单管连续加热电磁炉 **Flash** 单片机

HT45F0059

版本：V1.60　日期：2023-05-23

www.holtek.com

目录

特性

CPU 特性

- 工作电压：
  - ♦ f_{SYS}=16MHz：3.3V~5.5V
- V_{DD}=5V，系统时钟为 16MHz 时，指令周期为 0.25μs
- 提供暂停和唤醒功能，以降低功耗
- 振荡器类型：
  - ♦ 内部高速 16MHz RC – HIRC
  - ♦ 内部低速 32kHz RC – LIRC
- 多种工作模式：快速、低速、空闲和休眠模式
- 完全集成内部振荡器无需外接元器件
- 所有指令都可在 1~3 个指令周期内完成
- 查表指令
- 115 条指令
- 8 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器：4K×16
- 数据存储器：256×8
- True EEPROM 存储器：32×8
- 看门狗定时器功能
- 12 个双向输入 / 输出口
- 9 个外部通道 12-bit 分辨率的 A/D 转换器，具有内部参考电压 V_{BG}
- 9-bit 可编程脉冲发生器
  - ♦ 支持脉宽限制功能
  - ♦ 连续加热功能
  - ♦ 2 组 9-bit PPG 预载寄存器和 2 组 9-bit 计数器逼近寄存器
  - ♦ 支持不可重复触发控制 – 来自 8-bit 定时 / 事件计数器 1
  - ♦ 支持输出高电平有效脉冲、低电平有效脉冲、强制为高电平或强制为低电平
- 3 个 8-bit 定时 / 事件计数器
  - ♦ 定时 / 计数器 0 可用来计算同步脉冲数或测量同步脉冲高 / 低周期
  - ♦ 定时 / 计数器 1 可实现 PPG 不可重复触发功能
- 4 个比较器功能
- 1 个运算放大器功能 – OPAMP
- 1 个过压保护功能 – OVP
- 外围时钟输出
- I^2C 接口
- 内建 16-bit 循环冗余校验功能 – CRC

- 低电压复位功能
- 低电压检测功能
- 封装类型：16-pin NSOP

IGBT 驱动特性

- 内建低压降稳压器 – LDO
  - ♦ 输出驱动能力：30mA (Max.)
  - ♦ 低静态电流：5μA (Typ.)
  - ♦ 低压降
- 内建电平转换器
- 电压检测保护用于电平转换输出使能控制
  - ♦ 检测 V_{DD}
  - ♦ 检测 V_{CC2}

概述

该单片机是一款 8 位具有高性能精简指令集的 Flash 单片机，专门为电磁炉应用而设计。

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。此外存储器方面，还包含了一个 RAM 数据存储器和一个可用于存储序号、校准数据等非易失性数据的 True EEPROM 存储器。

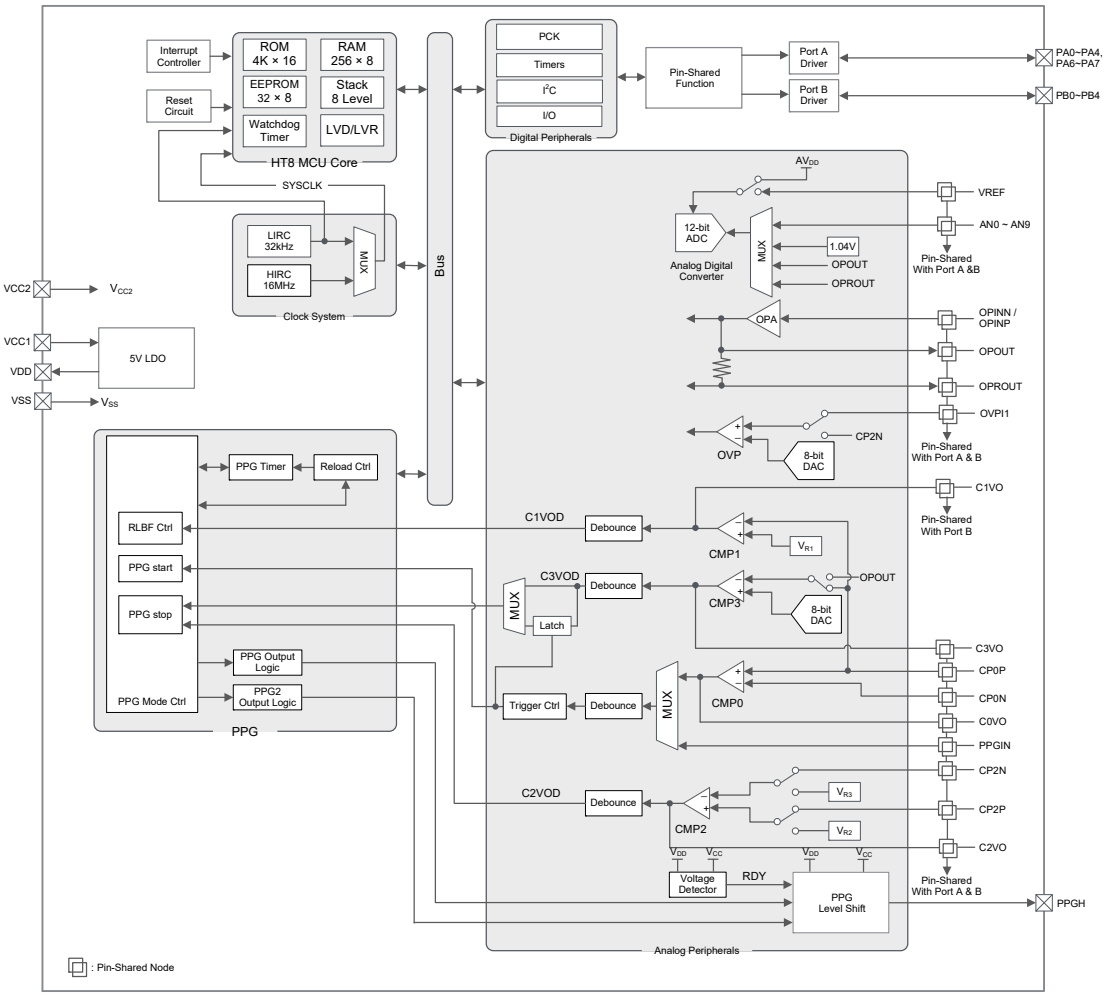
在模拟特性方面，该单片机包含一个多通道 12-bit A/D 转换器、一个 OPAMP 和多个比较器功能。内建的 I^2C 接口功能提供了一个易与外部硬件通信的接口。内部看门狗定时器、低电压复位和低电压检测等内部保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地运行。此外，该单片机可由内建的电平转换器产生两种电压爬升速率以驱动 IGBT 的闸极。用户可通过控制 IGBT 驱动脉波的时间及电压爬升速率，来达到电磁炉低功率连续加热及降低 EMI 影响的目的。

该单片机提供了完全集成的内部高速和低速振荡器，无需外接元器件。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。

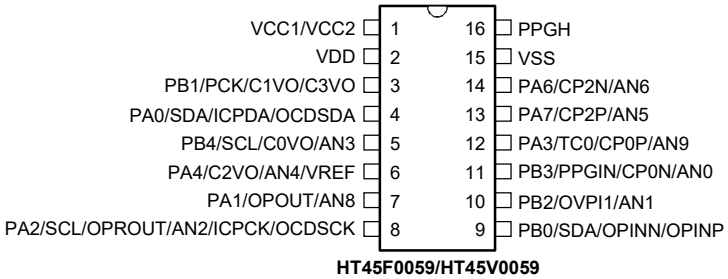
该芯片包含所有满足 IGBT 芯片高电压和大电流需求的电平转换电路。为了避免故障和可能的系统损害，内建电压检测器可监测系统电压电平来决定电平转换器是否使能。

外加 I/O 使用灵活、定时器功能、一个可编程脉冲发生器 PPG、过压保护以及提供外围时钟输出等其它特性，使这款单片机可以广泛应用于电磁炉应用产品中。

方框图



引脚图



HT45F0059/HT45V0059
16 NSOP-A

注：1. 若共用脚同时有多种输出，所需的引脚功能可通过相应的软件控制位决定。
2. OCDSDA 和 OCDSCK 引脚为片上调试功能专用引脚，HT45V0059 是 HT45F0059 的 EV 芯片。

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/SDA/ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SDA	PAS0 IFS	ST	CMOS	I²C 数据线
	ICPDA	—	ST	CMOS	ICP 数据 / 地址
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址，仅用于 EV 芯片
PA1/OPOUT/AN8	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
	OPOUT	PAS0	—	AN	OPAMP 输出引脚
	AN8	PAS0	AN	—	A/D 转换器外部输入通道 8
PA2/SCL/OPROUT/AN2/ICPCK/OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SCL	PAS0 IFS	ST	NMOS	I²C 时钟线
	OPROUT	PAS0	—	AN	OPAMP 输出引脚
	AN2	PAS0	AN	—	A/D 转换器外部输入通道 2
	ICPCK	—	ST	—	ICP 时钟引脚
	OCDSCK	—	ST	—	OCDS 时钟引脚，仅用于 EV 芯片
PA3/TC0/CP0P/AN9	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
	TC0	PAS0	ST	—	定时 / 事件计数器 0 时钟输入
	CP0P	PAS0	AN	—	比较器 0 同相输入
	AN9	PAS0	AN	—	A/D 转换器外部输入通道 9
PA4/C2VO/AN4/VREF	PA4	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	C2VO	PAS1	—	CMOS	比较器 2 输出
	AN4	PAS1	AN	—	A/D 转换器外部输入通道 4
	VREF	PAS1	AN	—	A/D 转换器外部参考电压输入
PA6/CP2N/AN6	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
	CP2N	PAS1	AN	—	比较器 2 反相输入
	AN6	PAS1	AN	—	A/D 转换器外部输入通道 6

引脚名称	功能	OPT	I/T	O/T	说明
PA7/CP2P/AN5	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
	CP2P	PAS1	AN	—	比较器 2 同相输入
	AN5	PAS1	AN	—	A/D 转换器外部输入通道 5
PB0/SDA/OPINN/OPINP	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SDA	PBS0 IFS	ST	CMOS	I²C 数据线
	OPINN	PBS0	AN	—	OPAMP 反相输入
	OPINP	PBS0	AN	—	OPAMP 同相输入
PB1/PCK/C1VO/C3VO	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PCK	PBS0	—	CMOS	PCK 输出
	C1VO	PBS0	—	CMOS	比较器 1 输出
	C3VO	PBS0	—	CMOS	比较器 3 输出
PB2/OVPI1/AN1	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OVPI1	PBS0	AN	—	OV 同相输入 1
	AN1	PBS0	AN	—	A/D 转换器外部输入通道 1
PB3/PPGIN/CP0N/AN0	PB3	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PPGIN	PBS0	ST	—	PPG 外部触发输入
	CP0N	PBS0	AN	—	比较器 0 反相输入
	AN0	PBS0	AN	—	A/D 转换器外部输入通道 0
PB4/SCL/C0VO/AN3	PB4	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SCL	PBS0 IFS	ST	NMOS	I²C 时钟线
	C0VO	PBS1	—	CMOS	比较器 0 输出
	AN3	PBS1	AN	—	A/D 转换器外部输入通道 3
PPGH	PPGH	—	—	CMOS	可编程脉冲发生器 (PPG) 电平转换输出引脚。该引脚内部连接一个 300kΩ 的下拉电阻。
VCC1/VCC2	VCC1	—	PWR	—	LDO 正电源电压引脚
	VCC2	—	PWR	—	电平转换器正电源电压引脚
VDD	VDD	—	PWR	—	数字正电源电压 / LDO 电压输出
VSS	VSS	—	PWR	—	数字负电源电压

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

内部连接信号

有一些引脚未引出至外部封装引脚。这些连线为单片机与 IGBT 驱动器的内部连接线，详见下表。

MCU 信号名称	IGBT 驱动器信号名称	功能	描述
PPG	PWM1	PPG	可编程脉冲发生器输出引脚。 内部连接到电平转换输入 PWM1。
		PWM1	电平转换输入 1 该引脚内部连接一个 25kΩ 的下拉电阻。 内部连接到 MCU I/O 线 PPG。
PA5/PPG2	PWM2	PA5	通用 I/O 口。
		PPG2	可编程脉冲电平发生器输出引脚。 内部连接到电平转换输入 PWM2。
		PWM2	电平转换输入 2 该引脚内部连接一个 25kΩ 的下拉电阻。 内部连接到 MCU I/O 线 PPG2。

注：MCU 的内部信号 PPG 和 PPG2 内部分别连接到 IGBT 驱动器的输入端 PWM1 和 PWM2，应合理设置以控制电平转换器。更多信息可参考“电平转换器”章节。

极限参数

电源供应电压 ..... V_{SS}-0.3V~24V
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。

工作电压特性

Ta=25°C

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 – HIRC	f_{SYS}=f_{HIRC}=16MHz	3.3	—	5.5	V
	工作电压 – LIRC	f_{SYS}=f_{LIRC}=32kHz	3.3	—	5.5	V

工作电流特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{DD}	工作电流 – HIRC	5V	$f_{SYS}=f_{HIRC}$=16MHz	—	3.2	4.8	mA
	工作电流 – LIRC	5V	$f_{SYS}=f_{LIRC}$=32kHz	—	30	50	μA

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. 所有的工作电流值都通过一个连续的 NOP 指令循环程序测得。

待机电流特性

Ta=25°C

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

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

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 (5V) 对 HIRC 进行频率精准度调整。

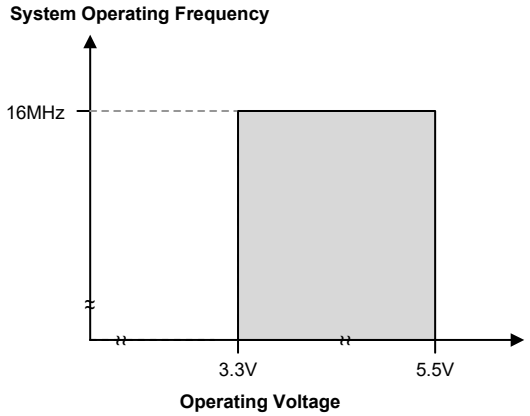
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{HIRC}	通过烧录器调整后的 16MHz HIRC 频率	5V	25°C	-1%	16	+1%	MHz
			-40°C~85°C	-2%	16	+2%	
		3.3V~5.5V	25°C	-2.5%	16	+2.5%	
			-40°C~85°C	-3%	16	+3%	

注：1. 烧录器可在 5V 这个固定电压下对 HIRC 频率进行调整，在此提供 V_{DD}=5V 时的参数值。
2. 5V 表格列下面提供的是全压条件下的参数值。

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}	LIRC 频率	3.3V~5.5V	-40°C~85°C	-7%	32	+7%	kHz
t_{START}	LIRC 启动时间	—	25°C	—	—	100	μs

工作频率电气特性曲线图



系统上电时间电气特性

Ta=25°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)	—	$f_{SYS}=f_H\sim f_H/64$，$f_H=f_{HIRC}$	—	16	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	系统启动时间 (从 f_{SYS} on 的状态下唤醒)	—	$f_{SYS}=f_H\sim f_H/64$，$f_H=f_{HIRC}$	—	2	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SYS}
	系统速度切换时间 (快速模式 → 低速模式或 低速模式 → 快速模式)	—	f_{HIRC} off → on	—	16	—	t_{HIRC}
t_{RSTD}	系统复位延迟时间 (上电复位或 LVR 硬件复位)	—	RR_{POR}=5V/ms	14	16	18	ms
	系统复位延迟时间 (LVRC/WDTC 软件复位)	—	—				
	系统复位延迟时间 (WDT 溢出复位)	—	—	14	16	18	ms
t_{SRESET}	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_{SYS}，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=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{IL}	I/O 口或输入引脚低电平输入电压	5V	—	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	
V_{IH}	I/O 口或输入引脚高电平输入电压	5V	—	3.5	—	5.0	V
		—		$0.8V_{DD}$	—	V_{DD}	
I_{OL}	I/O 口灌电流	5V	$V_{OL}=0.1V_{DD}$	32	65	—	mA
I_{OH}	I/O 口源电流	5V	$V_{OH}=0.9V_{DD}$	-8	-16	—	mA
R_{PH}	I/O 口上拉电阻 (注)	5V	—	10	30	60	kΩ
t_{TC}	TC0 输入引脚最小脉宽	—	—	25	—	—	ns

注：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引脚电流，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。

A/D 转换器电气特性

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_{ADI}	输入电压	—	—	0	—	V_{REF}	V
V_{REF}	参考电压	—	—	2	—	V_{DD}	V
N_R	分辨率	—	—	—	—	12	Bit
DNL	非线性微分误差	—	$V_{REF}=V_{DD}$，$t_{ADCK}=0.5\mu s$ Ta=-40°C~85°C	-3	—	+3	LSB
INL	非线性积分误差	—	$V_{REF}=V_{DD}$，$t_{ADCK}=0.5\mu s$ Ta=-40°C~85°C	-4	—	+4	LSB
I_{ADC}	A/D 转换器使能的额外电流	5V	无负载，$t_{ADCK}=0.5\mu s$	—	500	700	μA
t_{ADCK}	A/D 转换时钟周期	—	—	0.5	—	10.0	μs
t_{ON2ST}	A/D 转换器 On-to-Start 时间	—	—	4	—	—	μs
t_{ADS}	A/D 转换采样时间	—	—	—	4	—	t_{ADCK}
t_{ADC}	A/D 转换时间 (包括采样和保持时间)	—	—	—	16	—	t_{ADCK}

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				
V_{RW}	读 / 写工作电压	—	—	V_{DDmin}	—	V_{DDmax}	V
Flash 程序存储器 / 数据 EEPROM 存储器							
t_{DEW}	擦除 / 写时间 – 数据 EEPROM 存储器	—	—	—	4	6	ms
E_P	存储单元耐受性 – Flash 程序存储器	—	—	10K	—	—	E/W
	存储单元耐受性 – 数据 EEPROM 存储器	—	—	100K	—	—	
t_{RETD}	ROM 数据保存时间	—	Ta=25°C	—	40	—	Year
RAM 数据存储器							
V_{DR}	RAM 数据保存电压	—	—	1.0	—	—	V

注：“E/W”表示擦 / 写次数。

LVD & LVR 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_{LVR}	低电压复位电压	—	LVR 使能，电压选择 2.1V Ta=-40°C~85°C	-5%	2.1	+5%	V
		—	LVR 使能，电压选择 2.55V Ta=-40°C~85°C		2.55		
		—	LVR 使能，电压选择 3.15V Ta=-40°C~85°C		3.15		
		—	LVR 使能，电压选择 3.8V Ta=-40°C~85°C		3.8		
V_{LVD}	低电压检测电压	—	LVD 使能，电压选择 2.0V Ta=-40°C~85°C	-5%	2.0	+5%	V
		—	LVD 使能，电压选择 2.2V Ta=-40°C~85°C		2.2		
		—	LVD 使能，电压选择 2.4V Ta=-40°C~85°C		2.4		
		—	LVD 使能，电压选择 2.7V Ta=-40°C~85°C		2.7		
		—	LVD 使能，电压选择 3.0V Ta=-40°C~85°C		3.0		
		—	LVD 使能，电压选择 3.3V Ta=-40°C~85°C		3.3		
		—	LVD 使能，电压选择 3.6V Ta=-40°C~85°C		3.6		
		—	LVD 使能，电压选择 4.0V Ta=-40°C~85°C		4.0		

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
$I_{LVRLVDBG}$	工作电流	5V	LVD 使能，LVR 使能，VBGEN=0	—	20	25	μA
		5V	LVD 使能，LVR 使能，VBGEN=1	—	180	200	μA
t_{LVDS}	LVDO 稳定时间	—	LVR 使能，VBGEN=0，LVD off → on，Ta=-40°C~85°C	—	—	18	μs
t_{LVR}	产生 LVR 复位的低电压最短保持时间	—	TLVR[1:0]=00B	120	240	480	μs
			TLVR[1:0]=01B	0.5	1.0	2.0	ms
			TLVR[1:0]=10B	1	2	4	ms
			TLVR[1:0]=11B	2	4	8	ms
t_{LVD}	产生 LVD 中断的低电压最短保持时间	—	—	60	120	240	μs

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_{BG}	Bandgap 参考电压	—	Ta=-40°C~85°C	-5%	1.04	+5%	V
t_{BGS}	V_{BG} 开启稳定时间	—	无负载	—	—	150	μs

