSGS) 使能 ERSG 保护电路控制
0：除能
1：使能

Bit 3~2 未定义，读为“0”

Bit 1~0 **PWMPC21~PWMPC20**：ERSGS (软件产生位) PWM 保护控制选择位
00：PWM 立即关闭
01：PWM 不立即关闭，待执行完当前整个周期后再关闭
10：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，之后 PWM 关闭
11：PWM 会在每一个 PWM 周期调变，直到 PWMP 小于 PWMMINP，PWM 不关闭
有关 PWM 保护动作的详细介绍，请参考 “ERSG 保护电路”部分内容。

● **PLC 寄存器**

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

Bit 7~2 未定义，读为“0”

Bit 1 **PABC**：PWM1/PWM3 输出极性控制
0：同相
1：反相

Bit 0 **PATC**：PWM0/PWM2 输出极性控制
0：同相
1：反相

● **PWMPL 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**：12-bit PWM 周期缓冲寄存器低字节
PWM 12-bit PWMP bit 7 ~ bit 0

● **PWMPH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”
Bit 3~0 **D11~D8**：12-bit PWM 周期缓冲寄存器高字节
PWM 12-bit PWMP bit 11 ~ bit 8
PWM 周期 = f_{PWMPSC}/(PWMP[11:0]+1)，其中 f_{PWMPSC} 为 PWM 计数时钟，通过 PWMCK1~PWMCK0 位选择。

● **PWMDL 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**：12-bit PWM 占空比缓冲寄存器低字节
PWM 12-bit PWMD bit 7 ~ bit 0

● **PWMDH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”
Bit 3~0 **D11~D8**：12-bit PWM 占空比缓冲寄存器高字节
PWM 12-bit PWMD bit 11 ~ bit 8
PWMD 值需满足条件：1 ≤ PWMD ≤ (PWMP - 1)

● **PWMRL 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**：12-bit PWM 计数器寄存器低字节
PWM 12-bit PWMR bit 7 ~ bit 0

● **PWMRH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R	R	R	R
POR	—	—	—	—	0	0	0	0

Bit 7~4　未定义，读为“0”

Bit 3~0　**D11~D8**：12-bit PWM 计数器寄存器高字节
PWM 12-bit PWMR bit 11 ~ bit 8

● **DT0L 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**：12-bit PWM0/PWM2 死区时间缓冲寄存器低字节
PWM0/PWM2 12-bit DT0 bit 7 ~ bit 0

● **DT0H 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未定义，读为“0”

Bit 3~0　**D11~D8**：12-bit PWM0/PWM2 死区时间缓冲寄存器高字节
PWM0/PWM2 12-bit DT0 bit 11 ~ bit 8
PWM0/PWM2 死区时间 = (DT0[11:0]+1)/f_{DT}，其中 $f_{DT}=f_{PWM}$
当 DT0 设定大于 PWMD 时，计数最大值为 PWMD。

● **DT1L 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**：12-bit PWM1/PWM3 死区时间缓冲寄存器低字节
PWM1/PWM3 12-bit DT1 bit 7 ~ bit 0

● **DT1H 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未定义，读为“0”

Bit 3~0　**D11~D8**：12-bit PWM1/PWM3 死区时间缓冲寄存器高字节
PWM1/PWM3 12-bit DT1 bit 11 ~ bit 8
PWM1/PWM3 死区时间 = (DT1[11:0]+1)/f_{DT}，其中 $f_{DT}=f_{PWM}$
当 DT1 设定大于 PWMP-PWMD 时，计数最大值为 PWMP-PWMD。

● **DETL 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**：11-bit PWM2/PWM3 延迟时间缓冲寄存器低字节
PWM2/PWM3 11-bit DET bit 7 ~ bit 0

● **DETH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	D10	D9	D8
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 未定义，读为“0”
Bit 2~0 **D10~D8**：11-bit PWM2/PWM3 延迟时间缓冲寄存器高字节
PWM2/PWM3 11-bit DET bit 10 ~ bit 8
PWM2/PWM3 延迟时间长度 = (DET[10:0]+1)/f_{DT}，其中 $f_{DT}=f_{PWM}$

● **DETMAXL 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**：11-bit PWM2/PWM3 最大延迟时间值寄存器低字节
11-bit DETMAX bit 7 ~ bit 0

● **DETMAXH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	D10	D9	D8
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 未定义，读为“0”
Bit 2~0 **D10~D8**：11-bit PWM2/PWM3 最大延迟时间值寄存器高字节
11-bit DETMAX bit 10 ~ bit 8

● **PWMMAXPL 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**：12-bit PWM 最大周期值寄存器低字节
PWM 12-bit PWMMAXP bit 7 ~ bit 0

● **PWMMAXPH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未定义，读为“0”
Bit 3~0　**D11~D8**：12-bit PWM 最大周期值寄存器高字节
PWM 12-bit PWMMAXP bit 11 ~ bit 8

● **PWMMINPL 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**：12-bit PWM 最小周期值寄存器低字节
PWM 12-bit PWMMINP bit 7 ~ bit 0

● **PWMMINPH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未定义，读为“0”
Bit 3~0　**D11~D8**：12-bit PWM 最小周期值寄存器高字节
PWM 12-bit PWMMINP bit 11 ~ bit 8

● **DECPWMP 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**：8-bit PWM 周期递减值寄存器
8-bit DECPWMP 寄存器 bit 7 ~ bit 0

● **TMCC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	TMCCK2	TMCCK1	TMCCK0	TMCON	STC2	STC1	STC0
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 6~4　**TMCCK2~TMCCK0**：TMC 计数器时钟选择
000：$f_{SYS}/4$
001：f_{SYS}
010：$f_H/16$
011：$f_H/64$
100：f_{SUB}

101：f_{SUB}
110：TMCCKO 上升沿
111：TMCCKO 下降沿

此三位用于选择 TMC 的计数时钟源。内部 TMCCKO 信号作为时钟源时，能选择在上升沿或下降沿有效。f_{SYS} 是系统时钟，f_H 和 f_{SUB} 是其它的时钟源，细节方面请参考工作模式和系统时钟章节。

Bit 3 **TMCON**：TMC 计数器 On/Off 控制位
0：Off
1：On

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

Bit 2~0 **STC2~STC0**：选择定时时间
000：f_{PWM}/8192
001：f_{PWM}/16384
010：f_{PWM}/32768
011：f_{PWM}/65536
100：f_{PWM}/131072
101：f_{PWM}/262144
110：f_{PWM}/524288
111：f_{PWM}/1048576

f_{PWM} 时钟频率为 32MHz。此三位可设定 0.25ms、0.5ms、1ms、2ms、4ms、8ms、16ms 及 32ms 的定时时间。当设定的时间到时，10-bit TMC 计数器值将被储存到 TMCCCRA 寄存器，并且将 TMCA 中断标志位置高。

● TMCC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TMCM1	TMCM0	TMCIO1	TMCIO0	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

Bit 7~6 **TMCM1~TMCM0**：选择 TMC 工作模式
00/01：捕捉输入模式
10/11：定时 / 计数器模式

这两位设置 10-bit TMC 将工作在捕捉输入模式或是定时 / 计数器模式。

Bit 5~4 **TMCIO1~TMCIO0**：选择 TMC 功能 (捕捉输入模式)
00：在输入信号 (PHASE 信号) 上升沿捕捉
01：在输入信号 (PHASE 信号) 下降沿捕捉
10：在输入信号 (PHASE 信号) 双沿捕捉
11：输入捕捉除能

10-bit TMC 工作在定时 / 计数器模式时，这两位无效。

Bit 3~0 未定义，读为“0”

● TMCDL 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**：10-bit TMC 计数器寄存器低字节
TMC 10-bit 计数器 bit 7 ~ bit 0

● **TMCDH 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**：10-bit TMC 计数器寄存器高字节
TMC 10-bit 计数器 bit 9 ~ bit 8

● **TMCCCRAL 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**：10-bit TMCCCRA 寄存器低字节
10-bit TMCCCRA bit 7 ~ bit 0

● **TMCCCRAH 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**：10-bit TMCCCRA 寄存器高字节
10-bit TMCCCRA bit 9 ~ bit 8

● **PPDSITA 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**：相位侦测保护 Sita 寄存器 bit 7 ~ bit 0
$\Delta time=(PPDSITA[7:0]+1) \times 1/f_{PWM}$

● **FJTMC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	FJTMMD2	FJTMMD1	FJTMMD0	FJTON	—	FJTMPSC1	FJTMPSC0
R/W	—	R	R	R	R/W	—	R/W	R/W
POR	—	0	0	0	0	—	0	0

Bit 7　未定义，读为“0”
Bit 6~4　**FJTMMD2~FJTMMD0**：FJTIMER 工作区间指示
000：T1 工作区间 (t0~t1)
001：T2 工作区间 (t1~t2)
010：T3 工作区间 (t2~t3)
011：T4 工作区间 (t3~t4)
1xx：T5 工作区间 (t4~t0)

Bit 3 **FJTON**：FJTIMER 计数器使能控制
0：除能
1：使能
当 PWMON=0 或 PWMON=1&FJWMD=0 时，FJTIMER 可作为一般的定时器使用，载入计数器的值来自 FJTMR1 预载寄存器。当 FJTON 由 0 变 1 时，FJTMR1 的值会重新载入到 FJTIMER；FJTON 位为 FJTIMER 的使能 / 除能控制位，关闭 FJTIMER 及 FJTIMER 分频器时钟，可减少额外功耗。

Bit 2 未定义，读为“0”

Bit 1~0 **FJTMPSC1~FJTMPSC0**：FJTIMER 计数时钟 $f_{FJTIMER}$ 选择
00：$f_H/128$
01：$f_H/256$
10：$f_H/512$
11：$f_H/1024$

- **FJTMR1 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**：FJTIMER T1 工作区间预载寄存器
T1 区间长度 = (256 - FJTMR1[7:0]) × $f_{FJTIMER}$

- **FJTMR2 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**：FJTIMER T2 工作区间预载寄存器
T2 区间长度 = (256 - FJTMR2[7:0]) × $f_{FJTIMER}$

- **FJTMR3 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**：FJTIMER T3 工作区间预载寄存器
T3 区间长度 = (256 - FJTMR3[7:0]) × $f_{FJTIMER}$

- **FJTMR4 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**：FJTIMER T4 工作区间预载寄存器
T4 区间长度 = (256 - FJTMR4[7:0]) × $f_{FJTIMER}$

● **FJTMD 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**：FJTIMER 计数器寄存器

● **FJPAL 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**：12-bit PWM 周期逼近目标 A 低字节寄存器
12-bit FJPA bit 7 ~ bit 0

● **FJPAH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”
Bit 3~0 **D11~D8**：12-bit PWM 周期逼近目标 A 高字节寄存器
12-bit FJPA bit 11 ~ bit 8

● **FJPBL 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**：12-bit PWM 周期逼近目标 B 低字节寄存器
12-bit FJPB bit 7 ~ bit 0

● **FJPBH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”
Bit 3~0 **D11~D8**：12-bit PWM 周期逼近目标 B 高字节寄存器
12-bit FJPB bit 11 ~ bit 8

● **FJPHL 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**：抖频过程中 PWM 周期修改值低字节寄存器
12-bit FJPH bit 7 ~ bit 0

● **FJPHH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”
Bit 3~0 **D11~D8**：抖频过程中 PWM 周期修改值高字节寄存器
12-bit FJPH bit 11 ~ bit 8
在抖频功能启动后，硬件可利用 FJPH[11:0] 值，实现对 PWMP 和 PWMD 的修改。

● **DTMINL 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**：12-bit PWM 死区时间最小值低字节寄存器
12-bit PWM DTMIN bit 7 ~ bit 0

● **DTMINH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”
Bit 3~0 **D11~D8**：12-bit PWM 死区时间最小值高字节寄存器
12-bit PWM DTMIN bit 11 ~ bit 8
12-bit DTMIN 死区时间长度 = (DTMIN[11:0]+1)/f_{DT}，其中 $f_{DT}=f_{PWM}$

● **FJDT0AL 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**：12-bit PWM 死区时间 0 逼近目标 A 的低字节寄存器
12-bit FJDT0A bit 7 ~ bit 0

● **FJDT0AH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”

Bit 3~0 **D11~D8**：12-bit PWM 死区时间 0 逼近目标 A 的高字节寄存器
12-bit FJDT0A bit 11 ~ bit 8
FJDT0A 死区时间长度 = (FJDT0A[11:0]+1)/f_{DT}，其中 $f_{DT}=f_{PWM}$

● **FJDT0BL 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**：12-bit PWM 死区时间 0 逼近目标 B 的低字节寄存器
12-bit FJDT0B bit 7 ~ bit 0

● **FJDT0BH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”

Bit 3~0 **D11~D8**：12-bit PWM 死区时间 0 逼近目标 B 的高字节寄存器
12-bit FJDT0B bit 11 ~ bit 8
FJDT0B 死区时间长度 = (FJDT0B[11:0]+1)/f_{DT}，其中 $f_{DT}=f_{PWM}$

● **FJDT1AL 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**：12-bit PWM 死区时间 1 逼近目标 A 的低字节寄存器
12-bit FJDT1A bit 7 ~ bit 0

● **FJDT1AH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”

Bit 3~0 **D11~D8**：12-bit PWM 死区时间 1 逼近目标 A 的高字节寄存器
12-bit FJDT1A bit 11 ~ bit 8
FJDT1A 死区时间长度 = (FJDT1A[11:0]+1)/f_{DT}，其中 $f_{DT}=f_{PWM}$

● **FJDT1BL 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**：12-bit PWM 死区时间 1 逼近目标 B 的低字节寄存器
12-bit FJDT1B bit 7 ~ bit 0

● **FJDT1BH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未定义，读为“0”

Bit 3~0 **D11~D8**：12-bit PWM 死区时间 1 逼近目标 B 的高字节寄存器
12-bit FJDT1B bit 11 ~ bit 8
FJDT1B 死区时间长度 = (FJDT1B[11:0]+1)/f_{DT}，其中 $f_{DT}=f_{PWM}$

● **FJC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	FJWMD	FJEN	FJMDS	FJSMDS	FJHMDS	FJTSS	FJOTSS	FJADJF
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	0

Bit 7 **FJWMD**：抖频工作模式选择位
0：软件抖频模式 (不搭配使用 FJTIMER)
1：硬件抖频模式 (搭配使用 FJTIMER)
注：当 ATMPF 由 0 变为 1 时，FJWMD 位会被硬件清零。

Bit 6 **FJEN**：抖频功能使能控制位
0：除能
1：使能
注：1. 此位只有在 FJWMD=0 时，才可以通过软件写入值。
2. 若有 ERSG 中断发生，此位会被硬件自动清零。
3. 若 PWMON=0 或 ATMPF=1，不会启动抖频功能 (FJEN=0)。

Bit 5 **FJMDS**：软件抖频及硬件抖频 T1 和 T4 区间的工作模式选择
0：计数模式 (固定计数次数)
1：逼近模式 (往目标值增 / 减)

Bit 4 **FJSMDS**：软件抖频模式逼近寄存器选择
0：FJPA/FJDT0A/FJDT1A
1：FJPB/FJDT0B/FJDT1B
此位只在 FJMDS 位为 1 时有效。

Bit 3 **FJHMDS**：硬件抖频模式工作区间选择
0：T1~T4
1：T2~T3

Bit 2 **FJTSS**：硬件抖频模式启动触发方式选择
0：由外部信号 (OVP7OS 或 FJTR1 信号) 触发启动
1：由 FJTON 位控制
若选择外部信号，具体使用哪个信号则由 FJOTSS 位选择。

Bit 1 **FJOTSS**：触发启动硬件抖频的外部信号选择
0：OVP7OS
1：FJTR1
此位只在 FJTSS 位为 0 时有效。

Bit 0 **FJADJF**：抖频功能启动标志位
0：未启动
1：启动
注：当此位为 1 且 FJHD 为 0 时，PWMP、PWMD、FJM1C0@FJWMD=1、DT0@FJDT0EN=1/CFJDT0EN=1、FJM1C2 寄存器中的 FJDT0SA[2:0] 位 @ FJDT0EN=1、DT1@FJDT1EN=1/CFJDT1EN=1、以及 FJM1C3 寄存器中的 FJDT1SA[2:0] 位 @ FJDT1EN=1，这些寄存器内容不可通过软件修改。

● **FJC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	FJEAF	FJEBF	FJAF	FJBF	—	—	FJHD	FJRT
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7 **FJEAF**：PWMP 逼近 FJPA 完成标志位 (被最大值限制住无法继续逼近时设立)
0：未完成
1：已完成
注：该位只能软件清零，不能写 1。若此位为 1，则在 FJEN 位由 0 变 1 (软件抖频模式) 或 FJTON 位由 0 变 1 (硬件抖频模式) 时，硬件会将此位自动清为 0。若 FJMDS=1 & FJWMD=1 & FJHMDS=0，则在 FJTMMD[2:0] 由 010 变为 011 时，硬件会将此位自动清为 0。

Bit 6 **FJEBF**：PWMP 逼近 FJPB 完成标志位 (被最大值限制住无法继续逼近时设立)
0：未完成
1：已完成
注：该位只能软件清零，不能写 1。若此位为 1，则在 FJEN 位由 0 变 1 (软件抖频模式) 或 FJTON 位由 0 变 1 (硬件抖频模式) 时，硬件会将此位自动清为 0。若 FJMDS=1 & FJWMD=1 & FJHMDS=0，则在 FJTMMD[2:0] 由 001 变为 010 时，硬件会将此位自动清为 0。

Bit 5 **FJAF**：PWMP 逼近 FJPA 完成标志位
0：未完成
1：已完成
注：该位只能软件清零，不能写 1。若此位为 1，则在 FJEN 位由 0 变 1 (软件抖频模式) 或 FJTON 位由 0 变 1 (硬件抖频模式) 时，硬件会将此位自动清为 0。若 FJMDS=1 & FJWMD=1 & FJHMDS=0，则在 FJTMMD[2:0] 由 010 变为 011 时，硬件会将此位自动清为 0。

Bit 4 **FJBF**：PWMP 逼近 FJPB 完成标志位
0：未完成
1：已完成
注：该位只能软件清零，不能写 1。若此位为 1，则在 FJEN 位由 0 变 1 (软件抖频模式) 或 FJTON 位由 0 变 1 (硬件抖频模式) 时，硬件会将此位自动清为 0。若 FJMDS=1 & FJWMD=1 & FJHMDS=0，则在 FJTMMD[2:0] 由 001 变为 010 时，硬件会将此位自动清为 0。

Bit 3~2 未定义，读为“0”

Bit 1 **FJHD**：抖频模式中软件通过 FJPH 对 PWMP/PWMD 修改控制位
1→0：修改 PWMP/PWMD 内容值
注：软件需先将要对 PWMP 修改的值写到 FJPH 寄存器中，然后当 FJHD 由 1 变 0 时，硬件依据 FJPH 寄存器的值对 PWMP 及 PWMD 做修改 (PWMD 的值为 FJPH 的值除以 2 后无条件舍掉小数)。

Bit 0 **FJRT**：抖频变量重新计数控制位
0：无操作
1：清除计数值
注：若此位为 1 且当 FJHD 由 1 变 0 并正确写入时，内部计数值会被清零并重新计数，同时硬件会将 FJRT 位清零。

● **FJC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	FJDT0AF	FJDT0BF	FJDT1AF	FJDT1BF	—	FJTR1EG	FJTR1DB1	FJTR1DB0
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 **FJDT0AF**：DT0 逼近 FJDT0A 完成标志位
0：未完成
1：已完成
注：该位只能软件清零，不能写 1。若此位为 1，则在 FJEN 位由 0 变 1 (软件抖频模式) 或 FJTON 位由 0 变 1 (硬件抖频模式) 时，硬件会将此位自动清为 0。

Bit 6 **FJDT0BF**：DT0 逼近 FJDT0B 完成标志位
0：未完成
1：已完成
注：该位只能软件清零，不能写 1。若此位为 1，则在 FJEN 位由 0 变 1 (软件抖频模式) 或 FJTON 位由 0 变 1 (硬件抖频模式) 时，硬件会将此位自动清为 0。

Bit 5 **FJDT1AF**：DT1 逼近 FJDT1A 完成标志位
0：未完成
1：已完成
注：该位只能软件清零，不能写 1。若此位为 1，则在 FJEN 位由 0 变 1 (软件抖频模式) 或 FJTON 位由 0 变 1 (硬件抖频模式) 时，硬件会将此位自动清为 0。

Bit 4 **FJDT1BF**：DT1 逼近 FJDT1B 完成标志位
0：未完成
1：已完成
注：该位只能软件清零，不能写 1。若此位为 1，则在 FJEN 位由 0 变 1 (软件抖频模式) 或 FJTON 位由 0 变 1 (硬件抖频模式) 时，硬件会将此位自动清为 0。

Bit 3 未定义，读为“0”

Bit 2 **FJTR1EG**：FJTR1 触发 FJTIMER 信号源选择位
0：上升沿
1：下降沿

Bit 1~0 **FJTR1DB1~FJTR1DB0**：FJTR1 引脚去抖时间选择
00：旁路，无去抖
01：(7~8) × t_H
10：(15~16) × t_H
11：(23~24) × t_H

● **FJF0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	AMAXF	BMAXF	HMAXF	PMAXF	—	—	DT1MINF	DT0MINF
R/W	R	R	R	R	—	—	R	R
POR	0	0	0	0	—	—	0	0

Bit 7 **AMAXF**：FJPA 写入无效标志位
0：正确写入
1：写入无效 (FJPA > PWMMAXP)
注：当 PMAXPTE=0 时，无论 FJPA > 或 ≤ PWMMAXP，FJPA 都可以写入且 AMAXF=0；当 PMAXPTE=1 时，若 FJPA>PWMMAXP，AMAXF 会被设为 1 且 FJPA 写入无效，若 FJPA≤PWMMAXP，AMAXF=0 且 FJPA 可以正确写入。

Bit 6 **BMAXF**：FJPB 写入无效标志位
0：正确写入
1：写入无效 (FJPB > PWMMAXP)
注：当 PMAXPTE=0 时，无论 FJPB > 或 ≤ PWMMAXP，FJPB 都可以写入且 BMAXF=0；当 PMAXPTE=1 时，若 FJPB >PWMMAXP，BMAXF 会被设为 1 且 FJPB 写入无效，若 FJPB≤PWMMAXP，BMAXF=0 且 FJPB 可以正确写入。

Bit 5 **HMAXF**：FJPH 写入无效标志位
0：正确写入
1：写入无效 (FJPH > PWMMAXP)
注：当 PMAXPTE=0 时，无论 FJPH > 或 ≤ PWMMAXP，FJPH 都可以写入且 HMAXF=0；当 PMAXPTE=1 时，若 FJPH >PWMMAXP，HMAXF 会被设为 1 且 FJPH 写入无效，若 FJPH≤PWMMAXP，HMAXF=0 且 FJPH 可以正确写入。

Bit 4 **PMAXF**：PWMP 写入无效标志位
0：正确写入
1：写入无效 (PWMP > PWMMAXP)
注：当 PMAXPTE=0 时，无论 PWMP > 或 ≤ PWMMAXP，PWMP 都可以写入且 PMAXF=0；当 PMAXPTE=1 时，若 PWMP >PWMMAXP，PMAXF 会被设为 1 且 PWMP 写入无效，若 PWMP≤PWMMAXP，PMAXF=0 且 PWMP 可以正确写入。

Bit 3~2 未定义，读为“0”

Bit 1 **DT1MINF**：DT1 写入无效标志位
0：正确写入
1：写入无效 (DT1 < DTMIN)
注：1. 软件向 DT1 或 DTMIN 寄存器写值时，都会触发判断。当写入到 DT1 的数值小于 DTMIN 时，DT1MINF 会被硬件置位，此时 DT1 是不允许写入的。若写入 DTMIN 的值大于 DT1，是允许写入的。
2. 硬件向 DT1 寄存器写值时，若写入到 DT1 的数值小于 DTMIN 时，DT1MINF 会被硬件置位，此时 DT1 是不允许硬件写入的。

Bit 0 **DT0MINF**：DT0 写入无效标志位
0：正确写入
1：写入无效 (DT0 < DTMIN)
注：1. 软件向 DT0 或 DTMIN 寄存器写值时，都会触发判断。当写入到 DT0 的数值小于 DTMIN 时，DT0MINF 会被硬件置位，此时 DT0 是不允许写入的。若写入 DTMIN 的值大于 DT0，是允许写入的。
2. 硬件向 DT0 寄存器写值时，若写入到 DT0 的数值小于 DTMIN 时，DT0MINF 会被硬件置位，此时 DT0 是不允许硬件写入的。

● **FJM1C0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	FJCNT2	FJCNT1	FJCNT0	—	FJSA2	FJSA1	FJSA0
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 6~4　**FJCNT2~FJCNT0**：抖频逼近模式中 PWM 触发次数选择
000：1
001：2
010：3
011：4
100：5
101：6
110：7
111：8

Bit 3　未定义，读为“0”

Bit 2~0　**FJSA2~FJSA0**：抖频逼近模式中周期和占空比逼近数值选择
000：Period ±2，Duty ±1
001：Period ±4，Duty ±2
010：Period ±6，Duty ±3
011：Period ±8，Duty ±4
100：Period ±10，Duty ±5
101：Period ±12，Duty ±6
110：Period ±14，Duty ±7
111：Period ±16，Duty ±8

● **FJM1C1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	FJTIMES2	FJTIMES1	FJTIMES0	FJACNT1	FJACNT0	FJASA1	FJASA0
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 6~4　**FJTIMES2~FJTIMES0**：抖频逼近模式中调变次数选择
000：1
001：2
010：3
011：4
100：5
101：6
110：7
111：8
注：此处的一次调变包括改变 FJCNT[2:0]、FJSA[2:0]、FJDT1SA[2:0] 以及 FJDT0SA[2:0] 的值。

Bit 3~2　**FJACNT1~FJACNT0**：抖频逼近模式中触发次数改变选择 (只在 T2&T3 工作区间有效)
00：不变
01：不变
10：T2 区间为 +1；T3 区间为 -1
11：T2 区间为 -1；T3 区间为 +1
注：1. 当 FJCNT[2:0] 值递增到最大值 111 或递减到最小值 000 时，FJCNT[2:0] 值不会再依 FJACNT[1:0] 设定增加或减少，会固定于最大值 111 或最小值 000。
2. 当 PWM 已达到逼近值时，FJCNT[2:0] 保持不变。

Bit 1~0 **FJASA1~FJASA0**：抖频逼近模式中逼近数值改变选择 (只在 T2&T3 工作区间有效)
00：不变
01：不变
10：T2 区间为 +1；T3 区间为 -1
11：T2 区间为 -1；T3 区间为 +1
注：1. 当 FJSA[2:0] 值递增到最大值 111 或递减到最小值 000 时，FJSA[2:0] 值不会再依 FJASA[1:0] 设定增加或减少，会固定于最大值 111 或最小值 000。
2. 当 PWM 已达到逼近值时，FJSA[2:0] 保持不变。

● **FJM1C2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	FJDT0EN	FJDT0ASA1	FJDT0ASA0	FJDT0CNT1	FJDT0CNT0	FJDT0SA2	FJDT0SA1	FJDT0SA0
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 **FJDT0EN**：抖频逼近模式中 DT0 修改使能控制位
0：除能
1：使能

Bit 6~5 **FJDT0ASA1~FJDT0ASA0**：抖频逼近模式中 DT0 逼近数值改变选择 (只在 T2&T3 工作区间有效)
00：不变
01：不变
10：T2 区间为 +1；T3 区间为 -1
11：T2 区间为 -1；T3 区间为 +1
注：当 FJDT0SA[2:0] 值递增到最大值 111 或递减到最小值 000 时，FJDT0SA[2:0] 值不会再依 FJDT0ASA[1:0] 设定增加或减少，会固定于最大值 111 或最小值 000。

Bit 4~3 **FJDT0CNT1~FJDT0CNT0**：抖频逼近模式中修改 DT0 的 PWM 触发次数选择
00：4
01：8
10：12
11：16

Bit 2~0 **FJDT0SA2~FJDT0SA0**：抖频逼近模式中 DT0 的修改数值选择
000：1
001：2
010：3
011：4
100：5
101：6
110：7
111：8

● **FJM1C3 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	FJDT1EN	FJDT1ASA1	FJDT1ASA0	FJDT1CNT1	FJDT1CNT0	FJDT1SA2	FJDT1SA1	FJDT1SA0
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 **FJDT1EN**：抖频逼近模式中 DT1 修改使能控制位
0：除能
1：使能

Bit 6~5 **FJDT1ASA1~FJDT1ASA0**：抖频逼近模式中 DT1 逼近数值改变选择 (只在 T2&T3 工作区间有效)
00：不变
01：不变
10：T2 区间为 +1；T3 区间为 -1
11：T2 区间为 -1；T3 区间为 +1
注：当 FJDT1SA[2:0] 值递增到最大值 111 或递减到最小值 000 时，FJDT1SA[2:0] 值不会再依 FJDT1ASA[1:0] 设定增加或减少，会固定于最大值 111 或最小值 000。

Bit 4~3 **FJDT1CNT1~FJDT1CNT0**：抖频逼近模式中修改 DT1 的 PWM 触发次数选择
00：4
01：8
10：12
11：16

Bit 2~0 **FJDT1SA2~FJDT1SA0**：抖频逼近模式中 DT1 的修改数值选择
000：1
001：2
010：3
011：4
100：5
101：6
110：7
111：8

- **FJM0C0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	CFJCNT2	CFJCNT1	CFJCNT0	CFJS	CFJSA2	CFJSA1	CFJSA0
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 6~4 **CFJCNT2~CFJCNT0**：抖频计数模式中递增或递减次数选择
000：1
001：2
010：3
011：4
100：5
101：6
110：7
111：8

Bit 3 **CFJS**：抖频计数模式中周期和占空比数值修改方向选择
0：增加
1：减少

Bit 2~0 **CFJSA2~CFJSA0**：抖频计数模式中周期和占空比修改数值选择
000：Period ±2，Duty ±1
001：Period ±4，Duty ±2
010：Period ±6，Duty ±3
011：Period ±8，Duty ±4
100：Period ±10，Duty ±5
101：Period ±12，Duty ±6
110：Period ±14，Duty ±7
111：Period ±16，Duty ±8

● **FJM0C1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CFJDT0EN	—	CFJDT0CNT1	CFJDT0CNT0	CFJDT0S	—	CFJDT0SA1	CFJDT0SA0
R/W	R/W	—	R/W	R/W	R/W	—	R/W	R/W
POR	0	—	0	0	0	—	0	0

Bit 7 **CFJDT0EN**：抖频计数模式中 DT0 修改使能控制位
0：除能
1：使能

Bit 6 未定义，读为“0”

Bit 5~4 **CFJDT0CNT1~CFJDT0CNT0**：抖频计数模式中修改 DT0 的 PWM 触发次数选择
00：1
01：2
10：3
11：4

Bit 3 **CFJDT0S**：抖频计数模式中 DT0 数值修改方向选择
0：增加
1：减少

Bit 2 未定义，读为“0”

Bit 1~0 **CFJDT0SA1~CFJDT0SA0**：抖频计数模式中 DT0 修改数值选择
00：±1
01：±2
10：±3
11：±4

● **FJM0C2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CFJDT1EN	—	CFJDT1CNT1	CFJDT1CNT0	CFJDT1S	—	CFJDT1SA1	CFJDT1SA0
R/W	R/W	—	R/W	R/W	R/W	—	R/W	R/W
POR	0	—	0	0	0	—	0	0

Bit 7 **CFJDT1EN**：抖频计数模式中 DT1 修改使能控制位
0：除能
1：使能

Bit 6 未定义，读为“0”

Bit 5~4 **CFJDT1CNT1~CFJDT1CNT0**：抖频计数模式中修改 DT1 的 PWM 触发次数选择
00：1
01：2
10：3
11：4

Bit 3 **CFJDT1S**：抖频计数模式中 DT1 数值修改方向选择
0：增加
1：减少

Bit 2 未定义，读为“0”

Bit 1~0 **CFJDT1SA1~CFJDT1SA0**：抖频计数模式中 DT1 修改数值选择
00：±1
01：±2
10：±3
11：±4

功能描述

此高精度 PWM 产生器包括一个 12-bit 的计数器电路，用于产生互补式 PWM 输出。针对 PWM0/PWM2 和 PWM1/PWM3 这几路 PWM 输出，分别提供了 12-bit 可编程的死区插入电路及极性控制电路。同时也提供了完善的保护机制，当有错误信号发生时，可将 PWM 电路立即关闭或对 PWM 周期进行调变。

针对电磁炉的功能需求，该单片机同时提供了一系列电路用于实现零电流开关信号输出，检锅，相位侦测及脉宽测量，相位保护等功能，另外内置抖频功能，通过软件或者硬件即可实现频率抖动。以下将对各个电路功能进行具体描述。

PWM 信号产生电路

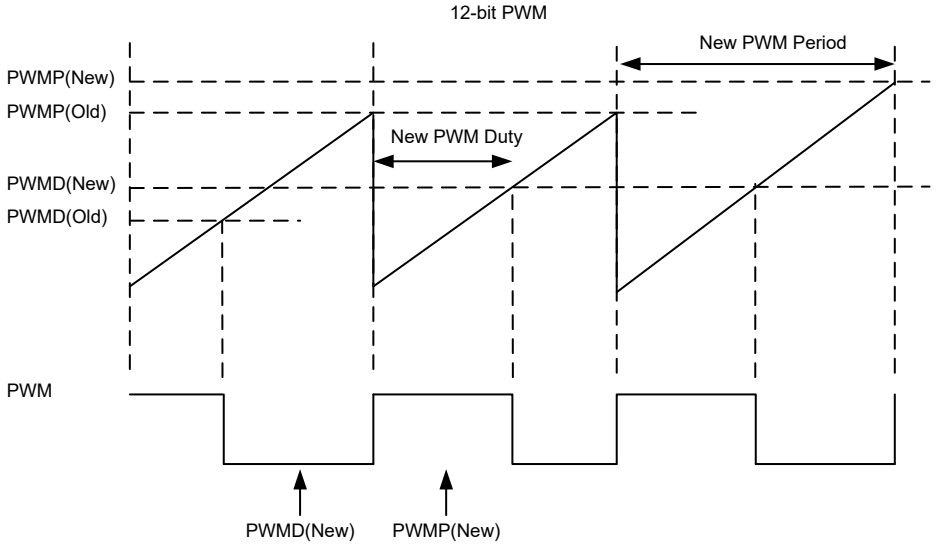
此单片机提供了一个 12-bit PWM 产生器，通过互补电路，可输出多路互补 PWM 信号。通过调整周期和占空比值，可提供灵活的 PWM 信号用于不同应用。12-bit PWMP 和 PWMD 寄存器用于对 PWM 波形的频率和占空比进行设置。以下为一个范例。

