连续血糖监测 Flash 单片机

BH66F2455

版本：V1.10　日期：2025-12-22

www.holtek.com

目录

特性

CPU 特性

- 工作电压：
  - ♦ MIRC
    - – f_{SYS}=400kHz：2.2V~5.5V
  - ♦ HIRC
    - – f_{SYS}=4MHz：2.2V~5.5V
- V_{DD}=5V，系统时钟为 4MHz 时，指令周期为 1μs
- 暂停和唤醒功能，以降低功耗
- 振荡器类型：
  - ♦ 内部高速 4MHz RC – HIRC
  - ♦ 内部中速 400kHz RC – MIRC
  - ♦ 内部低速 32.768kHz RC – LIRC
- 内部集成的振荡器，无需外接元件
- 多种工作模式：快速模式、低速模式、空闲模式和休眠模式
- 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：8K×16
- 数据存储器：512×8
- True EEPROM 存储器：512×8
- 在线应用编程功能 – IAP
- 128-bit 唯一 ID
- 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
- 两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
- 血糖仪 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
  - ♦ 3 个 12-bit D/A 转换器
  - ♦ 3 个运算放大器
- 全双工 / 半双工通用异步通信接口 – UART
- 独立串行外设接口 – SPI

- 内建 16-bit 循环冗余校验功能 – CRC
- 低电压复位功能
- 封装类型：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。

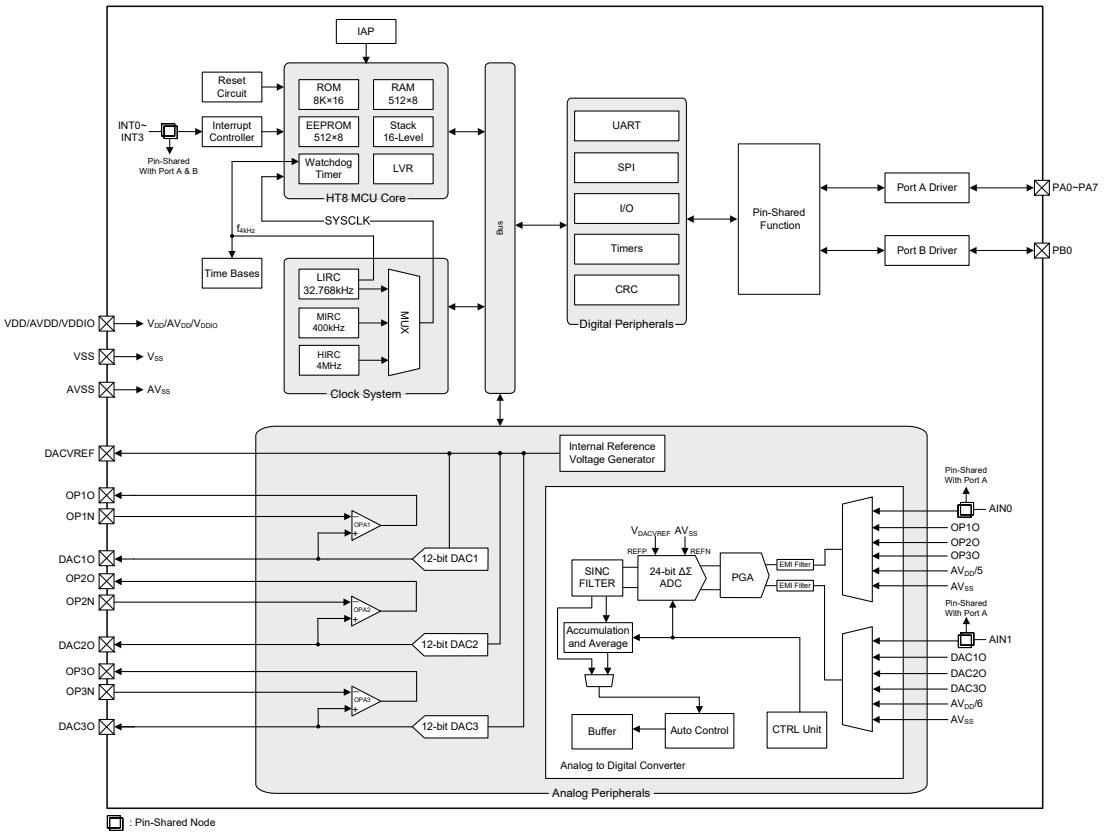
在存储器特性方面，Flash 存储器可多次编程的特性给用户提供了较大的方便。除了 Flash 程序存储器，还包含了一个 RAM 数据存储器和一个可用于存储序号、校准数据等非易失性数据的 True EEPROM 存储器。此外通过使用 IAP 功能，便于用户直接将测量的数据存储至程序存储器中或进行应用程序更新。

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、3 个 D/A 转换器和 3 个运算放大器。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 UART 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。

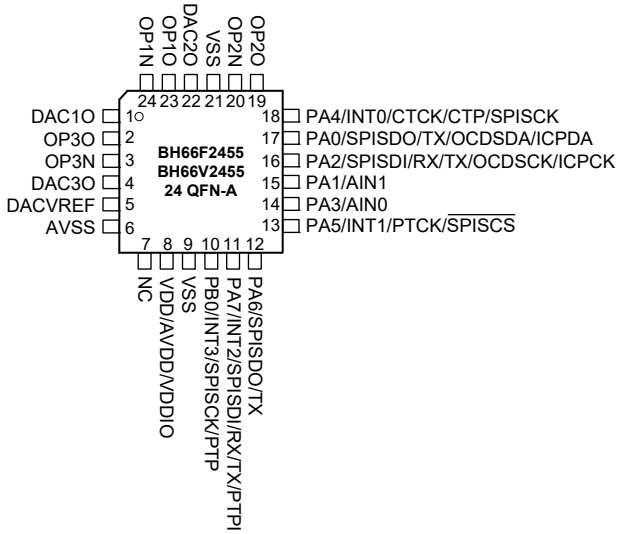
该单片机提供了丰富的内部高速和低速振荡器功能选项，完全内置的三个系统振荡器，无需外围元器件。其在不同工作模式之间动态切换的能力，为用户提供了一个优化单片机操作和减少功耗的手段。

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、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 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. OCDSCK 和 OCDSDA 引脚为片上调试功能专用引脚，仅存在于 BH66F2455 的 EV 芯片 BH66V2455。

引脚说明

每个引脚的功能如下表所述，而引脚配置的详细内容见规格书其它章节。

引脚名称	功能	OPT	I/T	O/T	说明
PA0/SPISDO/TX/OCDSDA/ICPDA	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
	SPISDO	PAS0	—	CMOS	SPI 数据输出
	TX	PAS0	—	CMOS	UART 串行数据输出
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址引脚，仅用于 EV 芯片
	ICPDA	—	ST	CMOS	ICP 数据 / 地址引脚
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/SPISDI/RX/TX/OCDSCK/ICPCK	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
	SPISDI	PAS0 IFS	ST	—	SPI 数据输入
	RX/TX	PAS0 IFS	ST	CMOS	UART 串行数据输入 (全双工通信)，UART 串行数据输入 / 输出 (单线通信模式)
	OCDSCK	—	ST	—	OCDS 时钟引脚，仅用于 EV 芯片
	ICPCK	—	ST	—	ICP 时钟引脚
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/CTCK/CTP/SPISCK	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
	CTCK	PAS1	ST	—	CTM 时钟输入
	CTP	PAS1	—	CMOS	CTM 输出
	SPISCK	PAS1 IFS	ST	CMOS	SPI 串行时钟
PA5/INT1/PTCK/$\overline{\text{SPISCS}}$	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
	$\overline{\text{SPISCS}}$	PAS1	ST	CMOS	SPI 从机选择

引脚名称	功能	OPT	I/T	O/T	说明
PA6/SPISDO/TX	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
	SPISDO	PAS1	—	CMOS	SPI 数据输出
	TX	PAS1	—	CMOS	UART 串行数据输出
PA7/INT2/SPISDI/ RX/TX/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
	SPISDI	PAS1 IFS	ST	—	SPI 数据输入
	RX/TX	PAS1 IFS	ST	CMOS	UART 串行数据输入 (全双工通信)，UART 串行数据输入 / 输出 (单线通信模式)
	PTPI	PAS1	ST	—	PTM 捕捉输入
PB0/INT3/SPISCK/ PTP	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
	SPISCK	PBS0 IFS	ST	CMOS	SPI 串行时钟
	PTP	PBS0	—	CMOS	PTM 输出
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 转换器参考电压输出
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/VDDIO	VDD	—	PWR	—	数字正电源
	AVDD	—	PWR	—	模拟正电源
	VDDIO	—	PWR	—	数字输入 / 输出引脚功能的电源
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；
PWR：电源
O/T：输出类型；
ST：施密特触发输入；
AN：模拟信号；

极限参数

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

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

直流电气特性

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

工作电压特性

Ta=-40°C~85°C

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 – HIRC	f_{SYS}=4MHz	2.2	—	5.5	V
	工作电压 – MIRC	f_{SYS}=400kHz	2.2	—	5.5	V
	工作电压 – LIRC	f_{SYS}=32.768kHz	2.2	—	5.5	V

工作电流特性

Ta=-40°C~85°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}=32.768kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	15	30	
		2.2V	f_{SYS}=32.768kHz, LVR 使能	—	10	20	
		3V		—	15	30	
		5V		—	20	40	
	快速模式 – MIRC	2.2V	f_{SYS}=400kHz	—	20	35	μA
		3V		—	36	47	
		5V		—	74	96	
	快速模式 – HIRC	2.2V	f_{SYS}=4MHz	—	0.16	0.19	mA
		3V		—	0.37	0.45	
		5V		—	0.75	0.90	

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 off, Time Base on, LIRC on	—	1.5	3.0	3.6	μA
		3V		—	1.5	3.0	3.7	
		5V		—	3	5	6	
		2.2V	WDT on, LIRC on	—	1.5	3.0	3.6	μA
		3V		—	1.5	3.0	3.7	
		5V		—	3	5	6	
	空闲模式 0 – LIRC	2.2V	f_{SUB} on	—	3.0	5.0	5.7	μA
		3V		—	3.0	5.0	5.7	
		5V		—	5	10	11	
	空闲模式 1 – MIRC	2.2V	f_{SUB} on, f_{SYS}=400kHz	—	8.5	11.0	19.0	μA
		3V		—	13.6	17.7	30.0	
		5V		—	26.4	34.3	56.0	
	空闲模式 1 – HIRC	2.2V	f_{SUB} on， f_{SYS}=4MHz	—	144	180	200	μA
		3V		—	180	220	230	
		5V		—	350	420	450	

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 – 频率精准度

程序烧录时，烧录器会依据用户选择的 HIRC 频率和工作电压 (3V 或 5V) 对 HIRC 进行频率精准度调整。

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{HIRC}	通过烧录器调整后的 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%	

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 3.3V~5.5V 时，建议烧录器电压固定在 5V。

内部中速振荡器电气特性 – MIRC

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{MIRC}	MIRC 频率	3V/5V	25°C	-2%	400	+2%	kHz
		2.2V~5.5V	-40°C~85°C	-7%	400	+7%	
t_{START}	MIRC 启动时间	—	-40°C~85°C	—	—	100	μs

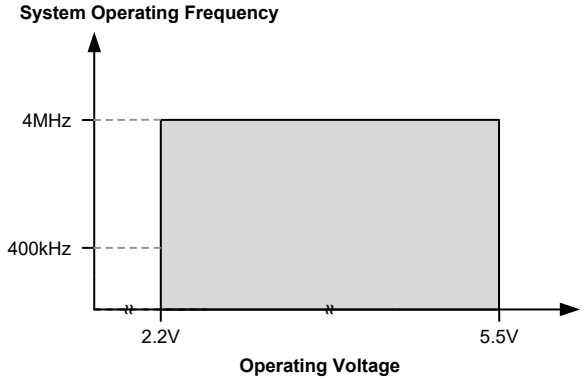
内部低速振荡器电气特性 – LIRC

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	3V	25°C	-1%	32.768	1%	kHz
			-10°C~50°C	-4%	32.768	4%	
			-40°C~85°C	-6%	32.768	6%	
		2.2V~3.6V	25°C	-4%	32.768	4%	
			-40°C~85°C	-10%	32.768	+10%	
t_{START}	LIRC 启动时间	—	-40°C~85°C	—	—	100	μs

注：1. 烧录器可在 3V 这个固定电压下对 LIRC 频率进行调整，在此提供 V_{DD}=3V 时的参数值。

2. 3V 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。

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)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$ 或 f_{MIRC}	—	16	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	系统启动时间 (从 f_{SYS} on 的状态下唤醒)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$ 或 f_{MIRC}	—	2	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	系统速度切换时间 (快速模式 → 低速模式或 低速模式 → 快速模式)	—	f_{HIRC} 或 f_{MIRC} off → on	—	16	—	t_{HIRC} 或 t_{MIRC}
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 / 输出口电气特性

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}	单片机电源	—	—	2.2	—	5.5	V
V_{DDIO}	I/O 口电源	—	—	2.2	—	V_{DD}	V
V_{IL}	I/O 口低电平输入电压	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DDIO}$	
V_{IH}	I/O 口高电平输入电压	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DDIO}$	—	V_{DDIO}	
I_{OL}	I/O 口灌电流	3V	$V_{OL}=0.1V_{DDIO}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	I/O 口源电流	3V	$V_{OH}=0.9V_{DDIO}$	-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	5V	$V_{IN}=V_{DDIO}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
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}	xTM 时钟输入最小脉宽	—	—	0.3	—	—	μs

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
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，$t_{CPW}=max(2\times t_{TMCLK}, t_{TPI})$

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

例 1：若 PTCAPTS=0，f_{TMCLK}=4MHz，t_{TPI}=0.3μs，则 $t_{CPW}=max(0.5\mu s, 0.3\mu s)=0.5\mu s$

例 2：若 PTCAPTS=1，f_{TMCLK}=4MHz，t_{TCK}=0.3μs，则 $t_{CPW}=max(0.5\mu s, 0.3\mu s)=0.5\mu s$

存储器电气特性

Ta=-40°C~85°C，除非另有说明

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
Flash 程序存储器							
t_{FER}	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	—	—	1	—	2	t_{LIRC}
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_{LVR}	低电压复位电压	—	LVR 使能，电压选择 1.7V	-5%	1.7	+5%	V
		—	LVR 使能，电压选择 1.9V	-3%	1.9	+3%	
		—	LVR 使能，电压选择 2.1V		2.1		
		—	LVR 使能，电压选择 2.55V		2.55		
		—	LVR 使能，电压选择 3.15V		3.15		
t_{LVR}	产生 LVR 复位的低电压最短保持时间	—	TLVR[1:0]=00B	80	240	480	μs
			TLVR[1:0]=01B	0.4	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	3V	—	—	4.5	6.0	μA
		5V	—	—	6.5	8.0	μA

模拟前端电路特性

工作电压电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
AV_{DD}	工作电压	—	—	2.2	—	3.6	V

OPA 电气特性

Ta=25°C，除非另有说明

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{OPA}	OPA 使能的额外电流	—	I_{OUT}=0	—	0.6	1.0	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	—	—	AV_{SS}	–	AV_{DD}	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=25°C，除非另有说明

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{IREF}	内部参考电压	3V	调整 V_{IREF}=1.25V±3% @ V_{IREF}=1.25V, IREFEN=1, PVREF=10000000B	-3%	1.25	+3%	V
		3V	调整 V_{IREF}=1.83V±4% @ V_{IREF}=1.83V, IREFEN=1, PVREF=10000000B	-4%	1.83	+4%	V
I_{IREF}	内部参考电压使能的额外电流	—	IREFEN=1	—	1.6	—	μA
TC_{IREF}	内部参考电压温度系数	3V	Ta=10°C~40°C	—	±40	—	ppm/°C

12-bit D/A 转换器电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
DNL	非线性差分误差	3V	DACVRS[1:0]=00B	—	—	±8	LSB
I_{DAC}	D/A 转换器使能的额外电流	—	—	—	0.4	—	μA
INL	非线性积分误差	3V	DACVRS[1:0]=00B	—	—	±20	LSB
Ro	R2R 输出电阻	3V	—	—	3480	—	kΩ
V_{DACO}	输出电压范围	—	—	0.00	—	1.00	$V_{DACVREF}$

A/D 转换器电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{ADC}	A/D 转换器和 PGA 使能的额外电流	3V	—	—	500	800	μA
I_{ADSTB}	待机电流	3V	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	—	AV_{DD}=2.6V, V_{REF}=1.25V, ΔSI=±450mV, ADGN=1, PGAGN=1	—	±50	—	ppm
NFB	无噪音位	—	f_{MCLK}=400kHz, FLMSPS=1, AV_{DD}=2.6V, V_{REF}=1.25V, ADGN=1, PGAGN=2, OSR=32768	—	17.7	—	Bit
ENOB	有效位	—	f_{MCLK}=400kHz, FLMSPS=1, AV_{DD}=2.6V, V_{REF}=1.25V, ADGN=1, PGAGN=2, OSR=32768	—	20.4	—	Bit
f_{ADCK}	A/D 转换器时钟频率	—	—	40.0	409.6	440.0	kHz

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{ADO}	A/D 转换器输出数据传输速率	—	f_{MCLK}=4MHz, FLMSPS=0, FLMS[2:0]=000B, SINC3[1:0]=11B	4	—	1042	Hz
		—	f_{MCLK}=4MHz, FLMSPS=0, FLMS[2:0]=010B, SINC3[1:0]=11B	10	—	2604	Hz
		—	f_{MCLK}=400kHz, FLMSPS=1, SINC3[1:0]=11B	12	—	3125	Hz
V_{REFP}	参考输入电压	—	—	V_{REFN} +0.8	—	$V_{DACVREF}$	V
V_{REFN}		—	—	0	—	V_{REFP} -0.8	V
V_{REF}		—	V_{REF}=(V_{REFP} - V_{REFN})	0.80	—	1.83	V

有效位数 – ENOB

AV_{DD}=V_{DD}=2.6V, V_{REF}=1.25V, ADGN = 1, FLMSPS=1, f_{ADCK}=400kHz, SINC3[1:0]=11B

数据传输速率 (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
12	20.4	20.4	20.4	20.2	20.0	19.4	18.6	17.5
24	20.4	20.4	20.1	20.1	19.6	19.0	18.1	17.2
49	20.0	20.0	20.0	19.7	19.2	18.6	17.6	16.7
98	19.5	19.5	19.5	19.2	18.8	18.0	17.1	16.2
195	18.4	18.4	18.4	18.4	18.1	17.5	16.6	15.6
391	17.4	17.4	17.4	17.4	17.2	16.8	16.0	15.1
781	16.9	16.9	16.9	16.8	16.6	16.1	15.3	14.5
1563	16.5	16.5	16.4	16.3	16.0	15.5	14.7	13.9
3125	15.9	15.9	15.8	15.7	15.4	14.9	14.2	13.6

PGA 转换器电气特性

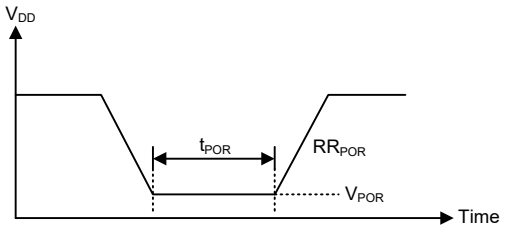
Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{CM_PGA}	共模电压范围	—	—	0.4	—	AV_{DD} -1.1	V
ΔD_I	差分输入电压范围	—	Gain=PGAGN×ADGN	-V_{REF} /Gain	—	+V_{REF} /Gain	V

上电复位特性

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_{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



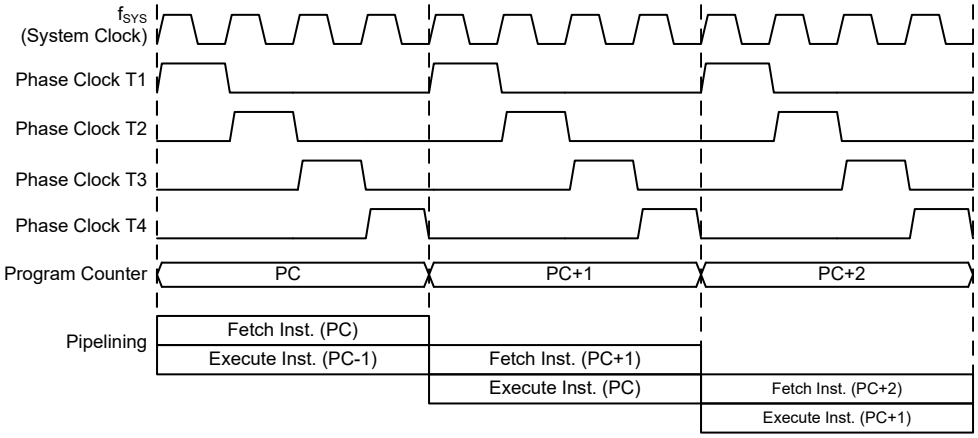
系统结构

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经济型和大量生产的控制应用。

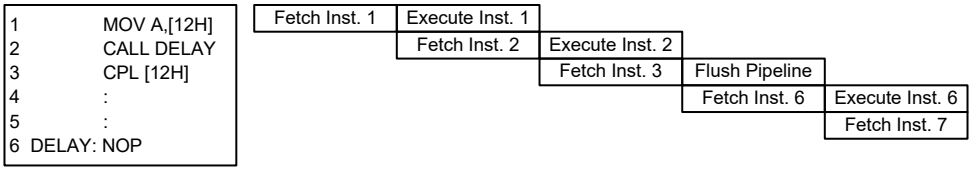
时序和流水线结构

主系统时钟由 HIRC、MIRC 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。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)
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。

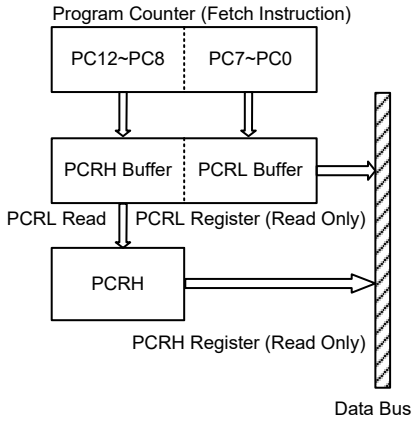
程序计数器读寄存器

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**：程序计数器读低字节寄存器 bit 7 ~ bit 0

- **PCRH 寄存器**

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

Bit 7~5 未定义，读为“0”

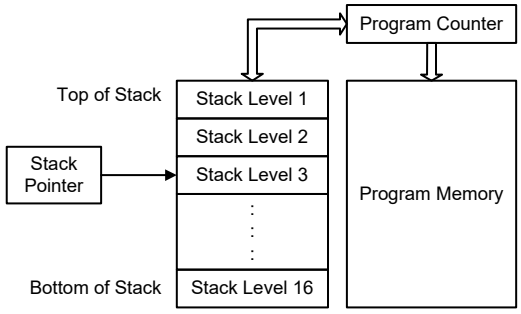
Bit 4~0 **D12~D8**：程序计数器读高字节寄存器 bit 4 ~ bit 0

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 位的变化如下：

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 位的变化如下：

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

- 算术运算：

ADD，ADDM，ADC，ADCM，SUB，SUBM，SBC，SBCM，DAA

LADD，LADDM，LADC，LADCM，LSUB，LSUBM，LSBC，LSBCM，LDAA

- 逻辑运算：

  AND，OR，XOR，ANDM，ORM，XORM，CPL，CPLA

  LAND，LANDM，LOR，LORM，LXOR，LXORM，LCPL，LCPLA

- 移位运算：

  RRA，RR，RRCA，RRC，RLA，RL，RLCA，RLC

  LRRA，LRR，LRRCA，LRRC，LRLA，LRL，LRLCA，LRLC

- 递增和递减：

  INCA，INC，DECA，DEC

  LINCA，LINC，LDECA，LDEC

- 分支判断：

  JMP，SZ，SZA，SNZ，SIZ，SDZ，SIZA，SDZA，CALL，RET，RETI

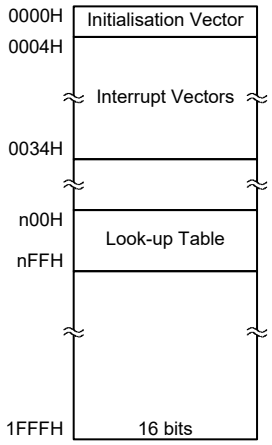
  LSNZ，LSZ，LSZA，LSIZ，LSIZA，LSDZ，LSDZA

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 8K×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

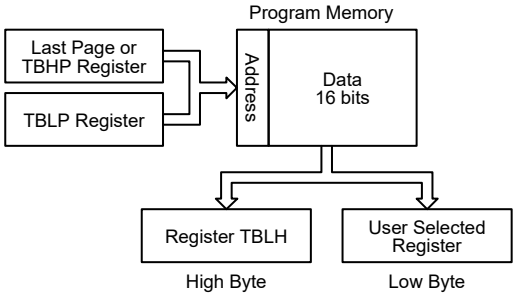
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 0000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