注：V_{BG} 电压用于 A/D 转换器内部信号输入。

过压保护电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{OVP}	工作电流	5V	OVPEN=1，D/A 转换器 $V_{REF}=V_{DD}$	—	500	750	μA
V_{OS}	输入失调电压	5V	已校准	-2	—	2	mV
V_{HYS}	迟滞	5V	HYS[1:0]=00B	0	0	5	mV
		5V	HYS[1:0]=01B	15	30	45	mV
		5V	HYS[1:0]=10B	40	60	80	mV
		5V	HYS[1:0]=11B	60	80	100	mV
V_{CM}	共模电压范围	5V	—	V_{SS}	—	V_{DD}-1	V
DNL	非线性微分误差	5V	D/A 转换器 $V_{REF}=V_{DD}$	-1	—	+1	LSB
INL	非线性积分误差	5V	D/A 转换器 $V_{REF}=V_{DD}$	-1.5	—	+1.5	LSB
t_{RP}	OVP 响应时间	5V	OVPDA=10110011B，OVPDEB[2:0]=000，D/A 转换器 $V_{REF}=V_{DD}$，OVP 输入 = 2.1V~3.6V	—	1.0	1.8	μs

运算放大器电气特性

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}	运算放大器使能的额外电流	5V	无负载	—	300	450	μA
		5V	无负载，OPG[1:0]=00B (1)	—	450	700	μA
V_{OS}	输入失调电压	—	未校准，(OOF[5:0]=100000B)	-15	—	15	mV
		—	已校准	-2	—	2	
V_{CM}	共模电压范围	5V	—	V_{SS}	—	V_{DD}-1.4	V
V_{OR}	最大输出电压范围	5V	—	V_{SS}+0.1	—	V_{DD}-0.1	V
SR	转换速率	5V	无负载	0.6	1.8	—	V/μs
GBW	增益带宽	5V	R_{LOAD}=1MΩ，C_{LOAD}=100pF	600	2200	—	kHz
PSRR	电源抑制比	5V	—	60	80	—	dB
CMRR	共模抑制比	5V	—	60	80	—	dB
Ga	PGA 增益精准度 (2)	5V	相对增益，Ta=-40°C~85°C	-5	—	5	%
R_{OPAR2}	OPAR2 电阻	5V	—	0.75	1.00	1.25	kΩ
R_{OPAR3}	OPAR3 电阻	5V	—	0.75	1.00	1.25	kΩ
R_{OPAR4}	OPAR4 电阻	5V	—	—	10	—	kΩ

注：1. 使 OPA 接成 PGA 形式，使用内部增益来量测，PGA 电流损耗包括放大电阻的电流损耗。
2. 仅当 PGA 输出电压满足 V_{OR} 规格时才能确保 PGA 增益精准度。

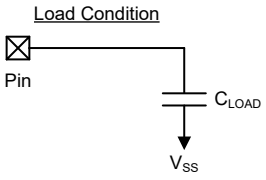
比较器电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{CMP}	比较器使能的额外电流	5V	CnEN=1 (n=0)	—	55	80	μA
		5V	CnEN=1 (n=1~2)	—	220	300	μA
		5V	CnEN=1 (n=3)	—	500	750	μA
V_{OS}	输入失调电压	—	未校准 (CnCOF[5:0]=100000B) (n=0)	-15	—	15	mV
		—	未校准 (CnCOF[4:0]=10000B) (n=1~3)	-15	—	15	mV
		—	已校准	-2	—	2	mV
V_{CM}	共模电压范围	—	—	V_{SS}	—	V_{DD}-1.0	V
V_{HYS}	迟滞	5V	—	20	45	75	mV
t_{CIRP}	CMPn 中断响应时间 (n=0)	—	迟滞除能，去抖除能	—	0.8	1.5	μs
		—	迟滞除能，去抖使能，C0DBC[5:0]=000001B~101111B	—	C0DBC[5:0] ×t_{PPGDCK}+0.8	C0DBC[5:0] ×t_{PPGDCK}+1.5	μs
			迟滞除能，去抖使能，C0DBC[5:0]=110000B~111111B	—	48×t_{PPGDCK} +0.8	48×t_{PPGDCK} +1.5	μs
	CMPn 中断响应时间 (n=1~3)	—	迟滞除能，去抖除能 (1)	—	0.8	1.5	μs
		—	迟滞除能，去抖使能 (1)	—	4×t_{PPGDCK} +0.8	4×t_{PPGDCK} +1.5	μs

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
t_{INTDY}	INT00 延迟时间 (包括去抖时间)	5V	PPGDL[5:0]=000001B~101111B CMPDBC0=00H	Typ. -0.2	PPGDL[5:0] ×t_{PPGDCK}+0.08	Typ. +0.2	μs
		5V	PPGDL[5:0]=110000B~111111B CMPDBC0=00H	Typ. -0.2	48×t_{PPGDCK} +0.08	Typ. +0.2	μs
V_{R1}	比较器 1 参考电压	5V	—	-5%	0.600	+5%	V_{DD}
					0.625		
					0.650		
					0.675		
					0.700		
					0.725		
					0.750		
					0.775		
V_{R2}	比较器 2 参考电压	5V	—	-5%	0.600	+5%	V_{DD}
					0.625		
					0.650		
					0.675		
					0.700		
					0.725		
					0.750		
					0.775		
V_{R3}	比较器 2 参考电压	5V	—	-5%	0.075	+5%	V_{DD}
					0.100		
					0.125		
					0.150		
					0.175		
					0.200		
					0.225		
					0.250		

注：负载条件：C_{LOAD}=50pF。



LDO 特性

V_{IN}=18V，Ta=-40°C~85°C，除非另有说明。(C_{IN}=10μF，C_O=4.7μF)

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{IN}	输入电压 (V_{CC1})	—	—	16	—	20	V
V_{OUT}	输出电压	—	Ta=25°C，Io=1mA	-2%	5	2%	V
		—	-40°C≤Ta<85°C，Io=1mA	-5%	5	5%	V
		—	-40°C≤Ta<85°C，Io=30mA	-5%	5	5%	V
I_{OUT}	输出电流 (注)	—	16V≤V_{IN}≤20V，V_{OUT}=5V	—	—	30	mA
I_{QS}	静态电流	—	V_{IN}=18V，I_O=0mA (无负载)	—	5	8	μA
ΔV_{LINE}	线性调节	—	16V≤V_{IN}≤20V，I_O=1mA	—	—	0.2	%/V
$\Delta V_{OUT}/\Delta Ta$	温度系数	—	I_O=1mA，0°C≤Ta<85°C	—	±1.5	±2	mV/°C

注：负载稳定度在恒结温条件下使用一个低 ON 时间的脉冲测得，测量时确保达到最大的功耗。功耗由输入 / 输出差分电压和输出电流决定。确保的最大功耗不允许超出全输入 / 输出范围。任何环境温度下的最大可允许功耗为 $P_D=(T_{J(MAX)} - T_a)/\theta_{JA}$。

电平转换 / 电压检测器电气特性

V_{DD}=5V，V_{CC2}=18V，Ta=25°C，除非另有说明。(C_{IN}=10μF，C_O=4.7μF)

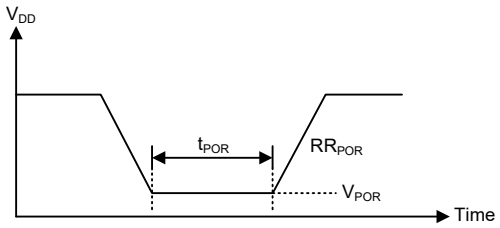
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{CC2}	电平转换电源电压	—	—	16	—	20	V
I_{OPR1}	V_{CC1} 工作电流 (注)	5V	V_{CC1}=18V，PPG/PPG2=0 (无负载)	—	5	8	μA
I_{OPR2}	V_{CC2} 工作电流 (注)	5V	V_{CC2}=18V，PPG/PPG2=0 (无负载)	—	15	33	μA
电平转换							
V_{IH}	输入高电压	—	—	0.6V_{DD}	—	V_{DD}	V
V_{IL}	输入低电压	—	—	0	—	0.3V_{DD}	V
I_{SOURCE}	输出源电流用于 PPGH 引脚	—	V_{OH}=0.9V_{CC2}，V_{CC2}=18V	-105	-150	—	mA
I_{SINK1}	输出灌电流用于 PPGH 引脚	—	V_{OL}=0.1V_{CC2}，V_{CC2}=18V	105	150	—	mA
R_{PD1}	带下拉电阻的电平转换输入 PPG/PPG2	—	—	-50%	25	+50%	kΩ
R_{PD2}	带下拉电阻的电平转换输出 PPGH	—	—	-50%	300	+50%	kΩ
电压检测器							
V_{DET1}	V_{DD} 检测电平	—	V_{DD}=0 → 4V，Ta=-40~85°C	-0.15	3.00	+0.15	V
	迟滞	—	V_{DD}=0 ↔ 4V，Ta=-40~85°C	—	250	—	mV
V_{DET2}	V_{CC2} 检测电平	—	V_{CC2}=0 → 12V，Ta=-40~85°C	-0.45	9.00	+0.45	V
	迟滞	—	V_{CC2}=0 ↔ 12V，Ta=-40~85°C	—	750	—	mV
V_{DET3}	V_{CC2} 检测电平	—	V_{CC2}=0 → 22V，Ta=-40~85°C	-1.03	20.60	+1.03	V
	迟滞	—	V_{CC2}=0 ↔ 22V，Ta=-40~85°C	—	1100	—	mV

注：工作电流或待机电流包括电平转换电路和电压检测器的功耗。

上电复位特性

Ta=-25°C

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



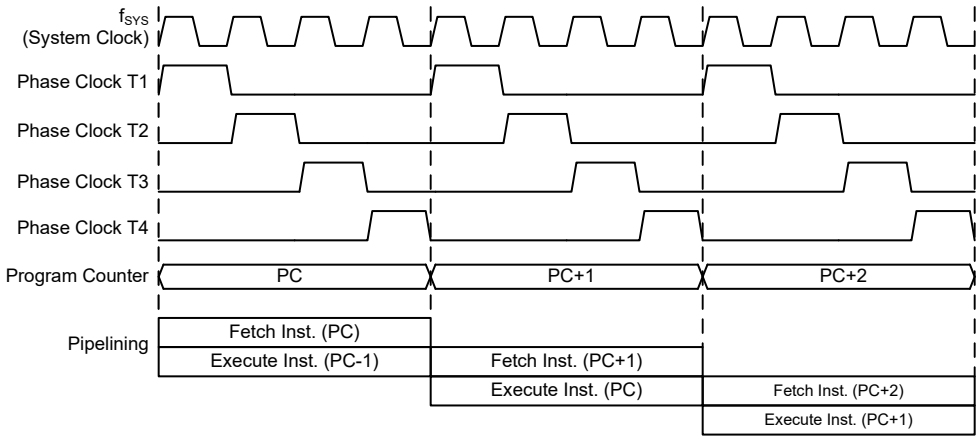
系统结构

内部系统结构是 Holtek 单片机具有良好性能的主要因素。由于采用 RISC 结构，此单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的取得和执行同时进行，此举使得除了跳转和调用指令需多一个指令周期外，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。使得这些单片机适用于低成本和大量生产的控制应用。

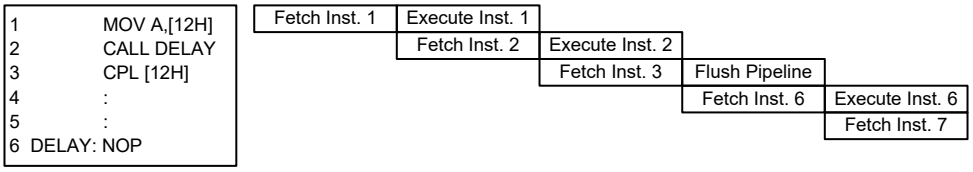
时序和流水线结构

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

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



系统时序和流水线



指令捕捉

程序计数器

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

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

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

程序计数器

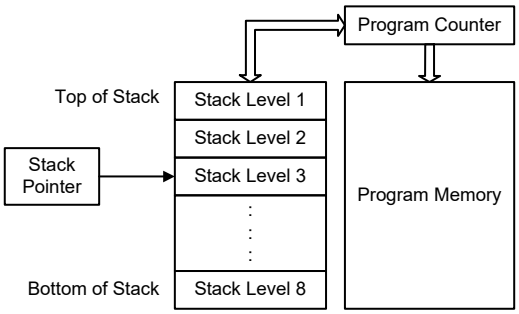
程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。程序计数器的低字节可由程序直接进行读取，PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

堆栈

堆栈是一个特殊的存储空间，用来存储程序计数器中的内容。该单片机有 8 层堆栈，堆栈是不可读取或写入的。当前层由堆栈指针 (SP) 加以指示，同样也是不可读写的。在子程序调用或中断响应服务时，程序计数器的内容被压入到堆栈中。当子程序或中断响应结束时，返回指令 (RET 或 RETI) 使程序计数器从堆栈中重新得到它以前的值。当一个芯片复位后，堆栈指针将指向堆栈顶部。

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

若堆栈溢出，则首个存入堆栈的程序计数器数据将会丢失。



算术逻辑单元 – ALU

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

- 算术运算：
  ADD，ADDM，ADC，ADCM，SUB，SUBM，SBC，SBCM，DAA
  LADD，LADDM，LADC，LADCM，LSUB，LSUBM，LSBC，LSBCM，LDAA
- 逻辑运算：
  AND，OR，XOR，ANDM，ORM，XORM，CPL，CPLA
  LAND，LANDM，LOR，LORM，LXOR，LXORM，LCPL，LCPLA
- 移位运算：
  RRA，RR，RRCA，RRC，RLA，RL，RLCA，RLC
  LRRA，LRR，LRRCA，LRRC，LRLA，LRL，LRLCA，LRLC

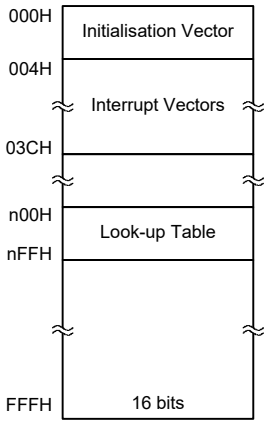
- 递增和递减：
  INCA，INC，DECA，DEC
  LINCA，LINC，LDECA，LDEC
- 分支判断：
  JMP，SZ，SZA，SNZ，SIZ，SDZ，SIZA，SDZA，CALL，RET，RETI
  LSZ，LSZA，LSNZ，LSIZ，LSIZA，LSDZ，LSDZA

Flash 程序存储器

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

结构

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



程序存储器结构

特殊向量

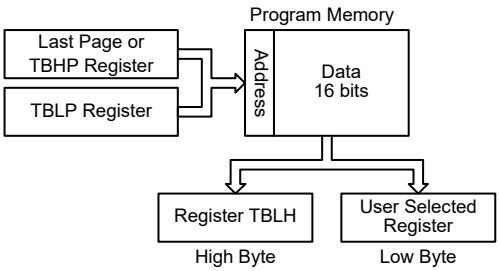
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 0000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

查表

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

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

下图是查表中寻址 / 数据流程：



查表范例

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

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

表格读取程序范例

```
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,0Fh           ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1      ; transfers value in table referenced by table pointer
                    ; data at program memory address "0F06H" transferred to
                    ; tempreg1 and TBLH
dec tblp            ; reduce value of table pointer by one
tabrd tempreg2      ; transfers value in table referenced by table pointer
                    ; data at program memory address "0F05H" transferred to
                    ; tempreg2 and TBLH
                    ; in this example the data "1AH" is transferred to
                    ; tempreg1 and data "0FH" to tempreg2
:
:
org 0F00h           ; sets initial address of program memory
```

```
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
:
:
```

在线烧录 – ICP

Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。

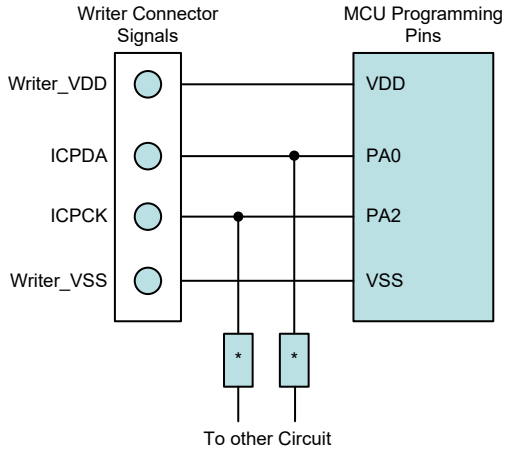
另外，Holtek 单片机提供 4 线接口的在线烧录方式。用户可将进行过烧录或未经过烧录的单片机芯片连同电路板一起制成，最后阶段进行程序的更新和程序的烧录，在无需去除或重新插入芯片的情况下方便地保持程序为最新版。

Holtek Flash 单片机与烧录器引脚对应表格如下：

Holtek 烧录器引脚名称	MCU 烧录引脚名称	引脚说明
ICPDA	PA0	烧录串行数据 / 地址
ICPCK	PA2	烧录时钟
VDD	VDD	电源
VSS	VSS	地

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

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



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

片上调试 – OCDS

EV 芯片 HT45V0059 用于单片机 HT45F0059 仿真。此单片机的 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。除了片上调试功能方面，EV 芯片和实际 MCU 在功能上几乎是兼容的。用户可将 OCDSDA 和 OCDSCK 引脚连接至 Holtek HT-IDE 开发工具，从而实现 EV 芯片对实际 MCU 的仿真。OCDSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户用 EV 芯片进行调试时，实际单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。关于 OCDS 功能的详细描述，请参考“Holtek e-Link for 8-bit MCU OCDS 使用手册”文件。

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

数据存储器

数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。

数据存储器分为两个部分，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。第二部分通用数据存储器是做一般用途使用，都可在程序控制下进行读取和写入。

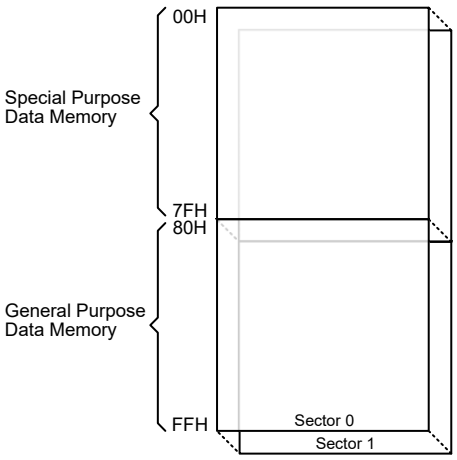
结构

数据存储器被分为 2 个 Sector，都位于 8 位存储器中。每个 Sector 分为两部分，即特殊功能数据存储器和通用数据存储器。当使用间接寻址方式时，切换不同区域可通过设置存储器指针实现。

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

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

数据存储器概要



数据存储器结构

数据存储器寻址

此单片机支持扩展指令集，它并没有可用于数据存储器的存储区指针。对于数据存储器，使用间接寻址访问方式时所需的 Sector 是通过 MP1H 或 MP2H 寄存器指定，而所选 Sector 的具体数据存储器地址的选择是通过 MP1L 或 MP2L 寄存器完成。

直接寻址可用于所有 Sector，通过扩展指令可以寻址所有可用的数据存储器空间。当所访问的数据存储器位于除 Sector 0 外的任何数据存储器 Sector 时，扩展指令可代替间接寻址方式用来访问数据存储器。标准指令和扩展指令的主要区别在于扩展指令中的数据存储器地址“m”有 9 个有效位，高字节表示选择的 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单独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，具体细节的介绍请参考相关特殊功能寄存器的部分。要注意的是，任何对存储器中未定义的地址进行的读取指令，都将返回“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	HIRCC	
12H	LVRC	
13H	LVDC	
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	PB	
19H	PBC	
1AH	PBPU	
1BH	PAS0	
1CH	PAS1	
1DH	PBS0	
1EH	PBS1	
1FH	WDTC	
20H	INTC0	
21H	INTC1	
22H	INTC2	
23H	TMR0C	
24H	TMR0	
25H	TMR1C	
26H	TMR1	
27H	TMR2C	
28H	TMR2	
29H	PSCR	
2AH	PCKC	
2BH	SADOL	
2CH	SADOH	
2DH	SADC0	
2EH	SADC1	
2FH	OVPC0	
30H	OVPC1	
31H	OVPC2	
32H	OVPDA	
33H	OPC	
34H	OPVOS	
35H	OPS	
36H	PPGC0	
37H	PPGC1	
38H	PPGC2	
39H	PPGTA	
3AH	PPGTB	
3BH	PPGTC	
3CH	PPGTEX	
3DH	PWLT	
3EH	PPGPC	
3FH	CMP0C	

