



连续血糖监测 Flash 单片机

BH66F2475

版本: V1.20 日期: 2023-09-12

www.holtek.com

目录

特性	6
CPU 特性	6
周边特性	6
概述	7
方框图	8
引脚图	8
引脚说明	9
极限参数	12
直流电气特性	12
工作电压特性	12
工作电流特性	12
待机电流特性	13
交流电气特性	14
内部高速振荡器 – HIRC – 频率精准度	14
内部低速振荡器电气特性 – LIRC	14
工作频率电气特性曲线图	15
系统上电时间电气特性	15
输入 / 输出口电气特性	16
存储器电气特性	16
LVR 电气特性	17
模拟前端电路特性	18
LDO 电气特性	18
OPA 电气特性	19
内部参考电压电气特性	19
12-bit D/A 转换器电气特性	19
A/D 转换器电气特性	20
有效位数 – ENOB	20
PGA 转换器电气特性	21
I ² C 电气特性	21
上电复位特性	22
系统结构	23
时序和流水线结构	23
程序计数器	24
堆栈	25
算术逻辑单元 – ALU	26
Flash 程序存储器	27
结构	27
特殊向量	27
查表	28
查表范例	28

在线烧录 – ICP	29
片上调试 – OCDS	30
在线应用编程 – IAP	30
数据存储器	44
结构	44
数据存储器寻址	45
通用数据存储器	45
特殊功能数据存储器	45
特殊功能寄存器	47
间接寻址寄存器 – IAR0, IAR1, IAR2	47
存储器指针 – MP0, MP1H/MP1L, MP2H/MP2L	47
程序存储区指针 – PBP	48
累加器 – ACC	49
程序计数器低字节寄存器 – PCL	49
查表寄存器 – TBLP, TBHP, TBLH	49
状态寄存器 – STATUS	49
EEPROM 数据存储器	51
EEPROM 数据存储器结构	51
EEPROM 寄存器	51
EEPROM 读数据	53
EEPROM 页擦操作	53
EEPROM 写操作	54
写保护	55
EEPROM 中断	55
编程注意事项	55
振荡器	58
振荡器概述	58
系统时钟配置	58
内部 RC 振荡器 – HIRC	58
内部 32kHz 振荡器 – LIRC	59
工作模式和系统时钟	59
系统时钟	59
系统工作模式	60
控制寄存器	61
工作模式切换	62
待机电流的注意事项	65
唤醒	65
看门狗定时器	66
看门狗定时器时钟源	66
看门狗定时器控制寄存器	66
看门狗定时器操作	67
复位和初始化	68
复位功能	68
复位初始状态	71

输入 / 输出端口	76
上拉电阻	76
PA 口唤醒	76
输入 / 输出端口控制寄存器	77
引脚共用功能	77
输入 / 输出引脚结构	79
读端口功能	80
编程注意事项	81
定时器模块 – TM	82
简介	82
TM 操作	82
TM 时钟源	82
TM 中断	82
TM 外部引脚	83
编程注意事项	83
简易型 TM – CTM	85
简易型 TM 操作	85
简易型 TM 寄存器介绍	85
简易型 TM 工作模式	89
周期型 TM – PTM	95
周期型 TM 操作	95
周期型 TM 寄存器介绍	95
周期型 TM 工作模式	99
内部参考电压发生器	108
内部参考电压寄存器介绍	108
测量电路	109
测量电路寄存器介绍	109
测量计时控制	115
A/D 转换器	116
A/D 转换器简介	116
内部电源供电	117
A/D 转换寄存器介绍	118
A/D 转换器数据传输率的定义	123
A/D 转换器操作	123
A/D 转换步骤	124
编程注意事项	124
A/D 转换器传输功能	124
A/D 转换数据	125
A/D 转换数据转为电压值	126
串行接口模块 – SIM	126
SPI 接口	126
SPI 通信	130
I ² C 接口	133

16 位乘除法单元 – MDU	142
MDU 寄存器	142
乘除法单元操作	143
循环冗余校验 – CRC	145
CRC 寄存器	145
CRC 操作	146
CRC 计算	146
中断	148
中断寄存器	148
中断操作	152
外部中断	153
A/D 转换器中断	154
多功能中断	154
TM 中断	154
EEPROM 中断	154
时基中断	154
SIM 中断	156
比较器中断	156
中断唤醒功能	156
编程注意事项	156
配置选项	157
应用电路	157
指令集	158
简介	158
指令周期	158
数据的传送	158
算术运算	158
逻辑和移位运算	158
分支和控制转换	159
位运算	159
查表运算	159
其它运算	159
指令集概要	160
惯例	160
扩展指令集	163
指令定义	165
扩展指令定义	177
封装信息	187
SAW Type 16-pin QFN (3mm×3mm×0.75mm, FP0.25mm) 外形尺寸	188
SAW Type 24-pin QFN (3mm×3mm×0.55mm) 外形尺寸	189

特性

CPU 特性

- 工作电压：
 - ◆ $f_{SYS}=4\text{MHz}$: 2.2V~5.5V
 - ◆ $f_{SYS}=8\text{MHz}$: 2.2V~5.5V
 - ◆ $f_{SYS}=12\text{MHz}$: 2.7V~5.5V
- $V_{DD}=5\text{V}$, 系统时钟为 12MHz 时, 指令周期为 0.33 μs
- 暂停和唤醒功能, 以降低功耗
- 振荡器类型:
 - ◆ 内部高速 4/8/12MHz RC – HIRC
 - ◆ 内部 32kHz RC – LIRC
- 多种工作模式: 快速模式、低速模式、空闲模式和休眠模式
- 内部集成的振荡器, 无需外接元件
- 所有指令都可在 1~3 个指令周期内完成
- 查表指令
- 115 条功能强大的指令系统
- 16 层硬件堆栈
- 位操作指令

周边特性

- Flash 程序存储器: 32K \times 16
- 数据存储器: 2048 \times 8
- True EEPROM 存储器: 2048 \times 8
- 在线应用编程功能 – IAP
- 看门狗定时器功能
- 多达 9 个双向 I/O 口
- 4 个与 I/O 口复用的外部中断输入
- 多个定时器模块用于时间测量、捕捉输入、比较匹配输出、PWM 输出及单脉冲输出
- 双时基功能, 用于产生固定时间的中断信号
- 连续血糖监测 AFE 集成电路
 - ◆ 2 个外部通道 24-bit 分辨率的 A/D 转换器
 - ◆ 内部参考电压发生器
 - ◆ 4 个 12-bit D/A 转换器
 - ◆ 3 个运算放大器
 - ◆ 1 个比较器
 - ◆ 内部 LDO
- 串行接口模块 – SIM, 用于 SPI 或 I²C 通信
- 内部乘除法单元 – MDU
- 内建 16-bit 循环冗余校验功能 – CRC

- 低电压复位功能
- 封装类型：16/24-pin QFN

概述

该单片机是一款 A/D 型具有 8 位高性能精简指令集的 Flash 单片机，内置连续血糖监测 AFE 模块，专门为连续血糖监测产品而设计。

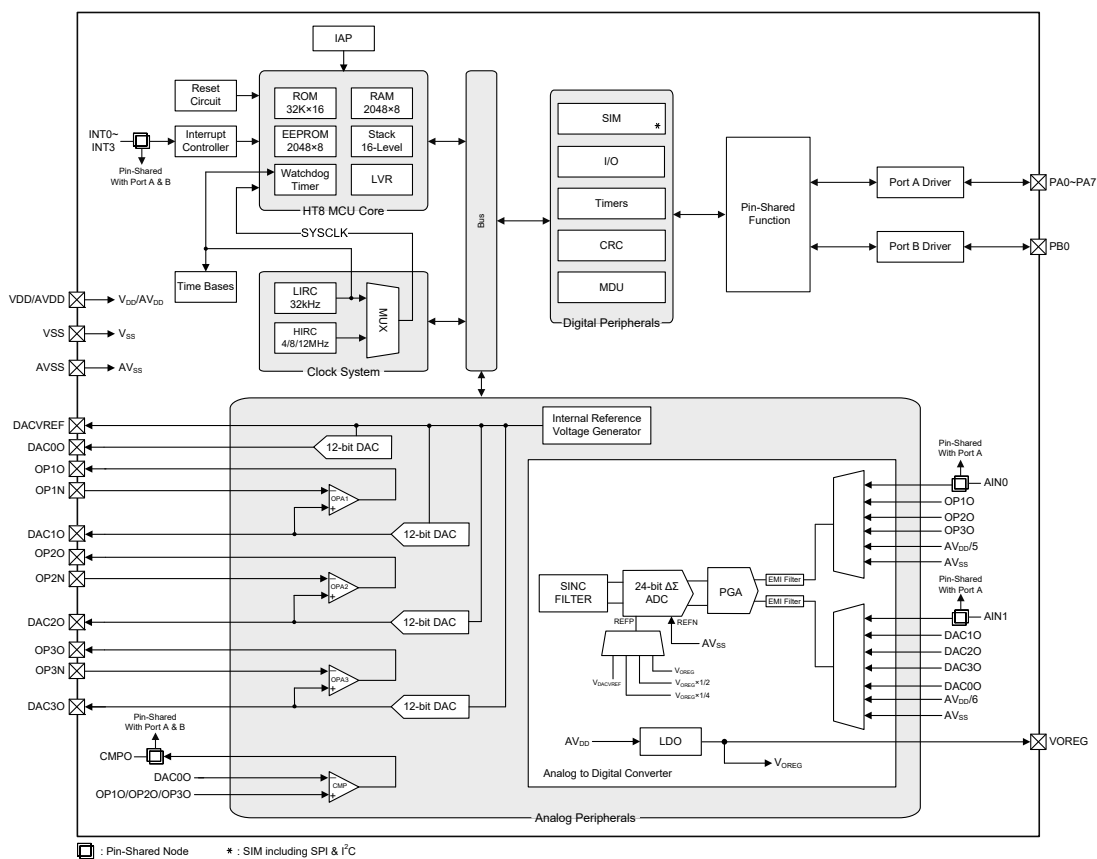
在存储器特性方面，Flash 存储器可多次编程的特性给用户提供了较大的方便。除了 Flash 程序存储器，还包含了一个 RAM 数据存储器和一个可用于存储序号、校准数据等非易失性数据的 True EEPROM 存储器。此外通过使用 IAP 功能，便于用户直接将测量的数据存储至程序存储器中或进行应用程序更新。

在模拟特性方面，该单片机包含一个多通道 24-bit Delta Sigma A/D 转换器、4 个 D/A 转换器、3 个运算放大器和 1 个比较器。该单片机还带有多个使用灵活的定时器模块，可提供定时功能、脉冲产生功能及 PWM 产生功能。内建完整的 SPI 和 I²C 接口功能，为设计者提供了一个易与外部硬件通信的接口。内部看门狗定时器、低压复位等内部保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地运行。

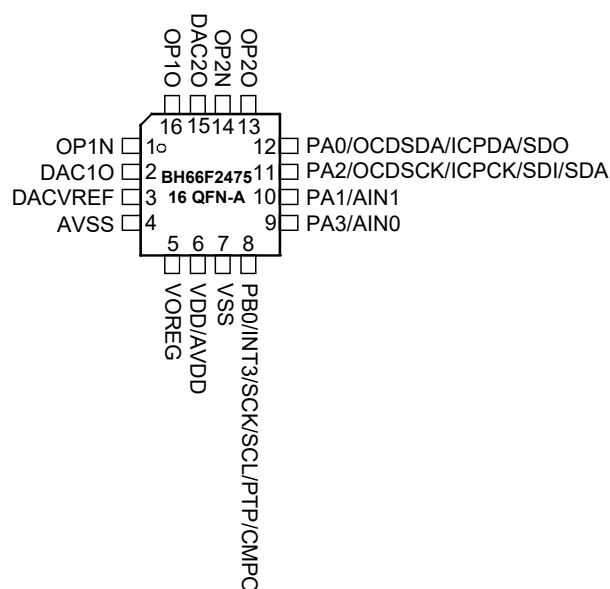
该单片机提供了丰富的内部高速和低速振荡器功能选项，完全内置的两个系统振荡器，无需外围元器件。其不同工作模式之间动态切换的能力，为用户提供了一个优化单片机操作和减少功耗的手段。

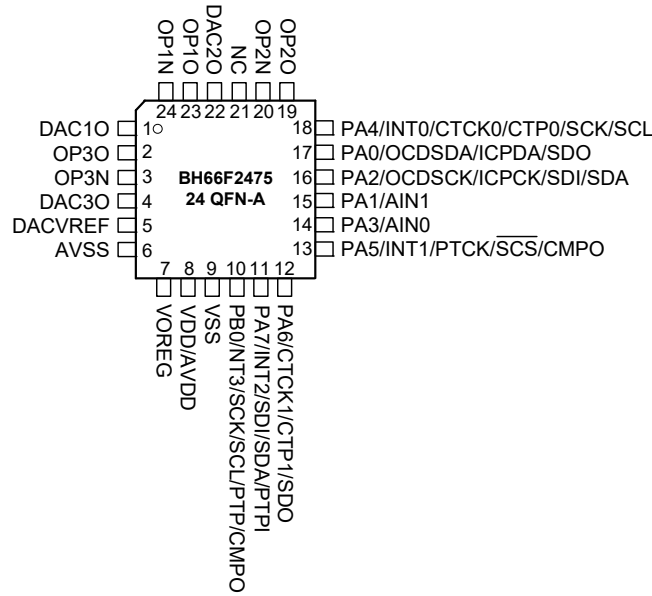
针对连续血糖监测应用，该单片机内置多个相关功能，例如内部参考电压发生器、内部 LDO、24-bit Delta Sigma A/D 转换器、12-bit D/A 转换器和运算放大器等。外加 I/O 使用灵活、16-bit MDU、16-bit CRC 功能和时基功能等其它特性，使这款单片机可以广泛应用于连续血糖监测产品。

方框图



引脚图





- 注：1. 若共用脚同时有多种输出，所需引脚共用功能通过引脚共用寄存器中相应的软件控制位控制。
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。

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。下述引脚功能表格是针对最大封装提供的引脚，对于小封装会有部分引脚未出现。

引脚名称	功能	OPT	I/T	O/T	说明
PA0/OCDSDA/ ICPDA/SDO	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
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址引脚
	ICPDA	—	ST	CMOS	ICP 数据 / 地址引脚
	SDO	PAS0	—	CMOS	SPI 数据输出
PA1/AIN1	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
	AIN1	PAS0	AN	—	A/D 转换器模拟输入
PA2/OCDSCK/ ICPCK/SDI/SDA	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
	OCDSCK	—	ST	—	OCDS 时钟引脚
	ICPCK	—	ST	—	ICP 时钟引脚
	SDI	PAS0 IFS	ST	—	SPI 数据输入
	SDA	PAS0 IFS	ST	NMOS	I ² C 数据线

引脚名称	功能	OPT	I/T	O/T	说明
PA3/AIN0	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
	AIN0	PAS0	AN	—	A/D 转换器模拟输入
PA4/INT0/CTCK0/ CTP0/SCK/SCL	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
	INT0	PAS1 INTEG INTC0	ST	—	外部中断
	CTCK0	PAS1	ST	—	CTM0 时钟输入
	CTP0	PAS1	—	CMOS	CTM0 输出
	SCK	PAS1 IFS	ST	CMOS	SPI 串行时钟
	SCL	PAS1 IFS	ST	NMOS	I ² C 时钟线
PA5/INT1/PTCK/ SCS/CMPO	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
	INT1	PAS1 INTEG INTC0	ST	—	外部中断
	PTCK	PAS1	ST	—	PTM 时钟输入或捕捉输入
	SCS	PAS1	ST	CMOS	SPI 从机选择
	CMPO	PAS1	—	CMOS	比较器输出
PA6/CTCK1/CTP1/ SDO	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
	CTCK1	PAS1	ST	—	CTM1 时钟输入
	CTP1	PAS1	—	CMOS	CTM1 输出
	SDO	PAS1	—	CMOS	SPI 数据输出
PA7/INT2/SDI/SDA/ PTPI	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
	INT2	PAS1 INTEG INTC1	ST	—	外部中断
	SDI	PAS1 IFS	ST	—	SPI 数据输入
	SDA	PAS1 IFS	ST	NMOS	I ² C 数据线
	PTPI	PAS1	ST	—	PTM 捕捉输入

引脚名称	功能	OPT	I/T	O/T	说明
PB0/INT3/SCK/SCL/ PTP/CMPO	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	INT3	PBS0 INTEG INTC2	ST	—	外部中断
	SCK	PBS0 IFS	ST	CMOS	SPI 串行时钟
	SCL	PBS0 IFS	ST	NMOS	I ² C 时钟线
	PTP	PBS0	—	CMOS	PTM 输出
	CMPO	PBS0	—	CMOS	比较器输出
VOREG	VOREG	—	—	AN	LDO 输出
		—	AN	—	A/D 转换器、PGA 正电源供电
OP1N	OP1N	—	AN	—	OPA1 负输入端
OP1O	OP1O	—	—	AN	OPA1 输出
OP2N	OP2N	—	AN	—	OPA2 负输入端
OP2O	OP2O	—	—	AN	OPA2 输出
OP3N	OP3N	—	AN	—	OPA3 负输入端
OP3O	OP3O	—	—	AN	OPA3 输出
DACVREF	DACVREF	—	—	AN	D/A 转换器参考电压
DAC1O	DAC1O	—	—	AN	D/A 转换器输出
DAC2O	DAC2O	—	—	AN	D/A 转换器输出
DAC3O	DAC3O	—	—	AN	D/A 转换器输出
VDD/AVDD	VDD	—	PWR	—	数字正电源
	AVDD	—	PWR	—	模拟正电源
VSS	VSS	—	PWR	—	数字负电源
AVSS	AVSS	—	PWR	—	模拟负电源
NC	NC	—	—	—	未连接

注：I/T：输入类型；

OPT：通过寄存器选项来配置；

CMOS：CMOS 输出；

AN：模拟信号；

O/T：输出类型；

ST：施密特触发输入；

NMOS：NMOS 输出；

PWR：电源；

极限参数

电源供应电压	$V_{SS}-0.3V\sim 6.0V$
端口输入电压	$V_{SS}-0.3V\sim V_{DD}+0.3V$
储存温度	$-60^{\circ}C\sim 150^{\circ}C$
工作温度	$-40^{\circ}C\sim 85^{\circ}C$
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、温度和程序指令等等。

工作电压特性

$T_a=-40^{\circ}C\sim 85^{\circ}C$

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 – HIRC	$f_{SYS}=4MHz$	2.2	—	5.5	V
		$f_{SYS}=8MHz$	2.2	—	5.5	
		$f_{SYS}=12MHz$	2.7	—	5.5	
	工作电压 – LIRC	$f_{SYS}=32kHz$	2.2	—	5.5	V

工作电流特性

$T_a=-40^{\circ}C\sim 85^{\circ}C$

符号	工作模式	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{DD}	低速模式 – LIRC	2.2V	$f_{SYS}=32kHz$	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	快速模式 – HIRC	2.2V	$f_{SYS}=4MHz$	—	0.3	0.5	mA
		3V		—	0.4	0.6	
		5V		—	0.8	1.2	
		2.2V	$f_{SYS}=8MHz$	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	
		2.7V	$f_{SYS}=12MHz$	—	1.0	1.4	mA
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	

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. 所有工作电流数值通过一个连续的 NOP 指令循环程序测得。

待机电流特性

Ta=25°C, 除非另有说明

符号	待机模式	测试条件		最小	典型	最大	最大 @85°C	单位
		V _{DD}	条件					
I _{STB}	休眠模式	2.2V	WDT off	—	0.45	0.80	7.00	μA
		3V		—	0.45	0.90	8.00	
		5V		—	0.5	2.0	10.0	
		2.2V	WDT on	—	1.5	3.0	3.6	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3	5	6	
	空闲模式 0 – LIRC	2.2V	f _{SUB} on	—	3	5	6	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	空闲模式 1 – HIRC	2.2V	f _{SUB} on, f _{SYS} =4MHz	—	144	200	240	μA
		3V		—	180	250	300	
		5V		—	400	600	720	
		2.2V	f _{SUB} on, f _{SYS} =8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	
		2.7V	f _{SUB} on, f _{SYS} =12MHz	—	432	600	720	μA
		3V		—	540	750	900	
		5V		—	800	1200	1440	

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。

内部高速振荡器 – HIRC – 频率精度

程序烧录时，烧录器会调整 HIRC 振荡器使其工作在用户选择的 HIRC 频率和工作电压 (3V 或 5V) 条件下。

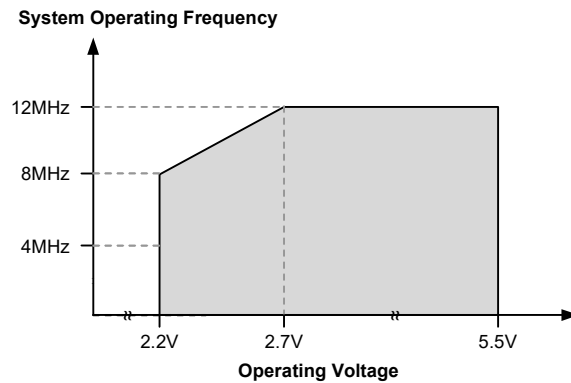
符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	温度				
f _{HIRC}	通过烧录器调整后的 4MHz HIRC 频率	3V/5V	25°C	-1%	4	+1%	MHz
			-40°C~85°C	-2%	4	+2%	
		2.2V~5.5V	25°C	-2.5%	4	+2.5%	
			-40°C~85°C	-3%	4	+3%	
	通过烧录器调整后的 8MHz HIRC 频率	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	
	通过烧录器调整后的 12MHz HIRC 频率	5V	25°C	-1%	12	+1%	MHz
			-40°C~85°C	-2%	12	+2%	
		2.7V~5.5V	25°C	-2.5%	12	+2.5%	
			-40°C~85°C	-3%	12	+3%	

- 注：1. 烧录器可在 3V/5V 这两个可选的固定电压下对 HIRC 频率进行调整，在此提供 V_{DD}=3V/5V 时的参数值。
2. 3V/5V 表格列下面提供的是全压条件下的参数值。当应用电压范围是 2.2V~3.6V 时，建议烧录器电压固定在 3V；当应用电压范围是 3.6V~5.5V 时，建议烧录器电压固定在 5V。
3. 表格中提供的最小和最大误差值仅在对应的烧录器调整频率下有效。当烧录器已将 HIRC 调整为某一固定频率，此后再通过程序中振荡器控制位将其频率改为其它值时，频率误差范围将增加到 ±20%。

内部低速振荡器电气特性 – LIRC

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	温度				
f _{LIRC}	振荡器频率	2.2V~5.5V	-40°C~85°C	-10%	32	+10%	kHz
t _{START}	LIRC 启动时间	—	-40°C~85°C	—	—	100	μs

工作频率电气特性曲线图



系统上电时间电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
t _{SST}	系统启动时间 (从 f _{SYS} off 的状态下唤醒)	—	f _{SYS} =f _H ~f _H /64, f _H =f _{HIRC}	—	16	—	t _{HIRC}
		—	f _{SYS} =f _{SUB} =f _{LIRC}	—	2	—	t _{LIRC}
	系统启动时间 (从 f _{SYS} on 的状态下唤醒)	—	f _{SYS} =f _H ~f _H /64, f _H =f _{HIRC}	—	2	—	t _H
		—	f _{SYS} =f _{SUB} =f _{LIRC}	—	2	—	t _{SUB}
	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} =5 V/ms	14	16	18	ms
	系统复位延迟时间 (LVRC/WDTC/RSTC 软件复位)	—	—				
	系统复位延迟时间 (WDT 溢出)	—	—				
t _{SRESET}	软件复位最小脉宽	—	—	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. 若 LIRC 被选择作为系统时钟源且在休眠模式下 LIRC 关闭，则上面表格中对应 t_{SST} 数值还需加上 LIRC 频率表格里提供的 LIRC 启动时间 t_{START}。
4. 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 / 输出电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{IL}	I/O 口低电平输入电压	5V	—	0	—	1.5	V
		—	—	0	—	0.2V _{DD}	
V _{IH}	I/O 口高电平输入电压	5V	—	3.5	—	5.0	V
		—	—	0.8V _{DD}	—	V _{DD}	
I _{OL}	I/O 口灌电流	3V	V _{OL} =0.1V _{DD}	16	32	—	mA
		5V		32	65	—	
I _{OH}	I/O 口源电流	3V	V _{OH} =0.9V _{DD}	-4	-8	—	mA
		5V		-8	-16	—	
R _{PH}	I/O 口上拉电阻 ⁽¹⁾	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I _{LEAK}	输入漏电流	3V	V _{IN} =V _{DD} or V _{IN} =V _{SS}	—	—	±1	μA
		5V		—	—	±1	
t _{INT}	中断引脚最小输入脉宽	—	—	10	—	—	μs
t _{TPI}	PTPI 输入最小输入脉宽	—	—	0.3	—	—	μs
t _{TCK}	TM 时钟输入最小脉宽	—	—	0.3	—	—	μs
f _{TMCLK}	PTM 计数器最大时钟源频率	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：将其接地并使能输入引脚的上拉电阻选项，然后在特定电源电压下测量该引脚上的电流，在此基础上测量上拉电阻上的分压从而得到此上拉电阻值。

2. t_{TMCLK}=1/f_{TMCLK}

若 PTCAPTS=0, t_{CPW}=max(2×t_{TMCLK}, t_{TPI})

若 PTCAPTS=1, t_{CPW}=max(2×t_{TMCLK}, t_{TCK})

例 1：若 PTnCAPTS=0, f_{TMCLK}=8MHz, t_{TPI}=0.3μs, 则 t_{CPW}=max(0.25μs, 0.3μs)=0.3μs

例 2：若 PTnCAPTS=1, f_{TMCLK}=8MHz, t_{TCK}=0.3μs, 则 t_{CPW}=max(0.25μs, 0.3μs)=0.3μs

例 3：若 PTnCAPTS=0, f_{TMCLK}=4MHz, t_{TPI}=0.3μs, 则 t_{CPW}=max(0.5μs, 0.3μs)=0.5μs

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}	擦除时间	—	FWERTS=0	—	3.2	3.9	ms
		—	FWERTS=1	—	3.7	4.5	
t _{FWR}	写时间	—	FWERTS=0	—	2.2	2.7	ms
		—	FWERTS=1	—	3.0	3.6	
E _P	存储单元耐受性	—	—	100K	—	—	E/W
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	—	—	32	—	64	μs

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
数据 EEPROM 存储器							
V _{DD}	读 / 写工作电压 V _{DD}	—	—	2.2	—	5.5	V
t _{EERD}	读时间	—	—	—	—	4	t _{sys}
t _{EEER}	擦除时间	—	EWERTS=0	—	3.2	3.9	ms
		—	EWERTS=1	—	3.7	4.5	
t _{EEWR}	写时间 (字节模式)	—	EWERTS=0	—	5.4	6.6	ms
		—	EWERTS=1	—	6.7	8.1	
	写时间 (页模式)	—	EWERTS=0	—	2.2	2.7	
		—	EWERTS=1	—	3.0	3.6	
E _P	存储单元耐受性	—	—	100K	—	—	E/W
t _{RETD}	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. “E/W”表示擦 / 写次数。

2. 在计算从空闲 / 休眠模式唤醒的系统总启动时间时，还需加上 ROM 激活时间 t_{ACTV}。

LVR 电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{DD}	工作电压	—	—	2.2	—	5.5	V
V _{LVR}	低电压复位电压	—	LVR 使能, 电压选择 2.1V	-3%	2.1	+3%	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 3.8V		3.8		
I _{LVR}	工作电流	3V	LVR 使能, V _{LVR} =2.1V	—	—	10	μA
		5V		—	10	15	μA
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	
I _{LVR}	LVR 使能的额外电流	5V	—	—	—	14	μA

模拟前端电路特性

LDO 电气特性

Ta=0°C~50°C, 除非另有说明

LDO 测试条件: MCU 进入 SLEEP 模式, 其他功能除能

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
AV _{DD}	LDO 输入电压	—	—	2.2	—	5.5	V
I _Q	LDO 静态电流	—	LDOVS[1:0]=00B, AV _{DD} =3.6V, 无负载	—	25	—	μA
V _{OUT_LDO}	LDO 输出电压	—	LDOVS[1:0]=00B, AV _{DD} =3.6V, I _{LOAD} =0.1mA	-5%	2.4	+5%	V
		—	LDOVS[1:0]=01B, AV _{DD} =3.6V, I _{LOAD} =0.1mA		2.6		
		—	LDOVS[1:0]=10B, AV _{DD} =3.6V, I _{LOAD} =0.1mA		2.2		
		—	LDOVS[1:0]=11B, AV _{DD} =3.6V, I _{LOAD} =0.1mA		2.3		
ΔV _{LOAD}	LDO 负载调整率	—	LDOVS[1:0]=00B, AV _{DD} =V _{OUT_LDO} +0.2V, 0mA≤I _{LOAD} ≤10mA	—	0.105	0.210	%/mA
V _{DROP_LDO}	LDO 压降	—	LDOVS[1:0]=00B, AV _{DD} =3.6V, I _{LOAD} =10mA, ΔV _{OUT_LDO} =2%	—	—	220	mV
		—	LDOVS[1:0]=01B, AV _{DD} =3.6V, I _{LOAD} =10mA, ΔV _{OUT_LDO} =2%	—	—	200	
		—	LDOVS[1:0]=10B, AV _{DD} =3.6V, I _{LOAD} =10mA, ΔV _{OUT_LDO} =2%	—	—	240	
		—	LDOVS[1:0]=11B, AV _{DD} =3.6V, I _{LOAD} =10mA, ΔV _{OUT_LDO} =2%	—	—	230	
TC _{LDO}	LDO 温度系数	—	Ta=0°C~50°C, LDOVS[1:0]=00B, AV _{DD} =3.6V, I _{LOAD} =100μA	—	—	±200	ppm/°C
ΔV _{LINE_LDO}	LDO 线性调整率	—	LDOVS[1:0]=00B, 2.6V≤AV _{DD} ≤5.5V, I _{LOAD} =100μA	—	—	1.0	%/V
		—	LDOVS[1:0]=00B, 2.6V≤AV _{DD} ≤3.6V, I _{LOAD} =100μA	—	—	0.7	%/V

OPA 电气特性

Ta=0°C~50°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{OREG}	OPA 电源电压	—	—	2.2	—	5.5	V
I _{OPA}	使用 OPA 的额外电流	—	—	—	0.4	—	μA
V _{OS}	输入失调电压	—	—	-3	—	3	mV
I _{OS}	输入失调电流	—	—	—	1	—	pA
I _B	输入偏置电流	—	—	—	1	—	pA
V _{CM_OPA}	OPA 共模电压范围	—	—	V _{SS}	—	V _{OREG}	V
PSRR	电源电压抑制比	—	—	—	80	—	dB
CMRR	共模抑制比	—	—	—	80	—	dB

内部参考电压电气特性

Ta=0°C~50°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 _{IREF}	内部参考电压	3V	IREFEN=1, PVREF=10000000B	-3%	1.25	+3%	V
I _{IREF}	使用内部参考电压的额外电流	—	IREFEN=1	—	1.8	—	μA
TC _{IREF}	内部参考电压温度系数	3V	Ta=0°C~50°C	—	±40	—	ppm/ °C

12-bit D/A 转换器电气特性

Ta=0°C~50°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{OREG}	DAC 电源电压	—	—	2.2	—	5.5	V
I _{DAC}	使用 DAC 的额外电流	—	—	—	1.2	—	μA
DNL	非线性差分误差	3V	DACVRS[1:0]=00B	—	—	±8	LSB
INL	非线性积分误差	3V	DACVRS[1:0]=00B	—	—	±20	LSB
R _o	R2R 输出电阻	3V	—	—	5	—	kΩ
V _{DACO}	输出电压范围	—	—	0.00	—	1.00	V _{DACVREF}

A/D 转换器电气特性

 $T_a=0^{\circ}\text{C}\sim 50^{\circ}\text{C}$

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{OREG}	A/D 转换器和 PGA 电源电压	—	LDOEN=0	2.2	—	2.6	V
		—	LDOEN=1	2.2	—	2.6	V
I_{ADC}	A/D 转换器使能的额外电流	—	—	—	400	—	μA
I_{ADSTB}	待机电流	—	MCU 进入休眠模式, 无负载	—	—	1	μA
N_R	分辨率	—	—	—	—	24	Bit
INL	非线性积分误差	—	$V_{OREG}=2.6\text{V}$, $V_{REF}=1.25\text{V}$, $\Delta SI=\pm 450\text{mV}$, PGAGN=1	—	± 50	—	ppm
NFB	无噪音位	—	$f_{MCLK}=4\text{MHz}$, FLMS[2:0]=000B, $V_{OREG}=2.6\text{V}$, $V_{REF}=1.25\text{V}$ PGAGN=128, OSR=16384	—	16	—	Bit
ENOB	有效位	—	$f_{MCLK}=4\text{MHz}$, FLMS[2:0]=000B, $V_{OREG}=2.6\text{V}$, $V_{REF}=1.25\text{V}$ PGAGN=128, OSR=16384	—	18.7	—	Bit
f_{ADCK}	A/D 转换器时钟频率	—	—	40.0	409.6	440.0	kHz
f_{ADO}	A/D 转换器输出数据传输速率	—	$f_{MCLK}=4\text{MHz}$, FLMS[2:0]=000B SINC3[1:0]=11B	4	—	1042	Hz
		—	$f_{MCLK}=4\text{MHz}$, FLMS[2:0]=010B SINC3[1:0]=11B	10	—	2604	Hz
V_{REFP}	参考输入电压	—	—	$V_{REFN}+0.8$	—	V_{OREG}	V
V_{REFN}		—	—	0	—	$V_{REFP}-0.8$	V
V_{REF}		—	$V_{REF}=(V_{REFP}-V_{REFN})$	0.8	—	1.75	V