若选择的 PWM 计数器时钟 $f_{PWMPSC}=f_{PWM}=32MHz$，设置 PWMP=511，PWMD=255，

PWM 周期 = $f_{PWMPSC}/(PWMP[11:0]+1)=32MHz/512=62.5kHz$，
占空比 = (255+1)/512=50%

需注意的是，应用中，PWMD 值需要满足：$1 \le PWMD \le (PWMP - 1)$

当 PWMP 和 PWMD 比较匹配发生时，会产生相应的 PWM 周期和占空比匹配中断。



当通过软件更新 PWMP、PWMD、DET、DT0 和 DT1 缓冲寄存器后，硬件会在 PWMR 计数器值为 0 时，更新 PWMP、PWMD、DET、DT0 和 DT1。

死区时间插入

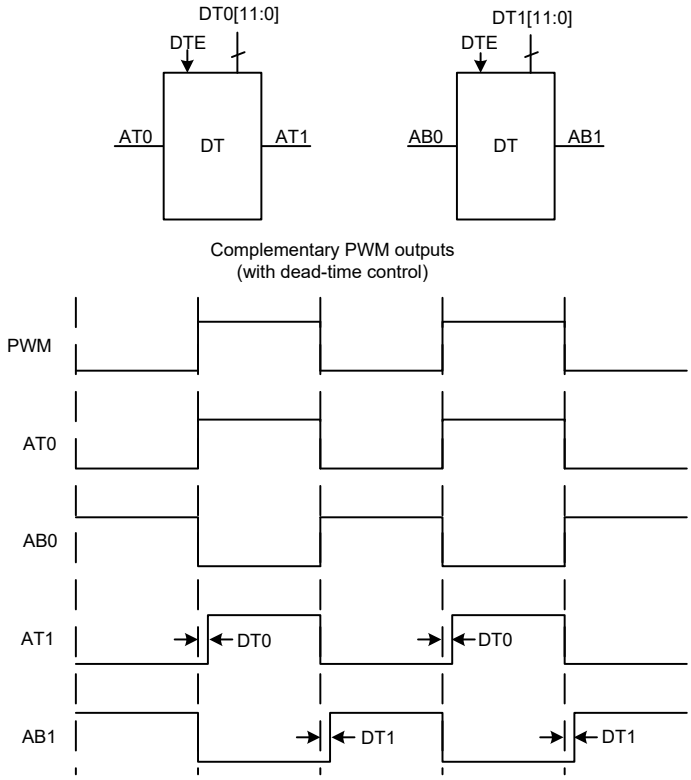
死区插入的目的：死区插入电路确保外部驱动电路的电晶体对在转态时不会瞬间导通 (上下臂 MOS 同时开启)，产生短路电流。为了消除这种危险，设计了二组独立 12-bit 死区时间，确保输出转态的过程中，两个晶体管都处于不导通的状态。死区时间可控制在 0μs~128μs 左右，使用软件通过 DT0 和 DT1 寄存器进行调整，二组独立调整死区时间为 $(DT0[11:0]+1)/f_{DT}$ 与 $(DT1[11:0]+1)/f_{DT}$，其

中 $f_{DT}=f_{PWM}$。

死区插入功能使能，PWM0~PWM3 的输出如下所示：

- PWM0/PWM2 输出上升沿时，延迟死区时间后再输出，调整寄存器为 DT0H&DT0L。
- PWM1/PWM3 输出上升沿时，延迟死区时间后再输出，调整寄存器为 DT1H&DT1L。

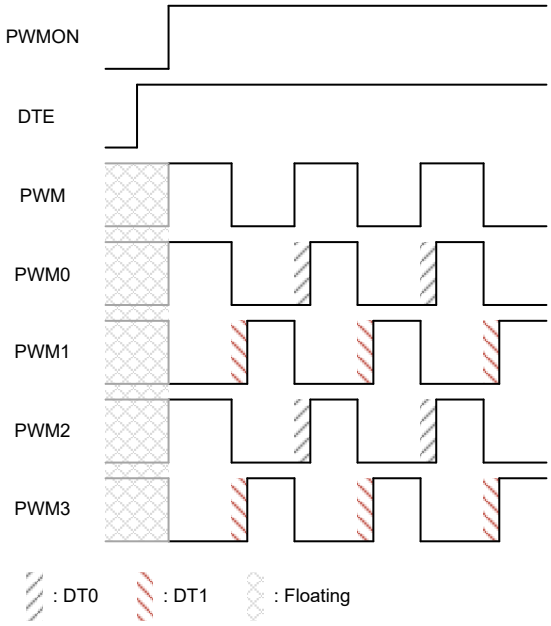
死区时间插入电路只在半桥 / 全桥电磁炉控制时使用，可利用 PWMC0 寄存器中的 DTE 位来控制。



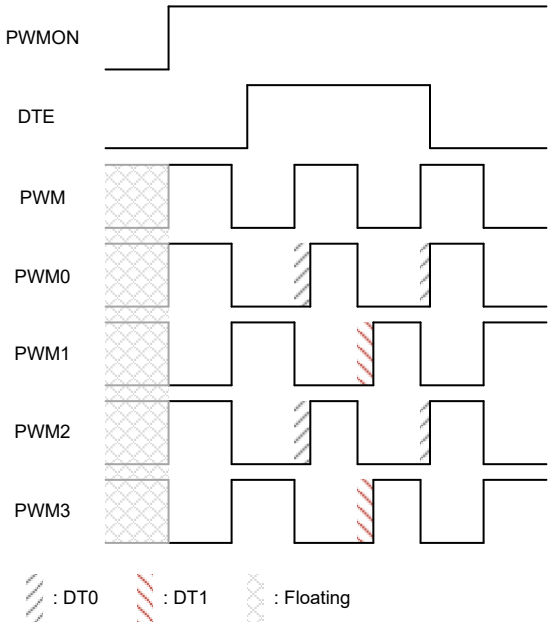
互补式输出插入死区时间

DTE 设定说明：

- 在 DTE=1 时，设定 PWMON 为 1，此时在 PWM0/PWM2 的第一个上升沿不会启动死区时间，但 PWM1/PWM3 第一个上升沿会立即启动死区时间。



- 在 PWMON=1 时，设定 DTE 由 0 变 1，PWM 会立即启动死区时间，并于下一个 PWM0/PWM2 或 PWM1/PWM3 的上升沿添加；当 DTE 由 1 变 0 时，PWM 会立即关闭死区时间，不会等到 DT0 或 DT1 计数完毕才关闭。



移相角度延迟时间

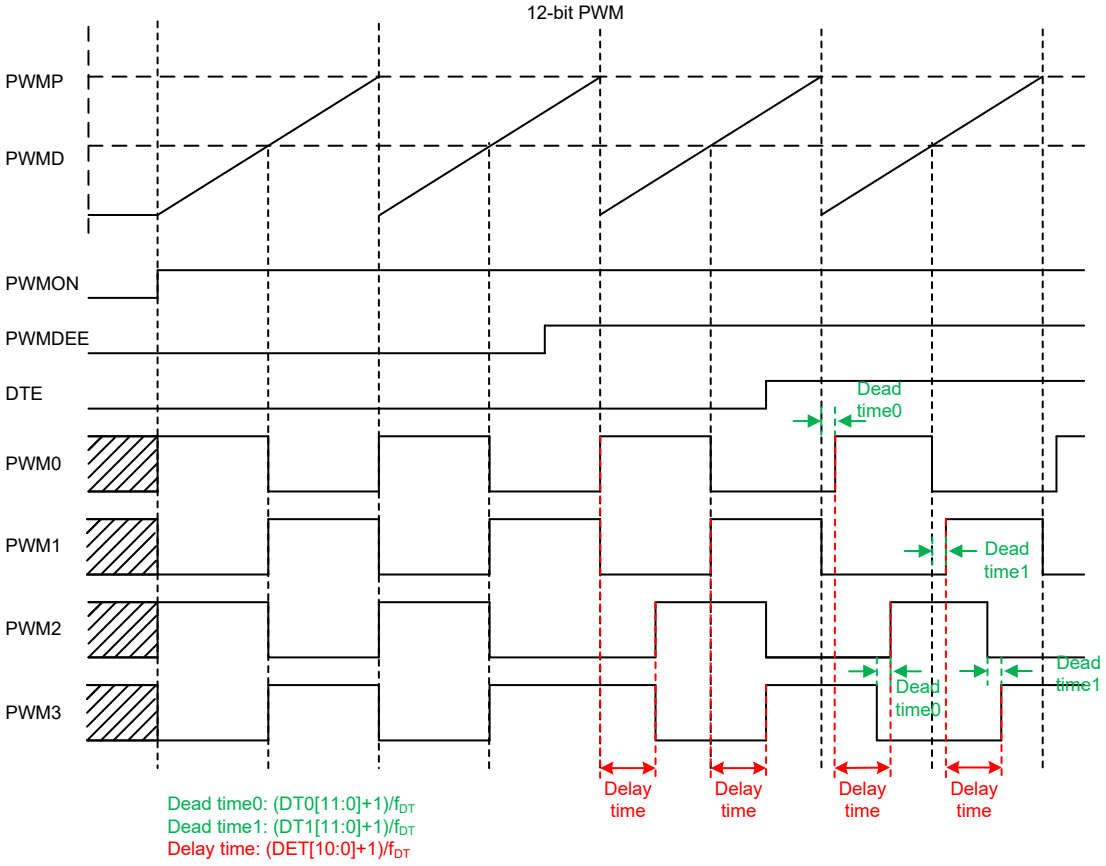
移相角度延迟时间设计的目的：移相角度延迟时间主要用于控制电磁炉输出功率，在两组互补式 PWM 输出之间添加一个延迟时间，设计了一组独立 11-bit 移相角度延迟时间。移相角度延迟时间可控制在 0μs~64μs 左右，使用软件设置

DETH&DETL 寄存器进行调整。移相角度延迟时间为 (DET[10:0]+1)/f_{DT}，其中 $f_{DT}=f_{PWM}$。

移相角度延迟时间插入使能后，PWM0~PWM3 输出关系如下所示：

- PWM0 输出上升沿时，添加移相角度延迟时间后再输出 PWM2，调整寄存器为 DETH&DETL。
- PWM1 输出上升沿时，添加移相角度延迟时间后再输出 PWM3，调整寄存器为 DETH&DETL。

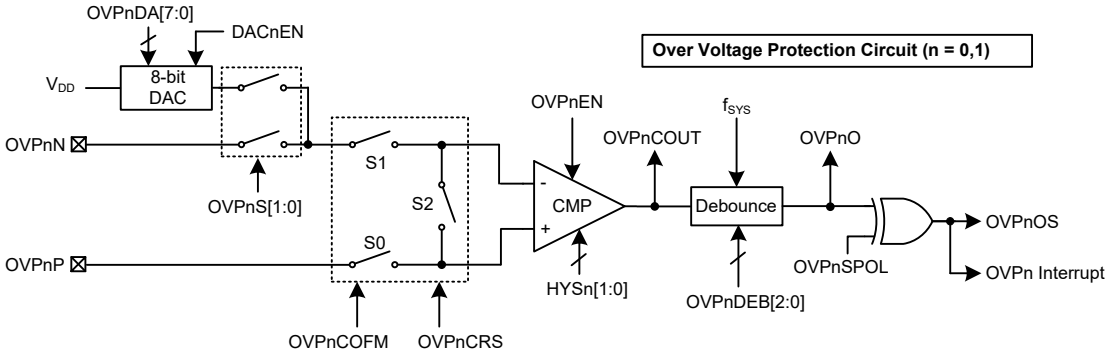
移相角度延迟时间插入电路只在全桥电磁炉控制时使用，可利用 PWMC0 寄存器中的 PWMDEE 位来控制。



注：当 PWMDEE 由 0 变 1 时，PWM2 与 PWM3 不会立即启动移相角度延迟，需当前整个周期执行完后才启动移相角度延迟，也就是在 PWM2 的上升沿与 PWM3 的下降沿时，启动移相角度延迟。

谐振电流转数字信号

谐振电流的两端信号经过电流传感器输入到 OVP0 与 OVP1 的正端与负端，OVP0 与 OVP1 的负端也可以选择由 8-bit D/A 转换器输出的 0V 电压，OVP0 与 OVP1 输出端就会产生电流数字信号 OVP0OS 与 OVP1OS。此信号会经过去抖处理 (去抖时间由使用者设定)，利用 OVP0OS 和 OVP1OS 信号来检测加热线圈上有无锅具。

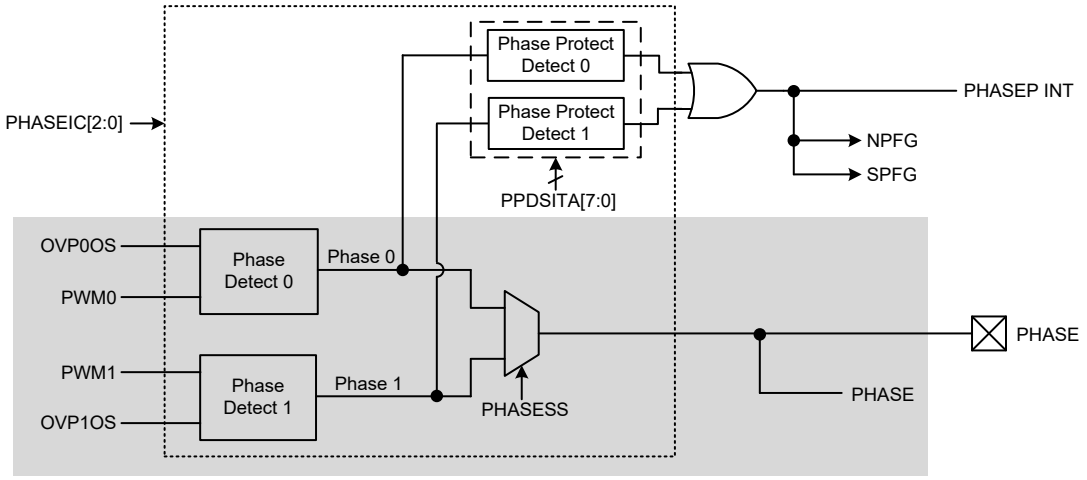


OVP0/OVP1 方框图

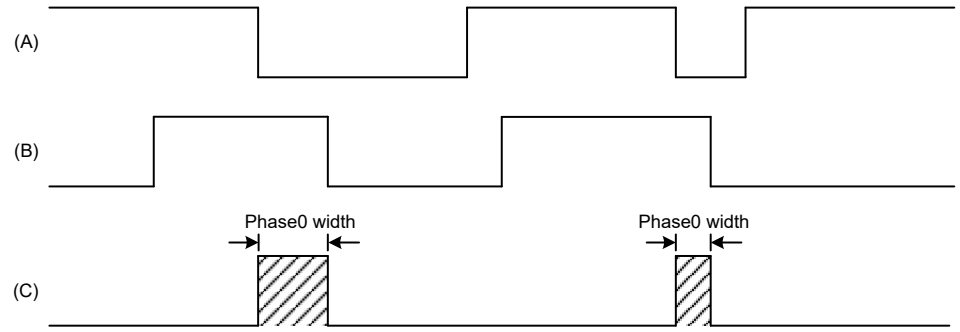
注：关于 OVP 细节方面请参考过电压保护章节。

相位侦测电路和保护机制

相位侦测电路有 2 组分为 Phase Detect 0 和 Phase Detect 1，Phase Detect 0 侦测 OVP0OS 和 PWM0 信号，Phase0 为其输出信号，Phase Detect 1 侦测 OVP1OS 和 PWM1 信号，Phase1 为其输出信号。相位侦测电路的结构示意图见下图中灰色阴影部分。



下图以 Phase Detect 0 电路来说明，当 OVP0OS 信号 (电流信号) 落后 PWM 信号 (电压信号)，此时在 Phase Detect 0 电路会产生 Phase 0 信号，利用 Phase 0 宽度可计算出电流落后电压相位宽度，来调整电磁炉输出功率。当电流信号和电压信号没有相位差，此时 Phase 0 宽度为 0，电磁炉输出功率最大。如果电流信号领先电压信号，此时在相位侦测电路会无法检测到 Phase 0 宽度，保护机制电路就会启动，Phase Detect 1 电路同理。

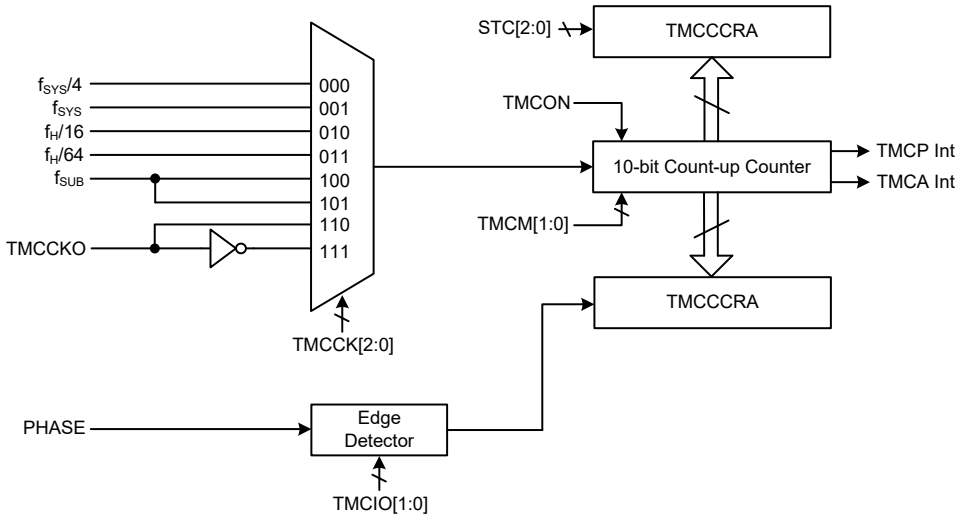


(A): PWM0 signal
(B): OVP0OS signal
(C): Phase0 signal

Phase Detect 0 侦测时序图

10-bit TMC 计数器

TMC 是一个 10-bit 向上计数器。时钟源可选择系统时钟 f_{SYS} 和内部高速时钟 f_H 的分频，内部低速时钟 f_{SUB} 以及内部 TMCCKO 信号驱动。TMC 有两种工作模式，分别是捕捉输入模式和定时 / 计数器模式。通过应用程序改变 10-bit 计数器值以及 TMCCCRA 值的唯一办法是使 TMCON 位发生上升沿跳变从而清除计数器。



10-bit TMC 方框图

定时 / 计数器模式

为使 TMC 工作在此模式，TMCC1 寄存器的 TMCM1 和 TMCM0 位需要设置为“10”或“11”。通过设置 TMCC0 寄存器的 TMCCK2~TMCCK0 位选择所需的时钟源。选择了定时 / 计数器模式，内部计数器会从 0 开始一直向上计数到 1023，当从 1023 溢出清零时，TMCP 中断标志位将被置为高。在定时 / 计数器模式下，通过 STC2~STC0 位设置的时间到达时，10-bit 计数器当前值将被储存到 TMCCCRA，同时 TMCA 中断标志位被置高，TMCON 位被清为零。

TMCCKO 允许内部信号作为 TMC 时钟源或用于事件计数。当选择内部 TMCCKO 信号作为 TMC 驱动时钟时，通过 STC2~STC0 位设置所需等待的时

间，然后设置 TMCON 位为高，当 STC2~STC0 位设置的时间到达后，TMCON 位被清零，计数器当前值被锁存到 TMCCCRA 寄存器，并设置 TMCA 中断标志位为高。

捕捉输入模式

为使 TMC 工作在此模式，TMCC1 寄存器的 TMCM1 和 TMCM0 位需要设置为“00”或“01”。此模式使能内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 TMCC1 寄存器的 TMCIO1 和 TMCIO0 位选择输入信号 (PHASE 信号) 的有效边沿类型，即上升沿，下降沿或双沿有效。通过应用程序将 TMCON 位由低变为高时，计数器启动。当选择输入的 PHASE 信号出现有效边沿转换时，计数器当前值被锁存到 TMCCCRA 寄存器，并设置 TMCA 中断标志位为高。无论 PHASE 信号发生哪种边沿转换，计数器将继续工作直到 TMCON 位发生下降沿跳变。

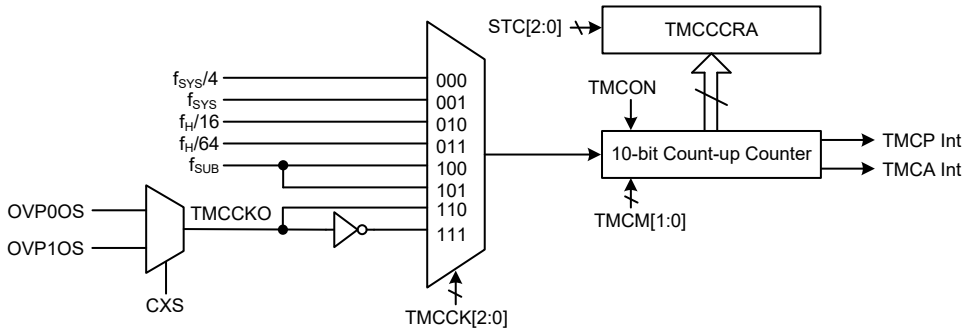
检锅和相位宽度测量电路

利用 OVP0OS 和 OVP1OS 检测加热线圈上有无锅具，把此信号接到 10-bit TMC 的 TMCCKO 输入，由计数数量来检测加热线圈上有无锅具。当 OVP0OS/OVP1OS (电流信号) 信号落后相应的 PWM (电压信号)，根据 Phase0/Phase1 信号宽度可用来检锅和测量相位宽度，此信号输入到 10-bit TMC，作为捕捉输入 PHASE 信号。利用 PHASE 宽度可用来计算电流落后电压相位宽度来调整电磁炉输出功率。

检锅电路

内部检测信号 OVP0OS 或 OVP1OS 信号可以接到 10-bit TMC 的 TMCCKO 输入，通过计算脉冲数量来检测加热线圈上有无锅具。操作步骤如下所示。

1. 选择 TMCCKO 信号作为 TMC 事件计数器来源信号，其中 TMCCKO 信号可以通过 PWMC1 寄存器的 CXS 位选择来自 OVP0OS 或是 OVP1OS 信号。
2. 通过 TMCC0 寄存器的 STC2~STC0 位设置所需要的等待时间。
3. 设置 TMCON 为高使能计数器工作。
4. 当等待时间到达后，TMCON 会被自动清零，计数器当前值被锁存到 TMCCCRA 寄存器，同时 TMCA 中断标志位被置高
5. 通过计数器当前值大小达到检锅。

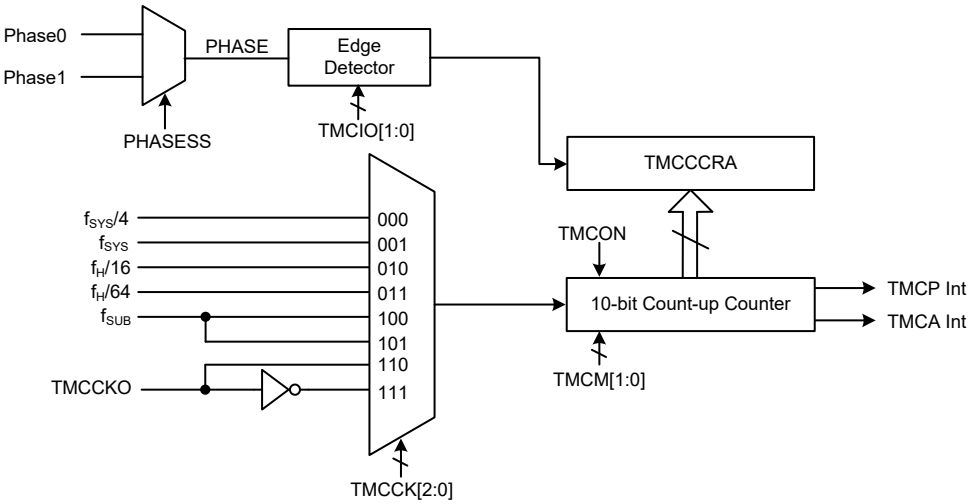


相位宽度测量电路

如果当 OVP0OS/OVP1OS 信号 (电流信号) 落后于对应的 PWM (电压信号)，此时在相位侦测电路会侦测到相位角宽度信号，Phase0/Phase1，此相位角宽度信号可被用来计算电流落后电压相位宽度。此单片机提供的 10-bit TMC 计数器可设置工作在捕捉输入模式，Phase 信号作为其捕捉输入，从而测量其相位角

宽度。宽度越大，全桥 / 半桥电磁炉功率输出越小，宽度越小，全桥 / 半桥电磁炉功率输出越大。当宽度为零时，电磁炉功率输出最大，此时的频率称为谐振频率。

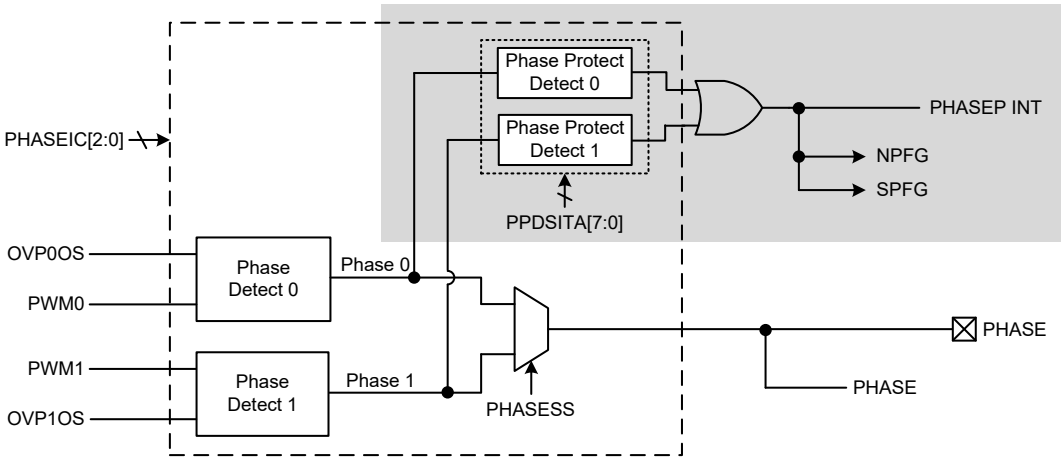
通常电磁炉操作频率会操作在比谐振频率稍大的安全值，可通过固件设定最大功率值并存入 PWMMAXP 寄存器，当作是全桥 / 半桥电磁炉输出的最大功率。设置 PMAXPTE 位为高使能 PWMP 最大值限制功能，则写入 PWMP 寄存器的值只允许小于或等于 PWMMAXP 寄存器的值。如果要写入的值比 PWMMAXP 中预设的值要大，则无法写入，同时 PMAXF 标志位会被硬件置位。通过这种方式，来控制全桥 / 半桥电磁炉输出功率无法超过设定的最大功率。



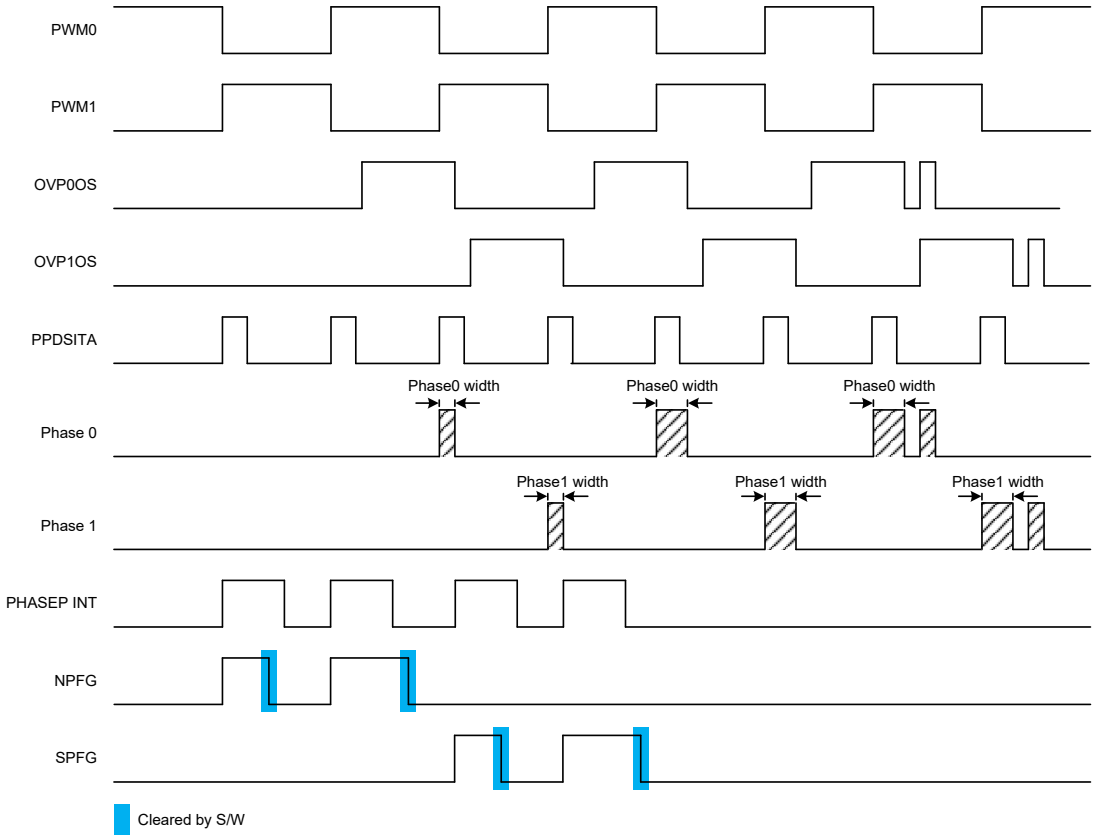
注：图中的 Phase 0 和 Phase 1 信号分别为 Phase Detect 0 和 Phase Detect 1 电路的输出。

相位保护侦测电路

相位保护侦测由相位保护侦测 0 和相位保护侦测 1 组成，分别用于对 Phase 0 和 Phase 1 信号宽度进行判断。若 Phase 0 或 Phase 1 的宽度小于 PPDSITA 设定的值，则 SPFG 标志位会被硬件设为 1。若 Phase 0 或 Phase 1 的宽度等于零，则 NPFG 标志位会被硬件置位。当上述两种情况有任一种发生时，都会产生 PHASEP 中断，会触发相位保护机制。如果 ERSGC 寄存器中的 ERSGC0 位为 1，就会启动相应的 ERSG 保护电路。其中 SPFG 与 NPFG 标志位只能由硬件置位，软件无法将其设 1，且只能通过软件清零，硬件不会将其清零。相位保护侦测电路的结构示意图见下图中灰色阴影部分。



相位保护侦测电路示意图



相位保护侦测电路时序图

相位保护侦测 0 动作原理

于侦测区间内没有侦测到相位宽度或相位宽度小于最小宽度时，会提供保护动作。

- 于侦测区间内，没有侦测到相位：

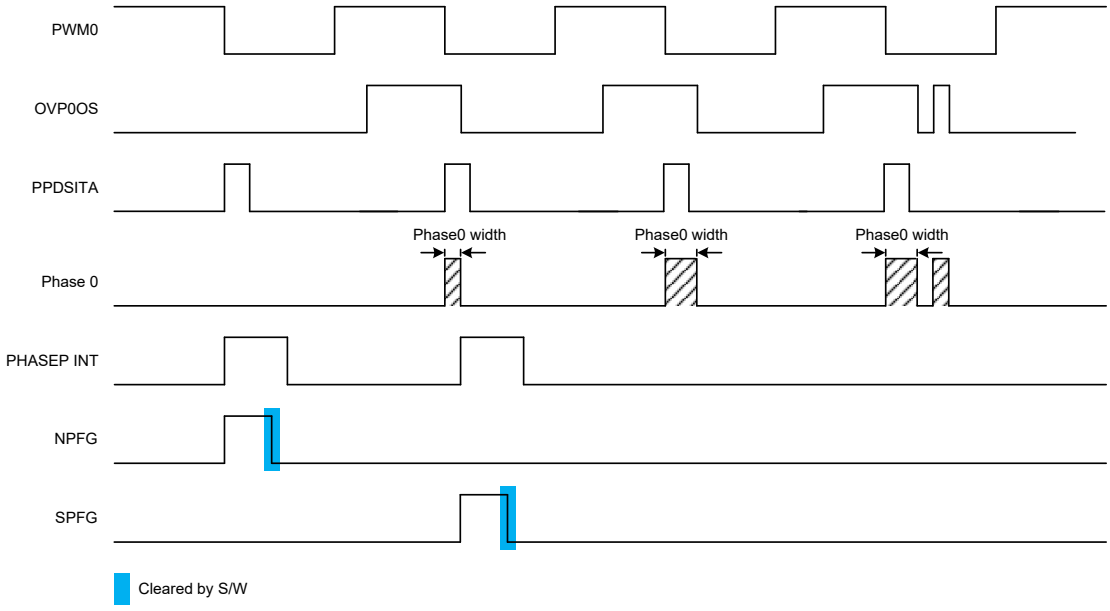
  当使用 PWM0 当触发信号时，OVP0OS 信号 = 0 则 PHASEP INT 信号与 NPFG 标志位会被置位为 1，反之，OVP0OS 信号 = 1 则 PHASEP INT 信号与 NPFG 不会被置位为 0。