	Sector 0	Sector 1
40H	CMP1C	EEC
41H	CMP2C	
42H	CMP3C	
43H	CMPVREF0	
44H	CMPVREF1	
45H	CMPCTL0	
46H	CMPCTL1	
47H	CMPDBC0	
48H	CMPDBC1	
49H	CMPHYS	
4AH	TLVRC	
4BH	EEA	
4CH	EED	
4DH	INTC3	
4EH	PPGTD	
4FH	PPGATC0	
50H	PPGATC1	
51H	PPGATC2	
52H	PPGTMC	
53H	PPGTMR1	
54H	PPGTMR2	
55H	PPGTMR3	
56H	PPGTMRD	
57H	ORMC	
58H	PPGRT	
59H	PPGRN	
5AH	PPG2CT	
5BH	PPG2C0	
5CH	PPG2C1	
5DH	PPG2C2	
5EH	PPG2C3	
5FH	IICC0	
60H	IICC1	
61H	IICD	
62H	IICA	
63H	IICTOC	
64H	CRCCR	
65H	CRCIN	
66H	CRCDL	
67H	CRCDH	
68H	IFS	
69H	IECC	
6AH		
6BH		
6CH		
6DH		
6EH		
6FH		
70H	C3LEBC	
71H	C3DA	
72H		
73H		
74H		
75H		
76H		
77H		
78H		
79H		
7AH		
7BH		
7CH		
7DH		
7EH		
7FH		

: Unused, read as 00H

特殊功能数据存储器结构

特殊功能寄存器

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

间接寻址寄存器 – IAR0, IAR1, IAR2

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

存储器指针 – MP0, MP1L/MP1H, MP2L/MP2H

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

范例 2

```
rambank 1 data1
data1 .section at 080H 'data'
adres1 db ?
adres2 db ?
```

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

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。

使用扩展指令直接寻址程序举例

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

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

累加器 – ACC

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

程序计数器低字节寄存器 – PCL

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

查表寄存器 – TBLP, TBHP, TBLH

这三个特殊功能寄存器对存储在程序存储器中的表格进行操作。TBLP 和 TBHP 为表格指针，指向表格数据存储的地址。它们的值必须在任何表格读取指令执行前加以设定，由于它们的值可以被如“INC”或“DEC”的指令所改变，这就提供了一种简单的方法对表格数据进行读取。表格读取数据指令执行之后，表格数据高字节存储在 TBLH 中。其中要注意的是，表格数据低字节会被传送到使用者指定的地址。

配置存储器映射寄存器 – 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 32 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~1FH 地址会一一对应到程序存储器最后一页的 E0H~FFH 地址。

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

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

- **ORMC 寄存器**

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

Bit 7~0 **ORMC7~ORMC0**：配置存储器映射特定序列

当将特定序列 55H 和 AAH 连续写入该寄存器，会使能对配置存储器的访问。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

此 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。

Z、OV、AC、C、SC 和 CZ 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
进位标志位 C 也受循环移位指令的影响。

EEPROM 数据存储器

此单片机的一个特性是内建 EEPROM 数据存储器，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空间，对设计者来说增加了许多新的应用机会。EEPROM 可以用来存储产品编号、校准值、用户特定数据、系统配置参数或其它产品信息等。EEPROM 的数据读取和写入过程也会变的更简单。

EEPROM 数据存储器结构

EEPROM 数据存储器容量为 32×8。由于映射方式与程序存储器和数据存储器不同，因此不能像其它类型的存储器一样寻址。使用一个地址和一个数据寄存器以及一个仅位于 Sector 1 中的一个控制寄存器，可以实现对 EEPROM 的单字节读写操作。

EEPROM 寄存器

有三个寄存器控制内部 EEPROM 数据存储器总的操作，地址寄存器 EEA、数据寄存器 EED 及控制寄存器 EEC。EEA 和 EED 位于 Sector 0 中，它们能像其它特殊功能寄存器一样直接被访问。EEC 位于 Sector 1 中，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
EEA	—	—	—	EEA4	EEA3	EEA2	EEA1	EEA0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	D7	—	—	—	WREN	WR	RDEN	RD

EEPROM 寄存器列表

- **EEA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	EEA4	EEA3	EEA2	EEA1	EEA0
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　**EEA4~EEA0：**数据 EEPROM 地址
数据 EEPROM 地址 bit 4 ~ bit 0

- **EED 寄存器**

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

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

- **EEC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	—	—	—	WREN	WR	RDEN	RD
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **D7**：保留，必须固定为“0”

Bit 6~4 未定义，读为“0”

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

Bit 2 **WR**：EEPROM 写控制位
0：写周期结束
1：启动写周期
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：启动读周期
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 WREN、WR、RDEN 和 RD 不能同时置为“1”。WR 和 RD 不能同时置为“1”。
2. 需确保 f_{SUB} 时钟运行稳定后才可执行写操作。
3. 需确保写操作已执行完毕后才可改写 EEPROM 相关寄存器。

从 EEPROM 中读取数据

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

写数据到 EEPROM

写数据至 EEPROM，EEPROM 中写入数据的地址要先放入 EEA 寄存器中，要写入的数据要存入 EED 寄存器中。写数据至 EEPROM，EEC 寄存器中的写使能位 WREN 先置为高以使能写功能。之后将 WR 位置为高，初始化一个写周期。这两个指令必须在两个连续的指令周期内执行。在执行任何写操作之前，总中断位 EMI 要先清零，写周期开始后，再将 EMI 置为高。需要注意的是若 WR 位已置为高而 WREN 位还未被设置则不能开始写操作。由于控制 EEPROM 写周期是一个内部时钟，与单片机的系统时钟异步，所以数据写入 EEPROM 的时间将有所延迟。可通过轮询 EEC 寄存器中的 WR 位或判断 EEPROM 中断以侦

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

写保护

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

EEPROM 中断

EEPROM 写周期结束后将产生 EEPROM 写中断，需先通过设置相关中断寄存器的 DEE 位使能 EEPROM 写中断。当 EEPROM 写周期结束，DEF 请求标志位将被置位。若 EEPROM 中断使能且堆栈未满的情况下将跳转到相应的 EEPROM 中断向量中执行。当中断被响应，EEPROM 中断标志位 DEF 位将自动清零且 EMI 位也会自动清零以除能其它中断。更多细节将在中断章节讲述。

编程注意事项

必须注意的是数据不会无意写入 EEPROM。在没有写动作时写使能位被正常清零可以增强保护功能。存储器指针高字节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。

写数据时，WREN 位置为“1”后，WR 需立即设置为高，以确保写周期正确地执行。写周期执行前总中断位 EMI 应先清零，写周期开始执行后再将此位重新使能。注意，EEPROM 读或写周期彻底完成前单片机不能进入空闲或休眠模式，否则将导致 EEPROM 读或写操作失败。

程序举例

从 EEPROM 中读取数据 – 轮询法

```
MOV  A, EEPROM_ADRES        ; user defined address
MOV  EEA, A
MOV  A, 40H                 ; setup memory pointer low byte MP1L
MOV  MP1L, A                ; MP1L points to EEC register
MOV  A, 01H                 ; setup Memory Pointer high byte MP1H
MOV  MP1H, A
SET  IAR1.1                 ; set RDEN bit, enable read operations
SET  IAR1.0                 ; start Read Cycle - set RD bit
BACK:
SZ   IAR1.0                 ; check for read cycle end
JMP  BACK
CLR  IAR1                   ; disable EEPROM read if no more read operations
                            ; are required
CLR  MP1H
MOV  A, EED                 ; move read data to register
MOV  READ_DATA, A
```