有效位数 – ENOB

 $AV_{DD}=V_{OREG}=2.4\text{V}$, $V_{REF}=1.25\text{V}$, $f_{ADCK}=133\text{kHz}$

数据传输速率 (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
4	21.3	21.3	21.2	21.1	21.0	20.4	19.8	18.4
8	20.9	20.9	20.9	20.8	20.5	20.1	19.0	18.3
16	19.2	19.2	19.1	19.0	18.9	18.7	18.9	18.0
33	18.9	18.9	18.9	18.9	18.7	18.6	18.3	17.4
65	18.8	18.8	18.8	18.7	18.6	18.3	17.7	16.9
130	18.5	18.5	18.5	18.4	18.4	18.0	17.4	16.5
260	18.4	18.3	18.3	18.3	18.1	17.5	16.8	15.9
521	17.6	17.6	17.5	17.5	17.5	17.2	16.3	15.4

$AV_{DD}=V_{OREG}=2.4V$, $V_{REF}=1.25V$, $f_{ADCK}=333kHz$

数据传输速率 (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
10	21.0	20.9	20.9	20.8	20.6	19.9	19.2	18.2
20	20.7	20.6	20.6	20.5	20.2	19.6	18.7	17.6
41	20.5	20.5	20.5	20.3	19.7	18.5	18.0	17.4
81	19.4	19.3	19.3	19.1	18.8	18.2	17.6	16.1
163	18.6	18.6	18.2	17.8	17.0	16.1	15.1	14.1
326	18.2	18.0	17.7	17.3	16.4	15.5	14.5	13.5
651	18.1	18.1	18.0	17.9	17.5	17.1	16.2	15.3
1302	17.8	17.8	17.6	17.6	17.2	16.6	15.8	15.0

PGA 转换器电气特性

$T_a=0^{\circ}C\sim 50^{\circ}C$

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
ΔD_I	差分输入电压范围	—	Gain=PGAGN \times ADGN	$-V_{REF}$ /Gain	—	$+V_{REF}$ /Gain	V

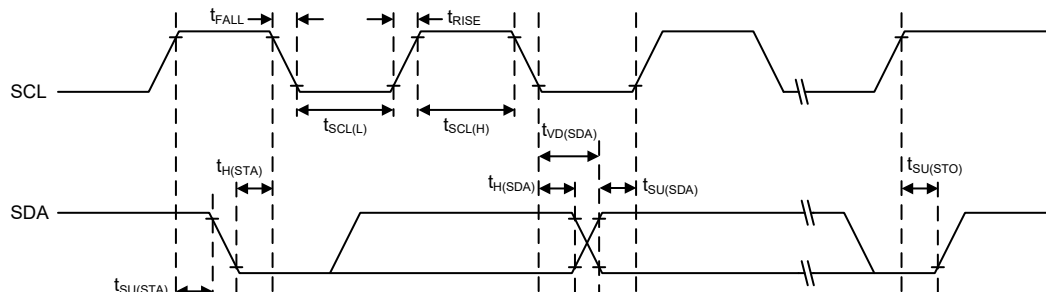
I²C 电气特性

$T_a=25^{\circ}C$

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{I2C}	I ² C 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	—	—	
	I ² C 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	3V/5V	标准模式	—	—	100	kHz
			快速模式	—	—	400	
$t_{SCL(H)}$	SCL 时钟高电平时间	3V/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	3V/5V	标准模式	3.5	—	—	μs
			快速模式	0.9	—	—	
t_{FALL}	SCL 和 SDA 下降沿时间	3V/5V	标准模式	—	—	1.3	μs
			快速模式	—	—	0.34	
t_{RISE}	SCL 和 SDA 上升沿时间	3V/5V	标准模式	—	—	1.3	μs
			快速模式	—	—	0.34	
$t_{SU(SDA)}$	SDA 数据建立时间	3V/5V	标准模式	0.25	—	—	μs
			快速模式	0.1	—	—	
$t_{H(SDA)}$	SDA 数据保持时间	3V/5V	—	0.1	—	—	μs

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
t _{VD(SDA)}	SDA 数据有效时间	3V/5V	—	—	—	0.6	μs
t _{SU(STA)}	START 条件建立时间	3V/5V	标准模式	3.5	—	—	μs
			快速模式	0.6	—	—	
t _{H(STA)}	START 条件保持时间	3V/5V	—	0.6	—	—	μs
t _{SU(STO)}	STOP 条件建立时间	3V/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。

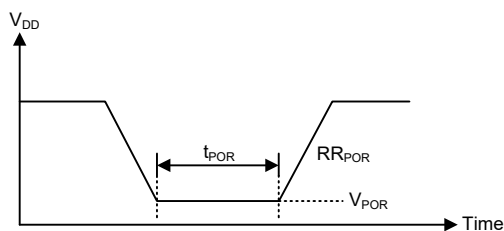


I²C 时序图

上电复位特性

T_a=-40°C~85°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



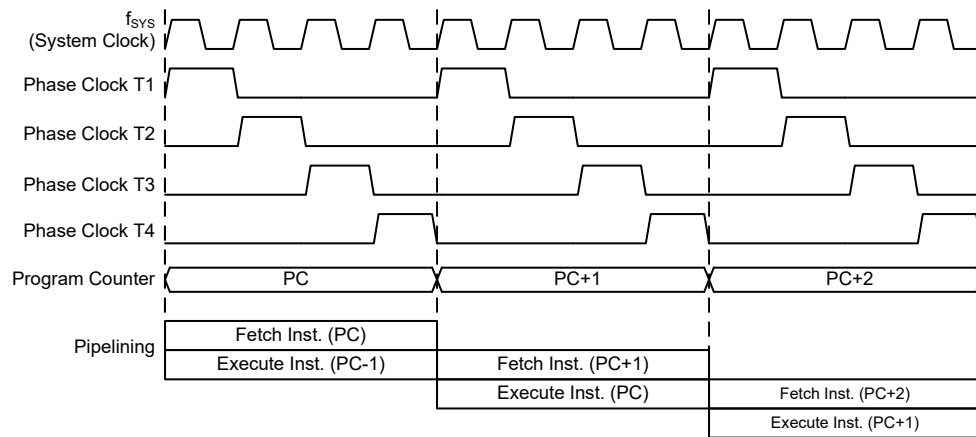
系统结构

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，指令的获取和执行同时进行，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。

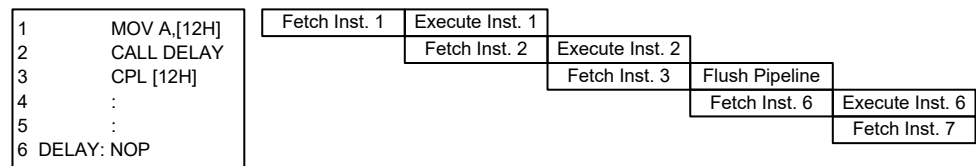
时序和流水线结构

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。由于单片机存储器容量大于 8K，程序存储器地址可能位于某一程序存储区，可通过程序存储区指针的 PBP1~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)
PBP1~PBP0, PC12~PC8	PCL7~PCL0

程序计数器

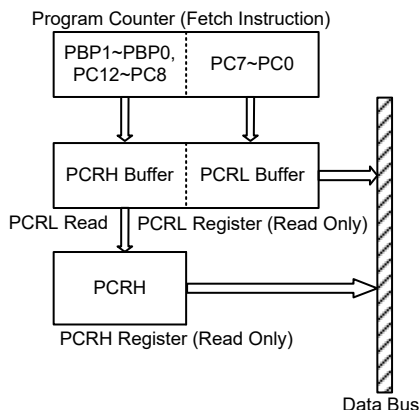
程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。程序计数器的低字节可由程序直接进行读取，PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

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:** 低字节寄存器 bit 7 ~ bit 0

● PCRH 寄存器

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

Bit 7 未定义，读为“0”

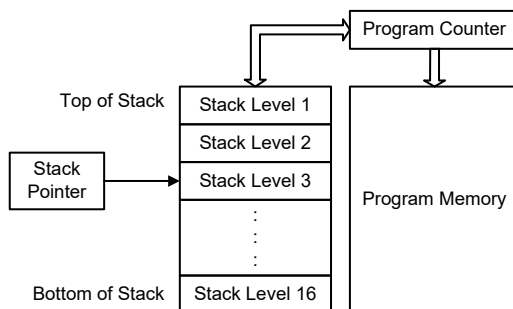
Bit 6~0 **D14~D8:** 高字节寄存器 bit 6 ~ bit 0

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 16 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[3:0]** 加以指示。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	—	—	—	D3	D2	D1	D0
R/W	R/W	—	—	—	R	R	R	R
POR	0	—	—	—	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 时, 或当堆栈为空再次执行 RET 指令时, 都会将 OSF 位置为 1。OSF 位只能由软件复位。

Bit 6~4 未定义, 读为 “0”

Bit 3~0 **D3~D0**: 堆栈指针寄存器 bit 3 ~ bit 0

下面举例说明当发生程序分支跳转时堆栈指针及溢出标志位是如何变化的。

- (1) 连续执行 17 次 CALL 指令, 期间未执行 RET 指令, STKPTR[3:0] 和 OSF 位内容变化如下:

CALL 执行次数	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
STKPTR[3:0] 值	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1
OSF 值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

- (2) 当 OSF 位为 1 时, 如果不清除 OSF 位, 无论执行多少次 RET 指令, OSF 位都为 1。

- (3) 当堆栈为空时, 连续执行 16 次 RET 指令, STKPTR[3:0] 和 OSF 位内容变化如下:

RET 执行次数	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
STKPTR[3:0] 值	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSF 值	0	1	1	1	1	1	1	1	1	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、借位或其它状态的变化, 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, LRR, LRRCA, LRRC, LRLA, LRL, LRLCA, LRLC

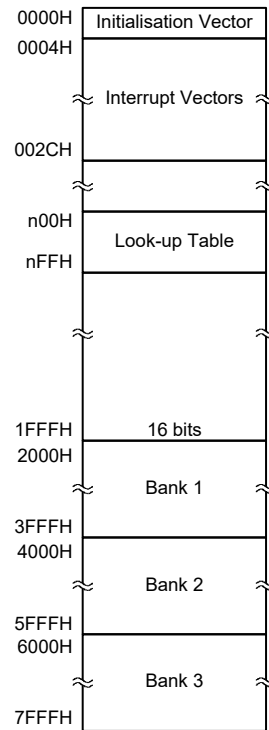
- 递增和递减：
INCA, INC, DECA, DEC
LINCA, LINC, LDECA, LDEC
- 分支判断：
JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI
LSNZ, LSZ, LSZA, LSIZ, LSIZA, LSDZ, LSDZA

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 32K×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

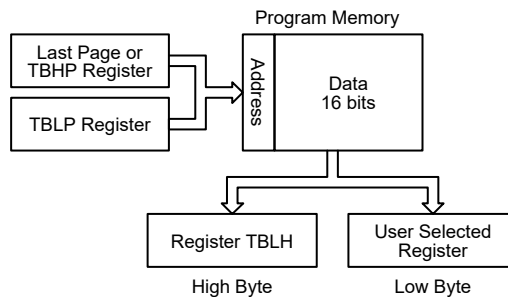
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 0000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

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 ROM Bank 3，指向的地址是 32K 程序存储器中最后一页的起始地址。表格指针低字节寄存器的初始值设为 06H，这可保证从数据表格读取的第一笔数据位于程序存储器地址 7F06H，即最后一页起始地址后的第六个地址。值得注意的是，假如“TABRD [m]”或“LTABRD [m]”指令被使用，则表格指针指向 TBHP 和 TBLP 所指定的地址。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 / 写寄存器，且能重新储存，若主程序和中断服务程序都使用表格读取指令，应该注意它的保护。使用表格读取指令，中断服务程序可能会改变 TBLH 的值，若随后在主程序中再次使用这个值，则会发生错误，因此建议避免同时使用表格读取指令。然而在某些情况下，如果同时使用表格读取指令是不可避免的，则在执行任何主程序的表格读取指令前，中断应该先除能，另外要注意的是所有与表格相关的指令，都需要两个指令周期去完成操作。

表格读取程序范例

rombank 1 code1

```

ds .section 'data'
tempreg1 db?      ; temporary register #1
tempreg2 db?      ; temporary register #2
code0 .section 'code'
mov a,06H          ; initialise table pointer - note that this address is
                   ; referenced
mov tblp,a         ; to the last page or the page that tbhp pointed
mov a,7FH          ; 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 "7F06H" 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 "7F05H" transferred to
                    ; tempreg2 and TBLH
                    ; in this example the data "1AH" is transferred to
                    ; tempreg1 and data "0FH" to tempreg2
                    ; the value "00H" will be transferred to the high byte
                    ; register TBLH
:
:
code1 .section 'code'
org 1F00H            ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

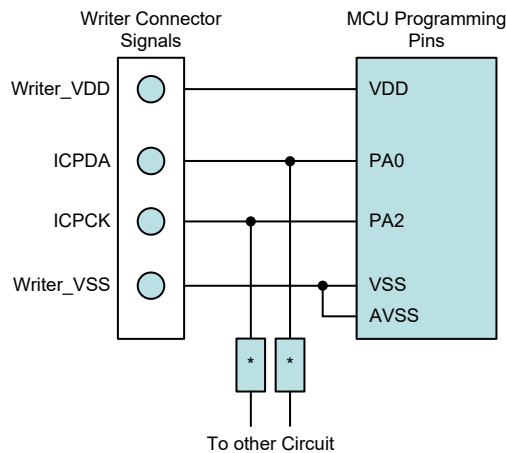
在线烧录 – ICP

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，在无需拔出再重新插入芯片的情况下方便地保持程序为最新版。

Holtek 烧录器引脚名称	MCU 在线烧录引脚名称	引脚描述
ICPDA	PA0	串行数据 / 地址烧录
ICPCK	PA2	时钟烧录
VDD	VDD	电源
VSS	VSS&AVSS	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。

烧录过程中，用户必须确保 ICPDA 和 ICPCK 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

该单片机提供片上调试功能 (OCDS) 用于开发过程中的单片机调试。用户可将 OCDSDA 和 OCDSCCK 引脚连接至 Holtek HT-IDE 开发工具，从而实现单片机的仿真。OCDSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCCK 引脚为 OCDS 时钟输入脚。当用户用 OCDS 功能进行调试时，单片机 OCDSDA 和 OCDSCCK 引脚上的其它共用功能无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。关于 OCDS 功能的详细描述，请参考“Holtek e-Link for 8-bit MCU OCDS 用户手册”文件。

Holtek e-Link 引脚名称	MCU 芯片引脚名称	功能
OCDSDA	PA0	片上调试串行数据 / 地址输入 / 输出
OCDSCCK	PA2	片上调试时钟输入
VDD	VDD	电源
VSS	VSS&AVSS	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。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 / 写大小

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 64 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。

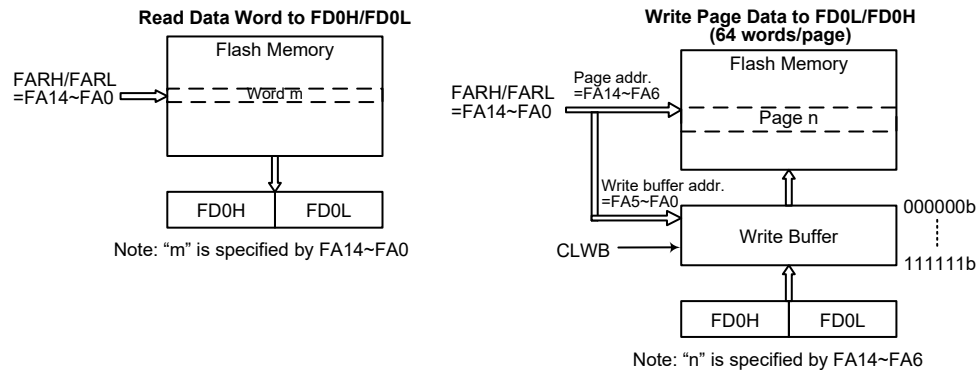
读出操作是通过一个特定的读出程序来执行的。FRDEN 位用于使能读出功能，由应用程序设置 FRD 位来启动读出程序，并指示读出操作的状态。当读出操作结束后该位将由硬件清零。

操作	格式
擦除	64 字 / 页
写入	64 字 / 次
读出	1 字 / 次
注：页大小 = 写入缓冲器大小 = 64 字	

IAP 操作格式

页	FARH	FARL [7:6]	FARL [5:0]
0	0000 0000	00	标记地址
1	0000 0000	01	
2	0000 0000	10	
3	0000 0000	11	
4	0000 0001	00	
:	:	:	
:	:	:	
510	0111 1110	10	
511	0111 1111	11	

页序号和地址选择



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每页 64 字。写入缓冲器的地址与存储器地址位 FA14~FA6 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 64 字的页为 111111b，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、四对 16 位数据寄存器和三个控制寄存器，这些寄存器都位于 Sector 1。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，即地址寄存器 FARL 和 FARH，数据寄存器 FDnL 和 FDnH(n=0~3)，控制寄存器 FC0、FC1 和 FC2。由于这些寄存器都位于 Sector 1 中，可通过扩展指令直接被访问，或者通过存储器指针 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	—	FA14	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，此位不可通过应用程序置位，对此位直接写“1”不会使能擦 / 写功能。此位可用于指示 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
其他值: 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: 不启动 Flash 存储器写入程序或 Flash 存储器写入程序已完成
1: 启动 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: 除能
1: 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: 不启动 Flash 存储器读出程序或 Flash 存储器读出程序已完成
1: 启动 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. 确保 f_{SUB} 时钟在执行擦或写动作前已稳定。
3. 当读、擦或写动作成功启动后, CPU 相关操作将停止。
4. 确保读、擦或写动作成功完成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**: 擦除时间和写入时间选择位
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: 不启动清除写入缓冲器或清除程序已完成
1: 启动清除写入缓冲器程序
此位由软件置“1”，当清除写入缓冲器程序完成后由硬件清零。

● 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

● **sFARH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	FA14	FA13	FA12	FA11	FA10	FA9	FA8
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 6~0 **FA14~FA8**: Flash 存储器地址 bit 14 ~ 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

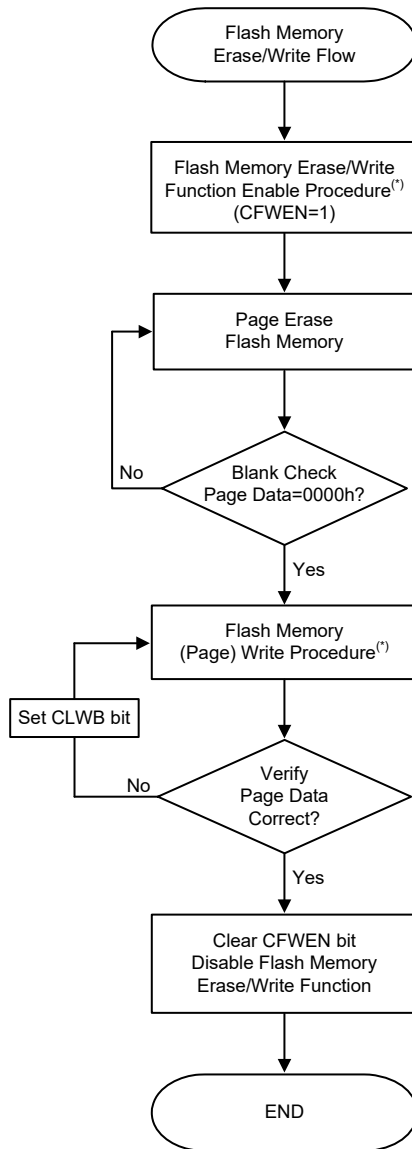
Flash 存储器擦 / 写流程

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程序开发，以确保 IAP 功能擦 / 写 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，此时才可执行擦 / 写 Flash 存储器操作。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 111111b，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 位来解除“Flash 存储器擦/写使能模式”。



Flash 存储器擦 / 写流程

注：“Flash 存储器擦 / 写使能程序”和“Flash 存储器写入程序”详见后续章节。

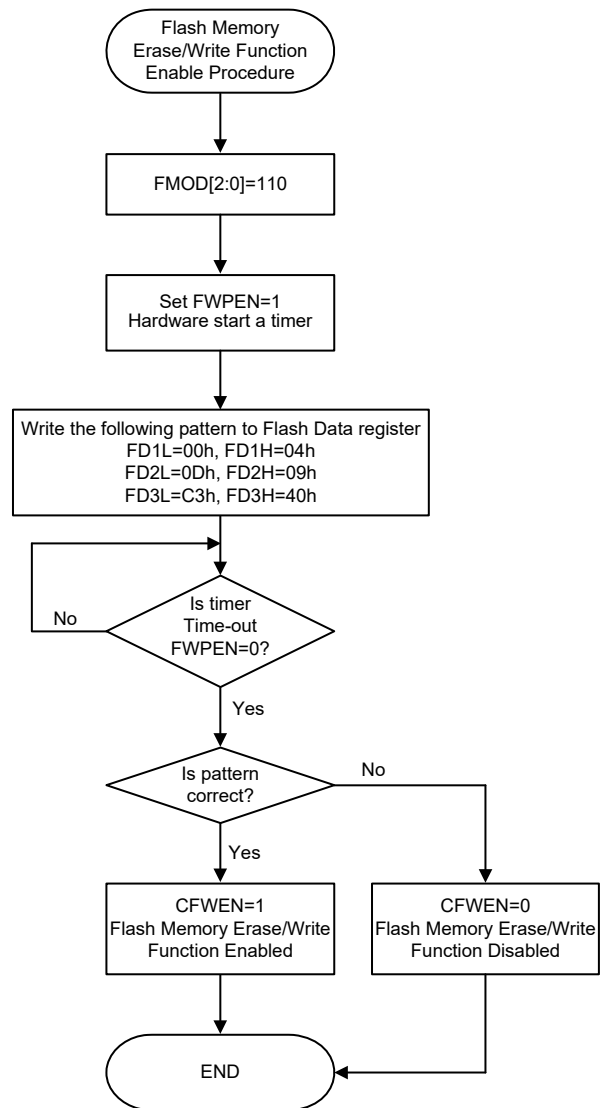
Flash 存储器擦 / 写使能程序

Flash 存储器擦 / 写使能模式是专门为保护 Flash 存储器内容不被轻易地修改而设计的。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 / 写使能程序说明

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. 使用者必须在 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 / 写使能程序

Flash 存储器写入步骤

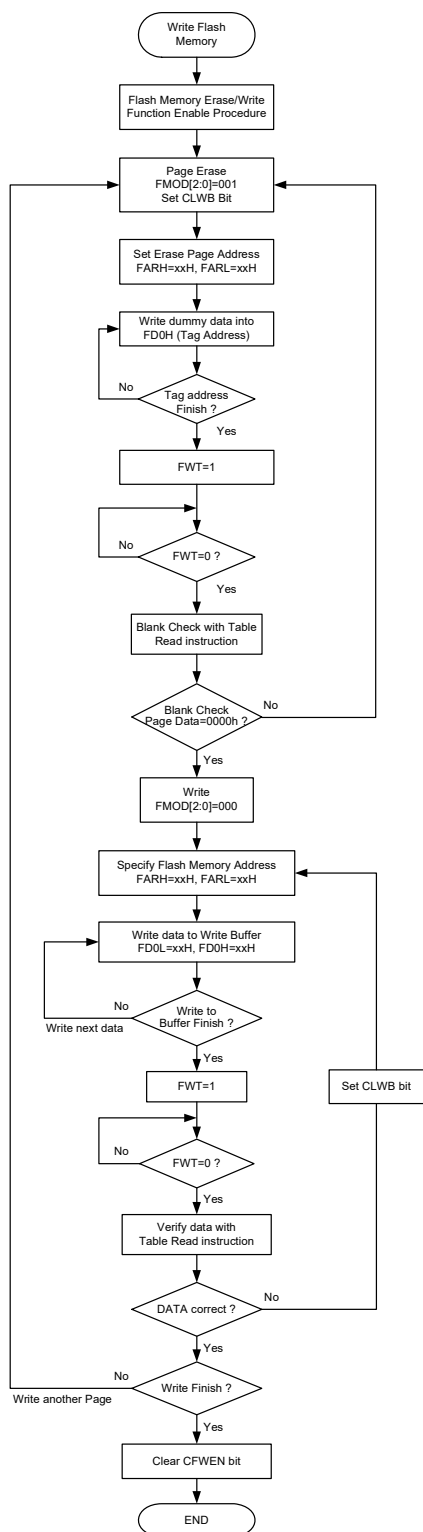
当 Flash 擦 / 写功能成功使能后, CFWEN 位会被硬件置高, 此时要写入 Flash 存储器的数据才能加载到写入缓冲器。在开始写入程序之前, 应先正确配置 IAP 控制寄存器, 将所选的 Flash 存储器页的数据擦除。

写入缓冲器的大小为每页 64 个字, 其地址与 FA14~FA6 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 64 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”, 要填入第二笔数据时, 可不用再指定地址 FARL 和 FARH。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, 并设定 CLWB 位为“1”, 清除“写入缓冲器”。设定 FWT 位为“1”, 擦除 FARH 和 FARL 指定且标记地址的目标页, 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 64 个字。
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

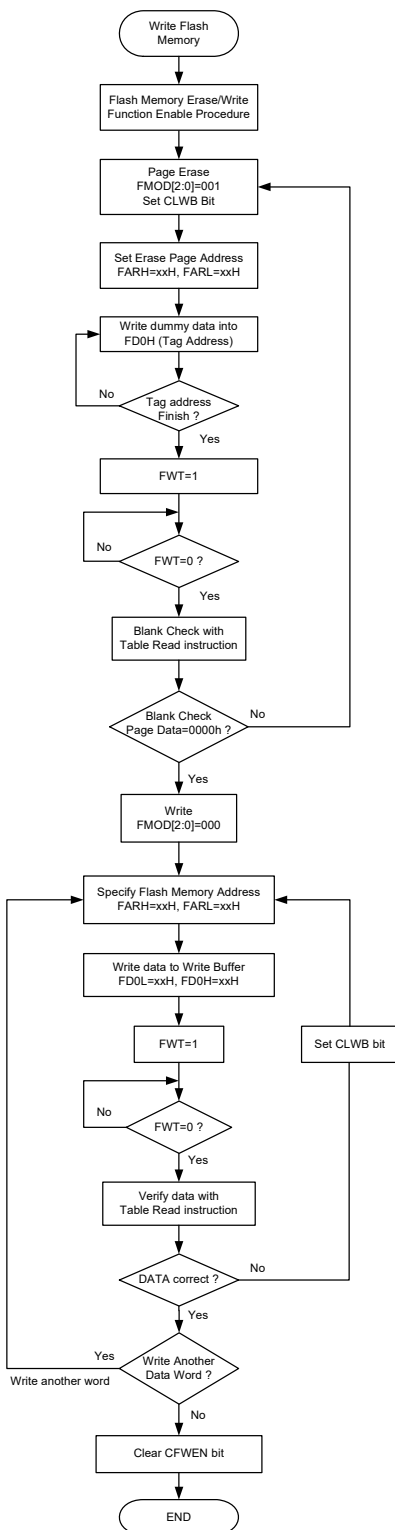
- 注：1. 当擦或写动作成功启动后，所有 CPU 相关操作将暂停。
2. 在擦除或写入操作中，FWT 位由高变低所需时间可以通过 FC2 寄存器中的 FWERTS 位选择。

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，并设置 CLWB 位为“1”清除“写入缓冲器”。设定 FWT 位为“1”，擦除 FARH 和 FARL 指定且已标记地址的目标页，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

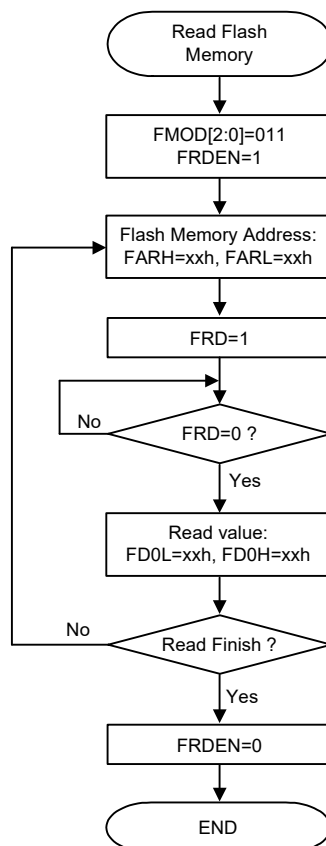
- 注：1. 当擦或写动作成功启动后，所有 CPU 相关操作将暂停。
2. 在擦除或写入操作中，FWT 位由高变低所需时间可以通过 FC2 寄存器中的 FWERTS 位选择。

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写入 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，且不清除 Flash 存储器，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 存储器读出程序

要启动 Flash 存储器读出程序，需将 FMODE[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. 当读动作成功启动后，所有 CPU 相关操作将暂停。
2. FRD 位由高变低所需时间为 3 个指令周期（典型值）。

数据存储器

数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。

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

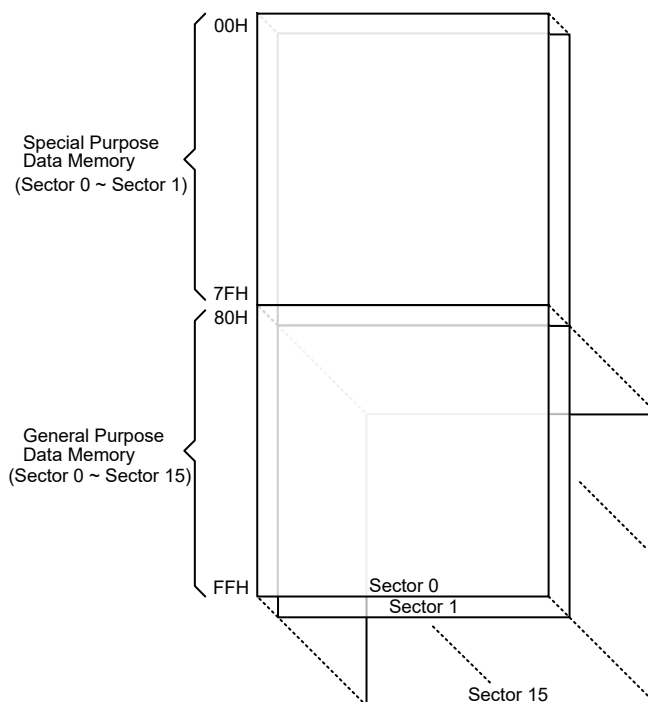
当使用间接寻址时，切换不同的数据存储器 Sector 通过设置正确的间接寻址指针值实现。

结构

数据存储器被分为多个 Sector，都位于 8 位存储器中。每个数据存储器 Sector 分为两种类型，即特殊功能数据存储器和通用数据存储器。特殊功能数据存储器地址范围为 00H~7FH，而通用数据存储器地址范围为 80H~FFH。

特殊功能数据存储器	通用数据存储器	
位于 Sector	容量	Sector: 地址
Sector 0: 00H~7FH Sector 1: 00H~7FH	2048×8	0: 80H~FFH
		1: 80H~FFH
		:
		15: 80H~FFH

数据存储器概要



数据存储器结构

数据存储寻址

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

直接寻址可用于所有 Sector，通过扩展指令可以寻址所有可用的数据存储器空间。当所访问的数据存储器位于除 Sector 0 外的任何数据存储器 Sector 时，扩展指令可代替间接寻址方式用来访问数据存储器。标准指令和扩展指令的主要区别在于扩展指令中的数据存储器地址“m”有 12 个有效位，高字节表示 Sector，低字节表示指定的地址。

通用数据存储器

所有的单片机程序需要一个读/写的存储区，让临时数据可以被储存和再使用，该 RAM 区域就是通用数据存储器。这个数据存储区可让使用者进行读取和写入的操作。使用位操作指令可对个别的位做置位或复位的操作，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	Sector 0		Sector 1
00H	IAR0		40H		EEC
01H	MP0		41H	EEAL	
02H	IAR1		42H	EEAH	
03H	MP1L		43H	EED	FC0
04H	MP1H		44H	PWRC	FC1
05H	ACC		45H	IREFC	FC2
06H	PCL		46H	PVREF	FARL
07H	TBLP		47H	OPA1C	FARH
08H	TBLH		48H	OPA2C	FD0L
09H	TBHP		49H	OPA3C	FD0H
0AH	STATUS		4AH	GSC1	FD1L
0BH	PBP		4BH	AFEDA1C	FD1H
0CH	IAR2		4CH	AFEDA1L	FD2L
0DH	MP2L		4DH	AFEDA1H	FD2H
0EH	MP2H		4EH	AFEDA2C	FD3L
0FH	RSTFC		4FH	AFEDA2L	FD3H
10H	SCC		50H	AFEDA2H	IFS
11H	HIRCC		51H	AFEDA3C	
12H	STKPTR		52H	AFEDA3L	
13H	IECC		53H	AFEDA3H	PAS0
14H	PA		54H	AFEDA0C	PAS1
15H	PAC		55H	AFEDA0L	PBS0
16H	PAPU		56H	AFEDA0H	
17H	PAWU		57H		
18H	RSTC		58H		
19H	LVRC		59H	MDUWR0	
1AH	TLVRC		5AH	MDUWR1	
1BH	MF10		5BH	MDUWR2	
1CH	MF11		5CH	MDUWR3	
1DH			5DH	MDUWR4	
1EH	WDTC		5EH	MDUWR5	
1FH	INTEG		5FH	MDUWCTRL	
20H	INTC0		60H	PGAC0	
21H	INTC1		61H	PGAC1	
22H	INTC2		62H	PGACS	
23H	INTC3		63H	ADRL	
24H	PB		64H	ADRM	
25H	PBC		65H	ADRH	
26H	PBPU		66H	ADCR0	
27H	PCRL		67H	ADCR1	
28H	PCRH		68H	SINC3	
29H			69H	ADCR2	
2AH			6AH	ADCR3	
2BH			6BH	ADCS	
2CH	PSCR		6CH		
2DH	TB0C		6DH	CMPC	
2EH	TB1C		6EH	PRFC	
2FH			6FH	TDAVRST	
30H		CTM0C0	70H	TDAVRLEN	
31H		CTM0C1	71H	TOPST	
32H		CTM0DL	72H	TOPLEN	
33H		CTM0DH	73H	TDAST	
34H		CTM0AL	74H	TDALLEN	
35H		CTM0AH	75H	TCPST	
36H		CTM1C0	76H	TCPLEN	
37H		CTM1C1	77H	TCHS0	
38H	PTMC0	CTM1DL	78H	TCHS1	
39H	PTMC1	CTM1DH	79H		
3AH	PTMDL	CTM1AL	7AH	SIMC0	
3BH	PTMDH	CTM1AH	7BH	SIMC1	
3CH	PTMAL	CRCCR	7CH	SIMD	
3DH	PTMAH	CRCIN	7DH	SIMA/SIMC2	
3EH	PTMRPL	CRCDL	7EH	SIMTOC	
3FH	PTMRPH	CRCDH	7FH		

: Unused, read as 00H
 : Reserved, cannot be changed

特殊功能数据存储结构

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存储器寻址不同，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。

存储器指针 – MP0, MP1H/MP1L, MP2H/MP2L

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

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

间接寻址程序举例

Example 1

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

Example 2

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block db ?
code .section at 0 'code'
org 00h
start:
    mov a,04h                ; setup size of block
    mov block,a
    mov a,01h                ; setup the memory sector
    mov mplh,a
    mov a,offset adres1      ; Accumulator loaded with first RAM address
    mov mp1l,a               ; setup memory pointer with first RAM address
loop:
    clr IAR1                 ; clear the data at address defined by MP1
    inc mp1l                 ; increment memory pointer MP1L
    sdz block                 ; check if last memory location has been cleared
    jmp loop
continue:
    :
```