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”指向的地址是 8K 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 1F06H，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

```
tempreg1 db?        ; temporary register #1
tempreg2 db?        ; temporary register #2
:
:
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,1FH           ; 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 "1F06H" 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 "1F05H" 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
:
:
org 1F00H           ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

唯一 ID – UID

该单片机内建一个 128-bit 唯一 ID。其不可更改，由单片机制造商决定。UID 格式如下：

映射在程序存储器最后一页的地址		UID 序号
0xE8	低字节	UID0
	高字节	UID1
0xE9	低字节	UID2
	高字节	UID3
0xEA	低字节	UID4
	高字节	UID5
0xEB	低字节	UID6
	高字节	UID7
0xEC	低字节	UID8
	高字节	UID9
0xED	低字节	UID10
	高字节	UID11
0xEE	低字节	UID12
	高字节	UID13
0xEF	低字节	CRC16[7:0]
	高字节	CRC16[15:8]

UID 位于 Option 存储器的 28H~2FH 地址，该地址会映射到程序存储器最后一页的 E8H~EFH 地址。通过 ORMC 寄存器使能 Option 存储器映射功能后，可通过查表指令在程序存储器最后一页相应位置读取到 UID 内容。具体操作请参考特殊功能寄存器章节中“Option 存储器映射寄存器 – ORMC”内容说明。

CRC 说明

计算顺序：UID0→UID1→UID2→⋯→UID11→UID12→UID13。

CRC 规格：

生成多项式：1021H，$X^{16}+X^{12}+X^{5}+1$

Polynomial=1021H

Initial Value=0000H

Final Xor Value=0000H

Input reflected=No

Output reflected=No

- **CRC 计算范例**

连续输入以下 4 个字节数据，CRC 校验和依序如下表所示：

CRC 数据输入 / CRC 多项式	78H→56H→34H→12H
1021H ($X^{16}+X^{12}+X^5+1$)	FF9FH→BBC3H→A367H→D0FAH

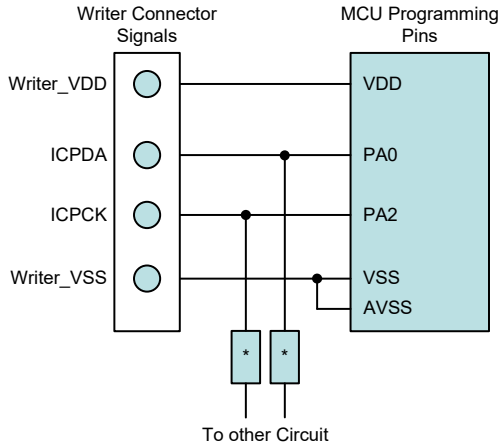
在线烧录 – ICP

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

EV 芯片 BH66V2455 用于 BH66F2455 单片机仿真。此 EV 芯片提供片上调试功能 (OCDS) 用于开发过程中的单片机调试。除了片上调试功能，单片机和 EV 芯片在功能上几乎是兼容的。用户可将 OCDSDA 和 OCDSCK 引脚连接至 Holtek

HT-IDE 开发工具，从而实现单片机的仿真。OCDSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户用 EV 芯片进行调试时，单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能对 EV 芯片无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。关于 OCDS 功能的详细描述，请参考“Holtek e-Link for 8-bit MCU OCDS 用户手册”文件。

Holtek e-Link 引脚名称	EV 芯片引脚名称	功能
OCDSDA	OCDSDA	片上调试串行数据 / 地址输入 / 输出
OCDSCK	OCDSCK	片上调试时钟输入
VDD	VDD	电源
VSS	VSS&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，例如 UART。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 32 字。注意，在执行写入操作之前必须先执行擦除操作。

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

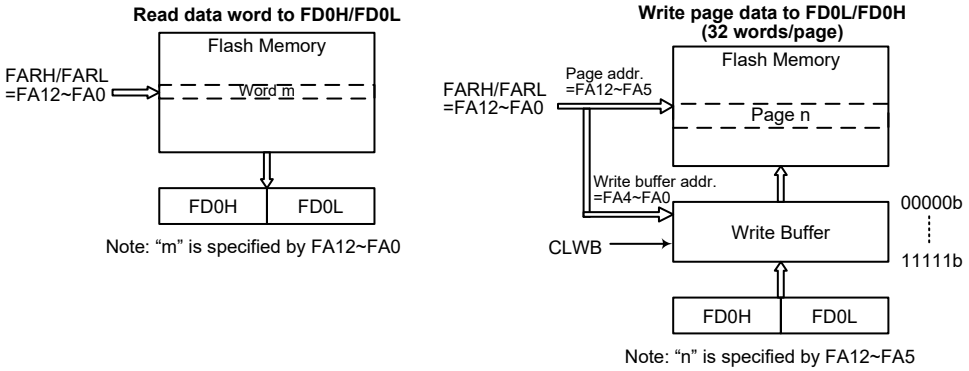
读出操作是通过一个特定的读出程序来执行的。FRDEN 位用于使能读出功能，由应用程序设置 FRD 位来启动读出程序，并指示读出操作的状态。当读出操作结束后该位将由硬件清零。

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

IAP 操作格式

页	FARH	FARL [7:5]	FARL [4:0]
0	0000 0000	000	标记地址
1	0000 0000	001	
2	0000 0000	010	
3	0000 0000	011	
4	0000 0000	100	
: :	: :	: :	
254	0001 1111	110	
255	0001 1111	111	

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 32 字。写入缓冲器的地址与存储器地址位 FA12~FA5 指定的 Flash 存储器页的地址相对应。写入到 FD0L 和 FD0H 寄存器的数据会被加载到写入缓冲器。当写入数据到高字节数据寄存器 FD0H 时，会将存储在 FD0L 和 FD0H 数据寄存器内的数据都加载到写入缓冲器，并使 Flash 存储器地址自动加一，之后新的地址会被加载到 FARH 和 FARL 地址寄存器。当 Flash 存储器地址到达当前页的最大地址，即 32 字的页为 11111b，地址将不再增加，并停在该页的最后一个地址，此时需要再设定一个新的页地址才可进行其它擦 / 写操作。

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

IAP Flash 程序存储器寄存器

与 IAP 相关的 Flash 存取寄存器有两个地址寄存器、四对 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对 MP1H/MP1L 或 MP2H/MP2L 和间接寻址寄存器 IAR1 或 IAR2 进行间接读取或写入。

寄存器名称	位							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	FWERTS	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	—	—	—	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

● **FARH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	FA12	FA11	FA10	FA9	FA8
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　**FA12~FA8**：Flash 存储器地址 bit 12 ~ 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

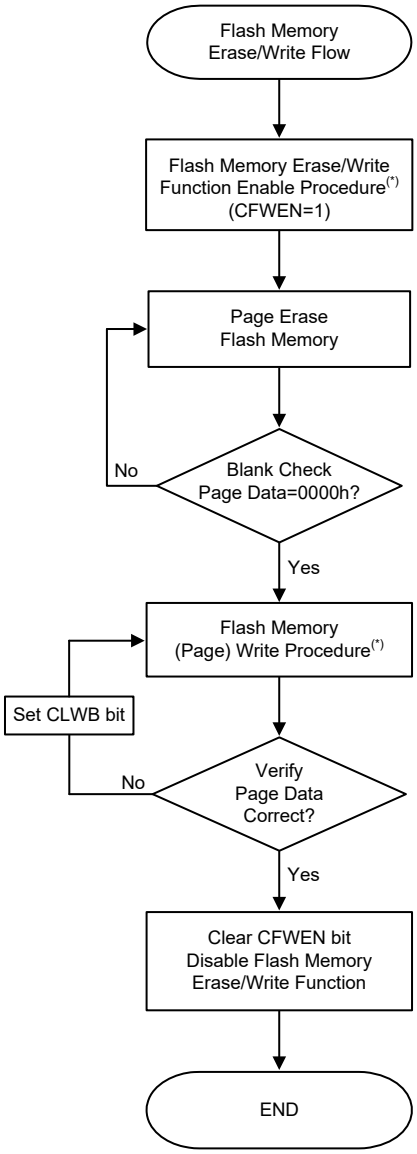
Flash 存储器擦 / 写流程

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 11111b，地址将不再增加，并停在该页的最后一个地址。注意写数据到 FD0H 是为了标记地址，这一操作必须执行以确定要擦除哪些地址。
3. 查空确认是否擦除成功，可采用 TABRD 指令进行读取并比对是否为“0000h”，如果擦除不成功返回步骤 2 再执行页擦除。
4. 写入数据至该页，详细内容请参考“Flash 存储器写入程序”。
5. 采用 TABRD 指令进行读取并比对写入数据是否正确，如果写入不成功，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：标上 * 的“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。

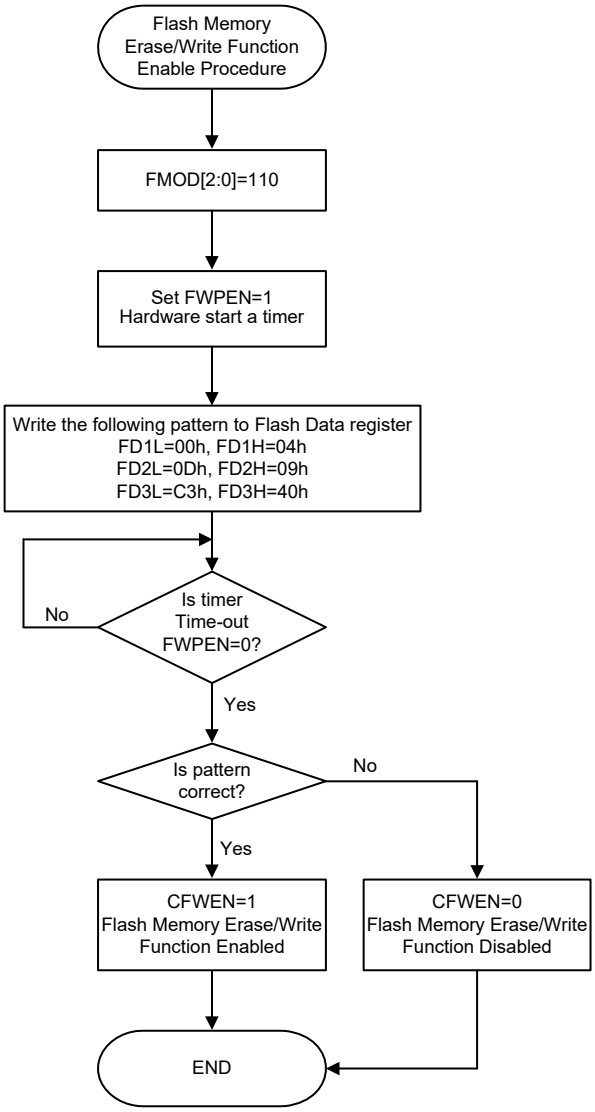
Flash 存储器擦 / 写使能程序

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 32 个字，其地址与 FA12~FA5 指定的 Flash 存储器页的地址为相对应关系。注意，写入缓冲器的地址与对应存储器的地址必须在相同页。

Flash 存储器连续地址写入步骤说明

对于写入操作每次写入的数据最多为 32 字。多笔连续地址的数据写入时，写入缓冲器的地址将自动加“1”。用户只需将第一笔数据的地址填入 FARL 和 FARH，并将第一笔数据依序填入 FD0L 和 FD0H 寄存器 (先写 FD0L 再写 FD0H，才会将 FD0L 和 FD0H 数据一起填入写入缓冲器)，写入缓冲器的地址将自动加“1”，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。

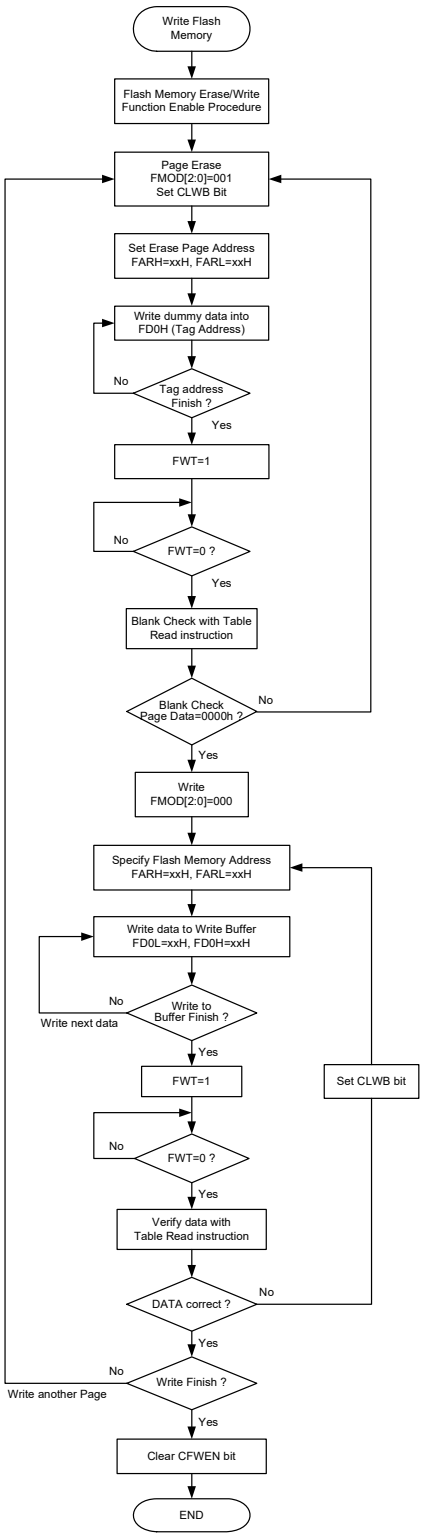
1. 启动“Flash 存储器擦 / 写使能程序”，确认 CFWEN 的值，如果 CFWEN 被硬件置高，表示可进行 IAP 擦 / 写操作。详细内容请参考“Flash 存储器擦 / 写使能程序”。
2. 设定 FMOD[2:0] 为“001”，选择擦除模式，并设定 CLWB 位为“1”，清除“写入缓冲器”。设定 FWT 位为“1”，擦除 FARH 和 FARL 指定且标记地址的目标页，直到 FWT 变为“0”。
3. 通过查表指令读出方式进行查空，以确保擦除操作已成功完成。

   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 32 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。

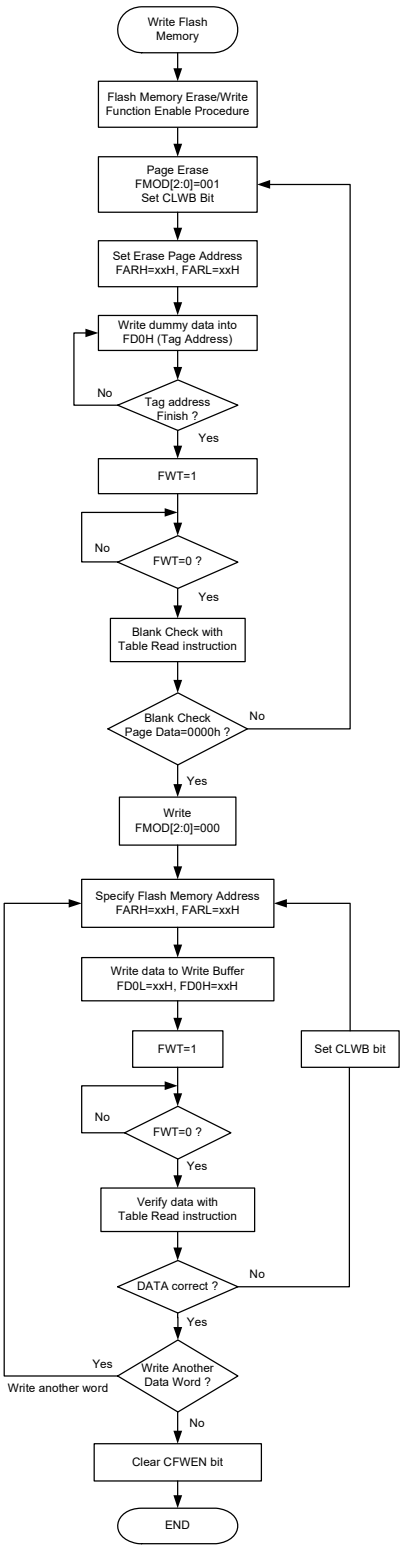
2. 在擦除或写入操作中，FWT 位由高变低所需时间可以通过 FC2 寄存器中的 FWERTS 位选择。

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。

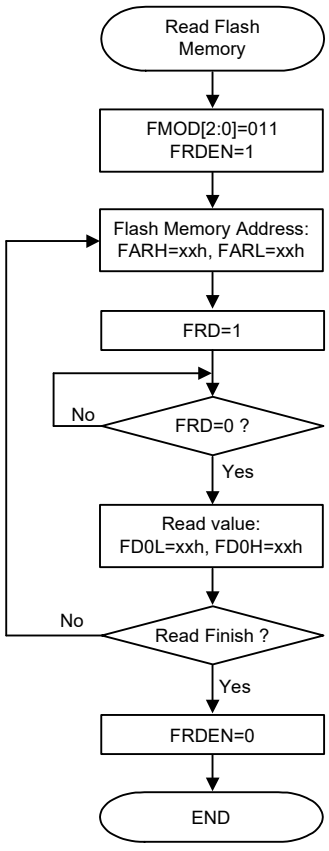
2. 在擦除或写入操作中，FWT 位由高变低所需时间可以通过 FC2 寄存器中的 FWERTS 位选择。

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



Flash 存储器读出步骤

注：1. 当读动作成功启动后，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。

该单片机还提供了专门的存储区用于存放 A/D 转换器自动模式数据。

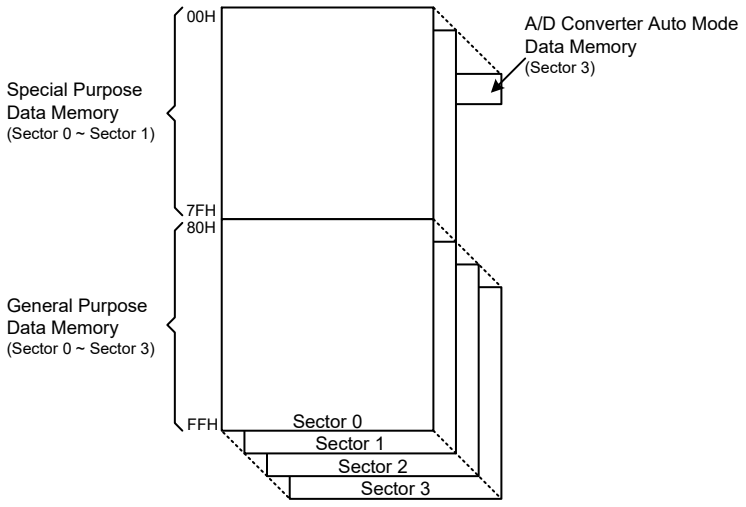
当使用间接寻址时，切换不同的数据存储器 Sector 通过设置正确的间接寻址指针值实现。

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。A/D 转换器自动模式数据存储器位于 Sector 3 的 00H~3FH。

特殊功能数据存储器	通用数据存储器		A/D 转换器自动模式数据存储器
位于 Sector	容量	Sector：地址	Sector：地址
Sector 0：00H~7FH Sector 1：00H~7FH	512×8	0：80H~FFH 1：80H~FFH 2：80H~FFH 3：80H~FFH	Sector 3：00H~3FH

数据存储器概要



数据存储器结构

数据存储器寻址

此单片机支持扩展指令架构，它并没有可用于数据存储器的存储区指针。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 10 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
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH		
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	SCC	
11H	IRCC	
12H	STKPTR	
13H	IECC	
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	RSTC	
19H	LVRC	
1AH	TLVRC	
1BH	MFI0	
1CH	MFI1	
1DH		
1EH	WDTC	
1FH	INTEG	
20H	INTC0	
21H	INTC1	
22H	INTC2	
23H	INTC3	
24H		
25H	PBC	
26H		
27H	PCRL	
28H	PCRH	
29H		
2AH		
2BH		
2CH	PSCR	
2DH	TB0C	
2EH	TB1C	
2FH	USR	
30H	UCR1	FC0
31H	UCR2	FC1
32H	UCR3	FC2
33H	BRDH	FARL
34H	BRDL	FARH
35H	UFCR	FD0L
36H	TXR_RXR	FD0H
37H	RxCNT	FD1L
38H	PTMC0	FD1H
39H	PTMC1	FD2L
3AH	PTMDL	FD2H
3BH	PTMDH	FD3L
3CH	PTMAL	FD3H
3DH	PTMAH	IFS
3EH	PTMRPL	
3FH	PTMRPH	

	Sector 0	Sector 1
40H		EEC
41H	EEAL	
42H	EEAH	
43H	EED	
44H	PWRC	
45H	IREFC	
46H	PVREF	
47H	OPAC	
48H	GSC1	
49H	AFEDA1C	
4AH	AFEDA1L	
4BH	AFEDA1H	
4CH	AFEDA2C	
4DH	AFEDA2L	
4EH	AFEDA2H	
4FH	AFEDA3C	
50H	AFEDA3L	
51H	AFEDA3H	
52H		
53H	PGAC	
54H	PGACS	
55H	MODC	
56H	ADRL	
57H	ADRM	
58H	ADRH	
59H	ADCR0	
5AH	ADCR1	
5BH		
5CH		
5DH	ADCS	
5EH		
5FH	SINC3	
60H	ADACCTRL	
61H	ADRL_AVG	
62H	ADRM_AVG	
63H	ADRH_AVG	
64H	CTRL	
65H	AUTOADCC	
66H	ERRCHKC	
67H	ERRCHKR	
68H		
69H		
6AH		
6BH	SPIC0	
6CH	SPIC1	
6DH	SPID	
6EH	ORMC	
6FH	CTMC0	
70H	CTMC1	
71H	CTMDL	
72H	CTMDH	
73H	CTMAL	
74H	CTMAH	
75H		
76H		
77H	CRCCR	
78H	CRCIN	
79H	CRCDL	
7AH	CRCDH	
7BH	PAS0	
7CH	PAS1	
7DH		
7EH		
7FH		

: Unused, read as 00H

: Reserved, cannot be changed

特殊功能数据存储器结构

特殊功能寄存器

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

间接寻址寄存器 – IAR0, IAR1, IAR2

间接寻址寄存器 IAR0、IAR1 和 IAR2 的地址虽位于数据存储区，但不同于普通寄存器，它们没有实际的物理地址。与定义实际存储器地址的直接存储器寻址不同，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器中能像普通的寄存器一般被操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由存储器指针所指定的地址。MP0、IAR0 用于访问 Sector 0，而 MP1L/MP1H 和 IAR1、MP2L/MP2H 和 IAR2 可根据 MP1H 或 MP2H 寄存器访问所有的 Sector。使用扩展指令可对所有的数据 Sector 进行直接寻址。

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

间接寻址程序举例

范例 1

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

范例 2

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
code .section at 0 'code'
org 00h
start:
    mov a,04h              ; setup size of block
    mov block,a
    mov a,01h              ; setup the memory sector
    mov mp1h,a
    mov a,offset adres1    ; Accumulator loaded with first RAM address
    mov mp1l,a             ; setup memory pointer with first RAM address
loop:
    clr IAR1               ; clear the data at address defined by MP1
    inc mp1l               ; increment memory pointer MP1L
    sdz block              ; check if last memory location has been cleared
    jmp loop
continue:
    :
```

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。

使用扩展指令直接寻址程序举例

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

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

累加器 – ACC

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

程序计数器低字节寄存器 – PCL

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

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。

Option 存储器映射寄存器 – ORMC

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

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

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

- **ORMC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ORMC7	ORMC6	ORMC5	ORMC4	ORMC3	ORMC2	ORMC1	ORMC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **ORMC7~ORMC0：** Option 存储器映射特定数据序列
当将特定数据序列 55H 和 AAH 连续写入该寄存器，会使能 Option 存储器映射功能。需注意，单片机从空闲 / 休眠模式唤醒后，该寄存器的内容将被清除。

状态寄存器 – STATUS

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

除了 PDF 和 TO 标志外，状态寄存器中的位像其它大部分寄存器一样可以被改变。任何数据写入到状态寄存器将不会改变 TO 或 PDF 标志位。另外，执行不同的指令后，与状态寄存器有关的运算可能会得到不同的结果。TO 标志位只会

受系统上电、看门狗溢出或执行“CLR WDT”或“HALT”指令影响。PDF 标志位只会受执行“HALT”或“CLR WDT”指令或系统上电影响。

SC、CZ、Z、OV、AC 和 C 标志位通常反映最近运算的状态。

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

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

- **STATUS 寄存器**

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

“x”：未知

Bit 7 **SC**：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 数据存储器

该单片机内建 EEPROM 数据存储器。由于其非易失的存储结构，即使在电源掉电的情况下存储器内的数据仍然保存完好。这种存储区扩展了存储器空间，对设计者来说增加了许多新的应用机会。EEPROM 可以用来存储产品编号、校准值、用户特定数据、系统配置参数或其它产品信息等。EEPROM 的数据读取和写入过程也会变的更简单。

EEPROM 数据存储器结构

该单片机的 EEPROM 数据存储器容量为 512×8 位。由于映射方式与程序存储器和数据存储器不同，因此不能像其它类型的存储器一样寻址。通过 EEC 控制寄存器中的 MODE 模式选择位，可以确定对 EEPROM 进行字节模式或页模式读写操作。

EEPROM 寄存器

有四个寄存器控制内部 EEPROM 数据存储器总的操作，地址寄存器 EEAL 和 EEAH、数据寄存器 EED 及控制寄存器 EEC。EEAL、EEAH 和 EED 位于 Sector 0 中，它们能像其它特殊功能寄存器一样直接被访问。EEC 位于 Sector 1 中，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	—	—	—	—	—	—	—	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	—	—	—	—	—	—	—	EEAH0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1　未定义，读为“0”

Bit 0　**EEAH0**：数据 EEPROM 地址高字节 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。无论通过 MODE 位选择了何种写模式，在写操作结束后硬件都会自动将 WREN 位清零。

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 5 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 5 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 5 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 5 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。

页擦操作需先将 EEPROM 目标页的起始地址放入 EEAH 和 EEAL 寄存器中，再将任意数据放入 EED 寄存器。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。

注：上述步骤必须依序操作，方能成功完成页擦操作，具体请参考对应的范例程序。

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。字节写操作需先将 EEPROM 目标地址放入 EEAH 和 EEAL 寄存器中，再将要写入的数据放入 EED 寄存器。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。

注：上述步骤必须依序操作，方能成功完成字节写操作，具体请参对应的考范例程序。

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 5 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 5 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。

页写操作需先将 EEPROM 目标页的起始地址放入 EEAH 和 EEAL 寄存器中，再将要写入的数据放入 EED 寄存器。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。

注：上述步骤必须依序操作，方能成功完成页写操作，具体请参考对应的范例程序。

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               ; MP1L 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               ; MP1L 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               ; MP1L 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               ; MP1L 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               ; MP1L 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 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。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。

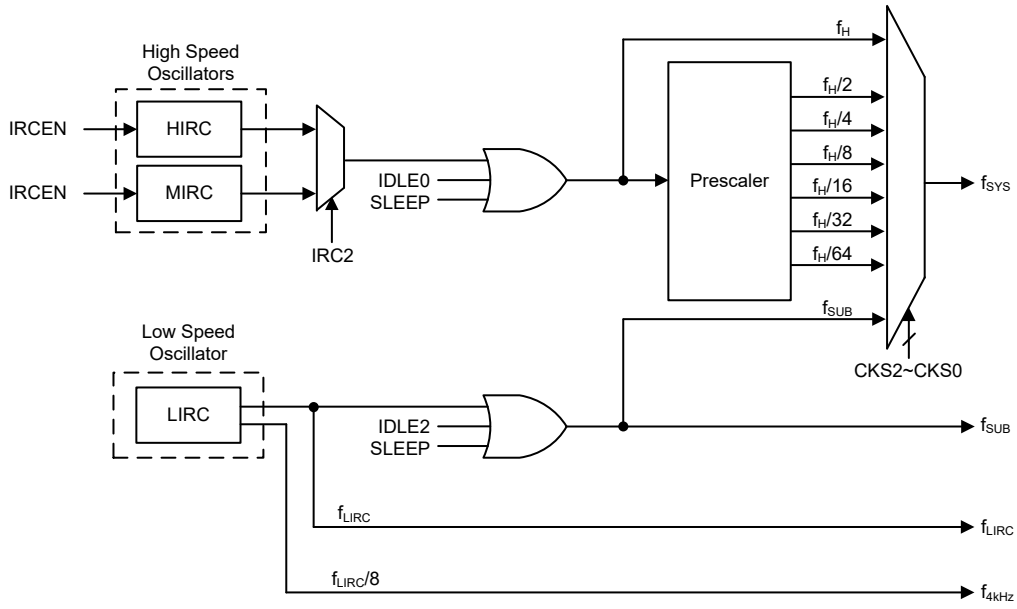
类型	名称	频率
内部高速 RC	HIRC	4MHz
内部中速 RC	MIRC	400kHz
内部低速 RC	LIRC	32.768kHz

振荡器类型

系统时钟配置

该单片机有三个系统振荡器，包括两个高速振荡器和一个低速振荡器。高速振荡器为内部 4MHz 高速振荡器 HIRC 和内部 400kHz 中速振荡器 MIRC，低速振荡器有内部 32.768kHz 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。

高速振荡器的实际时钟源由 IRCC 寄存器中的 IRC2 位选择。低速或高速系统时钟频率由 SCC 寄存器的 CKS2~CKS0 位决定的。请注意，两个振荡器必须做出选择，即一个高速和一个低速振荡器。



系统时钟配置

内部高速 RC 振荡器 – HIRC

内部高速 RC 振荡器是一个集成的系统振荡器，由 IRCC 寄存器中的 IRC2 位选择，无需其它外部器件。内部高速 RC 振荡器频率固定为 4MHz。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。

内部中速 RC 振荡器 – MIRC

内部中速 RC 振荡器是一个集成的系统振荡器，由 IRCC 寄存器中的 IRC2 位选择，无需其它外部器件。内部中速 RC 振荡器频率固定为 400kHz。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 32.768kHz 振荡器 – LIRC

内部 32.768kHz 系统振荡器是一个完全集成 RC 振荡器，典型频率值为 32.768kHz 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。

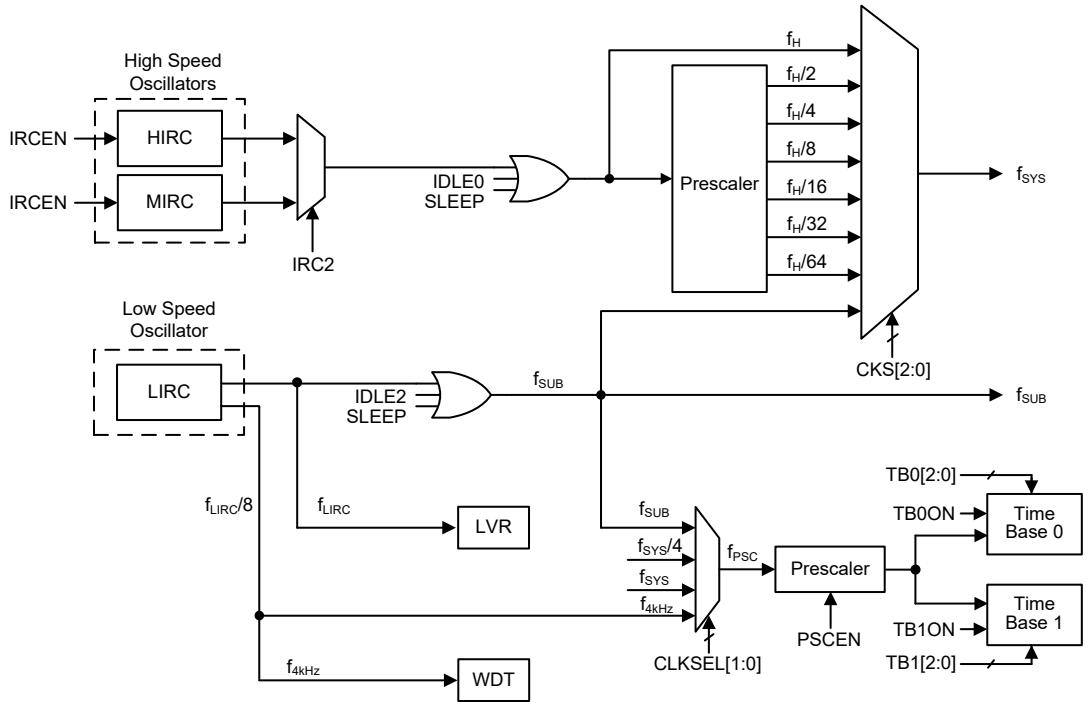
工作模式和系统时钟

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 或 MIRC 振荡器，可通过 IRCC 寄存器的 IRC2 位进行选择。低频系统时钟源来自 f_{SUB}，若被选择 f_{SUB}，该时钟来源于 LIRC 振荡器。其它系统时钟还有高速系统振荡器的分频 $f_H/2$~$f_H/64$。



单片机时钟选项

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~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}	f_{4kHz}
		FHIDEN	FSIDEN	CKS2~CKS0					
快速模式	On	x	x	000~110	f_H~f_H/64	On	On	On	On/Off(2)
低速模式	On	x	x	111	f_{SUB}	On/Off(1)	On	On	On/Off(2)
空闲模式 0	Off	0	1	000~110	Off	Off	On	On	On/Off(2)
				111	On				
空闲模式 1	Off	1	1	xxx	On	On	On	On	On/Off(2)
空闲模式 2	Off	1	0	000~110	On	On	Off	On	On/Off(2)
				111	Off				
休眠模式	Off	0	0	xxx	Off	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. 时基预分频器或 WDT 功能其中一个功能开启时，f_{4kHz} 时钟将开启；当时基预分频器和 WDT 功能均关闭时，f_{4kHz} 时钟将关闭。

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 或 MIRC 振荡器，可通过 IRCC 寄存器的 IRC2 位进行选择。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。若时基预分频器或看门狗定时器功能使能，f_{4kHz} 将继续运行。LIRC 将会开启，提供时钟源给时基预分频器和看门狗功能。

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 IRCC 用于控制系统时钟和相应的振荡器配置。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
IRCC	—	—	—	IRC2	—	—	IRCF	IRCEN

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	1	1	1	—	—	—	0	0

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

Bit 4~2 未定义，读为“0”

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

Bit 0 **FSIDEN**：CPU 关闭时低频振荡器控制位
0：除能
1：使能
此位用来控制在 CPU 执行 HALT 指令关闭后低速振荡器是被激活还是停止。

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。

● IRCC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	IRC2	—	—	IRCF	IRCEN
R/W	—	—	—	R/W	—	—	R	R/W
POR	—	—	—	0	—	—	0	0

Bit 7~5 未定义，读为“0”

Bit 4 **IRC2**：HIRC 或 MIRC 频率选择位
0：4MHz
1：400kHz
当 HIRC 或 MIRC 振荡器使能或通过应用程序改变 HIRC 或 MIRC 频率选择位时，在 IRCF 标志位置高后时钟频率会自动改变。

Bit 3~2 未定义，读为“0”

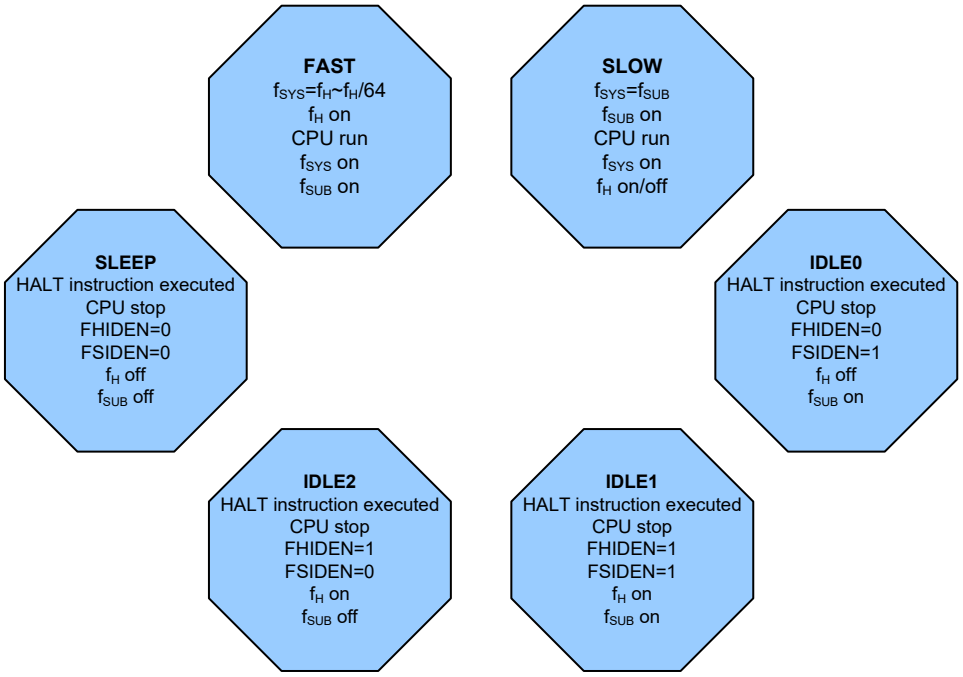
Bit 1 **IRCF**：HIRC 或 MIRC 振荡器稳定标志位
0：未稳定
1：稳定
此位用于表明 HIRC 或 MIRC 振荡器是否稳定。IRCEN 位置高使能 HIRC 或 MIRC 振荡器，或者通过应用程序改变 HIRC 或 MIRC 频率选择位时，IRCF 位会先被清零，待 HIRC 或 MIRC 振荡器稳定后会被置高。