- 于侦测区间内，侦测到最小相位：

  只计数第一个 Phase 宽度，若 Phase 宽度 < PPDSITA 设定值，则 PHASEP INT 信号与 SPFG 标志位会被置位为 1，反之，PHASEP INT 信号与 SPFG 为 0。

注：1. 相位保护侦测只侦测于侦测区间内的第一个 Phase 信号，其余 Phase 忽略不去计数保护。

2. SPFG 与 NPFG 需使用软件清零，否则一旦硬件将其设为 1 后，SPFG 与 NPFG 将永远为“1”。



相位保护侦测 0 时序图

相位保护侦测 1 动作原理

于侦测区间内没有侦测到相位宽度或相位宽度小于最小宽度时，会提供保护动作。

- 于侦测区间内，没有侦测到相位：

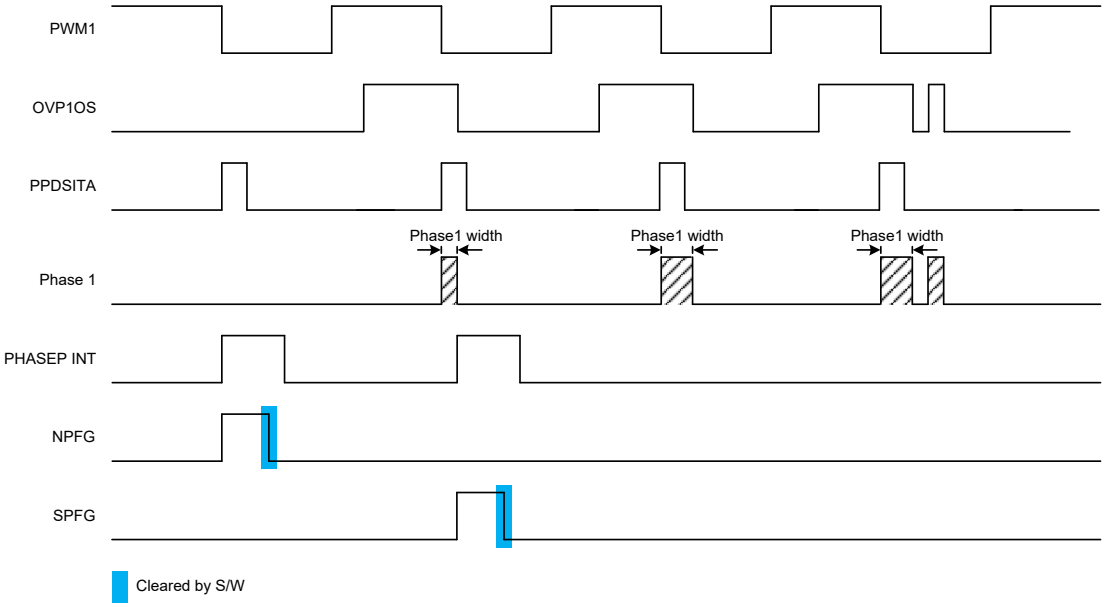
  当使用 PWM1 当触发信号时，OVP1OS 信号 = 0 则 PHASEP INT 信号与 NPFG 标志位会被置位为 1，反之，OVP1OS 信号 = 1 则 PHASEP INT 信号与 NPFG 不会被置位为 0。

- 于侦测区间内，侦测到最小相位：

  只计数第一个 Phase 宽度，若 Phase 宽度 < PPDSITA 设定值，则 PHASEP INT 信号与 SPFG 标志位会被置位为 1，反之，PHASEP INT 信号与 SPFG 为 0。

注：1. 相位保护侦测只侦测于侦测区间内的第一个 Phase 信号，其余 Phase 忽略不去计数保护。

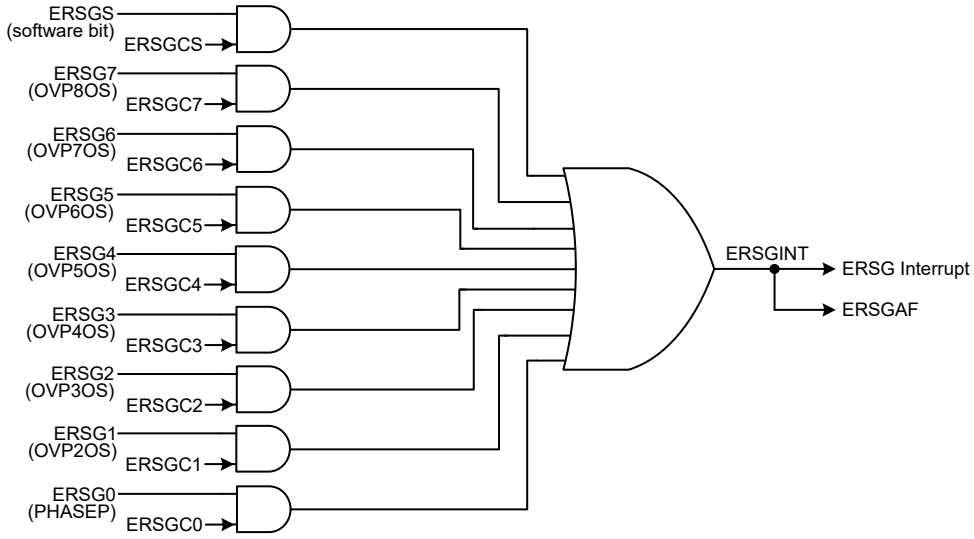
2. SPFG 与 NPFG 需使用软件清零，否则一旦硬件将其设为 1 后，SPFG 与 NPFG 将永远为“1”。



相位保护侦测 1 时序图

ERSG 保护电路

ERSG 保护电路共有 9 个来源输入可以触发其启动保护，信号分别为 ERSG0 (相位保护侦测电路输出 – PHASEP 信号)、ERSG1~ERSG7 (OVP2OS~OVP8OS 信号) 和 ERSGS (软件控制信号)。当 ERSGC 寄存器中的 ERSGC[7:0] 位或 PWMPC2 寄存器中的 ERSGCS 位为 1 且相应的信号为“高”，就会产生 ERSG 中断。



ERSG 中断结构

此保护电路针对每一个信号都有四种保护机制可设定。这四种保护机制主要是由两个电路组合而成，一为 PWM 关闭电路，另一为 PWM 自动调变电路，其依据 PWMPC0~PWMPC2 寄存器中的 PWMPCn[m+1:m] 设定产生相对应的保护，当有一个以上的 ERSG 输入信号时，PWMPCn[m+1:m] 保护的优先级为 00 > 01 > 10 > 11 模式。

举例来说：当 ERSG2 对应的 PWMPC0[5:4] 位设置为“11”模式，而 ERSG4 对应的 PWMPC1[1:0] 设置为“00”模式，此时先发生 ERSG2 的信号，那 ERSG 保护电路会依“11”模式做相关保护动作，而后 ERSG4 产生一触发信号时，因“00”模式优先权远高于“11”模式，那 ERSG 保护电路会立即依据 ERSG4 对应的 PWMPC1[1:0] 设置处理，做出相关的保护动作。

ERSG 保护机制动作方式

针对每一个 ERSG 触发信号，ERSG 保护电路提供了四种保护机制，可通过各自对应的 PWMPCn[m+1:m] 位进行选择，其中 n=0~2；m=0、2、4、 6。

当 PWMPCn[m+1:m]=00B：PWMP 不会自动调变，硬件会自动将 PWM 立即关闭，且将 MSS0 与 MSS1 位设置为 1，使 PWM0/PWM2 和 PWM1/PWM3 输出 MSD0 和 MSD1 的数据，PWMON 位由 1 变 0，FPWMCLF 位会被设为 1，完成 PWM 关闭电路。

当 PWMPCn[m+1:m]=01B：PWMP 不会自动调变，待当前整个 PWM 输出周期完成后，硬件会自动将 PWM 关闭且将 MSS0 与 MSS1 置为 1，使 PWM0/PWM2 和 PWM1/PWM3 输出 MSD0 和 MSD1 的数据，PWMON 位由 1 变 0，FPWMCLF 位会被设为 1，完成 PWM 关闭电路。

当 PWMPCn[m+1:m]=10B：PWMP 会在每一个 PWM 周期调变一次，且 ATMPF 位会被硬件设 1，若 (PWMP – DECPWMP) ≥ PWMMINP，就把此差值存入 PWMP 寄存器，直到 (PWMP – DECPWMP) < PWMMINP 后，此差值不存入 PWMP 寄存器，若中途 ATMPC 位 = 1，将会停止 PWM 自动调变动作，待执行到最后一次调变所对应的 PWM 输出周期完成后，硬件会自动将 PWM 关闭且将 MSS0 与 MSS1 置为 1，使 PWM0/PWM2 和 PWM1/PWM3 输出 MSD0 和 MSD1 的数据，PWMON 位由 1 变 0，FPWMCLF 位会被设为 1，完成 PWM 自动调变及关闭电路。

当 PWMPCn[m+1:m]=11B 且 ATMNPC=0：PWMP 会在每一个 PWM 周期调变一次，且 ATMPF 位会被硬件设 1，若 (PWMP – DECPWMP) ≥ PWMMINP，就把此差值存入 PWMP 寄存器，直到 (PWMP – DECPWMP) < PWMMINP，此差值不存入 PWMP 寄存器，若中途 ATMPC 位 = 1，将会停止 PWM 自动调变动作，完成 PWM 自动调变电路，PWMON 和 FPWMCLF 保持不变。

当 PWMPCn[m+1:m]=11B 且 ATMNPC=1 (仅对 ERSG1 及 ERSG3 信号有效)：PWMP 会在每一个 PWM 周期调变一次，且 ATMPF 位会被硬件设 1，若 (PWMP – DECPWMP) ≥ PWMMINP，就把此差值存入 PWMP 寄存器，直到 (PWMP – DECPWMP) < PWMMINP，此差值不存入 PWMP 寄存器。在此过程中若触发 PWM 自动调变及关闭电路的信号变为 0 时，将会停止 PWM 自动调变动作。若中途 ATMPC 位 = 1，也会使 PWM 自动调变动作停止。完成 PWM 自动调变电路，PWMON 和 FPWMCLF 保持不变。

注意：

1. 当 ERSG 保护电路还没运作时，改动 PWMPCn[m+1:m] 内容，新设置可立即生效。但 ERSG 保护电路开始运作时，改动 PWMPCn[m+1:m] 内容，不会立即生效，必须等到 ERSG 保护电路运作结束后，PWMPCn[m+1:m] 内容，才会改变生效，等待下一次的触发信号。
2. 当 PWMPCn[m+1:m] 为 00 或 01 时，有一 ERSG 中断触发信号，PWMP 不自动调变；若 PWMPCn[m+1:m] 为 10 或 11 时，有一 ERSG 中断触发信号，PWMP 与 PWMD 会在每一个 PWM 周期中调变一次，如果 (PWMP – DECPWMP) ≥ PWMMINP，硬件将 (PWMP – DECPWMP) 的值存入 PWMP 且 (PWMD – (DECPWMP/2)) 的值存入 PWMD，并重复此流程一直到 (PWMP – DECPWMP) < PWMMINP 后，(PWMP – DECPWMP) 的值不再存入 PWMP 结束此流程。下表为范例：

PWM 周期	ERSG INT	PWMPCn [m+1:m]	PWMMINP	DECPWMP	ATMPF	PWMP	PWMD
1	0	x	400	30	0	499	249
2	0	x	400	30	0	499	249
3	1	00 或 01	400	30	0	499	249
4	1	10 或 11	400	30	1	469	234
5	1	10 或 11	400	30	1	439	219
6	1	10 或 11	400	30	1	409	204
7	1	10 或 11	400	30	1	409	204

“x”：无关

下表为 ERSG1 信号 = 1，ATMNPC=1 且 PWMPC0[3:2]=11 之范例：

PWM 周期	ERSG1 INT	PWMMINP	DECPWMP	ATMPF	PWMP	PWMD
1	0	400	30	0	499	249
2	0	400	30	0	499	249
3	1	400	30	1	469	234
4	1	400	30	1	439	219
5	0	400	30	1	439	219
6	0	400	30	1	439	219
7	0	400	30	1	439	219

注：1. DECPWMP/2 若不是整数，小数无条件舍去。

2. 须注意当 PWMP 在自动调变时，硬件会限制 PWMP、PWMMINP、DECPWMP 和 PWMD 不让软件改写。

抖频功能

抖频功能主要可以分成 5 种工作模式，分别为软件 - 计数模式、软件 - 逼近模式、硬件 - 半计数 / 半逼近模式、硬件 - 全逼近模式以及硬件 - 半逼近模式，下表说明该如何设置相关位进行模式选择以及是否要搭配 FJTIMER 一起使用。

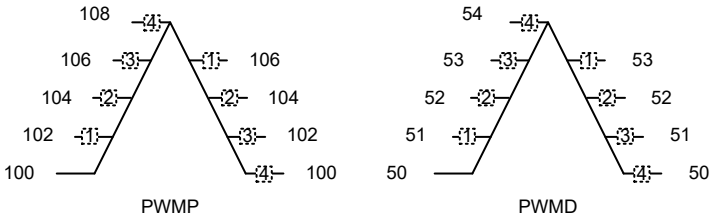
模式	名称	相关位设定			搭配使用 FJTIMER
		FJWMD	FJMDS	FJHMDS	
1	软件 - 计数模式	0	0	x	否
2	软件 - 逼近模式	0	1	x	否
3	硬件 - 半计数 / 半逼近模式	1	0	0	是
4	硬件 - 全逼近模式	1	1	0	是
5	硬件 - 半逼近模式	1	x	1	是

“x”：无关

由上表可以看出抖频的动作模式有两种分别为计数模式及逼近模式，以下说明两种模式的差异。

计数模式

计数模式下，主要是利用 FJM0C0~FJM0C2 寄存器来控制 PWMP、PWMD、DT0 以及 DT1 的变化。PWMP 及 PWMD 的改变方式是由 FJM0C0 来控制，利用 CFJCNT[2:0] 位决定步进的次数；CFJS 位来决定 PWMP 及 PWMD 的数值是要先加上 CFJSA[2:0] 所设定的量还是先减少 CFJSA[2:0] 所设定的量，这边要注意的是当 CFJCNT[2:0] 设定的次数符合时，PWMP 及 PWMD 就会以相反的方向减少或增加。例如：当 CFJCNT[2:0]=011 (四次)，CFJS=0 &CFJSA[2:0]=000 (Period+2; Duty+1)，PWMP=100，PWMD=50，PWMD 及 PWMP 的变化如下：



死区时间 0(DT0) 及死区时间 1(DT1) 的改变方式分别由 FJM0C1 和 FJM0C2 寄存器来控制。CFJDTnEN 位决定是否启用硬件自动调整 DTn；CFJDTnCNT[1:0] 位决定触发几次 PWM 周期后才调整 DTn；CFJDTnS 位决定每次 DTn 调整中，DTn 的数值是要加上 CFJDTnSA[1:0] 所设定的量还是要减去 CFJDTnSA[1:0] 所设定的量。以下以一个具体的范例来说明。

● 抖频计数模式下寄存器及相关设定如下表所示：

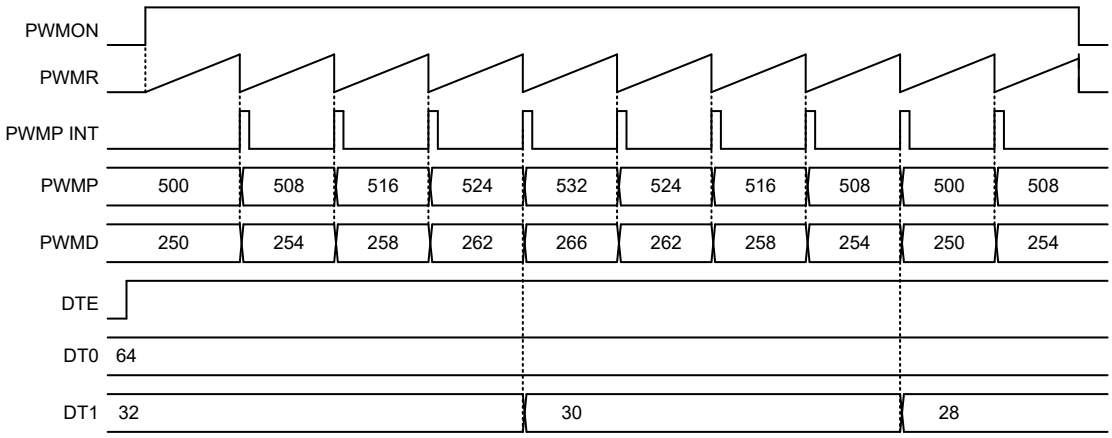
寄存器	数值	说明
PWMP	500	
PWMD	250	
DT0	64	
DT1	32	

寄存器	数值	说明
FJM0C0	33H	步进次数 =4； Period 数值加 8； Duty 数值加 4；
FJM0C1	13H	DT0 调整除能； 2 次触发后，DT0 数值加 4；
FJM0C2	B9H	DT1 调整使能； 4 次触发后，DT1 数值减 2；

● 执行后结果如下：

次数	寄存器				对应内部计数器值			
	PWMP	PWMD	DT0	DT1	PWMP	PWMD	DT0	DT1
0	500	250	64	32	500	250	64	32
1	508	254	64	32	500	250	64	32
2	516	258	64	32	508	254	64	32
3	524	262	64	32	516	258	64	32
4	532	266	64	30	524	262	64	32
1	524	262	64	30	532	266	64	30
2	516	258	64	30	524	262	64	30
3	508	254	64	30	516	258	64	30
4	500	250	64	28	508	254	64	30
1	508	254	64	28	500	250	64	28

● PWM 波形如下：



逼近模式

逼近模式下，主要是利用 FJM1C0~FJM1C3 寄存器来控制 PWMP、PWMD、DT0 以及 DT1 的变化。PWMP 及 PWMD 的逼近目标值是由 FJPA 及 FJPB 寄存器来决定。PWMP 和 PWMD 的调变方式由 FJM1C0 来控制。利用 FJCNT[2:0] 位决定触发的次数；FJSA[2:0] 位来决定 PWMP 及 PWMD 的数值增加或减少的量，另外 FJCNT[2:0] 以及 FJSA[2:0] 在抖频功能启动过程中，可以通过 FJM1C1 的寄存器设定在几次的调变后，FJCNT[2:0] 以及 FJSA[2:0] 是否要做“加一 / 减一”的变动或者不更动。

DT0 及 DT1 逼近的目标值由对应的 FJDTnA 或 FJDTnB 来决定，DT0 和 DT1 的调变方式分别由 FJM1C2 及 FJM1C3 来控制，利用 FJDTnEN 位来决定 DTn 是否要启动自动调变；FJDTnCNT[1:0] 位来决定几次的 PWM 触发后，DTn 要调变一次，而 DTn 每次调变的修改数值由 FJDTnSA[2:0] 位决定。另外 FJDTnSA[2:0] 可以通过 FJDTnASA[1:0] 位决定在硬件自动修改的过程中是否要做“加一 / 减一”的变动或者不更动。

以下以逼近 FJPA/FJDT0A/FJDT1A 来举例说明。

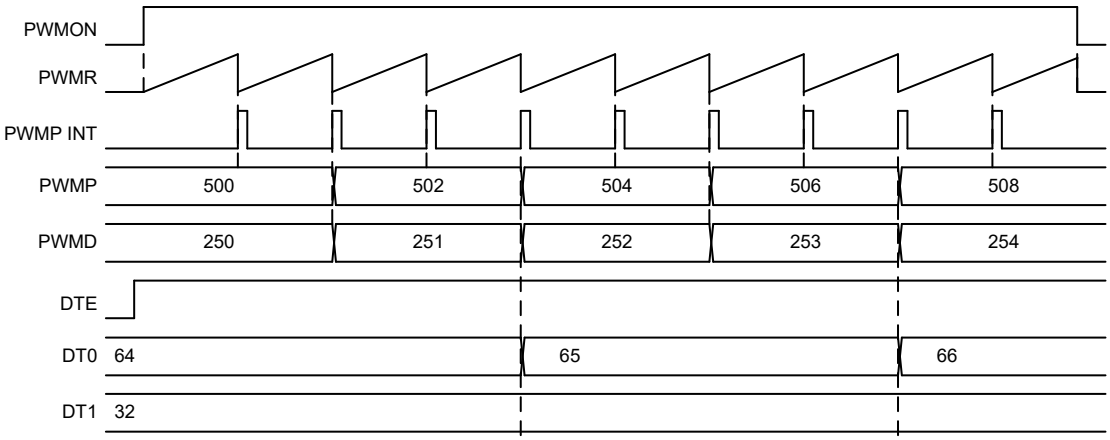
● 抖频逼近模式下寄存器及相关设定如下表所示：

寄存器	数值	说明
PWMP	500	
PWMD	250	
DT0	64	
DT1	32	
FJPA	510	
FJDT0A	68	
FJDT1A	30	
FJM1C0	10H	2 次触发后，Period+2，Duty+1
FJM1C1	10H	2 次逼近后，次数不变，数值不变
FJM1C2	80H	DT0 调整使能； 4 次触发后，DT0 数值 ±1；
FJM1C3	0CH	DT1 调整除能； 8 次触发后，DT1 数值 ±5；

● 执行后结果如下：

次数	寄存器							对应内部计数器值			
	PWMP	PWMD	DT0	DT1	FJPA	FJDT0A	FJDT1A	PWMP	PWMD	DT0	DT1
0	500	250	64	32	510	68	30	500	250	64	32
1	500	250	64	32				500	250	64	32
2	502	251	64	32				500	250	64	32
3	502	251	64	32				502	251	64	32
4	504	252	65	32				502	251	64	32
5	504	252	65	32				504	252	65	32
6	506	253	65	32				504	252	65	32
7	506	253	65	32				506	253	65	32
8	508	254	66	32				506	253	65	32
9	508	254	66	32				508	254	66	32

● 波形：



以上初步介绍两个模式 (计数模式和逼近模式) 的动作原理，以下将详细说明这五种工作模式相关的设置寄存器，以及需搭配哪些周边功能进行使用，将分成两类做介绍 (软件和硬件)。

软件抖频

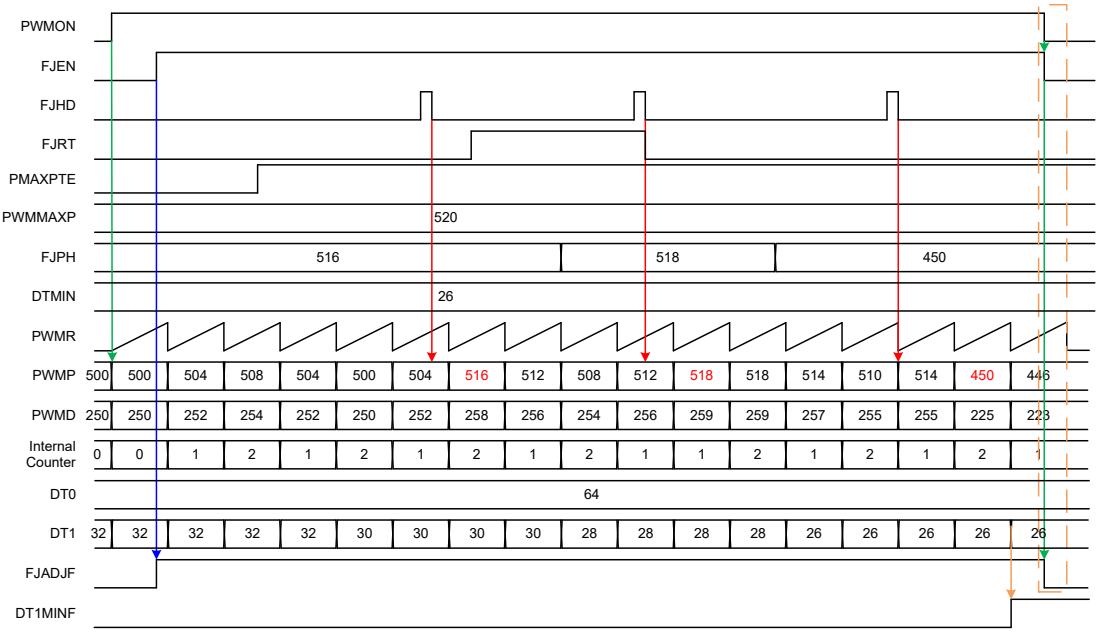
使用软件抖频功能时，用户将 FJEN 位设置为 1 后，PWM 周期依据抖频相关寄存器做对应调变，需注意当有 ERSG 中断发生或者 ERSG 自动调变功能开启即 ATMPF=1 时，若 FJEN=1 那硬件会将 FJEN 位清为 0，若此时软件将 FJEN 位由 0 变 1，那硬件会忽略此次触发，并将 FJEN 位清 0。

模式	名称	设定位			调变寄存器	
		FJWMD	FJMDS	FJSMDS	FJM0Cn (n=0~2)	FJM1Cn (n=0~3)
1	软件 - 计数模式	0	0	x	Y	x
2-1	软件 - 逼近模式 (A)	0	1	0	x	Y
2-2	软件 - 逼近模式 (B)	0	1	1	x	Y

● 模式 1：软件 - 计数模式

设定如下：

寄存器	数值	说明
PWMP	500	
PWMD	250	
DT0	64	
DT1	32	
FJM0C0	11H	步进次数 = 2； Period 数值加 4； Duty 数值加 2；
FJM0C1	13H	DT0 调整除能； 2 次触发后，DT0 数值加 4；
FJM0C2	B9H	DT1 调整使能； 4 次触发后，DT1 数值减 2；

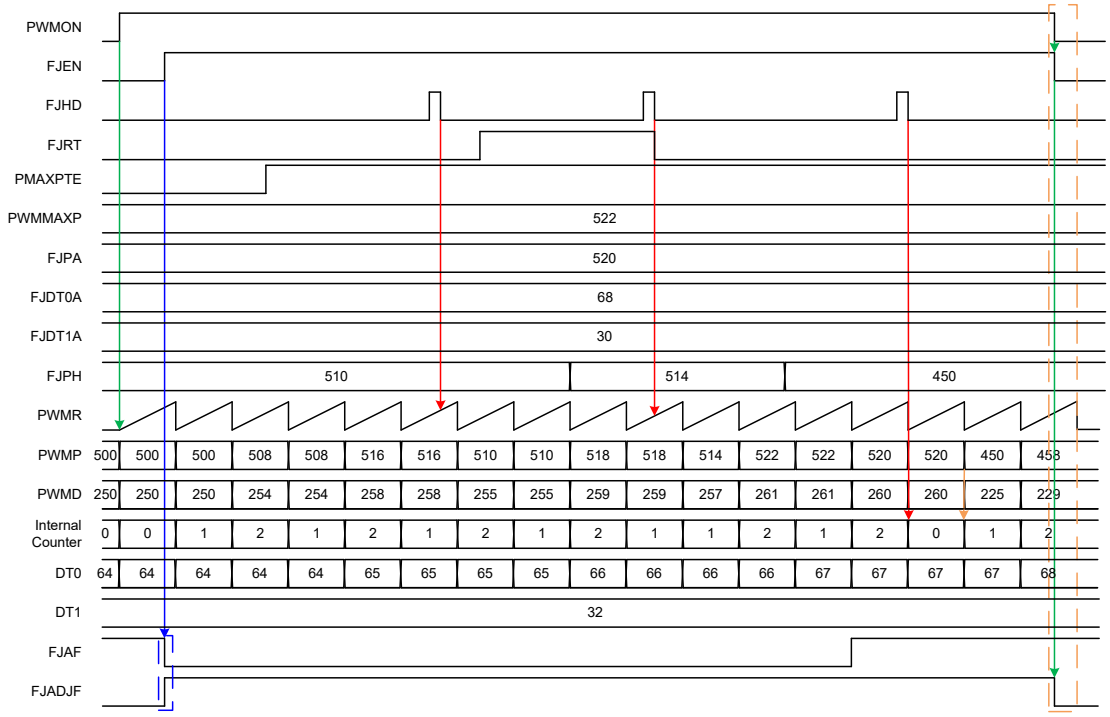


- 模式 2-1：软件 - 逼近模式 (A)

  以下给出了两个范例，说明相关寄存器设置及对应的时序图。

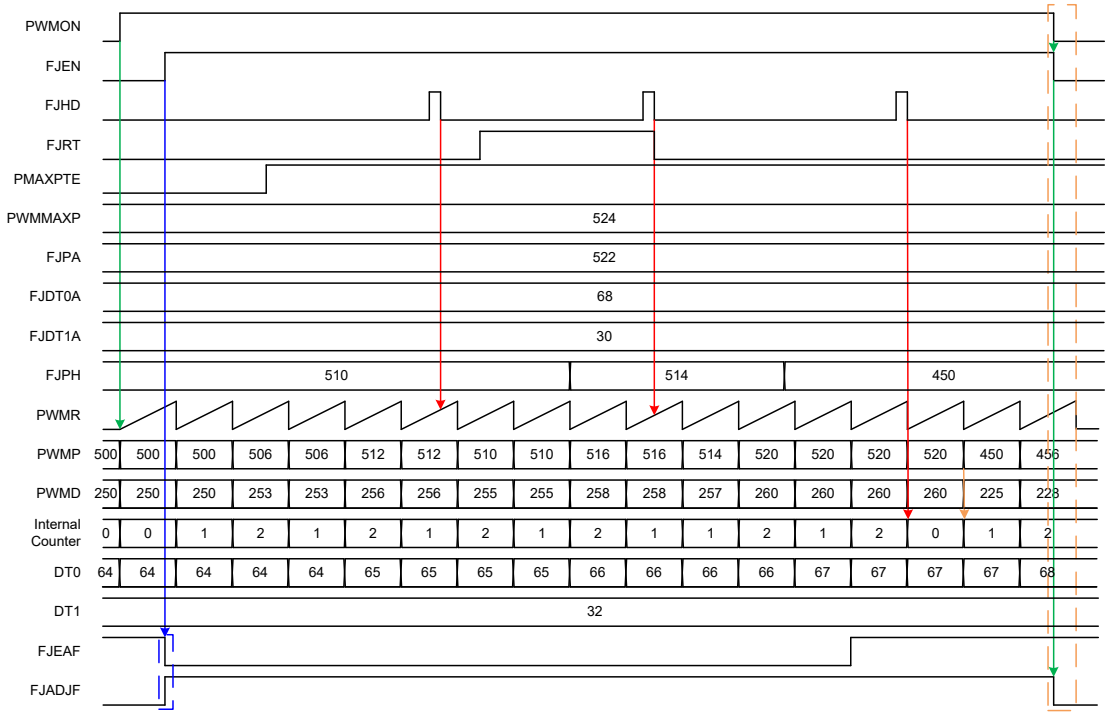
  范例 1

寄存器	数值	说明
PWMP	500	
PWMD	250	
DT0	64	
DT1	32	
FJPA	520	
FJDT0A	68	
FJDT1A	30	
FJM1C0	15H	2 次触发后，Period+8，Duty+4
FJM1C1	10H	2 次逼近后，次数不变，数值不变
FJM1C2	80H	DT0 调整使能； 4 次触发后，DT0 数值 ±1；
FJM1C3	0CH	DT1 调整除能； 8 次触发后，DT1 数值 ±5；



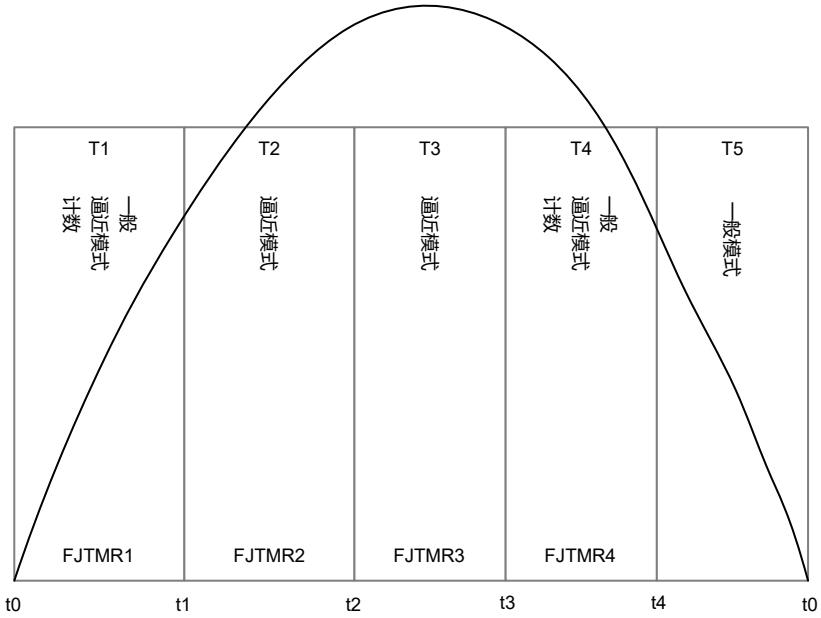
范例 2

寄存器	数值	说明
PWMP	500	
PWMD	250	
DT0	64	
DT1	32	
FJPA	522	
FJDT0A	68	
FJDT1A	30	
FJM1C0	14H	2 次触发后，Period+6，Duty+3
FJM1C1	10H	2 次逼近后，次数不变，数值不变
FJM1C2	80H	DT0 调整使能； 4 次触发后，DT0 数值 ±1；
FJM1C3	0CH	DT1 调整除能； 8 次触发后，DT1 数值 ±5；