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。

使用扩展指令直接寻址程序举例

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

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

程序存储区指针 – PBP

该单片机程序存储器被分为几个 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	—	—	—	—	—	—	PBP1	PBP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

Bit 1~0 **PBP1~PBP0**: 程序存储区选择
 00: Bank 0
 01: Bank 1
 10: Bank 2
 11: Bank 3

累加器 – ACC

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

程序计数器低字节寄存器 – PCL

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

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使用者指定的地址。

状态寄存器 – STATUS

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

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

SC、CZ、Z、OV、AC 和 C 标志位通常反映最近运算的状态。

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

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

● 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**: 当 OV 与当前指令操作结果 MSB 执行“XOR”所得结果
- Bit 6 **CZ**: 不同指令不同标志位的操作结果
 对于 SUB/SUBM/LSUB/LSUBM 指令, CZ 等于 Z 标志位。
 对于 SBC/SBCM/LSBC/LSBCM 指令, CZ 等于上一个 CZ 标志位与当前零标志位执行“AND”所得结果。对于其它指令, CZ 标志位无影响。
- Bit 5 **TO**: 看门狗溢出标志位
 0: 系统上电或执行“CLR WDT”或“HALT”指令后
 1: 看门狗溢出发生
- Bit 4 **PDF**: 暂停标志位
 0: 系统上电或执行“CLR WDT”指令后
 1: 执行“HALT”指令
- Bit 3 **OV**: 溢出标志位
 0: 无溢出
 1: 运算结果高两位的进位状态异或结果为 1
- Bit 2 **Z**: 零标志位
 0: 算术或逻辑运算结果不为 0
 1: 算术或逻辑运算结果为 0
- Bit 1 **AC**: 辅助进位标志位
 0: 无辅助进位
 1: 在加法运算中低四位产生了向高四位进位, 或减法运算中低四位未发生从高四位借位
- 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未发生借位
 C 也受循环移位指令的影响。

EEPROM 数据存储

该单片机内建 EEPROM 数据存储。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 数据存储结构

该单片机的 EEPROM 数据存储容量为 2048×8 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。使用 Sector 0 中的一对地址寄存器和一个数据寄存器以及 Sector 1 中的一个控制寄存器，可以实现对 EEPROM 的单字节读写操作。

EEPROM 寄存器

有四个寄存器控制内部 EEPROM 数据存储总的操作，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，仅可通过 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	—	—	—	—	—	EEAH2	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	—	—	—	—	—	EEAH2	EEAH1	EEAH0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 未定义，读为“0”

Bit 2~0 EEAH2~EEAH0: 数据 EEPROM 地址高字节 Bit 2 ~ 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:** 数据 EEPROM 擦除时间和写入时间选择位

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

此位用来使能数据 EEPROM 擦功能，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:** 数据 EEPROM 擦控制位

0: 擦周期结束

1: 开始擦周期

此位为数据 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:** 数据 EEPROM 操作模式选择位

0: 字节操作模式

1: 页操作模式

此位为数据 EEPROM 操作模式选择位。当此位为高，则选择页写、擦或读操作模式。当此位为 0，则选择字节写或读操作模式。EEPROM 页缓存器大小为 16 字节。

Bit 3 **WREN:** 数据 EEPROM 写使能位

0: 除能

1: 使能

此位为数据 EEPROM 写使能位，向数据 EEPROM 写操作之前需将此位置高。将此位清零时，则禁止向数据 EEPROM 写操作。

Bit 2 **WR:** EEPROM 写控制位

0: 写周期结束

1: 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. 确保 f_{SUB} 时钟在执行擦 / 写动作前已稳定。
3. 确保擦 / 写动作完成后才可改写 EEPROM 相关寄存器或启动 IAP 功能。

EEPROM 读数据

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，可执行 EEPROM 字节读操作。为了实现字节读操作，EEPROM 中要读取数据的地址需先放入 EEAH 和 EEAL 寄存器中，EEC 寄存器中的读使能位 RDEN 也要置高以使能读功能，然后再置高 RD 位以开始 EEPROM 字节读操作。注意，若 RD 为已置高而 RDEN 位还未被置高则不能开始读操作。若读周期结束，RD 位将自动清零，EEPROM 数据可以从 EED 寄存器中读取。读到的数据在其他读或写操作执行前将一直保留在 EED 寄存器中。应用程序可轮询 RD 位以确定数据可以有效地读取。

页读模式

当模式选择位 MODE 为 1 时，可执行 EEPROM 页读操作。页读操作中页大小可达 16 个字节。为了实现页读操作，EEPROM 中要读取页的起始地址需先放入 EEAH 和 EEAL 寄存器中，EEC 寄存器中的读使能位 RDEN 也要置高以使能读功能，然后再置高 RD 位以开始 EEPROM 页读操作。注意，若 RD 为已置高而 RDEN 位还未被置高则不能开始读操作。当前字节读周期结束时，RD 位将自动清零，此时可以从 EED 寄存器中读取 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读取。

EEPROM 地址高 7 位用来指定要读取页的位置，而低 4 位用来指向实际的地址。在页操作模式低 4 位地址将自动加一，而高 7 位地址不会自动增加。当 EEPROM 地址低 4 位自动递增到当前页的最大地址，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 7 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任意数据到 EED 寄存器，低 4 位地址将自动加一，而高 7 位地址不会自动增加。当 EEPROM 地址低 4 位自动递增到当前页的最大地址，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，要写入的任意数据也需存入 EED 寄存器中。一页的最大数据长度为 16 字节。注意写数据到 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。当一整页的任意数据都写入 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，所以擦除 EEPROM 数据的时间将有所延迟。可通过轮询 EEC 寄存器中的 ER 位或判断 EEPROM 中断以侦测擦周期是否完成。若擦除周期完成，ER 位将自动清零，通知用户数据已擦除。因此，应用程序将轮询 ER 位以确定擦除周期是否结束。擦操作结束后，EREN 位将会被硬件置低。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

此单片机有两种模式可实现写数据到 EEPROM，即字节写模式和页写模式，可通过 EEC 寄存器中的 EEPROM 操作模式选择位 MODE 选择。

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 位将自动清零，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。写操作结束后，WREN 位将会被硬件置低。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，务必确保已成功执行了相关的页擦除操作。当模式选择位 MODE 为 1 时，可执行 EEPROM 页写操作。EEPROM 一页可写入 16 个字节。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 7 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 7 位地址不会自动增加。当 EEPROM 地址低 4 位自动递增到当前页的最大地址，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。

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 位将自动清零，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。写操作结束后，WREN 位将会被硬件置低。

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 中断，需先通过设置相关中断寄存器的 DEE 位使能 EEPROM 中断。EEPROM 中断属于多功能中断。当 EEPROM 擦 / 写周期结束，DEF 请求标志位和相应的多功能中断请求标志位将被置位。若总中断、EEPROM 中断使能和多功能中断使能且堆栈未满的情况下将跳转到相应的多功能中断向量中执行。当中断被响应，多功能中断标志位将自动复位，而 EEPROM 中断标志位需通过应用程序复位。总中断标志位也会自动复位以除能其他中断。详情可参考中断章节。