Bit 0 **IRCEN**：HIRC 或 MIRC 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。

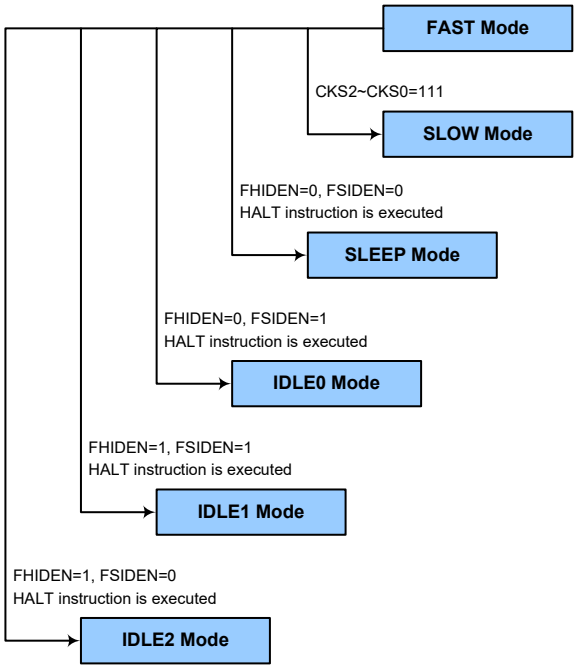
简单来说，快速模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



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。

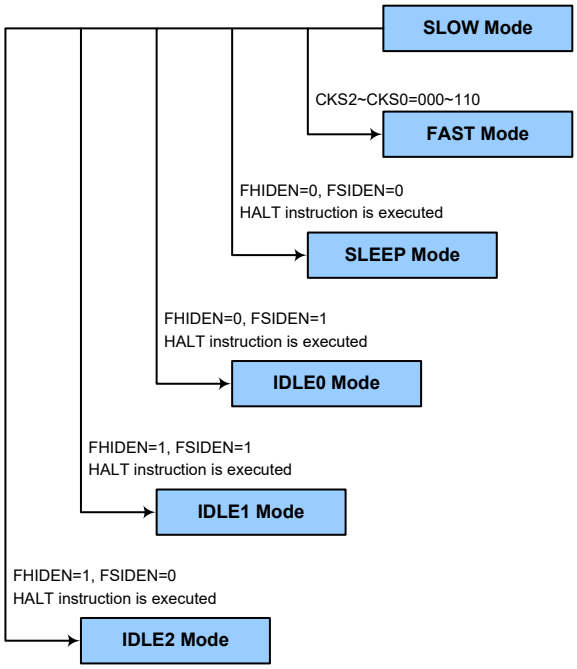
低速模式的时钟源来自 LIRC 振荡器，因此要求所选振荡器在所有模式切换动作发生前稳定下来。



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

然而，如果在低速模式下 f_H 因未使用而关闭，那么从低速模式切换到快速模式时，它需要一定的时间来重新起振和稳定，可通过检测 IRCC 寄存器中的 IRCF 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 或时基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。
- 所有的时钟都关闭，f_{4kHz} 时钟将会开启当 WDT 或时基功能使能。LIRC 振荡器会开启以提供时钟。
- 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。
- 所有的时钟都关闭，f_{4kHz} 时钟将会开启当 WDT 或时基功能使能。LIRC 振荡器会开启以提供时钟。
- 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 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。
- 所有的时钟都关闭，f_{4kHz} 时钟将会开启当 WDT 或时基功能使能。LIRC 振荡器会开启以提供时钟。
- 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 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。
- 所有的时钟都关闭，f_{4kHz} 时钟将会开启当 WDT 或时基功能使能。LIRC 振荡器会开启以提供时钟。
- 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。

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_{4kHz} 由内部低速振荡器 LIRC 提供。内部振荡器 LIRC 的频率大约为 32.768kHz，这个特殊的内部时钟周期会随 V_{DD}、温度和制成的不同而变化。看门狗定时器的时钟源可分频为 2^8~2^{18} 以提供更大的溢出周期，分频比由 WDTC 寄存器中的 WS2~WS0 位来决定。

看门狗定时器控制寄存器

WDTC 寄存器用于选择溢出周期、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	0

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：$2^8/f_{4kHz}$
001：$2^{10}/f_{4kHz}$
010：$2^{12}/f_{4kHz}$
011：$2^{14}/f_{4kHz}$
100：$2^{15}/f_{4kHz}$
101：$2^{16}/f_{4kHz}$
110：$2^{17}/f_{4kHz}$
111：$2^{18}/f_{4kHz}$
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 控制寄存器软件复位标志位
0：未发生
1：发生
当 WDT 控制寄存器软件复位发生时，此位被置为“1”，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”。

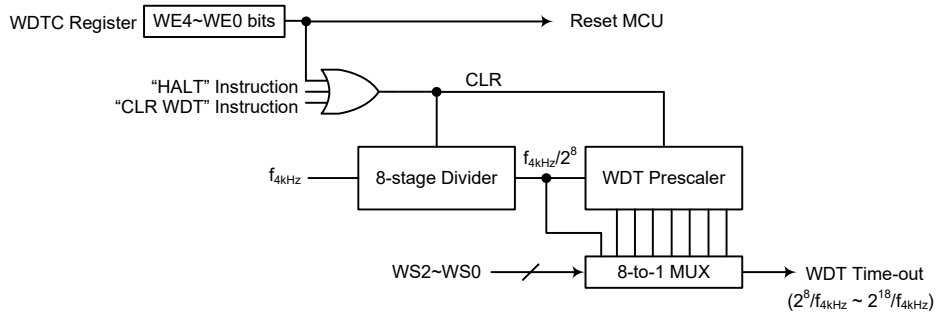
WE4~WE0 位	WDT 功能
10101B	除能
01010B	使能
其它值	单片机复位

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 32.768kHz LIRC 振荡器，分频比为 2^{18} 时最大溢出周期约 64s，分频比为 2^8 时最小溢出周期约 62.5ms。



看门狗定时器

复位和初始化

复位功能是任何单片机中基本的部分，使得单片机可以设定一些与外部参数无关的先置条件。最重要的复位条件是在单片机首次上电以后，经过短暂的延迟，内部硬件电路使得单片机处于预期的稳定状态并开始执行第一条程序指令。上电复位以后，在程序执行之前，部分重要的内部寄存器将会被设定为预先设定的状态。程序计数器就是其中之一，它会被清除为零，使得单片机从最低的程序存储器地址开始执行程序。

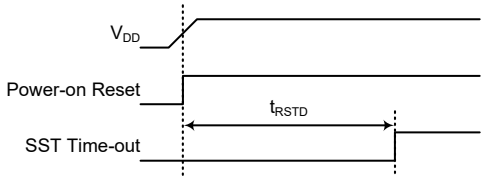
另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。另一种复位为看门狗溢出单片机复位，不同方式的复位操作会对寄存器产生不同的影响。

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。



上电复位时序图

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。

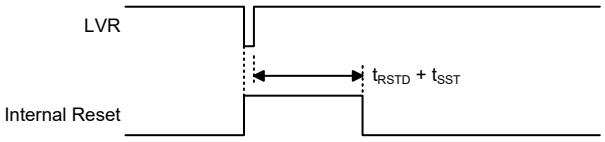
Bit 1　**LRF**：LVR 控制寄存器软件复位标志位
具体描述见低电压复位章节。

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。

LVR 功能可通过 LVRC 寄存器使能或除能。例如在更换电池的情况下，单片机供应的电压可能会在 0.9V~V_{LVR} 之间，这时 LVR 将会自动复位单片机且 RSTFC 寄存器中的 LVRF 标志位置位。所谓有效的 LVR 信号，即在 0.9V~V_{LVR} 的低电压状态的时间，必须超过 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 01100110B。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	1	0	0	1	1	0

Bit 7~0 **LVS7~LVS0**：LVR 电压选择
01100110：1.7V
01010101：1.9V
00110011：2.1V
10011001：2.55V
10101010：3.15V
11110000：LVR 除能
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。
除了上述六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：(7~8)×t_{LIRC}
01：(31~32)×t_{LIRC}
10：(63~64)×t_{LIRC}
11：(127~128)×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。

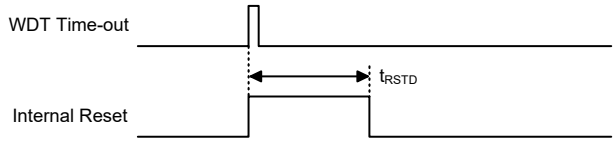
Bit 0 **WRF**：WDT 控制寄存器软件复位标志位
具体描述见看门狗定时器控制寄存器章节。

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

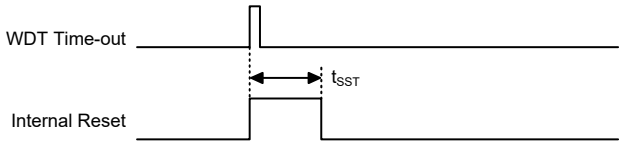
在快速模式或低速模式时看门狗溢出复位，看门狗溢出标志位 TO 将被设为“1”。



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	---x xxxx	---u uuuu	---u uuuu
STATUS	xx00 xxxx	uu1u uuuu	uu11 uuuu
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	111- --00	111- --00	uuu- --uu
IRCC	---0 --00	---0 --00	---u --uu
STKPTR	0--- 0000	0--- 0000	u--- 0000
IECC	0000 0000	0000 0000	uuuu uuuu
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
RSTC	0101 0101	0101 0101	uuuu uuuu
LVRC	0110 0110	0110 0110	uuuu uuuu
TLVRC	---- --01	---- --01	---- --uu
MFI0	--00 --00	--00 --00	--uu --uu
MFI1	-000 -000	-000 -000	-uuu -uuu
WDTC	0101 0010	0101 0010	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	--00 --00	--00 --00	--uu --uu
PB	---- ---1	---- ---1	---- ---u
PBC	---- ---1	---- ---1	---- ---u
PBPU	---- ---0	---- ---0	---- ---u
PCRL	0000 0000	0000 0000	uuuu uuuu
PCRH	---0 0000	---0 0000	---u uuuu
PSCR	---- -000	---- -000	---- -uuu
TB0C	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	u--- -uuu
USR	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
UCR2	0000 0000	0000 0000	uuuu uuuu
UCR3	---- ---0	---- ---0	---- ---u
BRDH	0000 0000	0000 0000	uuuu uuuu
BRDL	0000 0000	0000 0000	uuuu uuuu
UFCR	--00 0000	--00 0000	--uu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
RxCNT	---- -000	---- -000	---- -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	---- ---0	---- ---0	---- ---u
EED	0000 0000	0000 0000	uuuu uuuu
PWRC	---- 0---	---- 0---	---- u---
IREFC	---0 -000	---0 -000	---u -uuu
PVREF	0000 0000	0000 0000	uuuu uuuu
OPAC	000- ----	000- ----	uuu- ----
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
PGAC	-000 0000	-000 0000	-uuu uuuu
PGACS	0000 0000	0000 0000	uuuu uuuu
MODC	---- 0000	---- 0000	---- uuuu
ADRL	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRM	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR0	---0 000-	---0 000-	---u uuu-
ADCR1	0000 -00-	0000 -00-	uuuu -uu-
ADCS	---0 0000	---0 0000	---u uuuu
SINC3	1001 0011	1001 0011	uuuu uuuu
ADACCTRL	00-- -000	00-- -000	uu-- -uuu
ADRL_AVG	xxxx xxxx	xxxx xxxx	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
ADRM_AVG	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH_AVG	xxxx xxxx	xxxx xxxx	uuuu uuuu
CTRL	000- ----	000- ----	uuu- ----
AUTOADCC	0-00 0000	0-00 0000	u-uu uuuu
ERRCHKC	00-- ----	00-- ----	uu-- ----
ERRCHKR	0000 0000	0000 0000	uuuu uuuu
SPIC0	111- --00	111- --00	uuu- --uu
SPIC1	--00 0000	--00 0000	--uu uuuu
SPID	xxxx xxxx	xxxx xxxx	uuuu uuuu
ORMC	0000 0000	0000 0000	0000 0000
CTMC0	0000 0000	0000 0000	uuuu uuuu
CTMC1	0000 0000	0000 0000	uuuu uuuu
CTMDL	0000 0000	0000 0000	uuuu uuuu
CTMDH	---- --00	---- --00	---- --uu
CTMAL	0000 0000	0000 0000	uuuu uuuu
CTMAH	---- --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
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	---- --00	---- --00	---- --uu
FC0	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	uuuu uuuu
FC2	---- --00	---- --00	---- --uu
FARL	0000 0000	0000 0000	uuuu uuuu
FARH	---0 0000	---0 0000	---u 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	---- -000	---- -000	---- -uuu
EEC	0000 0000	0000 0000	uuuu uuuu

注：“u”表示不改变
“x”表示未知
“-”表示未定义

输入 / 输出端口

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

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	—	—	—	—	—	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，如果对输出口做读取动作时，程序读取到的是内部输出数据锁存器中的状态，而不是输出引脚上实际的逻辑状态。

- **PxC 寄存器**

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

PxCn：I/O Px 口类型选择位
0：输出
1：输入
PxCn 位用于控制引脚类型。这里的 x 可以是端口 A 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 PxSn，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、xTCK、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	—	—	—	—	—	RXTXPS	SPISDIPS	SPISCKPS
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	—	—	—	—	—	RXTXPS	SPISDIPS	SPISCKPS
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3　未定义，读为“0”

Bit 2　**RXTXPS**：RX/TX 输入源引脚选择
0：PA7
1：PA2

Bit 1　**SPISDIPS**：SPISDI 输入源引脚选择
0：PA7
1：PA2

Bit 0　**SPISCKPS**：SPISCK 输入源引脚选择
0：PB0
1：PA4

- **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：SPISDI
10：RX/TX
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：SPISDO
10：TX
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：SPISDI
10：RX/TX
11：PA7/INT2/PTPI

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

Bit 3~2 **PAS13~PAS12**：PA5 引脚共用功能选择
00：PA5/INT1/PTCK
01：PA5/INT1/PTCK
10：PA5/INT1/PTCK
11：$\overline{\text{SPISCS}}$

Bit 1~0 **PAS11~PAS10**：PA4 引脚共用功能选择
00：PA4/INT0/CTCK
01：CTP
10：SPISCK
11：PA4/INT0/CTCK

● **PBS0 寄存器**

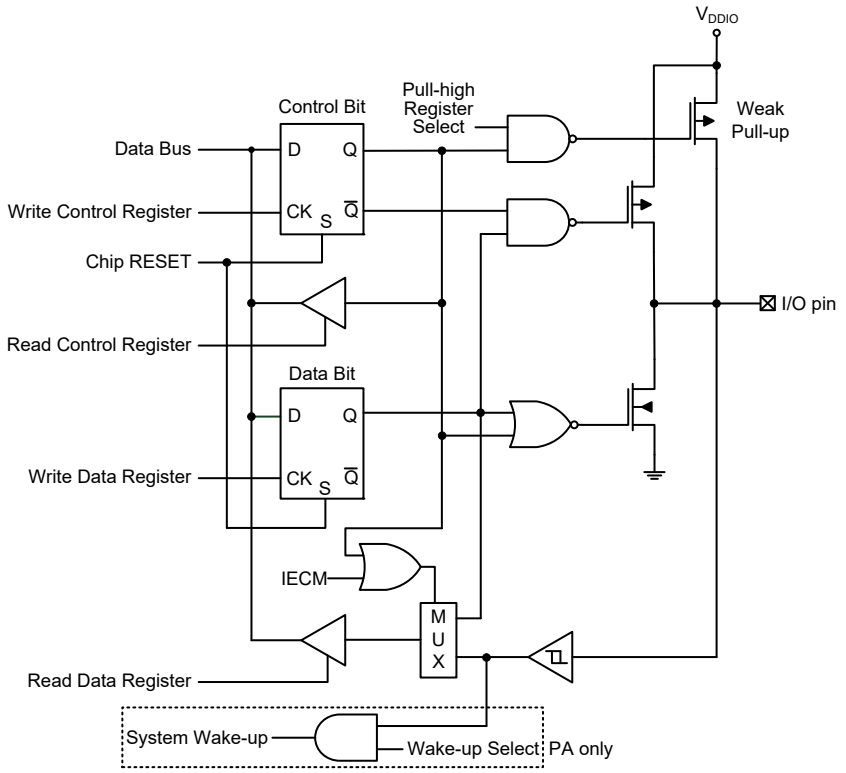
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PBS01	PBS00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

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：SPISCK
10：PTP
11：PB0/INT3

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 I/O 引脚的数据。该功能专为 I/O 功能和 A/D 通道上的 IEC 60730 自检测试而设计。I/O 功能和 A/D 通道自检完成后，必须立刻关闭读端口功能。在 I/O 功能和 A/D 通道自检之外的其他情况下，强烈建议除能读端口功能，以免影响其他外设功能，造成不可预期的后果。

寄存器 IECC 用于控制读端口功能。若向寄存器 IECC 写入一个特定的数据模式“11001010”，内部信号 IECM 将被置高以使能读端口功能。读端口功能使能后，数据读取路径来自 I/O 引脚。执行读端口指令“mov a, 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。若向 IECC 寄存器写入除 11001010 以外的其它值，内部信号 IECM 将被清零，除能读端口功能且数据读取路径来自数据锁存器或 I/O 引脚。若此功能除能，引脚将作为所选中的共用引脚功能进行正常操作。

- **IECC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	IECS7	IECS6	IECS5	IECS4	IECS3	IECS2	IECS1	IECS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

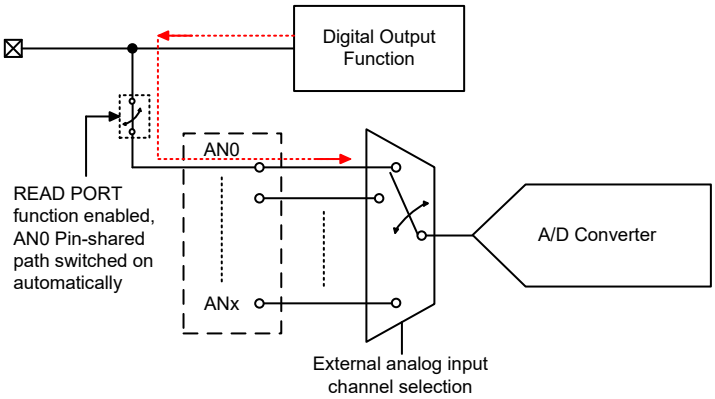
Bit 7~0 **IECS7~IECS0**：读端口功能使能控制 bit 7 ~ bit 0
11001010：IECM=1 – 读端口功能使能
其它值：IECM=0 – 读端口功能除能

读端口功能	除能		使能	
端口控制寄存器位 – PxC.n	1	0	1	0
I/O 功能	引脚值	数据锁存值	引脚值	
A/D 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，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。如下图举例示意，无论 AN0 是否选择作为模拟输入引脚使用，只要读端口功能使能，AN0 模拟输入通道都将开启。通过这种方式，AN0 模拟输入路径可与其共用引脚上的数字输出内部连接，然后在无外接模拟输入电压的情况下对相应的数字数据进行转换，从而实现 AN0 模拟输入通道检测。

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



A/D 通道输入路径内部连接

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 2 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 xTM 控制寄存器的 xTCK2~xTCK0 位，选择所需的时钟源，其中 x 代表 C 或 P 型 TM。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 xTCK 引脚。xTCK 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 xTCK，而周期型 TM 还有一个输入引脚 PTPI。xTM 输入引脚 xTCK 作为 xTM 时钟源输入脚，通过设置 xTMC0 寄存器中的 xTCK2~xTCK0 位进行选择。外部时钟源可通过该引脚来驱动内部

TM。xTCK 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。

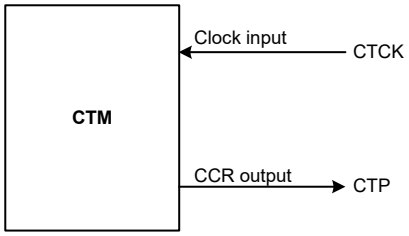
另一种 PTM 输入引脚 PTPI 作为捕捉输入脚，其有效边沿有上升沿、下降沿和双沿，通过设置 PTMC1 寄存器中的 PTIO1~PTIO0 位来选择有效边沿类型。除了 PTPI 引脚外，PTCK 引脚也可以用作捕捉输入模式的外部触发引脚。

每个 TM 都有一个输出引脚 xTP。当 TM 工作在比较匹配输出模式且比较匹配发生时，该引脚会由 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。

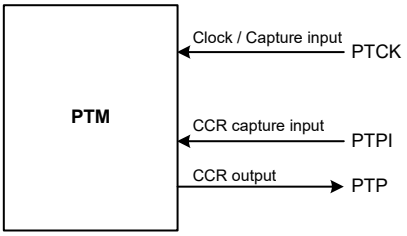
当 TM 输入和输出引脚与其它功能共用时，TM 输入和输出功能需要事先通过相关引脚共用功能选择寄存器先被设置。更多引脚共用功能选择详见引脚共用功能章节。

CTM		PTM	
输入	输出	输入	输出
CTCK	CTP	PTCK, PTPI	PTP

TM 外部引脚



CTM 功能引脚框图

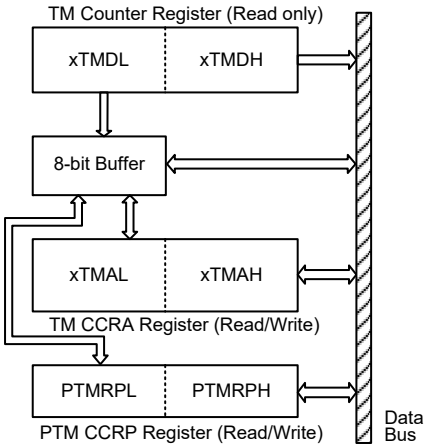


PTM 功能引脚框图

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 xTMAL 和 PTMRPL，否则可能导致无法预期的结果。

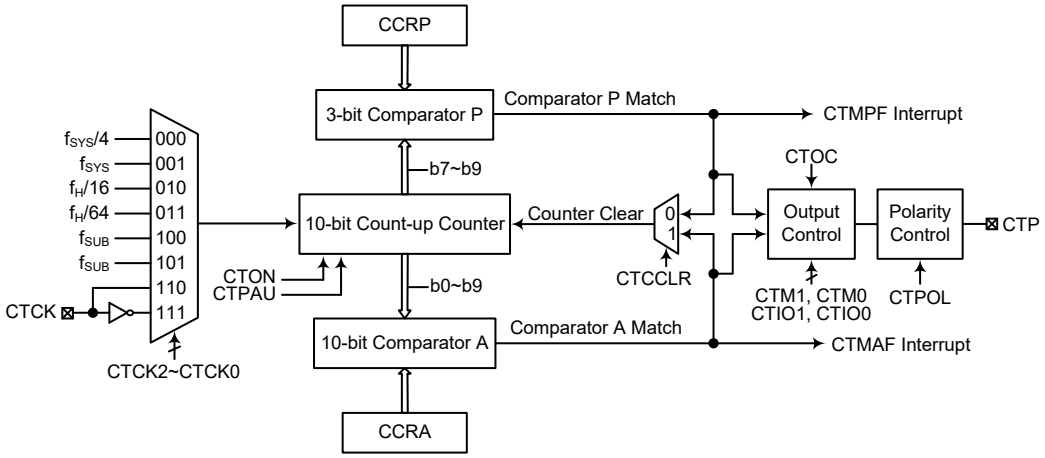


读写流程如下步骤所示：

- 写数据至 CCRA 或 CCRP
  - ♦ 步骤 1. 写数据至低字节寄存器 xTMAL 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 xTMAH 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 xTMDH、xTMAH 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 xTMDL、xTMAL 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：CTM 外部引脚与其他功能共用引脚，因此在使用 CTM 功能前，应该合理设置相关引脚共用功能选择寄存器以选择所需的 CTM 引脚功能。对于 CTCK 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

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 CTON 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 CTM 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
CTMC0	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
CTMC1	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
CTMDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMDH	—	—	—	—	—	—	D9	D8
CTMAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMAH	—	—	—	—	—	—	D9	D8

10-bit 标准型 TM 寄存器列表

● **CTMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
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 **CTPAU**：CTM 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，CTM 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 **CTCK2~CTCK0**：CTM 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：CTCK 上升沿时钟
111：CTCK 下降沿时钟
此三位用于选择 CTM 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 **CTON**：CTM 计数器 On/Off 控制位
0：Off
1：On
此位控制 CTM 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 CTM。清零此位将停止计数器并关闭 CTM 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 CTM 处于比较匹配输出模式或 PWM 输出模式，当 CTON 位经由低到高的转换时，CTM 输出脚将复位至 CTOC 位指定的初始值。

Bit 2~0 **CTRP2~CTRP0**：CTM CCRP 3-bit 寄存器，与 CTM 计数器 bit 9 ~ bit 7 比较
比较器 P 匹配周期 =
000：1024 个 CTM 时钟周期
001：128 个 CTM 时钟周期
010：256 个 CTM 时钟周期
011：384 个 CTM 时钟周期
100：512 个 CTM 时钟周期
101：640 个 CTM 时钟周期
110：768 个 CTM 时钟周期
111：896 个 CTM 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 CTCCLR 位设定为 0 时，此比较结果可用于清零内部计数器。CTCCLR 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。

● **CTMC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
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 **CTM1~CTM0**：CTM 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 CTM 需要的工作模式。为了确保操作可靠，CTM 应在 CTM1 和 CTM0 位有任何改变前先关掉。在定时 / 计数器模式，CTM 输出脚状态未定义。

Bit 5~4 **CTIO1~CTIO0**：CTM 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 CTM 外部引脚如何改变状态。这两位值的选择取决于 CTM 运行在哪种模式下。
在比较匹配输出模式下，CTIO1 和 CTIO0 位决定当从比较器 A 比较匹配输出发生时 CTM 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 CTM 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。CTM 输出脚的初始值通过 CTMC1 寄存器的 CTOC 位设置取得。注意，由 CTIO1 和 CTIO0 位得到的输出电平必须与通过 CTOC 位设置的初始值不同，否则当比较匹配发生时，CTM 输出脚将不会发生变化。在 CTM 输出脚改变状态后，通过 CTON 位由低到高电平的转换复位至初始值。
在 PWM 输出模式，CTIO1 和 CTIO0 决定比较匹配条件发生时怎样改变 CTM 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 CTM 关闭后才能改变 CTIO1 和 CTIO0 位的值。若在 CTM 运行时改变 CTIO1 和 CTIO0 的值，PWM 输出的值将无法预料。

Bit 3 **CTOC**：CTP 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 CTM 输出脚输出控制位。它取决于 CTM 此时正运行于比较匹配输出模式还是 PWM 输出模式。若 CTM 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 CTM 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 **CTPOL**：CTP 输出极性控制位
0：同相
1：反相
此位控制 CTP 输出脚的极性。此位为高时 CTM 输出脚反相，为低时 CTM 输出脚同相。若 CTM 处于定时 / 计数器模式时其无效。

Bit 1 **CTDPX**：CTM 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 **CTCCLR**：CTM 计数器清零条件选择位
0：CTM 比较器 P 匹配
1：CTM 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。CTCCLR 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。CTCCLR 位在 PWM 输出模式时未使用。

● CTMDL 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**：CTM 计数器低字节寄存器 bit 7 ~ bit 0
CTM 10-bit 计数器 bit 7 ~ bit 0

● CTMDH 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**：CTM 计数器高字节寄存器 bit 1 ~ bit 0
CTM 10-bit 计数器 bit 9 ~ bit 8

● CTMAL 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**：CTM CCRA 低字节寄存器 bit 7 ~ bit 0
CTM 10-bit CCRA bit 7 ~ bit 0

● CTMAH 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**：CTM CCRA 高字节寄存器 bit 1 ~ bit 0
CTM 10-bit CCRA bit 9 ~ bit 8

简易型 TM 工作模式

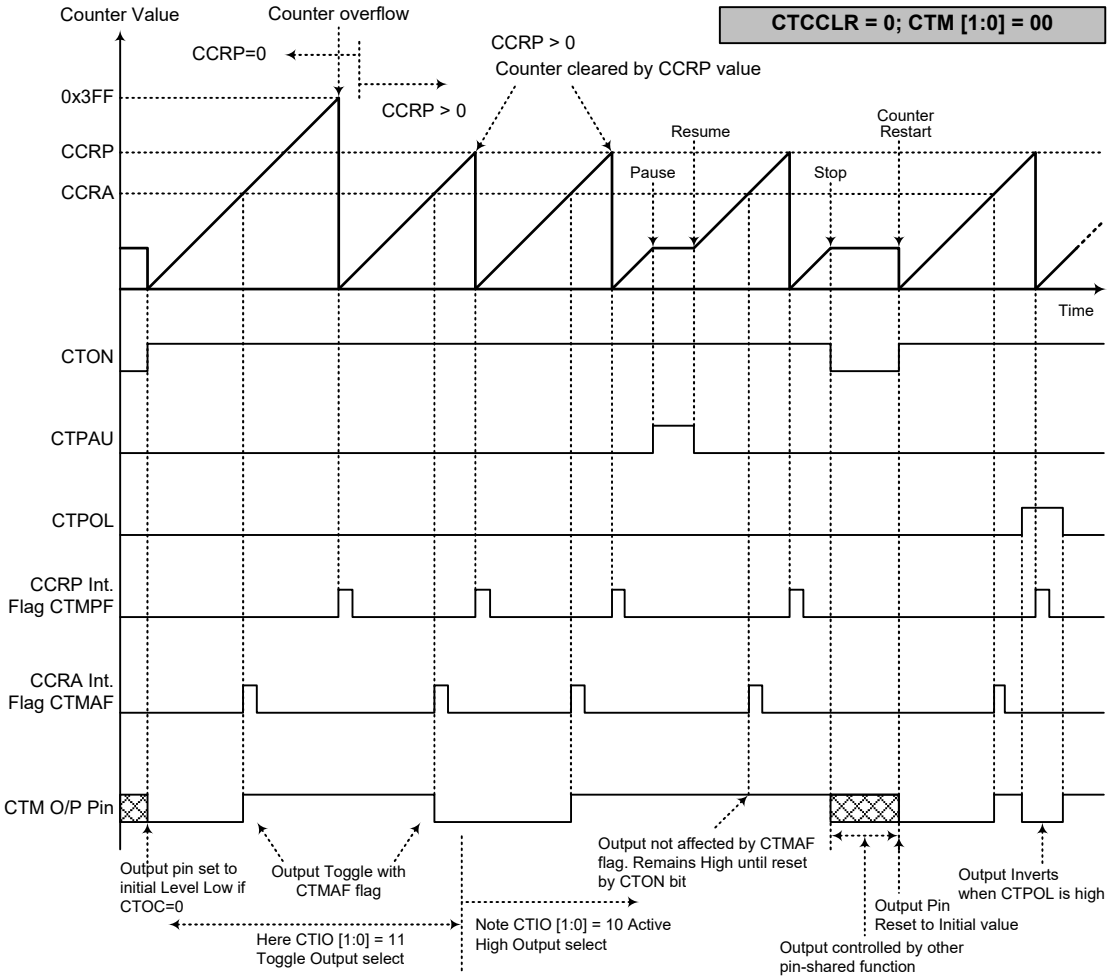
简易型 TM 有三种工作模式，即比较匹配输出模式，PWM 输出模式或定时 / 计数器模式。通过设置 CTMC1 寄存器的 CTM1 和 CTM0 位选择任意模式。