硬件抖频

硬件自动调变模式，会使用 FJTIMER 的 4 个计时区间 (T1~T4) 来决定 PWM 周期该如何变化。触发启动 FJTIMER 的方式有两种，通过 FJTSS 位来选择是利用外部信号上升沿或软件设置 FJTON 位为 1 来启动 FJTIMER。其中外部信号又可通过 FJOTSS 位进一步选择，是使用 OVP7OS 或是 FJTR1 引脚输入信号。PWM 周期在 FJTIMER 的四个计时区间内会有不同的动作，如下图所示，其中第一计时区间与第四计时区间可以一起说明，第二计时区间与第三计时区间可以一起说明。



a. 第一计时区间 (t0~t1, FJTMR1) / 第四计时区间 (t3~t4, FJTMR4)：有三种工作模式，动作分别如下：
- 计数模式：利用 FJM0C0~FJM0C2 来控制。
- 逼近模式：利用 FJM1C0~FJM1C3 来控制，但须注意 FJM1C1 及 FJM1C2/FJM1C3 中的 FJDTnASA[1:0] 在这两个计时区间设定为无效，也就是说 FJCNT[2:0]、FJSA[2:0] 和 FJDTnSA[2:0] 设定值不会做调变，第一计时区间针对 FJPB/FJDT0B/FJDT1B 调变，第四计时区间针对 FJPA/FJDT0A/FJDT1A 调变。
- 不调变模式 (一般模式)：PWMP 及 PWMD 不受 FJM0C0~FJM0C2 及 FJM1C0~FJM1C3 控制，也就是不做自动调变，可以通过软件做修改。

b. 第二计时区间 (t1~t2, FJTMR2) / 第三计时区间 (t2~t3, FJTMR3)：只有一种工作模式，那就是逼近模式，利用 FJM1C0~FJM1C3 来控制，FJCNT[2:0]、FJSA[2:0] 和 FJDTnSA[2:0] 设定值会受 FJM1C1 及 FJM1C2/FJM1C3 中的 FJDTnASA[1:0] 设定做调变，第二计时区间针对 FJPA/FJDT0A/FJDT1A 调变，第三计时区间针对 FJPB/FJDT0B/FJDT1B 调变。

c. 不计时区间 (t4~t0)：PWMP 及 PWMD 不受 FJM0C0~FJM0C2 及 FJM1C0~FJM1C3 控制，也就是不做自动调变，可以通过软件做修改。

模式	名称	设定位			调变寄存器	
		FJWMD	FJMDS	FJHMDS	FJM0Cn (n=0~2)	FJM1Cn (n=0~3)
3	硬件 - 半计数 / 半逼近模式	1	0	0	Y	Y
4	硬件 - 全逼近模式	1	1	0	x	Y
5	硬件 - 半逼近模式	1	x	1	x	Y

设定步骤：

步骤 1：写入 PWMP、PWMD、DT0、DT1、PWMMAXP、PWMMINP 初始值

步骤 2：写入 FJPA、FJPB、FJDT0A、FJDT0B、FJDT1A、FJDT1B、DTMIN 初始值

步骤 3：设定 PWM & 相位保护相关寄存器：PWMC0、PWMC1、PWMC2、PLC、PPDSITA 相关设定值

步骤 4：设定关于 ERSG 保护相关寄存器：PWMPC0、PWMPC1、PWMPC2、ERSGC、DECPWMP

步骤 5：设定 FJTIMER 相关寄存器：FJTMC、FJTMR1~FJTMR4

步骤 6：设定抖频 - 计数模式相关寄存器：FJM0C0~FJM0C2

步骤 7：设定抖频 - 逼近模式相关寄存器：FJM1C0~FJM1C3

步骤 8：设定其它 PWM 相关寄存器，此章节在描述逼近模式，故其它寄存器不多加解释。

步骤 9：如果要使用中断，则中断控制寄存器需要正确地设置，以确保逼近匹配中断功能是有效的。总中断控制位 EMI，相关多功能中断控制位以及逼近匹配中断位 FJEQUE 需要置位为“1”。

步骤 10：设定抖频功能相关寄存器：FJC0 (注意 FJWMD 位要在以上设定都完成后才可以设 1，否则当 FJTSS=0 时，外部信号有一触发信号就会启动抖频功能，可能会发生不可预期的情况)。

步骤 11：若 |PWMP-FJPA| ≤ FJSA 或 |PWMP-FJPB| ≤ FJSA 时，硬件会将 FJPA 或 FJPB 写到 PWMP，同时将 FJAF 或 FJBF 设置为”1”，若 PMAXPTE=1，PWMP+FJSA > PWMMAXP 时，PWMP 不会加 FJSA 的值，且硬件会将 FJEAF 或 FJEBF 设置为”1”，故使用者可以使用轮询 FJAF 或 FJBF 或 FJEAF 或 FJEBF 来判别 PWMP 是否已与 FJPA 或 FJPB 相等或已接近最大值。

步骤 12：读取 FJTMMD[2:0] 可以知道目前 FJTIMER 工作模式为何。

注：1. 若使用轮询 FJAF/FJBF/FJEAF 或 FJEBF 的方法，则上述的启动中断步骤可以忽略。

2. 进入中断向量后需再读取 FJAF/FJBF/FJEAF 或 FJEBF 是否为”1”，确保已正确逼近所设定之值。

3. 当 |PWMP-FJPA| > FJSA 或 |PWMP-FJPB| > FJSA 时，则 PWMP 以 PWMP-FJSA 写入；

   a. 若 PWMD±FJSA > 0，则 PWMD 以 PWMD±FJSA 写入；

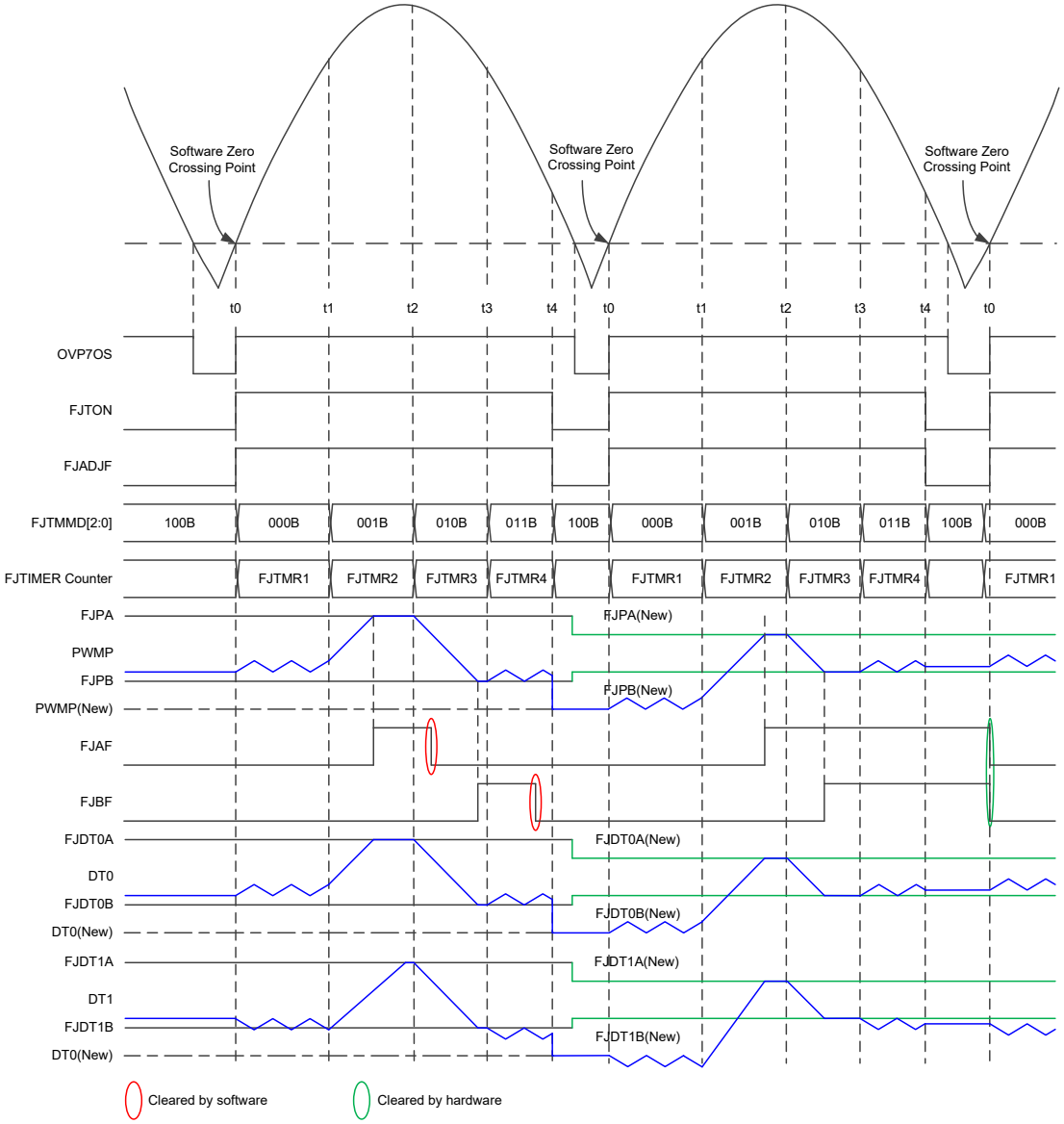
   b. 若 PWMD±FJSA < 0，则 PWMD 不修改。

4. 当 |PWMP-FJPA| ≤ FJSA 或 |PWMP-FJPB| ≤ FJSA 时，则 PWMP 以 FJPA 或 FJPB 写入；

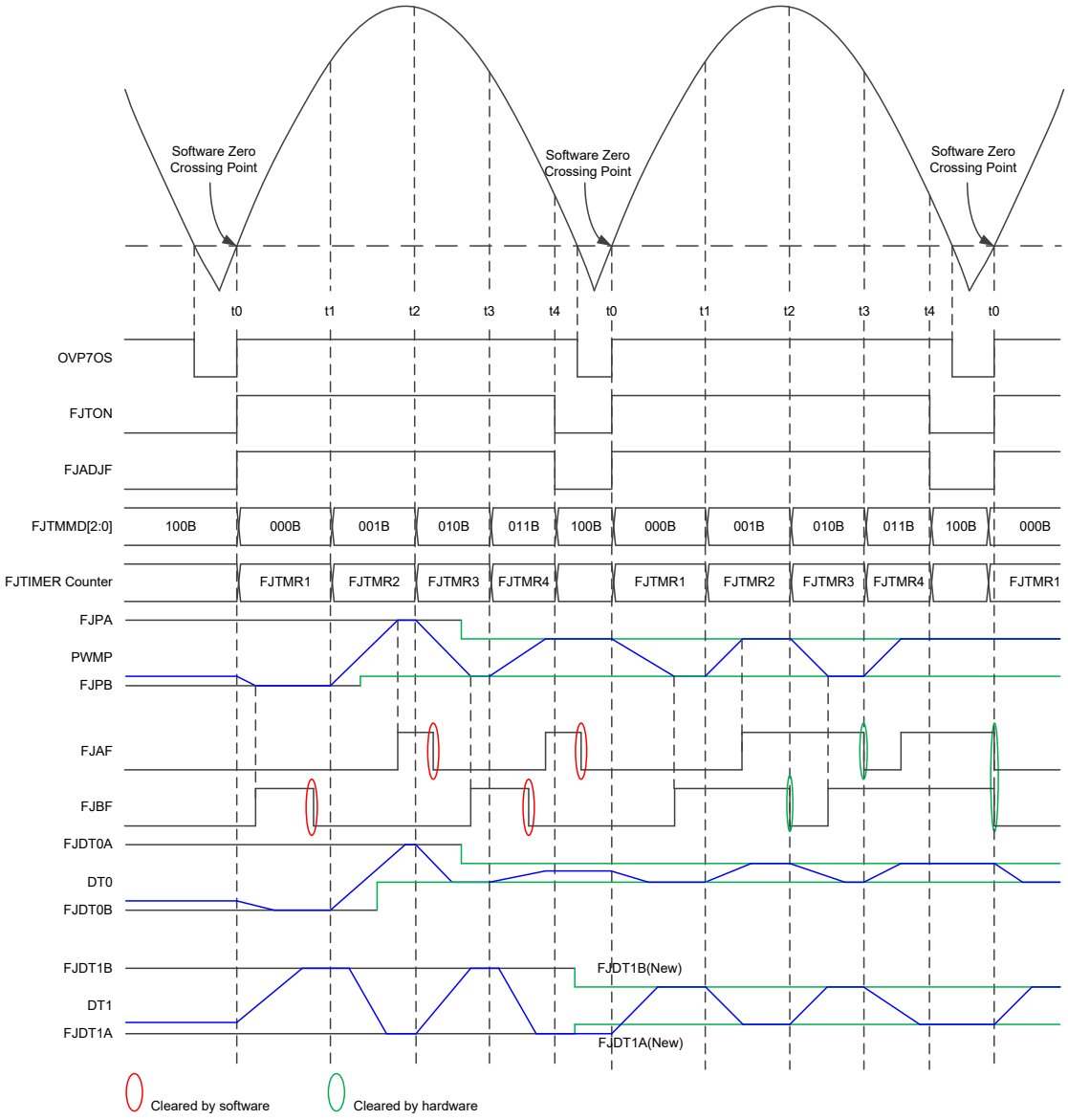
   a. 若 PWMD±FJSA > 0，则 PWMD 以 PWMD±(PWMP-FJPA)/2 或 PWMD±(PWMP-FJPB)/2 (小数无条件舍去) 写入；

   b. 若 PWMD±FJSA < 0，则 PWMD 不修改。

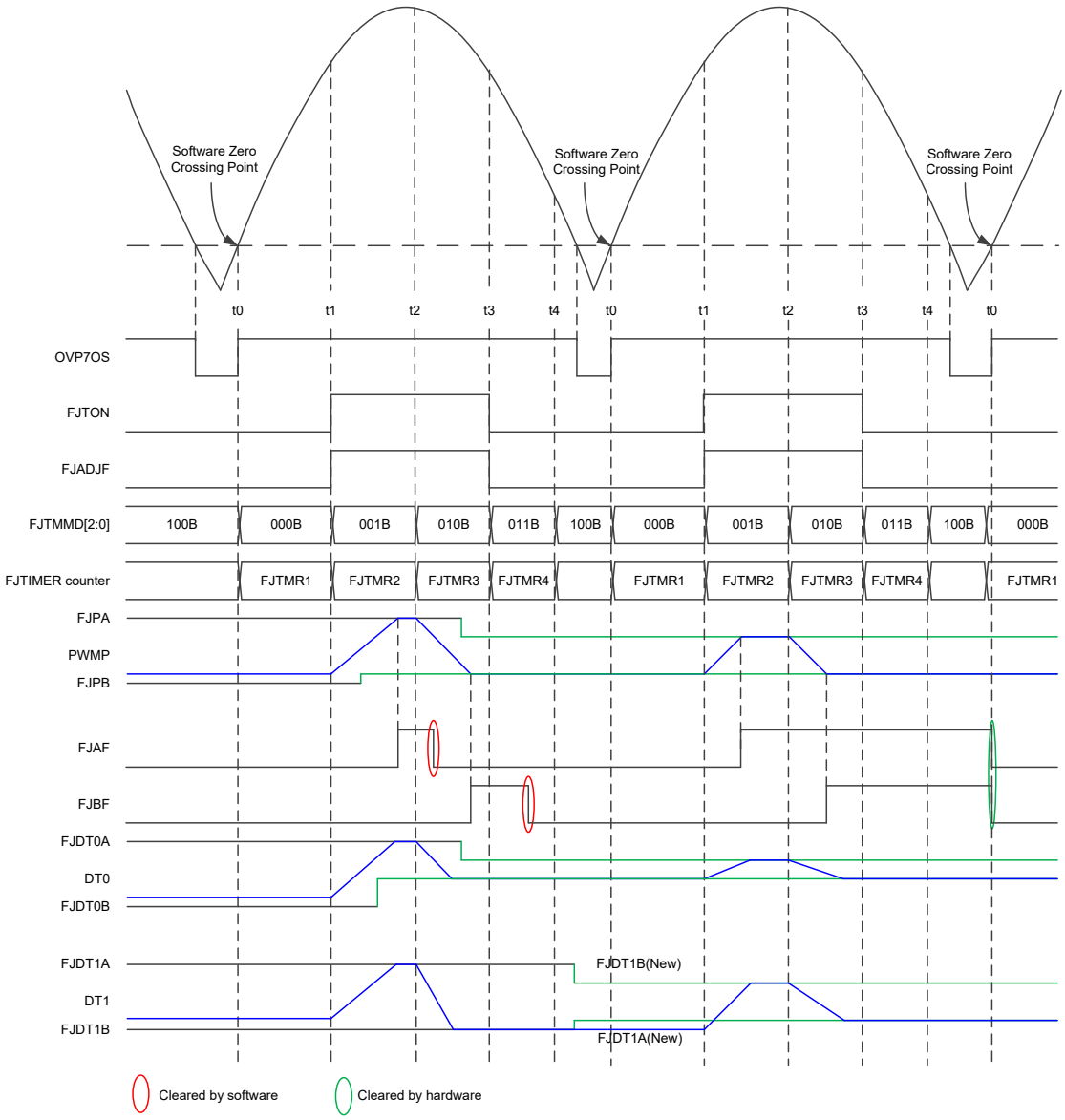
● 模式 3：硬件 - 半计数 / 半逼近模式



● 模式 4：硬件 - 全逼近模式


Software Zero Crossing Point
Software Zero Crossing Point
Software Zero Crossing Point
t0
t1
t2
t3
t4
OVP7OS
FJTON
FJADJF
FJTMMD[2:0]
100B
000B
001B
010B
011B
FJTIMER Counter
FJTMR1
FJTMR2
FJTMR3
FJTMR4
FJPA
PWMP
FJPB
FJAF
FJBF
FJDT0A
DT0
FJDT0B
FJDT1B
FJDT1B(New)
DT1
FJDT1A
FJDT1A(New)
Cleared by software
Cleared by hardware

● 模式 5：硬件 - 半逼近模式


Software Zero Crossing Point
Software Zero Crossing Point
Software Zero Crossing Point
t0
t1
t2
t3
t4
OVP7OS
FJTON
FJADJF
FJTMMD[2:0]
100B
000B
001B
010B
011B
FJTIMER counter
FJTMR1
FJTMR2
FJTMR3
FJTMR4
FJPA
PWMP
FJPB
FJAF
FJBF
FJDT0A
DT0
FJDT0B
FJDT1A
DT1
FJDT1B
FJDT1B(New)
FJDT1A(New)
Cleared by software
Cleared by hardware

启动抖频功能

不同的工作模式启动抖频功能的方法也不同，若是软件模式就是 F/W 将 FJEN 设置为 1 即可启动，若是硬件模式则是可以选择 F/W 将 FJTON 设置为 1 或外部信号有一上升沿时即可启动，但不论是在软件模式或硬件模式都需要注意以下几点：

1. 当 PWMON=0 时，不论 FJEN 为 1 或是为 0 都不会改变 PWMP 及抖频模式中内部计数器值，因抖频内部计数器是利用 PWMP 溢出时加 1 或 PWMON 由 0 变 1 时加 1。
2. 当 FJEN=1 时，有一 ERSG 中断产生，FJEN 会被硬件清 0，PWM 信号由 ERSG 保护电路控制。
3. 当 ATMPF=1 时，若此时软件将 FJEN 位设为 1 (软件抖频模式) 或是 FJTON 设 1 或接收到一外部触发信号 (硬件抖频模式)，这些都将被忽略且硬件会将其设 1 信号清 0。

关闭抖频功能

关闭抖频功能可以由软件将 PWMON 或 FJEN 清为 0，另外，当有 ERSG 中断发生，抖频功能会立即停止，FJTON (硬件抖频模式) 及 FJEN 位会被硬件清零。

PWMON	FJEN	ERSG INT	PWMPCn [m+1:m]	ATMPF	FPWMCLF	PWM 自动调变	PWM 功能	抖频功能
0	x	x	x	x	x	N	除能	除能
1	0	0	x	x	x	N	使能	除能
1	0	↑	00	0	1	N	使能 → 除能	除能
			01	0	1	N	使能 → 除能	除能
			10	1	0 → 1	Y (保护)	使能 → 除能	除能
			11	1	0	Y (保护)	使能	除能
1	1	0	x	x	x	Y (抖频)	使能	使能
1	1	↑	00	0	1	N	使能 → 除能	使能 → 除能
			01	0	1	N	使能 → 除能	使能 → 除能
			10	1	0 → 1	Y (保护)	使能 → 除能	使能 → 除能
			11	1	0	Y (保护)	使能	使能 → 除能

启动触发信号选择

启动硬件抖频功能的方式可以通过 FJTSS 及 FJOTSS 位来选择，是外部信号或是软件设置 FJTON 位由 0 变 1 时启动硬件抖频功能。

FJTSS	FJOTSS	信号说明
0	0	OVP7OS
0	1	FJTR1 引脚输入
1	x	FJTON 位控制

FJTIMER 工作状态

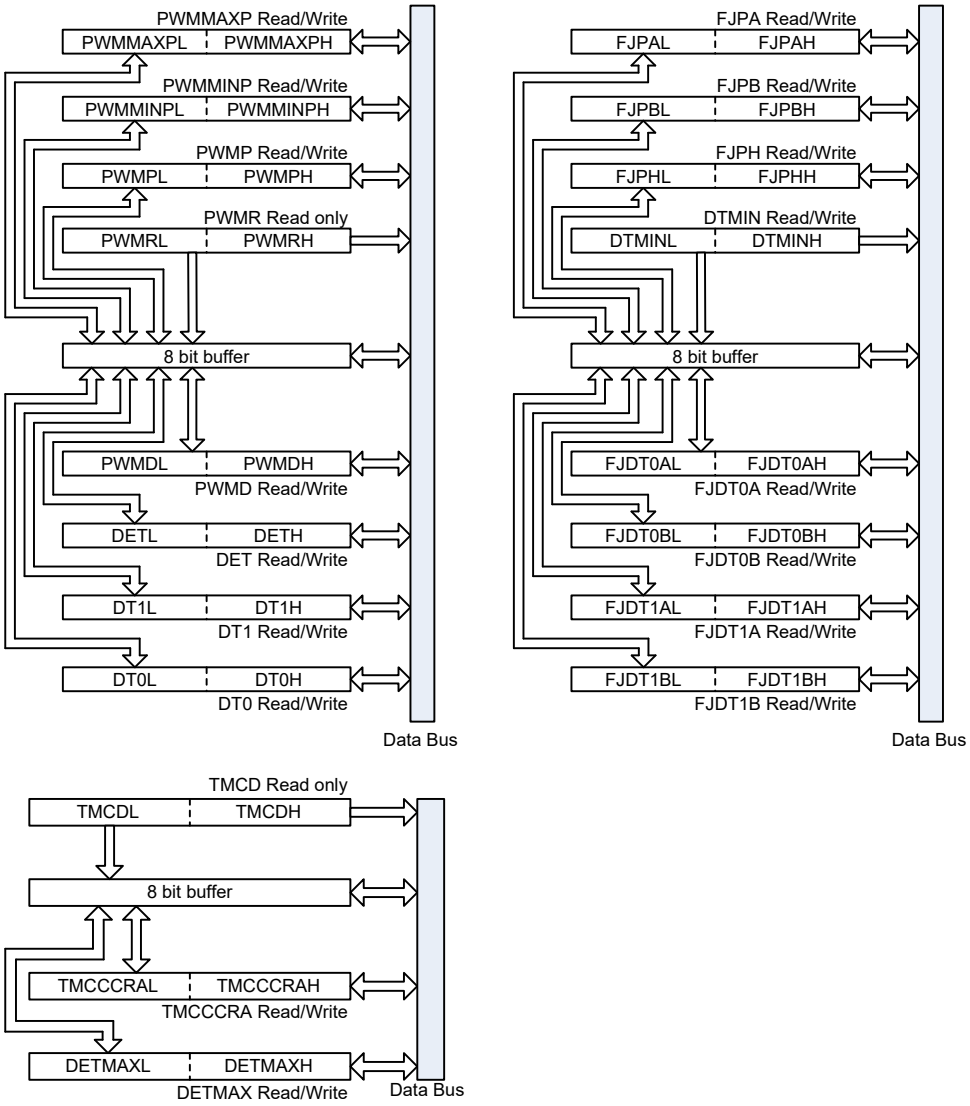
PWMON	FJWMD	FJTSS	FJOTSS	说明
0	X	X	X	一般定时器模式，由 FJTMR1 载入计数； FJTON 由软件设定。
↑	0	X	X	一般定时器模式，由 FJTMR1 载入计数； FJTON 由软件设定。
	1	X	X	若 FJTON=1 时，将 PWMON 设为 1， FJTON 会被硬件清为 0。
1	0	X	X	一般定时器模式，由 FJTMR1 载入计数； FJTON 由软件设定。
	1	0	0	硬件抖频模式，由 FJTMR1~4 加载计数； FJTON 由 OVP7OS 信号触发设定。
			1	硬件抖频模式，由 FJTMR1~4 加载计数； FJTON 由 FJTR1 信号触发设定。
		1	X	硬件抖频模式，由 FJTMR1~4 加载计数； FJTON 由软件设定。
	↑	X	X	若 FJTON=1 时，将 FJWMD 设为 1，FJTON 会被硬件清为 0。
	↓	X	X	抖频模式关闭，FJTON 不会被硬件清为 0。 作为一般定时器模式，由 FJTMR1 载入计数。

编程注意事项

在编程过程中，以下事项需注意。

数据寄存器存取

PWMR、PWMP、PWMD、DT0、DT1、PWMMAXP、PWMMINP、FJPA、FJPB、FJPH、FJDT0A、FJDT0B、FJDT1A、FJDT1B 和 DTMIN 为 12-bit 寄存器，DET 和 DETMAX 为 11-bit 寄存器，TMCD 和 TMCCCRA 为 10-bit 寄存器，均具有高字节与低字节的结构，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。这些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。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。



读写流程如下步骤所示：

- 写数据至 PWMP、PWMD、DT0、DT1、PWMMAXP、PWMMINP、DET、DETMAX、FJPA、FJPB、FJPH、FJDT0A、 FJDT0B、FJDT1A、FJDT1B、DTMIN 和 TMCCCRA
  - ♦ 步骤 1. 写数据至低字节寄存器 PWMPL/PWMDL/DT0L/DT1L/PWMMAXPL/PWMMINPL/DETL/DETMAXL/FJPAL/FJPBL/FJPHL/FJDT0AL/FJDT0BL/FJDT1AL/FJDT1BL/DTMINL/TMCCCRAL
    - 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 PWMPH/PWMDH/DT0H/DT1H/PWMMAXPH/PWMMINPH/DETH/DETMAXH/FJPAH/FJPBH/FJPHH/FJDT0AH/FJDT0BH/FJDT1AH/FJDT1BH/DTMINH/TMCCCRAH
    - 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 PWMP、PWMD、DT0、DT1、PWMMAXP、PWMMINP、DET、DETMAX、FJPA、FJPB、FJPH、FJDT0A、 FJDT0B、FJDT1A、FJDT1B、DTMIN、TMCCCRA 中读取数据
  - ♦ 步骤 1. 由高字节寄存 PWMRH/PWMPH/PWMDH/DT0H/DT1H/PWMMAXPH/PWMMINPH/DETH/DETMAXH/FJPAH/FJPBH/FJPHH/FJDT0AH/FJDT0BH/FJDT1AH/FJDT1BH/DTMINH/TMCDH/TMCCCRAH 读取数据
    - 注意，此时高字节寄存器中的数据直接读取，同时低字节寄存器中的数据锁存至 8-bit 缓存器中。
  - ♦ 步骤 2. 由低字节寄存器 PWMRL/PWMPL/PWMDL/DT0L/DT1L/PWMMAXPL/PWMMINPL/DETL/DETMAXL/FJPAL/FJPBL/FJPHL/FJDT0AL/FJDT0BL/FJDT1AL/FJDT1BL/DTMINL/TMCDL/TMCCCRAL 读取数据
    - 注意，此时读取 8-bit 缓存器中的数据。

PWMP/FJPA/FJPB/FJPH 最大值限制说明

a. 当 PMAXPTE=0 时，不论对 PWMP/FJPA/FJPB/FJPH 写任何值，都不会受到 PWMMAXP 的限制，且其相关的标志位 PMAXF/AMAXF/BMAXF/HMAXF 都为 0。

b. 当 PMAXPTE 由 0 变 1 时，此时硬件会去判别 PWMP/FJPA/FJPB/FJPH 是否大于 PWMMAXP，当 PWMP/FJPA/FJPB/FJPH > PWMMAXP，对应的 PMAXF/AMAXF/BMAXF/HMAXF 会被设为 1，反之清除为 0。

c. 当 PMAXPTE=1 时，PWMMAXP/PWMP/FJPA/FJPB/FJPH 有写入动作时，会触发最大值限制电路。该电路会去判别 PWMP/FJPA/FJPB/FJPH 是否大于 PWMMAXP，当 PWMP/FJPA/FJPB/FJPH > PWMMAXP 时，对应的 PMAXF/AMAXF/BMAXF/HMAXF 会被设为 1，反之清除为 0。

下表以向 PWMP 写入数据为例进行说明 (FJPA/FJPB/PJPH 同理)。

步骤	PMAXPTE	PWMMAXP	PWMP (写入)	PWMP (结果)	PMAXF
1	0	1000	500	500	0
2	0	1000	1024	1024	0
3	1	1000	—	1024	1
4	1	1000	1010	1024	1
5	1	1000	980	980	0
6	1	900	—	980	1
7	1	900	920	980	1
8	1	900	890	890	0
9	1	850	—	890	1
10	0	850	—	890	0

寄存器数据加载时机说明

a. PWMPCn[m+1:m] 修改：修改任意 PWMPCn[m+1:m] 设置，不论其相对应的输入信号当前为高还是为低，只要在 ERSG0~ERSG7 及 ERSGS 中有任一信号为 1，硬件都不会将其修改值载入，需等到 ERSG0~ERSG7 及 ERSGS 信号皆为 0 时才会更新 PWMPCn[m+1:m]。

b. PWMP、PWMD、DT0、DT1、DET 更新：通过软件对这些寄存器写入新的值之后，新数据不会立即加载到其内部的计数器，加载的时间点为 PWMR 计数器的值变为 0 时才会将 PWMP、PWMD、DT0、DT1、DET 寄存器数据载入到其内部的计数器中。

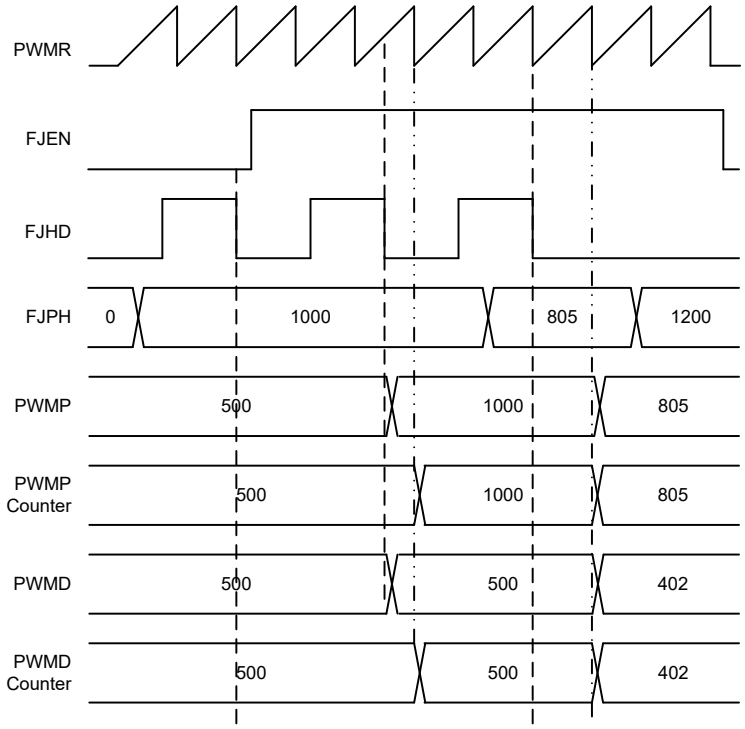
利用 FJPH 加载 PWMP 及 PWMD

当抖频功能启动时，PWMP 及 PWMD 由硬件自动调整其内容值，无法通过软件修改其内容值，若需要在抖频功能启动的情况下通过软件对 PWMP 及 PWMD 内容值进行修改，可以通过利用 FJC1 寄存器中的 FJHD 位搭配 FJPH 寄存器来实现对 PWMP 及 PWMD 的修改。修改的内容值是在下一个 PWM 周期才会加载，不会在当前的 PWM 周期就更新，需注意若软件修改的时间点与硬件修改的时间点相同或基本接近，则会优先加载硬件的数值，而软件的值则是等下一个 PWM 周期才会加载。

当软件将新的 PWMP 值写入到 FJPH 中后，对 FJHD 设 1→0 硬件会自动将 FJPH 寄存器的数据加载至 PWMP 以及将 FJPH 寄存器值除 2 后加载至 PWMD 寄存器，若除 2 后的值为非整数则无条件舍去小数，并于下一个 PWM 周期输出。若 FJHD 由 1 变 0 的时间点与 PWMP 重载时间 (PWMR 计数器值变为 0 时) 接近，硬件会将此次的软件修改延后 1 个 PWM 周期再加载，以下说明其用法。

模式	FJEN	接近硬件重载时间点？	从 FJPH 载入	说明
0	0	x	x	一般 PWM 模式
1	1	否	是	PWMP 及 PWMD 由 FJPH 载入
2	1	是	否	PWMP 及 PWMD 由抖频模式计算后载入
3	1	是	是	PWMP 及 PWMD 由抖频模式计算后载入
	1	是	否	PWMP 及 PWMD 由 FJPH 载入