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

程序举例

从 EEPROM 中读取一个字节数据 – 轮询法

```
MOV A, 40H                ; set memory pointer low byte MP1L
MOV MP1L, A                ; MP1 points to EEC register
MOV A, 01H                ; set 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, 40H ; set memory pointer low byte MP1L
MOV MP1L, A ; MP1 points to EEC register
MOV A, 01H ; set 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, 40H ; set memory pointer low byte MP1L
MOV MP1L, A ; MP1 points to EEC register
MOV A, 01H ; set 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 define data, erase mode don't care data
; value
MOV EED, A
RET
:
Erase_START:
CLR EMI
```

```
SET IAR1.6                ; set EREN bit, select 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, 40H                ; set memory pointer low byte MP1L
MOV MP1L, A                ; MP1 points to EEC register
MOV A, 01H                ; set memory pointer high byte MP1H
MOV MP1H, A

CLR IAR1.4                ; clear MODE bit, select byte write 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 define 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, 40H                ; set memory pointer low byte MP1L
MOV MP1L, A                ; MP1 points to EEC register
MOV A, 01H                ; set 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) ~~~~
CALLWRITE_BUF
CALLWRITE_BUF
:
:
JMP WRITE_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA        ; user define 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。振荡器选择及相关操作是通过应用程序和相关的控制寄存器共同完成的。

振荡器概述

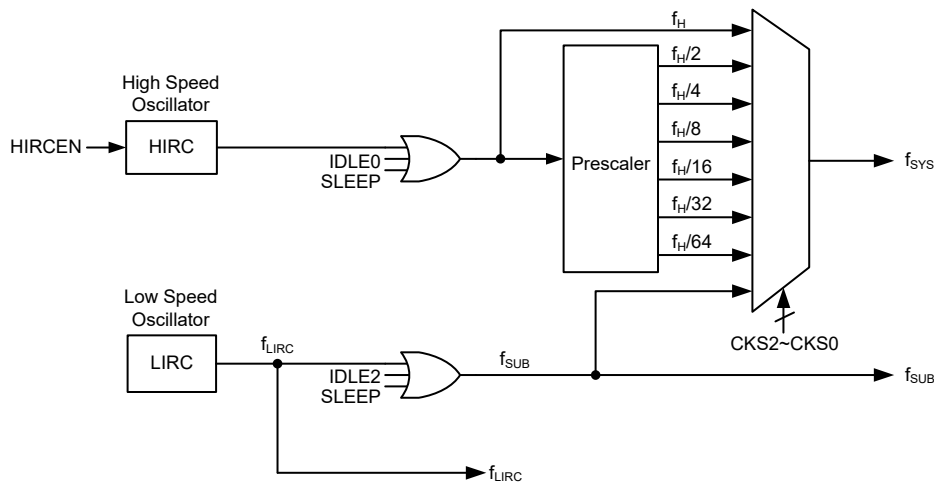
振荡器除了作为系统时钟源，还作为看门狗定时器和时基中断的时钟源。两个完全集成的内部振荡器不需要任何外围器件，它们提供的高速和低速系统振荡器具有较宽的频率范围。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

类型	名称	频率
内部高速 RC	HIRC	4/8/12MHz
内部低速 RC	LIRC	32kHz

振荡器类型

系统时钟配置

该单片机有两个系统振荡器，包括一个高速振荡器和一个低速振荡器。高速振荡器为内部 4/8/12MHz 高速振荡器 HIRC，低速振荡器有内部 32kHz 低速振荡器 LIRC。使用高速或低速振荡器作为系统时钟的选择是通过设置 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 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，不需其它外部器件。内部 RC 振荡器具有三种固定的频率：4MHz、8MHz 和 12MHz，可通过 HIRCC 寄存器中的

HIRC1~HIRC0 位进行选择。为了确保能达到交流电气特性里描述的 HIRC 频率精准度，HIRC1~HIRC0 位需要与配置选项中选择频率吻合。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 系统振荡器是一个完全集成 RC 振荡器，它的典型频率值为 32kHz 且无需外部元件。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响较大程度地降低。

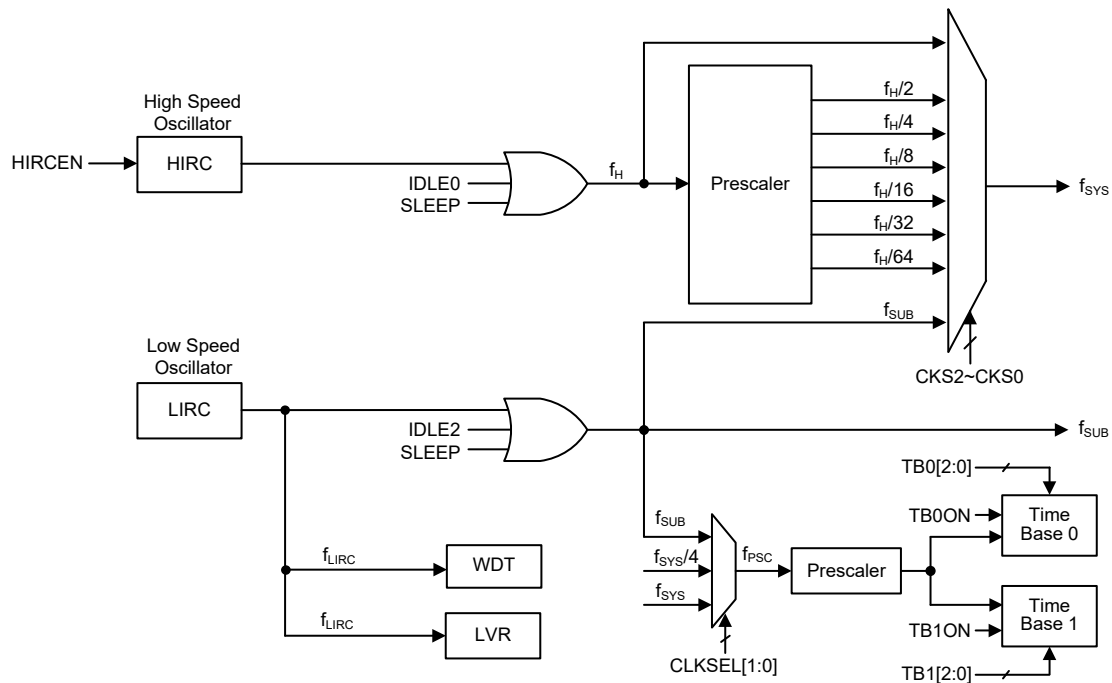
工作模式和系统时钟

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。低频系统时钟来自 LIRC 振荡器。其它系统时钟还有高速系统振荡器的分频 $f_H/2 \sim f_H/64$ 。



单片机时钟选项

注：当系统时钟源 f_{SYS} 由 f_H 到 f_{SUB} 转换时，可以通过设置相应的高速振荡器使能控制位，选择停止以节省耗电，或者继续振荡，为外围电路提供 $f_H \sim f_H/64$ 频率的时钟源。

系统工作模式

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

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/Off ⁽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，f_{LIRC} 开启或关闭由 WDT 功能使能或除能控制。

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。若看门狗定时器功能使能，f_{LIRC} 继续运行。

空闲模式 0

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 位为低、FSIDEN 位为高时，系统进入空闲模式 0。在空闲模式 0 中，CPU 停止，但低速振荡器会开启以驱动一些外围功能。

空闲模式 1

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，但高速和低速振荡器都会开启以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，但高速振荡器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	—	—	—	—	HIRC1	HIRC0	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	0	—	—	—	0	0

Bit 7~5 **CKS2~CKS0**: 系统时钟选择位

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

这三位用于选择系统时钟源。除了 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。

注：使用 CKS2~CKS0 位进行时钟切换设置之后，在相关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	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	1

Bit 7~4 未定义，读为“0”

Bit 3~2 **HIRC1~HIRC0**: HIRC 频率选择位

00: 4MHz
01: 8MHz
10: 12MHz
11: 4MHz

当 HIRC 振荡器使能或通过应用程序改变 HIRC 频率选择位时，在 HIRCF 标志位置高后时钟频率会自动改变。

建议这里选择的频率与配置选项中选定的频率保持一致，以确保能够达到交流电气特性中标示的 HIRC 频率精准度。

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，HIRCF 位会先被清零，在 HIRC 稳定后会被置高。

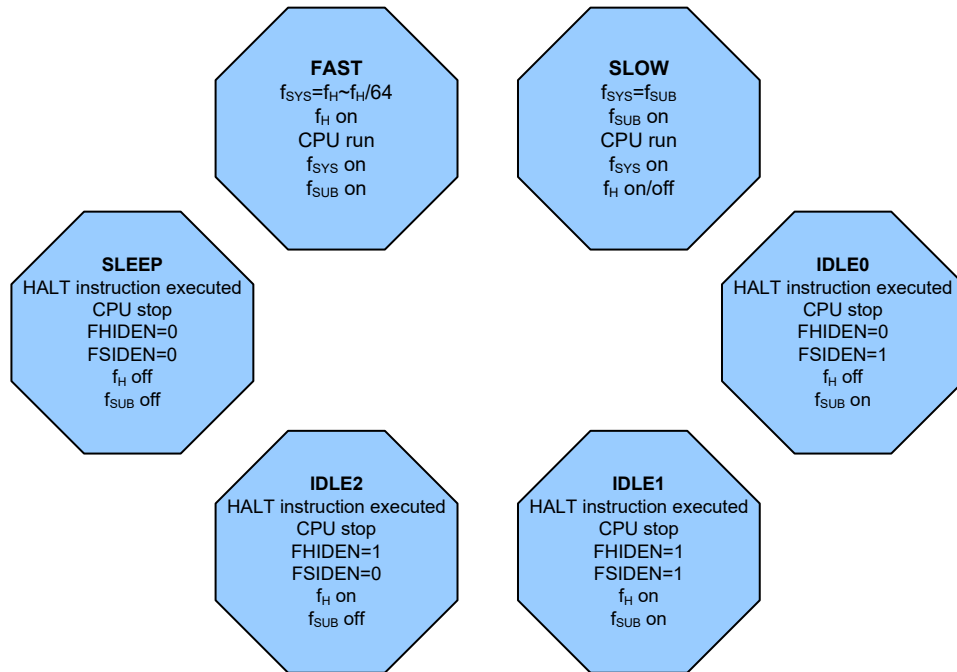
Bit 0 **HIRCEN**: HIRC 振荡器使能控制位

0: 除能
1: 使能

工作模式切换

单片机可在各个工作模式间自由切换，使得用户可根据所需选择较佳的性能 / 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。

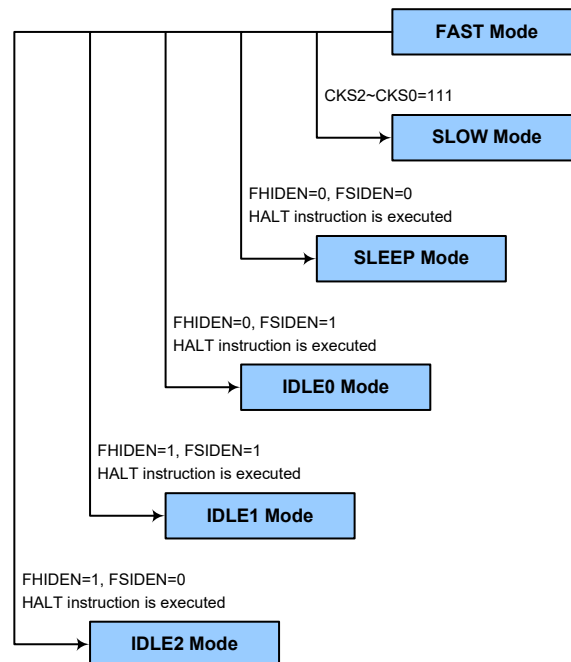
简单来说，快速模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



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。

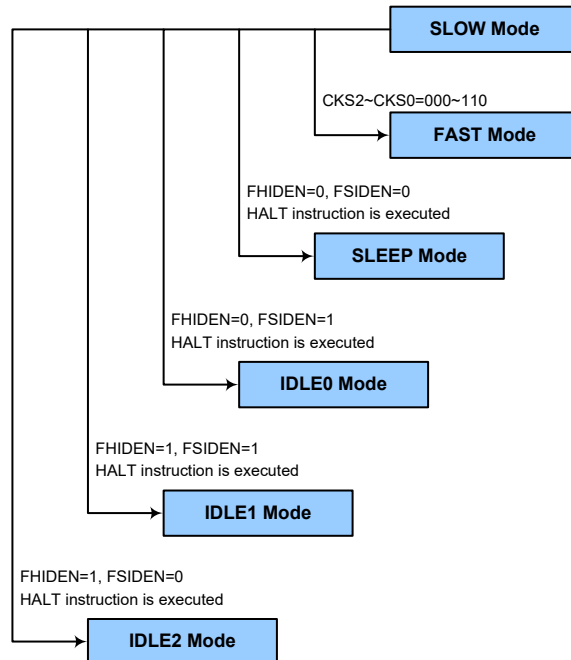
低速模式的时钟源来自 LIRC 振荡器，因此要求所选振荡器在所有模式切换动作发生前稳定下来。



低速模式切换到快速模式

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 $CKS2 \sim CKS0$ 位为“000”~“110”使系统时钟从 f_{SUB} 切换到 $f_H \sim f_H/64$ 。

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，既应用程序中执行“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。
- 如果 WDT 功能使能，WDT 将被清零并重新开始计数。如果 WDT 功能除能，WDT 将被清零并停止计数。

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。
- 如果 WDT 功能使能，WDT 将被清零并重新开始计数。如果 WDT 功能除能，WDT 将被清零并停止计数。

进入空闲模式 1

进入空闲模式 1 的方法仅有一种，既应用程序中执行“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。
- 如果 WDT 功能使能，WDT 将被清零并重新开始计数。如果 WDT 功能除能，WDT 将被清零并停止计数。

进入空闲模式 2

进入空闲模式 2 的方法仅有一种，既应用程序中执行“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。
- 如果 WDT 功能使能，WDT 将被清零并重新开始计数。如果 WDT 功能除能，WDT 将被清零并停止计数。

待机电流的注意事项

由于单片机进入休眠或空闲模式的主要原因是将单片机的电流降低到尽可能低，可能到只有几个微安的级别（空闲模式 1 和空闲模式 2 除外），所以如果要将电路的电流进一步降低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入 / 输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。

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

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

系统进入休眠或空闲模式之后，可以通过以下几种方式唤醒：

- PA 口下降沿
- 系统中断
- WDT 溢出

执行 HALT 指令，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 WDT 溢出唤醒，则会发生看门狗定时器复位，PDF 将被置位。看门狗计数器溢出将会置位 TO 标志并唤醒系统，这种复位会重置程序计数器和堆栈指针，其它标志保持原有状态。

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

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源 f_{LIRC} 由内部低速振荡器 LIRC 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 $2^8 \sim 2^{18}$ 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 寄存器用于选择溢出周期、控制 WDT 功能的使能 / 除能和单片机复位操作。

• WDTC 寄存器

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

Bit 7~3 **WE4~WE0:** WDT 功能控制

10101: 除能
01010: 使能
其它值: 单片机复位

如果由于不利的环境因素使这些位变为其它值，单片机将复位。复位动作发生在 t_{SRESET} 延迟时间后，且 RSTFC 寄存器的 WRF 位将置为“1”。

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	—	—	—	—	RSTF	LVRF	LRF	WRF
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	x	0	0

“x”：未知

Bit 7~4 未定义，读为“0”

Bit 3 **RSTF**：复位控制寄存器软件复位标志位
具体描述见内部复位控制章节。

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 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”，且只能通过应用程序清零。

看门狗定时器操作

当 WDT 溢出时，它产生一个单片机复位的动作。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除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，清除指令都不能被正确执行，此种情况下，看门狗将溢出以使单片机复位。看门狗定时器控制寄存器 WDTC 中的 WE4~WE0 位可提供使能 / 除能控制以及单片机复位操作。当 WE4~WE0 设置为“10101B”时除能 WDT 功能，而当设置为“01010B”时使能 WDT 功能。如果 WE4~WE0 设置为除“01010B”和“10101B”以外的值时，单片机将在 t_{SRESET} 延迟时间后复位。上电后这些位初始化为“01010B”。

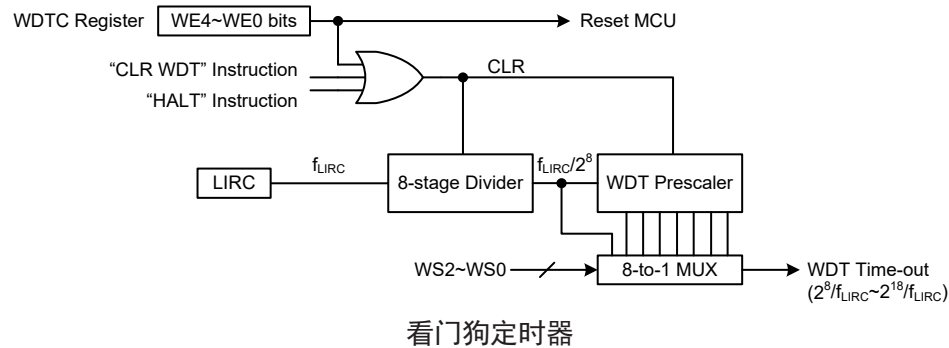
WE4~WE0 位	WDT 功能
10101B	除能
01010B	使能
其它值	单片机复位

看门狗定时器使能 / 除能控制

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



复位和初始化

复位功能是整个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。

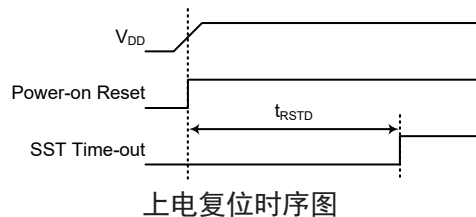
另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。另一种复位为看门狗溢出单片机复位，不同方式的复位操作会对寄存器产生不同的影响。

复位功能

单片机的几种内部复位方式将在此处做具体介绍。

上电复位

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



内部复位控制

内部复位控制寄存器 RSTC 用于为单片机在受到环境噪声干扰而异常工作时提供复位。如果 RSTC 寄存器的内容被设置为除 01010101B 或 10101010B 以外的任何值，单片机会在 t_{SRESET} 延迟时间后发生复位。上电后寄存器的值为 01010101B。

RSTC7~RSTC0 位	复位功能
01010101B	无操作
10101010B	无操作
其它值	单片机复位

内部复位功能控制

● RSTC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	RSTC7	RSTC6	RSTC5	RSTC4	RSTC3	RSTC2	RSTC1	RSTC0
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 **RSTC7~RSTC0**: 复位功能控制位

01010101: 无操作

10101010: 无操作

其它值: 单片机复位

如果由于不利的环境因素使这些位发生改变, 单片机将复位。复位动作发生在 t_{SRESET} 延迟时间后, 且 RSTFC 寄存器的 RSTF 位将置为“1”。除了 WDT 溢出硬件复位外, 其它所有复位发生时此寄存器恢复至上电复位值。

● RSTFC 寄存器

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

“x”: 未知

Bit 7~4 未定义, 读为“0”

Bit 3 **RSTF**: 复位控制寄存器软件复位标志位

0: 未发生

1: 发生

当 RSTC 控制寄存器软件复位发生时, 此位被置为“1”, 且只能通过应用程序清零。

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

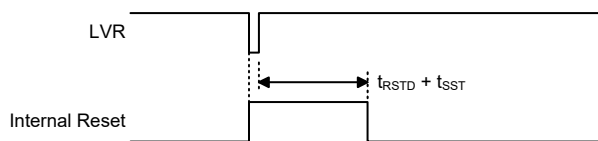
具体描述见看门狗定时器控制寄存器章节。

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。当电源电压低于某一预定值时, 它将复位单片机。

例如在更换电池的情况下, 单片机供应的电压可能会在 $0.9V \sim V_{LVR}$ 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。所谓有效的 LVR 信号, 即在 $0.9V \sim V_{LVR}$ 的低电压状态的时间, 必须超过 LVR 电气特性中 t_{LVR} 参数的值。如果低电压存在时间不超过 t_{LVR} 参数的值, 则 LVR 将会忽略它且不会执行复位功能。实际的 t_{LVR} 参数的值可通过 TLVRC 寄存器中的 TLVR1~TLVR0 位设置。实际的 V_{LVR} 参数的值可通过 LVRC 寄存器中的

LVS7~LVS0 位选择。若 LVS7~LVS0 位的值由于不利的环境因素如噪声而发生改变，单片机将在一段延时时间 t_{SRESET} 后复位。此时 RSTFC 寄存器的 LRF 位被置位。上电后寄存器的值为 01010101B。需要注意的是，当单片机进入空闲或休眠模式，LVR 功能将自动除能。



低电压复位时序图

• LVRC 寄存器

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

Bit 7~0 **LVS7~LVS0**: LVR 电压选择

01010101: 2.1V

00110011: 2.55V

10011001: 3.15V

10101010: 3.8V

其他值: 单片机复位 – 寄存器复位为 POR 值

当上述定义的相应的低电压出现，且低电压保持时间大于 t_{LVR} 值，则单片机复位发生。此时复位后的寄存器内容保持不变。

除了上述四个定义的寄存器值外，其他值将会产生单片机复位。复位操作会在 t_{SRESET} 时间后执行。注意的是此处单片机复位后，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: $(7\sim8) \times t_{LIRC}$

01: $(31\sim32) \times t_{LIRC}$

10: $(63\sim64) \times t_{LIRC}$

11: $(127\sim128) \times t_{LIRC}$

• RSTFC 寄存器

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

“x”: 未知

Bit 7~4 未定义，读为“0”

Bit 3 **RSTF**: 复位控制寄存器软件复位标志位
具体描述详见内部复位章节。

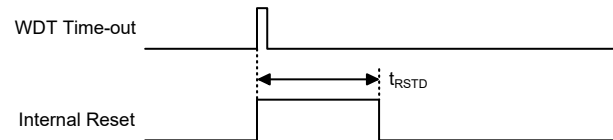
Bit 2	LVRF: LVR 复位标志位 0: 未发生 1: 发生 当特定的低电压复位条件发生时, 此位被置为“1”, 且只能通过应用程序清零。
Bit 1	LRF: LVR 控制寄存器软件复位标志位 0: 未发生 1: 发生 如果 LVRC 寄存器包含任何非定义的 LVR 电压值, 此位被置为“1”, 这类似与软件复位功能, 且只能通过应用程序清零。
Bit 0	WRF: WDT 控制寄存器软件复位标志位 具体描述见看门狗定时器控制寄存器章节。

IAP 复位

当写值“55H”至 FC1 寄存器时, 将产生一个复位信号将整个单片机复位。详见 IAP 章节。

正常运行时看门狗溢出复位

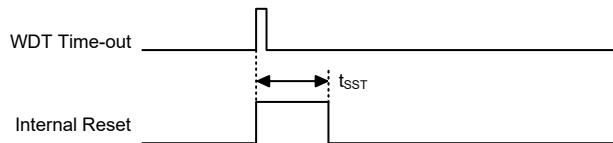
在快速模式或低速模式时看门狗溢出复位, 看门狗溢出标志位 TO 将被设为“1”。



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, 绝大部份的条件保持不变。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	空闲或休眠模式时的 WDT 溢出复位

“u”代表不改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

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

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

寄存器	上电复位	WDT 溢出 (正常运行)	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	-xxx xxxx	-uuu uuuu	-uuu uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu1l uuuu
PBP	---- --00	---- --00	---- --uu
IAR2	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- 0x00	---- uuuu	---- uuuu
SCC	000- --00	000- --00	uuu- --uu
HIRCC	---- 0001	---- 0001	---- uuuu
STKPTR	0--- 0000	0--- 0000	u--- uuuu
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
RSTC	0101 0101	0101 0101	uuuu uuuu
LVRC	0101 0101	0101 0101	uuuu uuuu
TLVRC	---- --01	---- --01	---- --uu
MFIO	0000 0000	0000 0000	uuuu uuuu
MFII	-000 -000	-000 -000	-uuu -uuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
WDTC	0101 0011	0101 0011	uuuu uuuu
INTEG	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	---0 ---0	---0 ---0	---u ---u
PB	---- ---1	---- ---1	---- ---u
PBC	---- ---1	---- ---1	---- ---u
PBPU	---- ---0	---- ---0	---- ---u
PCRL	0000 0000	0000 0000	uuuu uuuu
PCRH	-000 0000	-000 0000	-uuu uuuu
PSCR	---- --00	---- --00	---- --uu
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu
PTMC0	0000 0---	0000 0---	uuuu u---
PTMC1	0000 0000	0000 0000	uuuu uuuu
PTMDL	0000 0000	0000 0000	uuuu uuuu
PTMDH	0000 0000	0000 0000	uuuu uuuu
PTMAL	0000 0000	0000 0000	uuuu uuuu
PTMAH	0000 0000	0000 0000	uuuu uuuu
PTMRPL	0000 0000	0000 0000	uuuu uuuu
PTMRPH	0000 0000	0000 0000	uuuu uuuu
EEAL	0000 0000	0000 0000	uuuu uuuu
EEAH	---- -000	---- -000	---- -uuu
EED	0000 0000	0000 0000	uuuu uuuu
PWRC	00-- 0000	00-- 0000	uu-- uuuu
IREFC	---0 --00	---0 --00	---u --uu
PVREF	0000 0000	0000 0000	uuuu uuuu
OPA1C	--0- ----	--0- ----	--u- ----
OPA2C	--0- ----	--0- ----	--u- ----
OPA3C	--0- ----	--0- ----	--u- ----
GSC1	---- -000	---- -000	---- -uuu
AFEDA1C	---- --00	---- --00	---- --uu
AFEDA1L	0000 ----	0000 ----	uuuu ----
AFEDA1H	0000 0000	0000 0000	uuuu uuuu
AFEDA2C	---- --00	---- --00	---- --uu
AFEDA2L	0000 ----	0000 ----	uuuu ----
AFEDA2H	0000 0000	0000 0000	uuuu uuuu
AFEDA3C	---- --00	---- --00	---- --uu
AFEDA3L	0000 ----	0000 ----	uuuu ----
AFEDA3H	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
AFEDA0C	---- --00	---- --00	---- --uu
AFEDA0L	0000 ----	0000 ----	uuuu ----
AFEDA0H	0000 0000	0000 0000	uuuu uuuu
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-- ----
PGAC0	-00- 0000	-00- 0000	-uu- uuuu
PGAC1	-000 --00	-000 --00	-uuu --uu
PGACS	0000 0000	0000 0000	uuuu uuuu
ADRL	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRM	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR0	0-00 000-	0-00 000-	u-uu uuu-
ADCR1	000- -00-	000- -00-	uuu- -uu-
SINC3	1001 0011	1001 0011	uuuu uuuu
ADCR2	00-0 0010	00-0 0010	uu-u uuuu
ADCR3	0111 ---1	0111 ---1	uuuu ---u
ADCS	---0 0000	---0 0000	---u uuuu
CMPC	-000 0000	-000 0000	-uuu uuuu
PRFC	0--- 0000	0--- 0000	u--- 0000
TDAVRST	0000 0000	0000 0000	uuuu uuuu
TDAVRLEN	0000 0000	0000 0000	uuuu uuuu
TOPST	0000 0000	0000 0000	uuuu uuuu
TOPLEN	0000 0000	0000 0000	uuuu uuuu
TDAST	0000 0000	0000 0000	uuuu uuuu
TDALen	0000 0000	0000 0000	uuuu uuuu
TCPST	0000 0000	0000 0000	uuuu uuuu
TCPLEN	0000 0000	0000 0000	uuuu uuuu
TCHS0	0000 0000	0000 0000	uuuu uuuu
TCHS1	0--- ----	0--- ----	u--- ----
SIMC0	111- 0000	111- 0000	uuu- uuuu
SIMC1	1000 0001	1000 0001	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/SIMC2	0000 0000	0000 0000	uuuu uuuu
SIMTOC	0000 0000	0000 0000	uuuu uuuu
CTM0C0	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
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
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
EEC	0000 0000	0000 0000	uuuu uuuu
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	-000 0000	-000 0000	-uuu 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
IFS	--00 000-	--00 000-	--uu uu-
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	---- --00	---- --00	---- --uu

注：“u”表示不改变
“x”表示未知
“-”表示未定义

输入 / 输出端口

Holtek 单片机的输入 / 输出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~PB 双向输入 / 输出。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	—	—	—	—	—	—	—	PB0
PBC	—	—	—	—	—	—	—	PBC0
PBPU	—	—	—	—	—	—	—	PBPU0

“—”：未定义，读为“0”

I/O 逻辑功能寄存器列表

上拉电阻

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

需要注意的是，当 I/O 引脚设为数字输入或 NMOS 输出时，上拉功能才会受 PxPU 控制开启，其它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 口上拉电阻控制位

0: 除能

1: 使能

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

PA 口唤醒

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

需要注意的是，只有当引脚被设置为通用型输入类型且单片机处于空闲 / 休眠模式时，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 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 / 输出端口都具有各自的控制寄存器，即 PAC~PBC，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 信号被设为“0”时，如果对输出做读取动作时，程序读取到的是内部输出数据锁存器中的状态，而不是输出引脚上实际的逻辑状态。

● PxC 寄存器

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

PxCn: I/O Px 口类型选择位
0: 输出
1: 输入

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

引脚共用功能

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

引脚共用功能选择寄存器

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”输出功能选择寄存器“n”，记为 PxCn，和输入功能选择寄存器记为 IFS，这些寄存器可以用来选择共用引脚的特定功能。

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字段时，一些数字输入引脚如 INTn、xTCKn、PTPI

等，与对应的通用 I/O 口共用同一个引脚共用设置选项。要选择这些引脚功能，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
IFS	—	—	—	—	SDISDAPS1	SDISDAPS0	SCKSCLPS	—
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	—	—	—	—	—	—	PBS01	PBS00

引脚共用功能选择寄存器列表

● IFS 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SDISDAPS1	SDISDAPS0	SCKSCLPS	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4 未定义，读为“0”

Bit 3~2 **SDISDAPS1~SDISDAPS0**: SDI/SDA 输入源引脚选择

00: PA7

01: PA2

10: 保留

11: PA7

Bit 1 **SCKSCLPS**: SCK/SCL 输入源引脚选择

0: PB0

1: PA4

Bit 0 未定义，读为“0”

● PAS0 寄存器

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

Bit 7~6 **PAS07~PAS06**: PA3 引脚共用功能选择

00: PA3

01: PA3

10: PA3

11: AIN0