比较匹配输出模式

为使 CTM 工作在此模式，CTMC1 寄存器中的 CTM1 和 CTM0 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 CTCCLR 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 CTMAF 和 CTMPF 将分别置起。

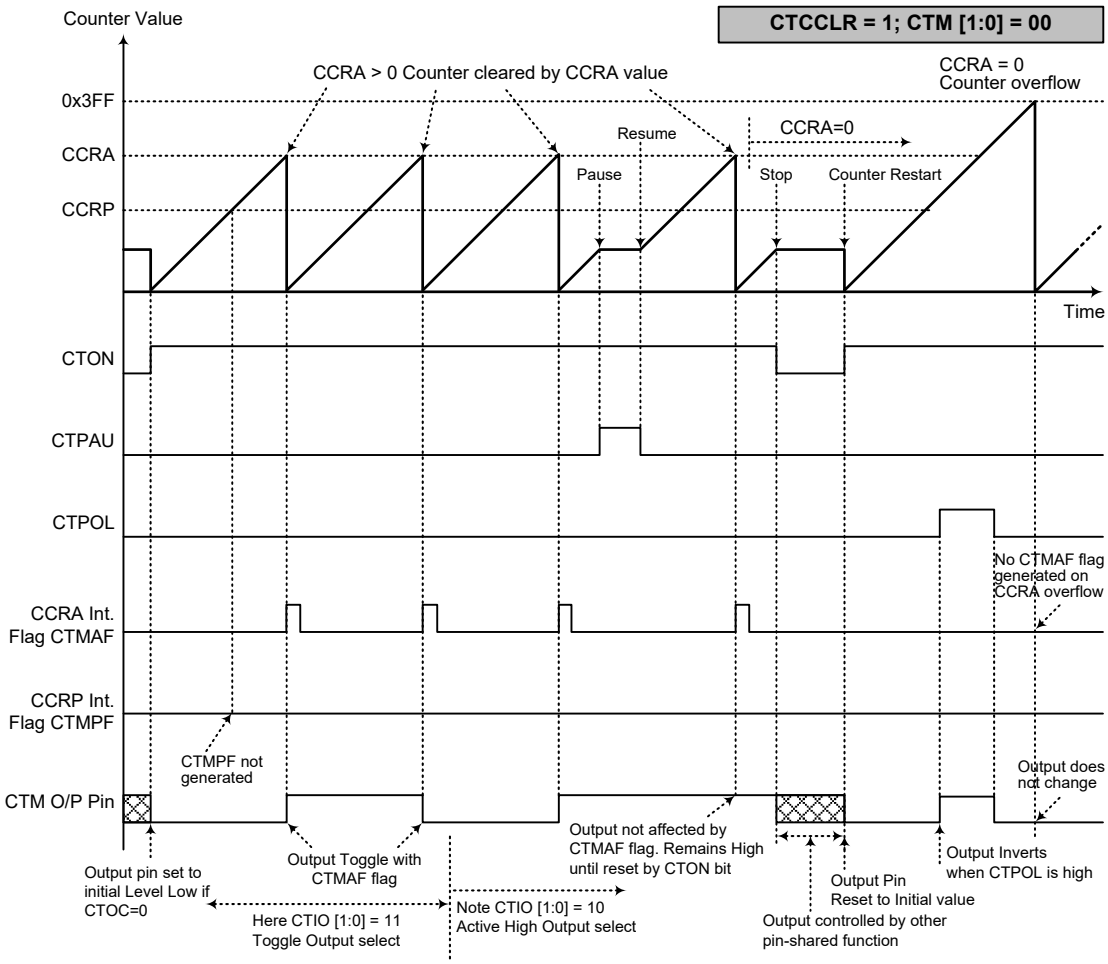
如果 CTMC1 寄存器的 CTCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 CTMAF 中断请求标志产生。所以当 CTCCLR 为高时，不产生 CTMPF 中断请求标志。如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 CTMAF 请求标志。

正如该模式名所言，当比较匹配发生后，CTM 输出脚状态改变。当比较器 A 比较匹配发生后 CTMAF 标志产生时，CTM 输出脚状态改变。比较器 P 比较匹配发生时产生的 CTMPF 标志不影响 CTM 输出脚。CTM 输出脚状态改变方式由 CTMC1 寄存器中 CTIO1 和 CTIO0 位决定。当比较器 A 比较匹配发生时，CTIO1 和 CTIO0 位决定 CTM 输出脚输出高，低或翻转当前状态。在 CTON 位由低到高电平的变化后，CTM 输出脚初始状态为 CTOC 位所指定的电平。注意，若 CTIO1 和 CTIO0 位同时为 0 时，引脚输出不变。



比较匹配模式 – CTCCLR=0

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



比较匹配模式 – CTCCLR=1

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

定时 / 计数器模式

为使 CTM 工作在此模式，CTMC1 寄存器中的 CTM1 和 CTM0 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 CTM 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 CTM 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 CTM 工作在此模式，CTMC1 寄存器中的 CTM1 和 CTM0 位需要设置为“10”。CTM 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 CTM 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，CTCCLR 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 CTMC1 寄存器的 CTDPX 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。CTMC1 寄存器中的 CTOC 位决定 PWM 波形的极性，CTIO1 和 CTIO0 位使能 PWM 输出或将 CTM 输出脚置为逻辑高或逻辑低。CTPOL 位对 PWM 输出波形的极性取反。

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

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

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

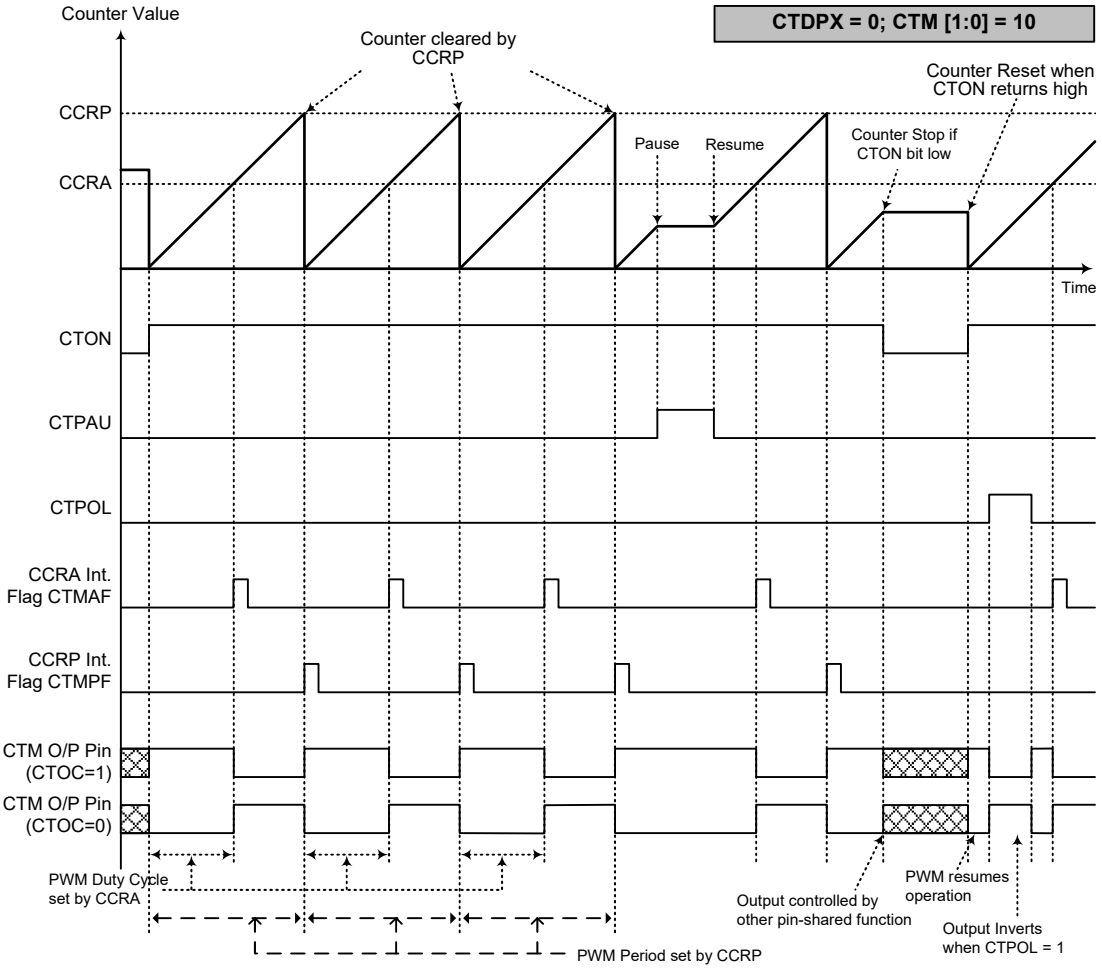
CTM PWM 输出频率 = $(f_{SYS}/4)/(4\times128)=f_{SYS}/2048$=1.9531kHz，duty=128/(4×128)= 25%。

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

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

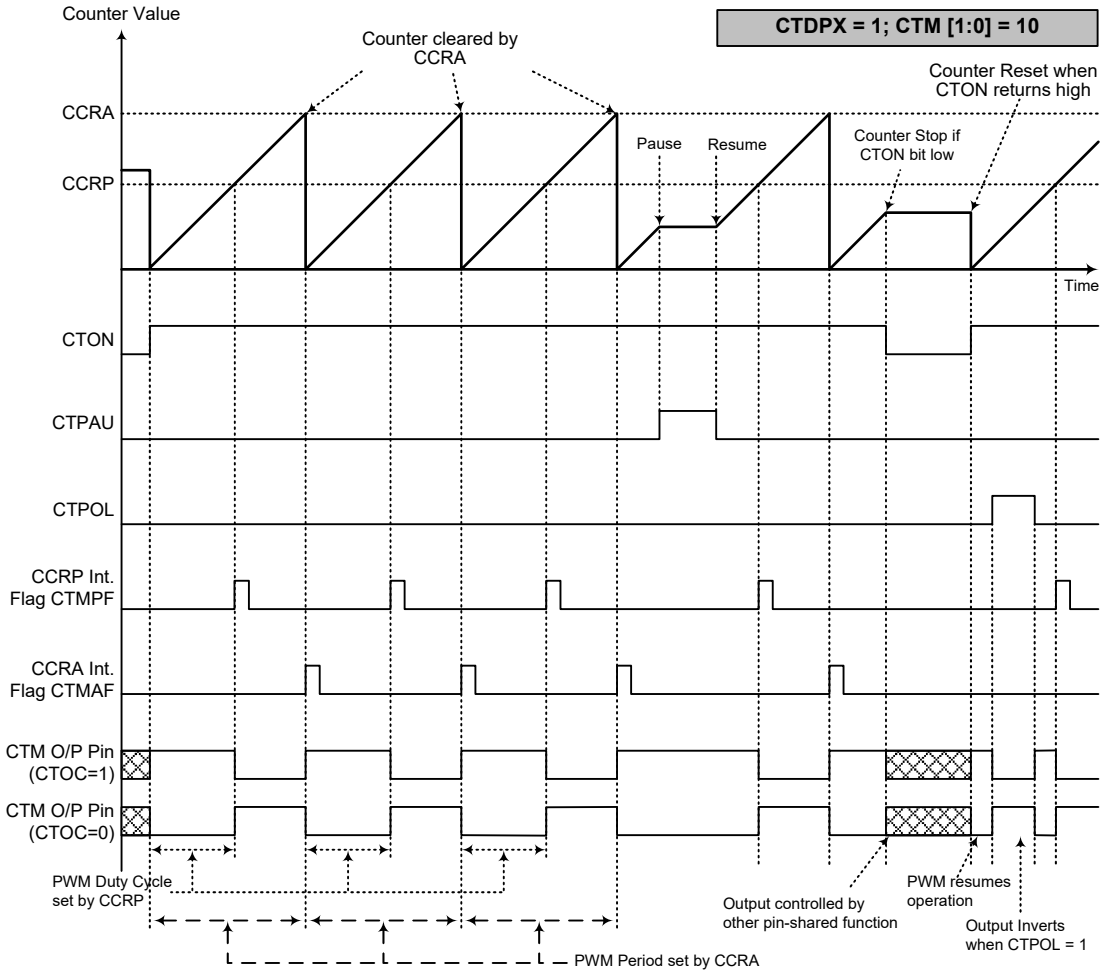
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

PWM 的输出周期由 CCRA 寄存器的值与 CTM 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 – CTDPX=0

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

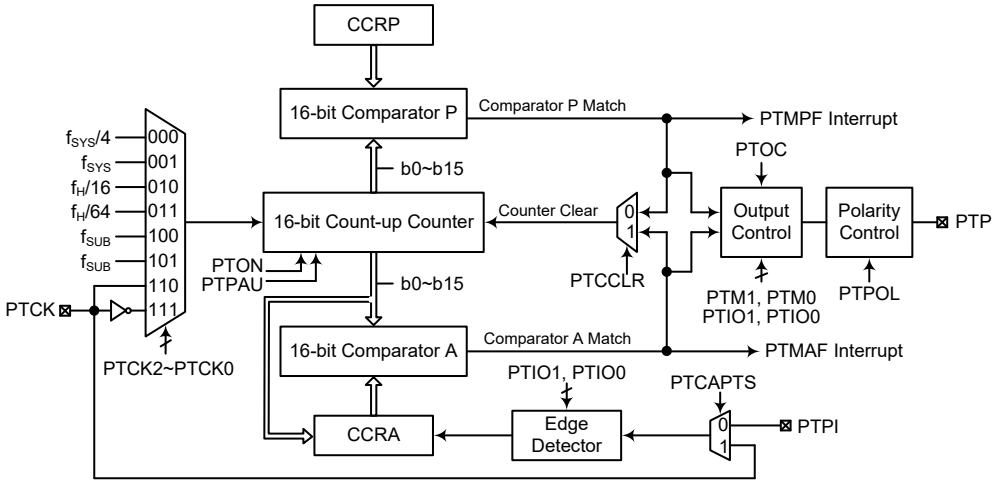


PWM 输出模式 – CTDPX=1

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

周期型 TM – PTM

周期型 TM 包括 5 种工作模式，即比较匹配输出，定时 / 事件计数器，捕捉输入，单脉冲输出和 PWM 输出模式。周期型 TM 由两个外部输入脚控制并驱动一个外部输出脚。



注：PTM 外部引脚与其他功能共用引脚，因此在使用 PTM 功能前，应合理设置相关引脚共用功能选择寄存器以选择所需的 PTM 引脚功能。对于 PTCK 和 PTPI 输入引脚还需设置相应的端口控制寄存器，将该引脚设置为输入口。

16-bit 周期型 TM 框图

周期型 TM 操作

周期型 TM 核心是一个由用户选择的内部或外部时钟源驱动的 16 位向上计数器，它还包括两个内部比较器即比较器 A 和比较器 P。这两个比较器将计数器的值与 CCRP 和 CCRA 寄存器中的值进行比较。CCRP 和 CCRA 是 16 位的，与计数器的所有位比较。

通过应用程序改变 16 位计数器值的唯一方法是使 PTON 位发生上升沿跳变清除计数器。此外，计数器溢出或比较匹配也会自动清除计数器。上述条件发生时，通常情况会产生 PTM 中断信号。周期型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，也可以控制输出脚。所有工作模式的设定都是通过设置相关寄存器来实现的。

周期型 TM 寄存器介绍

周期型 TM 的所有工作由一系列寄存器控制。一对只读寄存器用来存放 16 位计数器的值，两对读 / 写寄存器存放 16 位 CCRA 和 CCRP 的值。剩下两个控制寄存器设置不同的操作和控制模式。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	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

寄存器名称	位							
	7	6	5	4	3	2	1	0
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	PTCAPTS	PTCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PTM1~PTM0**：PTM 工作模式选择位
00：比较匹配输出模式
01：捕捉输入模式
10：PWM 输出模式或单脉冲输出模式
11：定时 / 计数器模式

这两位设置 PTM 需要的工作模式。为了确保操作可靠，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 **PTCAPTS**：PTM 捕捉触发源选择位
- 0：来自 PTPI 引脚
- 1：来自 PTCK 引脚

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。

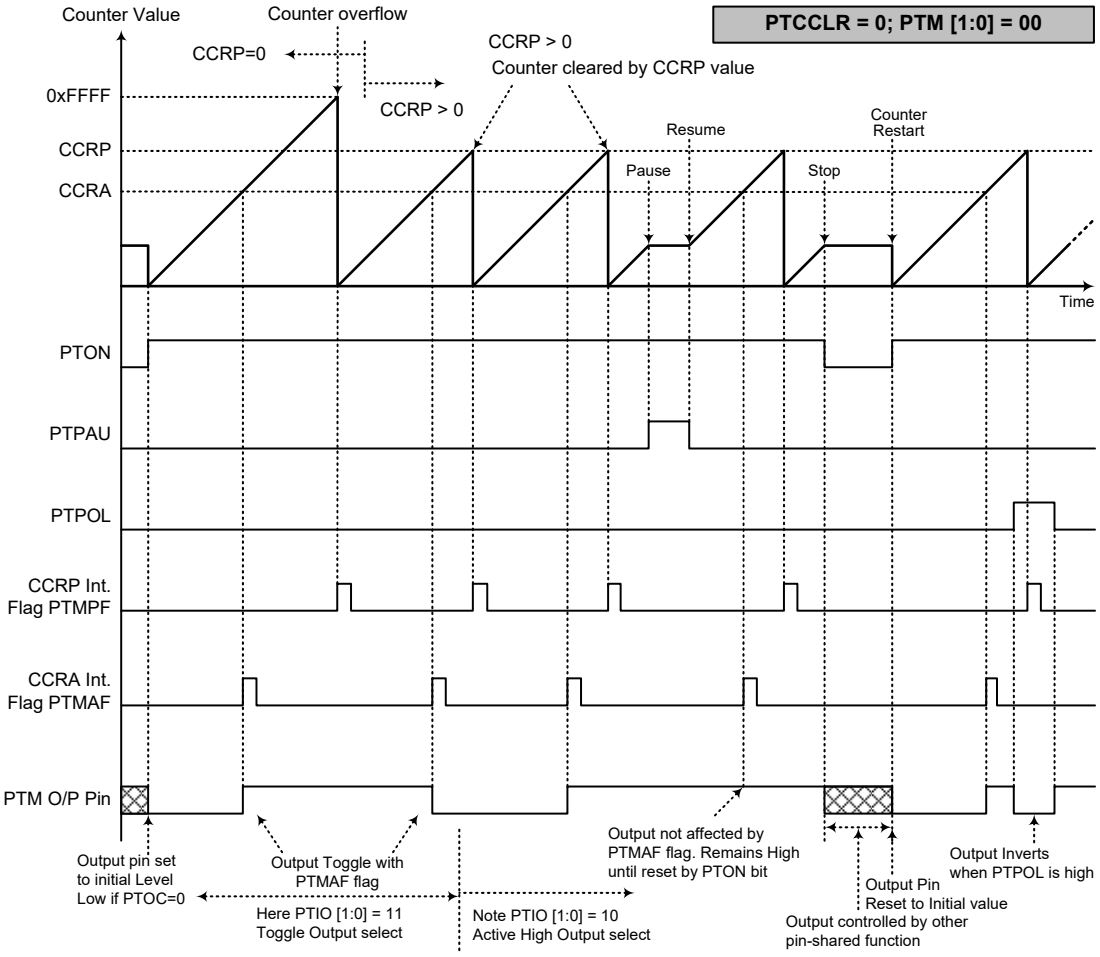
比较匹配输出模式

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”。

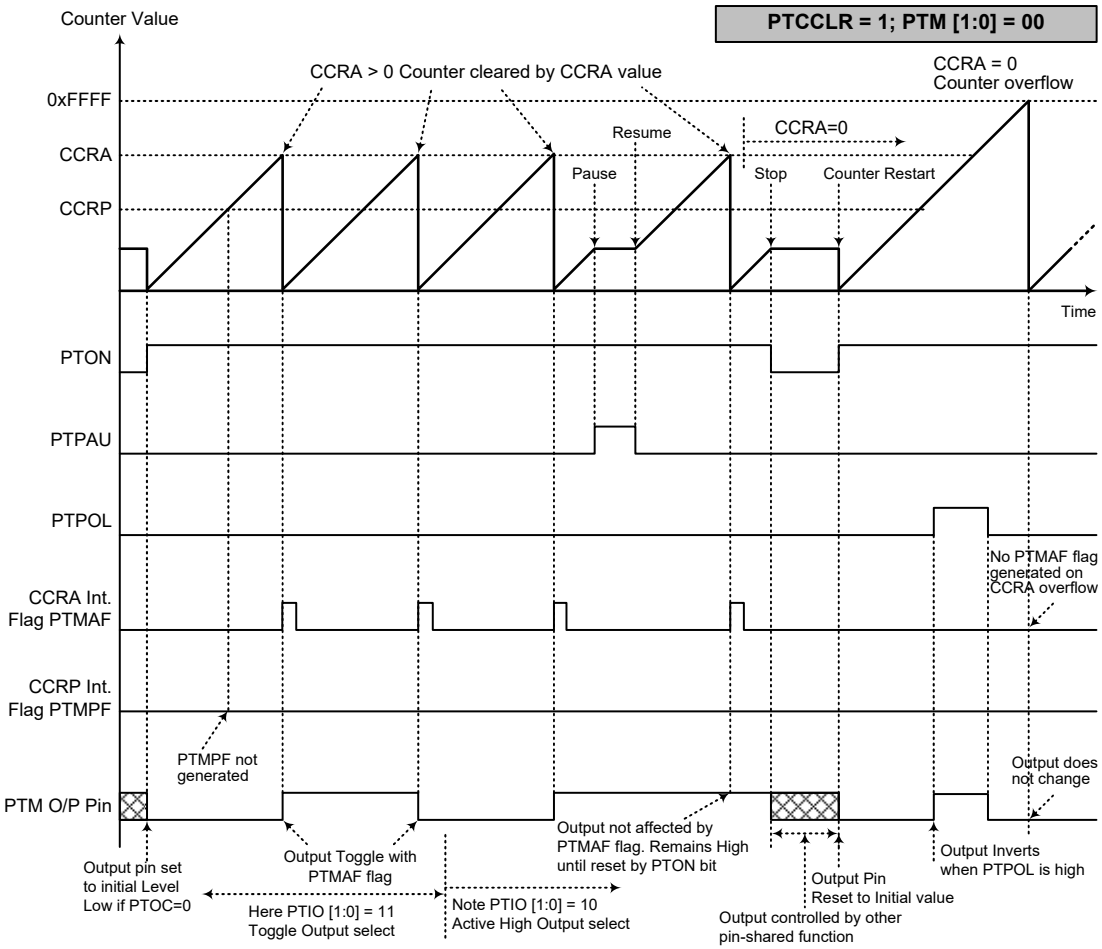
如果 CCRA 位都清除为零，当计数器的值达到 16 位最大值 FFFFH 时将溢出，但此时不会产生 PTMAF 中断请求标志。

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”。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 寄存器都用于控制 PWM 方波。CCRP 寄存器通过清除内部计数从而控制 PWM 周期，CCRA 寄存器设置 PWM 的占空比。PWM 波形的周期和占空比由 CCRP 和 CCRA 寄存器的值控制。

当比较器 A 或比较器 P 比较匹配发生时，将产生 CCRA 或 CCRP 中断标志。PTMC1 寄存器中的 PTOC 位选择 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。

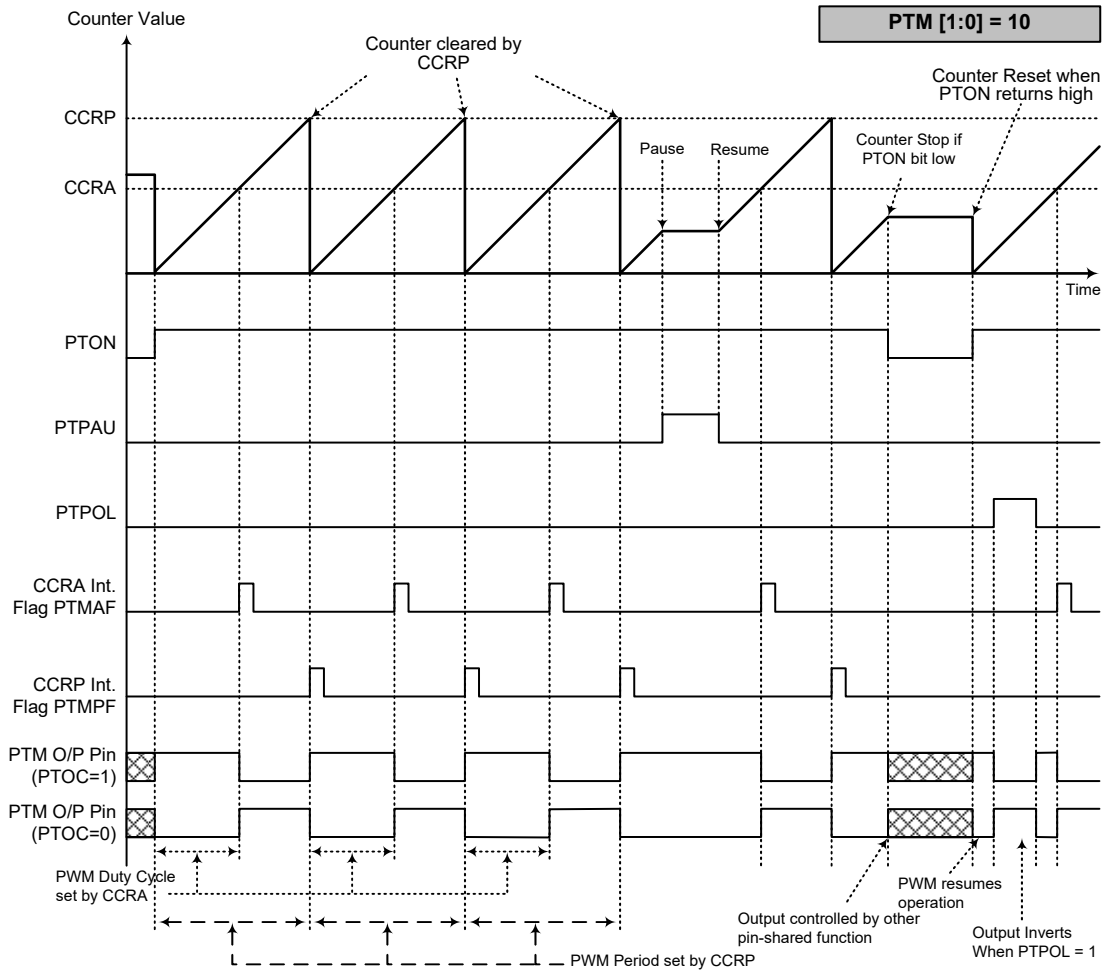
- **16-bit PTM，PWM 输出模式，边沿对齐模式**

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

若 f_{SYS}=4MHz，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=1.9531kHz，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

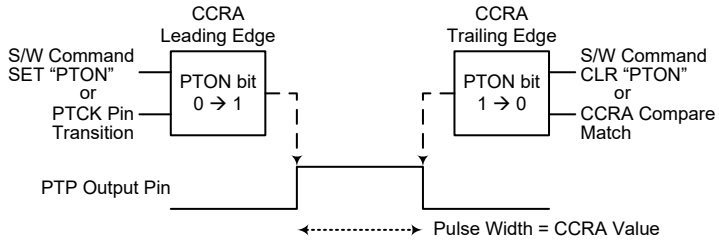
4. PTCCLR 位对 PWM 功能无影响

单脉冲输出模式

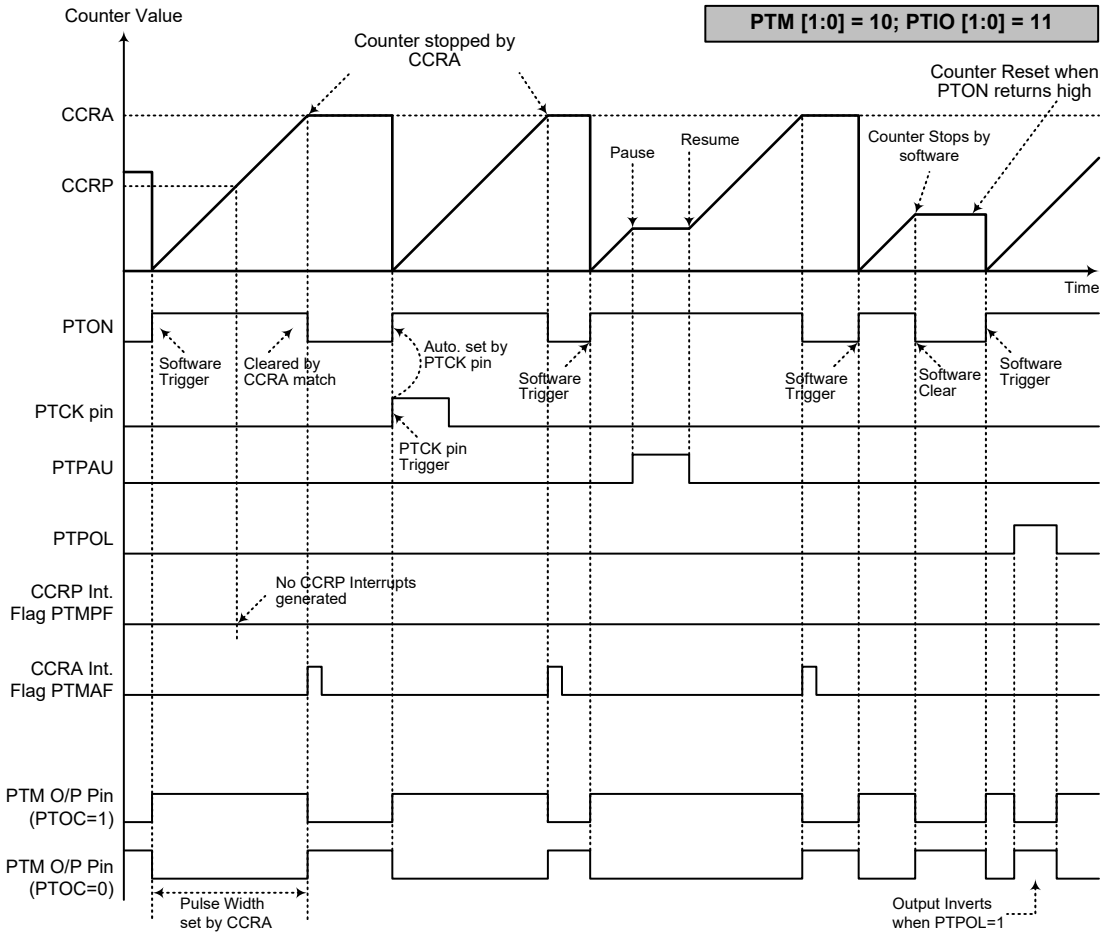
为使 PTM 工作在此模式，PTMC1 寄存器中的 PTM1 和 PTM0 位需要设置为“10”，同时 PTIO1 和 PTIO0 位需要设置为“11”。正如模式名所言，单脉冲输出模式，在 PTM 输出脚将产生一个脉冲输出。