注：对于每一个读操作，即使目标地址是连续的，每次执行读操作仍需重新定义地址寄存器，接着置位 RD 以启动一个读周期。

写数据到 EEPROM – 轮询法

```
MOV  A, EEPROM_ADRES        ; user defined address
MOV  EEA, A
MOV  A, EEPROM_DATA         ; user defined data
MOV  EED, A
MOV  A, 40H                 ; setup memory pointer low byte MP1L
MOV  MP1L, A                ; MP1L points to EEC register
MOV  A, 01H                 ; setup Memory Pointer high byte MP1H
MOV  MP1H, A
CLR  EMI
SET  IAR1.3                 ; set WREN bit, enable write operations
SET  IAR1.2                 ; start Write Cycle - set WR bit - executed
                            ; immediately after set 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。振荡器的选择和操作是通过应用程序控制寄存器实现的。

振荡器概述

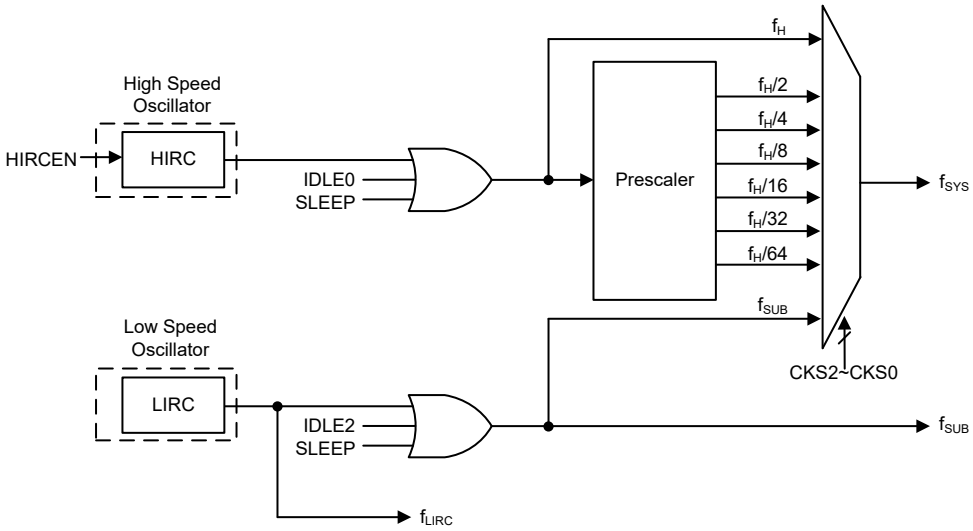
振荡器除了作为系统时钟源，还作为看门狗定时器和时基功能的时钟源。集成的两个内部振荡器不需要任何外围器件。它们提供高速和低速系统振荡器。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

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

振荡器类型

系统时钟配置

此单片机有两个系统振荡器，包括一个高速振荡器和一个低速振荡器。高频时钟 f_H 来自内部高速 16MHz RC 振荡器 HIRC。低频时钟 f_{SUB} 来自内部低速 32kHz RC 振荡器 LIRC。使用高速或低速振荡器作为系统时钟的选择是通过设置 SCC 寄存器中的 CKS2~CKS0 位决定的，系统时钟可动态选择。



系统时钟配置

内部 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，不需其它外部器件。内部 RC 振荡器具有固定频率为 16MHz。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因 V_{DD}、温度以及芯片制成工艺不同的影响较大程度的降低。应注意如果选择了该内部时钟，无需额外的引脚。

内部 32kHz 振荡器 – LIRC

内部 32kHz 系统振荡器是一个低频振荡器。该完全集成的 RC 振荡器在全压下运行的典型频率值为 32kHz 且无需外部元器件。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的降低。

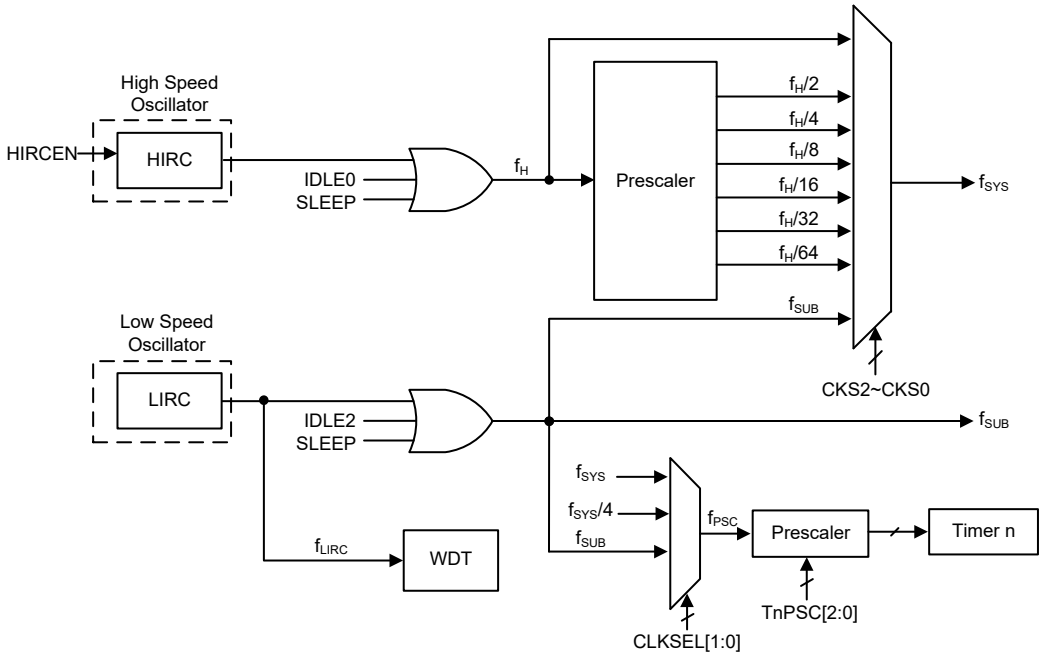
工作模式和系统时钟

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

系统时钟

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

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



单片机时钟配置

注：当系统时钟源 f_{SYS} 由 f_H 到 f_{SUB} 转换时，可以通过设置相应的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}
		FHIDEN	FSIDEN	CKS2~CKS0				
FAST	On	x	x	000~110	f_H~f_H/64	On	On	On
SLOW	On	x	x	111	f_{SUB}	On/Off(1)	On	On
IDLE0	Off	0	1	000~110	Off	Off	On	On
				111	On			
IDLE1	Off	1	1	xxx	On	On	On	On
IDLE2	Off	1	0	000~110	On	On	Off	On
				111	Off			
SLEEP	Off	0	0	xxx	Off	Off	Off	On(2)

“x”：无关

注：1. 在低速模式中，f_H 开启或关闭由相应的振荡器使能位控制。
2. 在休眠模式中，由于 WDT 功能始终使能，f_{LIRC} 将开启。

快速模式

这是主要的工作模式之一，单片机的所有功能均可在此模式中实现且系统时钟由一个高速振荡器提供。该模式下单片机正常工作的时钟源来自 HIRC 振荡器。高速振荡器频率可被分为 1~64 的不等比率，实际的比率由 SCC 寄存器中

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

低速模式

此模式的系统时钟虽为较低速时钟源，但单片机仍能正常工作。该低速时钟源来自 f_{SUB}，而 f_{SUB} 来自 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。由于看门狗定时器功能始终使能，f_{LIRC} 继续运行。

空闲模式 0

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

空闲模式 1

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，但高速和低速振荡器都会开启以确保一些外围功能继续工作。

空闲模式 2

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 位为高、FSIDEN 位为低时，系统进入空闲模式 2。在空闲模式 2 中，CPU 停止，但高速振荡器会开启以确保一些外围功能继续工作。

控制寄存器

寄存器 SCC 和 HIRCC 用于控制系统时钟和 HIRC 振荡器设置。

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

系统工作模式控制寄存器列表

- **SCC 寄存器**

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

Bit 7~5 **CKS2~CKS0**：系统时钟选择

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

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

Bit 4~2　未定义，读为“0”

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

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

注：使用 CKS2~CKS0 位进行时钟切换设置之后，在相关时钟成功切换至目标时钟源之前需要一定的延时。因此，若接下来执行的操作需要目标时钟源立即响应，则在此之前必须规划适当的延迟时间。

时钟切换延迟时间 = $4\times t_{SYS}+[0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$，其中 $t_{Curr.}$ 指代当前的时钟周期，$t_{Tar.}$ 指代目标时钟周期，t_{SYS} 指代当前系统时钟周期。

- **HIRCC 寄存器**

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

Bit 7~2　未定义，读为“0”

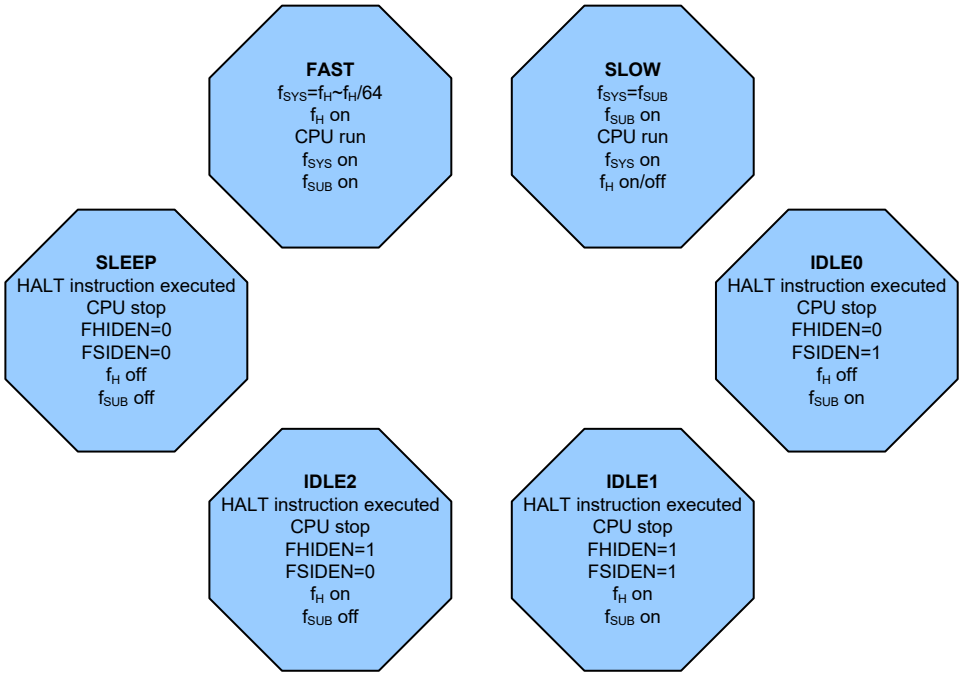
Bit 1　**HIRCF**：HIRC 振荡器稳定标志位
0：HIRC 未稳定
1：HIRC 稳定
此位用于表明 HIRC 振荡器是否稳定。HIRCEN 位置高使能 HIRC 振荡器，HIRCF 位会先被清零，在 HIRC 稳定后会被置高。

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

工作模式切换

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

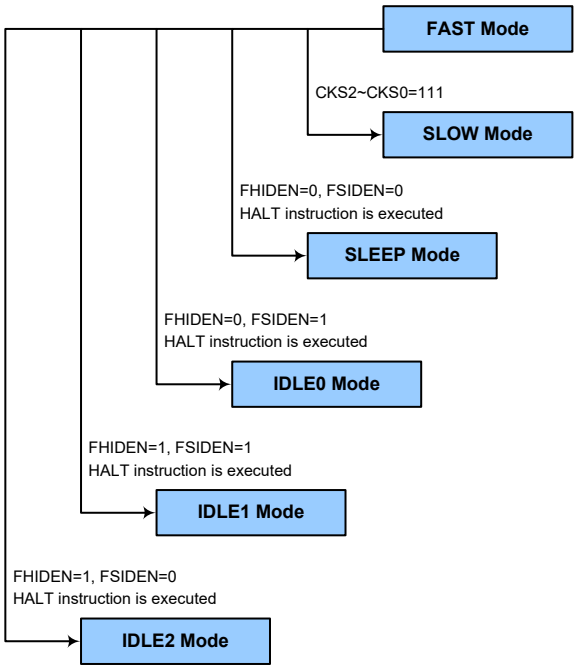
简单来说，快速模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



快速模式切换到低速模式

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

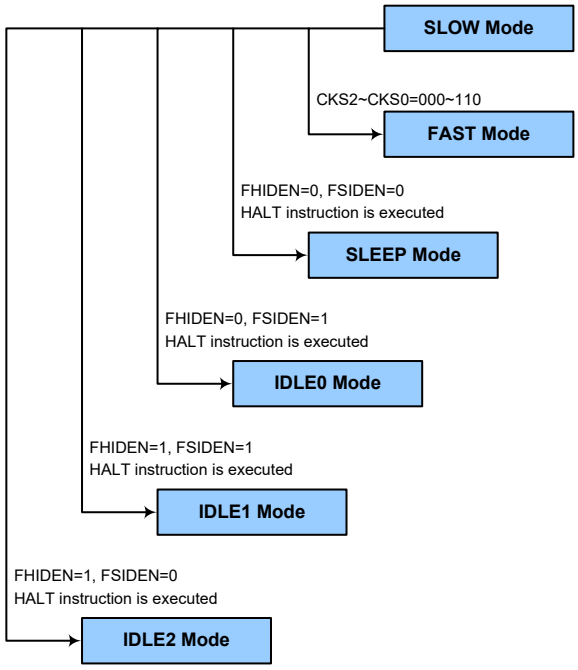
低速模式的时钟源来自 LIRC 振荡器，因此要求该振荡器在所有模式切换动作发生前稳定下来。



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 HIRCC 寄存器中的 HIRCF 位进行判断，所需的高速系统振荡器稳定时间在“系统上电时间电气特性”中有说明。



进入休眠模式

进入休眠模式的方法仅有一种，即应用程序中执行“HALT”指令前需设置 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为“0”。在这种模式下，除了 WDT 以外的所有时钟和功能都将关闭。在上述条件下执行该指令后，将发生的情况如下：

- 系统时钟停止运行，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 由于 WDT 功能始终使能，WDT 将被清零并重新开始计数。

进入空闲模式 0

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

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

- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 由于 WDT 功能始终使能，WDT 将被清零并重新开始计数。

进入空闲模式 1

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

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

进入空闲模式 2

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

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

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，休眠模式和空闲模式 0 下可能到只有几个微安的级别，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 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

单片机可进入休眠或空闲模式以节省功耗，此期间 MCU 将停止。然而，当单片机再次被唤醒，初始系统时钟的重启、稳定直到可为正常运行提供时钟源，这个过程需要一定的时间。

单片机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

当单片机执行 HALT 指令，PDF 将被置位。系统上电或执行清除看门狗的指令，会清零 PDF。若系统由 WDT 溢出唤醒，将会发生看门狗定时器复位。看门狗计数器溢出将会置位 TO 标志并唤醒系统，这种复位会重置程序计数器和堆栈指针，其它标志保持原有状态。

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

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源来自于内部时钟 f_{LIRC}，而 f_{LIRC} 的时钟源由 LIRC 振荡器提供。内部振荡器 LIRC 的周期大约为 32kHz。需要注意的是，这个特殊的内部时钟周期随 V_{DD}、温度和制成的不同而变化。看门狗定时器的时钟源可分频为 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 功能的使能及 MCU 软件复位操作。

- **WDTC 寄存器**

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

Bit 7~3 **WE4~WE0**：WDT 功能使能控制
01010 或 10101：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} 时间后，此复位完成后 RSTFC 寄存器中的 WRF 标志位会被置位。

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

- **RSTFC 寄存器**

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

“x”：未知

Bit 7~3　未定义，读为“0”

Bit 2　**LVRF**：LVR 复位标志位
具体描述见低电压复位章节。

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

Bit 0　**WRF**：WDT 控制寄存器软件复位标志位
0：未发生
1：发生
当 WDT 控制寄存器软件复位发生时，此位被置为“1”，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 位可提供使能控制以及控制看门狗定时器复位操作。当 WE4~WE0 设置为“10101B”或“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”。

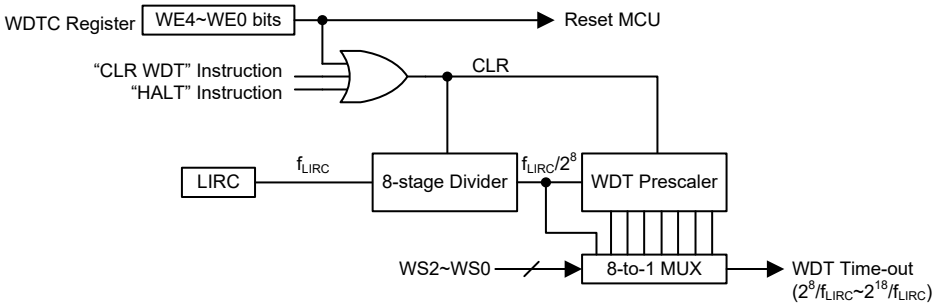
WE4~WE0 位	WDT 功能
01010B 或 10101B	使能
其它值	单片机复位

看门狗定时器功能控制

程序正常运行时，WDT 溢出将导致芯片复位，并置位状态标志位 TO。若系统处于休眠或空闲模式，当 WDT 发生溢出时，状态寄存器中的 TO 将被置位，仅程序计数器 PC 和堆栈指针 SP 将被复位。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 32kHz LIRC 振荡器，分频比为 2^{18} 时最大溢出周期约 8s，分频比为 2^{8} 时最小溢出周期约 8ms。



看门狗定时器

复位和初始化

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

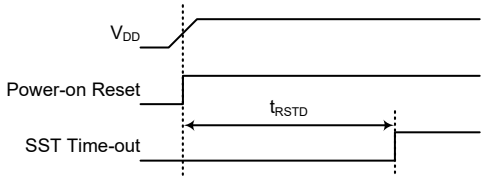
另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定的值时，系统会产生 LVR 复位。另一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。

复位功能

此单片机提供多种内部事件触发复位。

上电复位

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



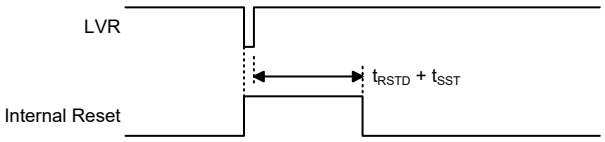
上电复位时序图

低电压复位 – LVR

单片机具有低电压复位电路，用来监测它的电源电压。在快速和低速模式下，低电压复位功能始终使能于特定的电压值，V_{LVR}。例如在更换电池的情况下，单片机供应的电压可能会在 0.9V~V_{LVR} 之间，这时 LVR 将会自动复位单片机且 RSTFC 寄存器中的 LVRF 标志位置位。有效的 LVR 信号，即在 0.9V~V_{LVR} 的低电压状态的时间，必须超过 LVD/LVR 电气特性中的 t_{LVR} 参数值。如果低电压存在时间没超过 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} 值可通过 TLVRC 寄存器中的 TLVR1~TLVR0 位设置。

V_{LVR} 参数值可通过 LVRC 寄存器中的 LVS7~LVS0 位设置。若由于受到干扰

LVS7~LVS0 变为其它值时，需经过 t_{SRESET} 时间发生复位。此时 RSTFC 寄存器的 LRF 位被置位。上电后寄存器的值为 01010101B。注意，当单片机进入空闲或休眠模式时，LVR 功能将自动除能。



低电压复位时序图

低电压复位寄存器

LVRC 和 TLVRC 寄存器用于控制低电压复位功能。

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

低电压复位寄存器列表

- **LVRC 寄存器**

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

Bit 7~0 **LVS7~LVS0**：LVR 电压选择
01010101：2.1V
00110011：2.55V
10011001：3.15V
10101010：3.8V
其它值：MCU 复位 – 寄存器复位为 POR 值
当上述定义的相应的低电压出现，且低电压保持时间超过 t_{LVR} 值，则单片机复位。实际的 t_{LVR} 值可通过 TLVRC 寄存器中的 TLVR1~TLVR0 位设置。需要经过 t_{SRESET} 时间响应复位。此时复位后的寄存器内容保持不变。
若将 LVRC 寄存器定义为其它值，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	0

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

“x”：未知

Bit 7~3　未定义，读为“0”

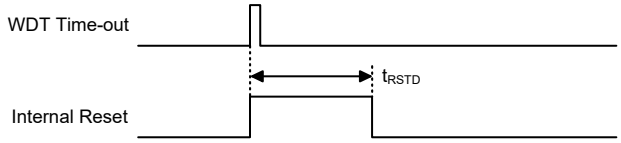
Bit 2　**LVRF**：LVR 复位标志位
0：未发生
1：发生
当特定的低电压复位条件发生时，此位被置为“1”，且只能通过应用程序清零。

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

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

正常运行时看门狗溢出复位

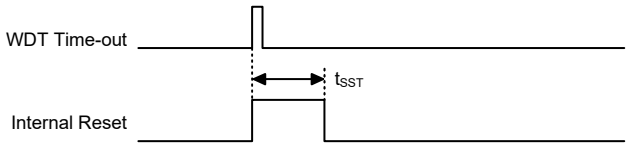
在快速和低速模式下正常运行时发生看门狗溢出复位，看门狗溢出标志位 TO 将被设为“1”。



正常运行时看门狗溢出复位时序图

休眠或空闲时看门狗溢出复位

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



休眠或空闲时看门狗溢出复位时序图

复位初始状态

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

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

“u”：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
WDT	复位后清除，WDT 重新计数
定时器	所有定时器除能
输入 / 输出口	I/O 口设置为输入类型
堆栈指针	堆栈指针指向堆栈顶端

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

寄存器	上电复位	WDT 溢出 (正常操作)	WDT 溢出 (空闲 / 休眠)
IAR0	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	---- xxxx	---- uuuu	---- 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	---- -x00	---- -uuu	---- -uuu
SCC	001- --00	001- --00	uuu- --uu
HIRCC	---- --01	---- --01	---- --uu
LVRC	0101 0101	0101 0101	uuuu uuuu
LVDC	--00 0000	--00 0000	--uu 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

寄存器	上电复位	WDT 溢出 (正常操作)	WDT 溢出 (空闲 / 休眠)
PB	---1 1111	---1 1111	---u uuuu
PBC	---1 1111	---1 1111	---u uuuu
PBPU	---0 0000	---0 0000	---u uuuu
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	uuuu uuuu
PBS0	0000 0000	0000 0000	uuuu uuuu
PBS1	---- --00	---- --00	---- --uu
WDTC	0101 0011	0101 0011	uuuu uuuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	uuuu uuuu
TMR0C	0000 1000	0000 1000	uuuu uuuu
TMR0	0000 0000	0000 0000	uuuu uuuu
TMR1C	00-0 -000	00-0 -000	uu-u -uuu
TMR1	0000 0000	0000 0000	uuuu uuuu
TMR2C	---0 -000	---0 -000	---u -uuu
TMR2	0000 0000	0000 0000	uuuu uuuu
PSCR	---- --00	---- --00	---- --uu
PCKC	-000 ---0	-000 ---0	-uuu ---u
SADO0L	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
			uuuu uuuu (ADRFS=1)
SADO0H	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
			---- uuuu (ADRFS=1)
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 0000	0000 0000	uuuu uuuu
OVPC0	000- -000	000- -000	uuu- -uuu
OVPC1	0001 0000	0001 0000	uuuu uuuu
OVPC2	---- 0000	---- 0000	---- uuuu
OVPDA	0000 0000	0000 0000	uuuu uuuu
OPC	00-- 00--	00-- 00--	uu-- uu--
OPVOS	0010 0000	0010 0000	uuuu uuuu
OPS	---0 0000	---0 0000	---u uuuu
PPGC0	0000 0000	0000 0000	uuuu uuuu
PPGC1	1000 0000	1000 0000	uuuu uuuu
PPGC2	---0 0000	---0 0000	---u uuuu
PPGTA	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGTB	xxxx xxxx	xxxx xxxx	uuuu uuuu

寄存器	上电复位	WDT 溢出 (正常操作)	WDT 溢出 (空闲 / 休眠)
PPGTC	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGTEX	-x-x -x-x	-x-x -x-x	-u-u -u-u
PWLT	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGPC	0000 0000	0000 0000	uuuu uuuu
CMP0C	0010 0000	0010 0000	uuuu uuuu
CMP1C	0001 0000	0001 0000	uuuu uuuu
CMP2C	0001 0000	0001 0000	uuuu uuuu
CMP3C	0001 0000	0001 0000	uuuu uuuu
CMPVREF0	-000 -000	-000 -000	-uuu -uuu
CMPVREF1	---- -000	---- -000	---- -uuu
CMPCTL0	00-0 0000	00-0 0000	uu-u uuuu
CMPCTL1	0000 -0-0	0000 -0-0	uuuu -u-u
CMPDBC0	--00 0000	--00 0000	--uu uuuu
CMPDBC1	000- ----	000- ----	uuu- ----
CMPHYS	---- 0000	---- 0000	---- uuuu
TLVRC	---- --00	---- --00	---- --uu
EEA	---0 0000	---0 0000	---u uuuu
EED	0000 0000	0000 0000	uuuu uuuu
INTC3	0000 0000	0000 0000	uuuu uuuu
PPGTD	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGATC0	0000 0000	0000 0000	uuuu uuuu
PPGATC1	0--0 0000	0--0 0000	u--u uuuu
PPGATC2	-000 0000	-000 0000	-uuu uuuu
PPGTMC	---0 0-00	---0 0-00	---u u-uu
PPGTMR1	0000 0000	0000 0000	uuuu uuuu
PPGTMR2	0000 0000	0000 0000	uuuu uuuu
PPGTMR3	0000 0000	0000 0000	uuuu uuuu
PPGTMRD	0000 0000	0000 0000	uuuu uuuu
ORMC	0000 0000	0000 0000	0000 0000
PPGRT	0000 0000	0000 0000	uuuu uuuu
PPGRN	-000 0000	-000 0000	-uuu uuuu
PPG2CT	0000 0000	0000 0000	uuuu uuuu
PPG2C0	0--- --00	0--- --00	u--- --uu
PPG2C1	0000 0000	0000 0000	uuuu uuuu
PPG2C2	1111 1111	1111 1111	uuuu uuuu
PPG2C3	--00 0000	--00 0000	--uu uuuu
IICC0	---- 000-	---- 000-	---- uuu-