Bit 5~4 **PAS05~PAS04**: PA2 引脚共用功能选择

00: PA2

01: SDI/SDA

10: 保留

11: PA2

Bit 3~2 **PAS03~PAS02**: PA1 引脚共用功能选择

00: PA1

01: PA1

10: PA1

11: AIN1

Bit 1~0 **PAS01~PAS00:** PA0 引脚共用功能选择
 00: PA0
 01: SDO
 10: 保留
 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
 00: PA7/INT2/PTPI
 01: SDI/SDA
 10: 保留
 11: PA7/INT2/PTPI

Bit 5~4 **PAS15~PAS14:** PA6 引脚共用功能选择
 00: PA6/CTCK1
 01: CTP1
 10: SDO
 11: 保留

Bit 3~2 **PAS13~PAS12:** PA5 引脚共用功能选择
 00: PA5/INT1/PTCK
 01: SCS
 10: PA5/INT1/PTCK
 11: CMPO

Bit 1~0 **PAS11~PAS10:** PA4 引脚共用功能选择
 00: PA4/INT0/CTCK0
 01: CTP0
 10: SCK/SCL
 11: PA4/INT0/CTCK0

• PBS0 寄存器

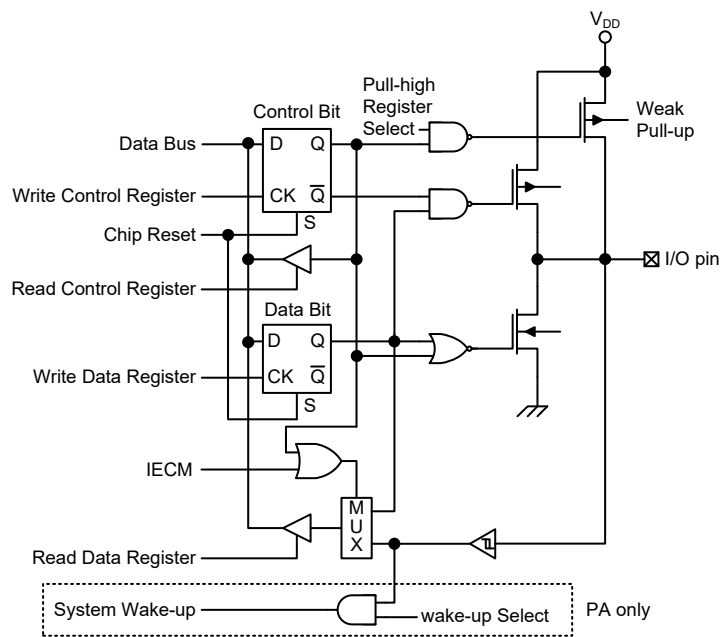
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PBS01	PBS00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

Bit 1~0 **PBS01~PBS00:** PB0 引脚共用功能选择
 00: PB0/INT3
 01: SCK/SCL
 10: PTP
 11: CMPO

输入 / 输出引脚结构

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



逻辑功能输入 / 输出端口结构

读端口功能

读端口功能用于读取从数据锁存或 I/O 引脚上输出的数据，该功能专为 I/O 功能和 A/D 通道的 IEC 60730 自诊断测试而设计。寄存器 IECC 用于控制读端口功能。若此功能除能，引脚将作为所选中的共用功能引脚。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。

• 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 bit 7 ~ bit 0

11001010: IECM=1 - 读端口功能使能

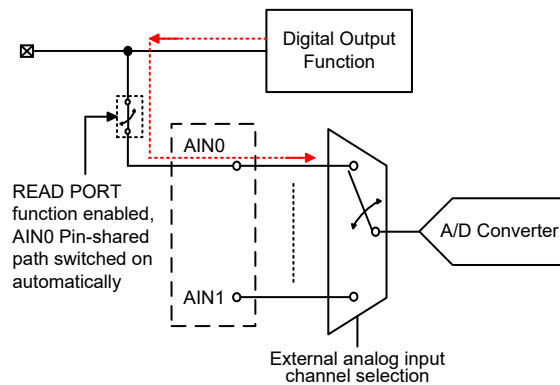
其他值: IECM=0 - 读端口功能除能

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

注：上方表格所呈现的值为执行了“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。

读端口模式的另一个功能是检查 A/D 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 A/D AIN0~AIN1，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，无论 AIN0 是否选择作为模拟输入引脚使用，只要读端口功能使能，AIN0 模拟输入通道都将开启。通过这种方式，AIN0 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 AIN0 模拟输入通道检测。

应注意，当使用读端口功能检测 A/D 通道时，A/D 转换器参考电压应等于 I/O 电源电压。



A/D 通道输入路径内部连接

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

TM 功能	CTM	PTM
定时 / 计数器	√	√
捕捉输入	—	√
比较匹配输出	√	√
PWM 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 或 P 型 TM，n 代表具体 TM 编号。由于该单片机只包含一个 PTM，PTM 相关的引脚、寄存器和控制位都不带编号。该时钟源来自系统时钟 f_{sys} 的分频比或内部高速时钟 f_H 或 f_{SUB} 时钟源或外部 xTCKn 引脚。xTCKn 引脚时钟源用于允许外部信号作为 TM 时钟源或用于事件计数。

TM 中断

每个简易型或周期型 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，而周期型 TM 还有一个输入引脚 PTPI。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。PTCK 引脚还可用作 PTM 单脉冲输出模式的外部触发引脚。

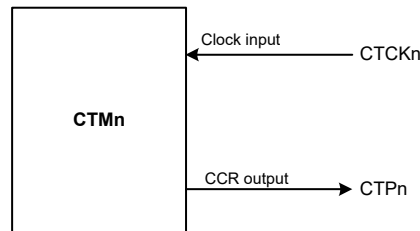
另一种 PTM 输入引脚 PTPI 作为捕捉输入脚，其有效边沿有上升沿、下降沿和双沿，通过设置 PTMC1 寄存器中的 PTIO1~PTIO0 位来选择有效边沿类型。除了 PTPI 引脚外，PTCK 引脚也可以用作捕捉输入模式的外部触发引脚。

每个 TM 都有一个输出引脚 xTPn。当 TM 工作在比较匹配输出模式且比较匹配发生时，这些引脚会由 TM 控制切换到高电平或低电平或翻转。外部输出引脚也被 TM 用来产生 PWM 输出波形。

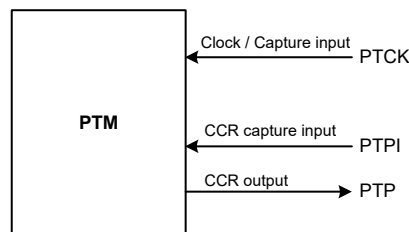
当 TM 输入和输出引脚与其它功能共用时，TM 输入和输出功能需要事先通过相关引脚共用功能选择寄存器先被设置。更多引脚共用功能选择详见引脚共用功能章节。

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

TM 外部引脚



CTM 功能引脚框图 (n=0~1)

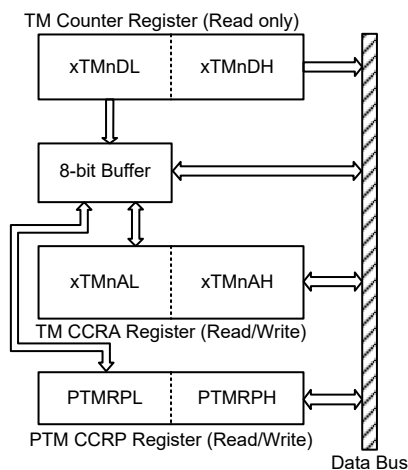


PTM 功能引脚框图

编程注意事项

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

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，即 xTMnAL 和 PTMRPL，否则可能导致无法预期的结果。

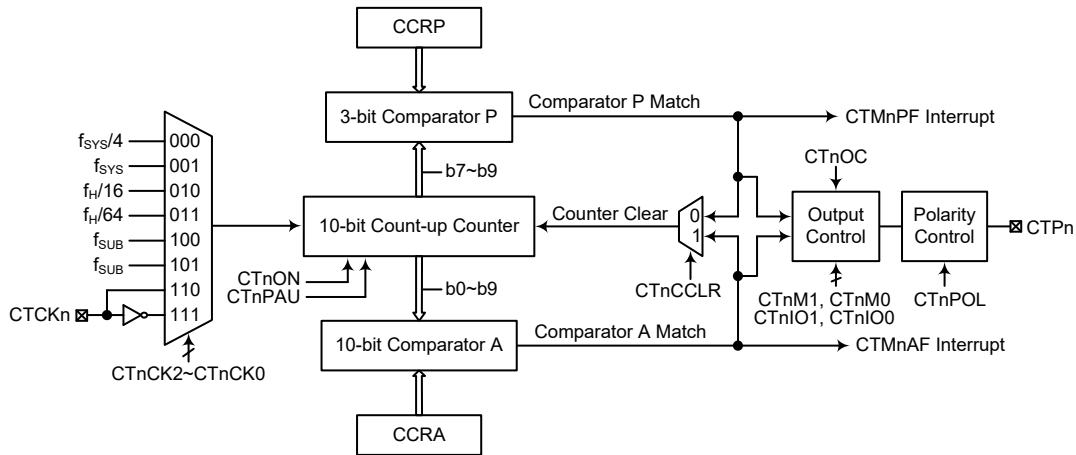


读写流程如下步骤所示：

- 写数据至 CCRA 或 CCRP
 - ◆ 步骤 1. 写数据至低字节寄存器 xTMnAL 或 PTMRPL
 - 注意，此时数据仅写入 8-bit 缓存器。
 - ◆ 步骤 2. 写数据至高字节寄存器 xTMnAH 或 PTMRPH
 - 注意，此时数据直接写入高字节寄存器，同时锁存在 8-bit 缓存器中的数据写入低字节寄存器。
- 从计数器寄存器、CCRA 或 CCRP 中读取数据
 - ◆ 步骤 1. 从高字节寄存器 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. 从低字节寄存器 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外部输出引脚。



注：CTMn 的外部引脚为多功能引脚，因此在使用 CTMn 功能之前应该合理设置相应引脚共用功能选择寄存器以使能 CTMn 引脚功能。对于 CTCKn 输入引脚还需要设置相应的端口控制寄存器，将该引脚设置为输入口。

10-bit 简易型 TM 框图 (n=0~1)

简易型 TM 操作

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

通过应用程序改变 10 位计数器值的唯一方法是使 CTnON 位发生上升沿跳变清除计数器。此外，计数器溢出或比较匹配也会自动清除计数器。上述条件发生时，通常情况会产生 CTMn 中断信号。简易型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，也可以控制输出脚。所有工作模式的设定都是通过设置相关寄存器来实现的。

简易型 TM 寄存器介绍

简易型 TM 的所有操作由一系列寄存器控制。一对只读寄存器用来存放 10 位计数器的值，一对读 / 写寄存器存放 10 位 CCRA 的值。剩下两个控制寄存器设置不同的操作和控制模式以及 CCRP 的 3 个位。

寄存器名称	位							
	7	6	5	4	3	2	1	0
CTMnC0	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
CTMnC1	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
CTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnDH	—	—	—	—	—	—	D9	D8
CTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnAH	—	—	—	—	—	—	D9	D8

10-bit 标准型 TM 寄存器列表 (n=0~1)

• **CTMnC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CTnPAU:** CTMn 计数器暂停控制位

0: 运行
1: 暂停

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

Bit 6~4 **CTnCK2~CTnCK0:** CTMn 计数时钟选择位

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，细节方面请参考振荡器章节。

Bit 3 **CTnON:** CTMn 计数器 On/Off 控制位

0: Off
1: On

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

若 CTMn 处于比较匹配输出模式或 PWM 输出模式，当 CTnON 位经由低到高的转换时，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 CTM 计数器 bit 9~bit 7 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零内部计数器。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，比较值是 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**: CTMn 工作模式选择位

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

这两位设置 CTMn 需要的工作模式。为了确保操作可靠，CTMn 应在 CTnM1 和 CTnM0 位有任何改变前先关掉。在定时 / 计数器模式，CTMn 输出脚状态未定义。

Bit 5~4 **CTnIO1~CTnIO0**: CTMn 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

- 00: PWM 输出无效状态
- 01: PWM 输出有效状态
- 10: PWM 输出
- 11: 未定义

定时 / 计数器模式

未使用

此两位用于决定在满足特定条件时 CTMn 外部引脚如何改变状态。这两位值的选择取决于 CTMn 运行在哪种模式下。

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 决定比较匹配条件发生时怎样改变 CTMn 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。仅在 CTMn 关闭后才能改变 CTnIO1 和 CTnIO0 位的值。若在 CTMn 运行时改变 CTnIO1 和 CTnIO0 的值，PWM 输出的值将无法预料。

Bit 3 **CTnOC**: CTPn 输出控制位

比较匹配输出模式

- 0: 初始低
- 1: 初始高

PWM 输出模式

- 0: 低有效
- 1: 高有效

这是 CTMn 输出脚输出控制位。它取决于 CTMn 此时正运行于比较匹配输出模式还是 PWM 输出模式。若 CTMn 处于定时 / 计数器模式，则其无效。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 / 计数器模式时其无效。

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**: CTMn 计数器清零条件选择位
0: CTMn 比较器 P 匹配
1: CTMn 比较器 A 匹配

此位用于选择清除计数器的方法。简易型 TM 包括两个比较器 – 比较器 A 和比较器 P。这两个比较器每个都可以用于清除内部计数器。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
CMn 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

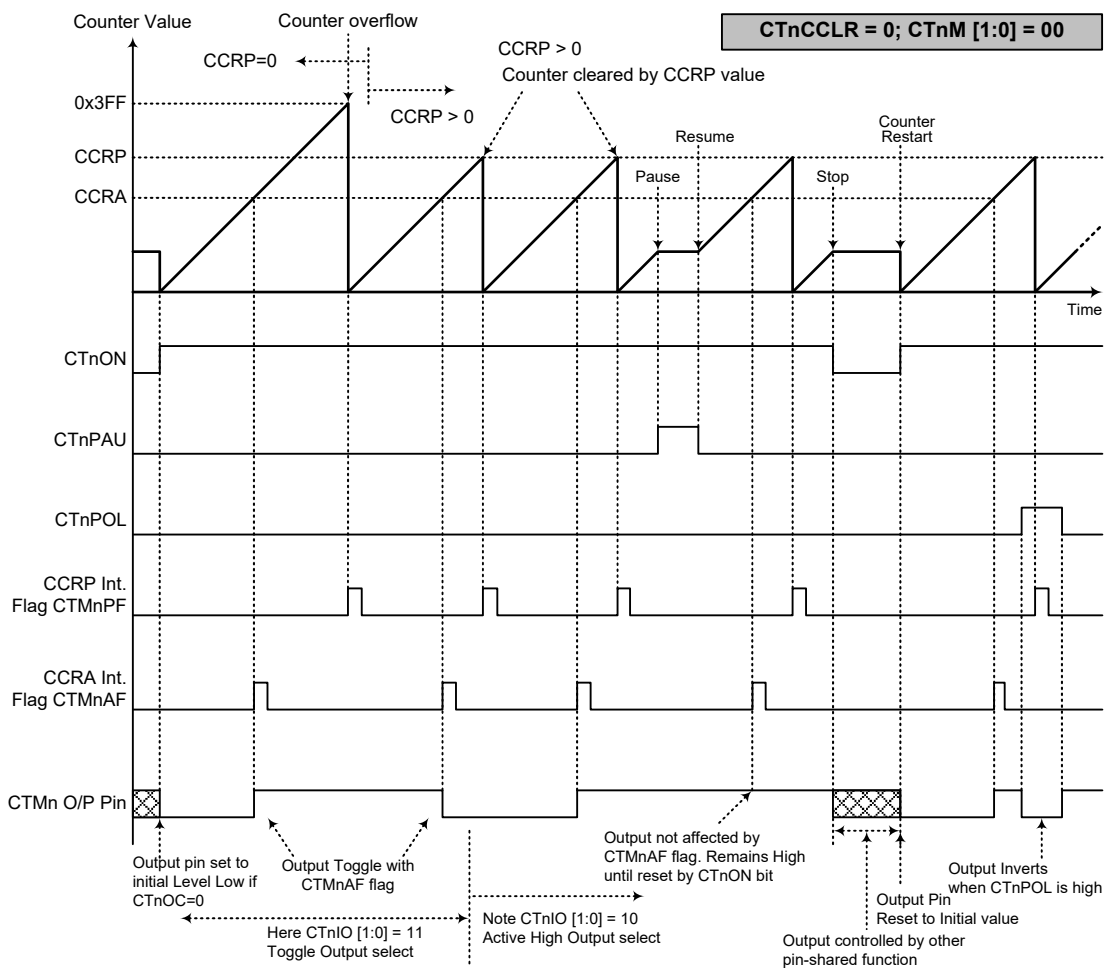
简易型 TM 有三种工作模式，即比较匹配输出模式，PWM 输出模式或定时 / 计数器模式。通过设置 CTMnC1 寄存器的 CTnM1 和 CTnM0 位选择任意模式。

比较匹配输出模式

为使 CTMn 工作在此模式，CTMnC1 寄存器中的 CTnM1 和 CTnM0 位需要设置为“00”。当工作在该模式，一旦计数器使能并开始计数，有三种方法来清零，分别是：计数器溢出，比较器 A 比较匹配发生和比较器 P 比较匹配发生。当 CTnCCLR 位为低，有两种方法清零计数器。一种是比较器 P 比较匹配发生，另一种是 CCRP 所有位设置为零并使得计数器溢出。此时，比较器 A 和比较器 P 的请求标志位 CTMnAF 和 CTMnPF 将分别置起。

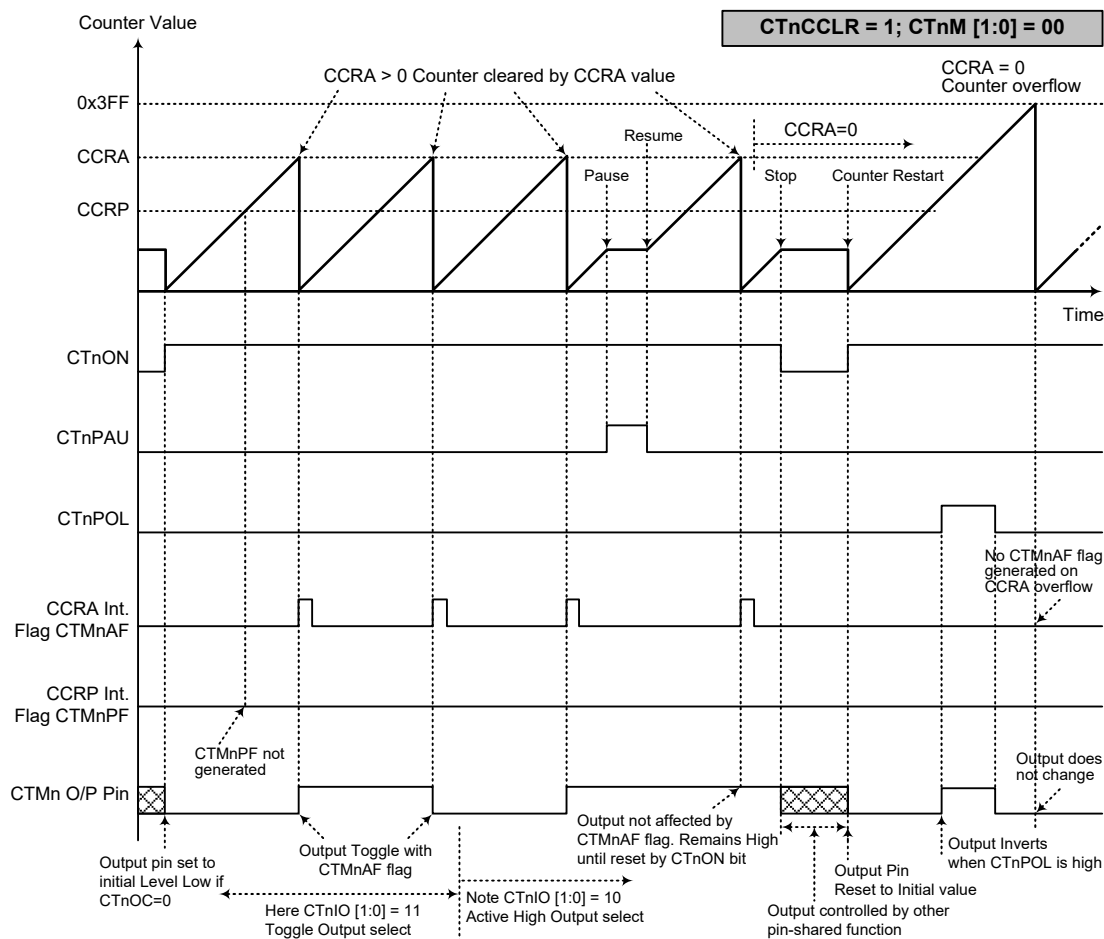
如果 CTMnC1 寄存器的 CTnCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 CTMnAF 中断请求标志产生。所以当 CTnCCLR 为高时，不产生 CTMnPF 中断请求标志。如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 CTMnAF 请求标志。

正如该模式名所言，当比较匹配发生后，CTMn 输出脚状态改变。当比较器 A 比较匹配发生后 CTMnAF 标志产生时，CTMn 输出脚状态改变。比较器 P 比较匹配发生时产生的 CTMnPF 标志不影响 CTMn 输出脚。CTMn 输出脚状态改变方式由 CTMnC1 寄存器中 CTnIO1 和 CTnIO0 位决定。当比较器 A 比较匹配发生时，CTnIO1 和 CTnIO0 位决定 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。



比较匹配模式 – CTnCCLR=0 (n=0~1)

- 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



比较匹配模式 – CTnCCLR=1 (n=0~1)

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

定时 / 计数器模式

为使 CTMn 工作在此模式，CTMnC1 寄存器中的 CTnM1 和 CTnM0 位需要设置为“11”。定时 / 计数器模式与比较匹配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 寄存器中的 CTnM1 和 CTnM0 位需要设置为“10”。CTMn 的 PWM 功能在马达控制，加热控制，照明控制等方面十分有用。给 CTMn 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就较为灵活。在 PWM 输出模式中，CTnCCLR 位不影响 PWM 操作。CCRA 和 CCRP 寄存器决定 PWM 波形，一个用来清零内部计数器并控制 PWM 波形的频率，另一个用来控制占空比。哪个寄存器控制频率或占空比取决于 CTMnC1 寄存器的 CTnDPX 位。所以 PWM 波形频率和占空比由 CCRA 和 CCRP 寄存器共同决定。

当比较器 A 或比较器 P 比较匹配发生时，将产生 CCRA 或 CCRP 中断标志。CTMnC1 寄存器中的 CTnOC 位决定 PWM 波形的极性，CTnIO1 和 CTnIO0 位使能 PWM 输出或将 CTMn 输出脚置为逻辑高或逻辑低。CTnPOL 位对 PWM 输出波形的极性取反。

• 10-bit CTMn, PWM 输出模式，边沿对齐模式，CTnDPX=0

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

若 $f_{SYS}=8\text{MHz}$ ，CTMn 时钟源选择 $f_{SYS}/4$ ，CCRP=4，CCRA=128，

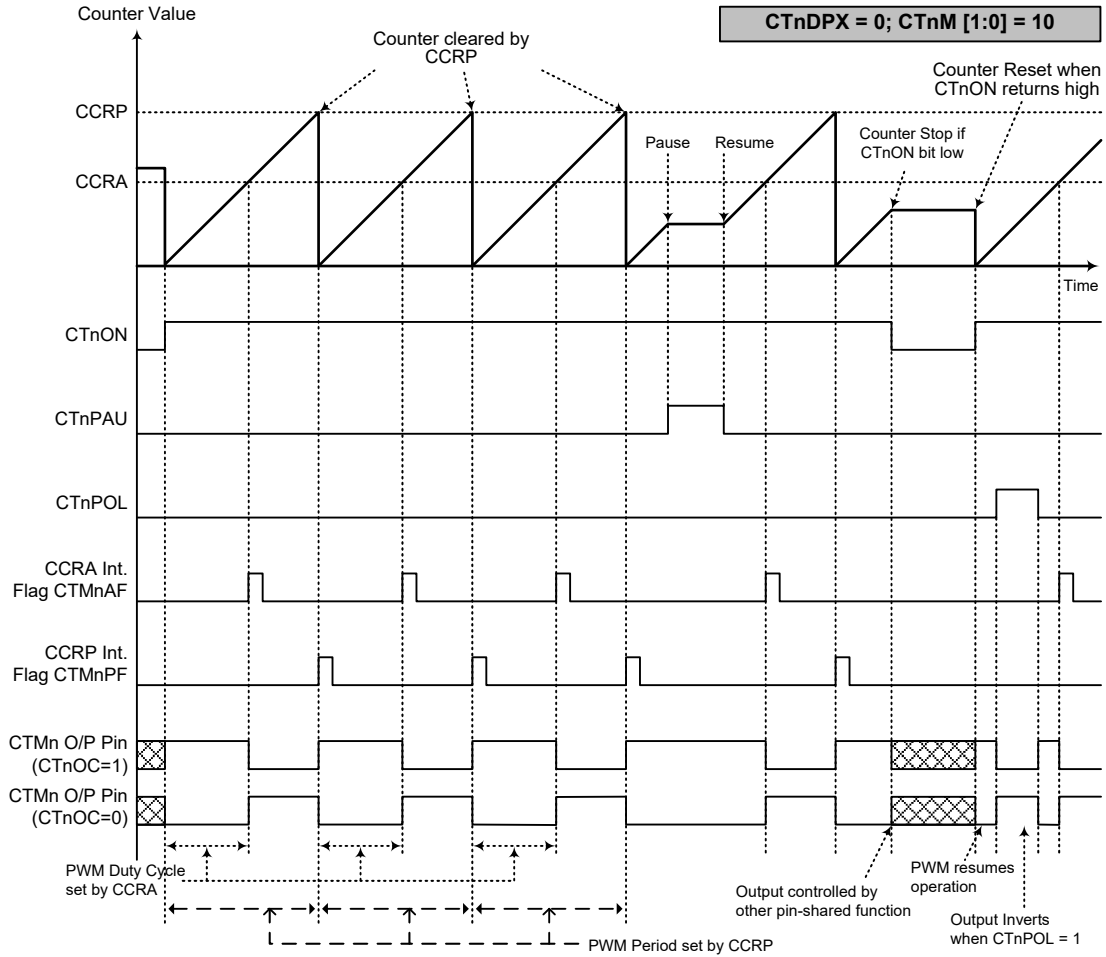
CTMn PWM 输出频率 = $(f_{SYS}/4)/(4 \times 128) = f_{SYS}/2048 = 3.9063\text{kHz}$ ，duty = $128/(4 \times 128) = 25\%$ 。

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 CTMn, PWM 输出模式，边沿对齐模式，CTnDPX=1

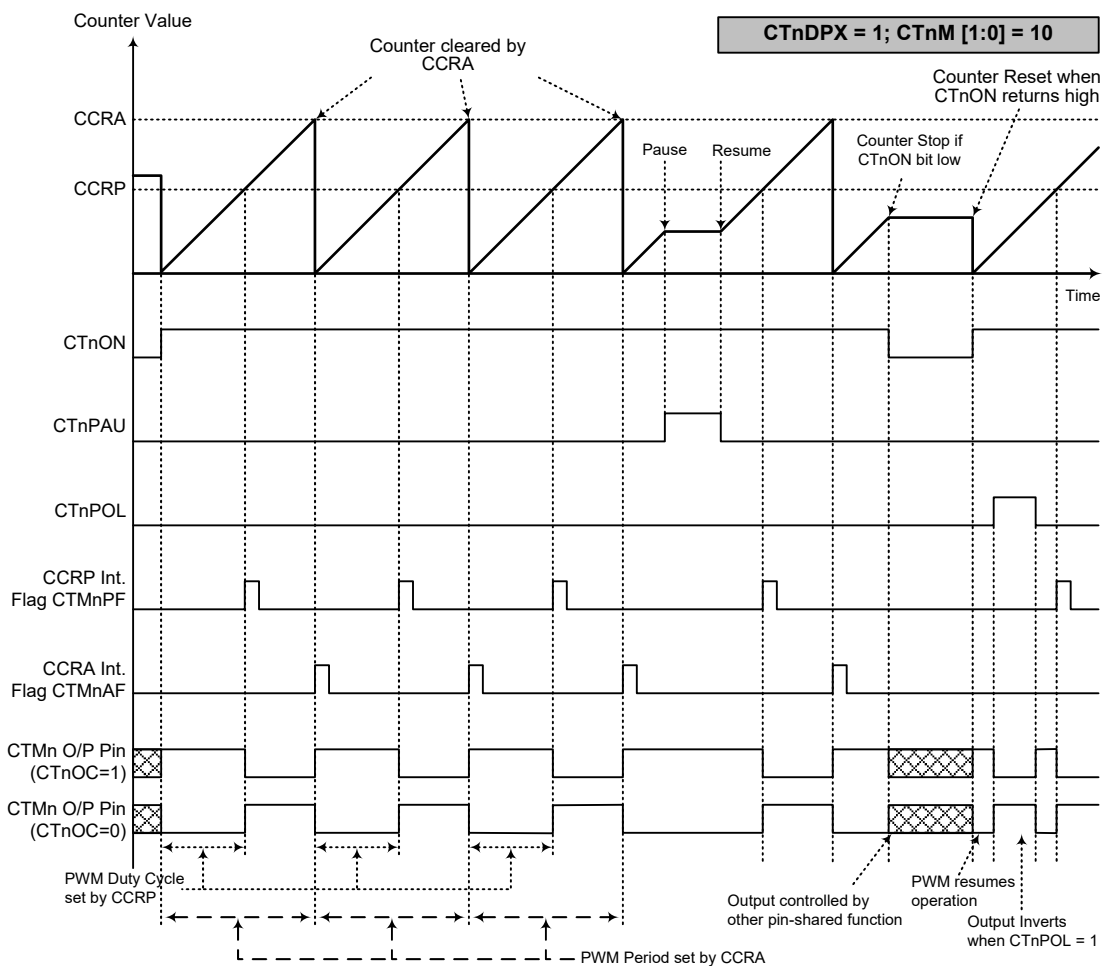
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

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=0~1)

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



PWM 输出模式 – CTnDPX=1 (n=0~1)

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

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	PTDPX	PTCCLR
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	D15	D14	D13	D12	D11	D10	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	D15	D14	D13	D12	D11	D10	D9	D8
PTMRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMRPH	D15	D14	D13	D12	D11	D10	D9	D8

16-bit 周期型 TM 寄存器列表

● PTMC0 寄存器

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

Bit 7 **PTPAU**: PTM 计数器暂停控制位

0: 运行
1: 暂停

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

Bit 6~4 **PTCK2~PTCK0**: PTM 计数时钟选择位

000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: PTCK 上升沿时钟
111: PTCK 下降沿时钟

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

Bit 3 **PTON**: PTM 计数器 On/Off 控制位

0: Off
1: On

此位控制 PTM 的总开关功能。设置此位为高则使能计数器使其运行，清零此位则除能 PTM。清零此位将停止计数器并关闭 PTM 减少耗电。当此位经由低到高转换时，内部计数器将复位清零；当此位经由高到低的转换时，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 位经由低到高的转换时，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	PTDPX	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, PTM 应在 PTM1 和 PTM0 位有任何改变前先关掉。在定时 / 计数器模式, PTM 输出脚状态未定义。

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

定时 / 计数器模式

未使用

此两位用于决定在满足特定条件时 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。

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 决定比较匹配条件发生时怎样改变 PTM 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。仅在 PTM 关闭后才可改变 PTIO1 和 PTIO0 位的值。若在 PTM 运行时改变 PTIO1 和 PTIO0 的值, PWM 输出的值将无法预料。

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, 则其无效。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 位由高变为低时 PTM 输出脚的逻辑电平值。

- 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 / 计数器模式时其无效。
- Bit 1** **PTDPX:** PTM 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** **PTCCLR:** PTM 计数器清零条件选择位
 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 TM 包括两个比较器 – 比较器 A 和比较器 P。这两个比较器每个都可以用于清除内部计数器。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 16-bit 计数器 bit 7 ~ bit 0

● **PTMDH 寄存器**

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

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

● PTMAH 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:** PTM CCRA 高字节寄存器 bit 7 ~ bit 0
PTM 16-bit CCRA bit 15 ~ 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 16-bit CCRP bit 7 ~ bit 0

● PTMRPH 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:** PTM CCRP 高字节寄存器 bit 15 ~ bit 8
PTM 16-bit CCRP bit 15 ~ 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

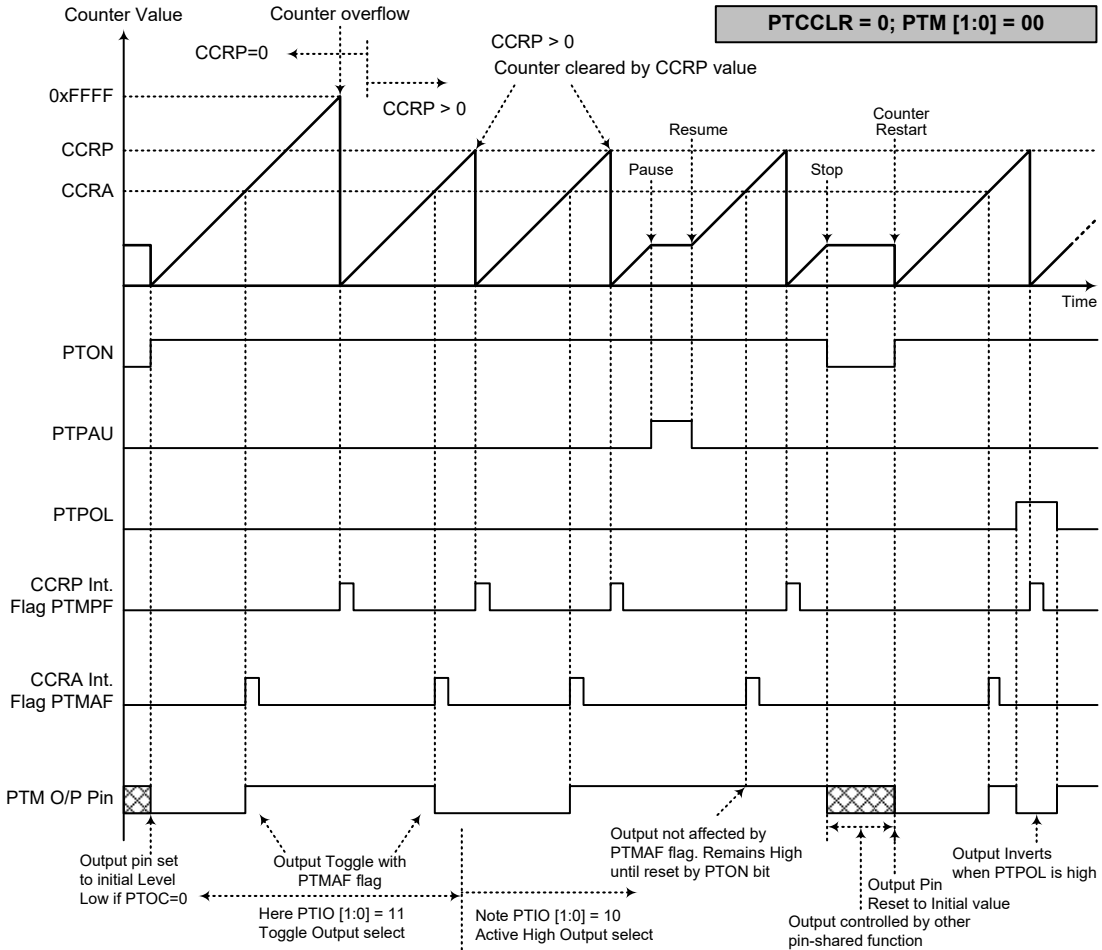
为使 PTM 工作在此模式，PTMC1 寄存器中的 PTM1 和 PTM0 位需要设置为“00”。当工作在该模式，一旦计数器使能并开始计数，有三种方法来清零，分别是：计数器溢出，比较器 A 比较匹配发生和比较器 P 比较匹配发生。当 PTCCLR 位为低，有两种方法清除计数器。一种是比较器 P 比较匹配发生，另一种是 CCRP 所有位设置为零并使得计数器溢出。此时，比较器 A 和比较器 P 的请求标志位 PTMAF 和 PTMPF 将分别置位。

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，仅产生 PTMAF 中断请求标志。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下，CCRA 不能设为“0”。

如果 CCRA 位都清除为零，当计数器的值达到 16 位最大值 FFFFH 时将溢出，但此时不会产生 PTMAF 中断请求标志。

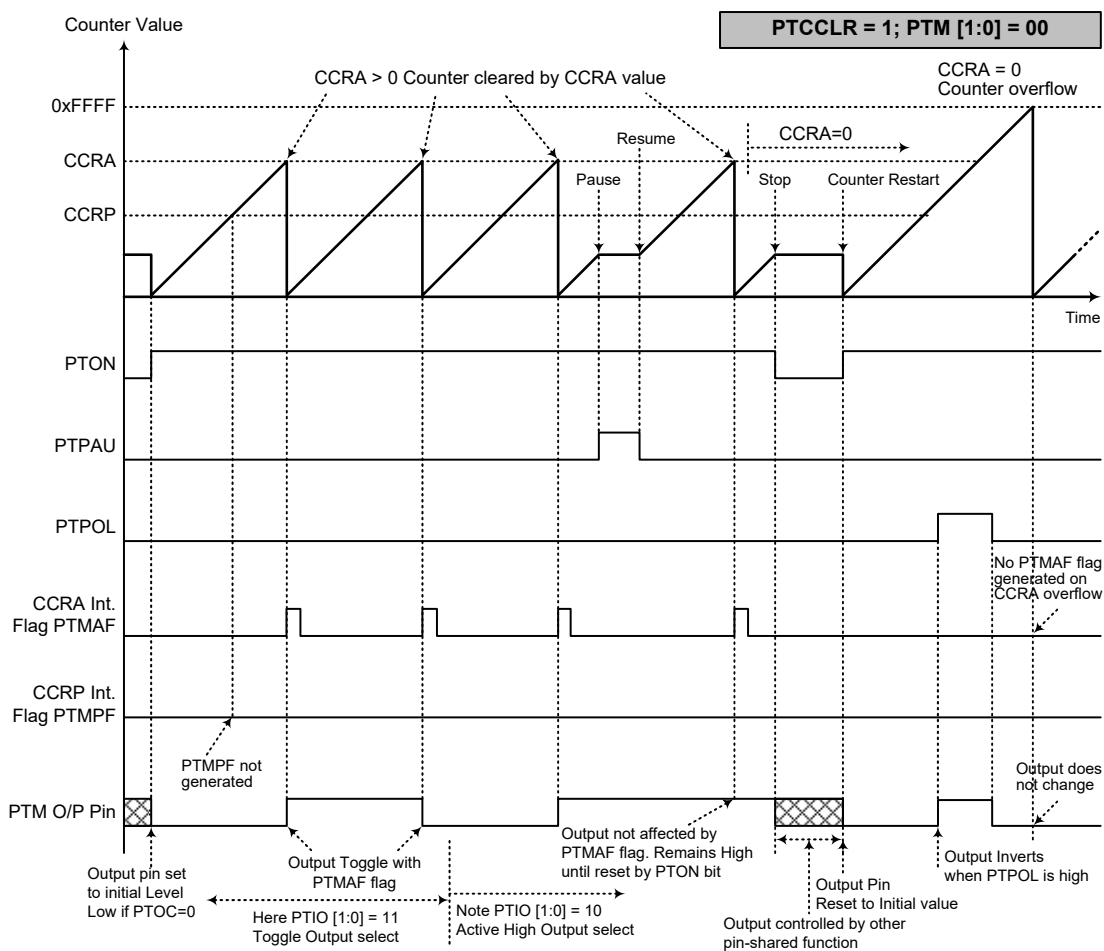
正如该模式名所言，当比较匹配发生后，PTM 输出脚状态改变。当比较器 A 比较匹配发生后 PTMAF 标志产生时，PTM 输出脚状态改变。比较器 P 比较匹配发生时产生的 PTMPF 标志不影响 PTM 输出脚。PTM 输出脚状态改变方式

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 位由低到高电平的变化后，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 寄存器中的 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 寄存器中的 PTM1 和 PTM0 位需要设置为“10”，且 PTIO1 和 PTIO0 位也需要设置为“10”。PTM 的 PWM 功能在马达控制，加热控制，照明控制等方面十分有用。给 PTM 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就较为灵活。在 PWM 输出模式中，PTCCLR 位不影响 PWM 周期。CCRA 和 CCRP 寄存器决定 PWM 波形，一个用来清除内部计数器并控制 PWM 波形的频率，另一个用来控制占空比。哪个寄存器控制频率或占空比取决于 PTMC1 寄存器的 PTDPX 位。所以 PWM 波形由 CCRA 和 CCRP 寄存器共同决定。

当比较器 A 或比较器 P 比较匹配发生时，将产生 CCRA 或 CCRP 中断标志。PTMC1 寄存器中的 PTOC 位决定 PWM 波形的极性，PTIO1 和 PTIO0 位使能 PWM 输出或将 PTM 输出脚置为逻辑高或逻辑低。PTPOL 位对 PWM 输出波形的极性取反。

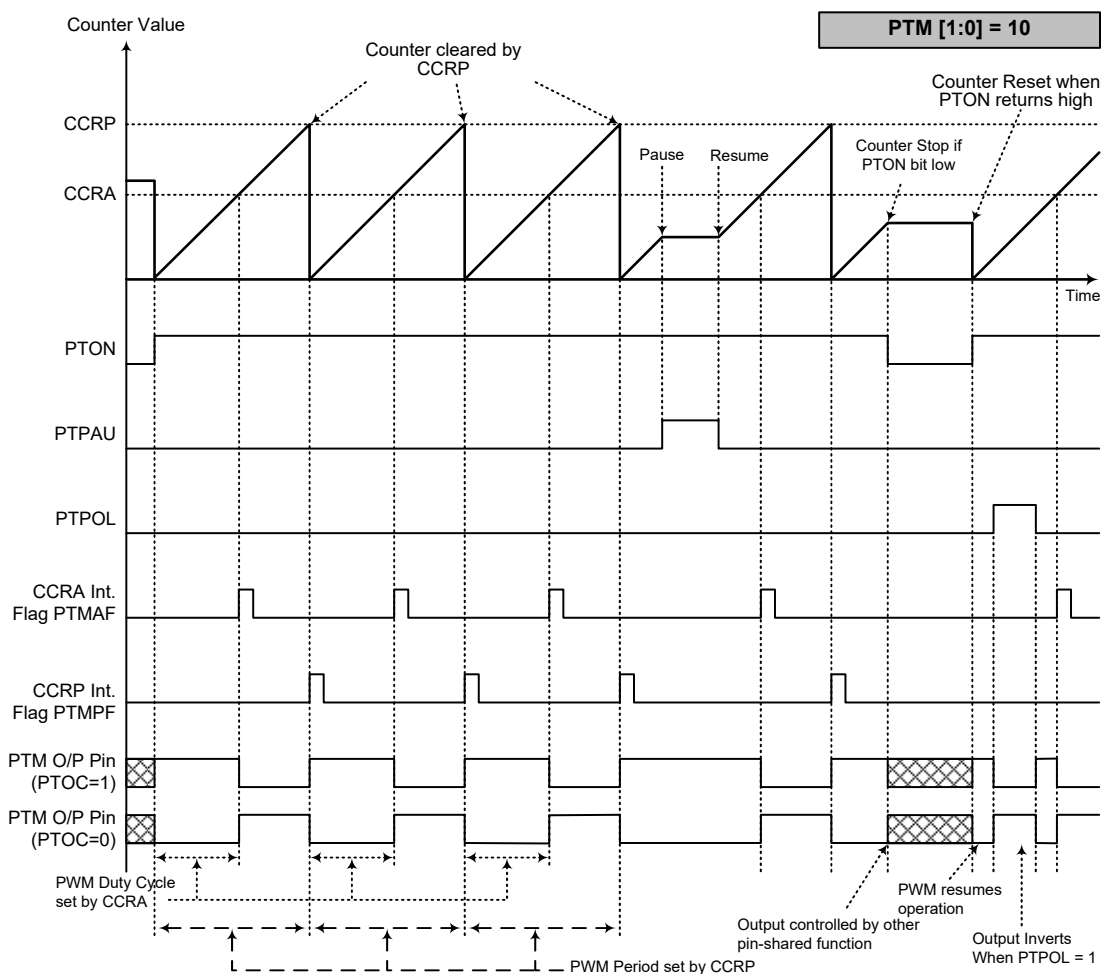
● 16-bit PTM, PWM 输出模式，边沿对齐模式

CCRP	1~65535	0
Period	1~65535	65536
Duty	CCRA	

若 $f_{SYS}=8\text{MHz}$ ，PTM 时钟源选择 $f_{SYS}/4$ ，CCRP=512，CCRA=128，

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

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



PWM 输出模式

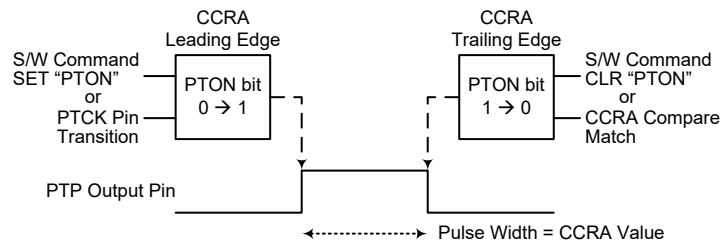
- 注：1. CCRP 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 PTIO[1:0]=00 或 01，PWM 功能不变
4. PTCCLR 位对 PWM 功能无影响

单脉冲输出模式

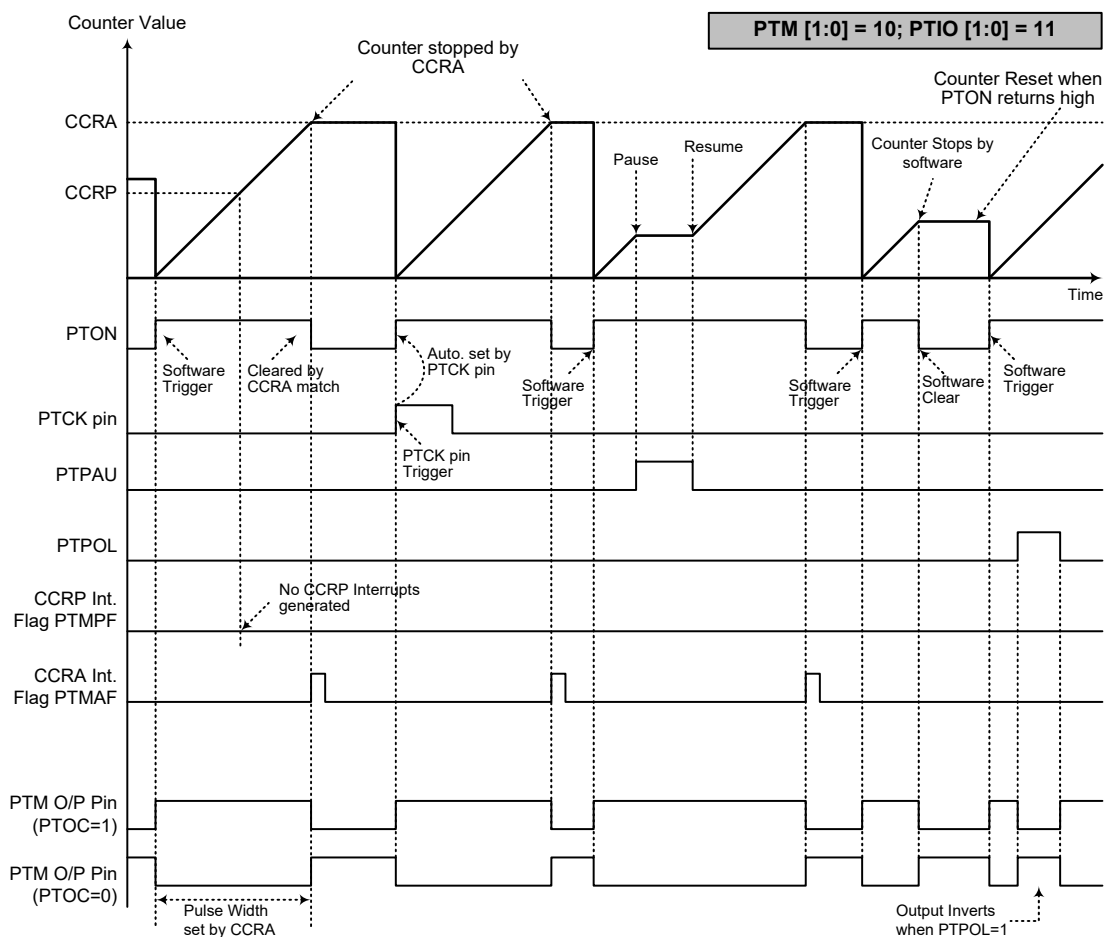
为使 PTM 工作在此模式，PTMC1 寄存器中的 PTM1 和 PTM0 位需要设置为“10”，同时 PTIO1 和 PTIO0 位需要设置为“11”。正如模式名所言，单脉冲输出模式，在 PTM 输出脚将产生一个脉冲输出。

脉冲输出可以通过应用程序控制 PTON 位由低到高的转变来触发。而处于单脉冲输出模式时，PTON 位可在 PTCK 脚发生有效边沿跳转时自动由低转变为高，进而开始单脉冲输出。当 PTON 位转变为高电平时，计数器将开始运行，并产生脉冲前沿。当脉冲有效时 PTON 位保持高电平。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。

然而，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、PTCCLR 和 PTDPX 位未使用。



单脉冲产生示意图



单脉冲输出模式

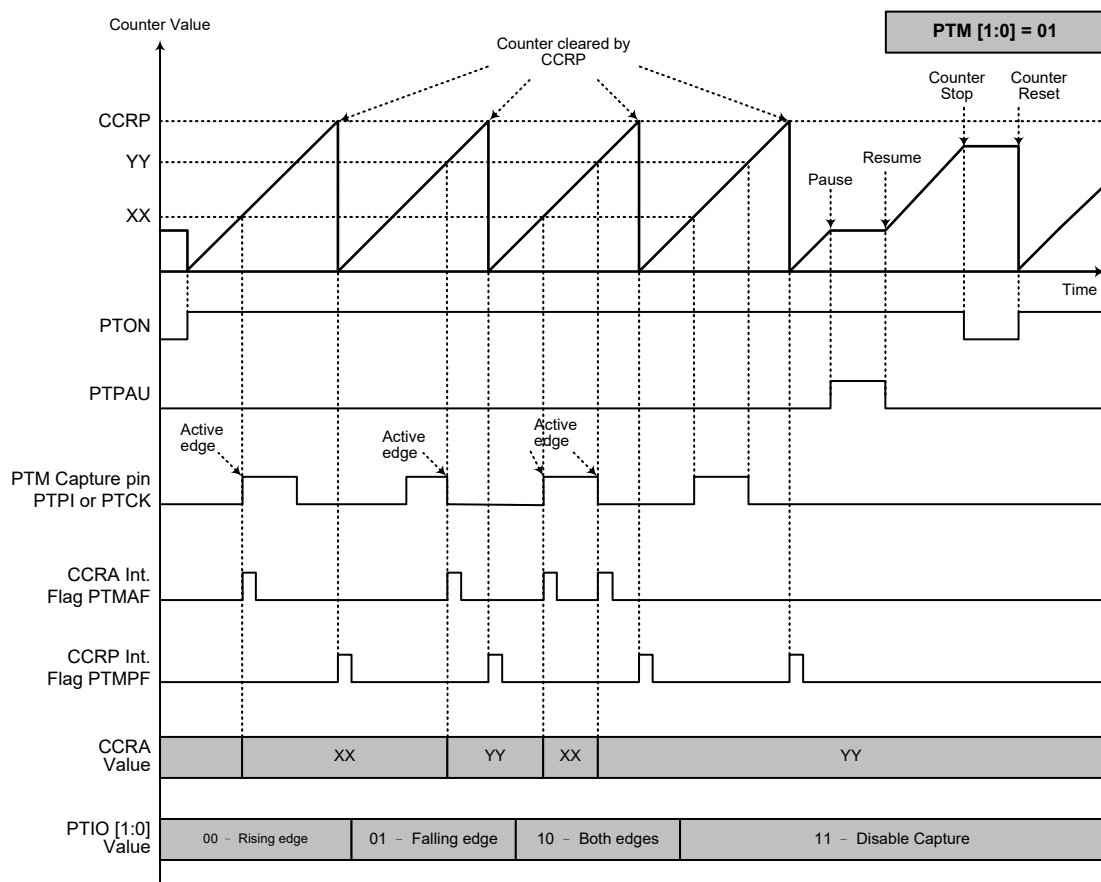
- 注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 PTCK 脚或设置 PTON 位为高来触发脉冲
4. PTCK 脚有效沿会自动置高位 PTON
5. 单脉冲输出模式中，PTIO[1:0] 需置位“11”，且不能更改

捕捉输入模式

为使 PTM 工作在此模式，PTMC1 寄存器中的 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。PTPI 或 PTCK 引脚上的外部信号，通过设置 PTMC1 寄存器的 PTCAPTS 位选择。可通过设置 PTMC1 寄存器的 PTIO1 和 PTIO0 位选择有效边沿类型，即上升沿，下降沿或双沿有效。通过应用程序将 PTON 位由低到高转变时，计数器启动。

当 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。无论 PTPI 或 PTCK 引脚发生哪种边沿转换，计数器将继续工作直到 PTON 位发生下降沿跳变。当 CCRP 比较匹配发生时计数器复位至零；CCRP 的值通过这种方式控制计数器的最大值。当比较器 P CCRP 比较匹配发生时，也会产生 PTM 中断。记录 CCRP 溢出中断信号的值可以测量长脉宽。通过设置 PTIO1 和 PTIO0 位选择 PTPI 或 PTCK 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。

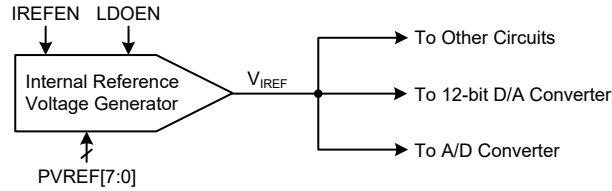


捕捉输入模式

- 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. 捕捉输入模需在有 PTM 计数时钟的情况下才可使用

内部参考电压发生器

该单片机包含一个内部参考电压发生器，以提供精准的参考电压 V_{IREF} 用于 24-bit Delta Sigma A/D 转换器或 12-bit D/A 转换器，或通过 DACVREF 引脚输出提供给其他电路使用，详细请参阅内部参考电压特性章节。



注：内部参考电压发生器由 IREFEN 位和 LDOEN 位控制。

IREFEN	LDOEN	内部参考电压发生器
0	0	OFF
0	1	ON
1	0	ON
1	1	ON

内部参考电压寄存器介绍

内部参考电压发生器由两个寄存器控制。IREFC 寄存器用于使能 / 除能控制，PVREF 寄存器用于对参考电压就行微调。

寄存器名称	位							
	7	6	5	4	3	2	1	0
IREFC	—	—	—	IREFEN	—	—	DACVRS1	DACVRS0
PVREF	D7	D6	D5	D4	D3	D2	D1	D0

内部参考电压寄存器列表

• IREFC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	IREFEN	—	—	DACVRS1	DACVRS0
R/W	—	—	—	R/W	—	—	R/W	R/W
POR	—	—	—	0	—	—	0	0

Bit 7~5 未定义，读为“0”

Bit 4 **IREFEN**: 内部参考电压发生器控制位

0: 除能

1: 使能

此位用于控制内部参考电压发生器的开 / 关。当此位置 1 时，内部参考电压 V_{IREF} 可用作参考电压。若此参考电压不提供给任何电路使用，应将此位清零以节省功耗。

Bit 3~2 未定义，读为“0”

Bit 1~0 **DACVRS1~DACVRS0**: 12-bit D/A 转换器参考电压 V_{DACVREF} 选择位

详细描述见 D/A 转换器章节。

● PVREF 寄存器

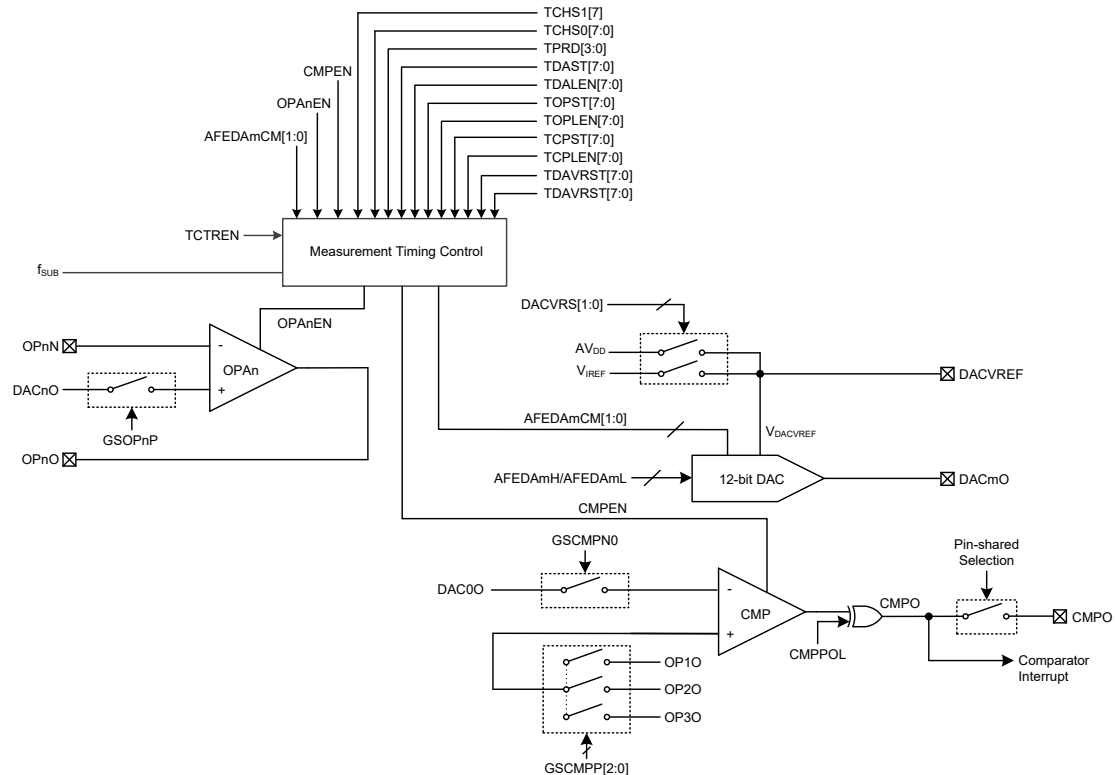
Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0:** 内部参考电压发生器微调控制

通过设置此寄存器可以对内部参考电压进行微调, 调整幅度为 $-60\text{mV}\sim+60\text{mV}$ (基于 $\text{PVREF}=80\text{H}$)。PVREF 寄存器的值每增加一, 输出的参考电压将减少约 $500\mu\text{V}$; 反之, 每减少一, 将增加约 $500\mu\text{V}$ 。

测量电路

该单片机包含一个测量电路, 包含 4 个 12-bit D/A 转换器、3 个运算放大器和 1 个比较器。每个 D/A 转换器可提供 $0\sim\text{V}_{\text{DACVREF}}$ 的电压给 OPA 正输入端使用。D/A 转换器输出电压 V_{DACO} 可通过 A/D 转换器测量。经合理的设置可将 OPA 的输出电压内部连接到 A/D 转换器输入通道进行测量。当检测到特定的 OPA 输入信号时, 比较器中断请求标志 CMPF 被置位, 比较器中断请求产生。



测量电路方框图 ($m=0\sim3$, $n=1\sim3$)

测量电路寄存器介绍

测量电路由一系列寄存器控制。寄存器 PRFC、TOPST、TDAST、TCPST、TDAVRST、TOPLEN、TDALEN、TCPLEN、TDAVRLEN、TCHS0 和 TCHS1 用于计时控制。寄存器 IREFC、AFEDAmC、AFEDAmL 和 AFEDAmH 用于 DAC 控制。寄存器 OPAnC 和 GSC1 用于 OPA 控制。CMPC 寄存器用于比较器控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PRFC	TCTREN	—	—	—	TPRD3	TPRD2	TPRD1	TPRD0
TOPST	D7	D6	D5	D4	D3	D2	D1	D0
TOPLEN	D7	D6	D5	D4	D3	D2	D1	D0
TDAVRST	D7	D6	D5	D4	D3	D2	D1	D0
TDAVRLEN	D7	D6	D5	D4	D3	D2	D1	D0
TDAST	D7	D6	D5	D4	D3	D2	D1	D0
TDALEN	D7	D6	D5	D4	D3	D2	D1	D0
TCPST	D7	D6	D5	D4	D3	D2	D1	D0
TCPLEN	D7	D6	D5	D4	D3	D2	D1	D0
TCHS0	TCPEN	TDA3EN	TDA2EN	TDA1EN	TDA0EN	TOP3EN	TOP2EN	TOP1EN
TCHS1	TDAVRLEN	—	—	—	—	—	—	—
IREFC	—	—	—	IREFEN	—	—	DACVRS1	DACVRS0