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 寄存器和 PTCCLR 位未使用。



单脉冲产生示意图



单脉冲输出模式

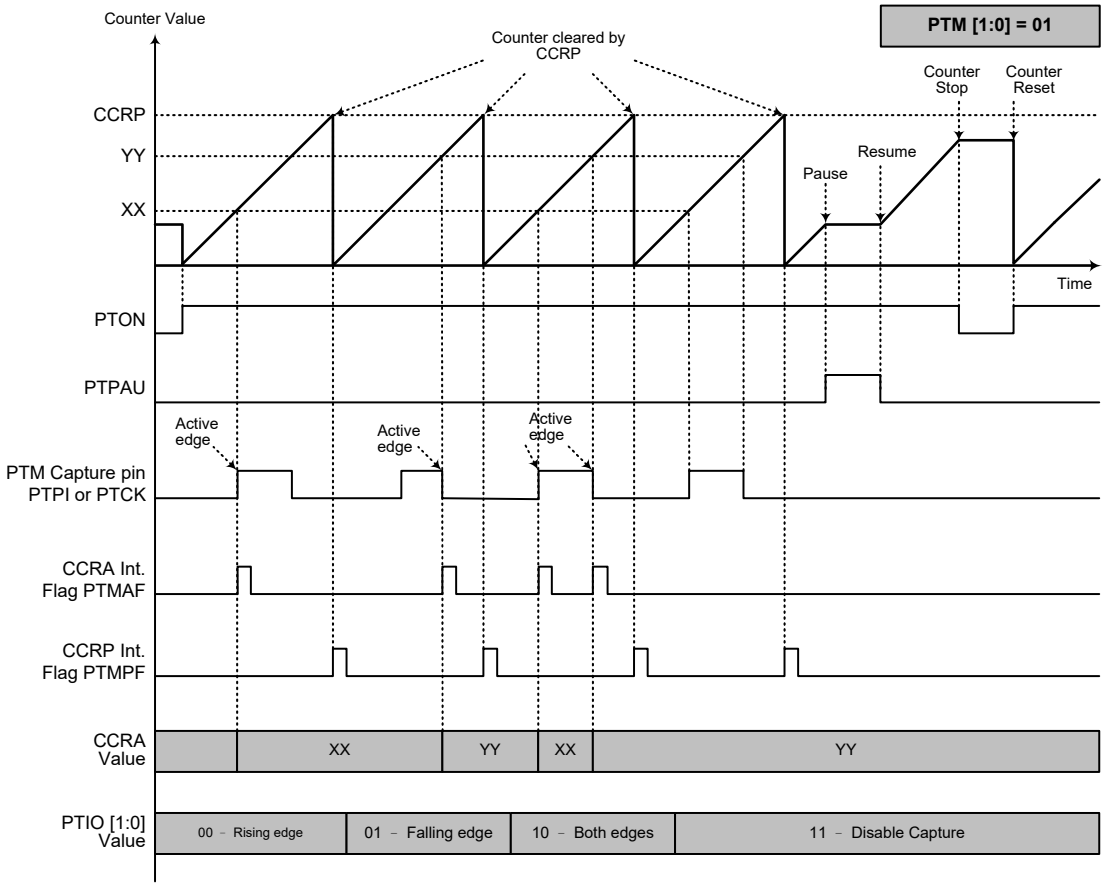
注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 PTCK 脚或设置 PTON 位为高来触发脉冲
4. PTCK 脚有效沿会自动置高位 PTON
5. 单脉冲输出模式中，PTIO[1:0] 需置位“11”，且不能更改

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。

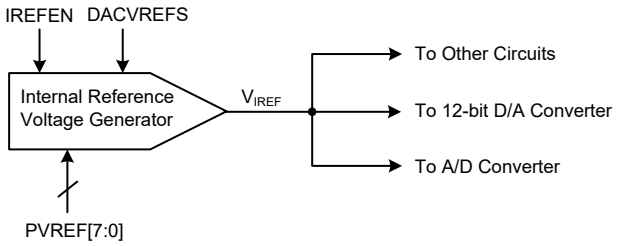


捕捉输入模式

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电气特性章节。



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进行微调。

寄存器名称	位							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IREFC	—	—	—	IREFEN	—	DACVREFS	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	—	DACVREFS	DACVRS1	DACVRS0
R/W	—	—	—	R/W	—	R/W	R/W	R/W
POR	—	—	—	0	—	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
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　未定义，读为“0”

Bit 2　**DACVREFS**：V_{IREF} 电压选择
0：1.25V
1：1.83V

Bit 1~0　**DACVRS1~DACVRS0**：12-bit D/A 转换器参考电压 $V_{DACVREF}$ 选择
详细描述见 D/A 转换器寄存器章节。

● **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

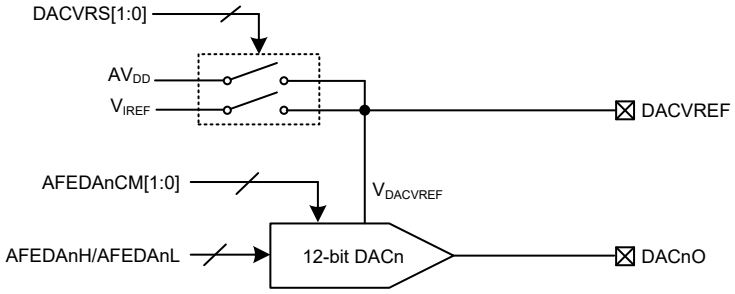
Bit 7~0 **D7~D0**：内部参考电压发生器微调控制

若 V_{IREF} 电压为 1.25V，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 -60mV~+60mV (基于 PVREF=80H)。PVREF 寄存器的值每增加一，输出的参考电压将减少约 500μV；反之，每减少一，将增加约 500μV。

若 V_{IREF} 电压为 1.83V，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 -70mV~+70mV (基于 PVREF=80H)。PVREF 寄存器的值每增加一，输出的参考电压将减少约 1000μV；反之，每减少一，将增加约 1000μV。

D/A 转换器 – DAC

该单片机内置三个 12-bit D/A 转换器，可提供 0~$V_{DACVREF}$ 的电压给 OPA 正输入端使用。D/A 转换器参考电压 $V_{DACVREF}$ 还可以用作 A/D 转换器的参考电压。



D/A 转换器结构 (n=1~3)

D/A 转换器寄存器介绍

D/A 转换器所有功能由多个寄存器控制。IREFC 寄存器中的 DACVRS1~DACVRS0 可用于选择 D/A 转换器参考电压，AFEDAnC 寄存器用于 D/A 转换器 n 的使能 / 除能控制，剩下的寄存器为三对 12-bit 寄存器 AFEDAnH 和 AFEDAnL 用于 D/A 转换器 n 输出控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
IREFC	—	—	—	IREFEN	—	DACVREFS	DACVRS1	DACVRS0
AFEDAnC	—	—	—	—	—	—	AFEDAnCM1	AFEDAnCM0
AFEDAnL	D3	D2	D1	D0	—	—	—	—
AFEDAnH	D11	D10	D9	D8	D7	D6	D5	D4

D/A 转换器寄存器列表 (n=1~3)

● **IREFC 寄存器**

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

Bit 7~5　未定义，读为“0”

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

Bit 3　未定义，读为“0”

Bit 2　**DACVREFS**：V_{IREF} 电压选择
具体描述见内部参考电压发生器章节。

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

● **AFEDAnC 寄存器 (n=1~3)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	AFEDAnCM1	AFEDAnCM0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未定义，读为“0”

Bit 1~0　**AFEDAnCM1~AFEDAnCM0**：12-bit D/A 转换器 n 模式控制
00：除能，输出为高阻抗状态
01/11：使能
10：除能，输出为接地状态

● **AFEDAnH & AFEDAnL 寄存器 (n=1~3)**

寄存器	AFEDAnH								AFEDAnL							
位	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	—	—	—	—

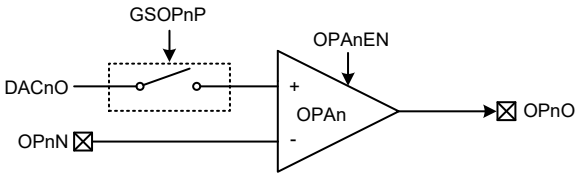
“—”：未定义，读为“0”

D11~D0：12-bit D/A 转换器 n 输出控制位
AFEDAnH 寄存器的 bit 7 ~ bit 0 结合 AFEDAnL 寄存器的 bit 7 ~ bit 4 形成了一个 12-bit D/A 转换器 n 的值，范围为 0~4095。
DACn 输出电压 = $V_{DACVREF}$×(DACn 值 / 4096)

注：需先写入 AFEDAnL 后再写入 AFEDAnH 寄存器。

运算放大器 – OPA

该单片机包含三个运算放大器 OPA1、OPA2 和 OPA3，可用于测量应用产品。12-bit D/A 转换器为运算放大器正输入端提供 0~$V_{DACVREF}$ 的电压。通过合理的设置可将 OPAn (n=1~3) 输出电压内部连接到 A/D 转换器输入通道进行测量。



运算放大器结构 (n=1~3)

运算放大器寄存器介绍

内部运算放大器的所有操作由一系列寄存器控制。OPAC 和 GSC1 寄存器用于 OPAn 功能的使能 / 除能控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
OPAC	OPA3EN	OPA2EN	OPA1EN	—	—	—	—	—
GSC1	—	—	—	—	—	GSOP3P	GSOP2P	GSOP1P

运算放大器寄存器列表

- **OPAC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OPA3EN	OPA2EN	OPA1EN	—	—	—	—	—
R/W	R/W	R/W	R/W	—	—	—	—	—
POR	0	0	0	—	—	—	—	—

Bit 7 **OPA3EN**：OPA3 使能 / 除能控制
0：除能
1：使能

Bit 6 **OPA2EN**：OPA2 使能 / 除能控制
0：除能
1：使能

Bit 5 **OPA1EN**：OPA1 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。

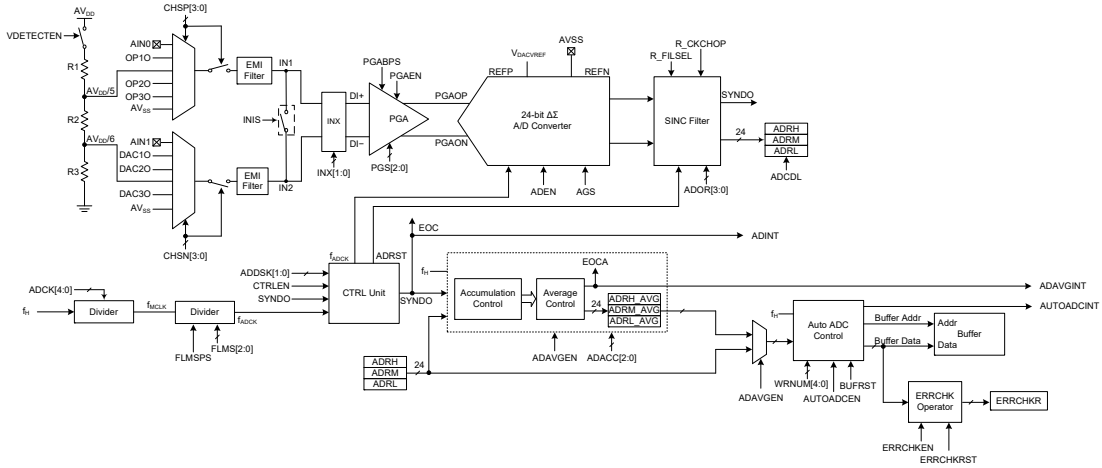
A/D 转换器

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 组差分外部输入通道组成。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 AV_{DD} 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

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。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PWRC	—	—	—	—	VDETECTEN	—	—	—
PGAC	—	INIS	INX1	INX0	AGS	PGS2	PGS1	PGS0
PGACS	CHSN3	CHSN2	CHSN1	CHSN0	CHSP3	CHSP2	CHSP1	CHSP0
MODC	—	—	—	—	ADEN	ADRST	PGABPS	PGAEN
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	—	—	—	ADOR3	ADOR2	ADOR1	ADOR0	—
ADCR1	FLMS2	FLMS1	FLMS0	FLMSPS	—	ADCDL	EOC	—
ADCS	—	—	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0
SINC3	D7	D6	D5	D4	D3	D2	R_FILSEL	R_CKCHOP
CTRL	CTRLEN	ADDSK1	ADDSK0	—	—	—	—	—
ADACCTRL	ADAVGEN	EOCA	—	—	—	ADACC2	ADACC1	ADACC0
ADRL_AVG	D7	D6	D5	D4	D3	D2	D1	D0
ADRM_AVG	D15	D14	D13	D12	D11	D10	D9	D8
ADRH_AVG	D23	D22	D21	D20	D19	D18	D17	D16
AUTOADCC	AUTOADCEN	—	WRNUM4	WRNUM3	WRNUM2	WRNUM1	WRNUM0	BUFRST
ERRCHKC	ERRCHKEN	ERRCHKRST	—	—	—	—	—	—
ERRCHKR	D7	D6	D5	D4	D3	D2	D1	D0

A/D 转换寄存器列表

可编程增益放大器寄存器 – PWRC, PGAC, PGACS, MODC

有四个与可编程增益相关的控制寄存器，PWRC、PGAC、PGACS 和 MODC。PWRC 寄存器用于控制 $AV_{DD}/5$ 和 $AV_{DD}/6$ 分压器。PGAC 寄存器用于选择 PGA 增益、A/D 转换器增益和定义 PGA 输入端连接。MODC 寄存器用于定义 A/D 转换器使能 / 除能控制、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。

- **PWRC 寄存器**

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

Bit 7~4　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 2~0　未定义，读为“0”

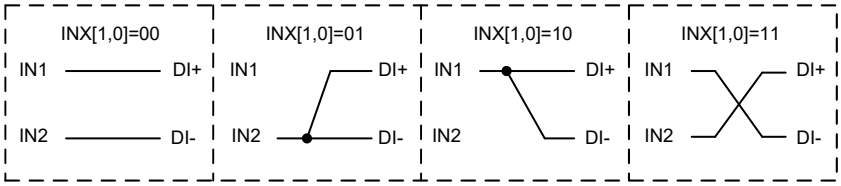
● **PGAC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	INIS	INX1	INX0	AGS	PGS2	PGS1	PGS0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7　未定义，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
当 PGAEN=0 且 ADEN=0 时，INIS 位的设定是无效的。

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



Bit 3　**AGS**：A/D 转换器 PGAOP/PGAON 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

● **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：$AV_{DD}/6$
1100：AV_{SS}
1101：AIN1
1110~1111：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

● **MODC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	ADEN	ADRST	PGABPS	PGAEN
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未定义，读为“0”

Bit 3　**ADEN**：A/D 转换器使能 / 除能控制
0：除能
1：使能

Bit 2　**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 1　**PGABPS**：PGA 旁路控制
0：正常模式
1：旁路 PGA 模式
若此位设置为高，则可以不考虑 PGAEN 位的设置。

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。

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, ADCS

寄存器 ADCR0、ADCR1 和 ADCS 用来控制 A/D 转换器的功能和操作。这些 8 位的寄存器定义包括选择 A/D 转换过采样率以及设置 A/D 转换器的时钟等。

● **ADCR0 寄存器**

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

Bit 7~5 未定义，读为“0”

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	FLMSPS	—	ADCDL	EOC	—
R/W	R/W	R/W	R/W	R/W	—	R/W	R/W	—
POR	0	0	0	0	—	0	0	—

Bit 7~5 **FLMS2~FLMS0**：A/D 转换器时钟 f_{ADCK} 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
注：当 FLMSPS=1，ADCK[4:0]=11111，IRC2=1 ($f_{ADCK}=f_{MCLK}=f_H=f_{MIRC}$=400kHz)，FLMS[2:0] 建议设置为 010，以获得较佳的测量性能。

Bit 4 **FLMSPS**：A/D 转换器时钟除频旁路控制
0：除能
1：使能，$f_{ADCK}=f_{MCLK}$

Bit 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。读取后该位可清零以除能 A/D 转换器数据锁存功能，以便存储后续转换的结果。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”

- **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 转换器时钟源 f_{MCLK} 设置
00000~11110：$f_{MCLK}=f_H/2/(ADCK[4:0]+1)$
11111：$f_{MCLK}=f_H$

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 应设为逻辑低)，数据延迟 = 3 个输出数据时钟
1：SINC3 滤波器 (R_CKCHOP 应设为逻辑高)，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 应设为逻辑低)
1：除能 (R_FILSEL 应设为逻辑高)

控制单元 (CTRL Unit) 寄存器 – CTRL

有一个与控制单元控制相关的寄存器，CTRL。该寄存器用于定义控制单元的使能 / 除能控制以及选择要跳过的 A/D 转换数据的数量。

- **CTRL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	CTRLEN	ADDSK1	ADDSK0	—	—	—	—	—
R/W	R/W	R/W	R/W	—	—	—	—	—
POR	0	0	0	—	—	—	—	—

Bit 7 **CTRLEN**：控制单元使能 / 除能控制
0：除能
1：使能
若控制单元使能，则可以解决 A/D 转换器电路复位后初始状态差的问题。第

一个转换的数据不需要被丢弃。并且在 A/D 转换完成后，控制单元将根据 ADDSK[1:0] 位的设置自动跳过转换后的数据值。如果除能控制单元，则不会跳过转换数据。

Bit 6~5 **ADDSK1~ADDSK0**：选择要跳过的 A/D 转换数据的数量
00：1
01：2
10：3
11：4

Bit 4~0 未定义，读为“0”

A/D 数据累加平均寄存器 – ADACCTRL, ADRL_AVG, ADRM_AVG, ADRH_AVG

寄存器 ADACCTRL、ADRL_AVG、ADRM_AVG 和 ADRH_AVG 用来控制 A/D 数据累加平均功能和操作。ADACCTRL 寄存器用于定义该功能使能 / 除能控制以及累积平均次数。ADRL_AVG、ADRM_AVG 和 ADRH_AVG 寄存器用于存放累加平均后的值。

● ADACCTRL 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ADAVGEN	EOCA	—	—	—	ADACC2	ADACC1	ADACC0
R/W	R/W	R/W	—	—	—	R/W	R/W	R/W
POR	0	0	—	—	—	0	0	0

Bit 7 **ADAVGEN**：A/D 转换数据累加平均使能 / 除能控制
0：除能
1：使能

Bit 6 **EOCA**：A/D 转换数据累加平均结束标志位
0：A/D 转换数据处于累加平均中
1：A/D 转换数据累加平均结束
此位必须通过软件清零。

Bit 5~3 未定义，读为“0”

Bit 2~0 **ADACC2~ADACC0**：选择 A/D 转换器输出执行累加平均次数
000：2^2 次
001：2^3 次
010：2^4 次
011：2^5 次
其他值：2^6 次

● ADRL_AVG 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 转换数据平均寄存器 bit 7 ~ bit 0

● ADRM_AVG 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 转换数据平均寄存器 bit 15 ~ bit 8

● **ADRH_AVG 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 转换数据平均寄存器 bit 23 ~ bit 16

A/D 转换器自动模式寄存器 – AUTOADCC

有一个与 A/D 转换器自动模式控制相关的寄存器，AUTOADCC。该寄存器用于定义 A/D 转换器自动模式的使能 / 除能控制以及选择要写入到缓冲器的 A/D 转换数据的数量。

● **AUTOADCC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	AUTOADCEN	—	WRNUM4	WRNUM3	WRNUM2	WRNUM1	WRNUM0	BUFRST
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 **AUTOADCEN**：A/D 转换器自动模式使能 / 除能控制
0：除能
1：使能
若 A/D 转换器自动模式使能，A/D 转换数据会自动发送到缓冲器中，直到写入缓冲器中的数据达到 WRNUM[4:0] 位设定的数量，产生 A/D 转换器自动模式中断，A/D 转换器自动模式会自动除能。

Bit 6 未定义，读为“0”

Bit 5~1 **WRNUM4~WRNUM0**：设置要写入到缓冲器的 A/D 转换数据的数量
00000：1
:
10011：20
10100~11111：21
缓冲器为 64 字节，一次最多可以将 21 个 A/D 转换数据写入到缓冲器中。

Bit 0 **BUFRST**：缓冲器地址软件复位使能 / 除能控制
0：除能
1：使能
此位用于重置缓冲器写入地址。若此位为高，则缓冲器写入地址将重置为 00H。复位完成后，此位由硬件自动清零。

ERRCHK 运算子寄存器 – ERRCHKC, ERRCHKR

有两个与 A/DERRCHK 运算子控制相关的寄存器，ERRCHKC 和 ERRCHKR。ERRCHKC 寄存器用于定义 ERRCHK 运算子的使能 / 除能控制以及 ERRCHK 运算结果软件复位的使能 / 除能控制。ERRCHKR 寄存器用于存放 ERRCHK 操作结果。

● **ERRCHKC 寄存器**

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

Bit 7 **ERRCHKEN**：ERRCHK 运算子使能 / 除能控制
0：除能
1：使能
若此位置高，将对写入缓冲器的每个数据执行按字节的异或运算。结果存放在 ERRCHKR 寄存器中。

Bit 6 **ERRCHKRST**：ERRCHK 运算结果软件复位使能 / 除能控制
0：除能
1：使能
此位用于清除 ERRCHKR 寄存器的值。清除后，此位会自动清零。

Bit 5~0 未定义，读为“0”

● **ERRCHKR 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

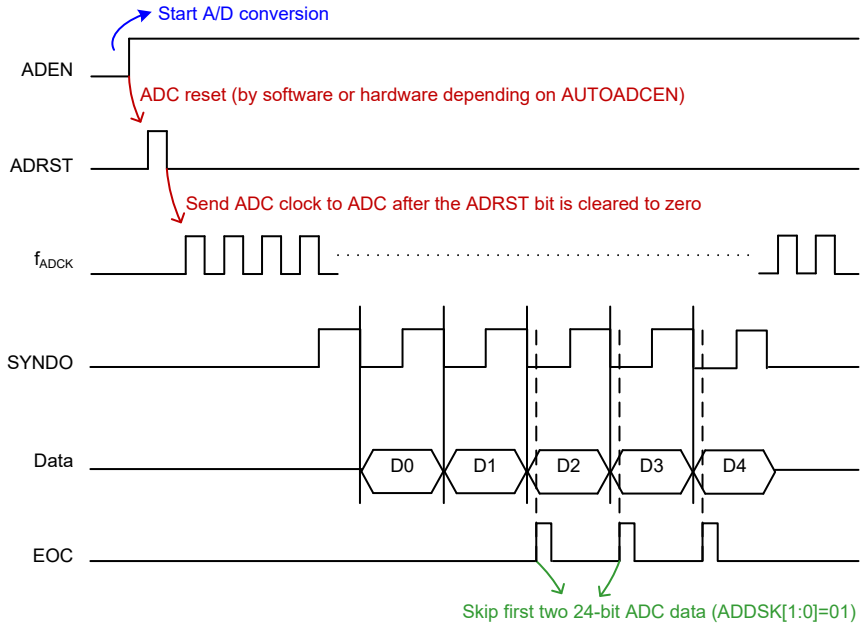
Bit 7~0 **D7~D0**：ERRCHK 操作结果寄存器 bit 7 ~ bit 0

控制单元 (CTRL Unit)

控制单元用于在 A/D 转换期间让单片机保持在空闲模式，并优化 A/D 转换器时序。必须确保在单片机执行 HALT 指令进入空闲模式之前，CTRLEN 和 ADEN 位已置高。

若 AUTOADCEN=0，即 A/D 转换器处于正常模式，当单片机设置 ADRST 位从逻辑低到逻辑高，然后再到逻辑低后，将 A/D 转换器时钟 f_{ADCK} 发送到 A/D 转换器。若 AUTOADCEN=1，即 A/D 转换器处于自动模式，PGAEN 和 ADEN 位由硬件置高以执行 A/D 转换。若确认 MODC 寄存器中的 ADEN 位为高，则将向 A/D 转换器发送 ADRST 信号，然后将 A/D 转换器时钟 f_{ADCK} 发送到 A/D 转换器。

待 A/D 转换完成后，根据 ADDSK[1:0] 位的设定，决定跳过几笔数据。若 ADDSK[1:0]=01，即跳过前 2 个 SYNDO 信号，第 3 个 SYNDO 信号产生后，才会置高 EOC 位。转换后的值可从 ADRL、ADRM 和 ADRH 寄存器中读取。若 A/D 转换器自动模式使能，ADRL、ADRM 和 ADRH 寄存器中的值会自动写入缓冲器。



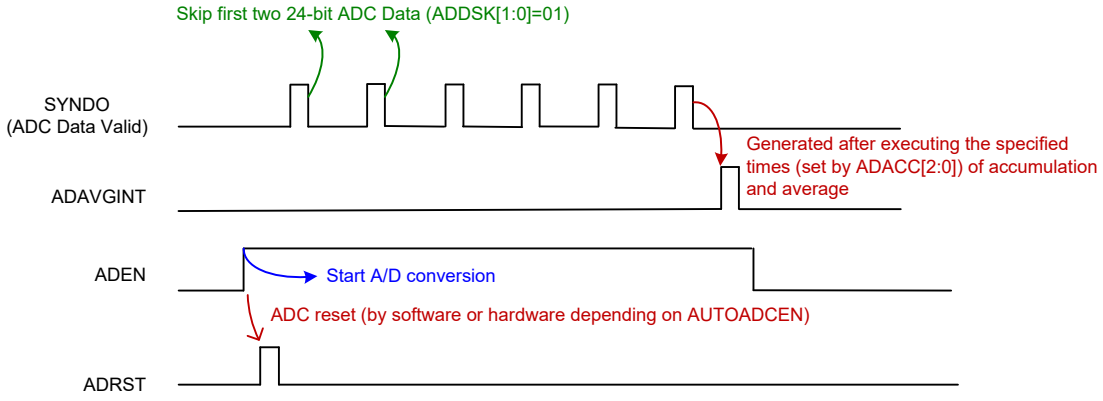
A/D 转换器控制单元时序图 – ADDSK[1:0]=01

A/D 数据累加平均

累加平均功能用于累加和平均 24-bit A/D 转换器的输出，此功能由硬件实现。此时，单片机可执行 HALT 指令以减少平均耗电流。必须确保在单片机执行 HALT 指令进入空闲模式之前，ADEN 位已置高。

累加平均的次数由 ADACC[2:0] 位设置。当累加平均执行完毕之后会产生一个 A/D 转换器平均中断。平均后的数据可从 ADRL_AVG、ADRM_AVG 和 ADRH_AVG 寄存器中读取。在中断产生前，ADRL_AVG、ADRM_AVG 和 ADRH_AVG 寄存器中的内容是无效的。若 A/D 转换器自动模式使能，ADRL_AVG、ADRM_AVG 和 ADRH_AVG 寄存器中的值会自动写入缓冲器。

置高 ADAVGEN 位以使能累加平均功能。当 ADEN 位为高时，启动累加平均功能。若在累加平均尚未完成时将 ADEN 位清零，表示累加平均被强制停止，此时不会产生中断。



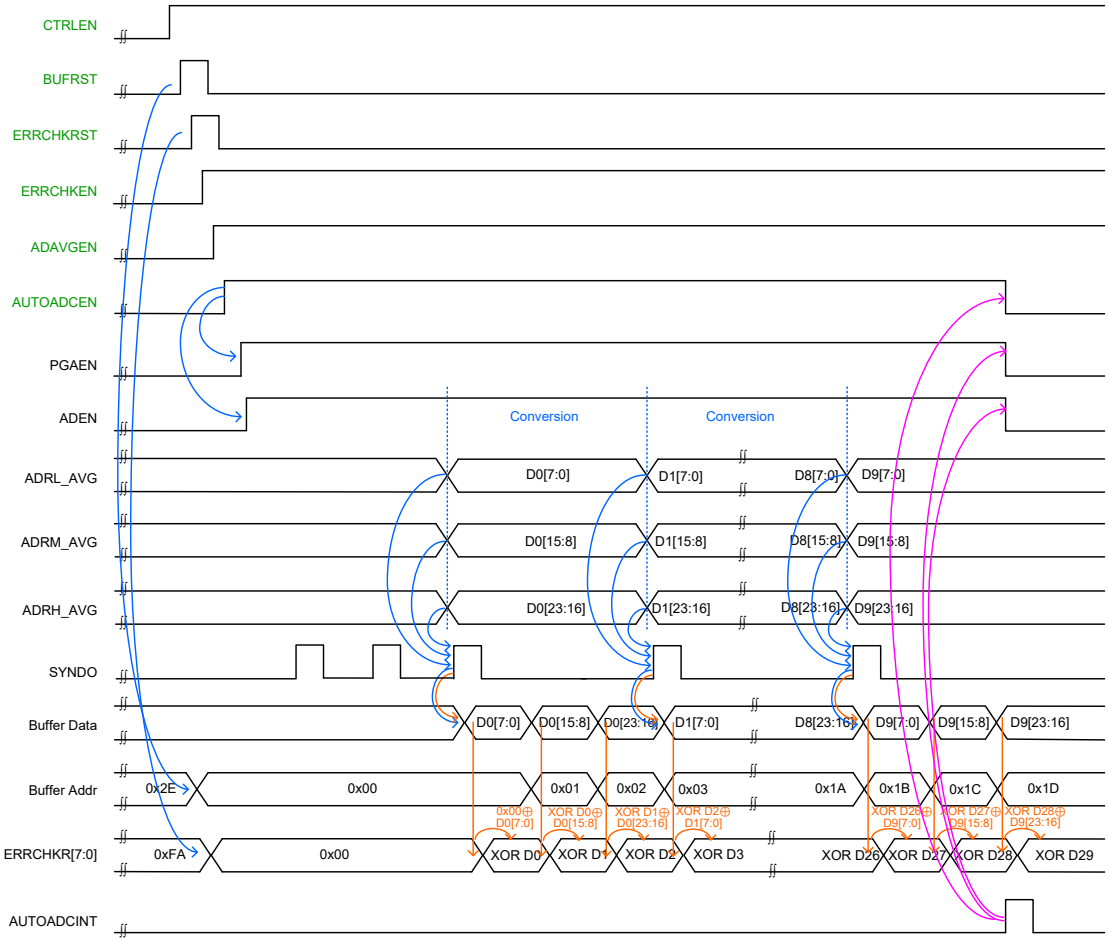
A/D 数据累加平均时序图 – ADACC[2:0]=000, ADAVGEN=1