IICC1	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA	0000 000-	0000 000-	uuuu uuu-

寄存器	上电复位	WDT 溢出 (正常操作)	WDT 溢出 (空闲 / 休眠)
IICTOC	0000 0000	0000 0000	uuuu uuuu
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
IFS	---- --00	---- --00	---- --uu
IECC	0000 0000	0000 0000	uuuu uuuu
EEC	0--- 0000	0--- 0000	u--- uuuu
C3DA	0000 0000	0000 0000	uuuu uuuu
C3LEBC	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	—	—	—	PB4	PB3	PB2	PB1	PB0
PBC	—	—	—	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	—	PBPU4	PBPU3	PBPU2	PBPU1	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。

需注意的是，当 I/O 引脚设为数字输入时，上拉功能才会受相应的上拉电阻控制寄存器控制开启，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 端口 x 引脚上拉功能控制位
0：除能
1：使能

PxPUn 位用于控制引脚上拉功能。此处的“x”可为 A 和 B。然而，每个 I/O 端口的实际有效位可能不同。

注意，上电后 PAPU 寄存器的 PAPU5 位需固定为“0”。

PA 口唤醒

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

需要注意的是，只有当引脚功能选为通用 I/O 功能输入类型且单片机处于休眠或空闲模式时，唤醒功能才会受 PAWU 控制开启，其它状态下此唤醒功能不可用。

- **PAWU 寄存器**

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

Bit 7~0 **PAWU7~PAWU0**：PA7~PA0 引脚唤醒功能控制位
0：除能
1：使能

注意，上电后 PAWU 寄存器的 PAWU5 位需固定为“0”。

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 端口 x 引脚类型选择
0：输出
1：输入

PxCn 位用于控制引脚类型选择。此处的“x”可为 A 和 B。然而，每个 I/O 端口的实际有效位可能不同。

注意，PAC 寄存器中的“D5”位必须清零，以保证在上电后相应的线为输出功能。

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单片机某些功能造成影响。然而，引脚功能共用功能通过引脚功能选择，使得小封装单片机具有更多不同的功能。该单片机具有端口“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。

需要注意的一点是要确保所需引脚共用功能被正确地选中和取消。对于多数引脚共用功能，要正确地选择所需引脚共用功能，需通过对相应的引脚共用控制寄存器正确配置来实现。接着配置相应的外围功能设定从而使能这些外围功能。但是，在设置相关引脚控制位域时，一些数字输入引脚如 TC0 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。要正确地取消选择的引脚共用功能，应先除能外围功能，接着修改相应的引脚共用功能控制寄存器以选择其它引脚共用功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	—	—	—	—	—	—	PBS11	PBS10
IFS	—	—	—	—	—	—	SCLPS	SDAPS

引脚共用功能选择寄存器列表

- **PAS0 寄存器**

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

Bit 7~6 **PAS07~PAS06**：PA3 引脚共用功能选择

00：PA3/TC0
01：CP0P
10：AN9
11：CP0P/AN9

Bit 5~4 **PAS05~PAS04**：PA2 引脚共用功能选择
00：PA2
01：OPROUT
10：AN2
11：SCL

Bit 3~2 **PAS03~PAS02**：PA1 引脚共用功能选择
00：PA1
01：OPOUT
10：AN8
11：OPOUT/AN8

Bit 1~0 **PAS01~PAS00**：PA0 引脚共用功能选择
00：PA0
01：SDA
10：PA0
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
01：CP2P
10：AN5
11：CP2P/AN5

Bit 5~4 **PAS15~PAS14**：PA6 引脚共用功能选择
00：PA6
01：CP2N
10：AN6
11：CP2N/AN6

Bit 3~2 **PAS13~PAS12**：PA5 引脚共用功能选择
00：PA5
01：保留
10：保留
11：PPG2
注意，PA5/PPG2 为内部连接信号，此位必须固定为“11”以选择 PPG2 功能。

Bit 1~0 **PAS11~PAS10**：PA4 引脚共用功能选择
00：PA4
01：C2VO
10：AN4
11：VREF

- **PBS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PBS07~PBS06**：PB3 引脚共用功能选择
00：PB3/PPGIN
01：CP0N
10：AN0
11：CP0N/AN0

Bit 5~4 **PBS05~PBS04**：PB2 引脚共用功能选择
00：PB2
01：OVPI1
10：AN1
11：OVPI1/AN1

Bit 3~2 **PBS03~PBS02**：PB1 引脚共用功能选择
00：PB1
01：PCK
10：C1VO
11：C3VO

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

- **PBS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PBS11	PBS10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

Bit 1~0 **PBS11~PBS10**：PB4 引脚共用功能选择
00：PB4
01：C0VO
10：AN3
11：SCL

- **IFS 寄存器**

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

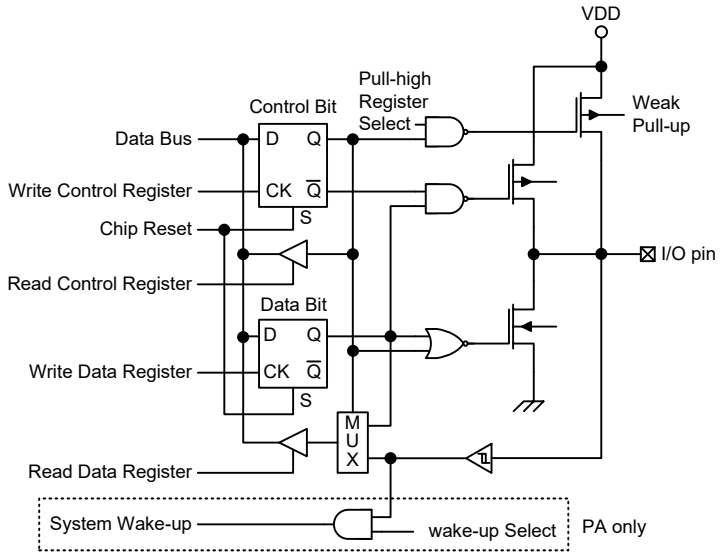
Bit 7~2 未定义，读为“0”

Bit 1 **SCLPS**：SCL 输入源引脚选择
0：PA2
1：PB4

Bit 0 **SDAPS**：SDA 输入源引脚选择
0：PA0
1：PB0

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



逻辑功能输入 / 输出结构

读端口功能

读端口功能用于读取从数据锁存器或 I/O 引脚上输出的数据，该功能专为 I/O 功能和 A/D 通道上的 IEC60730 自诊断测试而设计。寄存器 IECC 用于控制读端口功能。若此功能除能，引脚将作为所选中的共用功能引脚。若向寄存器 IECC 写入一个特定的数据模式“11001010”，内部信号 IECM 将被置高以使能读端口功能。读端口功能使能后执行读端口指令“mov acc, Px”，相应引脚上的值将被传至累加器 ACC，其中“x”代表相应的 I/O 端口名称。

- **IECC 寄存器**

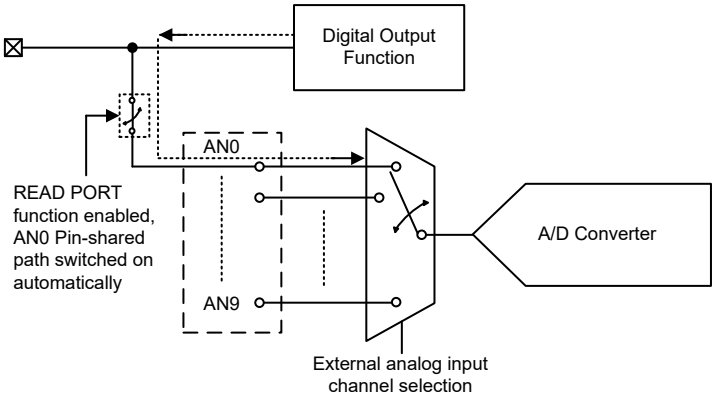
Bit	7	6	5	4	3	2	1	0
Name	IECS7	IECS6	IECS5	IECS4	IECS3	IECS2	IECS1	IECS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

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

读端口功能	除能		使能	
端口控制寄存器位 – PxC.n	1	0	1	0
I/O 功能	引脚值	数据锁存值	引脚值	
数字输入功能				
数字输出功能	0			
I^2C：SDA/SCL	引脚值			
模拟功能	0			

注：上方表格所呈现的值为执行了“mov a, Px”指令后 ACC 寄存器中的内容，其中“x”表示相关的端口名称。

读端口模式的另一个功能是检查 A/D 通道。当读端口功能除能，若相应的选择位没有选中 A/D 输入引脚功能，则从外部引脚到内部模拟输入的 A/D 通道将会被关闭。对于带 A/D 转换通道的单片机，如 A/D AN9~AN0，通过适当配置 A/D 控制寄存器中的外部模拟输入通道选择位并选中相应的模拟输入引脚功能，所需的 A/D 转换通道将被开启。而读端口模式的功能则是强制开启 A/D 通道。例如，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 转换器的参考电压应该等于 I/O 电源电压。



A/D 通道输入路径内部连接

编程注意事项

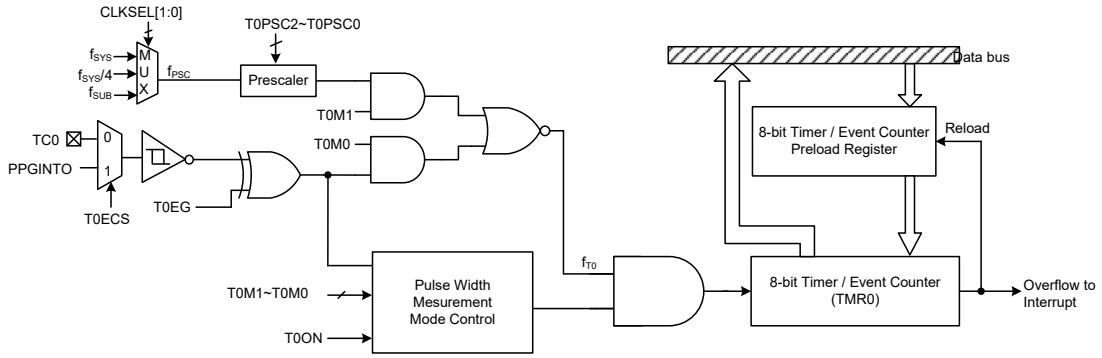
在编程中，最先要考虑的是端口的初始化。复位之后，所有的输入 / 输出数据及端口控制寄存器都将被设为逻辑高。所有输入 / 输出引脚默认为输入状态，而其电平则取决于其它相连接电路以及是否选择了上拉电阻。如果端口控制寄存器某些引脚位被设定输出状态，这些输出引脚会有初始高电平输出，除非数据寄存器端口在程序中被预先设定。设置哪些引脚是输入及哪些引脚是输出，可通过设置正确的值到适当的端口控制寄存器，或使用指令“SET [m].i”及“CLR [m].i”来设定端口控制寄存器中个别的位。注意，当使用这些位控制指令时，系统即将产生一个读 - 修改 - 写的操作。单片机需要先读入整个端口上的数据，修改个别的位，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

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

定时 / 事件计数器

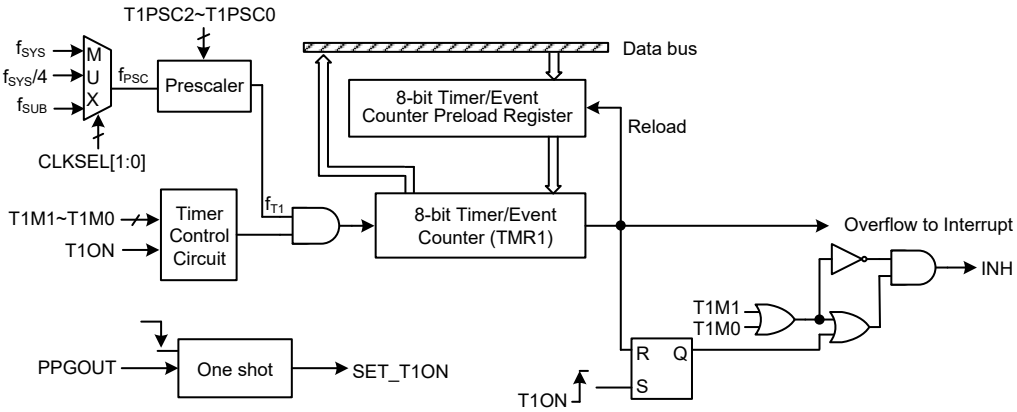
定时 / 事件计数器在任何单片机中都是一个很重要的部分，提供程序设计者一种实现和时间有关功能的方法。该芯片具有三个 8 位的向上计数器。定时 / 事件计数器 0 有三种不同的工作模式，可以当作一个普通定时器、事件计数器或脉冲宽度测量使用。定时 / 事件计数器 1 有两种不同的工作模式，可以当作一个普通定时器或模式 0 用于 PPG 不可重复触发功能。定时计数器 2 只有一种工作模式，即定时器模式，因此其没有“/ 事件”描述，模式选择位和相关描述。该单片机提供了一个内部时钟分频器，以扩大定时器的范围。

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



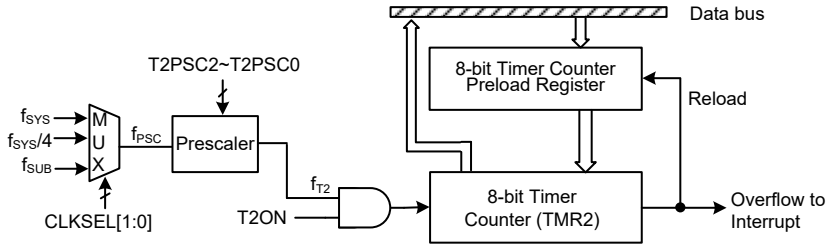
注：PPGINTO 信号来自 PPGIN 引脚或比较器 0 输出“C0VO”同相或反相去抖的信号，可通过软件选择。

8-bit 定时 / 事件计数器 0



注：INH 输出信号用于禁止 PPG 被再次触发。

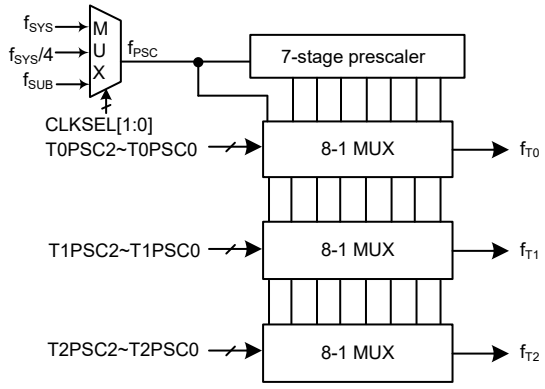
8-bit 定时 / 事件计数器 1



8-bit 定时计数器 2

配置定时 / 事件计数器输入时钟源

该定时 / 事件计数器 n 时钟 f_{Tn} 来自内部时钟 f_{PSC}，其来自内部系统时钟 f_{SYS}，$f_{SYS}/4$ 或 f_{SUB}，通过 PSCR 寄存器中的 CLKSEL[1:0] 位选择。然后 f_{PSC} 时钟由分频器分频，分频比由 TMRnC 定时器中的 TnPSC2~TnPSC0 位来选择。对于定时器 / 事件计数器 0，时钟源可以来自外部或内部，当选择来自于外部时，时钟源可以选择来自于 TC0 引脚或 PPGINTO 信号，通过 TMR0C 寄存器中的 T0ECS 位选择。外部信号输入可以用来计数事件、测量时间间隔或测量脉冲宽度，内部时钟可以用来产生精确的时基信号。



预分频器

- **PSCR 寄存器**

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

Bit 7~2　未定义，读为“0”

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

定时 / 事件计数寄存器 – TMR0, TMR1, TMR2

定时 / 事件计数寄存器 TMRn 是位于特殊功能数据存储器内的特殊功能寄存器，用于储存定时器的当前值。在用作内部定时且收到一个内部计数脉冲或用作外部计数且外部定时 / 事件计数器引脚发生状态跳变时，此寄存器的值将会加一。定时器将从预置寄存器所载入的值开始计数，到 FFH 时定时器溢出且会产生一

个内部中断信号。定时器的值随后被预置寄存器的值重新载入并继续计数。

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

- **TMRn 寄存器 (n=0~2)**

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**：定时 / 事件计数器 n 预置寄存器 bit 7~ bit 0

定时 / 事件计数器控制寄存器 – TMR0C, TMR1C, TMR2C

Holtek 单片机灵活的特性也表现在定时器的多功能上，定时 / 事件计数器能提供几种不同的工作模式，由相应的控制寄存器来选择定时 / 事件计数器的工作方式。定时 / 事件计数器控制寄存器为 TMRnC，配合相应的 TMRn 寄存器控制定时 / 事件计数器的全部操作。在使用定时器之前，需要先正确地设定定时 / 事件计数器控制寄存器，以便保证定时器能正确操作，而这个过程通常在程序初始化期间完成。

定时 / 事件计数控制寄存器的第 7 位和第 6 位，即 T0M1/T0M0 和 T1M1/T1M0，用来设定定时器的工作模式，计数器模式、定时器模式、脉冲宽度测量模式或是用于 PPG 不可重复触发功能的模式 0。定时 / 计数控制寄存器的第 4 位即 TnON，用于定时器开关控制，设定为逻辑高时，计数器开始计数，而清零时则停止计数。定时 / 事件计数器控制寄存器的第 0~2 位用来控制输入时钟预分频器。如果使用外部时钟源，预分频器位将不起作用。如果定时 / 事件计数器工作在事件计数模式或脉冲宽度测量模式，T0EG 位即 TMR0C 寄存器的第 3 位将可用来选择上升沿或下降沿触发。

- **TMR0C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	T0ECS	T0ON	T0EG	T0PSC2	T0PSC1	T0PSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7~6 **T0M1~T0M0**：定时 / 事件计数器 0 工作模式选择
00：无可用模式
01：事件计数器模式
10：定时器模式
11：脉冲宽度测量模式

Bit 5 **T0ECS**：定时 / 事件计数器 0 外部时钟源选择
0：TC0
1：PPGINTO

Bit 4 **T0ON**：定时 / 事件计数器 0 使能
0：除能
1：使能

Bit 3 **T0EG**：定时 / 事件计数器 0 有效边沿选择
事件计数器模式：
0：在上升沿计数
1：在下降沿计数
脉冲宽度测量模式：
0：在下降沿启动计数，在上升沿停止计数
1：在上升沿启动计数，在下降沿停止计数

Bit 2~0 **T0PSC2~T0PSC0**：定时 / 事件计数器 0 预分频比选择
定时 / 事件计数器 0 内部时钟 f_{T0}=
000：f_{PSC}
001：$f_{PSC}/2$
010：$f_{PSC}/4$
011：$f_{PSC}/8$
100：$f_{PSC}/16$
101：$f_{PSC}/32$
110：$f_{PSC}/64$
111：$f_{PSC}/128$

- **TMR1C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	T1M1	T1M0	—	T1ON	—	T1PSC2	T1PSC1	T1PSC0
R/W	R/W	R/W	—	R/W	—	R/W	R/W	R/W
POR	0	0	—	0	—	0	0	0

Bit 7~6 **T1M1~T1M0**：定时 / 事件计数器 1 工作模式选择
00：模式 0 (PPG 不可重复触发功能)
01：无可用模式
10：定时器模式
11：无可用模式

Bit 5 未定义，读为“0”

Bit 4 **T1ON**：定时 / 事件计数器 1 使能
0：除能
1：使能
为避免 PPG 异常操作，在模式 0 下时不能通过应用程序修改此位。

Bit 3 未定义，读为“0”

Bit 2~0 **T1PSC2~T1PSC0**：定时 / 事件计数器 1 预分频比选择
定时 / 事件计数器内部时钟 f_{T1}=
000：f_{PSC}
001：$f_{PSC}/2$
010：$f_{PSC}/4$
011：$f_{PSC}/8$
100：$f_{PSC}/16$
101：$f_{PSC}/32$
110：$f_{PSC}/64$
111：$f_{PSC}/128$

- **TMR2C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	T2ON	—	T2PSC2	T2PSC1	T2PSC0
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　**T2ON**：定时计数器 2 使能
0：除能
1：使能

Bit 3　未定义，读为“0”

Bit 2~0　**T2PSC2~T2PSC0**：定时计数器 2 预分频比选择
定时计数器内部时钟 f_{T2}=
000：f_{PSC}
001：$f_{PSC}/2$
010：$f_{PSC}/4$
011：$f_{PSC}/8$
100：$f_{PSC}/16$
101：$f_{PSC}/32$
110：$f_{PSC}/64$
111：$f_{PSC}/128$

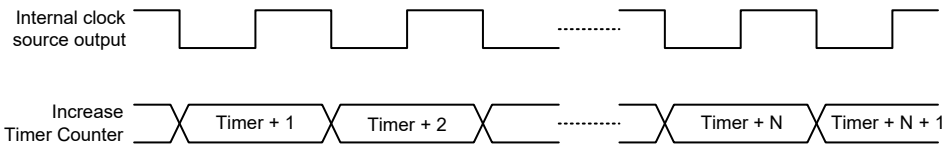
定时 / 事件计数器工作模式

定时器 / 事件计数器可以操作于不同的操作模式，定时器模式、事件计数器模式，脉冲宽度测量模式或用于 PPG 不可重复触发功能的模式 0。通过设置 TMRnC 寄存器中的 TnM1/TnM0 位选择任意工作模式。

定时器模式

为使定时 / 事件计数器 n 工作在定时器模式，TMRnC 寄存器中的 TnM1~TnM0 位必须分别设置为“10”。在这个模式下，定时器可以用来测量固定时间间隔，当定时器发生溢出时，就会产生一个内部中断信号。

在定时器模式中，定时器内部时钟 f_{Tn} 来自内部时钟 f_{PSC}，其来自内部系统时钟 f_{SYS}，$f_{SYS}/4$ 或 f_{SUB}，通过 PSCR 寄存器中的 CLKSEL[1:0] 位选择。然而，该时钟源可以被预分频器进一步分频，分频比是由 TMRnC 寄存器的 TnPSC2~TnPSC0 位来确定。定时器控制寄存器的 TnON 位需要设为逻辑高，才能令定时器工作。每次内部时钟由高到低的电平转换都会使定时器值增加一。当定时器计数已满即溢出时，会产生中断信号且定时器会重新载入预置寄存器的值，然后继续计数。定时器溢出以及相应的内部中断产生也是唤醒休眠或空闲模式的一种方法。通过设置相应的中断控制寄存器中的定时 / 事件计数器 n 中断使能位为 0，可以禁止计数器中断。



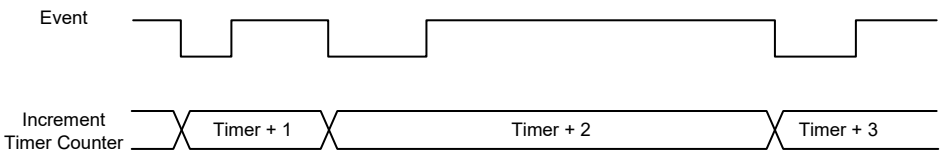
定时器模式时序图

事件计数器模式

仅定时 / 事件计数器 0 具有事件计数模式，可以通过定时 / 事件计数器 0 来记录发生在 TC0 引脚或 PPGINTO 信号的逻辑事件变化的次数。为使定时 / 事件计数器 0 工作在事件计数模式，工作模式选择位 T0M1/T0M0 必须分别设置为 0 和 1。

在事件计数模式，外部定时脚 TC0 或 PPGINTO 信号被用来当定时 / 计数器 0 的时钟源且不被内部预分频器进一步分频。在设置完定时 / 事件计数器控制寄存器其它位，定时 / 事件计数器 0 控制寄存器第 4 位，即 T0ON 位需要设为逻辑高，才能使计数器工作。当定时控制寄存器第 3 位，即 T0EG 设置为逻辑低时，每次 TC0 引脚或 PPTINTO 信号上接收到由低到高电平的转换将使计数器加一。而当 T0EG 为逻辑高时，每次 TC0 引脚或 PPTINTO 信号上接收到由高到低电平的转换将使计数器加一。当计数器计数满，即溢出时会产生中断信号且计数器会重新加载预置寄存器的值，然后继续计数。计数器溢出中断可通过设置相应的中断控制寄存器中的定时 / 事件计数器 0 中断使能位为 0 而禁止。

由于外部定时器引脚和普通输入 / 输出引脚共用，为了确保工作在事件计数模式，要注意两点。首先是要将定时 / 事件计数器 0 的工作模式设定在事件计数模式，其次是确定端口控制寄存器将这个引脚设定为输入状态。值得注意的是，在事件计数模式下，当单片机工作在休眠或是空闲模式时也保持对 TC0 引脚或 PPTINTO 信号的事件计数功能。当计数器溢出时，将产生一个定时器中断，并且可以作为唤醒休眠或空闲模式的一种方法。



事件计时器模式时序图 (T0EG=1)

脉冲宽度测量模式

仅定时 / 事件计数器 0 具有脉冲宽度测量模式，可以测量 TC0 引脚或 PPTINTO 信号上的脉冲宽度。为使定时 / 事件计数器 0 工作在脉冲宽度测量模式，工作模式选择位 T0M1~T0M0 必须分别设置为“11”。

在脉冲宽度测量模式中，TC0 引脚或 PPTINTO 信号被用来当定时 / 计数器 0 的时钟源且不被内部预分频器进一步分频。在设置完定时 / 事件计数器控制寄存器其它位，定时器控制寄存器第 4 位，即 T0ON 位需要设为逻辑高，才能使定时 / 计数器工作。然而，只有当在 TC0 引脚或 PPTINTO 信号上接收到有效的逻辑转换时，定时 / 事件计数器才真正开始启动计数。

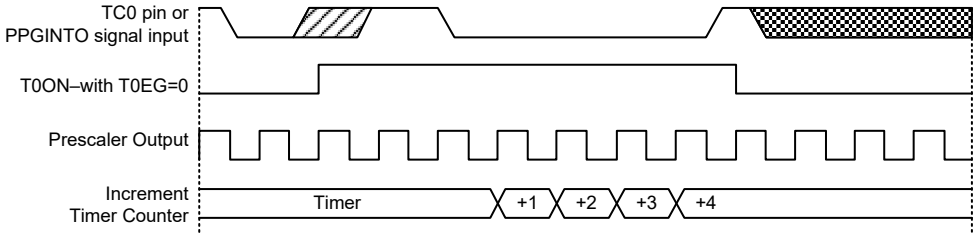
当定时控制寄存器第 3 位，即 T0EG 设置为逻辑低时，每次 TC0 引脚或 PPTINTO 信号接收到由高到低电平的转换时将开始计数直到 TC0 引脚或 PPTINTO 信号回到它原来的高电平。此时使能位将自动清除为 0 以停止计数。而当 T0EG 为逻辑高时，每次 TC0 引脚或 PPTINTO 信号接收到由低到高电平的转换时将开始计数直到 TC0 引脚或 PPTINTO 信号回到它原来的低电平。同样使能位将自动清除为 0 以停止计数。注意，在脉冲宽度测量模式中，当 TC0 引脚或 PPTINTO 信号回到它原来的电平时，使能位将自动地清除为 0。而在其它模式，使能位只能在程序控制下清除为 0。

可以通过程序读取定时 / 事件计数器剩余值，捕捉 TC0 引脚或 PPTINTO 信号上脉冲宽度。当使能位重新复位，任何出现在 TC0 引脚或 PPTINTO 信号的脉冲将被忽略。直到使能位被程序重新置高，开始重新测量脉冲。这种方式使得

测量单次脉冲将会很容易实现。

注意，在这种模式下，定时 / 事件计数器是通过 TC0 引脚或 PPTINTO 信号上的逻辑转换来控制，而不是通过逻辑电平。当定时 / 事件计数器计满，即溢出时会产生中断信号且定时 / 计数器会重新加载预置寄存器的值，然后继续向上计数。定时 / 事件计数器溢出中断可通过设置相应的中断寄存器中的定时 / 事件计数器 0 使能位为 0 而禁止。

由于 TC0 引脚和普通输入 / 输出引脚共用，为了确保工作在脉冲宽度测量模式，要注意两点。首先是要将定时 / 计数器 0 的工作模式设定在脉冲宽度测量模式，其次是确定端口控制寄存器将这个引脚设定为输入状态。



脉冲宽度测量模式的时序图 (T0EG=0)

模式 0

仅定时 / 事件计数器 1 具有模式 0，可以实现 PPG 不可重复触发功能。为使定时 / 事件计数器 1 工作在模式 0，工作模式选择位 T1M1/T1M0 必须分别设置为“00”。

在模式 0 中，定时 / 事件计数器 1 在 PPG 停止时开始计数，溢出时停止计数。也就是说一旦 PPG 停止，T1ON 将被置位，溢出将被清零。同其它模式一样，当定时 / 事件计数器 1 计满，即溢出时会产生中断信号且定时 / 事件计数器 1 会重新加载预置寄存器的值。定时 / 事件计数器溢出中断可通过设置相应的中断控制寄存器中的定时 / 计数器 1 使能位为 0 而禁止。

输入 / 输出接口

当定时 / 事件计数器 0 运行在事件计数或者脉冲宽度测量模式下，定时 / 事件计数器可使用外部定时器引脚以确保正确的动作。计时器 / 事件计数器 0 通过将 T0ECS 位清除为零来选择 TC0 引脚作为时钟源。由于 TC0 引脚是共用引脚，因此需要正确的将其配置为定时 / 事件计数器输入引脚。这可以通过定时 / 事件计数器控制寄存器的模式选择位来选择是事件计数模式或者脉冲宽度测量模式。此外，相应的端口控制寄存器位需要被设置为高，来确保该引脚是作为输入脚。即使该引脚用作定时 / 事件计数器输入，任何连接到这个引脚的上拉电阻将仍然是有效的。

编程注意事项

当定时 / 事件计数器工作在定时器模式时，内部的系统时钟作为定时器的时钟源，因此与单片机所有操作都能同步。在这个模式下，当定时器寄存器溢出时，单片机将产生一个内部中断信号，使程序进入相应的内部中断向量。对于脉冲宽度测量模式，定时器时钟源同样使用内部的系统时钟，然而，只有正确的逻辑条件出现在 TC0 引脚或 PPTINTO 信号上时，定时器才开始运行。当这个事件没有和内部定时器时钟同步时，只有当下一个定时器时钟到达时，单片机才会看到这个事件，因此在测量值上可能有小的差异，需要程序设计者在程序应用时加以注意。同样的情况发生在定时器设置为事件计数模式时，它的时钟来

源是事件，与内部系统时钟或者定时器时钟不同步。

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

当定时 / 计数器产生溢出，中断控制寄存器中相应的中断请求标志将置位。若中断允许，将会依次产生一个中断信号。不管中断是否允许，在休眠或空闲模式下，定时 / 事件计数器的溢出也会产生唤醒。这种情况可能发生在 TC0 引脚或 PPGINTO 信号变化的计数模式中。定时 / 计数器向上计数直至溢出并唤醒系统。若在休眠或空闲模式下，不需要定时器中断唤醒系统，可以在执行“HALT”指令之前将相应中断请求标志位置位。

定时器应用范例

这个例子说明了如何设置定时 / 事件计数器的寄存器，如何设置和控制中断。另外还需注意的是，怎样通过寄存器的第 4 位来启动 / 停止定时 / 事件计数器。此应用范例设置定时 / 事件计数器 0 为定时模式，时钟来源于内部的系统时钟。

定时器编程范例