“x”：无关



抖频功能正确启动并触发注意事项

抖频功能必须在无 ERSG 中断且 ATMPF=0 时才能正确启动。当 PWM 关闭时，FJEN 也会被硬件清为 0。软件在修改相关设置时，都会在一个完整的 PWM 周期完成后才会被正确的载入，不会立即载入修改值。

模式	PWMON	FJEN	PWM 保护电路	FJWMD	FJMDS	FJHMDS	说明
0	0	x	x	x	x	x	PWM 关闭
1	1	0	0	x	x	x	PWM 无自动更新功能
2	1	0	1	x	x	x	ERSG 保护模式
3	1	1	0	0	0	x	软件 - 计数模式
4	1	1	0	0	1	x	软件 - 逼近模式
5	1	1	0	1	0	0	硬件 - 半计数 / 半逼近模式
6	1	1	0	1	1	0	硬件 - 全逼近模式
7	1	1	0	1	x	1	硬件 - 半逼近模式

FJTIMER 设置注意事项

应用于电磁炉时，使用抖频 - 硬件模式，设置 FJTMR1~FJTMR4 寄存器需搭配 AC 半波频率来进行配置，若 (T1+T2+T3+T4) > 半波时间 (10ms @ 50Hz)，会发生 FJTMR4 时间尚未计数完毕又有一外部触发信号 (FJWMD=1 & FJTSS=0)，硬件会重新将 FJTMR1 加载至 FJTIMER 并开始计数，抖频相关的内部计数器会被清零并重新计数，而 FJTMMD[2:0] 会直接从 100 改为 000。

ATMPC 设置说明

a. 当 ATMPF=0 时软件设定 ATMPC=1，当发生 ERSG 电路保护触发 ATMPF=1 时，不会启动硬件自动调变电路，且硬件会在当前 PWM 周期结束时将 ATMPC 清 0，软件需要自行将 ATMPF 清 0。

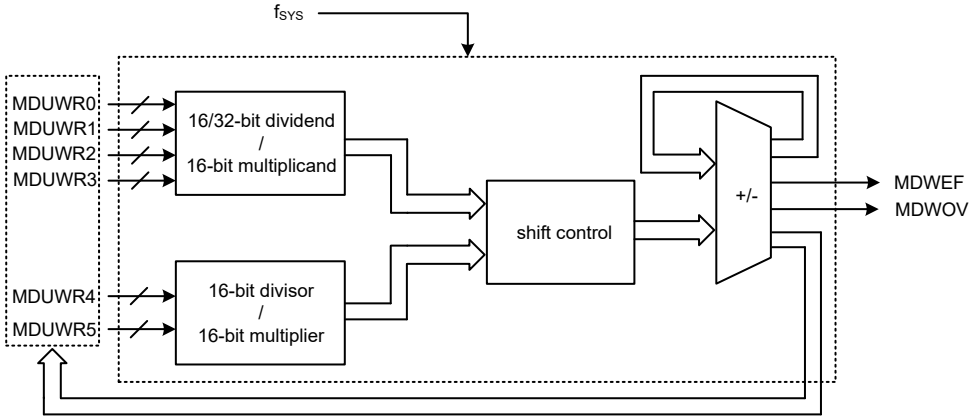
b. 当 ATMPF=1 时软件设定 ATMPC=1，硬件自动调变电路会在当前 PWM 周期结束时关闭，且硬件会将 ATMPC 清 0，软件需要自行将 ATMPF 清 0。

c. 在应用时，ATMPC 要在触发 ERSG 保护电路的信号消失后再设定为 1，否则硬件自动调变电路被关闭会造成应用问题。

步骤	ATMPC	ATMPF	工作状态
1	0	0	PWM 无调变。
2	1	0	PWM 无调变。
3	1	↑	PWM 无调变，并于当前 PWM 周期结束时将 ATMPC 清 0。
4	↓	1	PWM 无调变。
5	0	↓	PWM 无调变 。
6	0	1	PWM 依设定于每个周期调变一次。
7	1	0	于当前 PWM 周期结束时将 ATMPC 清 0，并且关闭硬件自动调变电路。

16 位乘除法单元 – MDU

此单片机内置 16 位乘除法单元，即 MDU，是 16 位无符号数乘法与 32 位 /16 位无符号数除法器。此乘除法器可取代软件乘除法的运算，节省大量运算时间、程序存储器和数据存储器空间，降低单片机负载，以达到提升整体系统性能。



16-bit MDU 方框图

MDU 寄存器

乘法和除法操作通过特定的步骤实现，即按照特定的顺序对一系列寄存器写值。状态寄存器 MDUWCTRL 用于指示运算操作的状态。六个数据寄存器每个寄存器用于存储不同 MDU 操作中的操作数。

寄存器名称	位							
	7	6	5	4	3	2	1	0
MDUWR0	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR1	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR2	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR3	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR4	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR5	D7	D6	D5	D4	D3	D2	D1	D0
MDUWCTRL	MDWEF	MDWOV	—	—	—	—	—	—

MDU 寄存器列表

- **MDUWRn 寄存器 (n=0~5)**

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**：16-bit MDU 数据寄存器 n

这些寄存器的使用取决于当前 MDU 的运算。

在 MDU 的相关计算未完成时，不能对 WDUWRn 寄存器进行读或写操作。否则会导致 MDU 运算结果异常并同时置位 MDU 错误标志位，MDWEF。

● **MDUWCTRL 寄存器**

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

Bit 7 **MDWEF**：16-bit MDU 错误标志位
0：正常
1：异常
如果在运算过程中 MDUWRn 寄存器被改写或被读取，MDWEF 位由硬件自动置位。当运算完成且 MDWEF 位为 1 时，可通过读取 MDUWCTRL 寄存器将此位清零。

Bit 6 **MDWOV**：16-bit MDU 溢出标志位
0：溢出未发生
1：乘法结果大于 FFFFH 或除数为 0
每完成一次运算，硬件自动更新此位以指示当前运算的情况。

Bit 5~0 未定义，读为“0”

乘除法单元操作

乘除法单元用于乘法运算还是除法运算取决于寄存器 MDUWR0~MDUWR5 的写入顺序。在写入被除数、被乘数、除数或乘数数据时，都是要先写低字节再写入高字节数据到对应的 MDU 数据寄存器中。在已按照正确顺序写入数据至 MDUWRn 寄存器，即完成最后一个 MDUWR5 寄存器写入后，开始执行相应的乘法和除法运算。写入顺序与 MDU 运算的对应关系如下：

- 32-bit/16-bit 除法运算：依序从 MDUWR0 写到 MDUWR5
- 16-bit/16-bit 除法运算：依序写入 MDUWR0、MDUWR1、MDUWR4 和 MDUWR5，跳过 MDUWR2 和 MDUWR3 不写
- 16-bit×16-bit 乘法运算：依序写入 MDUWR0、MDUWR4、MDUWR1 和 MDUWR5，跳过 MDUWR2 和 MDUWR3 不写

另外需要说明的是，在对 MDU 寄存器写入数据时必须要按照正确的顺序依次写入，但在时间上可以不需要连续，也就是说只要不影响写操作，在保证写入顺序正确的情况下，中间允许插入其它的非写入 MDUWRn 的指令或中断。

正确完成 MDU 寄存器写入后，MDU 开始执行对应的运算。不同的 MDU 运算所需的计算时间不同。在执行运算操作过程中，禁止对这六个 MDU 数据寄存器执行读或写动作。当每次运算操作完成后，应通过 MDUWCTRL 寄存器确认该运算是否正确。若运算正确，可按照特定顺序从对应的 MDU 数据寄存器中读取运算结果。MDU 运算及其所需的计算时间如下所示：

- 32-bit/16-bit 除法运算：$17 \times t_{SYS}$
- 16-bit/16-bit 除法运算：$9 \times t_{SYS}$
- 16-bit×16-bit 乘法运算：$11 \times t_{SYS}$

运算操作完成后，结果存储在对应的 MDU 数据寄存器中，必须按照指定的顺序读取这些寄存器，以得到准确的计算结果。运算结果读取顺序与 MDU 运算的对应关系如下：

- 32-bit/16-bit 除法运算：依序从 MDUWR0 到 MDUWR3 读取商数，依序从 MDURW4 和 MDUWR5 读取余数
- 16-bit/16-bit 除法运算：依序从 MDUWR0 和 MDUWR1 读取商数，依序从 MDUWR4 和 MDUWR5 读取余数

● 16-bit×16-bit 乘法运算：依序从 MDUWR0 到 MDUWR3 读取乘积

在读取结果数据时，必须按照正确的顺序读取，但在时间上可以不需要连续，也就是说只要不影响读操作，在保证读取顺序正确的情况下，中间允许插入其它非读取 MDUWRn 的指令或中断。

MDU 读 / 写顺序以及计算时间的注意事项总结如下表所示。

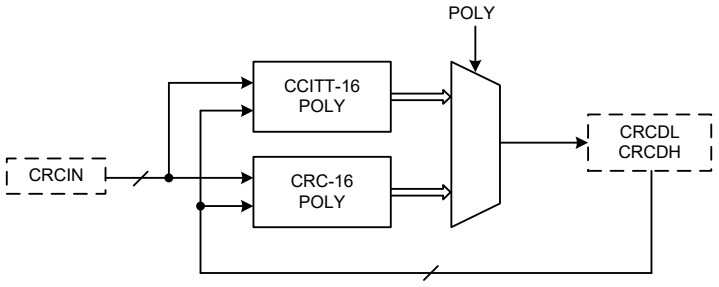
注意，在 MDU 操作未完成之前，单片机不可进入空闲或休眠模式，否则 MDU 操作失败。

运算 / 事项	32-bit / 16-bit 除法	16-bit / 16-bit 除法	16-bit × 16-bit 乘法
写顺序 最先写 ↓ ↓ ↓ ↓ 最后写	被除数字节 0 写入 MDUWR0 被除数字节 1 写入 MDUWR1 被除数字节 2 写入 MDUWR2 被除数字节 3 写入 MDUWR3 除数字节 0 写入 MDUWR4 除数字节 1 写入 MDUWR5	被除数字节 0 写入 MDUWR0 被除数字节 1 写入 MDUWR1 除数字节 0 写入 MDUWR4 除数字节 1 写入 MDUWR5	被乘数字节 0 写入 MDUWR0 乘数字节 0 写入 MDUWR4 被乘数字节 1 写入 MDUWR1 乘数字节 1 写入 MDUWR5
计算时间	17 × t_{SYS}	9 × t_{SYS}	11 × t_{SYS}
读顺序 最先读 ↓ ↓ ↓ ↓ 最后读	从 MDUWR0 读取商字节 0 从 MDUWR1 读取商字节 1 从 MDUWR2 读取商字节 2 从 MDUWR3 读取商字节 3 从 MDUWR4 读取余数字节 0 从 MDUWR5 读取余数字节 1	从 MDUWR0 读取商字节 0 从 MDUWR1 读取商字节 1 从 MDUWR4 读取余数字节 0 从 MDUWR5 读取余数字节 1	从 MDUWR0 读取乘积字节 0 从 MDUWR1 读取乘积字节 1 从 MDUWR2 读取乘积字节 2 从 MDUWR3 读取乘积字节 3

MDU 操作总结

循环冗余校验 – CRC

循环冗余校验 (CRC) 计算单元是一种错误检测技术测试算法，用于验证数据传输或存储数据的正确性。CRC 计算将数据流或数据块作为输入，并生成一个 16-bit 的输出余数。通常情况下，一个数据流带有 CRC 后缀码，当被发送或存储时该数据流可用作校验和。因此，被接收或重新存储的数据流都是通过上述相同的生成多项式计算得到的，详情见下方章节。



CRC 方框图

CRC 寄存器

CRC 发生器包含了一个 8-bit CRC 数据输入寄存器 CRCIN 和 CRC 校验和寄存器对 CRCDH 和 CRCDL。CRCIN 寄存器用于输入新数据，而 CRCDH 和 CRCDL 寄存器用于保持前一个 CRC 计算结果。CRCCR 控制寄存器用于选择使用哪一个 CRC 生成多项式。

寄存器名称	位							
	7	6	5	4	3	2	1	0
CRCCR	—	—	—	—	—	—	—	POLY
CRCIN	D7	D6	D5	D4	D3	D2	D1	D0
CRCDL	D7	D6	D5	D4	D3	D2	D1	D0
CRCDH	D15	D14	D13	D12	D11	D10	D9	D8

CRC 寄存器列表

- **CRCCR 寄存器**

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

Bit 7~1　未定义，读为“0”

Bit 0　**POLY**：16-bit CRC 多项式选择
0：CRC-CCITT：$X^{16} + X^{12} + X^5 + 1$
1：CRC-16：$X^{16} + X^{15} + X^2 + 1$

- **CRCIN 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**：CRC 输入数据寄存器

- **CRCDL 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**：16-bit CRC 校验和低字节数据寄存器

- **CRCDH 寄存器**

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

Bit 7~0　**D15~D8**：16-bit CRC 校验和高字节数据寄存器

CRC 操作

CRC 计算单元提供了基于 CRC16 和 CCITT CRC16 多项式的 16-bit CRC 计算结果。在该 CRC 发生器中，仅有两个多项式可用于数值计算，不支持其它生成多项式的 16-bit CRC 计算结果。

下方两个表达式可用于 CRC 生成多项式，通过 CRCCR 控制寄存器中的 POLY 位选择。CRC 计算结果称为 CRC 校验和 CRCSUM，并存储于 CRC 校验和寄存器对 CRCDH 和 CRCDL 中。

- CRC-CCITT：$X^{16} + X^{12} + X^5 + 1$
- CRC-16：$X^{16} + X^{15} + X^2 + 1$

CRC 计算

每次对 CRCIN 寄存器执行写操作，都会将存储在 CRCDH 和 CRCDL 寄存器中的前一个 CRC 校验值和新的输入数据进行计算。CRC 单元计算 CRC 数据寄存器值是按字节进行的。计算时间为一个 MCU 指令周期。

CRC 计算步骤：

1. 清除校验和寄存器 CRCDH 和 CRCDL。
2. 对 8-bit 输入数据字节和 16-bit CRCSUM 高字节进行异或操作，其结果称为临时 CRCSUM。
3. 将临时 CRCSUM 值左移一位，并向最低有效位 LSB 填入“0”。
4. 检查在步骤 3 中完成移位后的临时 CRCSUM 值。

   若 MSB 为“0”，则该移位后的临时 CRCSUM 将作为新的临时 CRCSUM。否则，对步骤 3 中移位后的临时 CRCSUM 和数据“8005H”进行异或操作。该操作结果将作为新的临时 CRCSUM。

   应注意的是对于 CRC-16 多项式，用于异或操作的数据为“8005H”，而对于 CRC-CCITT 多项式用于异或操作的数据则为“1021H”。
5. 重复步骤 3 到步骤 4，直到输入数据字节的所有位都经过计算。
6. 重复步骤 2 到步骤 5，直到所有输入数据字节都经过计算。此时，最新的计算结果则为最终的 CRC 校验和 CRCSUM。

CRC 计算范例

- 向 CRCIN 寄存器写入 1 个字节的输入数据，相应的 CRC 校验和将逐个被计算，如下表所示。

CRC 数据输入 / CRC 多项式	00H	01H	02H	03H	04H	05H	06H	07H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	0000H	1021H	2042H	3063H	4084H	50A5H	60C6H	70E7H
CRC-16 ($X^{16}+X^{15}+X^2+1$)	0000H	8005H	800FH	000AH	801BH	001EH	0014H	8011H

注：在每个 CRC 输入数据写入 CRCIN 寄存器之前，CRC 校验和寄存器对 CRCDH 和 CRCDL 的初始值为“0”。

- 向 CRCIN 寄存器连续写入 4 个字节的输入数据，相应的 CRC 检验和连续列于下表。

CRC 数据输入 / CRC 多项式	CRCIN = 78H→56H→34H→12H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	(CRCDH, CRCDL)=FF9FH→BBC3H→A367H→D0FAH
CRC-16 ($X^{16}+X^{15}+X^2+1$)	(CRCDH, CRCDL)=0110H→91F1H→F2DEH→5C43H

注：在连续的 CRC 数据输入操作之前，CRC 校验和寄存器对 CRCDH 和 CRCDL 的初始值为“0”。

程序存储器 CRC 校验和计算范例：

1. 清除校验和寄存器对 CRCDH 和 CRCDL。
2. 通过 CRCCR 寄存器中的 POLY 位选择 CRC-CCITT 或 CRC-16 多项式作为生成多项式。
3. 执行“TABRD [m]”指令，读取程序存储器数据值。
4. 将表格数据低字节写入 CRCIN 寄存器，并与当前 CRCSUM 值进行 CRC 计算。计算后将得到一个新的 CRCSUM 值并存储在 CRC 校验和寄存器对 CRCDH 和 CRCDL 中。
5. 将表格数据高字节写入 CRCIN 寄存器，并与当前 CRCSUM 值进行 CRC 计算。计算后将得到一个新的 CRCSUM 值并存储在 CRC 校验和寄存器对 CRCDH 和 CRCDL 中。
6. 重复步骤 3 到步骤 5，读取下一个程序存储器数据值并执行 CRC 计算，直到读取了所有的程序存储器数据，接着进行连续的 CRC 计算。计算后 CRC 校验和寄存器中的值为最终的 CRC 计算结果。

通用串行接口模块 – USIM

此单片机内有一个通用串行接口模块，包括三种易与外部设备通信的串行接口：四线 SPI、两线 I^2C 或两线 / 单线 UART 接口。这三种接口具有相当简单的通信协议，单片机可以通过这些接口与传感器、闪存或 EEPROM 内存等硬件设备通信。因为 USIM 接口引脚是与其它 I/O 引脚共用，因此在使用 USIM 功能前，要先通过相应的引脚共用功能选择寄存器选定 USIM 引脚功能。因为这三种接口共用引脚和寄存器，所以要先通过 SIMC0 寄存器中的 UART 模式选择位 UMD 和 SPI/I^2C 工作模式控制位 SIM2~SIM0 选择哪一种通信接口。若 USIM 功能使能，可通过上拉电阻控制寄存器选择与输入 / 输出口共用的 USIM 脚的上拉电阻。

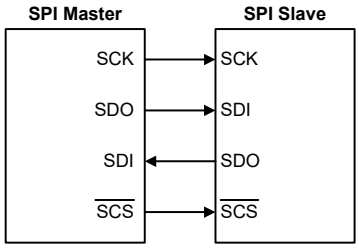
SPI 接口

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

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

SPI 接口操作

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

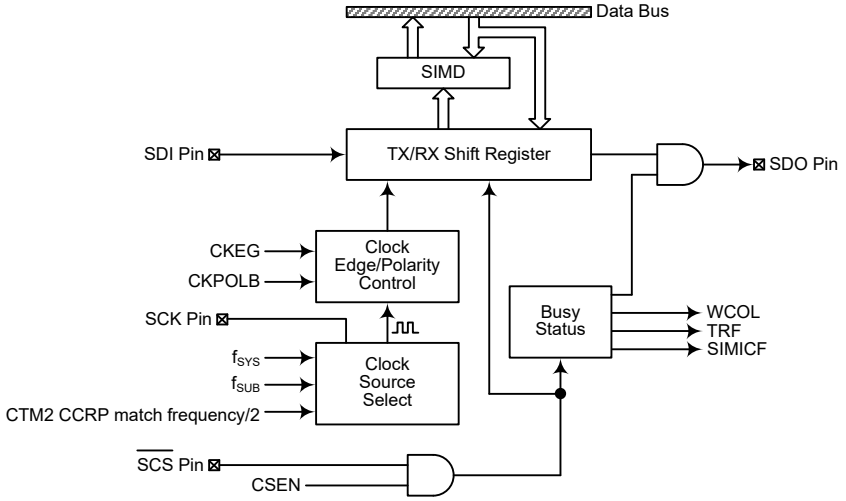


SPI 主 / 从机连接方式

该单片机的 SPI 功能具有以下特点：

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

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



SPI 方框图

SPI 寄存器

有三个内部寄存器用于控制 SPI 接口的所有操作，其中有一个数据寄存器 SIMD、两个控制寄存器 SIMC0 和 SIMC2。注意，只有在合理设置 SIMC0 寄存器中的 UMD 位和 SIM2~SIM0 位选择 SPI 模式后，SIMC2 和 SIMD 寄存器以及它们的上电复位值才有效。

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

SPI 寄存器列表

SPI 数据寄存器

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

- **SIMD 寄存器**

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

“x”：未知

Bit 7~0 **D7~D0**：USIM SPI/I²C 数据寄存器位 bit 7 ~ bit 0

SPI 控制寄存器

单片机中也有两个控制 SPI 接口功能的寄存器，SIMC0 和 SIMC2。寄存器 SIMC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 SIMC2 用于其它的控制功能如 LSB/MSB 选择，写冲突标志位等。

- **SIMC0 寄存器**

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

Bit 7~5 **SIM2~SIM0**：USIM SPI/I²C 工作模式控制位
000：SPI 主机模式；SPI 时钟为 $f_{SYS}/4$
001：SPI 主机模式；SPI 时钟为 $f_{SYS}/16$
010：SPI 主机模式；SPI 时钟为 $f_{SYS}/64$
011：SPI 主机模式；SPI 时钟为 f_{SUB}
100：SPI 主机模式；SPI 时钟为 CTM2 CCRP 匹配频率 / 2
101：SPI 从机模式
110：I²C 从机模式
111：未使用模式
当 UMD 位清零时，这几位用于设置 USIM SPI/I²C 功能的工作模式，除了选择 USIM 模块的 I²C 或 SPI 功能，还可选择 SPI 的主从模式和 SPI 的主机时钟频率。SPI 时钟源可来自于系统时钟和 f_{SUB} 也可以选择来自 CTM2。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 **UMD**：USIM UART 模式选择位
0：SPI 或 I²C 模式
1：UART 模式
此位为 UART 模式选择位。当此位清零时，选择 SPI 或 I²C 模式，而对 SPI 或 I²C 模式的选择是通过 SIM2~SIM0 位实现。

Bit 3~2 **SIMDEB1~SIMDEB0**：I²C 去抖时间选择位
这些位只有在 USIM 设置成 I²C 接口模式时才有效。请参考 I²C 寄存器部分。

Bit 1 **SIMEN**：USIM SPI/I²C 控制位
0：除能
1：使能
此位为 USIM SPI/I²C 接口的开 / 关控制位。此位为“0”时，USIM SPI/I²C 接口除能，SDI、SDO、SCK 和 $\overline{SCS}$ 或 SDA 和 SCL 脚将失去 SPI 或 I²C 功能，USIM 工作电流减小到最小值。此位为“1”时，USIM SPI/I²C 接口使能。若

USIM 经由 UMD 位和 SIM2~SIM0 位设置为工作在 SPI 接口，当 SIMEN 位由低到高转变时，SPI 控制寄存器中的设置不会发生变化，其首先应在应用程序中初始化。若 USIM 经由 UMD 位和 SIM2~SIM0 位设置为工作在 I²C 接口，当 SIMEN 位由低到高转变时，I²C 控制寄存器中的设置，如 HTX 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I²C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0 **SIMICF**：USIM SPI 未完成标志位
0：未发生
1：发生
此位仅当 USIM 配置在 SPI 从机模式时有效。如果 SPI 工作在从机模式且 SIMEN 和 CSEN 位都为“1”，但在 SPI 数据传输完全结束前 $\overline{SCS}$ 线被外部主机拉高，SIMICF 和 TRF 位都会被置高。在这种情况下，如果相应的中断功能使能将产生一个中断。然而，如果 SIMICF 位是由软件应用程序设为 1，那么 TRF 位将不会置高。

● SIMC2 寄存器

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

Bit 7~6 **D7~D6**：未定义位
用户可通过软件程序对这两位进行读写。

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

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

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

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

Bit 1 **WCOL**：SPI 写冲突标志位
0：无冲突
1：冲突

WCOL 标志位用于监测数据冲突的发生。此位为高时，表示在传输过程中有数据被写入 SIMD 寄存器。若数据正在被传输时，此写操作无效。此位可被应用程序清零。

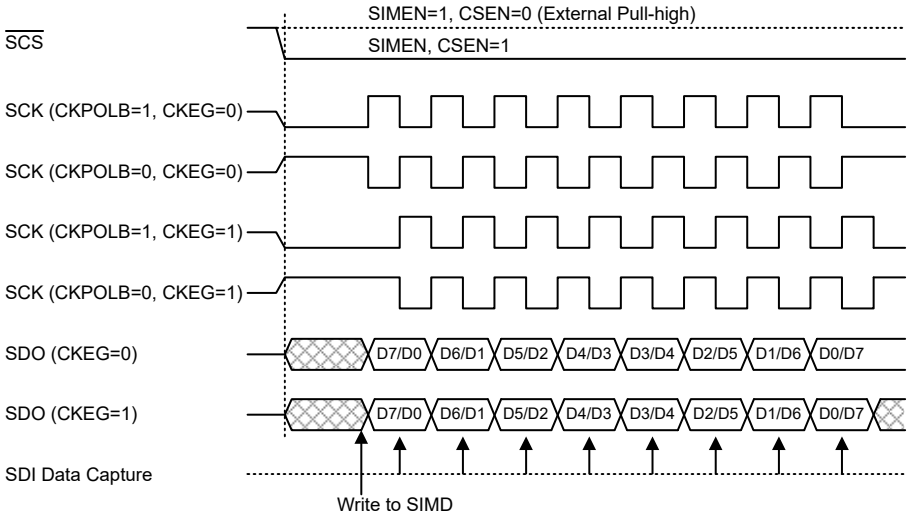
Bit 0 **TRF**：SPI 发送 / 接收结束标志位
0：数据正在发送
1：数据发送结束

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

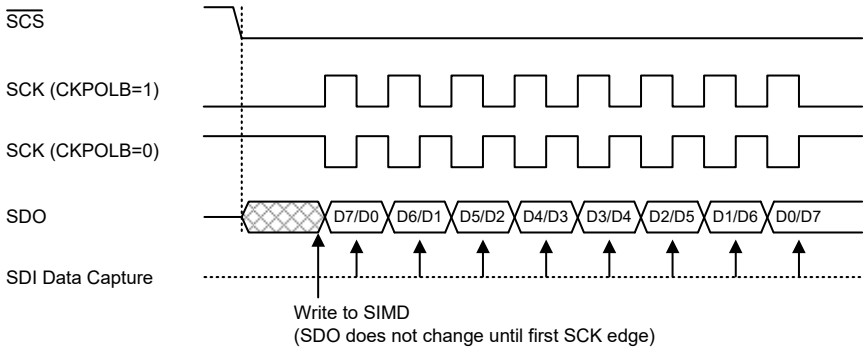
SPI 通信

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

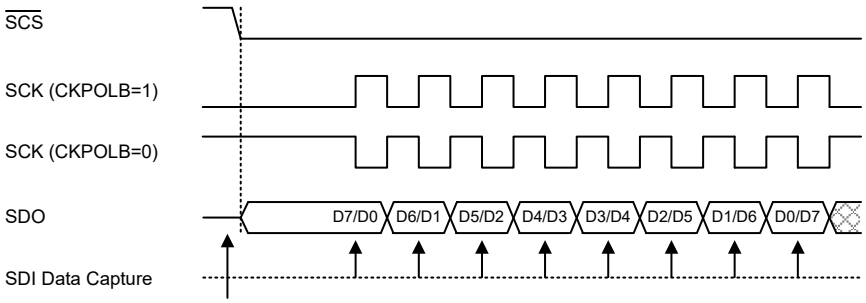
即使在单片机处于空闲模式时，若 SPI 接口使用的时钟源仍开启，SPI 功能仍将继续执行。



SPI 主机模式时序

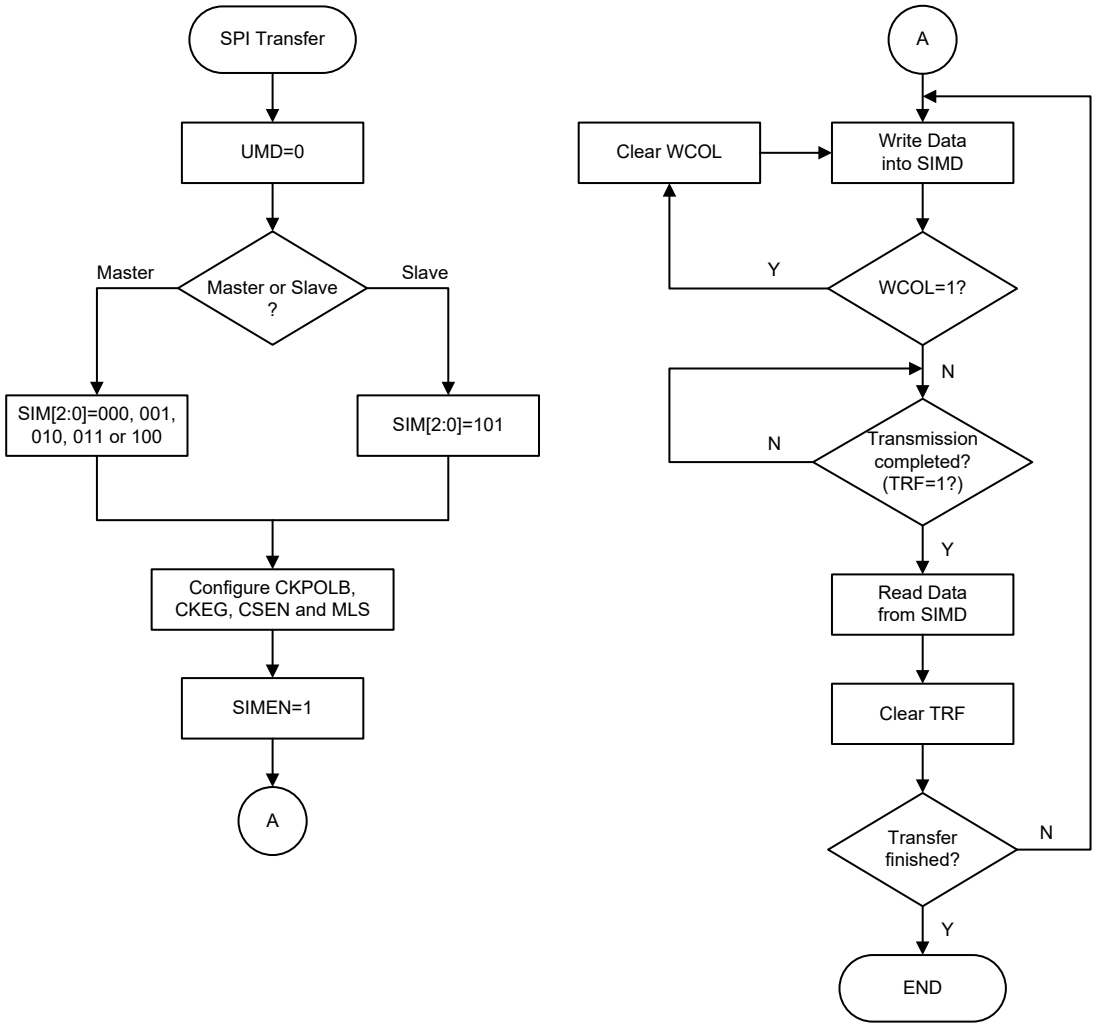


SPI 从机模式时序 – CKEG=0



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

SPI 从机模式时序 – CKEG=1



SPI 传输控制流程图

SPI 使能 / 除能

设置 CSEN=1、$\overline{SCS}$=0 将使能 SPI 总线，然后等待写数据到 SIMD 寄存器 (TXRX 缓存器)。单片机处于主机模式，数据写入 SIMD 寄存器后，自动开始数据传输或接收操作。数据传输完成时，TRF 位将自动被置位。单片机处于从机模式，SCK 引脚上收到脉冲信号之后，会传输 TXRX 中的数据，或将 SDI 引脚上的数据移入。

当 SPI 总线除能时，通过设置相应引脚共用控制位，SCK、SDI、SDO、$\overline{SCS}$ 可作为 I/O 口或其它功能引脚使用。

SPI 操作步骤

四线制 SPI 接口可完成所有主 / 从模式通信工作。

在 SIMC2 寄存器中，CSEN 位控制 SPI 接口的所有功能。设置此位为高，$\overline{SCS}$ 信号线有效将使能 SPI 接口。设置此位为低，SPI 接口将除能，$\overline{SCS}$ 信号线处于浮空状态因此不能控制 SPI 接口。CSEN 位和 SIMC0 寄存器中的 SIMEN 位设置为高，使得 SDI 信号线处于浮空状态且 SDO 信号线为高电平。主机模式中，如果 SCK 信号线为高还是低取决于 SIMC2 寄存器中的时钟极性选择位 CKPOLB。从机模式中，SCK 信号线处于浮空状态。如果 SIMEN 位设置为低，SPI 接口被除能，通过设置相应引脚共用控制位，$\overline{SCS}$、SDI、SDO 和 SCK 可作为 I/O 口或其它功能引脚使用。主机模式中，当数据被写入 SIMD 寄存器后，主机发起数据传输，并控制时钟信号。从机模式中，由外部主机发出数据传送 / 接收时钟信号。下面介绍主从模式中数据传输步骤。

主机模式：

- 步骤 1

  设置 SIMC0 控制寄存器中的 UMD 和 SIM2~SIM0 位，选择 SPI 主机模式和时钟源。

- 步骤 2

  设置 CSEN 和 MLS 位，选择高位或低位数据优先传送，这必须与从机设备一致。

- 步骤 3

  设置 SIMC0 控制寄存器中的 SIMEN 位，使能 SPI 接口功能。

- 步骤 4

  对于写操作：写数据到 SIMD 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。再使用 SCK 和 $\overline{SCS}$ 信号线将数据输出。接着执行步骤 5。