A/D 转换器自动模式

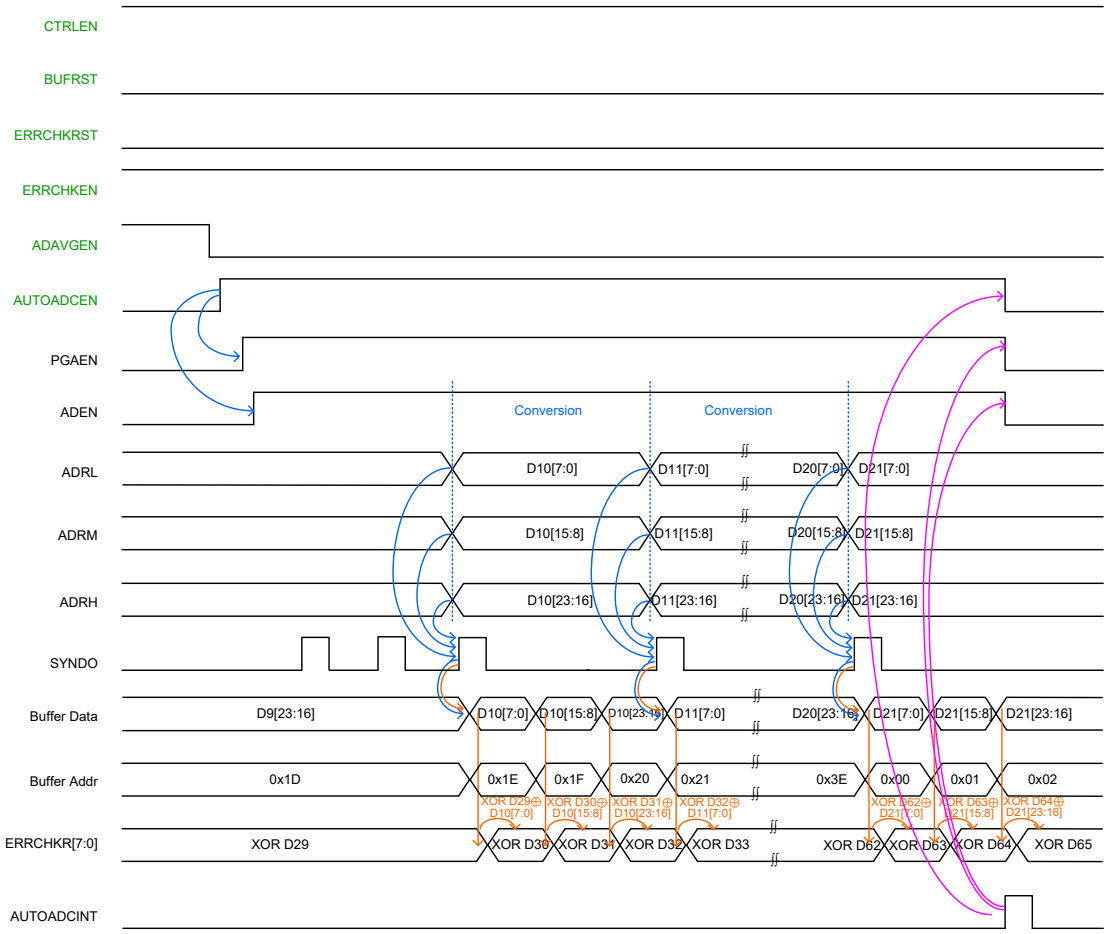
A/D 转换器自动模式用于在单片机处于空闲模式时自动将转换后的数据写入缓冲器，直到写入到缓冲器的数据达到 WRNUM[4:0] 位设定的数量时，会产生 A/D 转换器自动模式中断以唤醒单片机。必须确保在单片机执行 HALT 指令进入空闲模式之前，PGAEN 和 ADEN 位已清零，且 AUTOADCEN 位已置高。

置高 AUTOADCEN 位以使能 A/D 转换器自动模式，PGAEN 和 ADEN 位由硬件自动置高以开启 A/D 转换。自动控制电路会将 ADRL、ADRM 和 ADRH 寄存器或 ADRL_AVG、ADRM_AVG 和 ADRH_AVG 寄存器中的数据写入缓冲器中，取决于 ADAVGEN 位的设置。该缓冲器位于 RAM Sector 3 的 00H~3FH。一旦进入 A/D 转换器自动模式，CPU 就无法向此 RAM 区域写入 / 读取任何数据。当写入缓冲器中的数据达到 WRNUM[4:0] 位设定的数量时，PGAEN、ADEN 和 AUTOADCEN 位会被硬件清零并产生 A/D 转换器自动模式中断以唤醒单片机，除能 A/D 转换器自动模式。

RAM 地址由软件复位。若地址未复位，当地址写到 3EH 后，会从 00H 继续写入，覆盖先前的数据。在 A/D 转换器自动模式下，若 ERRCHKEN 位置高，则会对写入缓冲器的每个数据执行按字节的异或运算。可以从 ERRCHKR 寄存器中读取结果。



A/D 转换器自动模式时序图 – ADDSK[1:0]=01, ADAVGEN=1, WRNUM[4:0]=01001



A/D 转换器自动模式时序图 – ADDSK[1:0]=01, ADAVGEN=0, WRNUM[4:0]=01011

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

$=f_{ADCK}/(2\times OSR)$

$=(f_{MCLK}/N)/(2\times OSR)$

$=f_{MCLK}/(N\times 2\times OSR)$

SINC3 的数据传输率

$=f_{ADCK}/OSR$

$=(f_{MCLK}/N)/OSR$

$=f_{MCLK}/(N\times OSR)$

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_H 或 $f_H/2/(ADCK[4:0]+1)$，通过 ADCK[4:0] 位来设置；

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，可以得到数据传输率 = 4MHz/(30×2×8192)=8Hz。

对于 SINC3 滤波器，设置 ADOR[3:0]=0001B，选择过采样率为 16384。因此，可以得到数据传输率 = 4MHz/(30×16384)=8Hz。

A/D 转换器操作

要打开 A/D 转换器，首先应将 MODC 寄存器中的 ADEN 位置高以除能 A/D 转换器的暂停模式，以确保 A/D 转换器可以通电。MODC 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，会在 SINC 滤波器中启动一个模数转换周期。设置完成后，A/D 转换器可以开始工作。这两位用于控制内部 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_H 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

下面概述实现正常模式下单次 A/D 转换过程的各个步骤。

- 步骤 1

  通过 PGAC 寄存器，选择 PGA、A/D 转换器增益和 PGA 输入端连接。

- 步骤 2

  通过 IREFC 寄存器，选择 $V_{DACVREF}$ 电压。

- 步骤 3

  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 4

  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 5

  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 6

  通过 MODC 寄存器中的 PGAEN 和 ADEN 位，使能 PGA 和 A/D 转换器。

- 步骤 7

  通过置高 MODC 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

  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。

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±，等于 $V_{DACVREF}$ 电压。因此每一位可表示 ΔVR±/8388608 的模拟输入值。

1 LSB=ΔVR±/8388608

通过下面的等式可计算 A/D 转换器输入电压值：

ΔSI=PGAGN×ADGN×ΔDI

A/D 转换数据 = (ΔSI/ΔVR±)×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：经过放大后的差分输入信号

3. PGAGN：PGA 增益

4. ADGN：A/D 转换器增益

5. ΔDI：差分输入信号，来自外部通道或内部信号

6. ΔVR±：差分参考电压

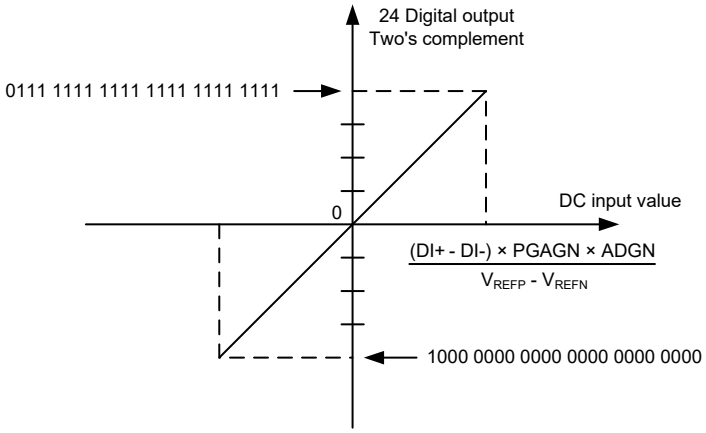
由于 Delta Sigma A/D 转换器的数字系统设计，其转换的最大值为 8388607，最小值为 -8388608，因此有一个中间值 0。A/D 转换数据公式说明了转换值的变化范围。

A/D 转换数据 (二进制补码，十六进制值)	十进制值
0x7FFFFF	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 × LSB)/(PGAGN × ADGN)

如果 MSB = 1 (转换数据为负数)

输入电压 = (转换数据的补码 × LSB)/(PGAGN × ADGN)

注：补码 = 反码 + 1

SPI 串行接口

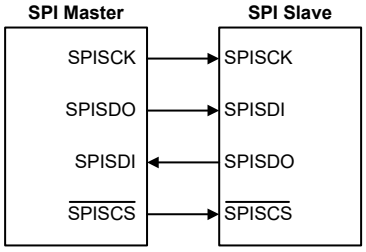
SPI 接口常用于与外部设备如传感器、闪存或 EEPROM 内存等通信。四线 SPI 接口最初是由摩托罗拉公司研制，是一个有相当简单的通信协议的串行数据接口，这个协议可以简化与外部硬件的编程要求。

SPI 通信模式为全双工模式，且能以主 / 从模式的工作方式进行通信，单片机既可以作为主机，也可以作为从机。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 $\overline{\text{SPISCS}}$。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：SPISDI、SPISDO、SPISCK 和 $\overline{\text{SPISCS}}$。SPISDI 和 SPISDO 是数据的输入和输出线。SPISCK 是串行时钟线，$\overline{\text{SPISCS}}$ 是从机的选择线。SPI 的接口引脚与其它功能共用引脚。通过设定引脚共用功能选择寄存器的对应位，来选择 SPI 接口引脚。SPI 接口可以通过 SPIC0 寄存器中的 SPIEN 位来除能或使能。连接到 SPI 接口的单片机以从主 / 从模式进行通信，且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 SPISCS 引脚，所以只能拥有一个从机设备。若 SPI 功能使能且引脚用作 SPI 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。

可通过软件控制 $\overline{\text{SPISCS}}$ 引脚使能与除能，设置 SPICSEN 位为“1”使能 $\overline{\text{SPISCS}}$ 功能，设置 SPICSEN 位为“0”，$\overline{\text{SPISCS}}$ 引脚将处于浮空状态。

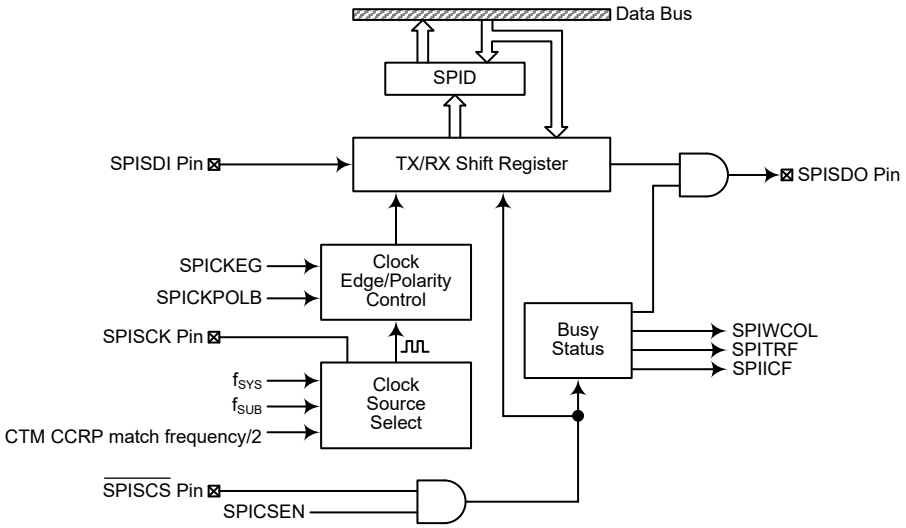


SPI 主 / 从机连接方式

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 SPICSEN、SPIEN 位的状态。



SPI 方框图

SPI 寄存器

有三个寄存器用于控制 SPI 接口的所有操作，其中有一个数据寄存器 SPID、两个控制寄存器 SPIC0 和 SPIC1。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SPIC0	SPIM2	SPIM1	SPIM0	—	—	—	SPIEN	SPIICF
SPIC1	—	—	SPICKPOLB	SPICKEG	SPIMLS	SPICSEN	SPIWCOL	SPITRF
SPID	D7	D6	D5	D4	D3	D2	D1	D0

SPI 寄存器列表

SPI 数据寄存器

SPID 用于存储发送和接收的数据。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 SPID 中。SPI 总线接收到数据之后，单片机就可以从 SPID 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 SPID 实现。

- **SPID 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**：SPI 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，SPIC0 和 SPIC1。寄存器 SPIC0 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 SPIC1 用于其它的控制功能如 LSB/MSB 选择，写冲突标志位等。

● **SPIC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SPIM2	SPIM1	SPIM0	—	—	—	SPIEN	SPIICF
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	1	1	1	—	—	—	0	0

Bit 7~5 **SPIM2~SPIM0**：SPI 工作模式控制位
000：SPI 主机模式；SPI 时钟为 $f_{SYS}/4$
001：SPI 主机模式；SPI 时钟为 $f_{SYS}/16$
010：SPI 主机模式；SPI 时钟为 $f_{SYS}/64$
011：SPI 主机模式；SPI 时钟为 f_{SUB}
100：SPI 主机模式；SPI 时钟为 CTM CCRP 匹配频率 / 2
101：SPI 从机模式
110：SPI 除能
111：SPI 除能
这几位用于设置 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 CTM。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4~2 未定义，读为“0”

Bit 1 **SPIEN**：SPI 控制位
0：除能
1：使能
此位为 SPI 接口的开 / 关控制位。此位为“0”时，SPI 接口除能，SPISDI、SPISDO、SPISCK 和 $\overline{\text{SPISCS}}$ 脚将失去 SPI 功能，SPI 工作电流减小到最小值。此位为“1”时，SPI 接口使能。

Bit 0 **SPIICF**：SPI 未完成标志位
0：未发生
1：发生
此位仅当 SPI 配置在 SPI 从机模式时有效。如果 SPI 工作在从机模式且 SPIEN 和 SPICSEN 位都为“1”，但在 SPI 数据传输完全结束前 $\overline{\text{SPISCS}}$ 线被外部主机拉高，SPIICF 和 SPITRF 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 SPIICF 位是由软件应用程序设为 1，那么 SPITRF 位将不会置高。

● **SPIC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	SPICKPOLB	SPICKEG	SPIMLS	SPICSEN	SPIWCOL	SPITRF
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 未定义，读为“0”

Bit 5 **SPICKPOLB**：SPI 时钟线的基础状态位
0：当时钟无效时，SPISCK 引脚为高电平
1：当时钟无效时，SPISCK 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 SPISCK 为低电平，若此位为低，当时钟无效时 SPISCK 为高电平。

Bit 4 **SPICKEG**：SPI 的 SPISCK 有效时钟边沿类型位
SPICKPOLB=0
0：SPISCK 为高电平且在 SPISCK 上升沿抓取数据
1：SPISCK 为高电平且在 SPISCK 下降沿抓取数据
SPICKPOLB=1
0：SPISCK 为低电平且在 SPISCK 下降沿抓取数据
1：SPISCK 为低电平且在 SPISCK 上升沿抓取数据

SPICKEG 和 SPICKPOLB 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。SPICKPOLB 位决定时钟线的基本状态，若时钟无效且此位为高，则 SPISCK 为低电平，若时钟无效且此位为低，则 SPISCK 为高电平。SPICKEG 位决定有效时钟边沿类型，取决于 SPICKPOLB 的状态。

Bit 3 **SPIMLS**：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 **SPICSEN**：SPI $\overline{\text{SPISCS}}$ 引脚控制位
0：除能
1：使能
SPICSEN 位用于 $\overline{\text{SPISCS}}$ 引脚的使能 / 除能控制。此位为低时，$\overline{\text{SPISCS}}$ 除能并处于浮空状态。此位为高时，$\overline{\text{SPISCS}}$ 作为选择脚。

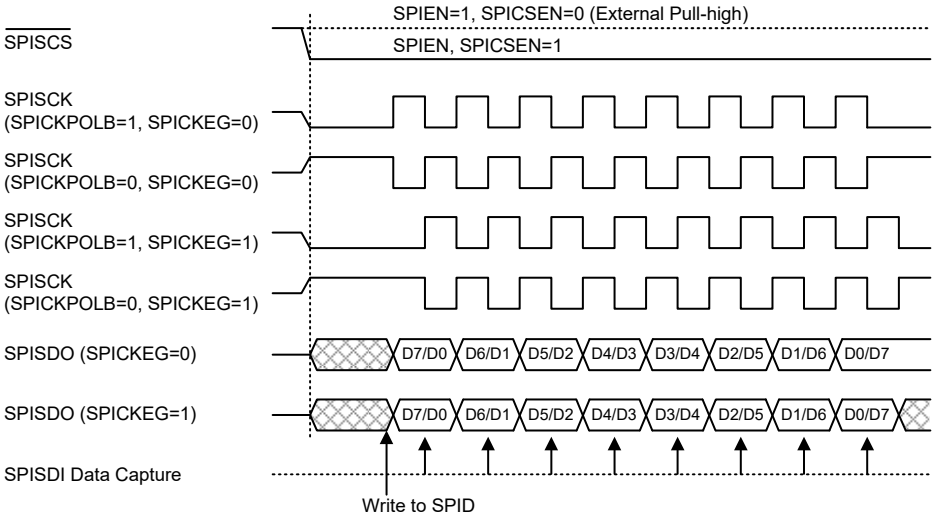
Bit 1 **SPIWCOL**：SPI 写冲突标志位
0：无冲突
1：冲突
SPIWCOL 标志位用于监测数据冲突的发生。此位为高时，表示在传输过程中有数据被写入 SPID 寄存器。若数据正在被传输时，此写操作无效。此位可被应用程序清零。

Bit 0 **SPITRF**：SPI 发送 / 接收结束标志位
0：数据正在发送
1：数据发送结束
SPITRF 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。

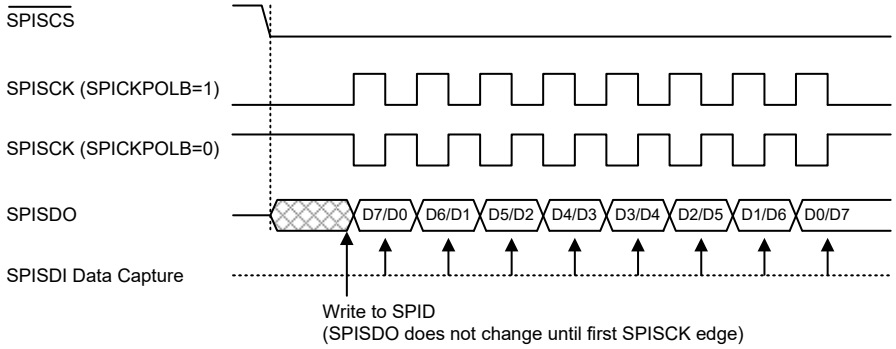
SPI 通信

将 SPIEN 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 SPID 的同时传输 / 接收开始进行。数据传输完成时，SPITRF 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 SPID 中的数据，而且在 SPISDI 引脚上的数据也会被移位到 SPID 寄存器中。主机应在输出时钟信号之前先输出一个 $\overline{\text{SPISCS}}$ 信号以使能从机，从机的数据传输功能也应在与 SPISCK 信号相关的适当时候准备就绪，这由 SPICKPOLB 和 SPICKEG 位决定。所附时序图表明了在 SPICKPOLB 和 SPICKEG 位各种设置情况下从机数据与 SPISCK 信号的关系。

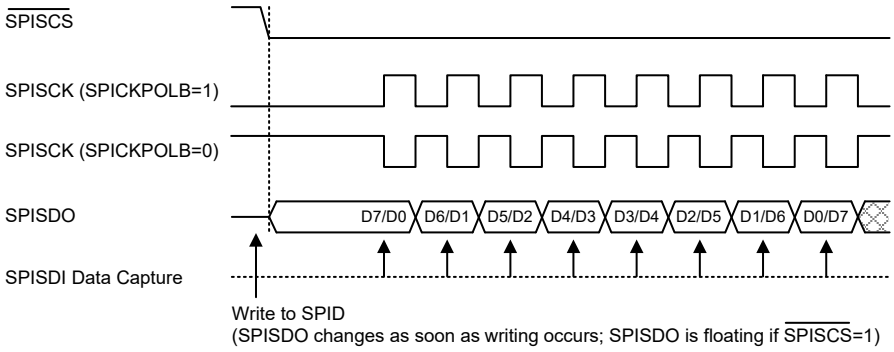
SPI 主机模式在 SPI 时钟运行情况下可以进行正常传输。



SPI 主机模式时序

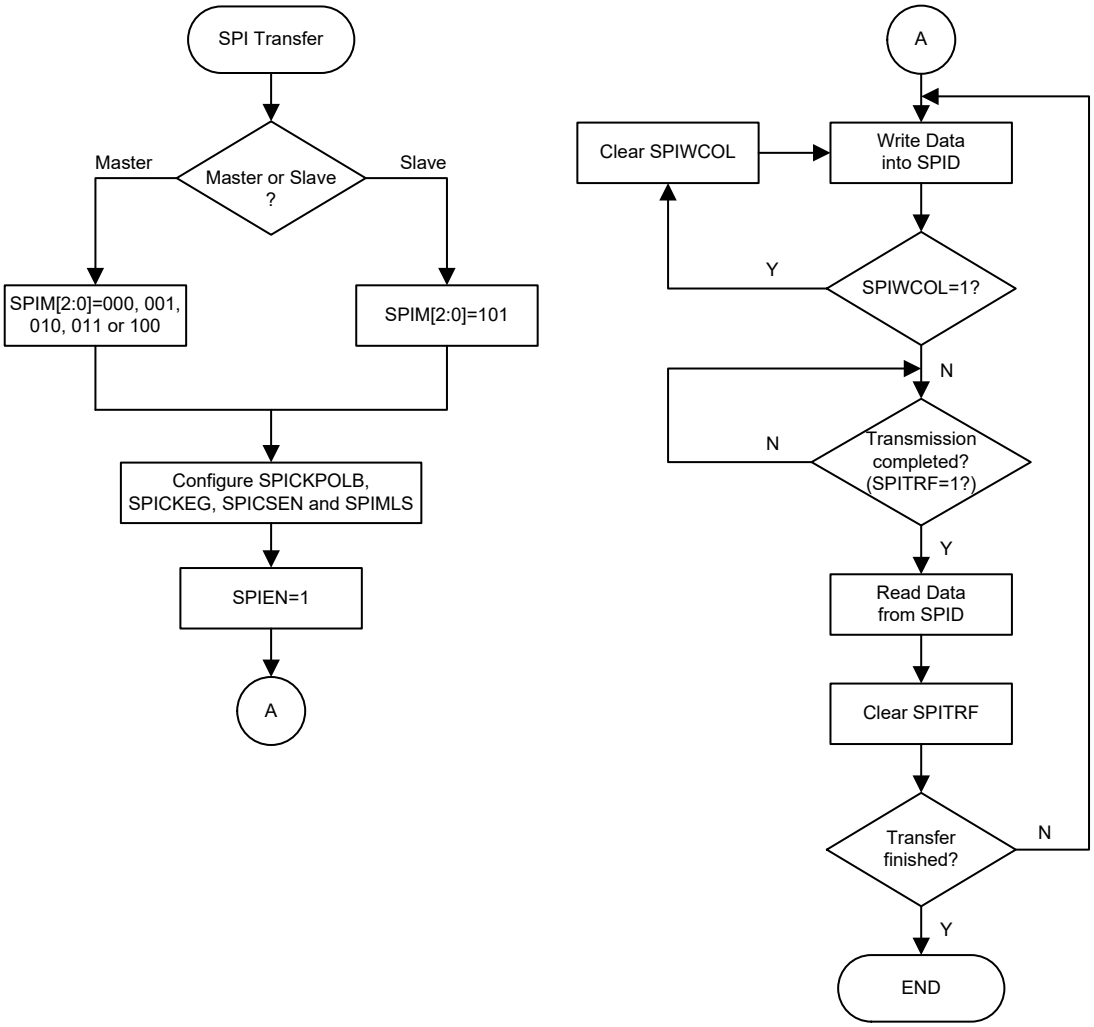


SPI 从机模式时序 – SPICKEG=0



Note: For SPI slave mode, if SPIEN=1 and SPICSEN=0, SPI is always enabled and ignores the SPISCS level.

SPI 从机模式时序 – SPICKEG=1



SPI 传输控制流程图

SPI 总线使能 / 除能

设置 SPICSEN=1、$\overline{\text{SPISCS}}$=0 将使能 SPI 总线，然后等待写数据到 SPID 寄存器 (TXRX 缓存器)。单片机处于主机模式，数据写入 SPID 寄存器后，自动开始数据传输或接收操作。数据传输完成时，SPITRF 位将自动被置位。单片机处于从机模式，SPISCK 引脚上收到脉冲信号之后，会传输 TXRX 中的数据，或 SPISDI 引脚上的数据也会被移入。

当 SPI 总线除能时，通过设置相应引脚共用控制位，SPISCK、SPISDI、SPISDO、$\overline{\text{SPISCS}}$ 可作为 I/O 口或其它共用引脚使用。

SPI 操作步骤

四线制 SPI 接口可完成所有主 / 从模式通信工作。由时序图可看出基本的总线工作流程。

在 SPIC1 寄存器中，SPICSEN 位控制 SPI 接口的所有功能。设置此位为高，$\overline{\text{SPISCS}}$ 信号线有效将使能 SPI 接口。设置此位为低，SPI 接口除能，$\overline{\text{SPISCS}}$ 信号线处于浮空状态因此不能控制 SPI 接口。SPICSEN 位和 SPIC0 寄存器中

的 SPIEN 位设置为高，使得 SPISDI 信号线处于浮空状态，SPISDO 信号线为高电平。主机模式中，如果 SPISCK 信号线为高还是低取决于 SPIC1 寄存器中的时钟极性选择位 SPICKPOLB。从机模式中，SPISCK 信号线处于浮空状态。如果 SPIEN 位设置为低，SPI 接口被除能，通过设置相应引脚共用控制位，SPISCK、SPISDI、SPISDO、$\overline{\text{SPISCS}}$ 可作为 I/O 口或其它共用引脚使用。主机模式中，当数据被写入 SPID 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 SPIC0 控制寄存器中的 SPIM2~SPIM0 位，选择 SPI 主机模式和时钟源。

- 步骤 2

  设置 SPICSEN 和 SPIMLS 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 SPIC0 控制寄存器中的 SPIEN 位，使能 SPI 接口功能。

- 步骤 4

  对于写操作：写数据到 SPID 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。再使用 SPISCK 和 SPISDO 信号线将数据输出。跳至步骤 5。

  对于读操作：从 SPISDI 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 SPID 寄存器。

- 步骤 5

  检测 SPIWCOL 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 SPITRF 位或等待 SPI 串行总线中断发生。

- 步骤 7

  从 SPID 寄存器中读数据。

- 步骤 8

  清除 SPITRF。

- 步骤 9

  跳回至步骤 4。

从机模式

- 步骤 1

  设置 SPIC0 控制寄存器中的 SPIM2~SPIM0 位，选择 SPI 从机模式。

- 步骤 2

  设置 SPICSEN 和 SPIMLS 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 SPIC0 控制寄存器中的 SPIEN 位，使能 SPI 接口功能。

- 步骤 4

  对于写操作：写数据到 SPID 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 SPISCK 信号和 $\overline{\text{SPISCS}}$ 信号。跳至步骤 5。

  对于读操作：从 SPISDI 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 SPID 寄存器。

- 步骤 5

  检测 SPIWCOL 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 SPITRF 位或等待 SPI 串行总线中断发生。

- 步骤 7

  从 SPID 寄存器中读数据。

- 步骤 8

  清除 SPITRF。

- 步骤 9

  跳回至步骤 4。

错误侦测

SPIC1 寄存器中的 SPIWCOL 位用于数据传输期间监测数据冲突的发生。此位由串行接口设置为高，而由应用程序来清除为零。在数据传输期间如果写数据到 SPID，此位被置高提示数据冲突发生，并阻止数据继续被写入。

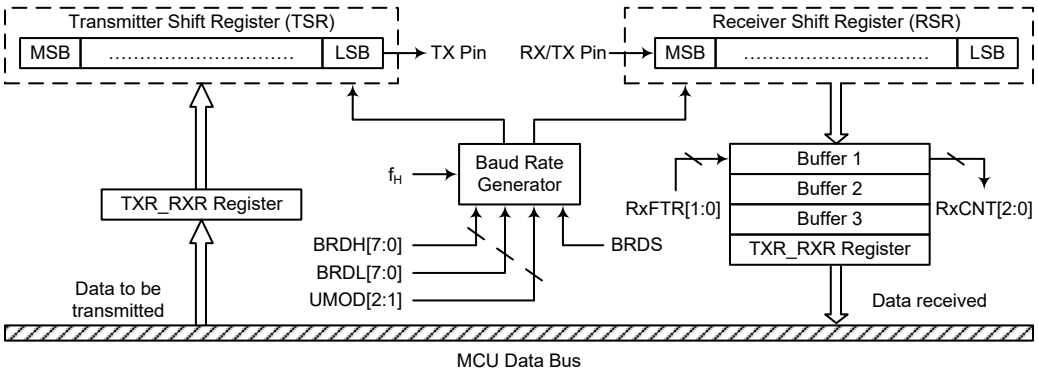
UART 串行接口

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

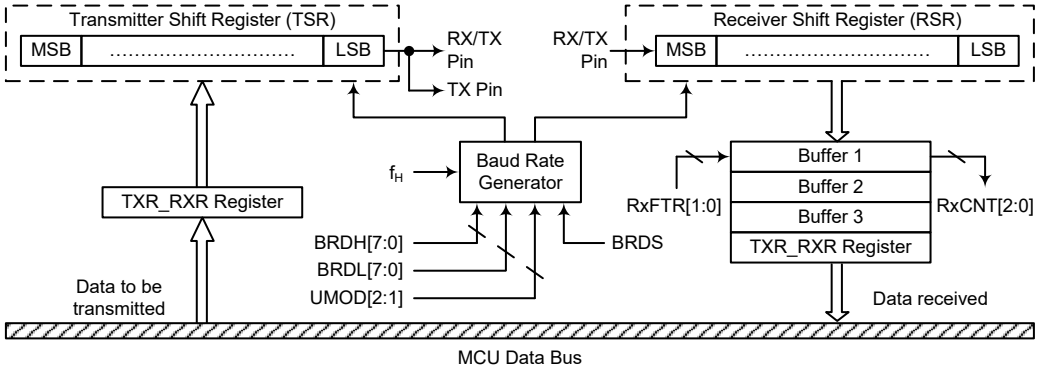
内置的 UART 功能包含以下特性：

- 全双工或半双工 (单线通信模式) 通用异步接收器 / 发送器
- 8 位或 9 位传输格式
- 奇校验、偶校验、Mark 校验、Space 校验或无校验
- 接收器可配置 1 位或 2 位停止位
- 发送器固定为 2 位停止位
- 16 位预分频的波特率发生器
- 奇偶、帧、噪声和溢出检测
- 支持地址检测中断 (最后一位 = 1)
- 独立的发送和接收使能
- 4-byte FIFO 接收数据缓冲器
- 1-byte FIFO 发送数据缓冲器
- RX/TX 引脚唤醒功能
- 发送和接收中断