```
org 0ch             ; PPG INT00 interrupt vector
org 10h             ; Timer Counter 0 interrupt vector
jmp tmr0int         ; jump here when Timer 0 overflows
:
org 20h             ; main program
:
                    ; internal Timer 0 interrupt routine
tmr0int:
:
                    ; Timer 0 main program placed here
begin:
                    ; setup Timer 0 registers
mov a, 09bh         ; setup Timer 0 preload value
mov tmr0, a
mov a, 081h         ; setup Timer 0 control register
mov tmr0c, a        ; timer mode and prescaler set to /2
                    ; setup interrupt register
mov a, 001h         ; enable master interrupt and both timer interrupts
mov intc0, a
mov a, 001h
mov intc1, a
:
set tmr0c.4         ; start Timer 0
```

A/D 转换器

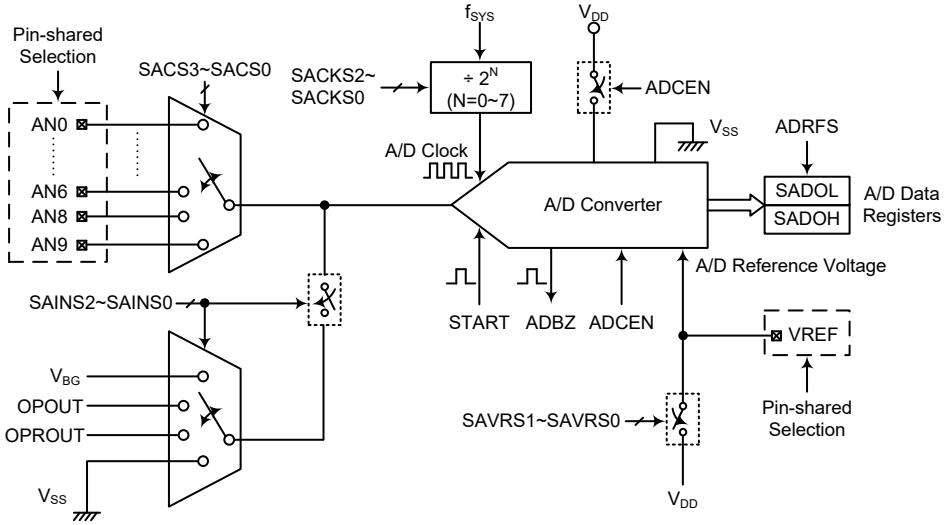
对于大多数电子系统而言，处理现实世界的模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先需要通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

A/D 转换器简介

该单片机包含了多通道的 A/D 转换器，它们可以直接接入外部模拟信号，如来自传感器或其它控制信号，并直接将这些信号转换成 12-bit 数字量。也可对内部模拟信号，如 Bandgap 参考电压 V_{BG}、运算放大器输出信号 OPOUT、运算放大器输出 OPROUT 或 A/D 转换器负电源电压 V_{SS} 进行 A/D 转换。选择转换外部或内部模拟信号由 SAINS2~SAINS0 位和 SACS3~SACS0 位共同控制。若要转换外部模拟信号，首先应正确设置好相应的共用引脚控制位，然后再通过 SAINS2~SAINS0 位和 SACS3~SACS0 位选择所需的外部输入通道。注意，若要转换内部模拟信号，SAINS2~SAINS0 位和 SACS3~SACS0 位应正确设置。关于 A/D 输入信号的详细描述请参考“A/D 转换器控制寄存器”和“A/D 转换器输入信号”两节内容。

外部输入通道	内部输入信号	A/D 输入选择位
9：AN0~AN6, AN8~AN9	4：V_{BG}, OPOUT, OPROUT, V_{SS}	SAINS2~SAINS0，SACS3~SACS0

下图显示了 A/D 转换器内部结构和相关的寄存器。



A/D 转换器结构

A/D 转换寄存器介绍

A/D 转换器的所有操作由一系列寄存器控制。一对只读寄存器来存放 12 位 A/D 转换数据的值。剩下两个控制寄存器设置 A/D 转换器的操作和控制功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

A/D 转换器寄存器列表

A/D 转换器数据寄存器 – SADOL, SADOH

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

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

A/D 数据寄存器

A/D 转换器控制寄存器 – SADC0, SADC1

寄存器 SADC0 和 SADC1 用来控制 A/D 转换器的功能和操作。这些 8 位的寄存器定义包括选择连接至内部 A/D 转换器的模拟通道，数字化数据格式，A/D 时钟源，并控制和监视 A/D 转换器的忙碌状态。由于每个单片机只包含一个实际的模数转换电路，因此这些外部和内部模拟信号中的每一个都需要分别被发送到转换器。SADC0 寄存器中的 SACS3~SACS0 位用于选择哪个外部模拟输入通道被连接到内部 A/D 转换器。SADC1 寄存器中的 SAINS2~SAINS0 位用于选择外部模拟输入通道或内部模拟信号被连接到内部 A/D 转换器。

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

- **SADC0 寄存器**

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

Bit 7 **START**：启动 A/D 转换位
0→1→0：启动
此位用于启动 A/D 转换过程。通常此位为低，但如果设为高再被清零，将启动 A/D 转换过程。

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

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

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

Bit 3~0 **SACS3~SACS0**：A/D 外部模拟通道输入选择位
0000：AN0
0001：AN1
：
0110：AN6
0111：保留，不能使用
1000：AN8
1001：AN9
1010~1111：未定义，输入浮空

- **SADC1 寄存器**

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

Bit 7~5 **SAINS2~SAINS0**：A/D 输入信号选择位
000：外部信号源 – 外部模拟通道输入，ANn
001：内部信号源 – 内部 Bandgap 参考电压，V_{BG}
010：内部信号源 – 运算放大器输出，OPOUT
011：内部信号源 – 运算放大器输出，OPROUT
100：内部信号源 – A/D 转换器负电源电压，V_{SS}
101~111：外部信号源 – 外部模拟通道输入，ANn
必须注意当 SAINS2~SAINS0 位为“001”~“100”选择转换内部模拟信号时，外部输入通道一定不能作为 A/D 输入，SACS3~SACS0 位需正确设置为“1010”~“1111”中的一个值。否则，外部输入通道将与内部模拟信号相连接，这将导致不可预期的后果，甚至不可逆的损坏。

Bit 4~3 **SAVRS1~SAVRS0**：A/D 转换器参考电压选择位
00：外部 VREF 引脚
01：内部 A/D 转换器电源 V_{DD}
10/11：外部 VREF 引脚
这几位用于选择 A/D 转换器的参考电压。必须注意当 SAVRS1~SAVRS0 为“01”选择内部 A/D 转换器电源作为参考电压时，需正确的设置相应的共用引脚功能控制位，不能将 VREF 引脚设置为参考电压输入。否则，VREF 引脚的外部输入电压也会连接到内部 A/D 转换器电源，这将导致无法预期的结果。

Bit 2~0 **SACKS2~SACKS0**：A/D 时钟源选择位
000：f_{SYS}
001：$f_{SYS}/2$
010：$f_{SYS}/4$

011：$f_{SYS}/8$
100：$f_{SYS}/16$
101：$f_{SYS}/32$
110：$f_{SYS}/64$
111：$f_{SYS}/128$

A/D 转换器操作

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

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

A/D 转换器的时钟源为系统时钟 f_{SYS} 或其分频，而分频系数由 SADC1 寄存器中的 SACKS2~SACKS0 位决定。虽然 A/D 时钟源是由系统时钟 f_{SYS} 和 SACKS2~SACKS0 位决定，但可选择的最大 A/D 时钟源则有一些限制。由于允许的 A/D 时钟周期 t_{ADCK} 的范围为 0.5μs~10μs，所以选择系统时钟速度时就必须小心。如果系统时钟速度为 8MHz 时，SACKS2~SACKS0 位不能设为“000”，“001”或“111”。必须保证设置的 A/D 转换时钟周期不小于时钟周期的最小值或不大于时钟周期的最大值，否则将会产生不准确的 A/D 转换值。使用者可以参考下面的表格，被标上星号 * 的数值是不允许的，因为它们超出了 A/D 转换时钟周期规定的范围。

f_{SYS}	A/D 时钟周期 (t_{ADCK})							
	SACKS[2:0] =000 (f_{SYS})	SACKS[2:0] =001 ($f_{SYS}/2$)	SACKS[2:0] =010 ($f_{SYS}/4$)	SACKS[2:0] =011 ($f_{SYS}/8$)	SACKS[2:0] =100 ($f_{SYS}/16$)	SACKS[2:0] =101 ($f_{SYS}/32$)	SACKS[2:0] =110 ($f_{SYS}/64$)	SACKS[2:0] =111 ($f_{SYS}/128$)
1MHz	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *	128μs *
2MHz	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *
4MHz	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *
8MHz	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *
16MHz	62.5ns *	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs

A/D 时钟周期范例

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

A/D 转换器参考电压

A/D 转换器参考电压可以来自 A/D 转换器电源 V_{DD} 或外部参考源引脚 VREF，可通过 SAVRS1~SAVRS0 位来选择。当 SAVRS1~SAVRS0 位为“01”时，A/D 转换器参考电压来自 V_{DD}。否则，当 SAVRS1~SAVRS0 位为“01”以外任何值时，A/D 转换器参考电压来自 VREF 引脚。由于 VREF 引脚与其它功能共用引脚，当该引脚被选作参考电压引脚时，相应的引脚共用功能选择位应被合理设置以除能其它引脚共用功能。然而，若选择 A/D 转换器电源作为 A/D 转换器的

参考电压，VREF 引脚不能配置为参考电压输入功能，以避免 VREF 引脚与 A/D 转换器电源 V_{DD} 内部相连接。模拟输入值一定不能超过所选的参考电压值。

A/D 转换器输入信号

所有的 A/D 外部模拟输入引脚都与 I/O 口及其它功能共用。使用 PxS0 和 PxS1 寄存器中的相应位，可以将它们设置为A/D转换器模拟输入脚或其它共用功能。如果对应的引脚作为 A/D 转换输入，那么它原来的引脚功能将除能。通过这种方式，引脚的功能可由程序来控制，灵活地切换引脚功能。如果将引脚设为 A/D 输入，则通过寄存器编程设置的所有上拉电阻会自动断开。请注意，端口控制寄存器不需要为使能 A/D 输入而先设定为输入模式，当 A/D 输入功能选择位使能 A/D 输入时，端口控制寄存器的状态将被重置。

A/D 转换器还有三个内部模拟信号，来自 Bandgap 参考电压、运算放大器输出 OPOUT 和运算放大器输出 OPROUT，可通过设置 SAINS2~SAINS0 位将其连接到 A/D 转换器作为模拟输入信号。如果选择外部输入通道，SAINS2~SAINS0 位应设为“000”或“101~111”，具体外部通道编号由 SACS3~SACS0 位决定。如果选择内部模拟信号，那么必须适当地设置 SACS3~SACS0 位为 1010~1111 中的一个值，将外部输入通道切换到无 A/D 输入通道状态。否则，外部输入通道将与内部模拟信号相连接，这将导致不可预期的后果。

SAINS[2:0]	SACS[3:0]	输入信号	描述
000, 101~111	0000~0110, 1000~1001	AN0~AN6, AN8~AN9	外部模拟通道输入
	1010~1111	—	输入浮空，未选择外部通道
001	1010~1111	V_{BG}	内部 Bandgap 参考电压
010	1010~1111	OPOUT	运算放大器输出
011	1010~1111	OPROUT	运算放大器输出
100	1010~1111	V_{SS}	A/D 转换器负电源电压

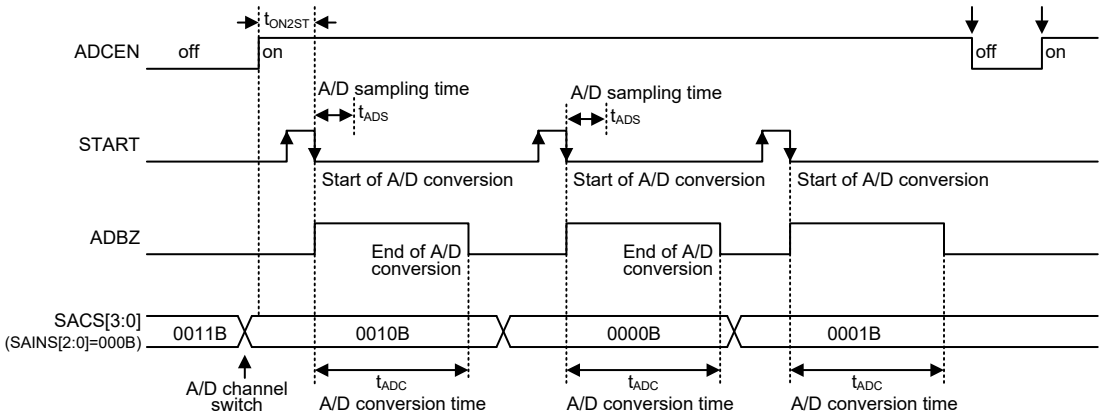
A/D 转换器输入信号选择

A/D 转换率及时序图

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

最大 A/D 转换率 = 1/(A/D 时钟周期 × 16)

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



A/D 转换时序图 – 外部通道输入

A/D 转换步骤

下面概述实现 A/D 转换过程的各个步骤。

- 步骤 1
  通过 SADC1 寄存器中的 SACKS2~SACKS0 位，选择所需的 A/D 转换时钟。
- 步骤 2
  将 SADC0 寄存器中的 ADCEN 位置高使能 A/D 转换器。
- 步骤 3
  通过 SADC1 寄存器中的 SAINS2~SAINS0 位，选择连接至内部 A/D 转换器的信号。
  若选择外部通道输入，接着执行步骤 4。
  若选择内部模拟信号，接着执行步骤 5。
- 步骤 4
  若已通过 SAINS2~SAINS0 位选择 A/D 输入信号来自外部通道输入，接着应设置相关的引脚共用控制位将该引脚规划为 A/D 输入引脚。通过设置 SACS3~SACS0 位选择哪个外部通道接至 A/D 转换器。接着执行步骤 6。
- 步骤 5
  在 SAINS2~SAINS0 位选择 A/D 输入信号来自内部模拟信号之前，需设置 SACS3~SACS0 为 1010~1111 中的一个值，将外部通道输入切换到无通道输入，然后再通过 SAINS2~SAINS0 位选择内部模拟信号。接着执行步骤 6。
- 步骤 6
  通过 SADC1 寄存器中的 SAVRS1~SAVRS0 位选择参考电压。
- 步骤 7
  设置 SADC0 寄存器中的 ADRFS 位选择 A/D 转换器输出数据格式。
- 步骤 8
  如果要使用中断，则中断控制寄存器需要正确地设置，以确保 A/D 中断功能是激活的。总中断控制位 EMI 以及 A/D 转换器中断位 ADE 需要置位为“1”。
- 步骤 9
  现在可以通过设置 SADC0 寄存器中的 START 位从“0”到“1”再回到“0”，开始模数转换的过程。

- 步骤 10

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

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

编程注意事项

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

A/D 转换功能

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

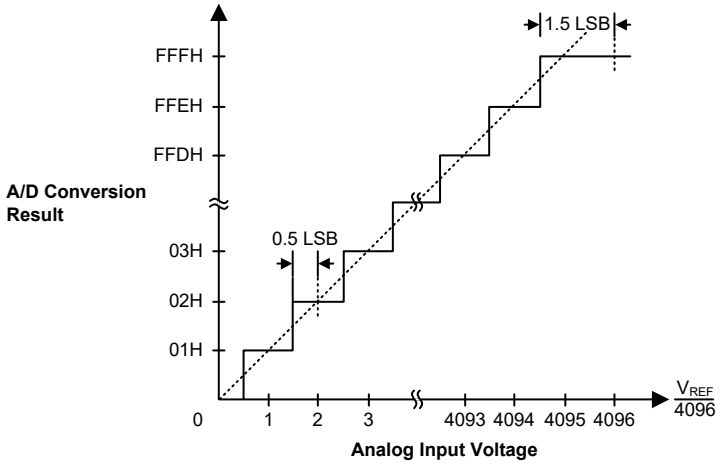
$$1\ LSB=V_{REF} \div 4096$$

通过下面的等式可估算 A/D 转换器输入电压值：

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

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

注意，这里的 V_{REF} 电压指代的是通过 SAVRS1~SAVRS0 位选择的实际 A/D 转换器参考电压。



理想的 A/D 转换功能

A/D 转换应用范例

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

范例 1：使用查询 ADBZ 的方式来检测转换结束