  对于读操作：从 SDI 信号线移入的数据将被存储在 TXRX 缓存器中，直到所有数据接收完毕，此时数据全部锁存至 SIMD 寄存器。

- 步骤 5

  检测 WCOL 位，若此位为高，则发生数据冲突并跳回至步骤 4；若为低，则继续执行下面的步骤。

- 步骤 6

  检测 TRF 位或等待 USIM SPI 串行总线中断发生。

- 步骤 7

  从 SIMD 寄存器中读数据。

- 步骤 8

  清除 TRF。

- 步骤 9

  跳回至步骤 4。

从机模式：

- 步骤 1

  设置 SIMC0 控制寄存器中的 UMD 和 SIM2~SIM0 位，选择 SPI 从机模式。

- 步骤 2

  设置 CSEN 和 MLS 位，选择高位或低位数据优先传送，这必须与主机设备一致。

- 步骤 3

  设置 SIMC0 控制寄存器中的 SIMEN 位，使能 SPI 接口功能。

- 步骤 4

  对于写操作：写数据到 SIMD 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。等待主机时钟 SCK 信号和 $\overline{SCS}$ 信号。跳至步骤 5。

  对于读操作：从 SDI 信号线移入的数据将被存储在 TXRX 缓存器中，直到所有数据接收完毕，此时数据全部锁存至 SIMD 寄存器。

- 步骤 5

  检测 WCOL 位，若此位为高，则发生数据冲突并跳回至步骤 4；若为低，则继续执行下面的步骤。

- 步骤 6

  检测 TRF 位或等待 USIM SPI 串行总线中断发生。

- 步骤 7

  从 SIMD 寄存器中读数据。

- 步骤 8

  清除 TRF。

- 步骤 9

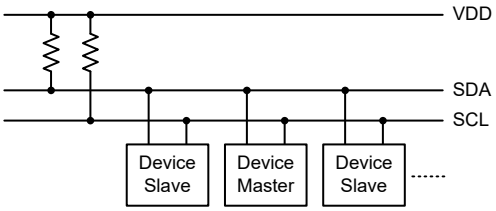
  跳回至步骤 4。

错误侦测

SIMC2 寄存器中的 WCOL 位用于数据传输期间监测数据冲突的发生。此位由 SPI 串行接口设置为高，而由应用程序来清除为零。在数据传输期间如果写数据到 SIMD，此位被置高提示数据冲突发生，并阻止数据继续被写入。

I^2C 接口

I^2C 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。I^2C 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。

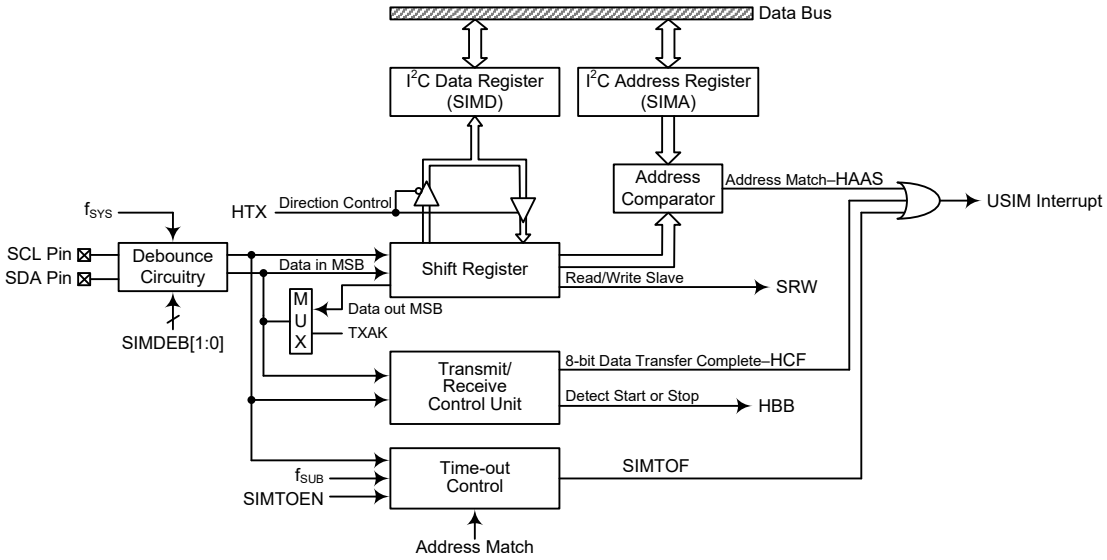


I²C 主从总线连接图

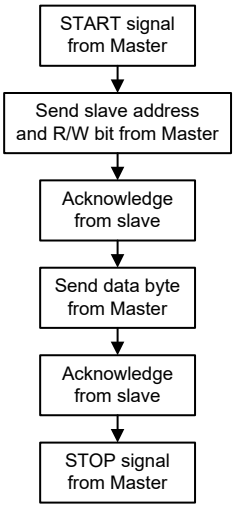
I²C 接口操作

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。即使 I²C 设备被激活，与 SCL/SDA 引脚共用的 I/O 口上拉电阻控制功能仍有效，其上拉电阻功能由相应的上拉电阻控制寄存器控制。



I²C 方框图



I²C 接口操作

SIMDEB1 和 SIMDEB0 位决定 I²C 接口的去抖时间。这个功能可以使用内部时钟在外部时钟上增加一个去抖间隔，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} > 2MHz	f_{SYS} > 4MHz
2 个系统时钟去抖时间	f_{SYS} > 4MHz	f_{SYS} > 8MHz
4 个系统时钟去抖时间	f_{SYS} > 4MHz	f_{SYS} > 8MHz

I²C 最小 f_{SYS} 频率要求

I²C 寄存器

I²C 总线有三个控制寄存器 SIMC0、SIMC1 和 SIMTOC，一个地址寄存器 SIMA 以及一个数据寄存器 SIMD。注意，只有在合理设置 SIMC0 寄存器中的 UMD 位和 SIM2~SIM0 位选择 I²C 模式后，SIMC1、SIMD、SIMA 和 SIMTOC 寄存器以及它们的上电复位值才有效。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMA	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C 寄存器列表

I²C 数据寄存器

SIMD 用于存储发送和接收的数据。这个寄存器由 SPI 和 I²C 功能所共用。在单片机将数据写入到 I²C 总线之前，要传输的数据应先存在 SIMD 中。I²C 总线接

收到数据之后，单片机就可以从 SIMD 数据寄存器中读取。所有通过 I²C 传输或接收的数据都必须通过 SIMD 实现。

- **SIMD 寄存器**

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

“x”：未知

Bit 7~0 **D7~D0**：USIM SPI/I²C 数据寄存器位 bit 7 ~ bit 0

I²C 地址寄存器

SIMA 寄存器也在 SPI 接口功能中使用，但其名称改为 SIMC2。SIMA 寄存器用于存放 7 位从机地址，寄存器 SIMA 中的 bit 7 ~ bit 1 是单片机的从机地址，bit 0 未定义。如果接至 I²C 的主机发送出的地址和寄存器 SIMA 中存储的地址相符，那么就选中了这个从机。

- **SIMA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~1 **SIMA6~SIMA0**：I²C 从机地址位
SIMA6~SIMA0 是从机地址 bit 6 ~ bit 0。

Bit 0 **D0**：保留位，此位可通过软件程序进行读写

I²C 控制寄存器

单片机中有三个控制 I²C 接口功能的寄存器，SIMC0、SIMC1 和 SIMTOC。寄存器 SIMC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 SIMC1 包括多个用于指示 I²C 传输状态的相关标志位。SIMTOC 寄存器用于控制 I²C 超时功能，此寄存器在 I²C 超时一节介绍。

- **SIMC0 寄存器**

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

Bit 7~5 **SIM2~SIM0**：USIM SPI/I²C 工作模式控制位
000：SPI 主机模式；SPI 时钟为 $f_{SYS}/4$
001：SPI 主机模式；SPI 时钟为 $f_{SYS}/16$
010：SPI 主机模式；SPI 时钟为 $f_{SYS}/64$
011：SPI 主机模式；SPI 时钟为 f_{SUB}
100：SPI 主机模式；SPI 时钟为 CTM2 CCRP 匹配频率 / 2
101：SPI 从机模式
110：I²C 从机模式
111：未使用模式
当 UMD 位清零时，这几位用于设置 USIM SPI/I²C 功能的工作模式，除了选择 USIM 模块的 I²C 或 SPI 功能，还可选择 SPI 的主从模式和 SPI 的主机时钟频率。

SPI 时钟源可来自于系统时钟和 f_{SUB} 也可以选择来自 CTM2。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 **UMD**：USIM UART 模式选择位
0：SPI 或 I²C 模式
1：UART 模式
此位为 UART 模式选择位。当此位清零时，选择 SPI 或 I²C 模式，而对 SPI 或 I²C 模式的选择是通过 SIM2~SIM0 位实现。

Bit 3~2 **SIMDEB1~SIMDEB0**：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
当设置 UMD 位为“0”、SIM2~SIM0 位为“110”将 USIM 设置为 I²C 接口功能时，这两个位用于选择 I²C 去抖时间。

Bit 1 **SIMEN**：USIM SPI/I²C 控制位
0：除能
1：使能
此位为 USIM SPI/I²C 接口的开 / 关控制位。此位为“0”时，USIM SPI/I²C 接口除能，SDI、SDO、SCK 和 $\overline{SCS}$ 或 SDA 和 SCL 脚将失去 SPI 或 I²C 功能，USIM 工作电流减小到最小值。此位为“1”时，USIM SPI/I²C 接口使能。若 USIM 经由 UMD 位和 SIM2~SIM0 位设置为工作在 SPI 接口，当 SIMEN 位由低到高转变时，SPI 控制寄存器中的设置不会发生变化，其首先应在应用程序中初始化。若 USIM 经由 UMD 位和 SIM2~SIM0 位设置为工作在 I²C 接口，当 SIMEN 位由低到高转变时，I²C 控制寄存器中的设置，如 HTX 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I²C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0 **SIMICF**：USIM SPI 未完成标志位
此位仅当 USIM 配置在 SPI 从机模式时有效。请参考 SPI 寄存器部分。

● SIMC1 寄存器

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

Bit 7 **HCF**：I²C 总线数据传输结束标志位
0：数据正在被传输
1：8 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的地址相同。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 总线空闲，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

从机接收完 8 位数据之后，该位将在第九个从机时钟时被传到总线上。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数据或接收来自 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，此时从机处于发送模式。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数据。

Bit 1 **IAMWU**：I²C 地址匹配唤醒控制位
0：除能
1：使能
此位设置为“1”则使能 I²C 地址匹配使系统从休眠或空闲模式中唤醒的功能。若进入休眠或空闲模式前 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 位为“0”，即表示 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即将有数据到达。数据的前 7 位是从机地址，高位在前，低位在后。如果发出的地址和从机地址匹配，SIMC1 寄存器的 HAAS 位会被置位，同时产生 USIM 中断。进入中断服务程序后，系统要检测 HAAS 位和 SIMTOF 位，以判断中断源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，要注意的是，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

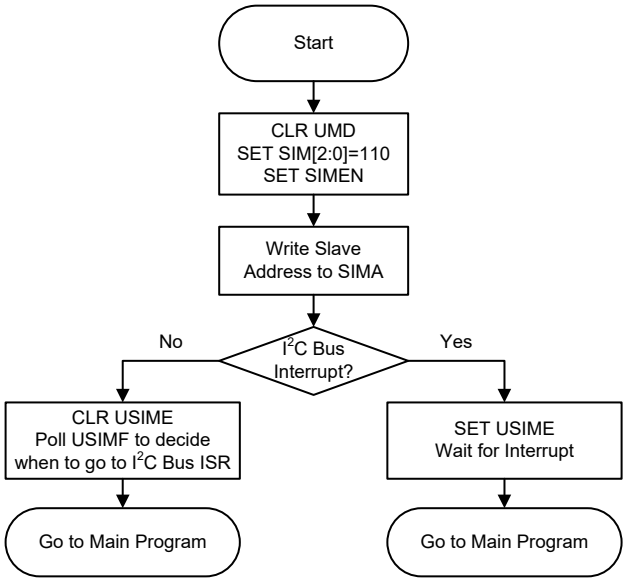
  设置 SIMC0 寄存器中 UMD 位为“0”、SIM2~SIM0 位为“110”和 SIMEN 位为“1”，以使能 I²C 总线。

- 步骤 2

  向 I²C 总线地址寄存器 SIMA 写入从机地址。

- 步骤 3

  设置中断控制寄存器中的 USIME 位以使能 USIM 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 总线的主机产生，而不是由从机产生。总线上的所有从机都可以侦测到起始信号。如果有从机侦测到起始信号，则表明 I²C 总线处于忙碌状态，并会置位 HBB。起始信号是指在 SCL 为高电平时，SDA 线上发生从高到低的电平变化。

I²C 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 USIM I²C 总线中断信号。地址位接下来的一位为读 / 写状态位 (即第 8 位)，将被保存到 SIMC1 寄存器的 SRW 位，从机随后发出一个低电平应答信号 (即第 9 位)。当从机地址匹配时，从机会将状态标志位 HAAS 置位。

USIM I²C 总线中断有三个中断源，当程序运行至中断服务子程序时，通过检测 HAAS 位和 SIMTOF 位，以判断 USIM 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 SIMD 寄存器，或是用于接收模式并从 SIMD 寄存器中读取空值以释放 SCL 线。

I²C 总线读 / 写信号

SIMC1 寄存器的 SRW 位用来表示主机是要从 I²C 总线上读取数据还是要将数据写到 I²C 总线上。从机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。

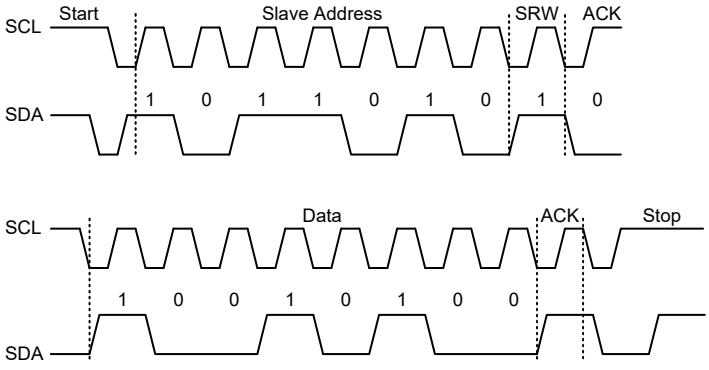
I²C 总线从机地址应答信号

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 SIMC1 寄存器的 HTX 位。如果 SRW 位为低，从机须设置成接收方，这样会清零 SIMC1 寄存器的 HTX 位。

I²C 总线数据和应答信号

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。如果从机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，此时主机方可发出 STOP 信号以释放 I²C 总线。所传送的数据存储在 SIMD 寄存器中。如果设置成发送方，从机必须先将欲传输的数据写到 SIMD 寄存器中；如果设置成接收方，从机必须从 SIMD 寄存器读取数据。

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

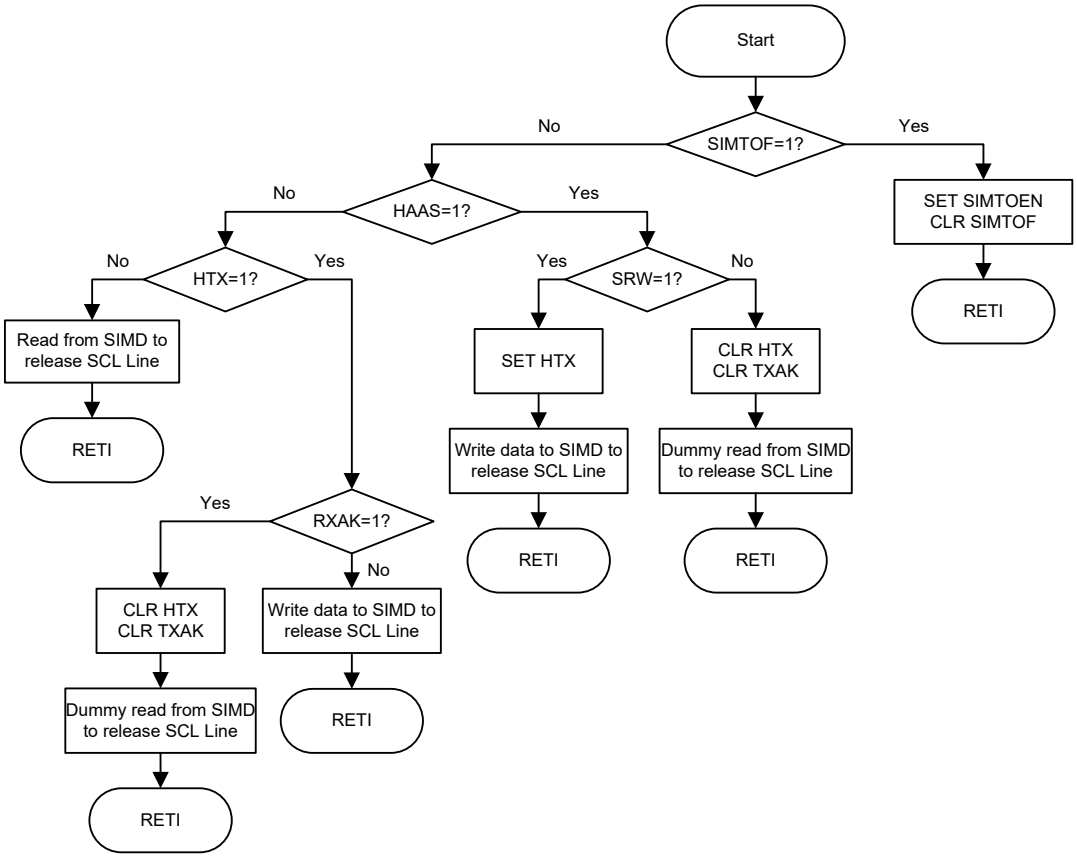


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

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

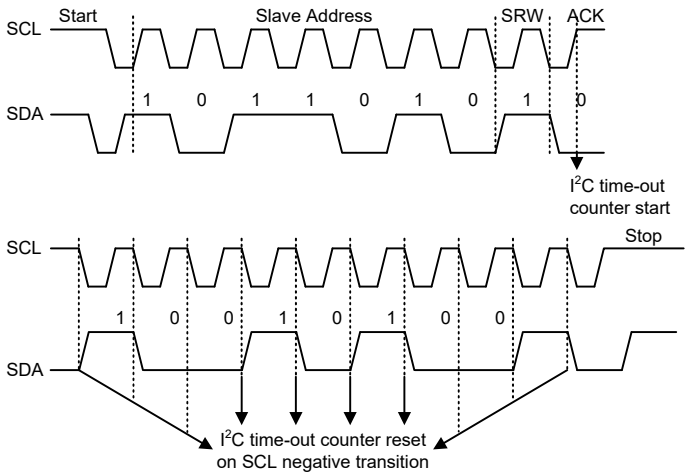
I²C 通信时序图



I²C 总线 ISR 流程图

I²C 超时控制

超时功能可减少 I²C 接收错误的时钟源而引起的锁死问题。如果连接到 I²C 总线的时钟源经过一段时间还未接收到，则在一定的超时周期后，I²C 电路和寄存器将复位。超时计数器在 I²C 总线“START”和“地址匹配”条件下开始计数，且在 SCL 下降沿清零。在下一个 SCL 下降沿到来之前，如果超时时间大于 SIMTOC 寄存器指定的超时周期，则超时发生。I²C“STOP”条件发生时超时功能终止。



I²C 超时时序图

当 I²C 超时计数器溢出时，计数器将停止计数，SIMTOEN 位被清零，且 SIMTOF 位被置高以表明超时计数器中断发生。超时计数器中断使用的也是 USIM 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
SIMD, SIMA, SIMC0	保持不变
SIMC1	复位至 POR

超时发生后的 I²C 寄存器

SIMTOF 标志位由应用程序清零。共有 64 个超时周期，可通过 SIMTOC 寄存器的 SIMTOS[5:0] 位进行选择。超时周期可通过公式计算：$((1\sim64)\times(32/f_{SUB}))$。由此可得超时周期范围为 1ms~64ms。

- **SIMTOC 寄存器**

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

Bit 7 **SIMTOEN**：USIM I²C 超时控制位
0：除能
1：使能

Bit 6 **SIMTOF**：USIM I²C 超时标志位
0：超时未发生
1：超时发生
当发生超时，此位由硬件自动置位；此位必须通过应用程序清零。

Bit 5~0 **SIMTOS5~SIMTOS0**：USIM I²C 超时时间选择位
I²C 超时时钟源是 $f_{SUB}/32$。
I²C 超时时间计算方法：$(SIMTOS[5:0]+1)\times(32/f_{SUB})$。

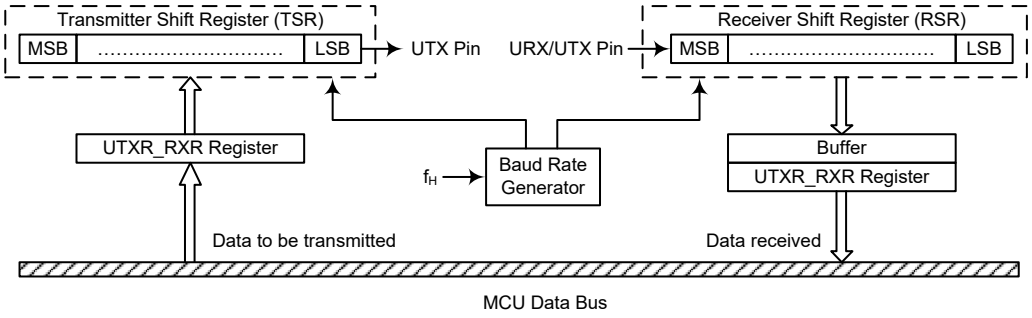
UART 接口

该单片机具有一个全双工或半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

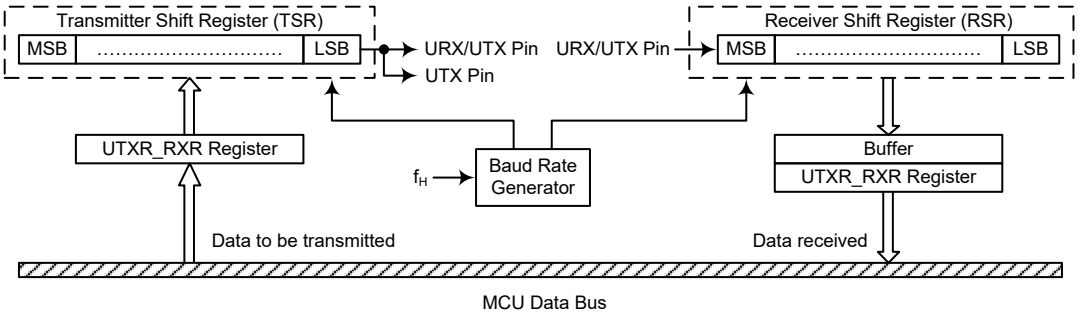
数据时，将数据组成一个 8 位或 9 位的数据块，连同数据特征位一并传输。具有检测数据覆盖或帧错误等功能。UART 功能与 SPI 和 I^2C 接口共用一个内部中断向量，当接收到数据或数据发送结束，触发中断。

内置的 UART 功能包含以下特性：

- 全双工或半双工 (单线通信模式) 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
- URX/UTX 引脚唤醒功能
- 发送和接收中断
- 中断可由下列条件触发：
  - ♦ 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 – USWM=0



UART 数据传输方框图 – USWM=1

UART 外部引脚

内部 UART 有两个外部引脚 UTX 和 URX/UTX，可与外部串行接口进行通信。UTX 和 URX/UTX 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 UTX 和 URX/UTX 引脚功能。当 UMD、UREN、UTXEN 和 URXEN 位置高时，将自动设置这些 I/O 脚或其它共用功能脚作为发送输出和接收输入。此时，用作发送输出的引脚其内部上拉电阻会被除能，而用作接收输入的引脚其内部上拉电阻由相应的上拉电阻控制位控制。当 UMD、UREN、UTXEN 或 URXEN 位清零除能 UTX 或 URX/UTX 引脚功能后，UTX 或 URX/UTX 引脚将处于浮空状态。这时 UTX 或 URX/UTX 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 单线模式

UART 功能支持单线模式通信，通过 UUCR3 寄存器中的 USWM 位选择。当设置该位为高，UART 将工作在单线模式。在单线模式下，单个 URX/UTX 引脚通过相关控制位的不同设置即可完成数据的发送与接收。设置 URXEN 位为高，URX/UTX 引脚用作接收引脚。将 URXEN 位清零，同时设置 UTXEN 位为高，URX/UTX 引脚用作发送引脚。

在单线模式下建议不要将 URXEN 位和 UTXEN 位同时设置为高。若 URXEN 位和 UTXEN 位同时为高，URXEN 位具有更高的优先级，此时 UART 为接收器状态。

需特别注意的是，UART 章节所有内容是基于 UART 全双工通信来对 UART 功能进行描述，相关的说明除引脚的使用外，对半双工通信 (单线模式) 同样适用。在理解单线模式通信时，全双工通信中使用的 UTX 引脚需取代为 URX/UTX 引脚。

在单线模式下，通过合理的软件配置，数据也可以在 UTX 引脚发送。因此数据可通过 URX/UTX 和 UTX 引脚输出。

UART 数据传输方案

UART 数据传输方框图显示了 UART 的整体结构。需要发送的数据首先写入 UTXR_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 UTX 引脚上，低位在前。UTXR_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 URX/UTX 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 UTXR_RXR 寄存器中。UTXR_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 UTXR_RXR 寄存器。

UART 状态和控制寄存器

与 UART 功能相关的有七个寄存器，SIMC0 寄存器中的 UMD 位用于选择 UART 接口功能。UUCR3 寄存器中的 USWM 位用于使能 / 除能 UART 单线模式。其它包括控制 UART 模块整体功能的 UUSR、UUCR1 和 UUCR2 寄存器，控制波特率的 UBRG 寄存器，管理发送和接收数据的数据寄存器 UTXR_RXR。注意，只有在 SIMC0 寄存器中的 UMD 位设置为“1”后，UART 相关的寄存器以及它们的上电复位值才有效。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM2	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
UUSR	UPERR	UNF	UFERR	UOERR	URIDLE	URXIF	UTIDLE	UTXIF
UUCR1	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
UUCR2	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
UUCR3	—	—	—	—	—	—	—	USWM
UTXR_RXR	UTXRX7	UTXRX6	UTXRX5	UTXRX4	UTXRX3	UTXRX2	UTXRX1	UTXRX0
UBRG	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0

UART 寄存器列表

- **SIMC0 寄存器**

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

Bit 7~5 **SIM2~SIM0**：USIM SPI/I²C 工作模式控制位
当 UMD 位清零时，这几位用于设置 USIM SPI/I²C 功能的工作模式。更多细节详见 SPI 或 I²C 寄存器章节。

Bit 4 **UMD**：UART 模式选择位
0：SPI 或 I²C 模式
1：UART 模式
此位为 UART 模式选择位。当此位清零时，选择 SPI 或 I²C 模式，而对 SPI 或 I²C 模式的选择是通过 SIM2~SIM0 位实现。

Bit 3~2 **SIMDEB1~SIMDEB0**：I²C 去抖时间选择位
详见 I²C 寄存器章节。

Bit 1 **SIMEN**：USIM SPI/I²C 控制位
0：除能
1：使能
此位仅当 UMD 位设置为“0”选择 SPI 或 I²C 模式时才有效。详见 SPI 或 I²C 寄存器章节。

Bit 0 **SIMICF**：USIM SPI 未完成标志位
详见 SPI 寄存器章节。

- **UUSR 寄存器**

寄存器 UUSR 是 UART 的状态寄存器，可以通过程序读取以得知当前 UART 状态。所有 UUSR 位是只读的。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	UPERR	UNF	UFERR	UOERR	URIDLE	URXIF	UTIDLE	UTXIF
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

Bit 7 **UPERR**：奇偶校验出错标志位
0：奇偶校验正确
1：奇偶校验出错
UPERR 是奇偶校验出错标志位。若 UPERR=0，奇偶校验正确；若 UPERR=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 UUSR 寄存器再读 UTXR_RXR 寄存器来清除此位。

Bit 6 **UNF**：噪声干扰标志位
0：没有受到噪声干扰
1：受到噪声干扰
UNF 是噪声干扰标志位。若 UNF=0，没有受到噪声干扰；若 UNF=1，UART 接收数据时受到噪声干扰。它与 URXIF 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 UUSR 寄存器再读 UTXR_RXR 寄存器将清除此标志位。

Bit 5 **UFERR**：帧错误标志位
0：无帧错误发生
1：有帧错误发生
UFERR 是帧错误标志位。若 UFERR=0，没有帧错误发生；若 UFERR=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 UUSR 寄存器再读 UTXR_RXR 寄存器来清除此位。

Bit 4 **UOERR**：溢出错误标志位
0：无溢出错误发生
1：有溢出错误发生
UOERR 是溢出错误标志位，表示接收缓冲器是否溢出。若 UOERR=0，没有溢出错误；若 UOERR=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 UUSR 寄存器再读 UTXR_RXR 寄存器将清除此标志位。

Bit 3 **URIDLE**：接收状态标志位
0：正在接收数据
1：接收器空闲
URIDLE 是接收状态标志位。若 URIDLE=0，正在接收数据；若 URIDLE=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，URIDLE 被置位，表明 UART 空闲，URX/UTX 引脚处于逻辑高状态。

Bit 2 **URXIF**：接收寄存器状态标志位
0：UTXR_RXR 寄存器为空
1：UTXR_RXR 寄存器含有有效数据
URXIF 是接收寄存器状态标志位。当 URXIF=0，UTXR_RXR 寄存器为空；当 URXIF=1，UTXR_RXR 寄存器接收到新数据。当数据从移位寄存器加载到 UTXR_RXR 寄存器中，如果 UUCR2 寄存器中的 URIE=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 UNF、UFERR 或 UPERR 会在同一周期内置位。读取 UUSR 寄存器再读 UTXR_RXR 寄存器，如果 UTXR_RXR 寄存器中没有新的数据，那么将清除 URXIF 标志。

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

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

● **UUCR1 寄存器**

UUCR1、UUCR2 和 UUCR3 是 UART 的三个控制寄存器，用来定义各种 UART 功能，例如 UART 的使能与除能、奇偶校验控制、传输数据的长度以及单线模式通信等等。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
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 **UREN**：UART 功能使能位
0：UART 除能，UTX 和 URX/UTX 引脚处于浮空状态
1：UART 使能，UTX 和 URX/UTX 引脚作为 UART 功能引脚
此位为 UART 的使能位。UREN=0，UART 除能，URX/UTX 和 UTX 处于浮空状态；UREN=1，若 UMD 位置高，UART 使能，UTX 和 URX/UTX 将分别由 USWM 模式选择位、UTXEN 和 URXEN 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，UTXEN、URXEN、UTXBRK、URXIF、UOERR、UFERR、UPERR 和 UNF 清零，而 UTIDLE、UTXIF 和 URIDLE 置位，UUCR1、UUCR2、UUCR3 和 UBRG 寄存器中的其它位保持不变。若 UART 工作时 UREN 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 **UBNO**：发送数据位数选择位
0：8-bit 传输数据
1：9-bit 传输数据
UBNO 是发送数据位数选择位。UBNO=1，传输数据为 9 位；UBNO=0，传输数据为 8 位。若选择了 9 位数据传输格式，URX8 和 UTX8 将分别存储接收和发送数据的第 9 位。

Bit 5 **UPREN**：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。UPREN=1，使能奇偶校验；UPREN=0，除能奇偶校验。

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

Bit 3 **USTOPS**：发送器停止位的长度选择位
0：有一位停止位
1：有两位停止位
此位用来设置发送器停止位的长度。USTOP=1，有两位停止位；USTOP=0，只有一位停止位。

Bit 2 **UTXBRK**：暂停字发送控制位
0：没有暂停字要发送
1：发送暂停字
UTXBRK 是暂停字发送控制位。UTXBRK=0，没有暂停字要发送，UTX 引脚正常操作；UTXBRK=1，将会发送暂停字，发送器将发送逻辑“0”。若 UTXBRK 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 UTXBRK 复位。

Bit 1 **URX8**：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。UBNO 是用来控制传输位数是 8 位还是 9 位。

Bit 0 **UTX8**：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。UBNO 是用来控制传输位数是 8 位还是 9 位。

- **UUCR2 寄存器**

UUCR2 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 USIM UART 模式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	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
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 **UTXEN**：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。UTXEN=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 UTX 引脚将处于浮空状态。若 UTXEN=1 且 UMD=1 及 UREN=1，则发送将被使能，UTX 引脚将由 UART 来控制。在数据传输时清除 UTXEN 将中止数据发送且复位发送器，此时 UTX 引脚将处于浮空状态。

Bit 6 **URXEN**：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。URXEN=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 URX/UTX 引脚将处于浮空状态。若 URXEN=1 且 UMD=1 及 UREN=1，则接收将被使能，URX/UTX 引脚将由 UART 来控制。在数据传输时清除 URXEN 将中止数据接收且复位接收器，此时 URX/UTX 引脚将处于浮空状态。

Bit 5 **UBRGH**：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 UBRG 寄存器一起控制 UART 的波特率。UBRGH=1，为高速模式；UBRGH=0，为低速模式。

Bit 4 **UADDEN**：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。UADDEN=1，地址检测使能，此时数据的第 8 位 (UBNO=0) 或第 9 位 (UBNO=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 **UWAKE**：URX/UTX 引脚下降沿唤醒 UART 功能使能位
0：URX/UTX 引脚下降沿唤醒 UART 功能除能
1：URX/UTX 引脚下降沿唤醒 UART 功能使能
此位用于控制 URX/UTX 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 URX/UTX 引脚唤醒 UART 功能无效。若此位置高且 UART 时钟 f_H 关闭，当 URX/UTX 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 URX/UTX 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 URX/UTX 引脚发生下降沿也无法恢复 UART 功能。

Bit 2 **URIE**：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 URIE=1，当 UOERR 或 URXIF 置位时，USIM 的中断请求标志 USIMF 置位；若 URIE=0，USIM 中断请求标志 USIMF 不受 UOERR 和 URXIF 影响。