- 中断可由下列条件触发：
  - ♦ 发送器为空
  - ♦ 发送器空闲
  - ♦ 接收器达到 FIFO 触发字节数
  - ♦ 接收器溢出
  - ♦ 地址检测



UART 数据传输方框图 – SWM=0



UART 数据传输方框图 – SWM=1

UART 外部引脚

内部 UART 有两个外部引脚 TX 和 RX/TX，可与外部串行接口进行通信。TX 和 RX/TX 与 I/O 口或其它功能共用引脚。在使用 UART 功能前，应先通过相应的引脚共用功能选择寄存器，选择 TX 和 RX/TX 引脚功能。当 UARTEN 和 TXEN/RXEN 位置高时，将自动设置这些 I/O 脚作为发送输出和接收输入。此时，用作发送输出的引脚其内部上拉电阻会被除能，而用作接收输入的引脚其内部上拉电阻由相应的上拉电阻控制位控制。当 UARTEN、TXEN 或 RXEN 位清零除能 TX 或 RX/TX 引脚功能后，TX 或 RX/TX 引脚将处于浮空状态。这时 TX 或 RX/TX 引脚是否连接内部上拉电阻是由相应的 I/O 上拉电阻控制位决定的。

UART 单线模式

UART 功能支持单线模式通信，通过 UCR3 寄存器中的 SWM 位选择。当设置该位为高，UART 将工作在单线模式。在单线模式下，单个 RX/TX 引脚通过相关控制位的不同设置即可完成数据的发送与接收。设置 RXEN 位为高，RX/TX

引脚用作接收引脚。将 RXEN 位清零，同时设置 TXEN 位为高，RX/TX 引脚用作发送引脚。

在单线模式下建议不要将 RXEN 位和 TXEN 位同时设置为高。若 RXEN 位和 TXEN 位同时为高，RXEN 位具有更高的优先级，此时 UART 为接收器状态。

需特别注意的是，UART 章节所有内容是基于 UART 全双工通信来对 UART 功能进行描述，相关的说明除引脚的使用外，对半双工通信 (单线模式) 同样适用。在理解单线模式通信时，全双工通信中使用的 TX 引脚需取代为 RX/TX 引脚。

在单线模式下，通过合理的软件配置，数据也可以在 TX 引脚发送。因此数据可通过 RX/TX 和 TX 引脚输出。

UART 数据传输方案

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

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

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

UART 状态和控制寄存器

与 UART 功能相关的有 9 个寄存器。UCR3 寄存器中的 SWM 位用于使能 / 除能 UART 单线模式。其它包括控制 UART 模块整体功能的 USR、UCR1、UCR2、UFCR 和 RxCNT 寄存器，控制波特率的 BRDH 和 BRDL 寄存器，管理发送和接收数据的数据寄存器 TXR_RXR。

寄存器名称	位							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	STOPS	ADDEN	WAKE	RIE	TIIE	TEIE
UCR3	—	—	—	—	—	—	—	SWM
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRDH	D7	D6	D5	D4	D3	D2	D1	D0
BRDL	D7	D6	D5	D4	D3	D2	D1	D0
UFCR	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
RxCNT	—	—	—	—	—	D2	D1	D0

UART 寄存器列表

- **USR 寄存器**

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

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

Bit 7 **PERR**：奇偶校验出错标志位
0：奇偶校验正确
1：奇偶校验出错

PERR 是奇偶校验出错标志位。若 PERR=0，奇偶校验正确；若 PERR=1，接收到的数据奇偶校验出错。只有使能了奇偶校验，选择了校验类型 (奇校验、偶校验、Mark 校验或 Space 校验)，此位才有效。可使用软件清除该标志位，即先读取 USR 寄存器再读 TXR_RXR 寄存器来清除此位。

Bit 6 **NF**：噪声干扰标志位
0：没有受到噪声干扰
1：受到噪声干扰

NF 是噪声干扰标志位。若 NF=0，没有受到噪声干扰；若 NF=1，UART 接收数据时受到噪声干扰。它与 RXIF 在同周期内置位，但不会与溢出标志位同时置位。可使用软件清除该标志位，即先读取 USR 寄存器再读 TXR_RXR 寄存器将清除此标志位。

Bit 5 **FERR**：帧错误标志位
0：无帧错误发生
1：有帧错误发生

FERR 是帧错误标志位。若 FERR=0，没有帧错误发生；若 FERR=1，当前的数据发生了帧错误。可使用软件清除该标志位，即先读取 USR 寄存器再读 TXR_RXR 寄存器来清除此位。

Bit 4 **OERR**：溢出错误标志位
0：无溢出错误发生
1：有溢出错误发生

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

Bit 3 **RIDLE**：接收状态标志位
0：正在接收数据
1：接收器空闲

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

Bit 2 **RXIF**：接收寄存器状态标志位
0：TXR_RXR 寄存器为空
1：TXR_RXR 寄存器含有有效数据且达到接收器 FIFO 触发等级

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

Bit 1 **TIDLE**：数据发送完成标志位
0：数据传输中
1：无数据传输

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

Bit 0 **TXIF**：发送数据寄存器 TXR_RXR 状态位
0：数据还没有从缓冲器加载到移位寄存器中
1：数据已从缓冲器加载到移位寄存器中 (TXR_RXR 数据寄存器为空)

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

- **UCR1 寄存器**

UCR1、UCR2 和 UCR3 是 UART 的三个控制寄存器，用来定义各种 UART 功能，例如 UART 的使能与除能、奇偶校验控制、传输数据的长度和单线模式通信等等。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

“x”：未知

Bit 7 **UARTEN**：UART 功能使能位
0：UART 除能，TX 和 RX/TX 脚处于浮空状态
1：UART 使能，TX 和 RX/TX 脚作为 UART 功能引脚

此位为 UART 的使能位。UARTEN=0，UART 除能，RX/TX 和 TX 处于浮空状态；UARTEN=1，UART 使能，TX 和 RX/TX 将分别由 SWM 模式选择位以及 TXEN 和 RXEN 使能位控制。

当 UART 被除能将清除缓冲器，所有缓冲器中的数据将被忽略，另外波特率计数器、错误和状态标志位被复位，TXEN、RXEN、TXBRK、RXIF、OERR、FERR、PERR 和 NF 以及 RxCNT 清零，而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2、UCR3、UFCR、BRDH 和 BRDL 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零，所有发送和接收将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

Bit 6 **BNO**：数据传输位数选择位
0：8-bit 传输数据
1：9-bit 传输数据

BNO 是数据传输位数选择位。BNO=1，传输数据为 9 位；BNO=0，传输数据为 8 位。若选择了 9 位数据传输格式，RX8 和 TX8 将分别存储接收和发送数据的第 9 位。

需要注意的是，若 BNO=1，奇偶校验使能时，数据的第 9 位为奇偶校验位，不会传送到 RX8。若 BNO=0，奇偶校验使能时，数据的第 8 位为奇偶校验位，不会传送到 TXRX7。

Bit 5 **PREN**：奇偶校验使能位
0：奇偶校验除能
1：奇偶校验使能

此位为奇偶校验使能位。PREN=1，使能奇偶校验；PREN=0，除能奇偶校验。

Bit 4~3 **PRT1~PRT0**：奇偶校验类型选择位
00：偶校验
01：奇校验
10：Mark 校验
11：Space 校验

这两位为奇偶校验类型选择位。当 PRT[1:0]=00，偶校验；当 PRT[1:0]=01，奇校验；当 PRT[1:0]=10，Mark 校验，校验位为 1；当 PRT[1:0]=11，Space 校验，校验位为 0。

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

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

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

- **UCR2 寄存器**

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

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

Bit 7 **TXEN**：UART 发送使能位
0：UART 发送除能
1：UART 发送使能
此位为发送使能位。TXEN=0，发送将被除能，发送器立刻停止工作。另外发送缓冲器将被复位，此时 TX 引脚将处于浮空状态。若 TXEN=1 且 UARTEN=1，则发送将被使能，TX 引脚将由 UART 来控制。在数据传输时清除 TXEN 将中止数据发送且复位发送器，此时 TX 引脚将处于浮空状态。

Bit 6 **RXEN**：UART 接收使能位
0：UART 接收除能
1：UART 接收使能
此位为接收使能位。RXEN=0，接收将被除能，接收器立刻停止工作。另外接收缓冲器将被复位，此时 RX/TX 引脚将处于浮空状态。若 RXEN=1 且 UARTEN=1，则接收将被使能，RX/TX 引脚将由 UART 来控制。在数据传输时清除 RXEN 将中止数据接收且复位接收器，此时 RX/TX 引脚将处于浮空状态。

Bit 5 **STOPS**：接收器停止位的长度选择位
0：有一位停止位
1：有两位停止位
此位用来设置接收器停止位的长度。STOPS=1，有两位停止位；STOPS=0，只有一位停止位。发送器固定使用两位停止位。

Bit 4 **ADDEN**：地址检测使能位
0：地址检测除能
1：地址检测使能
此位为地址检测使能和除能位。ADDEN=1，地址检测使能，此时数据的第 8 位即 TXRX7(BNO=0) 或第 9 位即 RX8 (BNO=1) 为高，那么接到的是地址而非数据。若相应的中断使能且接收到的值最高位为 1，那么中断请求标志将会被置位，若地址检测功能使能且最高位为 0，那么将不会产生中断且收到的数据也会被忽略。

Bit 3 **WAKE**：RX/TX 脚下降沿唤醒 UART 功能使能位
0：RX/TX 脚下降沿唤醒 UART 功能除能
1：RX/TX 脚下降沿唤醒 UART 功能使能

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

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

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

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

- **UCR3 寄存器**

UCR3 寄存器用于使能 UART 单线模式通信。顾名思义，在单线模式下只需要使用一条线，RX/TX，在 UCR2 寄存器中的 RXEN 和 TXEN 位控制即可完成通信。

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

Bit 7~1 未定义，读为“0”

Bit 0 **SWM**：单线模式使能控制
0：除能，RX/TX 引脚仅用作 UART 接收功能
1：使能，RX/TX 引脚在 RXEN 和 TXEN 位控制下可用作接收或发送功能
需注意的是，单线模式使能时，若将 RXEN 和 TXEN 位同时设置为高，RX/TX 引脚用作接收功能。

- **TXR_RXR 寄存器**

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

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

“x”：未知

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

● **BRDH 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**：波特率分频器高字节

波特率分频器 BRD (BRDH/BRDL) 用来定义 UART 时钟的分频比率。

波特率 = f_H/(BRD+UMOD/8)

BRD=16~65535 或 8~65535，取决于 BRDS

注：1. 当 BRDS=0 时，BRD 值不应小于 16；当 BRDS=1 时，BRD 值不应小于 8，否则可能发生错误。

2. 必须先对 BRDL 写值，再对 BRDH 写值，否则可能发生错误。

3. 不可在数据传输过程中修改 BRDH 寄存器。

● **BRDL 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**：波特率分频器低字节

波特率分频器 BRD (BRDH/BRDL) 用来定义 UART 时钟的分频比率。

波特率 = f_H/(BRD+UMOD/8)

BRD=16~65535 或 8~65535，取决于 BRDS

注：1. 当 BRDS=0 时，BRD 值不应小于 16；当 BRDS=1 时，BRD 值不应小于 8，否则可能发生错误。

2. 必须先对 BRDL 写值，再对 BRDH 写值，否则可能发生错误。

3. 不可在数据传输过程中修改 BRDL 寄存器。

● **UFCR 寄存器**

UFCR 寄存器是 FIFO 控制寄存器，用于 UART 调制控制、BRD 范围选择、RXIF 和中断的触发字节数选择。

Bit	7	6	5	4	3	2	1	0
Name	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 未定义，读为“0”

Bit 5~3 **UMOD2~UMOD0**：UART 调制控制位

该调制控制位用于校正接收到的或发送出的 UART 信号的波特率。这几位决定是否应该在一个 UART 位时间内加入额外的 UART 时钟周期。每个 UART 位时间 UMOD2~UMOD0 将被加入到内部累加器中。直到进位到 bit 3，对应的 UART 位时间增加一个 UART 时钟周期。

Bit 2 **BRDS**：BRD 范围选择

0：BRD=16~65535

1：BRD=8~65535

BRDS 位用于控制 UART 位时间内的采样点。若 BRDS=0，则在一个 UART 位时间内采样点为 BRD/2、BRD/2+1×f_H 和 BRD/2+2×f_H。若 BRDS=1，则在一个 UART 位时间内采样点为 BRD/2-1×f_H、BRD/2、BRD/2+2×f_H。

需注意，不可在数据传输过程中修改 BRDS 位。

Bit 1~0 **RxFTR1~RxFTR0**：接收器 FIFO 触发等级 (字节数)
00：接收器 FIFO 中有 4 个字节
01：接收器 FIFO 中有 1 个以上字节
10：接收器 FIFO 中有 2 个以上字节
11：接收器 FIFO 中有 3 个以上字节
对于接收器，这几位用于定义接收器 FIFO 中接收到的数据字节数，达到设定字节数将触发 RXIF 位置高，若 RIE 位使能，还将产生一个中断。为防止 OERR 置位，用户可配置接收器 FIFO 达 2 个字节时触发中断，避免超出 4 个字节无法由程序及时处理的溢出状态。复位后接收器 FIFO 为空。

- **RxCNT 寄存器**

RxCNT 寄存器是一个计数器，用来表示接收器 FIFO 中已接收到但还未被 MCU 读取的数据字节数。这个寄存器是只读的。

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

Bit 7~3 未定义，读为“0”

Bit 2~0 **D2~D0**：接收器 FIFO 计数器
RxCNT 寄存器是一个计数器，用来表示未被 MCU 读取的接收器 FIFO 中接收的数据字节数。当接收器 FIFO 接收到一个字节数据时，RxCNT 将自动加一；当 MCU 从接收器 FIFO 中读取一个字节数据时，RxCNT 将自动减一。如果接收器 FIFO 中有 4 个字节的数据，那么第 5 个数据将保存在移位寄存器中。如果有第 6 个数据，第 6 个数据将保存在移位寄存器中。但是 RxCNT 的值仍然是 4。当复位发生或 UARTE=1 时，RxCNT 将被清零。这个寄存器是只读的。

波特率发生器

UART 自身具有一个波特率发生器，通过它可以设定数据传输速率。波特率是由一个独立的内部 16 位计数器产生，它由 BRDH/BRDL 寄存器和 UART 调制控制位 UMOD2~UMOD0 来控制。为防止接收器波特率出现频率累积误差，建议使用 2 位停止位，使每一帧数据完成接收后重新同步。如果由 UART 时钟 f_H 生成所需的波特率 BR，则：

f_H/BR = 整数部分 + 小数部分

整数部分载入 BRD (BRDH/BRDL)，小数部分乘以 8，四舍五入后载入 UMOD 位域，如下：

BRD=TRUNC(f_H/BR)

UMOD=ROUND[MOD(f_H/BR)×8]

因此，实际波特率如下：

波特率 = f_H/[BRD+(UMOD/8)]

波特率和误差计算

若选用 4MHz 时钟频率且期望的波特率为 230400，计算 BRDH/BRDL 寄存器的值，实际波特率和误差。

根据上述公式，BRD=TRUNC(f_H/BR)=TRUNC(17.36111)=17

UMOD=ROUND[MOD(f_H/BR)×8]=ROUND(0.36111×8)=ROUND(2.88888)=3

实际波特率 = f_H/[BRD+(UMOD/8)]=230215.83

因此，误差 = (230215.83 - 230400)/230400=-0.08%

调制控制范例

为了得到 UART 调制控制位 UMOD2~UMOD0 的最佳拟合位序列，可以采用以下算法：首先，将理论除法因子的小数部分乘以 8。然后将结果四舍五入，并写入 UMOD2~UMOD0 位。每个 UART 位时间 UMOD2~UMOD0 将被加入到内部累加器中。直到进位到 bit 3，对应的 UART 位时间增加一个 UART 时钟周期。下面以之前计算的小数 0.36111 为例来做说明：UMOD[2:0]=ROUND(0.36111×8)=011b。

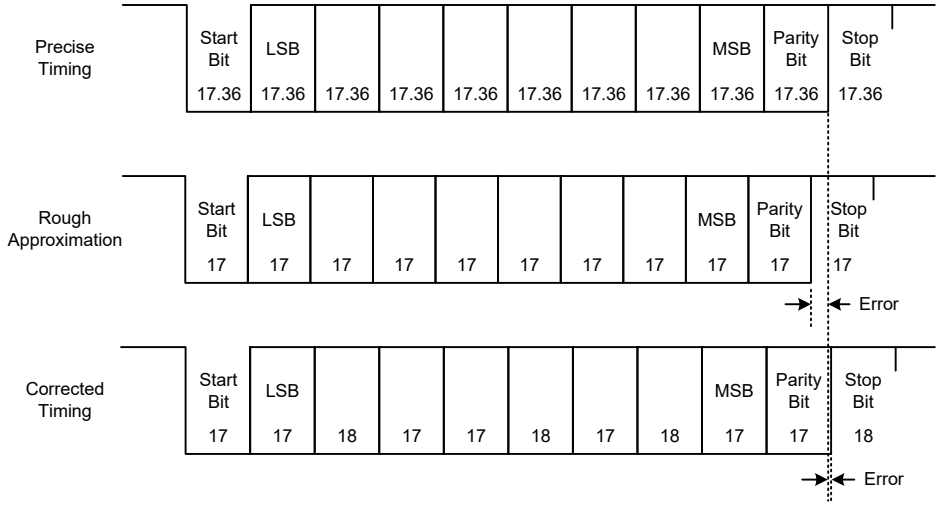
小数叠加	进位到 Bit 3	UART 位时间序列	额外的 UART 时钟周期
0000b+0011b=0011b	No	起始位	No
0011b+0011b=0110b	No	D0	No
0110b+0011b=1001b	Yes	D1	Yes
1001b+0011b=1100b	No	D2	No
1100b+0011b=1111b	No	D3	No
1111b+0011b=0010b	Yes	D4	Yes
0010b+0011b=0101b	No	D5	No
0101b+0011b=1000b	Yes	D6	Yes
1000b+0011b=1011b	No	D7	No
1011b+0011b=1110b	No	校验位	No
1110b+0011b=0001b	Yes	停止位	Yes

波特率矫正范例

下图为一个使用 UART 时钟 f_H 生成的波特率为 230400 的示例，数据格式是：8 位数据位，奇偶校验使能，无地址位，2 位停止位。

下图显示了三个不同的帧：

- 最上帧为准确帧，位长为 17.36 个 f_H 时钟周期 (400000/230400=17.36)。
- 中间帧采用粗略估计，位长为 17 个 f_H 时钟周期。
- 最下帧显示的是校正后的帧，采用 UART 调制控制位 UMOD2~UMOD0 的最佳拟合算法。



UART 的设置与控制

UART 采用标准的不归零码传输数据，这种方法通常被称为 NRZ 法。它由 1 位起始位，8 位或 9 位数据位和 1 位或者 2 位停止位组成。奇偶校验是由硬件自动完成的，可设置成奇校验、偶校验、Mark 校验、Space 校验或无校验。常用的数据传输格式由 8 位数据位，1 位停止位，无校验组成，用 8、N、1 表示，它是系统上电的默认格式。数据位数和奇偶校验由 UCR1 寄存器的 BNO、PRT1~PRT0 和 PREN 设定。发送器固定使用 2 位停止位，接收器停止位数由 STOPS 设定。用于数据发送和接收的波特率由一个内部的 8 位波特率发送器产生，数据传输时低位在前高位在后。尽管 UART 发送器和接收器在功能上相互独立，但它们使用相同的数据传输格式和波特率，在任何情况下，停止位是必须的。

UART 的使能和除能

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

UARTEN 清零将除能 TX 和 RX/TX，通过设置相关引脚共用控制位，这两个引脚可用作普通 I/O 口或其它引脚共用功能。当 UART 被除能时将清空缓冲器，所有缓冲器中的数据将被忽略，另外一些使能控制、错误标志和状态标志将被复位，如 TXEN、RXEN、TXBRK、RXIF、OERR、FERR、PERR 和 NF 以及 RxCNT 寄存器清零，而 TIDLE、TXIF 和 RIDLE 置位，UCR1、UCR2、UCR3、UFCR、BRDH 和 BRDL 寄存器中的其它位保持不变。若 UART 工作时 UARTEN 清零，所有发送和接收将停止，模块也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

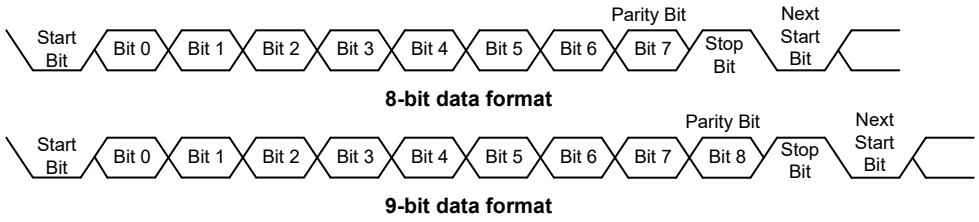
数据位、停止位位数以及奇偶校验的选择

数据传输格式由数据长度、是否校验、校验类型、地址位以及停止位长度组成。它们都是由 UCR1 和 UCR2 寄存器的各个位控制的。BNO 决定数据传输是 8 位还是 9 位；PRT1~PRT0 决定校验类型；PREN 决定是否选择奇偶校验；而 STOPS 决定接收器选用 1 位还是 2 位停止位，发送器则固定使用 2 位停止位。下表列出了各种数据传输格式。若地址检测功能使能，地址位，即数据字节的最高位，用来确定此帧是地址还是数据。停止位的长度和数据位的长度无关，且只有接收器需设置停止位长度。发送器固定使用 2 位停止位。

起始位	数据位	地址位	校验位	停止位
8 位数据位				
1	8	0	0	1 或 2
1	7	0	1	1 或 2
1	7	1	0	1 或 2
9 位数据位				
1	9	0	0	1 或 2
1	8	0	1	1 或 2
1	8	1	0	1 或 2

发送和接收数据格式

下面是传输 8 位和 9 位数据的波形。



UART 发送器

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

发送数据

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

启动数据发送的步骤如下：

- 正确地设置 BNO、PRT1~PRT0、PREN 位以确定数据长度和校验类型，而停止位固定为 2 位。
- 设置 BRDH/BRDL 寄存器以及 UMOD2~UMOD0 位，选择期望的波特率。
- 置高 TXEN，使能 UART 发送器且使 TX 作为 UART 的发送端。
- 读取 USR 寄存器，然后将待发数据写入 TXR_RXR 寄存器。注意，此步骤会清除 TXIF 标志位。

如果要发送多个数据只需重复上一步骤。

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

1. 读取 USR 寄存器
2. 写 TXR_RXR 寄存器

只读标志位 TXIF 由 UART 硬件置位。若 TXIF=1，TXR_RXR 寄存器为空，其它数据可以写入而不会覆盖之前的数据。若 TEIE=1，TXIF 标志位会产生中断。

在数据传输时，写 TXR_RXR 指令会将待发数据暂存在 TXR_RXR 寄存器中，当前数据发送完毕后，待发数据被加载到发送移位寄存器中。当发送器空闲时，写 TXR_RXR 指令会将数据直接加载到 TSR 寄存器中，数据传输立刻开始且 TXIF 置位。当发送完停止位或暂停帧后，表示一帧数据已发送完毕，此时 TIDLE 位将被置位。

可以通过以下步骤来清除 TIDLE：

1. 读取 USR 寄存器
2. 写 TXR_RXR 寄存器

清除 TXIF 和 TIDLE 软件执行次序相同。

发送暂停字

若 TXBRK=1 保持时间超过 [(BRD+1)×t_H] 且 TIDLE=1，下一帧将会发送暂停字。它是由一个起始位、13×N (N=1, 2……) 位逻辑 0 组成。置位 TXBRK 将会发送暂停字，而清除 TXBRK 将产生停止位，传输暂停字不会产生中断。需要注意的是，暂停字至少 13 位宽。若 TXBRK 持续为高，那么发送器会一直发送暂停字；当应用程序将 TXBRK 清零后，发送器结束最后一帧暂停字的发送后接着发送两位停止位。最后一帧暂停字的结尾自动为高电平，以确保下一帧数据起始位的检测。

UART 接收器

UART 接收器支持 8 位或者 9 位数据接收。若 BNO=1，数据长度为 9 位，而最高位 MSB 存放在 UCR1 寄存器的 RX8 中。接收器的核心是串行移位寄存器 RSR。RX/TX 引脚上的数据送入数据恢复器中，它在 16 倍波特率的频率下工作，而串行移位器工作在正常波特率下。当在 RX/TX 引脚上检测到停止位，若 TXR_RXR 寄存器为空，数据从 RSR 寄存器中加载到 TXR_RXR 寄存器。RX/TX 引脚上的每一位数据会被采样三次以判断其逻辑状态。RSR 不像其它寄存器一样映射在数据存储器，所以应用程序不能对其进行读写操作。

接收数据

当 UART 接收数据时，数据低位在前高位在后，连续地从 RX/TX 引脚进入移位寄存器。TXR_RXR 寄存器在内部总线和接收移位寄存器间形成一个缓冲。TXR_RXR 寄存器是一个四层的 FIFO 缓冲器，它能保存四帧数据的同时接收第五帧数据，应用程序必须保证在接收完第五帧前读取 TXR_RXR 寄存器，否则忽略第五帧数据并且发生溢出错误。如需连续多帧数据传输，强烈建议接收器使用 2 位停止位，以避免接收器波特率频率累积误差造成的接收错误。

启动数据接收的步骤如下：

- 正确地设置 BNO、PRT1~PRT0、PREN 和 STOPS 位，以确定数据长度、校验类型和停止位个数。
- 设置 BRDH/BRDL 寄存器和 UMOD2~UMOD0 位，选择期望的波特率。
- 置高 RXEN，使能 UART 接收器且使 RX/TX 引脚作为 UART 的接收端。

此时接收器被使能并检测起始位。

接收数据将会发生如下事件：

- 当 TXR_RXR 寄存器中包含有效数据时，USR 寄存器中的 RXIF 位将会置位，可通过轮询 RxCNT 寄存器的值来检查有效数据字节数。
- 当数据从 RSR 寄存器加载到 TXR_RXR 寄存器中，并且达到接收器 FIFO 触发字节数。若 RIE=1，将产生中断。
- 若接收器检测到帧错误、噪声干扰错误、奇偶出错或溢出错误，那么相应的错误标志位置位。

可以通过如下步骤来清除 RXIF：

1. 读取 USR 寄存器
2. 读取 TXR_RXR 寄存器

接收暂停字

UART 接收任何暂停字都会当作帧错误处理。接收器只根据 BNO 位的设置外加一个或两个停止位来确定一帧数据的长度。若暂停字位数大于 BNO 位指定的长度外加一个或两个停止位，接收器认为接收已完毕，RXIF 和 FERR 置位，TXR_RXR 寄存器清 0，若相应的中断允许且 RIDLE 为高将会产生中断。暂停字只会被认为包含信息 0 且会置位 FERR 标志位。如果检测到较长的暂停信号，接收器会将此信号视为包含一个起始位、数据位和无效的停止位的数据帧并且置位 FERR 标志位。在下个开始位到来之前，接收器必须等待一个或两个有效的停止位。接收器不会假定线上的暂停信号是下一个开始位。暂停字将会加载到缓冲器中，在接收到停止位前不会再接收数据，没有检测到停止位也会置位只读标志位 RIDLE。

UART 接收到暂停字会产生以下事件：

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

空闲状态

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

接收中断

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

如有其他子程序被调用且执行时间大于 UART 接收五帧数据的时间，若子程序执行期间无法及时读取 UART 接收数据，则需提前将 RXEN 清零，暂停接收数据；若子程序执行期间无法及时响应 UART 中断处理溢出错误，则需确保执行子程序时 EMI 与 RXEN 都是关闭的，子程序执行完成后，再开启 EMI 与 RXEN，继续接收 UART 数据。

接收错误处理

UART 会产生几种接收错误，下面部分将描述各错误以及怎样处理。

溢出 – OERR 标志

TXR_RXR 寄存器是一个四层的 FIFO 缓冲器，它能保存四帧数据的同时接收第五帧数据，应用程序必须保证在接收完第五帧前读取 TXR_RXR 寄存器，否则发生溢出错误。

产生溢出错误时将会发生以下事件：

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

当 OERR 位被设为“1”时，用户需要立即读取五帧数据 (四层接收缓冲器与移位寄存器数据)，避免无法预期的错误发生，例如 UART 无法再接收数据。如果发生上述错误，可将 RXEN 清为“0”再设为“1”，重新接收数据。

若要将 OERR 清零，先读取 USR 寄存器再读取 TXR_RXR 寄存器即可。

噪声干扰 – NF 标志

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

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

先读取 USR 寄存器再读取 TXR_RXR 寄存器可将 NF 清零。

帧错误 – FERR 标志

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

奇偶校验错误 – PERR 标志

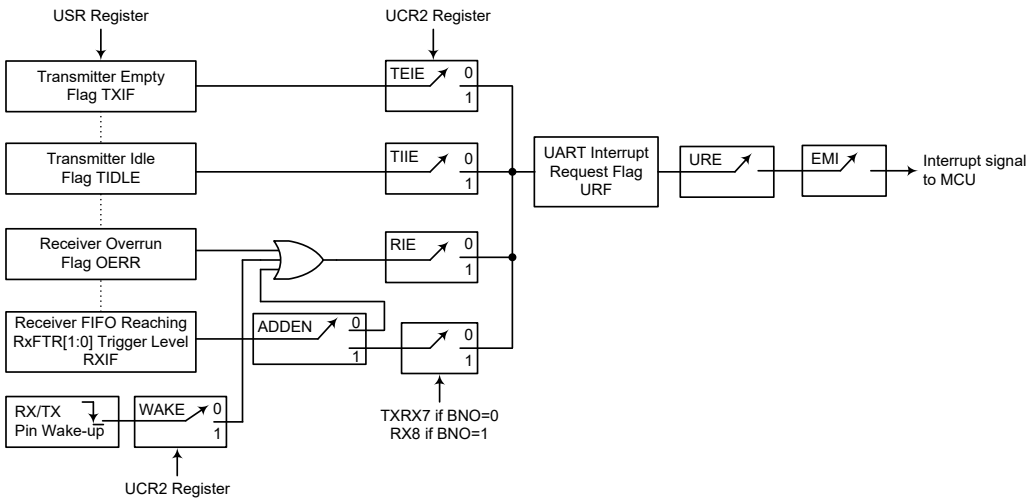
若接收到数据出现奇偶校验错误，USR 寄存器中只读标志 PERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。此标志位同接收的数据分别记录在 USR 寄存器和 TXR_RXR 中，此标志位可被任何复位清零。注意，在读取相应的数据之前必须先访问 USR 寄存器中的 FERR 和 PERR 错误标志位。

UART 中断结构

几个独立的 UART 条件可以产生一个 UART 中断。当所有条件满足时，会产生一个低脉冲信号。发送寄存器为空、发送器空闲、接收器 FIFO 达到触发中断的字节数、溢出、地址检测和 RX/TX 引脚唤醒都会产生中断。若总中断使能位及相应的中断控制位使能且堆栈未满，程序将会跳转到相应的中断向量执行中断服务程序，而后再返回主程序。其中四种情况，若其 UCR2 寄存器中相应中断允许位被置位，则 USR 寄存器中对应中断标志位将产生 UART 中断。发送器相关的两个中断情况有各自对应的中断允许位，而接收器相关的两个中断情况共用一个中断允许位。这些允许位可用于禁止个别的 UART 中断源。