```
clr  ADE                    ; disable A/D converter interrupt
mov  a,03h                  ; select fSYS/8 as A/D clock and select external
                            ; channel input and external reference input
mov  SADC1,a
mov  a,80h                  ; setup PBS0 to configure pin AN0
mov  PBS0,a
mov  a,20h
mov  SADC0,a                ; enable A/D and connect AN0 channel to A/D
                            ; converter
:
start_conversion:
clr  START                  ; high pulse on start bit to initiate conversion
set  START                  ; reset A/D
clr  START                  ; start A/D
polling_EOC:
sz   ADBZ                   ; poll the SADC0 register ADBZ bit to detect end
                            ; of A/D conversion
jmp  polling_EOC            ; continue polling
mov  a,SADOL                ; read low byte conversion result value
mov  SADOL_buffer,a         ; save result to user defined register
mov  a,SADOH                ; read high byte conversion result value
mov  SADOH_buffer,a         ; save result to user defined register
:
:
jmp  start_conversion       ; start next A/D conversion
```

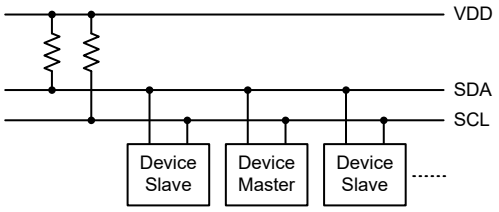
范例 2：使用中断的方式来检测转换结束