Bit 1 **UTIIE**：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 UTIIE=1，当发送器空闲触发 UTIDLE 置位时，USIM 的中断请求标志 USIMF 置位；若 UTIIE=0，USIM 中断请求标志 USIMF 不受 UTIDLE 的影响。

Bit 0 **UTEIE**：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 UTEIE=1，当发送器为空触发 UTXIF 置位时，USIM 的中断请求标志 USIMF 置位；若 UTEIE=0，USIM 中断请求标志 USIMF 不受 UTXIF 的影响。

- **UUCR3 寄存器**

UUCR3 寄存器用于使能 UART 单线模式通信。顾名思义，在单线模式下 UART 只需要使用一条线，URX/UTX，在 UUCR2 寄存器中的 URXEN 和 UTXEN 位控制下即可完成通信。

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

Bit 7~1 未定义，读为“0”

Bit 0 **USWM**：单线模式使能控制
0：除能，URX/UTX 引脚仅用作 UART 接收功能
1：使能，URX/UTX 引脚在 URXEN 和 UTXEN 位控制下可用作接收或发送功能
需注意的是，单线模式使能时，若将 URXEN 和 UTXEN 位同时设置为高，URX/UTX 引脚用作接收功能。

- **UTXR_RXR 寄存器**

UTXR_RXR 是一个数据寄存器，用来存储 UTX 引脚将要发送或 URX/UTX 引脚正在接收的数据。

Bit	7	6	5	4	3	2	1	0
Name	UTXRX7	UTXRX6	UTXRX5	UTXRX4	UTXRX3	UTXRX2	UTXRX1	UTXRX0
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 **UTXRX7~UTXRX0**：UART 发送 / 接收数据位 Bit 7~Bit 0

- **UBRG 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0
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 **UBRG7~UBRG0**：波特率值
软件设置 UUCR2 寄存器中的 UBRGH 位 (设置波特率发生器的速度) 和 UBRG 寄存器 (设置波特率的值)，一起控制 UART 的波特率。
注：若 UBRGH=0，波特率 = $f_H/[64\times(N+1)]$；
若 UBRGH=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 UBRG 寄存器和 UUCR2 寄存器的 UBRGH 位来控制。UBRGH 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。UBRG 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。

UUCR2 的 UBRGH 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 UBRGH 来选择相应的计算公式从而算出 UBRG 的值。由于 UBRG 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 UBRG 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 UBRGH=0，若期望的波特率为 4800，计算它的 UBRG 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 UBRG 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 UUCR1 寄存器的 UBNO、UPRT、UPREN 和 USTOPS 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 UUCR1 寄存器的 UREN 位来使能和除能的。当 SIMC0 寄存器中的 UMD 位已设置为“1”选择 UART 模式，若 UREN、UTXEN 和 URXEN 都为高，则 TX 和 URX/UTX 分别为 UART 的发送端口和接收端口。若没有数据发送，UTX 引脚默认状态为高电平。

UREN 清零将除能 UTX 和 URX/UTX，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 UTXEN、URXEN、UTXBRK、URXIF、UOERR、UFERR、UPERR 和 UNF 清零，而 UTIDLE、UTXIF 和 URIDLE 置位，UUCR1、UUCR2、UUCR3 和 UBRG 寄存器中的其它位保持不变。若 UART 工作时 UREN 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。

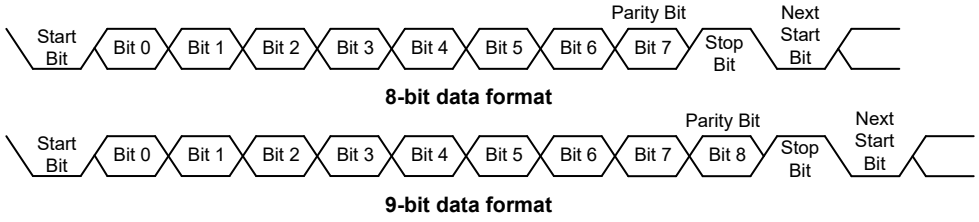
数据位、停止位位数以及奇偶校验的选择

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 UUCR1 寄存器的各个位控制的。UBNO 决定数据传输是 8 位还是 9 位；UPRT 决定校验类型；UPREN 决定是否选择奇偶校验；而 USTOPS 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

UUCR1 寄存器的 UBNO 位是控制数据传输的长度。UBNO=1 其长度为 9 位，第 9 位 MSB 存储在 UUCR1 寄存器的 UTX8 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 UTXR_RXR 提供，应用程序只须将发送数据写入 UTXR_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 UTXR_RXR 寄存器加载到 TSR 寄存器。TSR 不像其它寄存器一样映像到数据存储器，所以应用程序不能对其进行读写操作。UTXEN=1，发送使能，但若 UTXR_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 UTXR_RXR 寄存器再置高 UTXEN 也会触发发送。当发送器使能，若 TSR 寄存器为空，数据写入 UTXR_RXR 寄存器将会直接加载到 TSR 寄存器中。发送器工作时，UTXEN 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，UTX 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 UTX 引脚上，其低位在前高位在后。在发送模式中，UTXR_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 UUCR1 寄存器的 UTX8。

启动数据发送的步骤如下：

- 正确地设置 UBNO、UPRT、UPREN 和 USTOPS 位以确定数据长度、校验类型和停止位长度。
- 设置 UBRG 寄存器，选择期望的波特率。
- 置高 UTXEN，使能 UART 发送器且使 UTX 作为 UART 的发送端。
- 读取 UUSR 寄存器，然后将待发数据写入 UTXR_RXR 寄存器。注意，此步骤会清除 UTXIF 标志位。

如果要发送多个数据只需重复上一步骤。

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

1. 读取 UUSR 寄存器
2. 写 UTXR_RXR 寄存器

只读标志位 UTXIF 由 UART 硬件置位。若 UTXIF=1，UTXR_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 UTEIE=1，UTXIF 标志位会产生中断。在数据传输时，写 UTXR_RXR 指令会将待发数据暂存在 UTXR_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 UTXR_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 UTXIF 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 UTIDLE 位将被置位。

可以通过以下步骤来清除 UTIDLE：

1. 读取 UUSR 寄存器
2. 写 UTXR_RXR 寄存器

清除 UTXIF 和 UTIDLE 软件执行次序相同。

发送暂停字

若 UTXBRK=1 保持时间超过 $[(UBRG+1)\times t_H]$ 且 UTIDLE=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 UTXBRK 将会发送暂停字，而清除 UTXBRK 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 UTXBRK 持续为高，那么发送器会一直发送暂停字；当应用程序将 UTXBRK 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 UBNO=1，数据长度为 9 位，而最高位 MSB 存放在 UUCR1 寄存器的 URX8 中。接收器的核心是串行移位寄存器 RSR。URX/UTX 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 URX/UTX 引脚上检测到停止位，若 UTXR_RXR 寄存器为空，数据从 RSR 寄存器中加载到 UTXR_RXR 寄存器。URX/UTX 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像在数据存储器，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 URX/UTX 引脚进入移位寄存器。UTXR_RXR 寄存器在内部总线和接收移位寄存器间形成一个缓冲。UTXR_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 UTXR_RXR 寄存器，否则忽略第三帧数据并且发生溢出错误。

启动数据接收的步骤如下：

- 正确地设置 UBNO、UPRT 和 UPREN 位以确定数据长度和校验类型。
- 设置 UBRG 寄存器，选择期望的波特率。
- 置高 URXEN，使能 UART 接收器且使 URX/UTX 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 UTXR_RXR 寄存器中包含有效数据时，UUSR 寄存器中的 URXIF 位将会置位，溢出错误发生之前至多还有一帧数据可读。
- 若 URIE=1，数据从 RSR 寄存器加载到 UTXR_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 URXIF：

1. 读取 UUSR 寄存器
2. 读取 UTXR_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 UBNO 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 UBNO 位指定的长度外加一个停止位，接收器认为接收已完毕，URXIF 和 UFERR 置位，UTXR_RXR 寄存器清 0，若相应的中断允许且 URIDLE 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 UFERR 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 UFERR 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 URIDLE。

UART 接收到暂停字会产生以下事件：

- 帧错误标志位 UFERR 置位。
- UTXR_RXR 寄存器清零。
- UOERR、UNF、UPERR、URIDLE 或 URXIF 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，UUSR 寄存器的接收状态标志位 URIDLE 清零。在停止位和下一帧数据的起始位之间，URIDLE 被置位，表示接收器空闲。

接收中断

UUSR 寄存器的只读标志位 URXIF 由接收器的边沿触发置位。若 URIE=1，数据从移位寄存器 RSR 加载到 UTXR_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 – UOERR 标志

UTXR_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 UTXR_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：

- UUSR 寄存器中 UOERR 被置位。
- UTXR_RXR 寄存器中数据不会丢失。
- RSR 寄存器数据将会被覆盖。
- 若 URIE=1，将会产生中断。

先读取 UUSR 寄存器再读取 UTXR_RXR 寄存器可将 UOERR 清零。

噪声干扰 – UNF 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 URXIF 上升沿，UUSR 寄存器中只读标志位 UNF 置位。
- 数据从 RSR 寄存器加载到 UTXR_RXR 寄存器中。
- 不产生中断，但此位置位发生在 URXIF 置位产生中断的同周期内。

先读取 UUSR 寄存器再读取 UTXR_RXR 寄存器可将 UNF 清零。

帧错误 – UFERR 标志

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

奇偶校验错误 – UPERR 标志

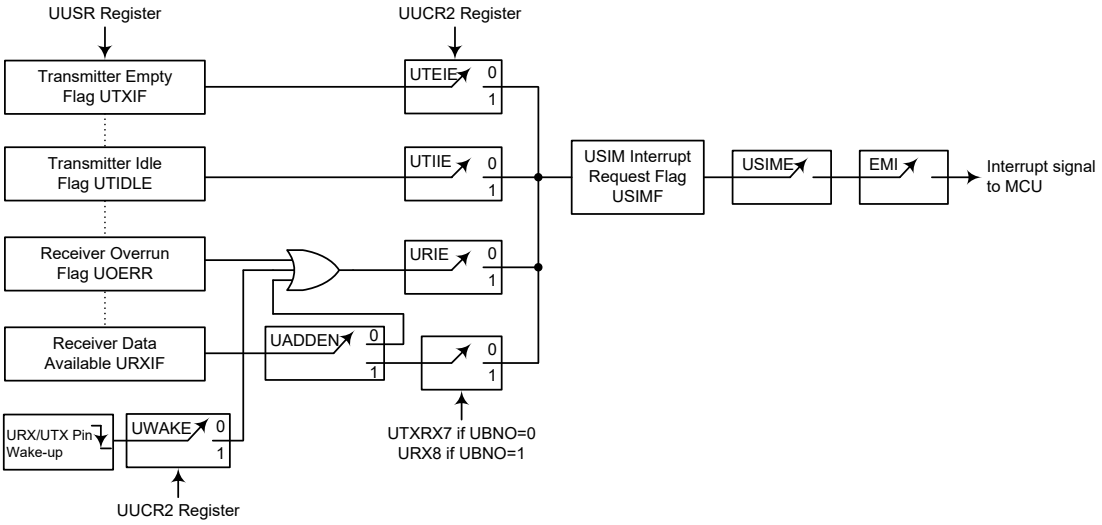
若接收到数据出现奇偶校验错误，UUSR 寄存器中只读标志 UPERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。此标志位同接收的数据分别记录在 UUSR 寄存器和 UTXR_RXR 寄存器中，此标志位可被任何复位清零。注意，在读取相应的数据之前必须先访问 UUSR 寄存器中的 UFERR 和 UPERR 错误标志位。

UART 模块中断结构

几个独立的 UART 条件可以产生一个 USIM 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 URX/UTX 引脚唤醒都会产生中断。若总中断使能、USIM 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 UUCR2 寄存器中相应中断允许位被置位，则 UUSR 寄存器中对应中断标志位将产生 USIM 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 USIM UART 模式中断源。

地址检测也是 USIM UART 模式的中断源，它没有相应的标志位，若 UUCR2 寄存器中 UADDEN=1，当检测到地址将会产生 USIM 中断。URX/UTX 引脚唤醒也可以产生 USIM 中断，它没有相应的标志位，当 UART 时钟源 f_H 关闭且 UUCR2 中的 UWAKE 和 URIE 位被置位，URX/UTX 引脚上有下降沿时会产生 USIM 中断。

注意，UUSR 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 USIM 中断控制寄存器中的相关中断使能控制位控制，以决定是否响应或屏蔽 UART 模块的中断请求。



UART 中断框图

地址检测模式

置位 UUCR2 寄存器中的 UADDEN 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 URXIF。若 UADDEN 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 USIME 和 EMI 也要使能才会产生中断。地址的最高位为第 9 位 (UBNO=1) 或第 8 位 (UBNO=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 UADDEN 除能，每接收到一个有效数据便会置位 URXIF，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。

UADDEN	9th Bit (UBNO=1) 8th Bit (UBNO=0)	产生 USIM 中断
0	0	√
	1	√
1	0	×
	1	√

UADDEN 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，UUSR、UUCR1、UUCR2、UUCR3、UTXR_RXR 以及 UBRG 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 URX/UTX 引脚的唤醒功能，由 UUCR2 寄存器中 UWAKE 位控制。当 UART 时钟 f_H 关闭时，若 UWAKE 位与 UART 模式选择位 UMD、UART 允许位 UREN、接收器使能位 URXEN 和接收器中断使能位 URIE 都被置位，则 URX/UTX 引脚的下降沿可触发产生 URX/UTX 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，URX/UTX 引脚上的任何数据将被忽略。

若要产生唤醒 UART 的中断，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 USIM 中断使能控制位 USIME 也必须置位；若这两控制位没有被置位，那么，可发生唤醒事件但不会产生中断。唤醒后系统需一定的延时才能正常工作，然后才会产生 USIM 中断。

低电压检测 – LVD

该单片机具有低电压检测功能，即 LVD。通过使能该功能可对电源电压 V_{DD} 或 LVDIN 引脚输入的外部电压进行监测。若检测到电压低于指定值可提供一个警告信号。此功能在电池类产品中非常有用，在电池电压较低时产生警告信号。低电压检测也可产生中断信号。

LVD 寄存器

低电压检测功能由 LVDC 寄存器控制。VLVD2~VLVD0 位用于选择 8 个固定电压中的一个参考点。LVDO 位被置位时表示有低电压情况发生，若 LVDO 位为低表明 V_{DD} 电压或 LVDIN 输入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 LVD 功能，反之，关闭内部低电压检测电路。低电压检测会有一定的功耗，在不使用时可考虑关闭此功能，此举在功耗要求严格的电池供电应用中值得考虑。

- **LVDC 寄存器**

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

Bit 7~6　　未定义，读为“0”

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

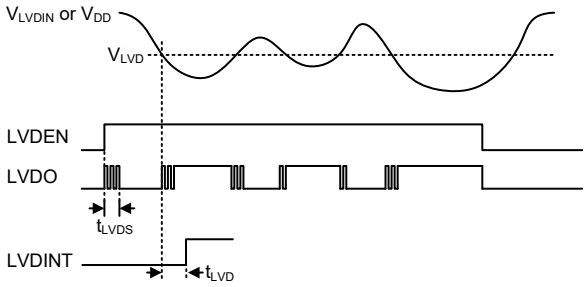
Bit 4 **LVDEN**：低电压检测控制位
0：除能
1：使能

Bit 3 **VBGEN**：Bandgap 缓冲器控制位
0：除能
1：使能
当 LVD、LVR 使能或者 VBGEN 位置高时，Bandgap 电路使能。

Bit 2~0 **VLVD2~VLVD0**：LVD 功能工作条件选择位
000：V_{LVDIN}≤1.23V
001：2.2V
010：2.4V
011：2.7V
100：3.0V
101：3.3V
110：3.6V
111：4.0V
当设置 VLVD2~VLVD0 位为 000 时，LVD 功能会对 LVDIN 引脚输入电压和 1.23V 参考电压进行比较，执行对 LVDIN 输入电压的监测。当 VLVD2~VLVD0 位设为 000 以外的的其它值时，LVD 将电源电压与这些位指定的参考电压进行比较以监测电源电压值。

LVD 操作

通过比较电源电压 V_{DD} 或 LVDIN 输入电压 V_{LVDIN} 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。低电压检测参考电压可以设置为 1.23V~4.0V 之间的八个固定电压值。当电源电压 V_{DD} 或 LVDIN 输入电压低于预置电压值时，LVDO 位被置为高，表明低电压产生。当单片机进入休眠模式时，即使 LVDEN 位为高，低电压检测器也会被除能。低电压检测器使能后，读取 LVDO 位前，电路稳定需要一定的延时 t_{LVDS}。注意，V_{DD} 或 V_{LVDIN} 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，包含于多功能中断中。LVD 中断是除了轮询 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} 或 V_{LVDIN} 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，在单片机进入空闲模式前应将 LVF 标志置为高。在休眠模式下，LVD 功能总是除能。

中断

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

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特殊功能数据存储器中的一系列寄存器控制的。 寄存器总的分为两类。第一类是 INTC0~INTC3 寄存器，用于设置基本的中断；第二类是 MFI0~MFI8 寄存器，用于设置多功能中断。

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	—	—
外部中断	INTnE	INTnF	n=0~1
错误信号 (ERSG)	ERSGE	ERSGF	—
相位保护	PHASEE	PHASEF	—
USIM	USIME	USIMF	—
A/D 转换器	ADE	ADF	—
多功能中断	MFnE	MFnF	n=0~8
OVPn 功能	OVPnE	OVPnF	n=0~8
抖频功能	FJTME	FJTMF	—
	FJEQUE	FJEQUF	—
时基	TBnE	TBnF	n=0~1
PWM 产生器	PWMPE	PWMPF	—
	PWMDE	PWMDF	—
低电压检测	LVE	LVF	—
EEPROM	DEE	DEF	—
10-bit TMC	TMCPE	TMCPF	—
	TMCAE	TMCAF	
定时器模块	CTMnPE	CTMnPF	n=0~3
	CTMnAE	CTMnAF	
	PTMPE	PTMPF	—
	PTMAE	PTMAF	

中断相关位命名模式

寄存器名称	位							
	7	6	5	4	3	2	1	0
INTC0	—	PHASEF	ERSGF	INT0F	PHASEE	ERSGE	INT0E	EMI
INTC1	USIMF	MF1F	MF0F	INT1F	USIME	MF1E	MF0E	INT1E
INTC2	MF4F	MF3F	ADF	MF2F	MF4E	MF3E	ADE	MF2E
INTC3	MF8F	MF7F	MF6F	MF5F	MF8E	MF7E	MF6E	MF5E
MFI0	—	OVP8F	OVP7F	OVP6F	—	OVP8E	OVP7E	OVP6E
MFI1	OVP5F	OVP4F	OVP3F	OVP2F	OVP5E	OVP4E	OVP3E	OVP2E
MFI2	—	—	FJEQUF	FJTMF	—	—	FJEQUE	FJTME
MFI3	—	—	CTM3AF	CTM3PF	—	—	CTM3AE	CTM3PE
MFI4	—	TB0F	PWMDF	PWMPF	—	TB0E	PWMDE	PWMPE
MFI5	OVP1F	OVP0F	PTMAF	PTMPF	OVP1E	OVP0E	PTMAE	PTMPE
MFI6	TB1F	TMCPF	CTM0AF	CTM0PF	TB1E	TMCPE	CTM0AE	CTM0PE
MFI7	LVF	TMCAF	CTM1AF	CTM1PF	LVE	TMCAE	CTM1AE	CTM1PE
MFI8	—	DEF	CTM2AF	CTM2PF	—	DEE	CTM2AE	CTM2PE

中断寄存器列表

- **INTC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PHASEF	ERSGF	INT0F	PHASEE	ERSGE	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 **PHASEF**：相位保护中断请求标志位
0：无请求
1：中断请求

Bit 5 **ERSGF**：错误信号 (ERSG) 中断请求标志位
0：无请求
1：中断请求

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

Bit 3 **PHASEE**：相位保护中断控制位
0：除能
1：使能

Bit 2 **ERSGE**：错误信号 (ERSG) 中断控制位
0：除能
1：使能

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

Bit 0 **EMI**：总中断控制位
0：除能
1：使能

● **INTC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	USIMF	MF1F	MF0F	INT1F	USIME	MF1E	MF0E	INT1E
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 **USIMF**：USIM 中断请求标志位
0：无请求
1：中断请求

Bit 6 **MF1F**：多功能中断 1 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 **INT1F**：外部中断 1 请求标志位
0：无请求
1：中断请求

Bit 3 **USIME**：USIM 中断控制位
0：除能
1：使能

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

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

Bit 0 **INT1E**：外部中断 1 中断控制位
0：除能
1：使能

● **INTC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MF4F	MF3F	ADF	MF2F	MF4E	MF3E	ADE	MF2E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **MF4F**：多功能中断 4 请求标志位
0：无请求
1：中断请求

Bit 6 **MF3F**：多功能中断 3 请求标志位
0：无请求
1：中断请求

Bit 5 **ADF**：A/D 转换完成中断请求标志位
0：无请求
1：中断请求

Bit 4 **MF2F**：多功能中断 2 请求标志位
0：无请求
1：中断请求

Bit 3 **MF4E**：多功能中断 4 中断控制位
0：除能
1：使能

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

Bit 1 **ADE**：A/D 转换完成中断控制位
0：除能
1：使能

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

- **INTC3 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MF8F	MF7F	MF6F	MF5F	MF8E	MF7E	MF6E	MF5E
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 **MF8F**：多功能中断 8 请求标志位
0：无请求
1：中断请求

Bit 6 **MF7F**：多功能中断 7 请求标志位
0：无请求
1：中断请求

Bit 5 **MF6F**：多功能中断 6 请求标志位
0：无请求
1：中断请求

Bit 4 **MF5F**：多功能中断 5 请求标志位
0：无请求
1：中断请求

Bit 3 **MF8E**：多功能中断 8 中断控制位
0：除能
1：使能

Bit 2 **MF7E**：多功能中断 7 中断控制位
0：除能
1：使能

Bit 1 **MF6E**：多功能中断 6 中断控制位
0：除能
1：使能

Bit 0 **MF5E**：多功能中断 5 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	—	OVP8F	OVP7F	OVP6F	—	OVP8E	OVP7E	OVP6E
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 **OVP8F**：OVP8 中断请求标志位
0：无请求
1：中断请求

Bit 5 **OVP7F**：OVP7 中断请求标志位
0：无请求
1：中断请求

Bit 4 **OVP6F**：OVP6 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 **OVP8E**：OVP8 中断控制位
0：除能
1：使能

Bit 1 **OVP7E**：OVP7 中断控制位
0：除能
1：使能

Bit 0 **OVP6E**：OVP6 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	OVP5F	OVP4F	OVP3F	OVP2F	OVP5E	OVP4E	OVP3E	OVP2E
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 **OVP5F**：OVP5 中断请求标志位
0：无请求
1：中断请求

Bit 6 **OVP4F**：OVP4 中断请求标志位
0：无请求
1：中断请求

Bit 5 **OVP3F**：OVP3 中断请求标志位
0：无请求
1：中断请求

Bit 4 **OVP2F**：OVP2 中断请求标志位
0：无请求
1：中断请求

Bit 3 **OVP5E**：OVP5 中断控制位
0：除能
1：使能

Bit 2 **OVP4E**：OVP4 中断控制位
0：除能
1：使能

Bit 1 **OVP3E**：OVP3 中断控制位
0：除能
1：使能

Bit 0 **OVP2E**：OVP2 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	—	—	FJEQUF	FJTMF	—	—	FJEQUE	FJTME
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　**FJEQUF**：FJEQU 中断请求标志位
0：无请求
1：中断请求

Bit 4　**FJTMF**：FJTMR 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　**FJEQUE**：FJEQU 中断控制位
0：除能
1：使能

Bit 0　**FJTME**：FJTMR 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	—	—	CTM3AF	CTM3PF	—	—	CTM3AE	CTM3PE
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　**CTM3AF**：CTM3 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4　**CTM3PF**：CTM3 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求

Bit 3~2　未定义，读为“0”

Bit 1　**CTM3AE**：CTM3 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0　**CTM3PE**：CTM3 比较器 P 匹配中断控制位
0：除能
1：使能

● **MFI4 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	TB0F	PWMDF	PWMPF	—	TB0E	PWMDE	PWMPE
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　**PWMDF**：PWM 占空比匹配中断请求标志位
0：无请求
1：中断请求

Bit 4　**PWMPF**：PWM 周期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　**PWMDE**：PWM 占空比匹配中断控制位
0：除能
1：使能

Bit 0　**PWMPE**：PWM 周期匹配中断控制位
0：除能
1：使能

● **MFI5 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OVP1F	OVP0F	PTMAF	PTMPF	OVP1E	OVP0E	PTMAE	PTMPE
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　**OVP1F**：OVP1 中断请求标志位
0：无请求
1：中断请求

Bit 6　**OVP0F**：OVP0 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　**OVP1E**：OVP1 中断控制位
0：除能
1：使能

Bit 2　**OVP0E**：OVP0 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

● **MFI6 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TB1F	TMCPF	CTM0AF	CTM0PF	TB1E	TMCPE	CTM0AE	CTM0PE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

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

Bit 6 **TMCPF**：TMC 周期中断请求标志位
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 **TB1E**：时基 1 中断控制位
0：除能
1：使能

Bit 2 **TMCPE**：TMC 周期中断控制位
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

● **MFI7 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	LVF	TMCAF	CTM1AF	CTM1PF	LVE	TMCAE	CTM1AE	CTM1PE
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 **LVF**：LVD 中断请求标志位
0：无请求
1：中断请求

Bit 6 **TMCAF**：TMCA 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 **LVE**：LVD 中断控制位
0：除能
1：使能

Bit 2 **TMCAE**：TMCA 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

- **MFI8 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	DEF	CTM2AF	CTM2PF	—	DEE	CTM2AE	CTM2PE
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 **CTM2AF**：CTM2 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4 **CTM2PF**：CTM2 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 **CTM2AE**：CTM2 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0 **CTM2PE**：CTM2 比较器 P 匹配中断控制位
0：除能
1：使能

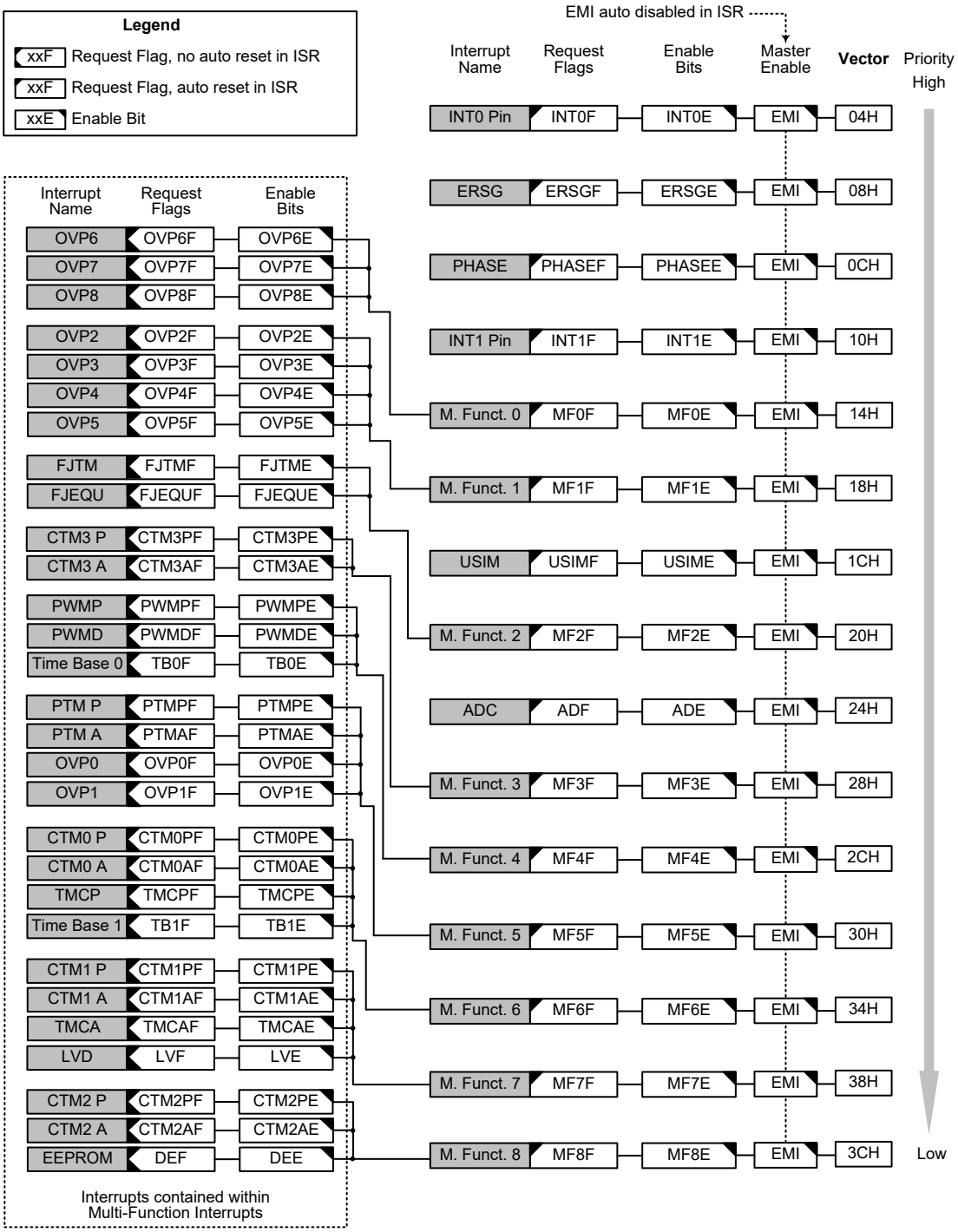
中断操作

若中断事件的条件产生，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“JMP”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。所有中断请求标志位在被置起时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中断请求标志置起。


Legend
xxF Request Flag, no auto reset in ISR
xxF Request Flag, auto reset in ISR
xxE Enable Bit
EMI auto disabled in ISR
Interrupt Name
Request Flags
Enable Bits
Master Enable
Vector
Priority
High
Low
INT0 Pin INT0F INT0E EMI 04H
ERSG ERSGF ERSGE EMI 08H
PHASE PHASEF PHASEE EMI 0CH
INT1 Pin INT1F INT1E EMI 10H
M. Funct. 0 MF0F MF0E EMI 14H
M. Funct. 1 MF1F MF1E EMI 18H
USIM USIMF USIME EMI 1CH
M. Funct. 2 MF2F MF2E EMI 20H
ADC ADF ADE EMI 24H
M. Funct. 3 MF3F MF3E EMI 28H
M. Funct. 4 MF4F MF4E EMI 2CH
M. Funct. 5 MF5F MF5E EMI 30H
M. Funct. 6 MF6F MF6E EMI 34H
M. Funct. 7 MF7F MF7E EMI 38H
M. Funct. 8 MF8F MF8E EMI 3CH
OVP6 OVP6F OVP6E
OVP7 OVP7F OVP7E
OVP8 OVP8F OVP8E
OVP2 OVP2F OVP2E
OVP3 OVP3F OVP3E
OVP4 OVP4F OVP4E
OVP5 OVP5F OVP5E
FJTM FJTMF FJTME
FJEQU FJEQUF FJEQUE
CTM3 P CTM3PF CTM3PE
CTM3 A CTM3AF CTM3AE
PWMP PWMPF PWMPE
PWMD PWMDF PWMDE
Time Base 0 TB0F TB0E
PTM P PTMPF PTMPE
PTM A PTMAF PTMAE
OVP0 OVP0F OVP0E
OVP1 OVP1F OVP1E
CTM0 P CTM0PF CTM0PE
CTM0 A CTM0AF CTM0AE
TMCP TMCPF TMCPE
Time Base 1 TB1F TB1E
CTM1 P CTM1PF CTM1PE
CTM1 A CTM1AF CTM1AE
TMCA TMCAF TMCAE
LVD LVF LVE
CTM2 P CTM2PF CTM2PE
CTM2 A CTM2AF CTM2AE
EEPROM DEF DEE
Interrupts contained within Multi-Function Interrupts

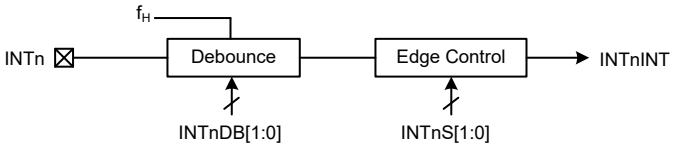
中断结构

外部中断

通过 INT0~INT1 引脚上的信号变化可控制外部中断。当触发沿选择位设置好触发类型，INT0~INT1 引脚的状态发生相应变化时，外部中断请求标志 INT0F~INT1F 被置位，外部中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和相应中断使能位 INT0E~INT1E 需先被置位。此外，必须使用 INTEG 寄存器使能外部中断功能并选择触发沿类型。外部中断引脚和普通 I/O 口共用，需通过相关的引脚共用功能选择寄存器选择 INTn 引脚功能，并通过引脚控制寄存器选择对应引脚为输入类型。当中断使能，堆栈未满并且外部中断脚状态改变，将调用外部中断向量子程序。当响应外部中断服务子程序时，中断请求标志位 INT0F~INT1F 会自动复位且 EMI 位会被清零以除能其它中断。

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

INT0~INT1 引脚都配置了去抖动电路，该功能使用内部时钟 f_H 产生一个去抖时间，以减少外部中断线上毛刺导致单片机误动作的可能性。通过设置 INTDB 寄存器的 INTnDB[1:0] 位，使能相应引脚的去抖功能且可依不同应用配置一个合适的去抖时间。需注意的是，当 INTnDB[1:0] 位设置为 01、10 或 11 启用引脚去抖功能后，如果 f_H 时钟关闭，则无法通过外部中断唤醒单片机。



外部中断 (n=0~1)