AFEDAmC	—	—	—	—	—	—	AFEDAmCM1	AFEDAmCM0
AFEDAmL	D3	D2	D1	D0	—	—	—	—
AFEDAmH	D11	D10	D9	D8	D7	D6	D5	D4
OPAnC	—	—	OPAnEN	—	—	—	—	—
GSC1	—	—	—	—	—	GSOP3P	GSOP2P	GSOP1P
CMPC	—	CMPPOL	CMPO	CMPEN	GSCMPN0	GSCMPP2	GSCMPP1	GSCMPP0

测量电路寄存器列表 (m=0~3, n=1~3)

● PRFC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TCTREN	—	—	—	TPRD3	TPRD2	TPRD1	TPRD0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **TCTREN**: 计时控制使能 / 除能

0: 使能

1: 除能

Bit 6~4 未定义, 读为 “0”

Bit 3~0 **TPRD3~TPRD0**: 计时控制时钟周期选择 ($f_{PRF}=f_{SUB}/16384$)

0000: f_{PRF}

0001: $2/f_{PRF}$

0010: $3/f_{PRF}$

0011: $4/f_{PRF}$

0100: $5/f_{PRF}$

0101: $6/f_{PRF}$

0110: $7/f_{PRF}$

0111: $8/f_{PRF}$

1000: $9/f_{PRF}$

1001: $10/f_{PRF}$

1010: $11/f_{PRF}$

1011: $12/f_{PRF}$

1100: $13/f_{PRF}$

1101: $14/f_{PRF}$

1110: $15/f_{PRF}$

1111: $16/f_{PRF}$

• TOPST 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:** OPA 启动延迟时间

注：这个寄存器可以定义多达 255 个节拍。f_{base}=f_{SUB}/128。

当 TCTREN 位为高时，TOPST 可设定计时控制电路延迟几个节拍，然后启动 OPA。例如，1 个节拍的时间是 1/f_{base}，设置 TOPST=4，则计时控制将会延迟 4 个 1/f_{base} 后启动 OPA。

• TOPLEN 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:** OPA 开启持续时间

注：这个寄存器可以定义多达 255 个节拍。f_{base}=f_{SUB}/128。

当 TCTREN 位为高时，TOPLEN 可以设定计时控制电路开启 OPA，使它持续开启几个节拍，然后关闭 OPA。例如一个节拍的时间是 1/f_{base}，设置 TOLEN=50，则计时控制开启 OPA，持续 50 个 1/f_{base} 后关闭 OPA。

• TDAVRST 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:** DACVREF 启动延迟时间

注：与 TOPST 寄存器功能相同，TDAST 寄存器用于控制 DACVREF 启动延迟时间。

• TDAVRLEN 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:** DACVREF 开启持续时间

注：与 TOPLEN 寄存器功能相同，TDALEN 寄存器用于控制 DACVREF 开启持续时间。

• **TDAST 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:** DAC 启动延迟时间

注：与 TOPST 寄存器功能相同，TDAST 寄存器用于控制 DAC 启动延迟时间。

• **TDALEN 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:** DAC 开启持续时间

注：与 TOPLEN 寄存器功能相同，TDALEN 寄存器用于控制 DAC 开启持续时间。

• **TCPST 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:** 比较器启动延迟时间

注：与 TOPST 寄存器功能相同，TCPST 寄存器用于控制比较器启动延迟时间。

• **TCPLEN 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:** 比较器开启持续时间

注：与 TOPLEN 寄存器功能相同，TCPLEN 寄存器用于控制比较器开启持续时间。

• **TCHS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TCPEN	TDA3EN	TDA2EN	TDA1EN	TDA0EN	TOP3EN	TOP2EN	TOP1EN
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 **TCPEN:** 比较器计时控制

0: 除能

1: 使能

Bit 6 **TDA3EN:** DAC3 计时控制

0: 除能

1: 使能

Bit 5 **TDA2EN:** DAC2 计时控制

0: 除能

1: 使能

Bit 4	TDA1EN: DAC1 计时控制 0: 除能 1: 使能
Bit 3	TDA0EN: DAC0 计时控制 0: 除能 1: 使能
Bit 2	TOP3EN: OPA3 计时控制 0: 除能 1: 使能
Bit 1	TOP2EN: OPA2 计时控制 0: 除能 1: 使能
Bit 0	TOP1EN: OPA1 计时控制 0: 除能 1: 使能

注: 1. 当 TCTREN 位为低时, 此寄存器除能。DAC、OPA 和 CMP 功能分别由 AFEDAmCM[1:0]、OPAnEN 和 CMPEN 位控制 (m=0~3, n=1~3)。
2. 当 TCTREN 位为高时, 对于 DAC、OPA 和 CMP 功能, TCHS0 寄存器比 AFEDAmCM[1:0]、OPAnEN 和 CMPEN 位具有较高的控制优先权。
例如 (TCTREN=1):
当 TOP1EN=1, 无论 OPA1EN 位为 0 或 1, OPA1 都会受到周期性的计时控制 (开启或关闭)。
当 TOP1EN=0, OPA1 则会受到 OPA1EN 位的设置恒定开或恒定关。
当 TDAmEN (m=0~3)=0 时, 相应的 DAC 除能, 对应的 DAC 输出为接地状态。

● TCHS1 寄存器

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

Bit 7 **TDAVREN:** DACVREF 计时控制
0: 除能
1: 使能

Bit 6~0 未定义, 读为 “0”

注: 1. 当 TCTREN 位为低时, 此寄存器除能, DACVREF 由 IREFEN 位控制。
2. 当 TCTREN 位为高时, 对于 DACVREF, TCHS1 寄存器比 IREFEN 位具有较高的优先权。

● IREFC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	IREFEN	—	—	DACVRS1	DACVRS0
R/W	—	—	—	R/W	—	—	R/W	R/W
POR	—	—	—	0	—	—	0	0

Bit 7~5 未定义, 读为 “0”

Bit 4 **IREFEN:** 内部参考电压发生器控制
具体描述见内部参考电压发生器章节

Bit 3~2 未定义, 读为 “0”

Bit 1~0 **DACVRS1~DACVRS0:** 12-bit D/A 转换器参考电压 $V_{DACVREF}$ 选择
00/01: V_{IREF}
10: AV_{DD}
11: 浮空

● AFEDAmC 寄存器 (m=0~3)

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	AFEDAmCM1	AFEDAmCM0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

Bit 1~0 **AFEDAmCM1~AFEDAmCM0**: 12-bit D/A 转换器模式控制位
 00: DAC 除能，输出处于高阻抗状态
 01/11: DAC 使能
 10: DAC 除能，输出为接地状态

● AFEDAmH & AFEDAmL 寄存器 (m=0~3)

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

“—”：未定义，读为“0”

D11~D0: 12-bit D/A 转换器输出控制位
 AFEDAmH 寄存器的 bit 7 ~ bit 0 结合 AFEDAmL 寄存器的 bit 7 ~ bit 4 形成了一个 12-bit DAC 值，范围为 0 ~ 4095。
 DAC 输出电压 $V_{DACO} = V_{DACVREF} \times (DAC \text{ 值} / 4096)$
 注：需先写入 AFEDAmL 后再写入 AFEDAmH 寄存器。

● OPAnC 寄存器 (n=1~3)

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

Bit 7~6 未定义，读为“0”

Bit 5 **OPAnEN**: OPAn 使能 / 除能控制
 0: 除能
 1: 使能

Bit 4~0 未定义，读为“0”

● GSC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	GSOP3P	GSOP2P	GSOP1P
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 未定义，读为“0”

Bit 2 **GSOP3P**: OP3P 选择开关用于接 DAC3O 引脚
 0: Off
 1: On

Bit 1 **GSOP2P**: OP2P 选择开关用于接 DAC2O 引脚
 0: Off
 1: On

Bit 0 **GSOP1P**: OP1P 选择开关用于接 DAC1O 引脚
0: Off
1: On
注: 当 OPAnEN 使能时, GSOPnP 必须使能。

• CMPC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	CMPPOL	CMPO	COMPEN	GSCMPN0	GSCMPP2	GSCMPP1	GSCMPP0
R/W	—	R/W	R	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 **CMPPOL**: 比较器输出极性控制位
0: 同相
1: 反相
此位控制比较器输出极性。为 “0” 时, CMPO 反映比较器同相后输出; 为 “1” 时, CMPO 位反映比较器反相后输出。

Bit 5 **CMPO**: 比较器输出位
CMPPOL=0
0: CMPINP < CMPINN
1: CMPINP > CMPINN
CMPPOL=1
0: CMPINP > CMPINN
1: CMPINP < CMPINN
此位为比较器输出位。此位的极性由比较器输入电压和 CMPPOL 位的状态决定。

Bit 4 **COMPEN**: 比较器使能 / 使能控制
0: 除能
1: 使能
当比较器输出由低到高, 比较器会产生一个中断。

Bit 3 **GSCMPN0**: CMP 选择开关用于接 DAC0O 引脚
0: 除能
1: 使能

Bit 2 **GSCMPP2**: CMP 选择开关用于接 OP3O 引脚
0: 除能
1: 使能

Bit 1 **GSCMPP1**: CMP 选择开关用于接 OP2O 引脚
0: 除能
1: 使能

Bit 0 **GSCMPP0**: CMP 选择开关用于接 OP1O 引脚
0: 除能
1: 使能

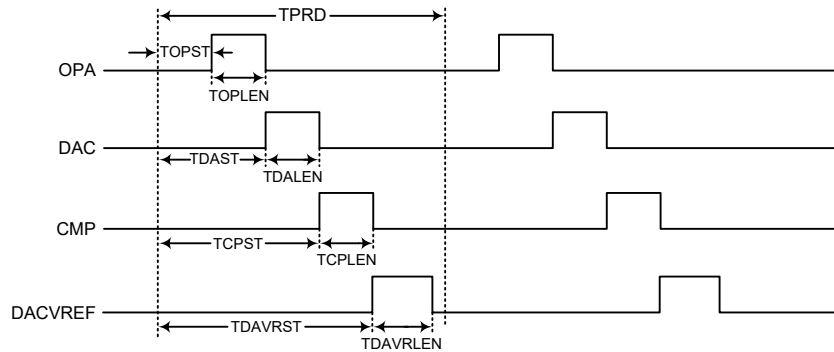
注: 当 COMPEN 使能时, GSCMPN0 必须使能。GSCMPP2、GSCMPP1 和 GSCMPP0 同时只能有一个导通。

测量计时控制

计时控制通过硬件来周期性地启动各组 DAC、DACVREF、OPA 和 CMP 进行应用。

TPRD3~TPRD0 位用于设置计时控制的动作周期, 最低持续时间为 0.5s。
TOPST、TDAVRST、TDAST 和 TCPST 寄存器用于设置控制周期开始后第几个节拍启动 OPA、DACVREF、DAC 和 CMP。

TOPLEN、TDAVRLEN、TDALEN 和 TCPLN 寄存器用于设置 DAC、DACVREF、OPA 和 CMP 的开启持续时间。其中有 4 个 DAC 转换器，DAC3~DAC0，3 个运算放大器，OPA1~OPA3。TCHS0 和 TCHS1 寄存器可以用来设置计时控制的对象。但是假设 TCHS0 设置要启动 DAC0 和 DAC2，由于这两个功能的启动延迟时间和开启持续时间都是由 TDAST 和 TDALEN 寄存器控制的，所以 DAC0 和 DAC2 的启动延迟时间和开启持续时间是相同的。同理，OPA 也一样。当完整的周期结束时，即 TPRD 计数结束，则会强制关闭所有 DAC、OPA、CMP 和 DACVREF，重新执行新的周期动作。



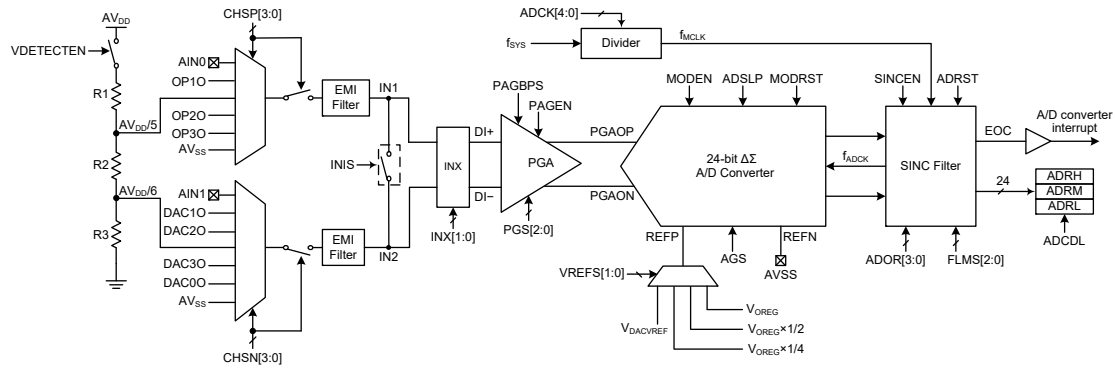
A/D 转换器

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

此单片机包含一个高精度多通道的 24-bit Delta Sigma 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）并直接将这些信号转换成 24 位的数字量。

另外，A/D 转换器输入信号的放大增益由 PGA 增益控制和 A/D 转换器增益控制共同确定。设计者可以选择较佳增益组合为输入信号提供所需的放大增益。下面的方框图说明了 A/D 转换器的基本操作功能。A/D 转换器外部输入通道由 2 个单端 A/D 输入通道或 1 组差分输入通道组成。在 PGA 进入 24 位 Delta Sigma A/D 转换器之前，输入信号被放大。Delta Sigma A/D 转换调制器将 1-bit 转换后的数据输出到 SINC 滤波器，然后会转换成 24-bit 的数据，并将它们存储到特殊数据寄存器。这种高精度和高性能的特点，使得该单片机非常适用于体重秤及相关产品。

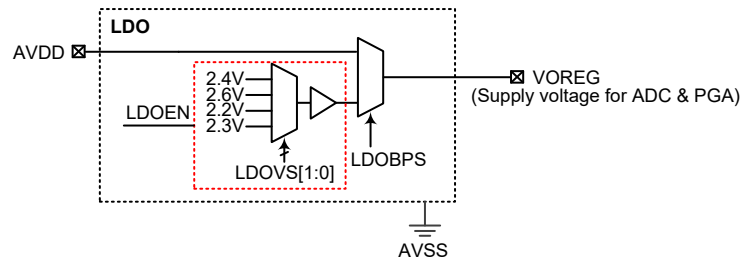


注：PGA 和 A/D 转换器是由 V_{REG} 和 AV_{SS} 供电。SINC 滤波器是由 V_{DD} 和 V_{SS} 供电。

A/D 转换器结构

内部电源供电

该单片机内部集成一个 LDO，用于产生稳定的电源电压，其基本功能操作如下图所示。内部的 LDO 电路为 PGA、A/D 转换器或外部器件提供了一个固定电压。LDO 可提供 2.4V、2.6V、2.2V 或 2.3V 四个输出电压，通过 PWRC 寄存器中的 LDOVS1~LDOVS0 位选择。LDO 功能可由 LDOEN 位控制，可将其关闭以降低功耗。



内部电源供电方框图

• PWRC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	LDOEN	LDOSTS	—	—	VDETECTEN	LDOBPS	LDOVS1	LDOVS0
R/W	R/W	R/W	—	—	R/W	R/W	R/W	R/W
POR	0	0	—	—	0	0	0	0

Bit 7 **LDOEN**: LDO 功能控制位

0: 除能
1: 使能

如果 LDO 除能，将不产生功耗，LDO 会在一个内部小下拉电阻的作用下输出低电平。

Bit 6 **LDOSTS**: 当 LDO 除能时，LDO 输出带下拉电阻或浮空控制

0: 带下拉电阻
1: 浮空

注：当 LDO 旁路模式时，将此位设为高选择浮空可降低功耗。

Bit 5~4 未定义，读为“0”

Bit 3 **VDETECTEN**: 分压器 $AV_{DD}/5$ 和 $AV_{DD}/6$ 控制位

0: 除能
1: 使能

- Bit 2 **LDOBPS**: LDO 旁路功能控制位
0: 除能
1: 使能
- Bit 1~0 **LDOVS1~LDOVS0**: LDO 输出电压选择位
00: 2.4V
01: 2.6V
10: 2.2V
11: 2.3V

A/D 转换寄存器介绍

A/D 转换器的所有工作由一系列寄存器控制。3 个只读寄存器用来存放 24 位 A/D 转换数据的值。剩下的控制寄存器设置增益和 A/D 转换器的功能控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PWRC	LDOEN	LDOSTS	—	—	VDETECTEN	LDOBPS	LDOVS1	LDOVS0
PGAC0	—	VREFS1	VREFS0	—	AGS	PGS2	PGS1	PGS0
PGAC1	—	INIS	INX1	INX0	—	—	PGABPS	PGAEN
PGACS	CHSN3	CHSN2	CHSN1	CHSN0	CHSP3	CHSP2	CHSP1	CHSP0
ADRL	D7	D6	D5	D4	D3	D2	D1	D0
ADRM	D15	D14	D13	D12	D11	D10	D9	D8
ADRH	D23	D22	D21	D20	D19	D18	D17	D16
ADCR0	ADIRST	—	SINCEN	ADOR3	ADOR2	ADOR1	ADOR0	—
ADCR1	FLMS2	FLMS1	FLMS0	—	—	ADCDL	EOC	—
ADCR2	D7	D6	—	MODEN	D3	D2	D1	D0
ADCR3	MODRST	D6	D5	D4	—	—	—	D0
ADCS	—	—	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0
SINC3	D7	D6	D5	D4	D3	D2	R_FILSEL	R_CKCHOP

A/D 转换寄存器列表

可编程增益放大器寄存器 – PGAC0, PGAC1, PGACS

有三个与可编程增益相关的控制寄存器，PGAC0、PGAC1 和 PGACS。PGAC0 寄存器用于选择 PGA 增益、A/D 转换器增益和 A/D 转换器参考电压。PGAC1 寄存器用于定义输入端连接、PGA 使能 / 除能控制和 PGA 旁路控制。PGACS 寄存器用于选择 PGA 的输入端信号。因此，必须通过 CHSP3~CHSP0 和 CHSN3~CHSN0 位来选择模拟输入通道、OPA 输出、DAC 输出或内部电源中的哪些被连接到内部差分 A/D 转换器。

● PGAC0 寄存器

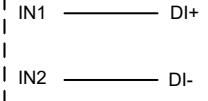
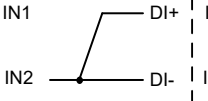
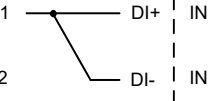
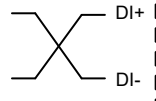
Bit	7	6	5	4	3	2	1	0
Name	—	VREFS1	VREFS0	—	AGS	PGS2	PGS1	PGS0
R/W	—	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	—	0	0	—	0	0	0	0

- Bit 7 未定义，读为 “0”
- Bit 6~5 **VREFS1~VREFS0**: A/D 转换器参考电压选择
00: $V_{DACVREF}$
01: V_{OREG}
10: $V_{OREG} \times 1/2$
11: $V_{OREG} \times 1/4$

- Bit 4 未定义，读为“0”
- Bit 3 **AGS**: A/D 转换器 PGAOP/PAGON 差分输入信号增益选择位
0: ADGN=1
1: ADGN=2
- Bit 2~0 **PGS2~PGS0**: PGA DI+/DI- 差分通道输入增益选择位
000: PGAGN=1
001: PGAGN=2
010: PGAGN=4
011: PGAGN=8
100: PGAGN=16
101: PGAGN=32
110: PGAGN=64
111: PGAGN=128

● PGAC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	INIS	INX1	INX0	—	—	PGABPS	PGAEN
R/W	—	R/W	R/W	R/W	—	—	R/W	R/W
POR	—	0	0	0	—	—	0	0

- Bit 7 未定义，读为“0”
- Bit 6 **INIS**: 选择输入端 IN1 和 IN2 内部连接控制位
0: 不连接
1: 连接
- Bit 5~4 **INX1~INX0**: 选择输入端 IN1/IN2 以及 PGA 差分输入端 DI+/DI- 连接控制位
- | INX[1,0]=00 | INX[1,0]=01 | INX[1,0]=10 | INX[1,0]=11 |
|---|---|--|---|
|  |  |  |  |
- Bit 3~2 未定义，读为“0”
- Bit 1 **PGABPS**: PGA 旁路控制
0: 正常模式
1: 旁路 PGA 模式
- Bit 0 **PGAEN**: PGA 使能 / 除能控制
0: 除能
1: 使能
在使用 PGA 功能前，PGAEN 位必须使能。

● PGACS 寄存器

Bit	7	6	5	4	3	2	1	0
Name	CHSN3	CHSN2	CHSN1	CHSN0	CHSP3	CHSP2	CHSP1	CHSP0
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 7~4 **CHSN3~CHSN0**: 反相输入端 IN2 输入信号选择位
0000~0111: 保留
1000: DAC1O
1001: DAC2O
1010: DAC3O
1011: DAC0O
1100: AV_{DD}/6

1101: AV_{SS}
1110: AIN1
1111: 保留

Bit 3~0 **CHSP3~CHSP0**: 正相输入端 IN1 输入信号选择位
0000~0111: 保留
1000: OP1O
1001: OP2O
1010: OP3O
1011: AV_{SS}
1100: AV_{DD}/5
1101: AIN0
1110~1111: 保留

A/D 转换器数据寄存器 – ADRL, ADRM, ADRH

对于具有 24 位 Delta Sigma A/D 转换器的单片机，需要 3 个数据寄存器存放转换结果，一个高字节寄存器 ADRH、一个中字节寄存器 ADRM 和一个低字节寄存器 ADRL。在 A/D 转换完毕后，单片机可以直接读取这些寄存器以获得转换结果。D23~D0 是 A/D 转换数据结果位。

• ADRL 寄存器

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

“x”：未知

Bit 7~0 **D7~D0**: A/D 转换数据寄存器 bit 7 ~ bit 0

• ADRM 寄存器

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

“x”：未知

Bit 7~0 **D15~D8**: A/D 转换数据寄存器 bit 15 ~ bit 8

• ADRH 寄存器

Bit	7	6	5	4	3	2	1	0
Name	D23	D22	D21	D20	D19	D18	D17	D16
R/W	R	R	R	R	R	R	R	R
POR	x	x	x	x	x	x	x	x

“x”：未知

Bit 7~0 **D23~D16**: A/D 转换数据寄存器 bit 23 ~ bit 16

A/D 转换器控制寄存器 – ADCR0, ADCR1, ADCR2, ADCR3, ADCS

寄存器 ADCR0、ADCR1、ADCR2、ADCR3 和 ADCS 用来控制 A/D 转换器的功能和操作。这些 8 位的寄存器定义包括选择 A/D 转换过采样率，A/D 转换器的时钟源，并控制 A/D 转换器和调制器的开始等。

• ADCR0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ADRST	—	SINCEN	ADOR3	ADOR2	ADOR1	ADOR0	—
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	—
POR	0	—	0	0	0	0	0	—

- Bit 7 **ADRST:** A/D 转换器软件复位控制
 0: 除能
 1: 使能
 此位可用来复位 A/D 转换器内部数字 SINC 滤波器。此位为低时, A/D 转换正常工作, 若将此位设为高, 将复位内部数字 SINC 滤波器, 同时当前 A/D 转换数据失效。再清零此位, 将开始一次新的 A/D 转换。
- Bit 6 未定义, 读为 “0”
- Bit 5 **SINCEN:** SINC 功能控制
 0: 除能
 1: 使能
- Bit 4~1 **ADOR3~ADOR0:** A/D 转换器过采样率 (OSR) 选择
 0000: OSR=32768
 0001: OSR=16384
 0010: OSR=8192
 0011: OSR=4096
 0100: OSR=2048
 0101: OSR=1024
 0110: OSR=512
 0111: OSR=256
 1000: OSR=128
 其他: 保留
- Bit 0 未定义, 读为 “0”

• ADCR1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	FLMS2	FLMS1	FLMS0	—	—	ADCDL	EOC	—
R/W	R/W	R/W	R/W	—	—	R/W	R/W	—
POR	0	0	0	—	—	0	0	—

- Bit 7~5 **FLMS2~FLMS0:** A/D 转换器时钟 (f_{ADCK}) 分频比和除数因子 (N) 选择
 000: $f_{ADCK}=f_{MCLK}/30$, N=30
 010: $f_{ADCK}=f_{MCLK}/12$, N=12
 其他: 保留
- Bit 4~3 未定义, 读为 “0”
- Bit 2 **ADCDL:** A/D 转换器数据锁存功能控制位
 0: 除能数据锁存功能
 1: 使能数据锁存功能
 如果使能 A/D 转换数据锁存功能, 最新转换的数据将被锁存, 且不会更新后面的转换结果直到该功能被除能。虽然转换后的数据被锁存到数据寄存器, A/D 转换电路仍正常运行, 但并不产生中断, EOC 也不改变。建议在读取 ADRL、ADRM 和 ADRH 寄存器中的转换数据之前先将该位置高。读取后该位会被清零以除能 A/D 转换器数据锁存功能, 以便下一笔转换结果的存储。这样可以防止在 A/D 转换器转换过程中得到不需要的数据。
- Bit 1 **EOC:** A/D 转换结束标志位
 0: A/D 转换中
 1: A/D 转换结束
 此位必须通过软件清零。
- Bit 0 未定义, 读为 “0”

• **ADCR2 寄存器**

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

Bit 7~6 **D7~D6:** 保留, 不可变

Bit 5 未定义, 读为 “0”

Bit 4 **MODEN:** 调制器控制
0: 除能
1: 使能

Bit 3~0 **D3~D0:** 保留位, 不可变

• **ADCR3 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MODRST	D6	D5	D4	—	—	—	D0
R/W	R/W	R/W	R/W	R/W	—	—	—	R/W
POR	0	1	1	1	—	—	—	1

Bit 7 **MODRST:** 调制器复位控制
0: 正常
1: 复位

Bit 6~4 **D6~D4:** 保留位, 不可变

Bit 3~2 未定义, 读为 “0”

Bit 1 **D0:** 保留位, 不可变

• **ADCS 寄存器**

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

Bit 7~5 未定义, 读为 “0”

Bit 4~0 **ADCK4~ADCK0:** A/D 转换时钟源 (f_{MCLK}) 设置
00000~11110: $f_{MCLK} = f_{SYS}/2/(ADCK[4:0]+1)$
11111: $f_{MCLK} = f_{SYS}$

SINC 滤波器寄存器 – SINC3

有一个与 SINC 滤波器控制相关的寄存器, SINC3。该寄存器用于输出数字滤波器选择和斩波器功能控制。

• **SINC3 寄存器**

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

Bit 7~2 **D7~D2:** 保留位, 不可变

- Bit 1 **R_FILSEL:** 输出数字滤波器选择位
0: SINC2 滤波器 (R_CKCHOP 为逻辑低), 数据延迟 = 3 个输出数据时钟
1: SINC3 滤波器 (R_CKCHOP 位逻辑高), 数据延迟 = 4 个输出数据时钟
- Bit 0 **R_CKCHOP:** 系统斩波器功能选择位
0: 使能 (R_FILSEL 为逻辑低)
1: 除能 (R_FILSEL 为逻辑高)

A/D 转换器数据传输率的定义

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

SINC2 的数据传输率

$$\begin{aligned} &= f_{ADCK} / (2 \times OSR) \\ &= (f_{MCLK} / N) / (2 \times OSR) \\ &= f_{MCLK} / (N \times 2 \times OSR) \end{aligned}$$

SINC3 的数据传输率

$$\begin{aligned} &= f_{ADCK} / OSR \\ &= (f_{MCLK} / N) / OSR \\ &= f_{MCLK} / (N \times OSR) \end{aligned}$$

f_{ADCK} : A/D 转换器时钟频率, 来自 f_{MCLK}/N ;

f_{MCLK} : A/D 转换器时钟源, 来自 f_{SYS} 或 $f_{SYS}/2/(ADCK[4:0]+1)$, 通过 ADCK[4:0] 位来选择个设置;

N: 除数因子, 可为 12 或 30, 通过 FLMS[2:0] 为来选择;

OSR: 过采样率, 通过 ADOR[3:0] 位来选择。

例如, 若需要一个 8Hz 的数据传输率, 可以选择 A/D 时钟源 f_{MCLK} 为 4MHz, 然后设置 FLMS[2:0]=000B, 即获得 A/D 转换时钟为 A/D 时钟源的 30 分频。

对于 SINC2 滤波器, 设置 ADOR[3:0]=0010B, 选择过采样率为 8192。因此, 可以得到一个数据传输率 = $4\text{MHz}/(30 \times 2 \times 8192) = 8\text{Hz}$ 。

对于 SINC3 滤波器, 设置 ADOR[3:0]=0001B, 选择过采样率为 16384。因此, 可以得到一个数据传输率 = $4\text{MHz}/(30 \times 16384) = 8\text{Hz}$ 。

A/D 转换器操作

要打开 A/D 转换器, 首先应除能 A/D 转换器的暂停模式, 以确保 A/D 转换器可以通电。ADCR0 寄存器中的 ADRST 位, 用于上电后打开和复位 A/D 转换器。当单片机设定此位从逻辑低到逻辑高, 然后再到逻辑低, 一个模数转换后的数据就会开始在 SINC 滤波器中进行转换。设置完成后, A/D 转换器可以开始工作。此位用于控制内部 A/D 转换器的开启动作。

ADCR1 寄存器中的 EOC 位用于表明模数转换过程的完成。在转换周期结束后, EOC 位会被单片机自动地置为“1”。此外, 也会置位中断控制寄存器内相应的 A/D 中断请求标志位, 如果中断使能, 就会产生对应的内部中断信号。A/D 内部中断信号将引导程序到相应的 A/D 内部中断入口。如果 A/D 内部中断被禁止, 可以让单片机轮询 ADCR1 寄存器中的 EOC 位, 检查此位是否被置高, 以作为另一种侦测 A/D 转换周期结束的方法。A/D 转换数据将不断更新, 如果 A/D 转换数据锁存功能使能, 最新的转换数据会被锁存, 这样后面再转换的数据不会被保存, 直到该功能被关闭。

A/D 转换器的时钟源通常固定在 4MHz, 来自系统时钟 f_{SYS} 或其分频, 分频系数由 ADCS 寄存器中的 ADCK4~ADCK0 位决定, 以获得固定 4MHz 的 ADC 时钟源。

A/D 转换器差分参考电压可以来自内部参考源。

A/D 转换步骤

下面概述实现 A/D 转换过程的各个步骤。

- 步骤 1
使能 LDO，以提供电源给 PGA 和 A/D 转换器。
- 步骤 2
通过 PGAC0 寄存器，选择 PGA、A/D 转换器增益和参考电压。
- 步骤 3
通过 PGAC1 寄存器，选择 PGA 使能 / 除能控制、输入引脚连接、输入偏压和旁路控制。
- 步骤 4
通过 ADCS 寄存器中的 ADCK4~ADCK0 位，选择所需的 A/D 转换时钟源。
- 步骤 5
通过 ADCR0 寄存器中的 ADOR3~ADOR0 位以及 ADCR1 寄存器中的 FLMS2~FLMS0 位，选择输出数据传输率。
- 步骤 6
通过 PGACS 寄存器中的 CHSP3~CHSP0 和 CHSN3~CHSN0 位，选择连接至内部 PGA 的通道。
- 步骤 7
通过置高 ADCR0 寄存器中的 ADRST 位来复位 A/D 转换器，清零该位来释放复位状态。
- 步骤 8
如果要使用 A/D 转换器中断，则相关的中断控制寄存器需要正确地设置，以确保 A/D 之后中断功能时激活的。总中断控制位 EMI 需要置位为“1”，以及 A/D 转换器中断位 ADE 也需要置位为“1”。
- 步骤 9
可以轮询 ADCR1 寄存器中的 EOC 位，检查模数转换过程是否完成。当此位成为逻辑高时，表示转换过程已经完成。转换完成后，可读取 A/D 数据寄存器 ADRL、ADRM 和 ADRH 获得转换后的值。另一种方法是，若 A/D 转换器中断和总中断使能且中断未满，则程序等待 A/D 转换器中断发生。

注：若使用轮询 ADCR1 寄存器中 EOC 位的状态的方法来检查转换过程是否结束时，则中断使能的步骤可以省略。

编程注意事项

在编程时，如果 A/D 转换器未定义，关闭 A/D 内部电路以减少电源功耗。此时，不考虑输入脚的模拟电压，内部 A/D 转换器电路不产生功耗

A/D 转换器传输功能

该单片机含有一组 24 位 Delta Sigma A/D 转换器，它的转换范围为 -8388608~8388607。转换后的数据以二进制补码的形式表示，最高为是转换数据的符号位。由于模拟输入最大值等于差分参考输入电压值， ΔVR_{\pm} ，由 PGAC0 寄存器中的 VREFS[1:0] 位选择。因此每一位可表示 $\Delta VR_{\pm}/8388608$ 的模拟输入值。

$$1 \text{ LSB} = \Delta VR_{\pm} / 8388608$$

通过下面的等式可计算 A/D 转换器输入电压值：

$$\Delta SI_I = PGAGN \times ADGN \times \Delta DI \pm$$

$$A/D \text{ 转换数据} = (\Delta SI_I / \Delta VR \pm) \times K$$

其中， $K=2^{23}$

注：1. PGAGN 和 ADGN 的值分别由 PGS[2:0] 和 AGS 控制位决定。