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

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



UART 中断框图

地址检测模式

置位 UCR2 寄存器中的 ADDEN 将启动地址检测模式。若此位为“1”，可产生接收数据有效中断，其请求标志位为 RXIF。若 ADDEN 位使能，只有在接收到数据最高位为“1”才会产生中断请求，注意 URE 和 EMI 中断使能位也要使能才会产生中断。地址的最高位为第 9 位 (BNO=1) 或第 8 位 (BNO=0)，若此位为高，则接收到的是地址而非数据。只有接收的数据的最后一位为高才会产生中断。若 ADDEN 除能，每接收到一个有效数据便会置位 RXIF，而不用考虑数据的最后一位。地址检测和奇偶校验在功能上相互排斥，若地址检测模式使能，为了确保操作正确，必须将奇偶校验使能位清零以除能奇偶校验。

ADDEN	9th Bit (BNO=1) 8th Bit (BNO=0)	产生 UART 中断
0	0	√
	1	√
1	0	×
	1	√

ADDEN 位功能

UART 暂停和唤醒

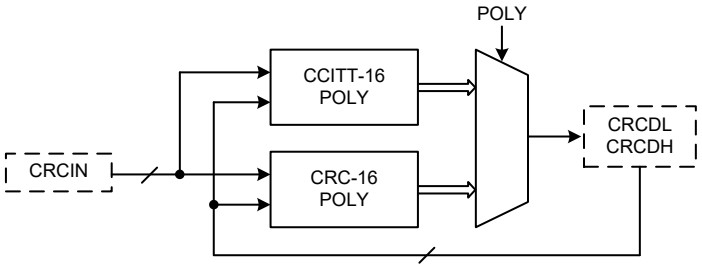
UART 时钟 f_H 关闭后 UART 将停止运行。当传送数据时 UART 时钟 f_H 关闭，发送将停止直到 UART 时钟再次使能。同样地，当接收数据时单片机进入空闲或休眠模式，数据接收也会停止。当单片机进入空闲或休眠模式，USR、UCR1、UCR2、UCR3、UFCR、RxCNT、TXR_RXR 以及 BRDH 和 BRDL 寄存器都不会受到影响。建议在单片机进入空闲或休眠模式前先确保数据发送或接收已完成。

UART 具备 RX/TX 引脚的唤醒功能，由 UCR2 寄存器中 WAKE 位控制。当单片机进入空闲或休眠模式且 UART 时钟 f_H 关闭时，若 WAKE 位、UART 使能位 UARTEN、接收器使能位 RXEN 和接收器中断使能位 RIE 都被置位，则 RX/TX 引脚的下降沿可触发产生 RX/TX 引脚唤醒 UART 的中断。唤醒后系统需延时一段时间才能正常工作，在此期间，RX/TX 引脚上的任何数据将被忽略。

若要唤醒并产生 UART 中断，除了唤醒使能控制位和接收中断使能控制位需置位外，总中断使能位 EMI 和 UART 中断使能控制位 URE 也必须置位；若这两个控制位没有被置位，那么单片机将可以被唤醒但不会产生中断。同样唤醒后系统需一定的延时才能正常工作，然后才会产生 UART 中断。

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

● **CRCDH 寄存器**

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

Bit 7~0 **D15~D8**：16-bit CRC 校验和高字节数据寄存器

● **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 CRCDH 和 CRCDL 中。

● CRC-CCITT：$X^{16}+X^{12}+X^5+1$

● CRC-16：$X^{16}+X^{15}+X^2+1$

CRC 计算

每次对 CRCIN 寄存器进行写操作，都会将存储在 CRCDH 和 CRCDL 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 CRCDH 和 CRCDL。
2. 对 8-bit 输入数据字节和 16-bit CRCSUM 高字节进行异或操作，其结果称为临时 CRCSUM。
3. 将临时 CRCSUM 值左移一位，并向最低有效位 LSB 填入“0”。

4. 检查在步骤 3 中完成移位后的临时 CRCSUM 值。

   若 MSB 为“0”，则该移位后的临时 CRCSUM 将作为新的临时 CRCSUM。否则，对步骤 3 中移位后的临时 CRCSUM 和数据“8005H”进行异或操作。该操作结果将作为新的临时 CRCSUM。

   应注意的是对于 CRC-16 多项式，用于异或操作的数据为“8005H”，而对于 CRC-CCITT 多项式用于异或操作的数据则为“1021H”。

5. 重复步骤 3 到步骤 4，直到输入数据字节的所有位都经过计算。
6. 重复步骤 2 到步骤 5，直到所有输入数据字节都经过计算。此时，最新的计算结果则为最终的 CRC 校验和 CRCSUM。

CRC 计算范例

- 向 CRCIN 寄存器写入 1 个字节的输入数据，相应的 CRC 校验和将逐个被计算，如下表所示。

CRC 数据输入 / CRC 多项式	00H	01H	02H	03H	04H	05H	06H	07H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	0000H	1021H	2042H	3063H	4084H	50A5H	60C6H	70E7H
CRC-16 ($X^{16}+X^{15}+X^2+1$)	0000H	8005H	800FH	000AH	801BH	001EH	0014H	8011H

注：在每个 CRC 输入数据写入 CRCIN 寄存器之前，CRC 校验和寄存器对 CRCDH 和 CRCDL 的初始值为“0”。

- 向 CRCIN 寄存器连续写入 4 个字节的输入数据，相应的 CRC 检验和连续列于下表。

CRC 数据输入 / CRC 多项式	CRCIN=78H→56H→34H→12H
CRC-CCITT ($X^{16}+X^{12}+X^5+1$)	(CRCDH, CRCDL)=FF9FH→BBC3H→A367H→D0FAH
CRC-16 ($X^{16}+X^{15}+X^2+1$)	(CRCDH, CRCDL)=0110H→91F1H→F2DEH→5C43H

注：在连续的 CRC 数据输入操作之前，CRC 校验和寄存器对 CRCDH 和 CRCDL 的初始值为“0”。

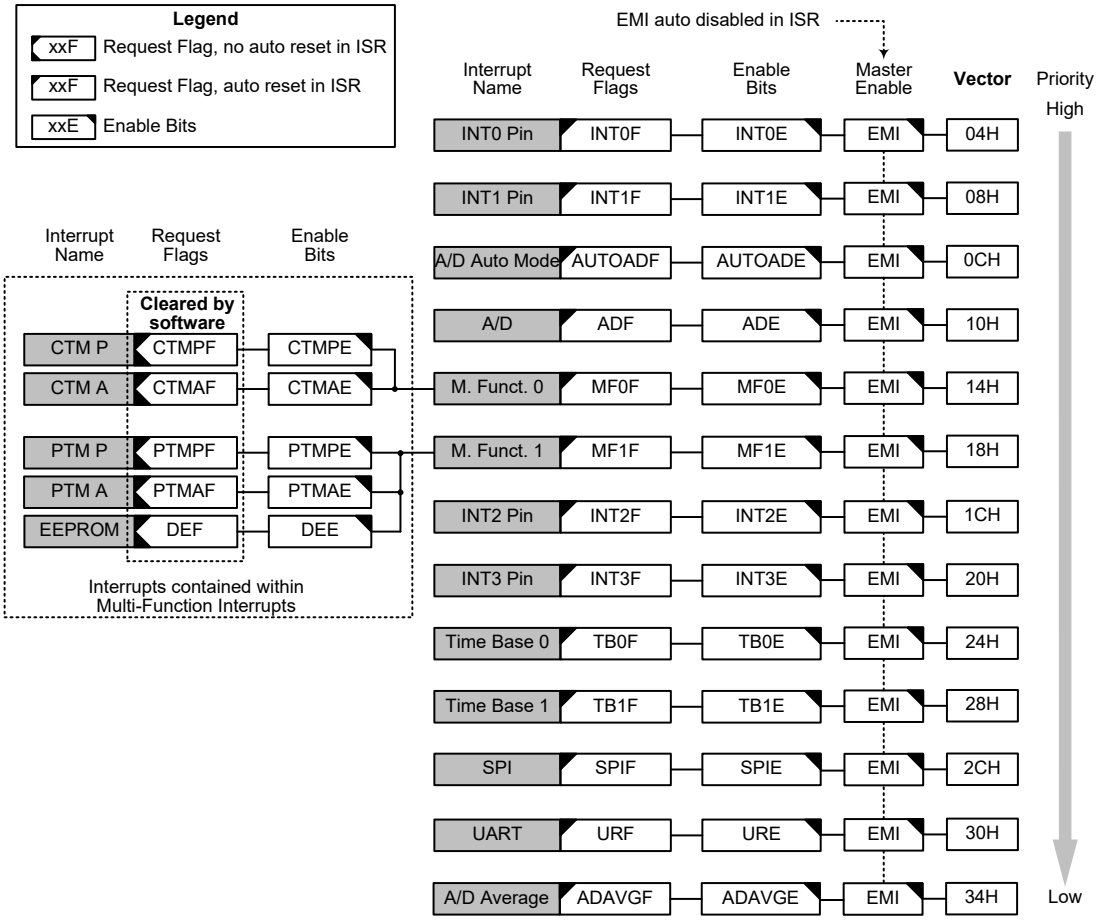
程序存储器 CRC 校验和计算范例：

1. 清除校验和寄存器对 CRCDH 和 CRCDL。
2. 通过 CRCCR 寄存器中的 POLY 位选择 CRC-CCITT 或 CRC-16 多项式作为生成多项式。
3. 执行表格读取指令，读取程序存储器数据值。
4. 将表格数据低字节写入 CRCIN 寄存器，并结合当前 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。
5. 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。
6. 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、SPI、USRT 和 A/D 转换器等等。

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，而有些中断则共用多功能中断向量。



中断结构

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	ADINT
A/D 转换器自动模式	AUTOADE	AUTOADF	AUTOADCINT
A/D 转换器平均中断	ADAVGE	ADAVGF	ADAVGINT
多功能中断	MFnE	MFnF	n=0 或 1
时基	TBnE	TBnF	n=0 或 1
SPI	SPIE	SPIF	—
UART	URE	URF	—
EEPROM	DEE	DEF	—
CTM	CTMPE	CTMPF	—
	CTMAE	CTMAF	
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	—	AUTOADF	INT1F	INT0F	AUTOADE	INT1E	INT0E	EMI
INTC1	INT2F	MF1F	MF0F	ADF	INT2E	MF1E	MF0E	ADE
INTC2	SPIF	TB1F	TB0F	INT3F	SPIE	TB1E	TB0E	INT3E
INTC3	—	—	ADAVGF	URF	—	—	ADAVGE	URE
MFI0	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE
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	—	AUTOADF	INT1F	INT0F	AUTOADE	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 **AUTOADF**：A/D 转换器自动模式中断请求标志位
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 **AUTOADE**：A/D 转换器自动模式中断控制位
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	MF1F	MF0F	ADF	INT2E	MF1E	MF0E	ADE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **INT2F**：INT2 中断请求标志位
0：无请求
1：中断请求
Bit 6 **MF1F**：多功能中断 1 请求标志位
0：无请求
1：中断请求
Bit 5 **MF0F**：多功能中断 0 请求标志位
0：无请求
1：中断请求
Bit 4 **ADF**：A/D 转换器中断请求标志位
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 **MF1E**：多功能中断 1 控制位
0：除能
1：使能

Bit 1 **MF0E**：多功能中断 0 控制位
0：除能
1：使能

Bit 0 **ADE**：A/D 转换器中断控制位
0：除能
1：使能

• INTC2 寄存器

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

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

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

Bit 4 **INT3F**：INT3 中断请求标志位
0：无请求
1：中断请求

Bit 3 **SPI**：SPI 中断控制位
0：除能
1：使能

Bit 2 **TB1E**：时基 1 中断控制位
0：除能
1：使能

Bit 1 **TB0E**：时基 0 中断控制位
0：除能
1：使能

Bit 0 **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	—	—	ADAVGF	URF	—	—	ADAVGE	URE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 未定义，读为“0”

Bit 5 **ADAVGF**：A/D 转换器平均中断请求标志位
0：无请求
1：中断请求

Bit 4 **URF**：UART 中断请求标志位
0：无请求
1：中断请求
Bit 3~2 未定义，读为“0”
Bit 1 **ADAVGE**：A/D 转换器平均中断控制位
0：除能
1：使能
Bit 0 **URE**：UART 中断控制位
0：除能
1：使能

- **MFI0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 未定义，读为“0”
Bit 5 **CTMAF**：CTM 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求
注意，当中断响应时此位必须通过应用程序清零。
Bit 4 **CTMPF**：CTM 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求
注意，当中断响应时此位必须通过应用程序清零。
Bit 3~2 未定义，读为“0”
Bit 1 **CTMAE**：CTM 比较器 A 匹配中断控制位
0：除能
1：使能
Bit 0 **CTMPE**：CTM 比较器 P 匹配中断控制位
0：除能
1：使能

- **MFI1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	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
注意，当中断响应时此位必须通过应用程序清零。
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
注意，当中断响应时此位必须通过应用程序清零。
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
注意，当中断响应时此位必须通过应用程序清零。
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。

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，执行优先级如中断结构图所示。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。

A/D 转换器自动模式中断

当 A/D 转换器自动模式中断请求标志被置位，即当写入到缓冲器的数据达到 WRNUM[4:0] 位设定的数量时，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 转换器自动模式中断使能位 AUTOADE 需先被置位。当中断使能，堆栈未满且上述情况发生时，将调用 A/D 转换器自动模式中断向量子程序。当响应中断服务子程序时，A/D 转换器自动模式中断请求标志位 AUTOADF 会自动复位且 EMI 位会被清零以除能其它中断。

A/D 转换器平均中断

当 A/D 转换器平均中断请求标志被置位，即累加平均过程完成时，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 转换器平均中断使能位 ADAVGE 需先被置位。当中断使能，堆栈未满且累加平均动作结束时，将调用 A/D 转换器平均中断向量子程序。当响应中断服务子程序时，A/D 转换器平均中断请求标志位 ADAVGF 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。

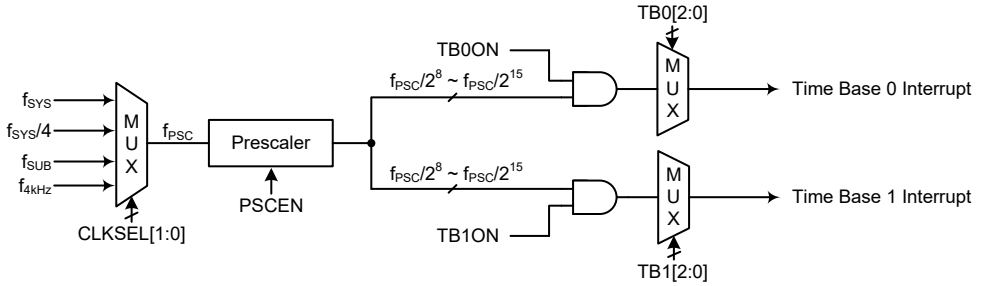
EEPROM 中断

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}、$f_{SYS}/4$、f_{SUB} 或 f_{4kHz}。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	—	—	—	—	—	PSCEN	CLKSEL1	CLKSEL0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3　未定义，读为“0”

Bit 2　**PSCEN**：预分频器时钟使能控制
0：除能
1：使能
此位为预分频器时钟使能 / 除能控制位。除能预分频时钟可减少额外的功耗。

Bit 1~0　**CLKSEL1~CLKSEL0**：分频器时钟源选择
00：f_{SYS}
01：$f_{SYS}/4$
10：f_{SUB}
11：f_{4kHz}

● **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：$2^{15}/f_{PSC}$

SPI 接口中断

当一个字节数据已由 SPI 接口接收或发送完，中断请求标志 SPIF 被置位，SPI 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 SPIE 需先被置位。当中断使能，堆栈未满且一个字节数据已被传送或接收完毕时，将调用 SPI 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 SPIF 会自动复位且 EMI 位会被清零以除能其它中断。

UART 中断

UART 中断由几种 UART 传输条件来控制。当发送器为空、发送器空闲、接收器 FIFO 达到触发等级、接收器溢出、地址检测和 RX/TX 引脚唤醒，UART 中断请求标志位 URF 被置位，UART 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和 UART 中断使能位 URE 需先被置位。当中断使

能，堆栈未满且以上任何一种情况发生时，将调用 UART 中断向量子程序。当 UART 中断响应时，UART 中断请求标志位 URF 会自动复位且 EMI 位也会被清零以除能其他中断。然而 USR 寄存器里的标志位只有在对 UART 执行特定动作是才会被清零，详细参考 UART 章节。

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。

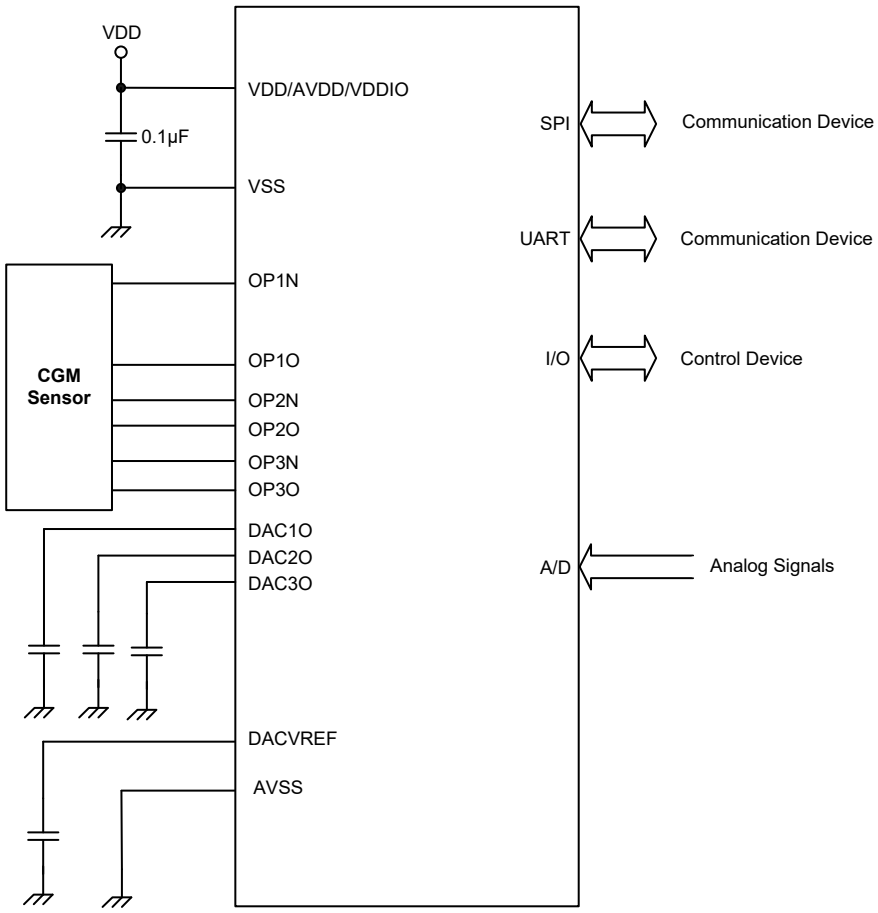
建议在中断服务子程序中不要使用“CALL 子程序”指令。中断通常发生在不可预料的情况或是需要立刻执行的某些应用。假如只剩下一层堆栈且没有控制好中断，当“CALL 子程序”在中断服务子程序中执行时，将破坏原来的控制序列。

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。

应用电路


VDD
0.1μF
VDD/AVDD/VDDIO
VSS
OP1N
OP1O
OP2N
OP2O
OP3N
OP3O
DAC1O
DAC2O
DAC3O
DACVREF
AVSS
CGM
Sensor
SPI
Communication Device
UART
Communication Device
I/O
Control Device
A/D
Analog Signals

指令集

简介

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

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

指令周期

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

数据的传送

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

算术运算

算术运算和数据处理是大部分单片机应用所必需具备的能力，在 Holtek 单片机内部的指令集中，可直接实现加与减的运算。当加法的结果超出 255 或减法的结果少于 0 时，要注意正确的处理进位和借位的问题。INC、INCA、DEC 和 DECA 指令提供了对一个指定地址的值加一或减一的功能。

逻辑和移位运算

标准逻辑运算例如 AND、OR、XOR 和 CPL 全都包含在 Holtek 单片机内部的指令集中。大多数牵涉到数据运算的指令，数据的传送必须通过累加器。在所有逻辑数据运算中，如果运算结果为零，则零标志位将被置位，另外逻辑数据运用形式还有移位指令，例如 RR、RL、RRC 和 RLC 提供了向左或向右移动一位的方法。不同的移位指令可满足不同的应用需要。移位指令常用于串行端口的程序应用，数据可从内部寄存器转移至进位标志位，而此位则可被检验，移位运算还可应用在乘法与除法的运算组成中。

分支和控制转换

程序分支是采取使用 JMP 指令跳转至指定地址或使用 CALL 指令调用子程序的形式，两者之不同在于当子程序被执行完毕后，程序必须马上返回原来的地址。这个动作是由放置在子程序里的返回指令 RET 来实现，它可使程序跳回 CALL 指令之后的地址。在 JMP 指令中，程序则只是跳到一个指定的地址而已，并不需如 CALL 指令般跳回。一个非常有用的分支指令是条件跳转，跳转条件是由数据存储器或指定位来加以决定。遵循跳转条件，程序将继续执行下一条指令或略过且跳转至接下来的指令。这些分支指令是程序走向的关键，跳转条件可能是外部开关输入，或是内部数据位的值。

位运算

提供数据存储器中单个位的运算指令是 Holtek 单片机的特性之一。这特性对于输出端口位的设置尤其有用，其中个别的位或端口的引脚可以使用“SET [m].i”或“CLR [m].i”指令来设定其为高位或低位。如果没有这特性，程序设计师必须先读入输出口的 8 位数据，处理这些数据，然后再输出正确的新数据。这种读入 - 修改 - 写出的过程现在则被位运算指令所取代。

查表运算

数据的储存通常由寄存器完成，然而当处理大量固定的数据时，它的存储量常常造成对个别存储器的不便。为了改善此问题，Holtek 单片机允许在程序存储器中建立一个表格作为数据可直接存储的区域，只需要一组简易的指令即可对数据进行查表。

其它运算

除了上述功能指令外，其它指令还包括用于省电的“HALT”指令和使程序在极端电压或电磁环境下仍能正常工作的看门狗定时器控制指令。这些指令的使用则请查阅相关的章节。

指令集概要

当要操作的数据存储器位于数据存储器 Sector 0 时，下表说明了与数据存储器存取有关的指令。

惯例

x：立即数
m：数据存储器地址
A：累加器
i：第 0~7 位
addr：程序存储器地址

助记符	说明	指令周期	影响标志位
算术运算			
ADD A,[m]	ACC 与数据存储器相加，结果放入 ACC	1	Z, C, AC, OV, SC
ADDM A,[m]	ACC 与数据存储器相加，结果放入数据存储器	1 注	Z, C, AC, OV, SC
ADD A, x	ACC 与立即数相加，结果放入 ACC	1	Z, C, AC, OV, SC
ADC A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	1	Z, C, AC, OV, SC
ADCM A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	1 注	Z, C, AC, OV, SC
SUB A, x	ACC 与立即数相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SUB A,[m]	ACC 与数据存储器相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SUBM A,[m]	ACC 与数据存储器相减，结果放入数据存储器	1 注	Z, C, AC, OV, SC, CZ
SBC A, x	ACC 与立即数、进位标志相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SBC A,[m]	ACC 与数据存储器、进位标志相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	1 注	Z, C, AC, OV, SC, CZ
DAA [m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	1 注	C
逻辑运算			
AND A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	1	Z
OR A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	1	Z
XOR A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	1	Z
ANDM A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	1 注	Z
ORM A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	1 注	Z
XORM A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	1 注	Z
AND A, x	ACC 与立即数做“与”运算，结果放入 ACC	1	Z
OR A, x	ACC 与立即数做“或”运算，结果放入 ACC	1	Z
XOR A, x	ACC 与立即数做“异或”运算，结果放入 ACC	1	Z
CPL [m]	对数据存储器取反，结果放入数据存储器	1 注	Z
CPLA [m]	对数据存储器取反，结果放入 ACC	1	Z
递增和递减			
INCA [m]	递增数据存储器，结果放入 ACC	1	Z
INC [m]	递增数据存储器，结果放入数据存储器	1 注	Z
DECA [m]	递减数据存储器，结果放入 ACC	1	Z
DEC [m]	递减数据存储器，结果放入数据存储器	1 注	Z

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

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

注：1. 对跳转指令而言，如果比较的结果牵涉到跳转即需 2 个周期，如果没有发生跳转，则只需一个周期。
2. 任何指令若要改变 PCL 的内容将需要 2 个周期来执行。

扩展指令集

扩展指令用来提供更大范围的数据存储器寻址。当被存取的数据存储器位于 Sector 0 之外的任何数据存储器 Sector，扩展指令可直接存取数据存储器而无需使用间接寻址，此举不仅可节省 Flash 存储器空间的使用，同时可提高 CPU 执行效率。

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

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

注：1. 对扩展跳转指令而言，如果比较的结果牵涉到跳转即需 3 个周期，如果没有发生跳转，则只需两个周期。

2. 任何扩展指令若要改变 PCL 的内容将需要 3 个周期来执行。

指令定义

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

RETI	Return from interrupt
指令说明	将堆栈寄存器中的程序计数器值恢复且中断功能通过设置 EMI 位重新使能。EMI 是控制中断使能的主控制位。如果在执行 RETI 指令之前还有中断未被响应，则这个中断将在返回主程序之前被响应。
功能表示	Program Counter ←Stack EMI ← 1
影响标志位	无
RL [m]	Rotate Data Memory left
指令说明	将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。
功能表示	[m].(i+1) ← [m].i (i=0~6) [m].0 ← [m].7
影响标志位	无
RLA [m]	Rotate Data Memory left with result in ACC
指令说明	将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位，结果送到累加器，而指定数据存储器的内容保持不变。
功能表示	ACC.(i+1) ← [m].i (i=0~6) ACC.0 ←[m].7
影响标志位	无
RLC [m]	Rotate Data Memory Left through Carry
指令说明	将指定数据存储器的内容连同进位标志左移 1 位，第 7 位取代进位标志且原本的进位标志移到第 0 位。
功能表示	[m].(i+1) ← [m].i (i=0~6) [m].0 ← C C ← [m].7
影响标志位	C
RLC A [m]	Rotate Data Memory left through Carry with result in ACC
指令说明	将指定数据存储器的内容连同进位标志左移 1 位，第 7 位取代进位标志且原本的进位标志移到第 0 位，移位结果送回累加器，但是指定数据寄存器的内容保持不变。
功能表示	ACC.(i+1) ← [m].i (i=0~6) ACC.0 ← C C ← [m].7
影响标志位	C

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

SUB A, [m] Subtract Data Memory from ACC

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

功能表示 ACC ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

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

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

功能表示 [m] ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

SUB A, x Subtract immediate Data from ACC

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

功能表示 ACC ← ACC – x

影响标志位 OV、Z、AC、C、SC、CZ

SWAP [m] Swap nibbles of Data Memory

指令说明 将指定数据存储器的低 4 位和高 4 位互相交换。

功能表示 [m].3~[m].0 ↔ [m].7~[m].4

影响标志位 无

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

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

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

影响标志位 无

SZ [m] Skip if Data Memory is 0

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

功能表示 如果 [m]=0，跳过下一条指令执行

影响标志位 无

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

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

功能表示 ACC ← [m]，如果 [m]=0，跳过下一条指令执行

影响标志位 无

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

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

功能表示 如果 [m].i=0，跳过下一条指令执行

影响标志位 无

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

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

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

影响标志位 无

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

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

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

影响标志位 无

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

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

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

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

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

扩展指令定义

扩展指令被用来直接存取存储在任何数据存储器 Sector 中的数据。

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

LRR [m] Rotate Data Memory right

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

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

影响标志位 无

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

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

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

影响标志位 无

LRRC [m] Rotate Data Memory right through Carry

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

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

影响标志位 C

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

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

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

影响标志位 C

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

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

功能表示 ACC ← ACC – [m] – $\overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

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

LSDZ [m] Skip if Decrement Data Memory is 0
指令说明 将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 [m] ← [m] – 1，如果 [m]=0 跳过下一条指令执行
影响标志位 无

LSDZA [m] Skip if decrement Data Memory is zero with result in ACC
指令说明 将指定数据存储器内容减 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果将存放到累加器，但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 ACC ← [m] – 1，如果 ACC=0 跳过下一条指令执行
影响标志位 无

LSET [m] Set Data Memory
指令说明 将指定数据存储器的每一个位置位为 1。
功能表示 [m] ← FFH
影响标志位 无

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

LSIZ [m] Skip if increment Data Memory is 0

指令说明 将指定的数据存储器的内容加 1，判断是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 [m] ←[m] + 1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

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

指令说明 将指定数据存储器的内容加 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果会被存放到累加器，但是指定数据存储器的内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 ACC ←[m] + 1，如果 ACC=0 跳过下一条指令执行

影响标志位 无

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

指令说明 判断指定数据存储器的第 i 位，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果为 0，则程序继续执行下一条指令。

功能表示 如果 [m].i≠0，跳过下一条指令执行

影响标志位 无

LSNZ [m] Skip if Data Memory is not 0

指令说明 指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定数据存储器，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果为 0，则程序继续执行下一条指令。

功能表示 如果 [m]≠0，跳过下一条指令执行

影响标志位 无

LSUB A, [m] Subtract Data Memory from ACC

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

功能表示 ACC ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

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

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

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

LSZ [m] Skip if Data Memory is 0
指令说明 指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定数据存储器的内容是否为 0，若为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 如果 [m]=0，跳过下一条指令执行
影响标志位 无

LSZA [m] Skip if Data Memory is 0 with data movement to ACC
指令说明 将指定数据存储器内容复制到累加器，并判断指定数据存储器的内容是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 ACC ← [m]，如果 [m]=0，跳过下一条指令执行
影响标志位 无

LSZ [m].i Skip if bit i of Data Memory is 0
指令说明 判断指定数据存储器的第 i 位是否为 0，若为 0，则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示 如果 [m].i=0，跳过下一条指令执行
影响标志位 无

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

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

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

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

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

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

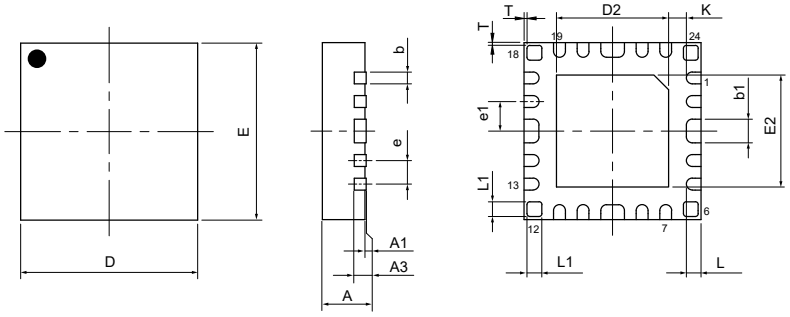
封装信息

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

封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

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

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