- **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 外部中断边沿控制
00：除能
01：上升沿
10：下降沿
11：双沿

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

● **INTDB 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1DB1	INT1DB0	INT0DB1	INT0DB0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未定义，读为“0”

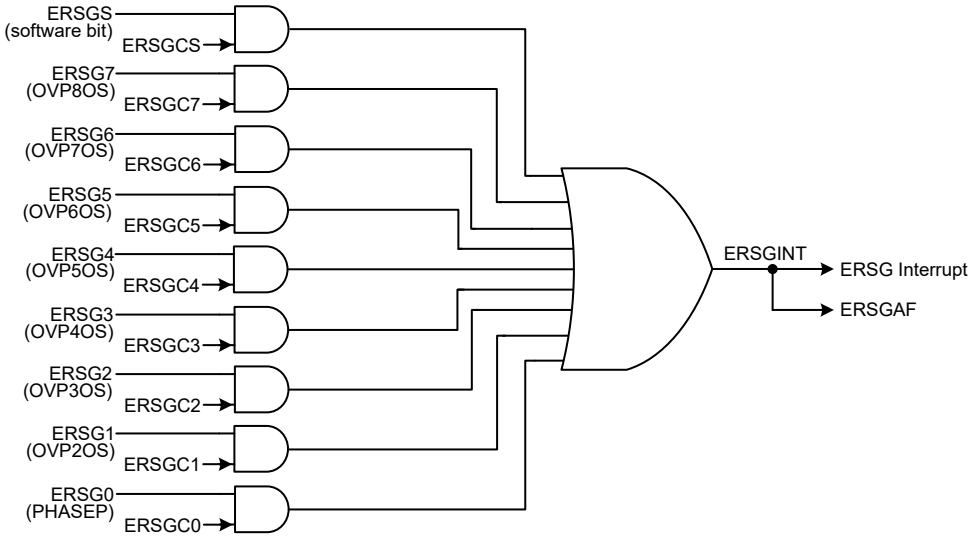
Bit 3~2　**INT1DB1~INT1DB0**：INT1 外部中断去抖时间选择
00：旁路，无去抖
01：(7~8) × t_H
10：(15~16) × t_H
11：(23~24) × t_H

Bit 1~0　**INT0DB1~INT0DB0**：INT0 外部中断去抖时间选择
00：旁路，无去抖
01：(7~8) × t_H
10：(15~16) × t_H
11：(23~24) × t_H

注：1. $t_H=1/f_H$
2. 当 INTnDB[1:0] 不为 00 时，若 f_H 时钟关闭，则无法通过外部中断唤醒单片机。

错误信号中断

共有九种信号可触发错误信号 (ERSG) 中断，分别为相位保护信号 (PHASEP)，OVP2~OVP8 输出信号 (OVP2OS~OVP8OS) 以及 ERSG 软件触发位 (ERSGS)。



如上图所示，当有其中某一错误信号发生且其对应的错误信号控制位 ERSGCn 位或 ERSGCS 位为高时，将会置位 ERSGF 中断请求标志位，ERSG 中断请求发生。若要跳转到相应的中断向量地址，总中断使能位 EMI 和 ERSG 中断使能位 ERSGE 需先被置位。当中断使能，堆栈未满并且有错误信号产生且对应的控制位为高时，将调用 ERSG 中断向量子程序。当响应中断服务子程序时，ERSG 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。

相位保护中断

相位保护中断由相位保护侦测 0 及相位保护侦测 1 电路控制。当相位保护侦测电路的输入信号，Phase0 或 Phase1 未检测到相位或者宽度过窄 (小于设定值)，

相位保护中断请求标志 PHASEF 被置位，相位保护中断请求产生。若要跳转到相应中断向量地址，总中断使能位 EMI 和相位保护中断使能位 PHASEE 需先被置位。当中断使能，堆栈未满并且相位信号宽度为零或过小时，将调用相位保护中断向量子程序。当响应中断服务子程序时，相位保护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。

USIM 中断

通用串行接口模块中断，即 USIM 中断。当 USIM 接口中断标志位 USIMF 置位时，产生中断请求。由于 USIM 接口可工作在三个模式下：SPI 模式、I^2C 模式和 UART 模式，USIMF 标志位置位可由不同情况触发，取决于所选择的接口模式。

若选择 SPI 或 I^2C 模式，当一个字节数据已由 SPI/I^2C 接口接收或发送完，或 I^2C 从机地址匹配，或 I^2C 超时，中断请求标志 USIMF 被置位，USIM 中断请求产生。若选择 UART 模式，USIM 中断由几种 UART 传输条件控制。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，USIM 中断请求标志 USIMF 被置位，USIM 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 和通用串行接口中断使能位 USIME 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，将调用相应的 USIM 中断向量子程序。当响应中断服务子程序时，通用串行接口中断标志位 USIMF 会自动复位且 EMI 将被自动清零以除能其它中断。

注意，当 USIM 中断是由 UART 接口触发产生的，当中断响应后，UUSR 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。

A/D 转换器中断

A/D 转换器中断由一次手动触发的 A/D 转换或一组自动触发的 A/D 转换动作的结束来控制。当 A/D 转换器中断请求标志被置位，即 A/D 转换过程完成时，中断请求发生。若要跳转到相应中断向量地址， 总中断控制位 EMI 和 A/D 转换器中断使能位 ADE 需先被置位。当中断使能，堆栈未满且 A/D 转换动作结束时，将调用 A/D 转换器中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 ADF 会自动复位且 EMI 位会被清零以除能其它中断。

多功能中断

此单片机中有多达 9 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，分别为时基中断，PWM 周期和占空比匹配中断，OVPn 中断，抖频功能中 FJTIMER 溢出中断以及逼近匹配中断、TM 中断，TMC 中断，LVD 中断和 EEPROM 中断。

当多功能中断中任何一个中断源请求发生，其所在的多功能中断请求标志位 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，将调用此多功能中断向量中的子程序。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，虽然多功能中断请求标志会自动复位，但多功能中断源的请求标志位不会自动复位，必须由应用程序清零。

OVPn 中断

OVP 中断，即过电压保护中断，属于多功能中断。当有预设的过电压事件发生时，OVPn 中断请求标志位 OVPnF 被置位，OVPn 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，中断使能位 OVPnE 以及相应的多功能中断使能位都需先被置位。当中断使能，堆栈未满并且有过电压情况发生时，将调用相应的多功能中断向量子程序。当 OVPn 中断响应，EMI 将被自动清零以除能其它中断，相关多功能中断请求标志也可被自动清零，但 OVPnF 标志位需在应用程序中手动清零。

FJTMR 中断

抖频 FJTIMER 溢出中断，即 FJTMR 中断，属于多功能中断。当抖频功能使用的 FJTIMER 计数器溢出时，相应的中断请求标志位 FJTMF 将置位，FJTMR 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，FJTMR 中断使能位 FJTME 以及相应的多功能中断使能位都需先被置位。当中断使能，堆栈未满且 FJTIMER 计数器溢出时，可跳转至相关多功能中断向量子程序中执行。当 FJTMR 中断响应，EMI 将被自动清零以除能其它中断，相关多功能中断标志也可自动清零，但 FJTMF 请求标志位需在应用程序中手动清零。

FJEQU 中断

抖频逼近匹配中断，即 FJEQU 中断，属于多功能中断。在实施抖频过程中，PWMP/DT0/DT1 逼近目标 A 或目标 B 完成时，相应的中断请求标志位 FJEQUF 将置位，FJEQU 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，FJEQU 中断使能位 FJEQUE 以及相应的多功能中断使能位都需先被置位。当中断使能，堆栈未满且 PWMP/DT0/DT1 逼近匹配完成时，可跳转至相关多功能中断向量子程序中执行。当 FJEQU 中断响应，EMI 将被自动清零以除能其它中断，相关多功能中断标志也可自动清零，但 FJEQUF 请求标志位需在应用程序中手动清零。

TM 中断

每个简易型或周期型 TM 各有两个中断，一个来自比较器 A 匹配，一个来自比较器 P 匹配，都属于多功能中断。简易型和周期型 TM 分别有两个中断请求标志位为 xTMnPF 和 xTMnAF，及两个使能位分别为 xTMnPE 和 xTMnAE。当 TM 比较器 P 或比较器 A 匹配情况发生时，相应 TM 中断请求标志和对应多功能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，相关多功能中断标志也可自动清零，但 TM 中断请求标志需在应用程序中手动清零。

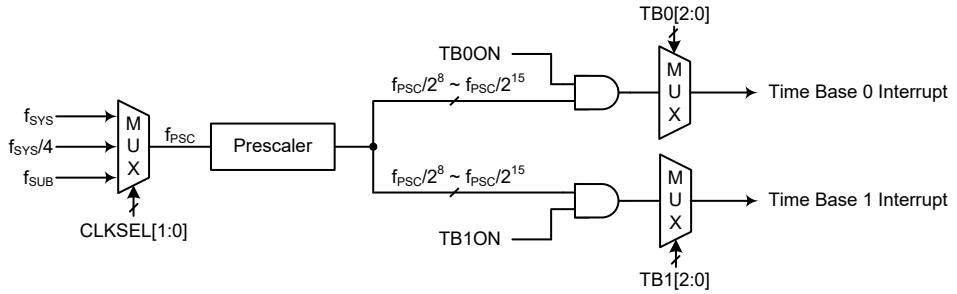
PWM 产生器中断

PWM 产生器包含 2 个中断，一个 PWM 周期匹配中断即 PWMP 中断和一个 PWM 占空比匹配中断即 PWMD 中断，它们都属于多功能中断。当 PWM 占空比匹配或 PWM 周期匹配时，中断请求标志位 PWMDF 或 PWMPF 以及对应多功能中断请求标志位被置位，PWM 占空比匹配或周期匹配中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI、相应多功能中断使能位、PWM 周期或占空比匹配中断使能位，PWMPE 或 PWMDE，需先被置位。当中断使能，堆栈未满并且 PWM 占空比匹配或 PWM 周期匹配时，将调用相应的多功能中断向量子程序。当响应此中断服务子程序时，EMI 位会被清零以除能其它中断，多功能中断请求标志也可自动清零，但 PWMDF 或 PWMPF 标志需在应用程序中手动清零。

时基中断

时基中断属于多功能中断。时基中断用于提供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。当时基溢出时，时基中断请求标志位 TBnF 被置位，时基中断请求发生。当总中断使能位 EMI，时基使能位 TBnE 以及所在的多功能中断使能位被置位，允许程序跳转到相应的多功能中断向量地址。当中断使能，堆栈未满且时基溢出时，将调用相应的多功能中断向量子程序。当响应中断服务子程序时，EMI 位会被清零以除能其它中断，多功能中断请求标志也可自动清零。而时基中断请求标志位 TBnF 需使用应用程序手动清零。

时基中断的目的是提供一个固定周期的中断信号。时钟源 f_{PSC} 来自内部时钟源 f_{SYS}、$f_{SYS}/4$ 或 f_{SUB}，通过 PSCR 寄存器的 CLKSEL[1:0] 位选择。选择的内部时钟源时钟首先经过分频器，之后再由 TBnC 寄存器的 TBn[2:0] 位对分频值进行选择以提供更长的时基中断周期。



时基中断

- **PSCR 寄存器**

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

Bit 7~2　未定义，读为“0”

Bit 1~0　**CLKSEL1~CLKSEL0**：分频器时钟源 f_{PSC} 选择
00：f_{SYS}
01：$f_{SYS}/4$
1x：f_{SUB}

- **TB0C 寄存器**

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

Bit 7　**TB0ON**：时基 0 使能控制
0：除能
1：使能

Bit 6~3　未定义，读为“0”

Bit 2~0　**TB02~TB00**：时基 0 溢出周期选择
000：$2^8/f_{PSC}$
001：$2^9/f_{PSC}$
010：$2^{10}/f_{PSC}$
011：$2^{11}/f_{PSC}$

100：$2^{12}/f_{PSC}$
101：$2^{13}/f_{PSC}$
110：$2^{14}/f_{PSC}$
111：$2^{15}/f_{PSC}$

- **TB1C 寄存器**

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

Bit 7 **TB1ON**：时基 1 使能控制
0：除能
1：使能

Bit 6~3 未定义，读为“0”

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

TMC 中断

10-bit TMC 定时 / 事件计数器包含两个中断，即 TMCP 和 TMCA 中断。TMCP 中断由 10-bit TMC 计数器溢出触发；当预设的时间到达或者在 TMC 输入捕捉到有效的边沿转换时，计数器值被移入 TMCCCRA 寄存器，触发 TMCA 中断。这两个 TMC 中断都属于多功能中断。当相应的中断触发事情发生时，会置位其中断请求标志位，TMCPF 和 TMCAF，中断请求发生。

若要程序跳转到相应中断向量地址，总中断控制位 EMI、相应 TMCA 或 TMCP 中断使能位和相关多功能中断使能位需先被置位。当中断使能，堆栈未满且发生 TMC 计数器溢出或 TMC 计数器当前值被移入 TMCCCRA 寄存器时，可跳转至相关多功能中断向量子程序中执行。当 TMCA 或 TMCP 中断响应，EMI 将被自动清零以除能其它中断，相关多功能中断标志也可自动清零，但 TMCAF 或 TMCPF 中断请求标志需在应用程序中手动清零。

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源电压或 LVDIN 引脚输入低电压时，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 以及相应的多功能中断使能位都需先被置位。当中断使能，堆栈未满且低电压条件发生时，可跳转至相关多功能中断向量子程序中执行。当低电压中断响应，EMI 将被自动清零以除能其它中断，相关多功能中断标志也可自动清零，但 LVF 请求标志位需在应用程序中手动清零。

EEPROM 中断

EEPROM 中断属于多功能中断。当擦 / 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 擦 / 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零，但 DEF 标志需在应用程序中手动清零。

中断唤醒功能

每个中断都具有将处于休眠或空闲模式的单片机唤醒的能力。当中断请求标志由低到高转换时唤醒动作产生，其与中断是否使能无关。因此，尽管单片机处于休眠或空闲模式且系统振荡器停止工作，如有外部中断脚上产生外部边沿跳变或低电压都可能导致其相应的中断请求标志被置位，由此产生中断，因此必须注意避免伪唤醒情况的发生。若要除能中断唤醒功能，应在单片机进入休眠或空闲模式前将相应中断请求标志置起。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可以自动清零，但各自的请求标志需在应用程序中手动清零。

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中断请求标志置为高。

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。

应用说明 – 半桥电磁炉

半桥电磁炉产品主要分成四项功能，功率控制、相位侦测、保护、功率测量和显示，半桥电磁炉通过一组互补式 PWM 信号来实现对半桥电磁炉的功率控制。该互补式 PWM 信号产生电路包含了一组可编程死区时间插入电路。另外支持抖频功能。在串联的谐振电路中通过电流传感器和 PWM 信号做相位侦测；在功率控制过程中侦测 IGBT 谐振过电流、IGBT 谐振短路电流、AC 过电流、AC 电压过零，当侦测到异常时，启动保护功能，每一个信号皆有 4 个保护等级可以选择；A/D 转换器内建 2 通道的自动转换功能；另外包含测量功率与显示。Holtek 半桥电磁炉专用 MCU HT45F0075 提供了上述所需功能的控制信号，现介绍如下。

半桥电磁炉功率控制工作原理

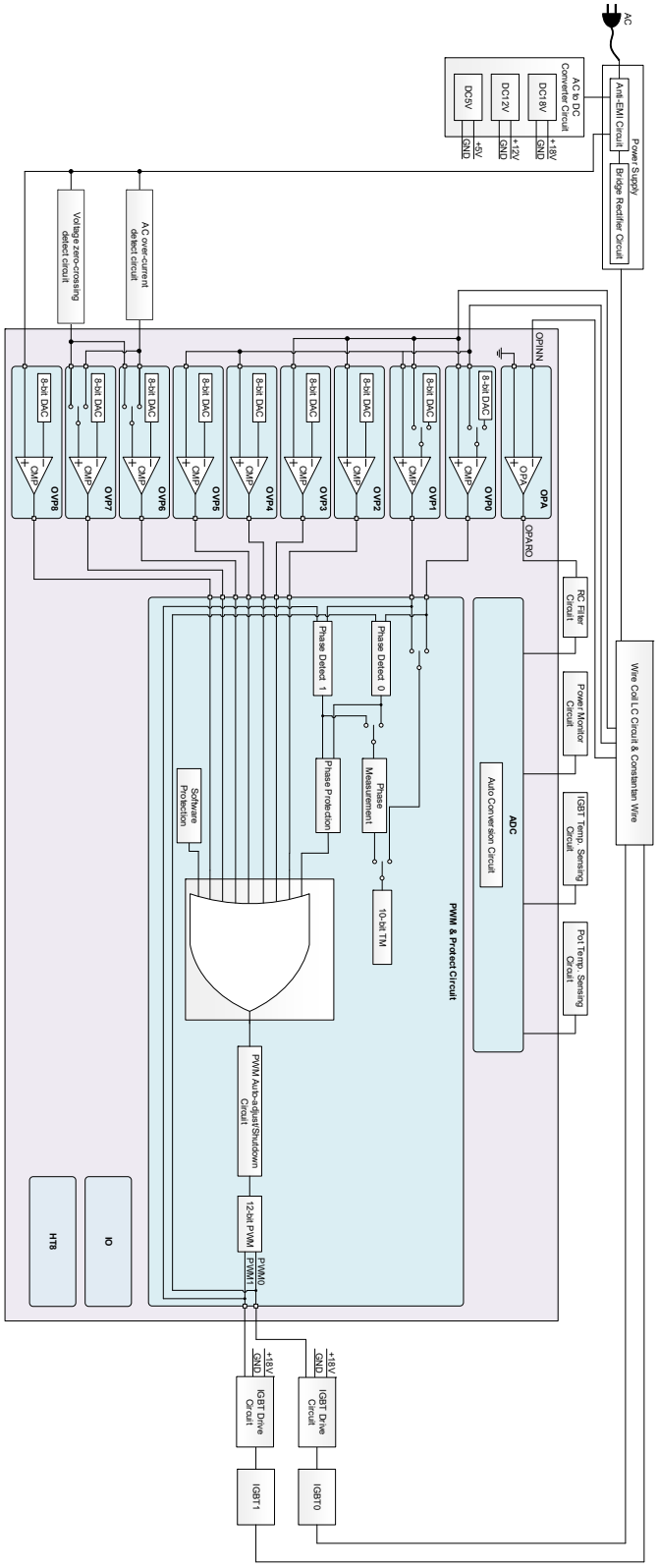
通过控制互补式 PWM 输出频率高低来决定半桥电磁炉功率的大小。当 PWM 频率大于谐振频率，电压相位超前电流相位越多，谐振电流会越小，半桥电磁炉的功率会越小；当 PWM 频率越接近谐振频率，电压相位超前电流相位越小，谐振电流会越大，半桥电磁炉的功率会越大。

死区插入设计的目的：死区插入电路可防止外部驱动电路晶体管对在转态时，上下臂 MOS 同时开启，发生上下臂瞬间导通产生短路电流。为了消除这种危险，此单片机内置了两组死区插入电路，各自具有 12-bit 死区时间控制，分别对两路 PWM 信号进行控制。通过对死区时间的合理设置可确保输出转态的过程中，两个晶体管不会处于都导通的状态。

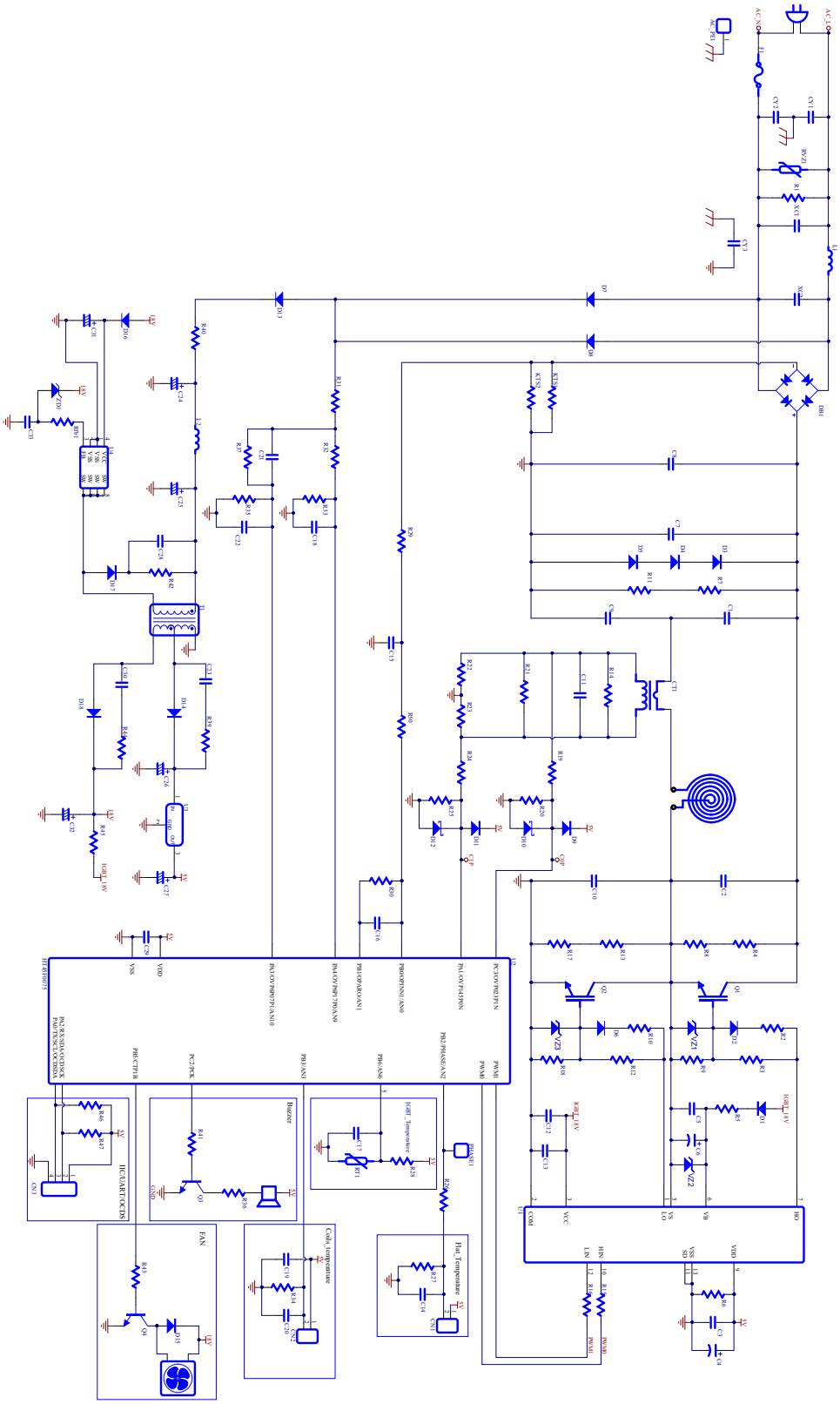
相位侦测和保护工作原理

相位侦测电路有 2 组分为 Phase Detect 0 和 Phase Detect 1。Phase Detect 0 侦测 C0X (OVP0OS，电流信号) 和 PWM0 (电压信号) 输入信号，Phase0 为输出信号，Phase Detect 1 侦测 C1X (OVP1OS，电流信号) 和 PWM1 (电压信号) 输入信号，Phase1 为输出信号，如果电流信号落后电压信号，此时在相位侦测 0/1 电路会有 Phase 信号，利用 Phase 信号宽度可用来计算电流落后电压相位宽度来调整电磁炉输出功率。当电流信号和电压信号没有相位差，此时 Phase 宽度为 0，半桥电磁炉输出功率最大。如果电流信号领先电压信号，此时在相位锁定电路会无法侦测到 Phase 宽度，保护机制电路就会启动。当保护机制电路启动后，会根据设定决定执行其中一个保护机制。保护机制有四种，分别为：PWM 立即关闭；PWM 不立即关闭，待执行完目前整个周期后关闭；PWM 会在每一个 PWM 周期调变，直到 PWM 调变至小于设定值后关闭；PWM 会在每一个 PWM 周期调变，直到 PWM 调变至小于设定值后但不关闭。

硬件方框图


AC
Power Supply
Anti-EMI Circuit
Bridge Rectifier Circuit
AC to DC Converter Circuit
DC5V
DC12V
DC18V
+5V
GND
+12V
GND
+18V
GND
Voltage zero-crossing detect circuit
AC over-current detect circuit
OPINN
OPA
OPAO
OVP0
OVP1
OVP2
OVP3
OVP4
OVP5
OVP6
OVP7
OVP8
8-bit DAC
CMP
RC Filter Circuit
Power Monitor Circuit
IGBT Temp. Sensing Circuit
Pcl Temp. Sensing Circuit
Wire Coil LC Circuit & Constantan Wire
ADC
Auto Conversion Circuit
PWM & Protect Circuit
Phase Detect 1
Phase Detect 0
Phase Protection
Phase Measurement
10-bit TM
Software Protection
PWM Auto-Reset/Shutdown Circuit
12-bit PWM
PWM0
PWM1
HTB
IO
+18V
GND
IGBT Drive Circuit
IGBT1
IGBT0

硬件电路图



指令集

简介

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

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

指令周期

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

数据的传送

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

算术运算

算术运算和数据处理是大部分单片机应用所必需具备的能力，在 Holtek 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 Holtek 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 Holtek 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，Holtek 单片机允许在程序存储器中建立一个表格作为数据可直接存储的区域，只需要一组简易的指令即可对数据进行查表。

其它运算

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

指令集概要

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

惯例

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

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

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

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

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

扩展指令集

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

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

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

注：1. 对扩展跳转指令而言，如果比较的结果牵涉到跳转即需 3 个周期，如果没有发生跳转，则只需两个周期。

2. 任何扩展指令若要改变 PCL 的内容将需要 3 个周期来执行。

指令定义

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

RETI Return from interrupt

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

功能表示 Program Counter ←Stack
EMI ← 1

影响标志位 无

RL [m] Rotate Data Memory left

指令说明 将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。

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

影响标志位 无

RLA [m] Rotate Data Memory left with result in ACC

指令说明 将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位，结果送到累加器，而指定数据存储器的内容保持不变。

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

影响标志位 无

RLC [m] Rotate Data Memory Left through Carry

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

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

影响标志位 C

RLC A [m] Rotate Data Memory left through Carry with result in ACC

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

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

影响标志位 C

RR [m] Rotate Data Memory right

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

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

影响标志位 无

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

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

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

影响标志位 无

RRC [m] Rotate Data Memory right through Carry

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

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

影响标志位 C

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

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

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

影响标志位 C

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

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

功能表示 ACC ← ACC – [m] – $\overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

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

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

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

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

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

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

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

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

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

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

SUB A, [m] Subtract Data Memory from ACC

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

功能表示 ACC ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

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

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

功能表示 [m] ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

SUB A, x Subtract immediate Data from ACC

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

功能表示 ACC ← ACC – x

影响标志位 OV、Z、AC、C、SC、CZ

SWAP [m] Swap nibbles of Data Memory

指令说明 将指定数据存储器的低 4 位和高 4 位互相交换。

功能表示 [m].3~[m].0 ↔ [m].7~[m].4

影响标志位 无

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

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

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

影响标志位 无

SZ [m] Skip if Data Memory is 0

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

功能表示 如果 [m]=0，跳过下一条指令执行

影响标志位 无

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

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

功能表示 ACC ← [m]，如果 [m]=0，跳过下一条指令执行

影响标志位 无

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

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

功能表示 如果 [m].i=0，跳过下一条指令执行

影响标志位 无

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

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

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

影响标志位 无

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

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

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

影响标志位 无

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

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

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

影响标志位 无

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

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

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

影响标志位 无

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

指令说明 将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。

功能表示 ACC ← ACC“XOR”[m]

影响标志位 Z

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

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

功能表示 [m] ← ACC“XOR”[m]

影响标志位 Z

XOR A, x Logical XOR immediate data to ACC

指令说明 将累加器的数据与立即数逻辑异或，结果存放到累加器。

功能表示 ACC ← ACC“XOR”x

影响标志位 Z

扩展指令定义

扩展指令被用来直接存取存储在任何数据存储器 Sector 中的数据。

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

LRR [m] Rotate Data Memory right

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

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

影响标志位 无

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

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

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

影响标志位 无

LRRC [m] Rotate Data Memory right through Carry

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

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

影响标志位 C

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

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

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

影响标志位 C

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

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

功能表示 ACC ← ACC – [m] – $\overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

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

LSDZ [m] Skip if Decrement Data Memory is 0
指令说明 将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 [m] ← [m] – 1，如果 [m]=0 跳过下一条指令执行
影响标志位 无

LSDZA [m] Skip if decrement Data Memory is zero with result in ACC
指令说明 将指定数据存储器内容减 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果将存放到累加器，但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 ACC ← [m] – 1，如果 ACC=0 跳过下一条指令执行
影响标志位 无

LSET [m] Set Data Memory
指令说明 将指定数据存储器的每一个位置位为 1。
功能表示 [m] ← FFH
影响标志位 无

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

LSIZ [m] Skip if increment Data Memory is 0

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

功能表示 [m] ←[m] + 1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

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

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

功能表示 ACC ←[m] + 1，如果 ACC=0 跳过下一条指令执行

影响标志位 无

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

指令说明 判断指定数据存储器的第 i 位，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果为 0，则程序继续执行下一条指令。

功能表示 如果 [m].i≠0，跳过下一条指令执行

影响标志位 无

LSNZ [m] Skip if Data Memory is not 0

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

功能表示 如果 [m]≠0，跳过下一条指令执行

影响标志位 无

LSUB A, [m] Subtract Data Memory from ACC

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

功能表示 ACC ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

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

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

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

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

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

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

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

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

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

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

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

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

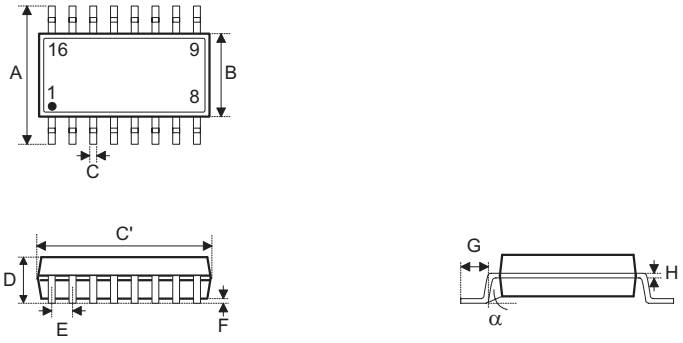
封装信息

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

封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

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

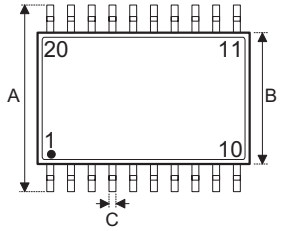
16-pin NSOP (150mil) 外形尺寸

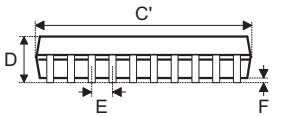


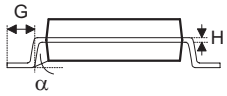
符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A		0.236 BSC	
B		0.154 BSC	
C	0.012	—	0.020
C’		0.390 BSC	
D	—	—	0.069
E		0.050 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A		6.00 BSC	
B		3.90 BSC	
C	0.31	—	0.51
C’		9.90 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 SOP (300mil) 外形尺寸



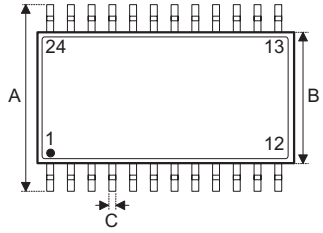


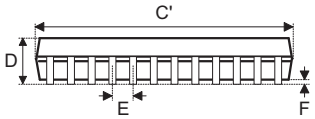


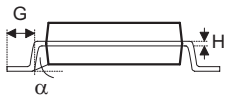
符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A		0.406 BSC	
B		0.295 BSC	
C	0.012	—	0.020
C’		0.504 BSC	
D	—	—	0.104
E		0.050 BSC	
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A		10.30 BSC	
B		7.50 BSC	
C	0.31	—	0.51
C’		12.80 BSC	
D	—	—	2.65
E		1.27 BSC	
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

24-pin SOP (300mil) 外形尺寸



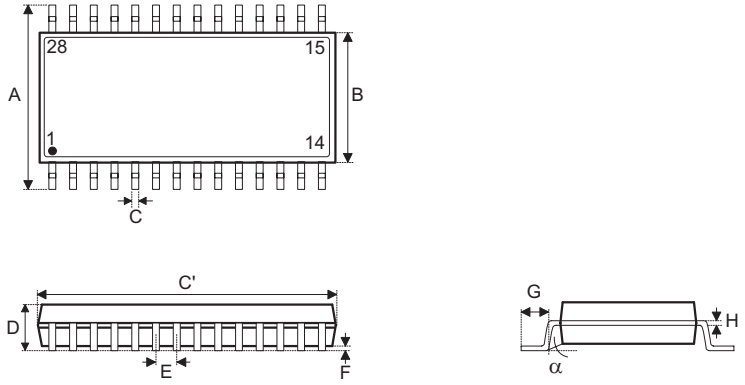




符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A		0.406 BSC	
B		0.295 BSC	
C	0.012	—	0.020
C’		0.606 BSC	
D	—	—	0.104
E		0.050 BSC	
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A		10.30 BSC	
B		7.50 BSC	
C	0.31	—	0.51
C’		15.40 BSC	
D	—	—	2.65
E		1.27 BSC	
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

28-pin SOP (300mil) 外形尺寸



符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A		0.406 BSC	
B		0.295 BSC	
C	0.012	—	0.020
C’		0.705 BSC	
D	—	—	0.104
E		0.050 BSC	
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A		10.30 BSC	
B		7.50 BSC	
C	0.31	—	0.51
C’		17.90 BSC	
D	—	—	2.65
E		1.27 BSC	
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°