2. ΔSI_I ：经过放大和偏置校准后的差分输入信号

3. PGAGN：PGA 增益

4. ADGN：A/D 转换器增益

5. $\Delta DI \pm$ ：差分输入信号，来自外部通道或内部信号

6. $\Delta VR \pm$ ：差分参考电压

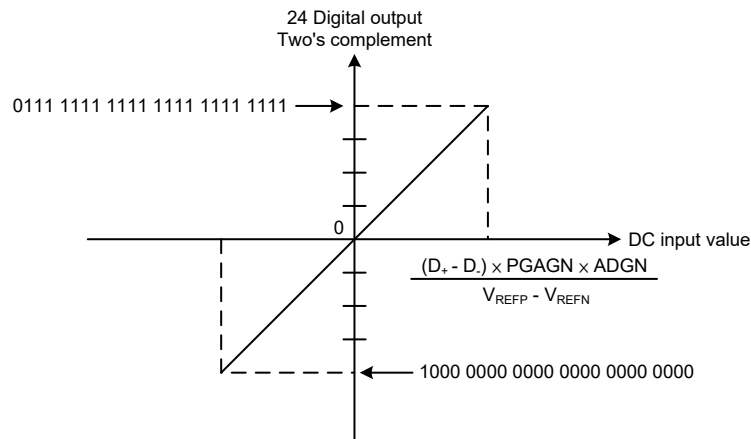
由于 Delta Sigma A/D 转换器的数字系统设计，其转换的最大值为 8388607，最小值为 -8388608，因此有一个中间值 0。A/D 转换数据公式说明了转换值的变化范围。

A/D 转换数据 (二进制补码，十六进制值)	十进制值
0x7FFFFFFF	8388607
0x800000	-8388608

A/D 转换数据范围

说明的 A/D 转换数据表说明了 A/D 转换值的范围。

下图显示直流输入电压值和 A/D 转换数据 (以二进制补码形式表示) 之间的关系。



A/D 转换数据

A/D 转换数据与输入电压和 PGA 的设置有关。A/D 转换输出数据以二进制补码的形式表示，代码的长度为 24 位，最高位为符号位。最高位“0”表示输出为正数，最高位“1”表示输出为负数。最大值是 8388607，最小值是 -8388608。如果输入信号大于最大值，转换后的数据最大不超过 8388607；如果输入信号小于最小值，转换后的数据最小不低于 -8388608。

A/D 转换数据转为电压值

设计者可以通过下面的公式来由转换之后的数据推导电压值。

如果 $MSB = 0$ (转换数据为正数)

输入电压 = (转换数据 \times LSB) / (PGAGN \times ADGN)

如果 $MSB = 1$ (转换数据为负数)

输入电压 = (转换数据的补码 \times LSB) / (PGAGN \times ADGN)

注：补码 = 反码 + 1

串行接口模块 – SIM

此单片机内建一个串行接口模块，包括两种易与外部设备通信的串行接口：四线 SPI 或两线 I²C 接口。这两种接口具有相当简单的通信协议，单片机可以通过这些接口与外部基于 SPI 或 I²C 的传感器、闪存等硬件设备通信。因为 SIM 接口引脚是与其它 I/O 引脚共用，因此在使用 SIM 功能前，要先通过相应的引脚共用功能选择寄存器选定 SIM 引脚功能。因为 SPI 和 I²C 这两种接口共用引脚和寄存器，所以要先通过 SIMC0 寄存器中的 SIM2~SIM0 位选择哪一种通信接口。若 SIM 功能使能且引脚用作 SIM 输入脚，可通过对应上拉电阻控制寄存器选择此脚的上拉电阻。

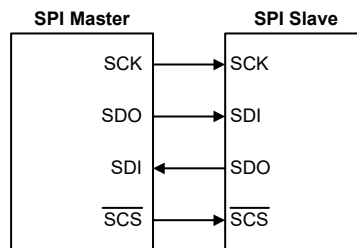
SPI 接口

SPI 接口常用于与外部设备如传感器、Flash 存储器或 EEPROM 存储器等通信。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 I²C 的功能脚共用。通过设定相关引脚共用选择位以及 SIMC0 和 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。

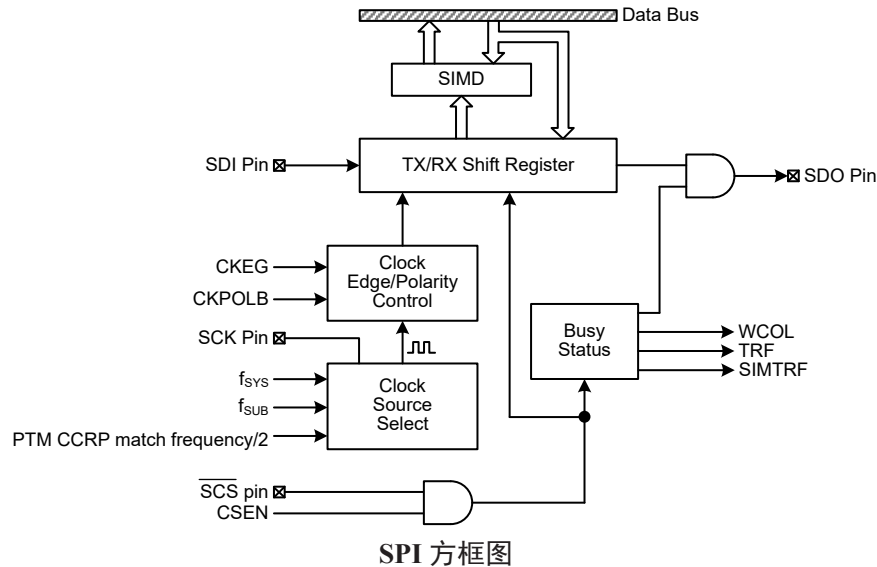


SPI 主 / 从机连接方式

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。SIMC1 寄存器仅用于 I²C 接口。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	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**: SIM 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。应注意的是 SIMC2 与 I²C 接口功能中的寄存器 SIMA 是同一个寄存器。SPI 功能不会用到寄存器 SIMC1，SIMC1 寄存器仅在工作于 I²C 接口时才被使用。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	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7~5 **SIM2~SIM0**: SIM 工作模式控制位

- 000: SPI 主机模式; SPI 时钟为 $f_{\text{SYS}}/4$
- 001: SPI 主机模式; SPI 时钟为 $f_{\text{SYS}}/16$
- 010: SPI 主机模式; SPI 时钟为 $f_{\text{SYS}}/64$
- 011: SPI 主机模式; SPI 时钟为 f_{SUB}
- 100: SPI 主机模式; SPI 时钟为 PTM CCRP 匹配频率 / 2
- 101: SPI 从机模式
- 110: I²C 从机模式
- 111: 未使用模式

这几位用于设置 SIM 功能的工作模式，除了选择 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 PTM。若选择的是作为 SPI 模式，则其时钟源从外部主机而得。

Bit 4 未定义，读为 “0”

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C 去抖时间选择位

这些位只有在 SIM 设置成 I²C 接口模式时才有效。请参考 I²C 寄存器部分。

Bit 1 **SIMEN**: SIM 控制位

- 0: 除能
- 1: 使能

此位为 SIM 接口的开 / 关控制位。此位为 “0” 时，SIM 接口除能，SDI、SDO、SCK 和 SCS 或 SDA 和 SCL 脚将失去 SPI 或 I²C 功能，SIM 工作电流减小到最小值。此位为 “1” 时，SIM 接口使能。若 SIM 经由 SIM2~SIM0 位设置为工作在 SPI 接口，当 SIMEN 位由低到高转变时，SPI 控制寄存器中的设置不会发生变化，其首先应在应用程序中初始化。若 SIM 经由 SIM2~SIM0 位设置为工作在 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**: SIM SPI 未完成标志位

- 0: 未发生
- 1: 发生

此位仅当 SIM 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

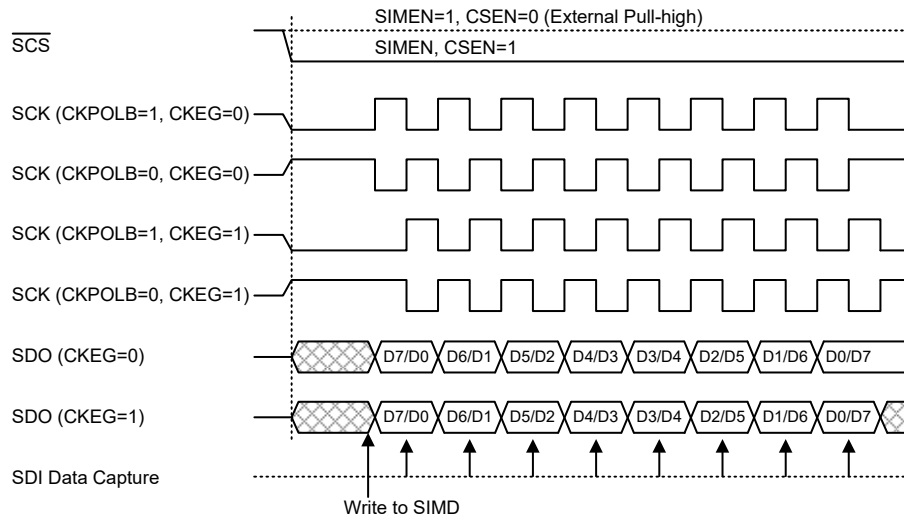
Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- 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 SCK 有效时钟边沿类型位
CKPOLB=0
0: SCK 为高电平且在 SCK 上升沿抓取数据
1: SCK 为高电平且在 SCK 下降沿抓取数据
CKPOLB=1
0: SCK 为低电平且在 SCK 下降沿抓取数据
1: SCK 为低电平且在 SCK 上升沿抓取数据
CKEG 和 CKPOLB 位用于设置 SPI 总线上时钟信号输入和输出方式。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 位决定时钟线的基本状态, 若时钟无效且此位为高, 则 SCK 为低电平, 若时钟无效且此位为低, 则 SCK 为高电平。CKEG 位决定有效时钟边沿类型, 取决于 CKPOLB 的状态。
- Bit 3 MLS:** SPI 数据移位命令位
0: LSB 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。

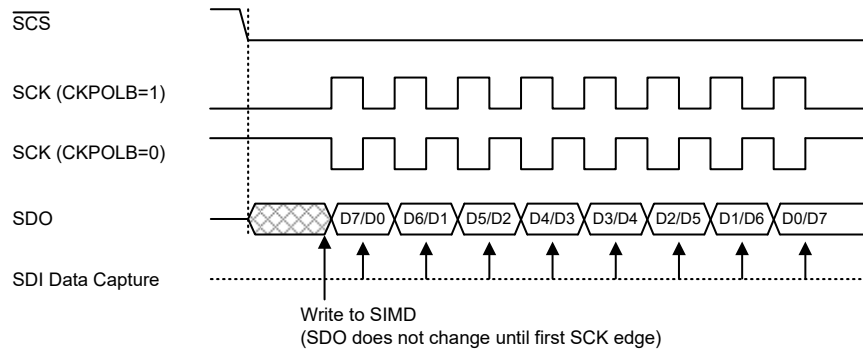
SPI 通信

将 SIMEN 设置为高，使能 SPI 功能之后，单片机处于主机模式，当数据写入到寄存器 SIMD 的同时传输 / 接收开始进行。数据传输完成时，TRF 位将自动被置位但清除只能通过应用程序完成。单片机处于从机模式时，收到主机发来的信号之后，会传输 SIMD 中的数据，而且在 SDI 引脚上的数据也会被移位到 SIMD 寄存器中。主机应在输出时钟信号之前先输出一个 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 CKPOLB 和 CKEG 位各种设置情况下从机数据与 SCK 信号的关系。

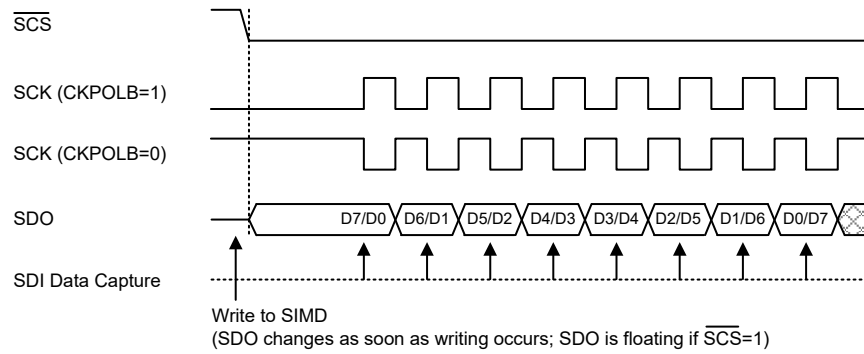
即使在单片机处于空闲模式时，若 SPI 接口使用的时钟源仍开启，SPI 功能仍将继续执行。



SPI 主机模式时序

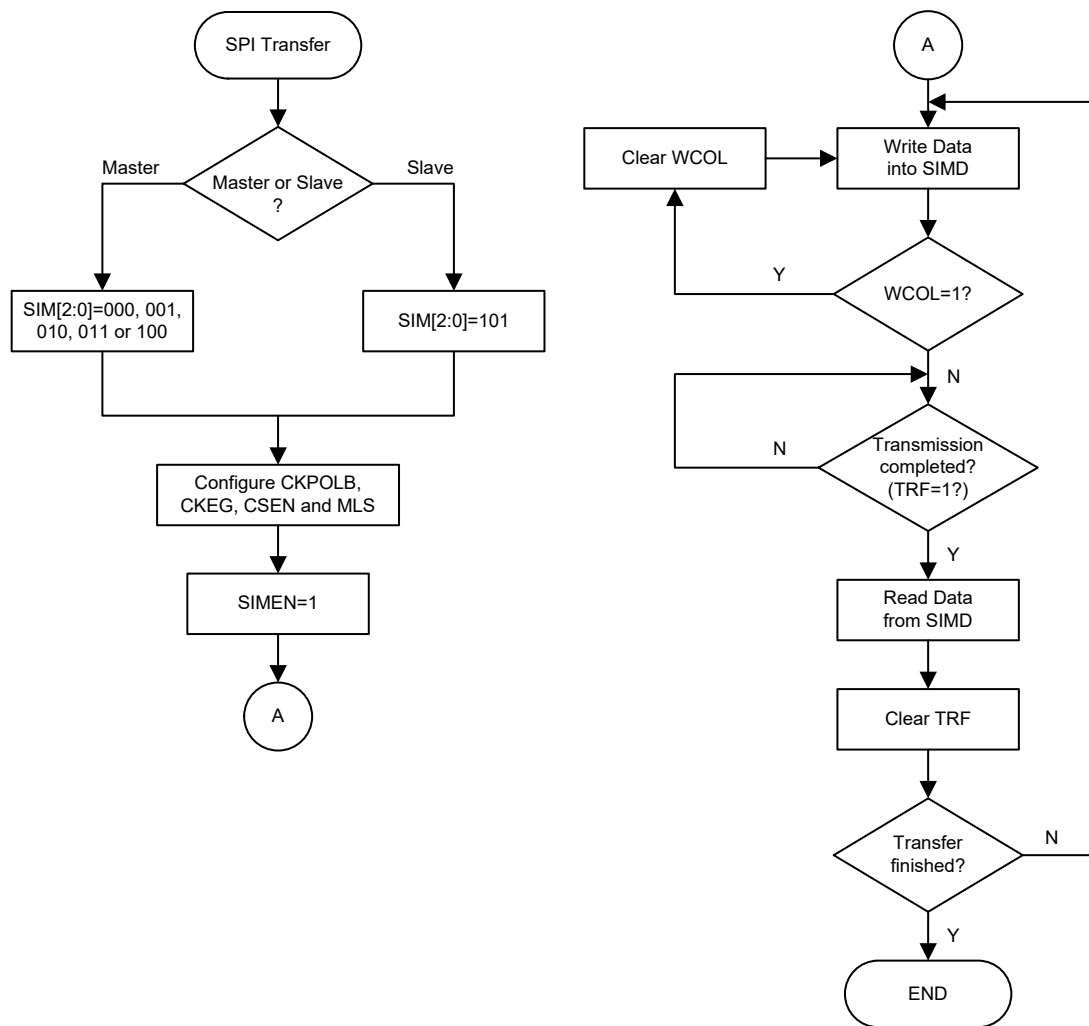


SPI 从机模式时序 – CKEG=0



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 $\overline{\text{CS}}=1$ 、 $\overline{\text{SCS}}=0$ 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、 $\overline{\text{SCS}}$ 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， $\overline{\text{SCS}}$ 信号线有效将使能 SPI 接口。设置此位为低，SPI 接口将除能， $\overline{\text{SCS}}$ 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， $\overline{\text{SCS}}$ 、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 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 SDO 信号线将数据输出。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 SIM 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

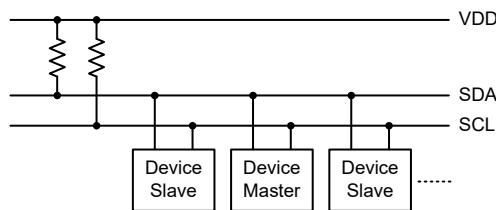
- 步骤 1
设置 SIMC0 控制寄存器中的 SIM2~SIM0 位，选择 SPI 从机模式。
- 步骤 2
设置 CSEN 和 MLS 位，选择高位或低位数据优先传送，这必须与主机设备一致。
- 步骤 3
设置 SIMC0 控制寄存器中的 SIMEN 位，使能 SPI 接口功能。
- 步骤 4
对于写操作：写数据到 SIMD 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。等待主机时钟 SCK 信号和 SCS 信号。跳至步骤 5。
对于读操作：从 SDI 信号线移入的数据将被存储在 TXRX 缓存器中，直到所有数据接收完毕，此时数据全部锁存至 SIMD 寄存器。
- 步骤 5
检测 WCOL 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 SIM 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。

I²C 接口

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

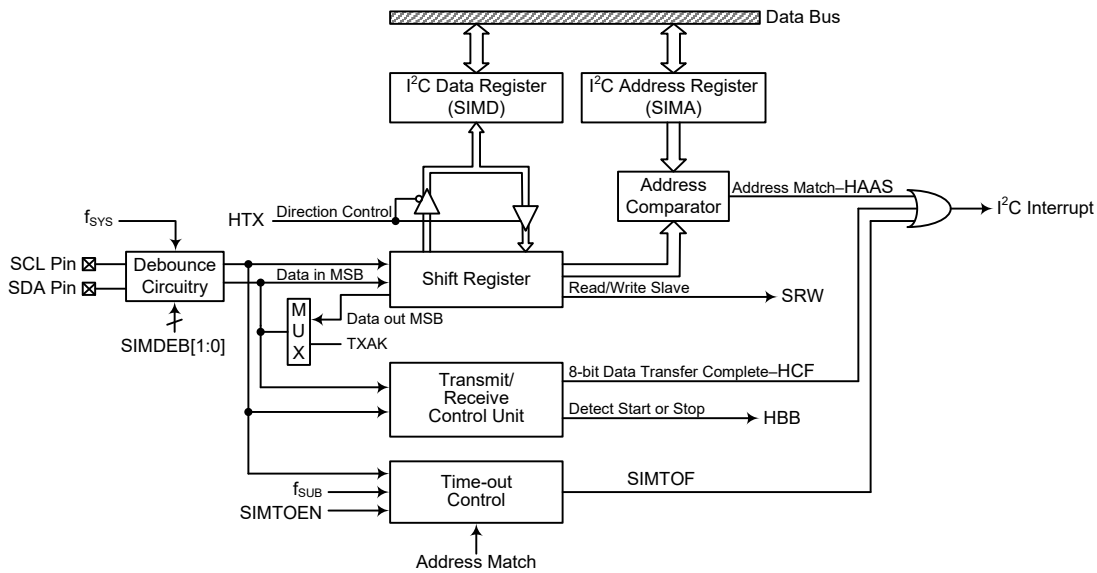


I²C 主 / 从总线连接图

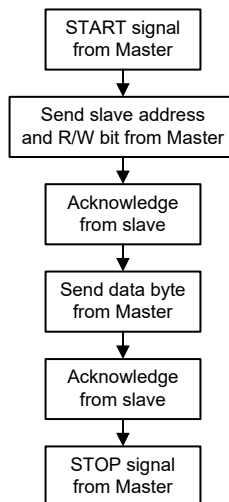
I²C 接口操作

I²C 串行接口是一个双线的接口，有一条串行数据线 SDA 和一条串行时钟线 SCL。由于可能有多个设备在同一条总线上相互连接，所以这些设备的输出都是开漏型输出。因此应在这些输出上都加上拉电阻。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。

I ² C 去抖时间选择	I ² C 标准模式 (100kHz)	I ² C 快速模式 (400kHz)
无去抖时间	$f_{\text{SYS}} > 2\text{MHz}$	$f_{\text{SYS}} > 4\text{MHz}$
2 个系统时钟去抖时间	$f_{\text{SYS}} > 4\text{MHz}$	$f_{\text{SYS}} > 8\text{MHz}$
4 个系统时钟去抖时间	$f_{\text{SYS}} > 4\text{MHz}$	$f_{\text{SYS}} > 8\text{MHz}$

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。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	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 寄存器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**: SIM 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。应注意的是寄存器 SIMA 和 SPI 接口使用的寄存器 SIMC2 共用同一个寄存器地址。

● 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或写

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	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7~5 **SIM2~SIM0**: SIM 工作模式控制位
 000: SPI 主机模式; SPI 时钟为 $f_{\text{SYS}}/4$
 001: SPI 主机模式; SPI 时钟为 $f_{\text{SYS}}/16$
 010: SPI 主机模式; SPI 时钟为 $f_{\text{SYS}}/64$
 011: SPI 主机模式; SPI 时钟为 f_{SUB}
 100: SPI 主机模式; SPI 时钟为 PTM CCRP 匹配频率 / 2
 101: SPI 从机模式
 110: I²C 从机模式
 111: 未使用模式

这几位用于设置 SIM 功能的工作模式, 用于选择 SPI 的主从模式和 SPI 的主机时钟频率及 I²C 或 SPI 功能。SPI 时钟源可来自于系统时钟和 f_{SUB} 也可以选择来自 PTM。若选择的是作为 SPI 从机, 则其时钟源从外部主机而得。

Bit 4 未定义, 读为 “0”

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C 去抖时间选择位

00: 无去抖时间
 01: 2 个系统时钟去抖时间
 1x: 4 个系统时钟去抖时间

当设置 SIM2~SIM0 位为 “110” 将 SIM 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**: SIM 控制位

0: 除能
 1: 使能

此位为 SIM 接口的开 / 关控制位。此位为 “0” 时, SIM 接口除能, SDI、SDO、SCK 和 SCS 或 SDA 和 SCL 脚将失去 SPI 或 I²C 功能, SIM 工作电流减小到最小值。此位为 “1” 时, SIM 接口使能。若 SIM 经由 SIM2~SIM0 位设置为工作在 SPI 接口, 当 SIMEN 位由低到高转变时, SPI 控制寄存器中的设置不会发生变化, 其首先应在应用程序中初始化。若 SIM 经由 SIM2~SIM0 位设置为工作在 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**: SIM SPI 未完成标志位
此位仅当 SIM 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 总线数据传输完成标志位
0: 数据正在被传输
1: 8 位数据传输完成
HCF 位是数据传输完成标志位。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。以下是两个字节 I²C 数据传输的流程的例子。第一步，I²C 从机接收来自主机的起始信号，HCF 位自动清零。第二步，I²C 从机完成接收第一个字节数据，HCF 位自动置高。第三步，用户通过应用程序从 SIMD 寄存器中读取第一个字节数据，HCF 位自动清零。第四步，I²C 从机完成第二个字节数据的接收，HCF 位自动置高，以此类推。最后 I²C 从机接收到来自主机的停止信号，HCF 位自动置高。

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
TXAK 位是发送应答标志位。从机接收到 8 位数据后，将在第九个时钟发送应答标志到总线。如果从机想要接收更多数据，必须始终将此位设置为“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，此时从机处于传输模式。当 SRW 位为“0”时，主机往总线上写数据，从机处于接收模式以读取数据。

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

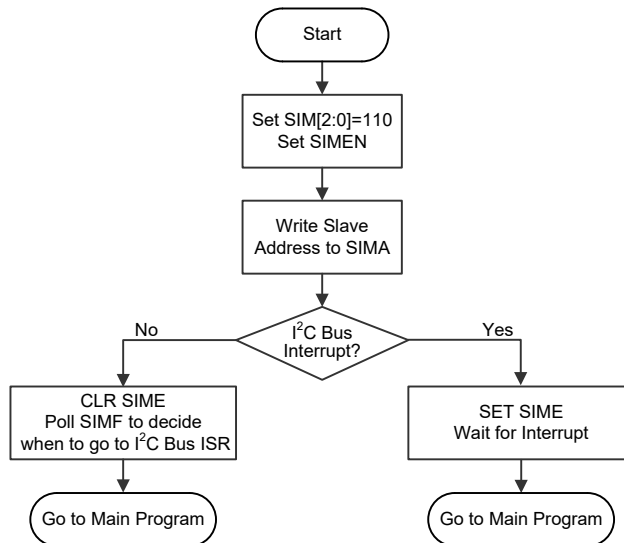
- 0: 从机接收到应答标志
- 1: 从机没有接收到应答标志

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收到一个应答信号。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 SIM 中断。进入中断服务程序后，系统要检测 HAAS 位和 SIMTOF 位，以判断 I²C 总线中断是来自从机地址匹配，还是来自 8 位数据传递完毕，或是来自 I²C 超时。在数据传递中，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 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 SIME 位，以使能 SIM 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。

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 SIM 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。

SIM 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 SIM 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。

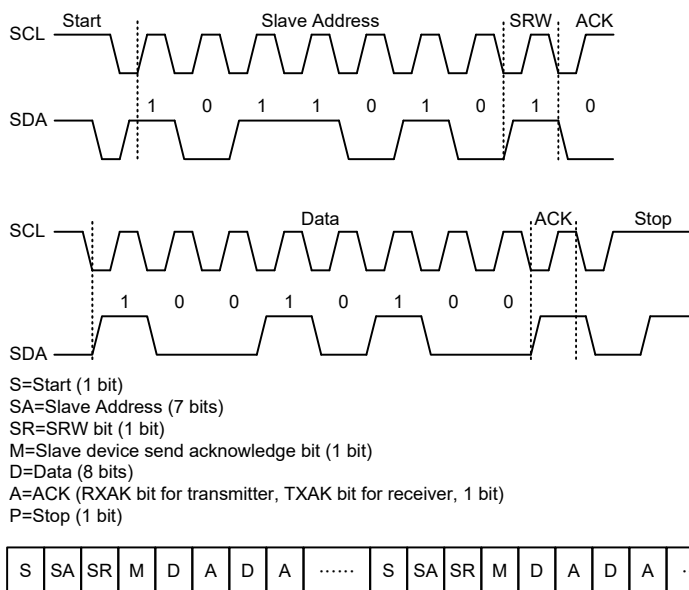
I²C 总线从机地址应答信号

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

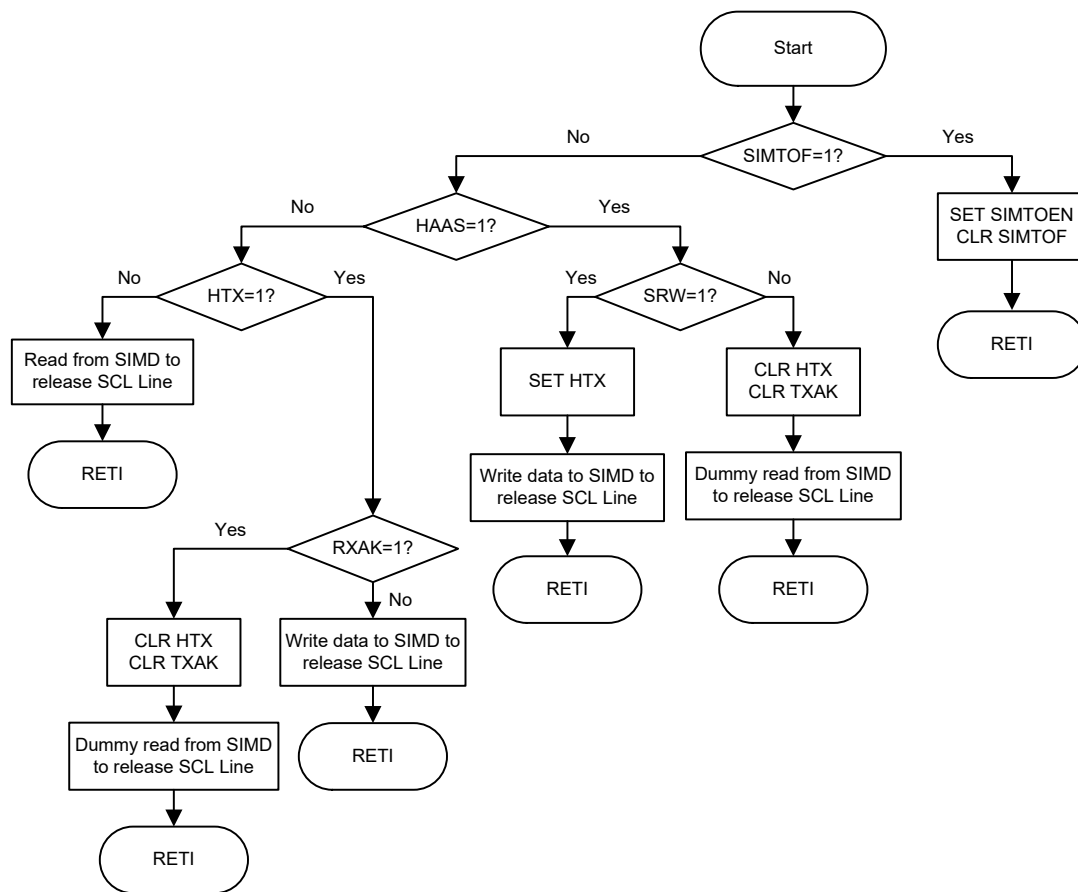
在从机确认接收到从地址后，会进行 8 位宽度的数据传输。这个数据传输顺序是的高位在前，低位在后。接收方在接收到 8 位数据后必须发出一个应答信号 (“0”) 以继续接收下一个数据。如果从机发送方没接收到来自主机接收方的应答信号，发送方将释放 SDA 线，此时主机方可发出 STOP 信号以释放 I²C 总线。所传送的数据存储在 SIMD 寄存器中。如果设置成发送方，从机必须先将欲传输的数据写到 SIMD 寄存器中；如果设置成接收方，从机必须从 SIMD 寄存器读取数据。

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。



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。

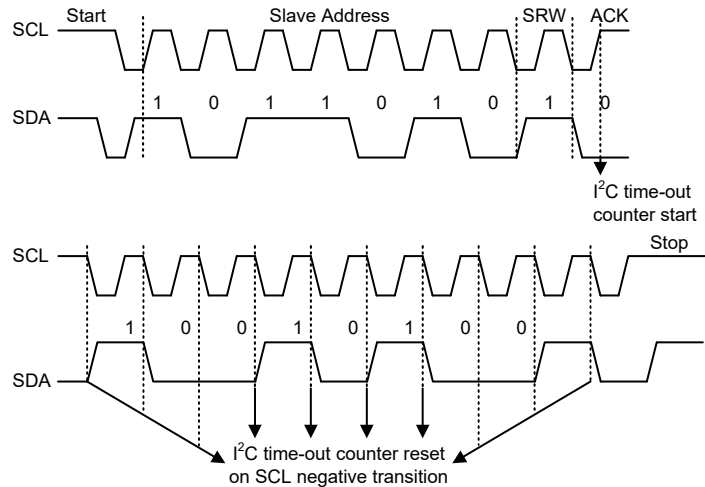
I²C 通信时序图



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 SIM 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	I ² C 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 SIMTOS5~SIMTOS0 位进行选择。超时周期可通过公式计算： $((1\sim64)\times(32/f_{\text{SUB}}))$ 。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**: I²C 超时控制位

0: 除能
1: 使能

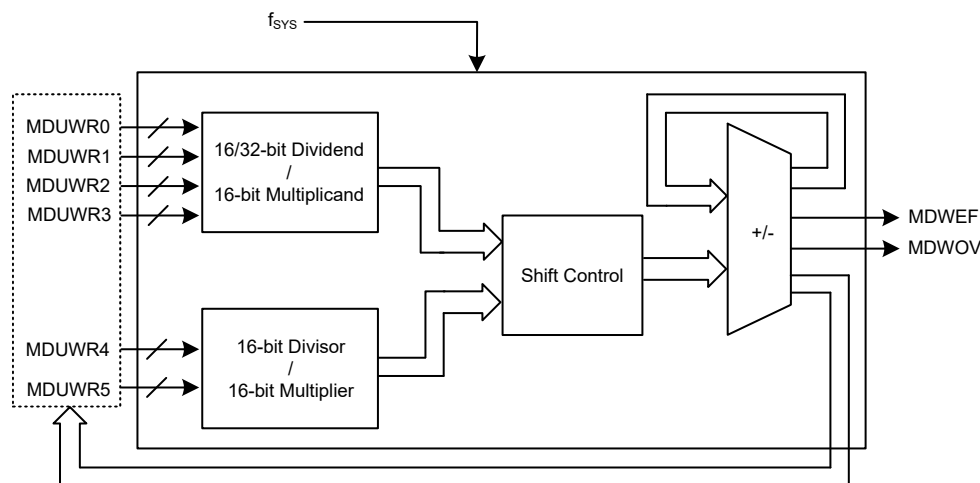
Bit 6 **SIMTOF**: I²C 超时标志位

0: 超时未发生
1: 超时发生

Bit 5~0 **SIMTOS5~SIMTOS0**: 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: $(SIMTOS[5:0]+1) \times (32/f_{SUB})$ 。

16 位乘除法单元 – MDU

该单片机内置了 16 位乘除法单元，即 MDU，是 16 位无符号乘法与 32 位 /16 位无符号除法器。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

● 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。无论是被除数、被乘数、除数或乘数的低字节数据, 必须在其高字节数据之前先写入对应的 MDU 数据寄存器中。所有的 MDU 操作都在已按照正确顺序写入 MDUWRn 寄存器且 MDUWR5 寄存器被写入后开始执行。不一定要连续往 MDU 数据寄存器中写数据, 但必须要按照正确的顺序写入。因此, 只要不影响写操作, 在正确的写入顺序中间允许插入非写入 MDUWRn 的指令或中断。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

写顺序确定后, 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，且需按照特定的顺序读取。不一定要连续从 MDU 数据寄存器中读取结果，但必须按照正确的顺序读取。因此，只要不影响读操作，在正确的读取顺序中间允许插入非读取 MDUWRn 的指令或中断。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 MDUWR4 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

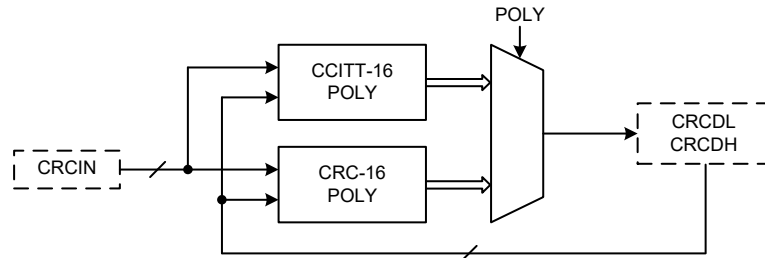
MDU 读 / 写顺序以及计算时间的注意事项总结如下表所示。注意，在 MDU 操作未完成之前，单片机不可进入空闲或休眠模式，否则 MDU 操作失败。

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 读取商数字节 0 从 MDUWR1 读取商数字节 1 从 MDUWR2 读取商数字节 2 从 MDUWR3 读取商数字节 3 从 MDUWR4 读取余数字节 0 从 MDUWR5 读取余数字节 1	从 MDUWR0 读取商数字节 0 从 MDUWR1 读取商数字节 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
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
CRCCR	—	—	—	—	—	—	—	POLY

CRC 寄存器列表

• 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

● **CRC DH 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

● **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 生成多项式选择

0: CRC-CCITT: $X^{16}+X^{12}+X^5+1$

1: CRC-16: $X^{16}+X^{15}+X^2+1$

CRC 操作

CRC 发生器提供了基于 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 CRC DH 和 CRC DL 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 CRC IN 寄存器进行写操作，都会将存储在 CRC DH 和 CRC DL 寄存器对中的前个 CRC 值和新的输入数据结合起来。CRC 单元计算 CRC 数据寄存器值是按字节进行的。CRC 校验和的计算需要一个 MCU 指令周期。

CRC 计算步骤:

1. 清除校验和寄存器对 CRC DH 和 CRC DL。
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 校验和连续列于下表。

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：

- 清除校验和寄存器对 CRCDH 和 CRCDL。
- 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。
- 执行表格读取指令，读取程序存储器数据值。
- 将表格数据低字节写入 CRCIN 寄存器，并结合当前 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。
- 将表格数据高字节写入 CRCIN 寄存器，并结合当前 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。
- 重复步骤 3 到步骤 5 以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。

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~INT3 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、时基、EEPROM、SIM、比较器和 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专用数据存储器中的一系列寄存器控制的。寄存器总的分为三类。第一类是 INTC0~INTC3 寄存器，用于设置基本的中断；第二类是 MFI0~MFI1 寄存器，用于设置多功能中断；最后一种有 INTEG 寄存器，用于设置外部中断边沿触发类型。

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

功能	使能位	请求标志	注释
总中断	EMI	—	—
INTn 引脚	INTnE	INTnF	n=0~3
A/D 转换器	ADE	ADF	—
多功能中断	MFnE	MFnF	n=0 或 1
时基	TBnE	TBnF	n=0 或 1
SIM	SIME	SIMF	—
比较器	CMPE	CMPF	—
EEPROM	DEE	DEF	—
CTM	CTMnPE	CTMnPF	n=0 或 1
	CTMnAE	CTMnAF	
PTM	PTMPE	PTMPF	—
	PTMAE	PTMAF	—

中断寄存器位命名模式

寄存器名称	位							
	7	6	5	4	3	2	1	0
INTEG	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
INTC1	INT2F	CMPF	MF1F	MF0F	INT2E	CMPE	MF1E	MF0E
INTC2	—	TB1F	TB0F	INT3F	—	TB1E	TB0E	INT3E
INTC3	—	—	—	SIMF	—	—	—	SIME
MFI0	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
MFI1	—	DEF	PTMAF	PTMPF	—	DEE	PTMAE	PTMPE

中断寄存器列表

• INTEG 寄存器

Bit	7	6	5	4	3	2	1	0
Name	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **INT3S1~INT3S0:** INT3 脚中断边沿控制位
 00: 除能
 01: 上升沿
 10: 下降沿
 11: 双沿

Bit 5~4 **INT2S1~INT2S0:** INT2 脚中断边沿控制位
 00: 除能
 01: 上升沿
 10: 下降沿
 11: 双沿

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

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

• INTC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 未定义，读为“0”

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

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

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

Bit 3 **ADE:** A/D 转换器中断控制位
 0: 除能
 1: 使能

Bit 2 **INT1E:** INT1 中断控制位
 0: 除能
 1: 使能

Bit 1 **INT0E:** INT0 中断控制位
 0: 除能
 1: 使能

Bit 0 **EMI**: 总中断控制位
0: 除能
1: 使能

● **INTC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	INT2F	CMPF	MF1F	MF0F	INT2E	CMPE	MF1E	MF0E
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 **INT2F**: INT2 中断请求标志位
0: 无请求
1: 中断请求

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

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

Bit 4 **MF0F**: 多功能中断 0 请求标志位
0: 无请求
1: 中断请求

Bit 3 **INT2E**: INT2 中断控制位
0: 除能
1: 使能

Bit 2 **CMPE**: 比较器中断控制位
0: 除能
1: 使能

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

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

● **INTC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	TB1F	TB0F	INT3F	—	TB1E	TB0E	INT3E
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 **TB1F**: 时基 1 中断请求标志位
0: 无请求
1: 中断请求

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

Bit 4 **INT3F**: INT3 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 **TB1E**: 时基 1 中断控制位
0: 除能
1: 使能
- Bit 1 **TB0E**: 时基 0 中断控制位
0: 除能
1: 使能
- Bit 0 **INT3E**: INT3 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	—	—	—	SIMF	—	—	—	SIME
R/W	—	—	—	R/W	—	—	—	R/W
POR	—	—	—	0	—	—	—	0

- Bit 7~5 未定义，读为“0”
- Bit 4 **SIMF**: SIM 中断请求标志位
0: 无请求
- Bit 3~1 未定义，读为“0”
- Bit 0 **SIME**: SIM 中断控制位
0: 除能
1: 使能

● **MF10 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	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 **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 6 **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 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 **CTM1AE**: CTM1 比较器 A 匹配中断控制位
0: 除能
1: 使能
- Bit 2 **CTM1PE**: CTM1 比较器 P 匹配中断控制位
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

● **MF11 寄存器**

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

- Bit 7 未定义，读为“0”
- Bit 6 **DEF**: 数据 EEPROM 中断请求标志位
0: 无请求
1: 中断请求
- Bit 5 **PTMAF**: PTM 比较器 A 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **PTMPF**: PTM 比较器 P 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 3 未定义，读为“0”
- Bit 2 **DEE**: 数据 EEPROM 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

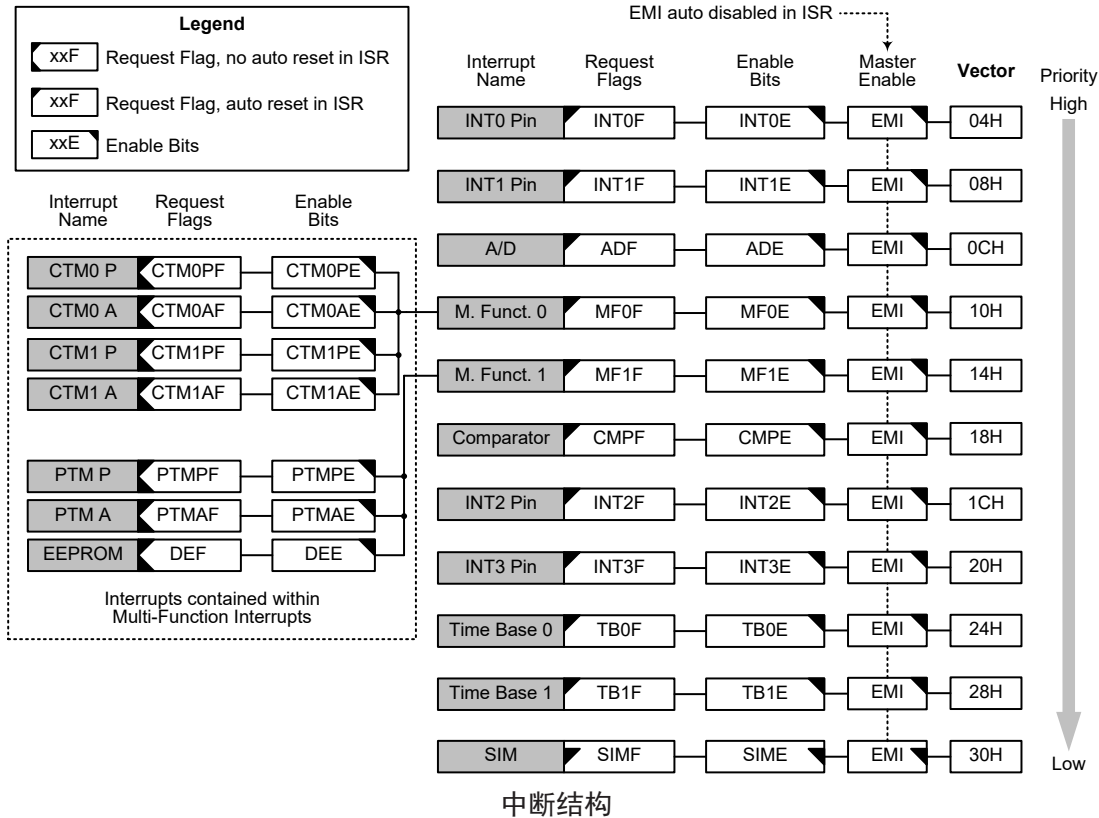
中断操作

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

当中断发生时，下条指令的地址将被压入堆栈。相应的中断向量地址加载至 PC 中。系统将从此向量取下条指令。中断向量处通常为“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。所有被置起的中断请求标志都可把单片机从休眠或空闲模式中唤醒，若要防止唤醒动作发生，在单片机进入休眠或空闲模式前应将相应的标志置起。



外部中断

通过 INT0~INT3 引脚上的信号变化可控制外部中断。当触发沿选择位设置好触发类型，INT0~INT3 引脚的状态发生变化，外部中断请求标志 INT0F~INT3F 被置位时外部中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和相应中断使能位 INT0E~INT3E 需先被置位。此外，必须使用 INTEG 寄存器使能外部中断功能并选择触发沿类型。外部中断引脚和普通 I/O 口共用，如果相应寄存器中的中断使能位被置位，并且通过引脚共用寄存器选择外部中断脚，此引脚将被作为外部中断脚使用。此时该引脚必须通过设置控制寄存器，将该引脚设置为输入口。当中断使能，堆栈未满并且外部中断脚发生了所选择的边沿跳转，将调用外部中断向量子程序。当响应外部中断服务子程序时，中断请求标志位 INT0F~INT3F 会自动复位且 EMI 位会被清零以除能其它中断。注意，即使此引脚被用作外部中断输入，其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。

A/D 转换器中断

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，A/D 转换器中断请求标志位 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

此单片机中有两个多功能中断，与其它中断不同，它没有独立源，而是由其它现有的中断源构成，即 TM 中断和 EEPROM 中断。

当多功能中断中任何一种中断请求发生，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，将调用多功能中断向量子程序。当响应中断服务子程序时，多功能请求标志位会自动复位且 EMI 位会自动清零以除能其它中断。

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

TM 中断

简易型和周期型 TM 各自有两个中断，分别来自比较器 P、A 匹配，都属于多功能中断。所有类型的 TM 都有两个中断请求标志位及两个使能位。当 TM 比较器 P、A 匹配情况发生时，相应 TM 中断请求标志被置位，TM 中断请求产生。

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

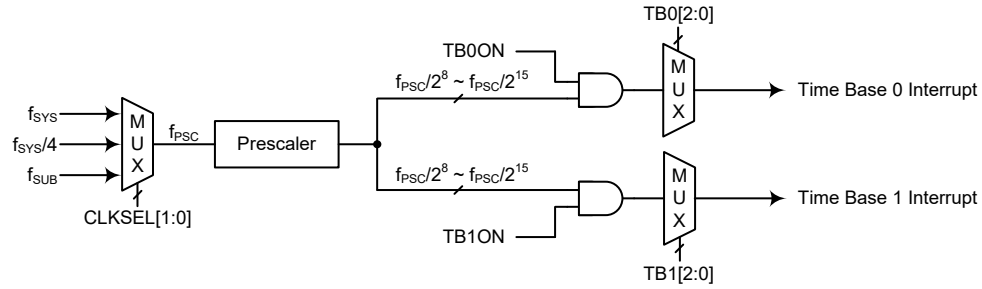
EEPROM 中断

EEPROM 中断属于多功能中断。当擦或写周期结束，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 和相关多功能中断使能位 MFnE 需先被置位。当中断使能，堆栈未满且 EEPROM 擦或写周期结束时，可跳转至相关多功能中断向量子程序中执行。当 EEPROM 中断响应时，EMI 位会被清零以除能其它中断，相关 MFnF 标志也可自动清除，但 EEPROM 中断请求标志需在应用程序中手动清除。

时基中断

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

时基中断的目的是提供一个固定周期的中断信号。其时钟源 f_{PSC} 来自内部时钟源 f_{SYS} 、 $f_{SYS}/4$ 或 f_{SUB} 。 f_{PSC} 输入时钟首先经过分频器，分频率由程序设置 TB0C 和 TB1C 寄存器相关位获取合适的分频值以提供更长的时基中断周期。时基中断周期控制时钟 f_{PSC} 的时钟源可通过 PSCR 寄存器的 CLKSEL[1:0] 位选择。



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 / 除能控制位
 0: 除能
 1: 使能

Bit 6~3 未定义，读为“0”

Bit 2~0 **TB02~TB00**: 选择时基 0 溢出周期位
 000: $2^8/f_{PSC}$
 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 / 除能控制位
 0: 除能
 1: 使能

Bit 6~3	未定义，读为“0”
Bit 2~0	TB12~TB10 : 选择时基 1 溢出周期位
	000: $2^8/f_{PSC}$
	001: $2^9/f_{PSC}$
	010: $2^{10}/f_{PSC}$
	011: $2^{11}/f_{PSC}$
	100: $2^{12}/f_{PSC}$
	101: $2^{13}/f_{PSC}$
	110: $2^{14}/f_{PSC}$
	111: $2^{15}/f_{PSC}$

SIM 中断

串行接口模块中断，即 SIM 中断。当一个字节数据已由 SIM 接口接收或发送完、或 I²C 从机地址匹配、或 I²C 超时，中断请求标志 SIMF 被置位，SIM 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和串行接口中断使能位 SIME 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 SIM 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 SIMF 会自动复位且 EMI 位会被清零以除能其它中断。

比较器中断

比较器中断由在测量电路内的比较器控制。当检测到特定的 OPA 输入信号时，比较器中断请求标志 CMPF 被置位，比较器中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和比较器中断使能位 CMPE 需先被置位。当中断使能，堆栈未满并且比较器输入产生比较器输出变化时，将调用比较器中断向量子程序。当响应中断服务子程序时，比较器中断请求标志位会自动复位且 EMI 位会被清零以除能其它中断。

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可能导致其相应的中断标志被置位，由此产生中断，因此必须注意避免伪唤醒情况的发生。如果要除能中断唤醒功能，单片机进入休眠或空闲模式前相应中断请求标志应被置起。中断唤醒功能不受中断使能位的影响。

编程注意事项

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

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

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

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

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

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

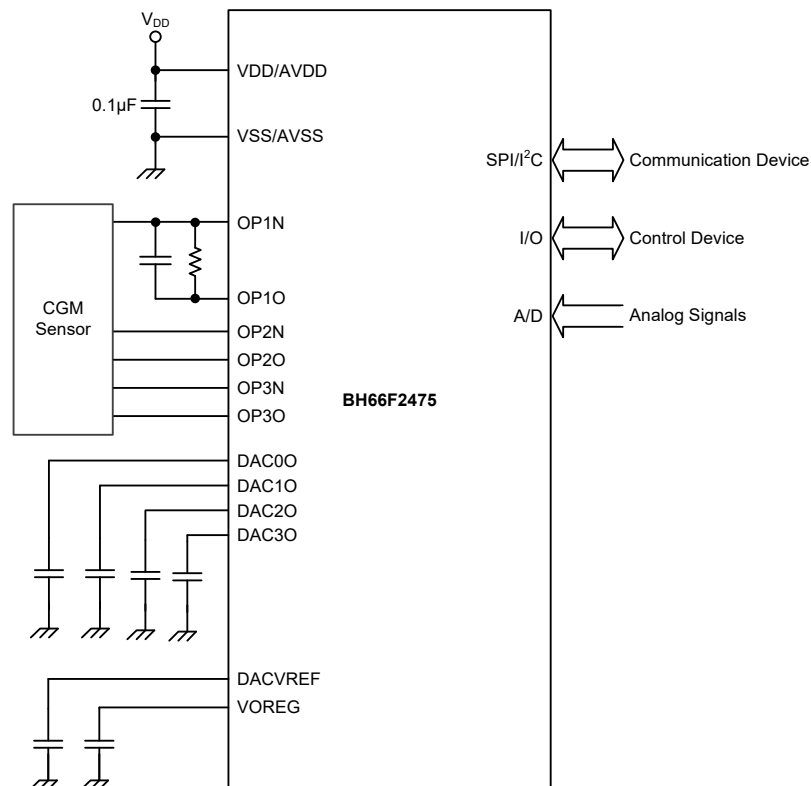
配置选项

配置选项在烧写程序时写入芯片。通过 HT-IDE 的软件开发环境，使用者在开发过程中可以选择配置选项。所有位必须按系统的需要定义，具体内容可参考下表：

序号	选项
振荡器选项	
1	HIRC 频率选择： 4MHz, 8MHz 和 12MHz

注：当 HIRC 配置选项已选定上表中的一个频率时，HIRC1 和 HIRC0 位选择的频率应与其保持一致，以确保能够达到交流电气特性中标示的 HIRC 精准度。

应用电路



指令集

简介

任何单片机成功运作的核心在于它的指令集，此指令集为一组程序指令码，用来指导单片机如何去执行指定的工作。在 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 0.5 μ s 中执行完成，而分支或调用操作则将在 1 μ s 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入口的 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 and TBLH	2 ^注	无
TABRDL [m]	读取最后页的 ROM 内容，并送至数据存储器 and 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 and 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 and 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 and TBLH	3 注	无
LTABRDL [m]	读取最后页的 ROM 内容，并送至数据存储器 and 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 and 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 and 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	$ACC \leftarrow ACC + [m] + C$
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	$[m] \leftarrow ACC + [m] + C$
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	$ACC \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC + x$
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	$[m] \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC \text{ "AND" } [m]$
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	$ACC \leftarrow ACC \text{ “AND” } x$
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	$[m] \leftarrow ACC \text{ “AND” } [m]$
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	$Stack \leftarrow Program\ Counter + 1$ $Program\ Counter \leftarrow addr$
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	$[m] \leftarrow 00H$
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	$[m].i \leftarrow 0$
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 $TO \ \& \ PDF \leftarrow 0$
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	$[m] \leftarrow \overline{[m]}$
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	$ACC \leftarrow \overline{[m]}$
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	$[m] \leftarrow ACC + 00H$ 或 $[m] \leftarrow ACC + 06H$ 或 $[m] \leftarrow ACC + 60H$ 或 $[m] \leftarrow ACC + 66H$
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	$[m] \leftarrow [m] - 1$
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	$ACC \leftarrow [m] - 1$
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	$TO \leftarrow 0$ $PDF \leftarrow 1$
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	$[m] \leftarrow [m] + 1$
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	$ACC \leftarrow [m] + 1$
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	$Program\ Counter \leftarrow addr$
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	$ACC \leftarrow [m]$
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	$ACC \leftarrow x$
影响标志位	无

MOV [m], A 指令说明 功能表示 影响标志位	Move ACC to Data Memory 将累加器的内容复制到指定的数据存储器。 $[m] \leftarrow \text{ACC}$ 无
NOP 指令说明 功能表示 影响标志位	No operation 空操作，接下来顺序执行下一条指令。 无操作 无
ORA, [m] 指令说明 功能表示 影响标志位	Logical OR Data Memory to ACC 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。 $\text{ACC} \leftarrow \text{ACC} \text{ "OR" } [m]$ Z
ORA, x 指令说明 功能表示 影响标志位	Logical OR immediate data to ACC 将累加器中的数据和立即数逻辑或，结果存放到累加器。 $\text{ACC} \leftarrow \text{ACC} \text{ "OR" } x$ Z
ORM A, [m] 指令说明 功能表示 影响标志位	Logical OR ACC to Data Memory 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。 $[m] \leftarrow \text{ACC} \text{ "OR" } [m]$ Z
RET 指令说明 功能表示 影响标志位	Return from subroutine 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。 $\text{Program Counter} \leftarrow \text{Stack}$ 无
RET A, x 指令说明 功能表示 影响标志位	Return from subroutine and load immediate data to ACC 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。 $\text{Program Counter} \leftarrow \text{Stack}$ $\text{ACC} \leftarrow x$ 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 \leftarrow Stack
影响标志位	EMI \leftarrow 1 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	$[m].(i+1) \leftarrow [m].i$ (i=0~6) $[m].0 \leftarrow [m].7$
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) \leftarrow [m].i (i=0~6) ACC.0 \leftarrow [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	$[m].(i+1) \leftarrow [m].i$ (i=0~6) $[m].0 \leftarrow C$ $C \leftarrow [m].7$
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) \leftarrow [m].i (i=0~6) ACC.0 \leftarrow C $C \leftarrow [m].7$
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	$[m].i \leftarrow [m].(i+1) (i=0\sim6)$ $[m].7 \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1) (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
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	$[m].i \leftarrow [m].(i+1) (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1) (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC \leftarrow ACC - [m] - \bar{C}$
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	$ACC \leftarrow ACC - [x] - \bar{C}$
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	$[m] \leftarrow ACC - [m] - \bar{C}$
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	$[m] \leftarrow [m] - 1$ ，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	$ACC \leftarrow [m] - 1$ ，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	$[m] \leftarrow FFH$
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	$[m].i \leftarrow 1$
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	$[m] \leftarrow [m] + 1$ ，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	$ACC \leftarrow [m] + 1$ ，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 $[m].i \neq 0$ ，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 $[m] \neq 0$ ，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	$ACC \leftarrow ACC - [m]$
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	$[m] \leftarrow ACC - [m]$
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	$ACC \leftarrow ACC - x$
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	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
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单元内。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 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	$[m] \leftarrow \text{程序代码 (低字节)}$ $TBLH \leftarrow \text{程序代码 (高字节)}$
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	$[m] \leftarrow \text{程序代码 (低字节)}$ $TBLH \leftarrow \text{程序代码 (高字节)}$
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	$ACC \leftarrow ACC \text{ "XOR" } [m]$
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	$[m] \leftarrow ACC \text{ "XOR" } [m]$
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	$ACC \leftarrow ACC \text{ "XOR" } x$
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	$ACC \leftarrow ACC + [m] + C$
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	$[m] \leftarrow ACC + [m] + C$
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	$ACC \leftarrow ACC + [m]$
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	$[m] \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC \text{ “AND” } [m]$
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	$[m] \leftarrow ACC \text{ “AND” } [m]$
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	$[m] \leftarrow 00H$
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	$[m].i \leftarrow 0$
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	$[m] \leftarrow \overline{[m]}$
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	$ACC \leftarrow \overline{[m]}$
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	$[m] \leftarrow ACC + 00H$ 或 $[m] \leftarrow ACC + 06H$ 或 $[m] \leftarrow ACC + 60H$ 或 $[m] \leftarrow ACC + 66H$
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	$[m] \leftarrow [m] - 1$
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	$ACC \leftarrow [m] - 1$
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	$[m] \leftarrow [m] + 1$
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	$ACC \leftarrow [m] + 1$
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	$ACC \leftarrow [m]$
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	$[m] \leftarrow ACC$
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	$ACC \leftarrow ACC \text{ "OR" } [m]$
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 and 累加器逻辑或，结果放到数据存储器。
功能表示	$[m] \leftarrow ACC \text{ "OR" } [m]$
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	$[m].(i+1) \leftarrow [m].i \ (i=0\sim6)$ $[m].0 \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i \ (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
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	$[m].(i+1) \leftarrow [m].i \ (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i \ (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
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	$[m].i \leftarrow [m].(i+1) (i=0\sim6)$ $[m].7 \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1) (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
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	$[m].i \leftarrow [m].(i+1) (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位	C
LRRC A [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	$ACC.i \leftarrow [m].(i+1) (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC \leftarrow ACC - [m] - \bar{C}$
影响标志位	OV、Z、AC、C、SC、CZ

LSBCM A, [m]	Subtract Data Memory from ACC with Carry and result in Data Memory
指令说明	将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$[m] \leftarrow ACC - [m] - \overline{C}$
影响标志位	OV、Z、AC、C、SC、CZ
LSDZ [m]	Skip if Decrement Data Memory is 0
指令说明	将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 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	$[m] \leftarrow [m] - 1$ ，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	$ACC \leftarrow [m] - 1$ ，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	$[m] \leftarrow FFH$
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	$[m].i \leftarrow 1$
影响标志位	无

LSIZ [m] 指令说明	Skip if increment Data Memory is 0 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	$[m] \leftarrow [m] + 1$ ，如果 $[m]=0$ 跳过下一条指令执行
影响标志位	无
LSIZA [m] 指令说明	Skip if increment Data Memory is zero with result in ACC 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	$ACC \leftarrow [m] + 1$ ，如果 $ACC=0$ 跳过下一条指令执行
影响标志位	无
LSNZ [m].i 指令说明	Skip if bit i of Data Memory is not 0 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 $[m].i \neq 0$ ，跳过下一条指令执行
影响标志位	无
LSNZ [m] 指令说明	Skip if Data Memory is not 0 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 $[m] \neq 0$ ，跳过下一条指令执行
影响标志位	无
LSUBA, [m] 指令说明	Subtract Data Memory from ACC 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	$ACC \leftarrow ACC - [m]$
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	$[m] \leftarrow ACC - [m]$
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	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
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	$ACC \leftarrow [m]$ ，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	$ACC \leftarrow ACC \text{ "XOR" } [m]$
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	$[m] \leftarrow ACC \text{ "XOR" } [m]$
影响标志位	Z

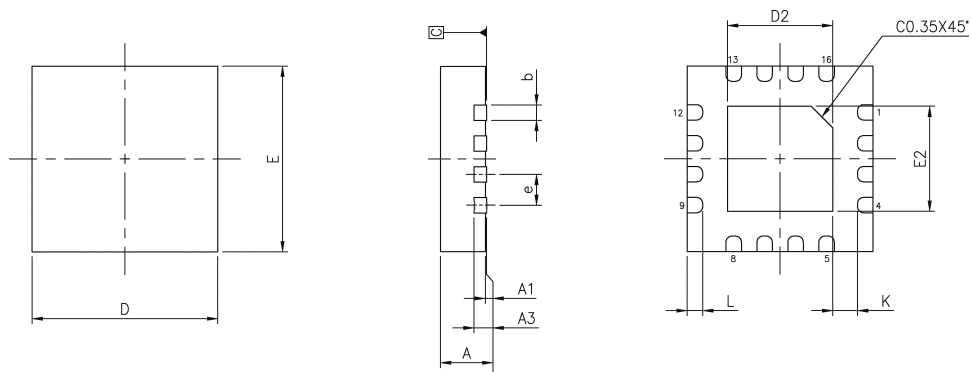
封装信息

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

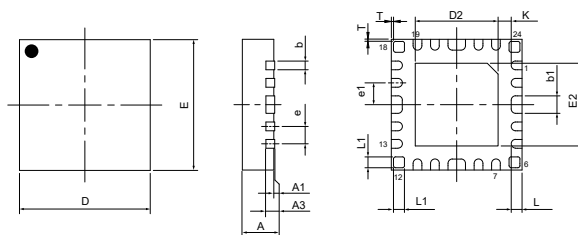
SAW Type 16-pin QFN (3mm×3mm×0.75mm, FP0.25mm) 外形尺寸



符号	尺寸 (单位: inch)		
	最小值	典型值	最大值
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3	0.008 REF		
b	0.007	0.010	0.012
D	0.118 BSC		
E	0.118 BSC		
e	0.020 BSC		
D2	0.063	—	0.069
E2	0.063	—	0.069
L	0.008	0.010	0.012
K	0.008	—	—

符号	尺寸 (单位: mm)		
	最小值	典型值	最大值
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	0.203 REF		
b	0.18	0.25	0.30
D	3.00 BSC		
E	3.00 BSC		
e	0.50 BSC		
D2	1.60	—	1.75
E2	1.60	—	1.75
L	0.20	0.25	0.30
K	0.20	—	—

SAW Type 24-pin QFN (3mm×3mm×0.55mm) 外形尺寸



符号	尺寸 (单位: inch)		
	最小值	典型值	最大值
A	0.020	0.022	0.024
A1	0.000	0.001	0.002
A3	0.006 REF		
b	0.006	0.008	0.010
b1	0.014	0.016	0.018
D	0.118 BSC		
E	0.118 BSC		
e	0.016 BSC		
e1	0.020 BSC		
D2	0.073	—	0.077
E2	0.073	—	0.077
L	0.006	0.010	0.014
L1	0.008	0.010	0.012
K	0.008	—	—
T	0.000	0.002	0.004

符号	尺寸 (单位: mm)		
	最小值	典型值	最大值
A	0.50	0.55	0.60
A1	0.00	0.02	0.05
A3	0.15 REF		
b	0.15	0.20	0.25
b1	0.35	0.40	0.45
D	3.00 BSC		
E	3.00 BSC		
e	0.40 BSC		
e1	0.50 BSC		
D2	1.85	—	1.95
E2	1.85	—	1.95
L	0.15	0.25	0.35
L1	0.20	0.25	0.30
K	0.20	—	—
T	0.00	0.05	0.10

Copyright© 2023 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

本文件出版时 HOLTEK 已针对所载信息为合理注意，但不保证信息准确无误。文中提到的信息仅是提供作为参考，且可能被更新取代。HOLTEK 不担保任何明示、默示或法定的，包括但不限于适合商品化、令人满意的质量、规格、特性、功能与特定用途、不侵害第三方权利等保证责任。HOLTEK 就文中提到的信息及该信息之应用，不承担任何法律责任。此外，HOLTEK 并不推荐将 HOLTEK 的产品使用在会由于故障或其他原因而可能会对人身安全造成危害的地方。HOLTEK 特此声明，不授权将产品使用于救生、维生或安全关键零部件。在救生 / 维生或安全应用中使用 HOLTEK 产品的风险完全由买方承担，如因该等使用导致 HOLTEK 遭受损害、索赔、诉讼或产生费用，买方同意出面进行辩护、赔偿并使 HOLTEK 免受损害。HOLTEK (及其授权方，如适用) 拥有本文件所提供信息 (包括但不限于内容、数据、示例、材料、图形、商标) 的知识产权，且该信息受著作权法和其他知识产权法的保护。HOLTEK 在此并未明示或暗示授予任何知识产权。HOLTEK 拥有不事先通知而修改本文件所载信息的权利。如欲取得最新的信息，请与我们联系。