```
clr  ADE                    ; disable ADC interrupt
mov  a,03H
mov  SADC1,a                ; select fSYS/8 as A/D clock and select external
                            ; channel and external reference input
set  ADCEN
mov  a,80h                  ; setup PBS0 to configure pin AN0 and pin VREF
mov  PBS0,a
mov  a,20h
mov  SADC0,a                ; enable A/D converter and connect AN0 channel
                            ; to A/D converter
Start_conversion:
clr  START                  ; high pulse on START bit to initiate conversion
set  START                  ; reset A/D
clr  START                  ; start A/D
clr  ADF                    ; clear ADC interrupt request flag
set  ADE                    ; enable ADC interrupt
set  EMI                    ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov  acc_stack,a            ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a         ; save STATUS to user defined memory
:
:
mov  a,SADOL                ; read low byte conversion result value
mov  SADOL_buffer,a         ; save result to user defined register
mov  a,SADOH                ; read high byte conversion result value
```

```
mov  SADOH_buffer,a          ; save result to user defined register
:
:
EXIT_INT_ISR:
mov  a,status_stack
mov  STATUS,a                ; restore STATUS from user defined memory
mov  a,acc_stack             ; restore ACC from user defined memory
reti
```

I²C 接口

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

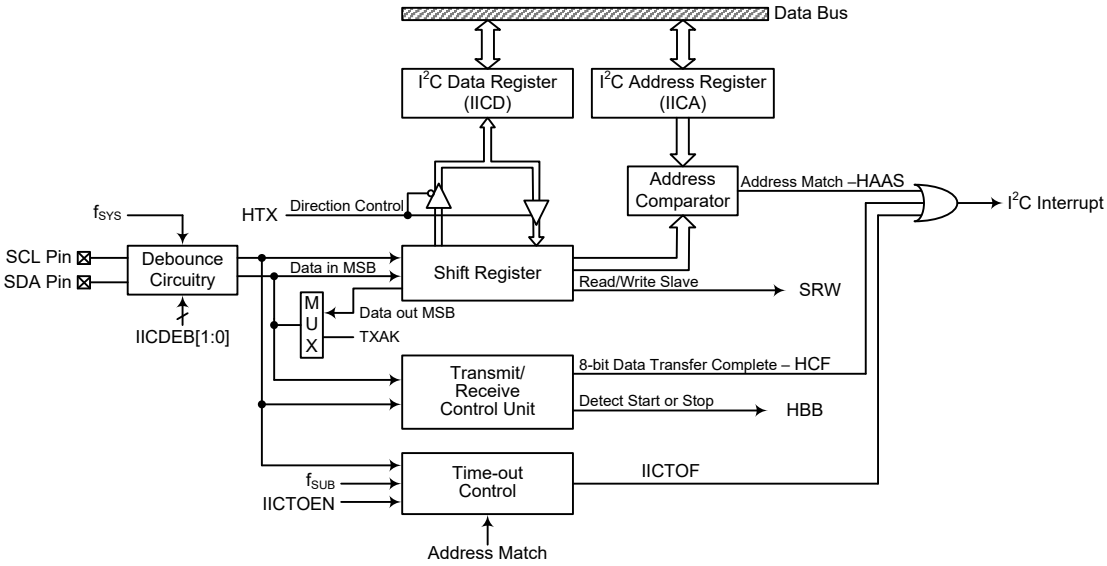


I²C 主从总线连接图

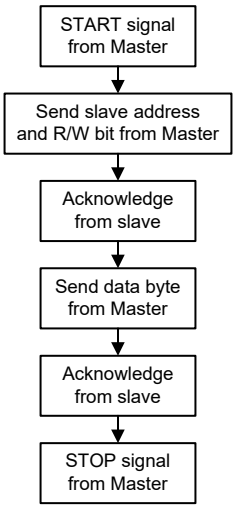
I²C 接口操作

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

如果有两个设备通过双向的 I²C 总线进行通信，那么就存在一个主机和一个从机。主机和从机都可以用于传输和接收数据，但只有主机才可以控制总线动作。那些处于从机模式的设备，要在 I²C 总线上传输数据只有两种方式，一是从机发送模式，二是从机接收模式。即使 I²C 设备被激活，与 SCL/SDA 引脚共用的 I/O 口上拉电阻控制功能仍有效，其上拉电阻功能由相应的上拉电阻控制寄存器控制。



I²C 方框图



I²C 接口操作

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

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

I²C 最小 f_{SYS} 频率要求

I²C 寄存器

I²C 总线有三个控制寄存器 IICC0、IICC1 和 IICTOC，一个地址寄存器 IICA 以及一个数据寄存器 IICD。

寄存器名称	位							
	7	6	5	4	3	2	1	0
IICC0	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
IICC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
IICD	D7	D6	D5	D4	D3	D2	D1	D0
IICA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
IICTOC	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0

I²C 寄存器列表

I²C 数据寄存器

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

- **IICD 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**：I²C 数据寄存器位 bit 7 ~ bit 0

I²C 地址寄存器

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

- **IICA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	0	0	0	0	0	0	0	—

Bit 7~1 **IICA6~IICA0**：I²C 从机地址位
IICA6~IICA0 是从机地址 bit 6 ~ bit 0。

Bit 0 未定义，读为“0”

I²C 控制寄存器

单片机中有三个控制 I²C 接口功能的寄存器，IICC0、IICC1 和 IICTOC。寄存器 IICC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 IICC1 包括多个用于指示 I²C 传输状态的相关标志位。IICTOC 寄存器用于控制 I²C 超时功能，此寄存器在 I²C 超时一节介绍。

- **IICC0 寄存器**

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

Bit 7~4　未定义，读为“0”

Bit 3~2　**IICDEB1~IICDEB0**：I²C 去抖时间选择位
00：无去抖时间
01：2 个系统时钟去抖时间
10/11：4 个系统时钟去抖时间
需注意的是，如果系统时钟 f_{SYS} 来自 f_H 时钟或 IAMWU 等于 0，I²C 去抖电路将正常工作。否则去抖电路将不受影响而被忽视。

Bit 1　**IICEN**：I²C 使能控制位
0：除能
1：使能
此位为 I²C 接口的开 / 关控制位。此位为“0”时，I²C 接口除能，SDA 和 SCL 脚将失去 I²C 功能，I²C 工作电流减小到最小值。此位为“1”时，I²C 接口使能。当 IICEN 位由低到高转变时，I²C 控制寄存器中的设置，如 HTX 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I²C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0　未定义，读为“0”

- **IICC1 寄存器**

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

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

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

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

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

Bit 3　**TXAK**：I²C 总线发送应答标志位
0：从机发送应答标志
1：从机没有发送应答标志
此位是 I²C 总线发送应答标志位。从机接收完 8 位数据之后，该位将在第九个从机时钟时被传到总线上。如果从机想要接收更多的数据，则应在接收数据之

前将此位设置为“0”。

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

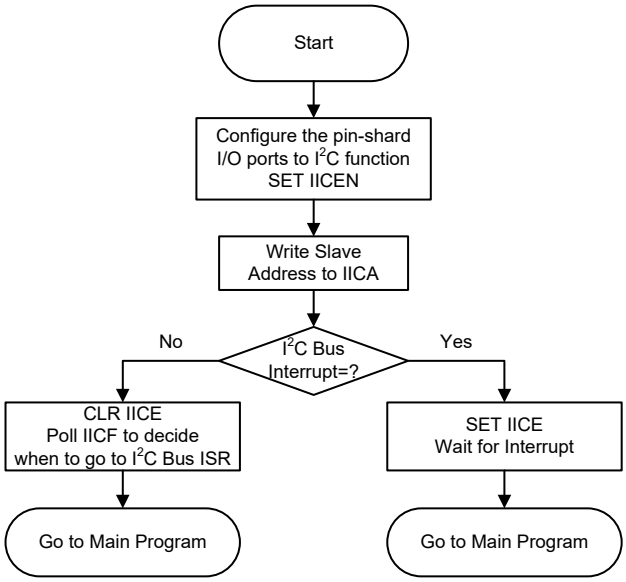
Bit 1 **IAMWU**：I²C 地址匹配唤醒控制位
0：除能
1：使能
此位设置为“1”则使能 I²C 地址匹配使系统从休眠或空闲模式中唤醒的功能。若进入休眠或空闲模式前 IAMWU 已经置高以使能 I²C 地址匹配唤醒功能，在系统唤醒后须软件清除此位以确保单片机正确地运行。

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

I²C 总线通信

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

- 步骤 1
  设置相应的引脚共用功能为 I²C 功能引脚和 IICC0 寄存器的 IICEN 位为“1”，以使能 I²C 总线。
- 步骤 2
  向 I²C 总线地址寄存器 IICA 写入从机地址。
- 步骤 3
  设置中断控制寄存器的 IICE 中断使能位以使能 I²C 中断。



I²C 总线初始化流程图

I²C 总线起始信号

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

I²C 从机地址

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

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

I²C 总线读 / 写信号

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

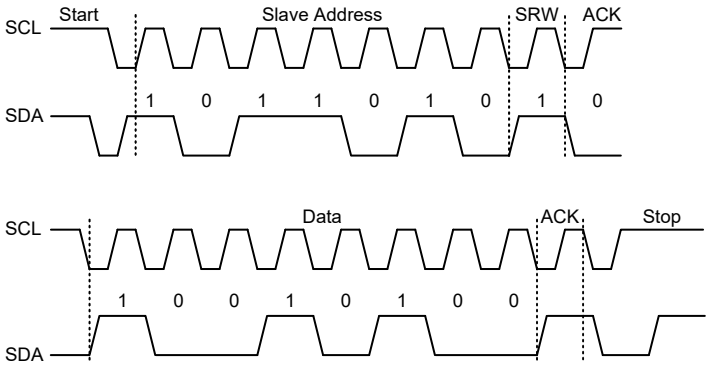
I²C 总线从机地址应答信号

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

I²C 总线数据和应答信号

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

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

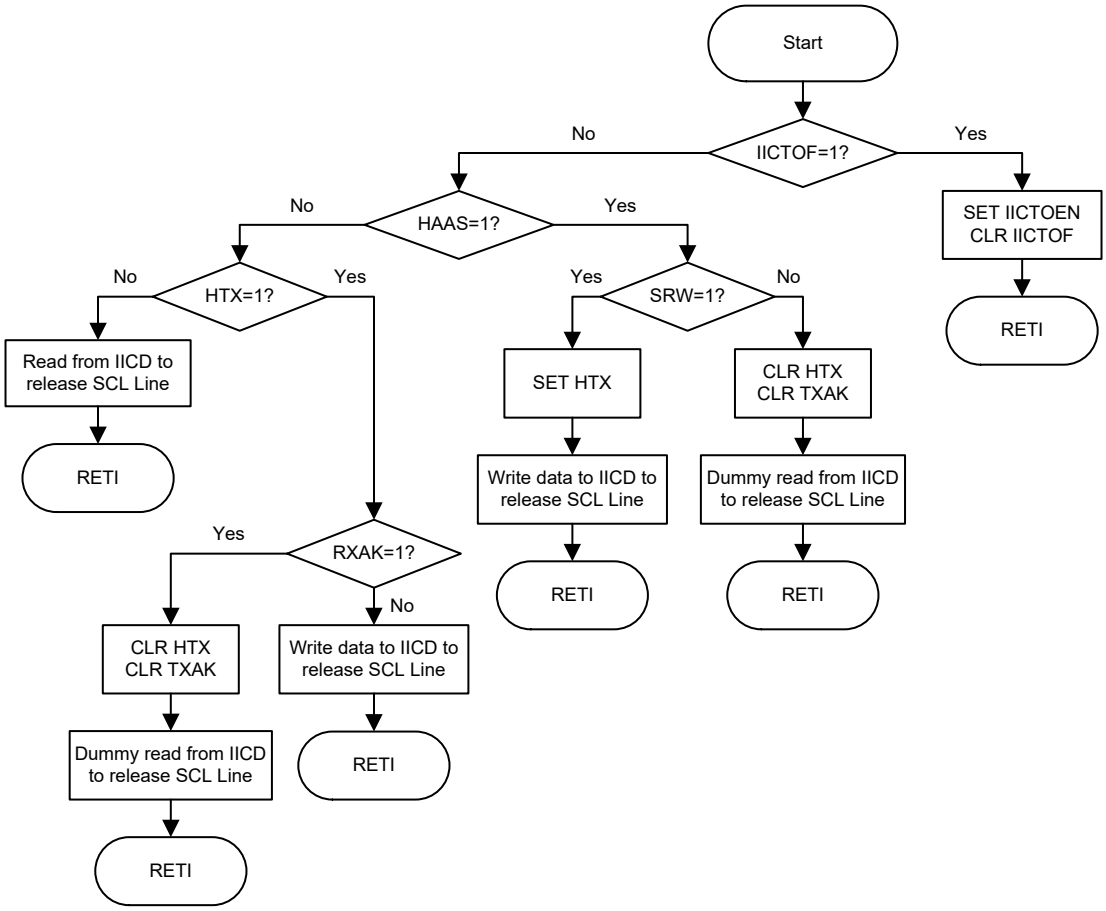


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

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

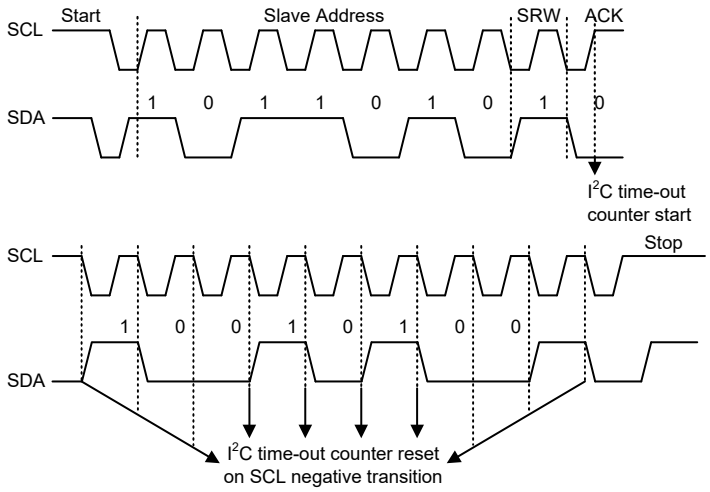
I²C 通信时序图



I²C 总线 ISR 流程图

I²C 超时控制

超时功能可减少 I²C 接收错误的时钟源而引起的锁死问题。如果连接到 I²C 总线的时钟源经过一段时间还未接收到，则在一定的超时周期后，I²C 电路和寄存器将复位。超时计数器在 I²C 总线“START”和“地址匹配”条件下开始计数，且在 SCL 下降沿清零。在下一个 SCL 下降沿到来之前，如果超时时间大于 IICTOC 寄存器指定的超时周期，则超时发生。I²C“STOP”条件发生时超时功能终止。



I²C 超时时序图

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

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

超时发生后的 I²C 寄存器

IICTOF 标志位由应用程序清零。共有 64 个超时周期，可通过 IICTOC 寄存器的 IICTOS 位进行选择。超时周期可通过公式计算：[(1~64)×32]/f_{SUB}。由此可得超时周期范围为 1ms~64ms。

- **IICTOC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

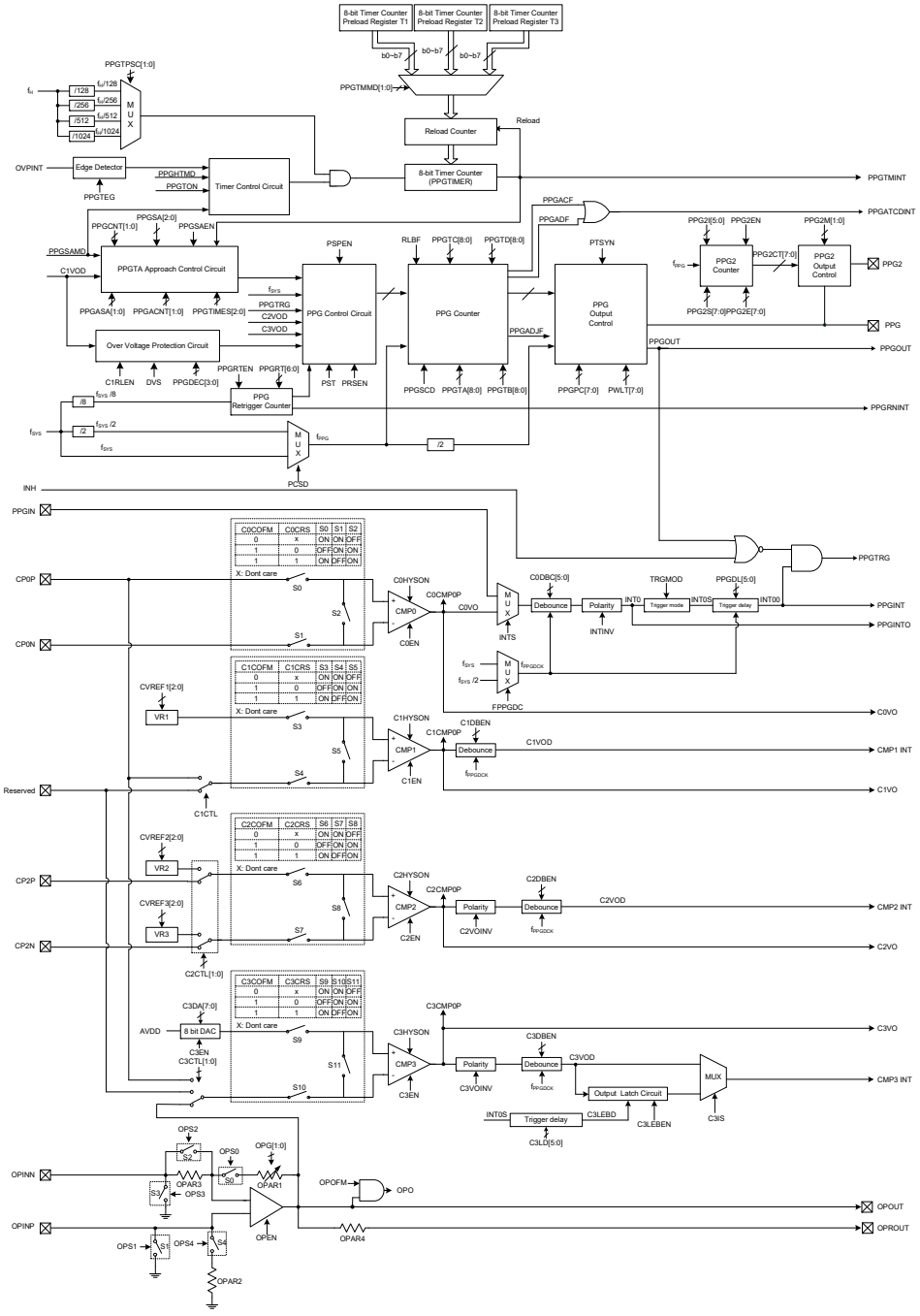
Bit 7 **IICTOEN**：I²C 超时控制位
0：除能
1：使能

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

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

电磁炉电路

该单片机具有一个多功能完全集成的电磁炉电路，具有 PPG 输出最大的应用灵活性。其还提供了多功能保护机制。电磁炉电路由一个 PPG 模块、四个比较器和一个运算放大器组成。



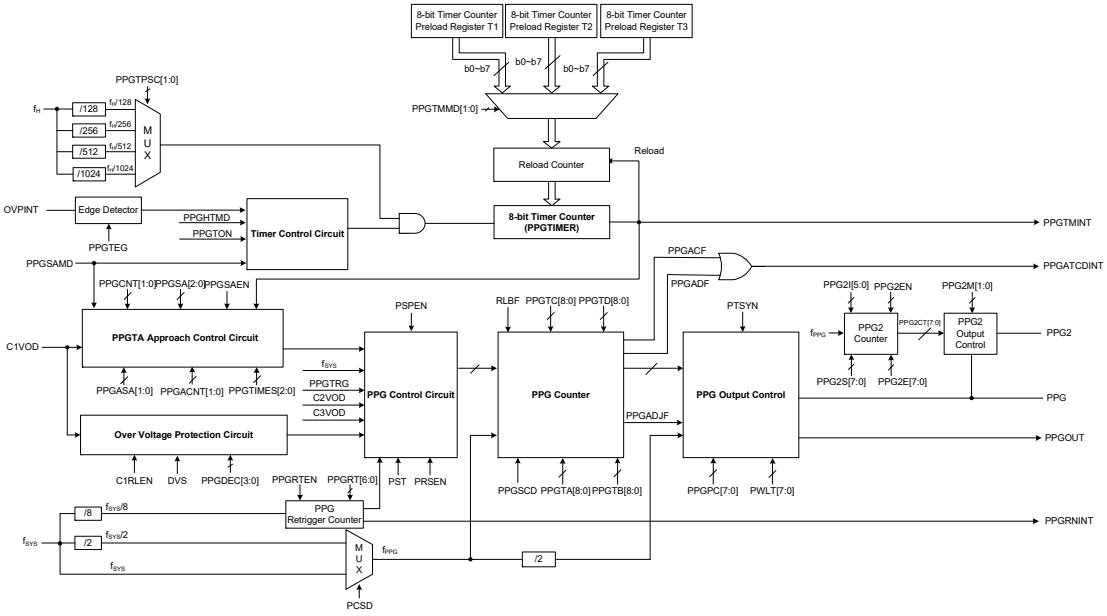
注：1. CMPn 中断由 CMPnINT 下降沿触发。(n=1~3)

2. PPG 和 PPG2 信号由内部分别连接至电平转换器的 PWM1 和 PWM2。若要使用此信号，需合理设置其相关引脚共用控制位。

电磁炉电路方框图

可编程脉冲发生器 – PPG

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



注：PPG 和 PPG2 信号由内部分别连接至电平转换器的 PWM1 和 PWM2。若要使用此信号，需合理设置其相关引脚共用控制位。

PPG 方框图

当 PPG 检测到有一个触发信号输入，就会输出一个脉冲。触发源可以通过软件配置来自 INT00 触发输入或软件触发位。通过设置极性控制寄存器 PPGPC，PPG 引脚可以输出一个低电平有效脉冲，一个高电平有效脉冲，强制为高电平或强制为低电平，当选择低电平有效脉冲或高电平有效脉冲时，需要外接一个上拉或下拉电阻。

PPG 模块由一个 PPG 控制电路，一个 PPGTIMER 计数器、一个 PPG 计数器和一个 PPG 输出控制器组成。而 PPG 计数器由一个 9 位向上计数器，两组 9 位预载数据寄存器和两组 9-bit 计数器逼近寄存器组成。可编程脉冲发生器 PPG 从预载寄存器的当前值开始计数，直到内部数据由“1FFH→000H”时停止计数。写入“000H”到 PPGTA[8:0] 和 PPGTB[8:0] 位会产生一个 512×T 的脉宽输出。一旦计数器溢出，预载寄存器里的值会自动装载到计数器中，同时产生一个信号去停止 PPG 计数器。PPG 计数器溢出的同时软件触发位 PST 也会被清零。

以下任一情况发生，PPG 计数器重新载入：

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

通常情况下，如果 RLBF=0，PPG 计数器会重新载入预载寄存器 A 的值。如果 C1RLEN=1，来自 C1VOD 信号下降沿产生会置高 RLBF 位，使得 PPG 计数器重新载入预载寄存器 B 的值，直到 RLBF 被软件清零。

PRSEN 位用于使能或除能 PPG 的启动信号是来自 INT00 的触发输入还是重复触发定时器，如果 PRSEN 位使能，PPG 模块输出可以通过 INT00 触发、重复

触发定时器有效或软件将 PST 设置为“1”重新启动。一旦 INT00 信号的下降沿产生，PPG 计数器就会开始计数。

无论 PPG 是否处于有效周期，C2VOD 或 C3VOD 信号下降沿都将清除 PRSEN 位，PPG 计数器就会停止计数。这样可以防止 PPG 再由 INT00 的下降沿触发启动，PPG 的启动只能通过软件控制，直到 PRSEN 位再次由软件置位。

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

在 PPG 计数过程中，如果有一个 INT00 的下降沿产生、重复触发定时器有效或 PST 被置位，PPG 计数器将不受影响，也就是说此时来自 INT00、重复触发定时器有效或 PST 的再启动信号无效。PST 也可用作 PPG 计数器输出的状态标志位。

PPG 输出由 PPGPC 寄存器决定。如果 PPGPC[7:0] 位被设为 01010101，PPG 输出低电平有效脉冲；如果 PPGPC[7:0] 位被设为 10101010，PPG 输出高电平有效脉冲；如果 PPGPC[7:0] 位被设为 00110011，PPG 输出强制为高电平；如果 PPGPC[7:0] 位被设为 00110010，PPG 输出强制为低电平，如果 PPGPC[7:0] 位被设为 10001101，PPG 输出高电平有效脉冲，低电平无效脉冲；如果 PPGPC[7:0] 位被设为除以上五个设定值外的其它任意值，PPG 输出为浮空，PPG 输出若选择 10001101 以外的输出状态，PPG2 输出将被固定为低。

另外一个功能，即 PPG 计数器是否与时钟同步，由 PPGC0 寄存器中的 PTSYN 位决定。

在使用 PPG 功能时，需先将 CMP1 设置完成且使 C1VOD=1，再设定 PPGC2 寄存器。否则，当 C1VOD 状态未知且 PPGDEC[3:0]≠0000，PPGTA 就会每隔 $8/f_{SYS}$ 或 $16/f_{SYS}$ 时间递增一个特定值直至 1FFH，每次的递增值由 PPGDEC[3:0] 位确定。

PPG 寄存器

PPG 的功能和操作由一系列寄存器控制。在修改相关位时必须考虑下表的注意事项。

C1VOD 信号	PPGSAMD	PPGSAEN	PPGTMMD [1:0]	PPGDEB [3:0]	软件无法更新位
0	x	x	xx	0000	—
				0001~1111	PPGTA[8:0]
1	0	0	xx	xxxx	—
	0	1	xx		PPGTA[8:0], PPGCNT[1:0], PPGSA[2:0]
	1	x	01/10		PPGTA[8:0], PPGCNT[1:0], PPGSA[2:0]
	1	x	00/11		—

“x”：无关

寄存器名称	位							
	7	6	5	4	3	2	1	0
PPGC0	PST	PRSEN	PSPEN	RLBF	PTSYN	PCSD	TRGMOD	C1RLEN
PPGC1	INTS	FPPGDC	PPGDL5	PPGDL4	PPGDL3	PPGDL2	PPGDL1	PPGDL0
PPGC2	—	—	—	DVS	PPGDEC3	PPGDEC2	PPGDEC1	PPGDEC0
PPGTA	PPGTA7	PPGTA6	PPGTA5	PPGTA4	PPGTA3	PPGTA2	PPGTA1	PPGTA0
PPGTB	PPGTB7	PPGTB6	PPGTB5	PPGTB4	PPGTB3	PPGTB2	PPGTB1	PPGTB0
PPGTC	PPGTC7	PPGTC6	PPGTC5	PPGTC4	PPGTC3	PPGTC2	PPGTC1	PPGTC0
PPGTD	PPGTD7	PPGTD6	PPGTD5	PPGTD4	PPGTD3	PPGTD2	PPGTD1	PPGTD0
PPGTEX	—	PPGTD8	—	PPGTB8	—	PPGTC8	—	PPGTA8
PWLT	D7	D6	D5	D4	D3	D2	D1	D0
PPGPC	PPGPC7	PPGPC6	PPGPC5	PPGPC4	PPGPC3	PPGPC2	PPGPC1	PPGPC0
PPGATC0	PPGSAEN	PPGSAMD	PPGSCD	PPGADJF	PPGTMMD1	PPGTMMD0	PPGACF	PPGADF
PPGATC1	PPGHTMD	—	—	PPGCNT1	PPGCNT0	PPGSA2	PPGSA1	PPGSA0
PPGATC2	—	PPGTIMES2	PPGTIMES1	PPGTIMES0	PPGACNT1	PPGACNT0	PPGASA1	PPGASA0
PPGTMC	—	—	—	PPGTON	PPGTEG	—	PPGTPSC1	PPGTPSC0
PPGTMR1	D7	D6	D5	D4	D3	D2	D1	D0
PPGTMR2	D7	D6	D5	D4	D3	D2	D1	D0
PPGTMR3	D7	D6	D5	D4	D3	D2	D1	D0
PPGTMRD	D7	D6	D5	D4	D3	D2	D1	D0
PPGRT	PPGRTEN	PPGRT6	PPGRT5	PPGRT4	PPGRT3	PPGRT2	PPGRT1	PPGRT0
PPGRN	—	PPGRN6	PPGRN5	PPGRN4	PPGRN3	PPGRN2	PPGRN1	PPGRN0
PPG2CT	PPG2CT7	PPG2CT6	PPG2CT5	PPG2CT4	PPG2CT3	PPG2CT2	PPG2CT1	PPG2CT0
PPG2C0	PPG2EN	—	—	—	—	—	PPG2M1	PPG2M0
PPG2C1	PPG2S7	PPG2S6	PPG2S5	PPG2S4	PPG2S3	PPG2S2	PPG2S1	PPG2S0
PPG2C2	PPG2E7	PPG2E6	PPG2E5	PPG2E4	PPG2E3	PPG2E2	PPG2E1	PPG2E0
PPG2C3	—	—	PPG2I5	PPG2I4	PPG2I3	PPG2I2	PPG2I1	PPG2I0

可编程脉冲发生器寄存器列表

- **PPGC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PST	PRSEN	PSPEN	RLBF	PTSYN	PCSD	TRGMOD	C1RLEN
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 **PST**：PPG 软件触发位
0：停止 PPG
1：重新启动 PPG

Bit 6 **PRSEN**：INT00 触发输入重新启动 PPG 模块输出的使能控制
0：除能
1：使能
除能 INT00 触发输入或重复触发定时器重新启动 PPG 模块输出时，PPG 模块输出只能通过设置软件控制位 PST 为 1 重新启动。使能 INT00 触发输入重新启动 PPG 模块输出时，PPG 输出可以通过 INT00 下降沿触发、重复触发定时器或软件控制位 PST 为 1 时重新启动。
注意，当开启台阶检测功能 (C3LEBEN=1)，且 PRSEN=1 时，发生 C3VOD 触发但不会清除 PRSEN 位。

Bit 5 **PSPEN**：C2VOD 或 C3VOD 触发输入停止 PPG 模块输出的使能控制
0：除能
1：使能

除能 C2VOD 或 C3VOD 触发输入停止 PPG 模块输出时，PPG 模块输出只能通过软件控制位 PST 停止。使能 C2VOD 或 C3VOD 触发输入停止 PPG 模块输出时，PPG 模块输出可以通过 C2VOD 或 C3VOD 下降沿停止，也可以通过软件控制将 PST 清除为“0”来停止。

注意，当开启台阶检测功能 (C3LEBEN=1)，且 PSPEN=1 时，发生 C3VOD 触发但不会使 PPG 定时器停止。

Bit 4 **RLBF**：PPG 重新载入控制位
0：PPG 计数器重载值来自预载寄存器 A，PPGTA[8:0]
1：PPG 计数器重载值来自预载寄存器 B，PPGTB[8:0]

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

Bit 2 **PCSD**：PPG 计数器和脉冲宽度限制器时钟源 f_{PPG} 选择
0：f_{SYS}
1：$f_{SYS}/2$

Bit 1 **TRGMOD**：选择 INT0 输出的 1 个下降沿或 2 个下降沿来产生触发延时电路输入 INT00
0：1 个下降沿
1：2 个下降沿

Bit 0 **C1RLEN**：C1VOD 下降沿发生时 RLBF 被置位的使能 / 除能位
0：除能
1：使能

若该位设置为 1，当 C1VOD 下降沿发生时，PPG 计数器将从预载寄存器 B 重新载入计数值，RLBF 被置 1。

- **PPGC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	INTS	FPPGDC	PPGDL5	PPGDL4	PPGDL3	PPGDL2	PPGDL1	PPGDL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	0	0	0	0	0

Bit 7 **INTS**：INT00 输入源选择
0：PPGIN
1：C0VO

Bit 6 **FPPGDC**：f_{PPGDCK} 时钟选择
0：f_{SYS}
1：$f_{SYS}/2$

Bit 5~0 **PPGDL5~PPGDL0**：PPG 触发延迟时间选择 (f_{PPGDCK}=8MHz)
000000：无延迟
000001：$1/f_{PPGDCK}$，0.125μs
000010：$2/f_{PPGDCK}$，0.25μs
：
101111：$47/f_{PPGDCK}$，5.875μs
110000~111111：$48/f_{PPGDCK}$，6μs

1. 触发延迟是指从 INT0S 下降沿到 PPG 硬件触发信号 INT00 发送时的时间，INT0S 是 INT0 信号的单个或双下降沿。INT0 信号可以来自 PPGIN 引脚或 C0VO 信号，其通过 INTS 位选择。INT0 输入信号的去抖时间和极性分别由 C0DBC[5:0] 和 INTINV 位控制。
2. 在触发延迟时间内发生的 INT0S 下降沿将被忽略。

- **PPGC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	DVS	PPGDEC3	PPGDEC2	PPGDEC1	PPGDEC0
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　**DVS**：PPG 反压递减周期选择
0：每隔 $8/f_{SYS}$
1：每隔 $16/f_{SYS}$
当 C1VOD=0，PPGDEC[3:0]≠0000，PPGTA 自动递增一次，之后每隔 $8/f_{SYS}$ 或 $16/f_{SYS}$ 自动递增一次特定值。递增特定值由 PPGDEC[3:0] 位确定。当 C1VOD=1，PPGTA 不递增。

Bit 3~0　**PPGDEC3~PPGDEC0**：PPGTA 自动递增值
0000：0
0001：1
0010：2
0011：3
0100：4
0101：5
0110：6
0111：7
1000：8
1001：9
1010：10
1011：11
1100：12
1101：13
1110：14
1111：15

- **PPGTA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPGTA7	PPGTA6	PPGTA5	PPGTA4	PPGTA3	PPGTA2	PPGTA1	PPGTA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”：未知

Bit 7~0　**PPGTA7~PPGTA0**：PPG 计数器预载寄存器 A bit 7 ~ bit 0

- **PPGTB 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPGTB7	PPGTB6	PPGTB5	PPGTB4	PPGTB3	PPGTB2	PPGTB1	PPGTB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”：未知

Bit 7~0　**PPGTB7~PPGTB0**：PPG 计数器预载寄存器 B bit 7 ~ bit 0

- **PPGTC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPGTC7	PPGTC6	PPGTC5	PPGTC4	PPGTC3	PPGTC2	PPGTC1	PPGTC0
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 **PPGTC7~PPGTC0**：PPG 计数器逼近寄存器 C bit 7 ~ bit 0

- **PPGTD 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPGTD7	PPGTD6	PPGTD5	PPGTD4	PPGTD3	PPGTD2	PPGTD1	PPGTD0
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 **PPGTD7~PPGTD0**：PPG 计数器逼近寄存器 D bit 7 ~ bit 0

- **PPGTEX 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PPGTD8	—	PPGTB8	—	PPGTC8	—	PPGTA8
R/W	—	R/W	—	R/W	—	R/W	—	R/W
POR	—	x	—	x	—	x	—	x

“x”：未知

Bit 7 未定义，读为“0”
Bit 6 **PPGTD8**：PPG 计数器逼近寄存器 D bit 8
Bit 5 未定义，读为“0”
Bit 4 **PPGTB8**：PPG 计数器预载寄存器 B bit 8
Bit 3 未定义，读为“0”
Bit 2 **PPGTC8**：PPG 计数器逼近寄存器 C bit 8
Bit 1 未定义，读为“0”
Bit 0 **PPGTA8**：PPG 计数器预载寄存器 A bit 8

- **PWLT 寄存器**

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

“x”：未知

Bit 7~0 **D7~D0**：PPG 脉宽限制计数器寄存器 bit 7 ~ bit 0
脉宽限制为 (256-PWLT)/(f_{PPG}/2)

- **PPGPC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPGPC7	PPGPC6	PPGPC5	PPGPC4	PPGPC3	PPGPC2	PPGPC1	PPGPC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PPGPC7~PPGPC0**：PPG 输出控制位
00110010：PPG 输出强制为低电平
00110011：PPG 输出强制为高电平
01010101：PPG 输出低电平有效脉冲
10001101：PPG 输出高电平有效脉冲，低电平无效脉冲
10101010：PPG 输出高电平有效脉冲
其它值：PPG 输出浮空
注：PPG 输出若选择 10001101 以外的输出状态，PPG2 输出将被固定为低。

- **PPGATC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPGSAEN	PPGSAMD	PPGSCD	PPGADJF	PPGTMMD1	PPGTMMD0	PPGACF	PPGADF
R/W	R/W	R/W	R/W	R	R	R	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PPGSAEN**：PPGTA 逼近模式使能位
0：除能
1：使能
该位只在 PPGSAMD=0 (S/W 模式) 时有效，在 PPGSAMD=1 (H/W 模式) 时，由于 PPGTA 逼近操作由硬件控制，软件写入无效。

PPGSAMD	PPGSAEN		
	S/W 写入	H/W 写入	读取
0	0	x	0
	1		1
1	0	0	0
	0	1	1
	1	0	0
	1	1	1

“x”：无关

Bit 6 **PPGSAMD**：PPGTA 逼近模式选择
0：S/W 逼近模式
1：H/W 逼近模式
该位从 0 变为 1 时，PPGTON 位将被清零，PPGTIMER 计数器将重新加载 PPGTMR1 寄存器值。

Bit 5 **PPGSCD**：PPGTA 逼近位选择
0：PPGTC[8:0]
1：PPGTD[8:0]
此位只在 PPGSAMD=0B (S/W 逼近模式) 时有效。

Bit 4 **PPGADJF**：PPG 寄存器修改标志位
0：PPG 相关寄存器可以改变
1：PPG 相关寄存器不可改变
当此位为 1 时，表示 PPGTA 与 PPGATC1 寄存器中 PPGCNT[1:0] 及 PPGSA[2:0] 位不能通过软件改变其内容值。

Bit 3~2 **PPGTMMD1~PPGTMMD0**：PPG 计数器模式
00：PPGTA 浮动模式 (t0~t1 区间)
01：PPGTA 逼近 PPGTC 模式 (t1~t2 区间)
10：PPGTA 逼近 PPGTD 模式 (t2~t3 区间)
11：PPGTA 浮动模式 (t3~t0 区间)
这些位仅在 PPGSAMD=1 时有效。

Bit 1 **PPGACF**：PPGTA 逼近 PPGTC 操作完成标志位
0：PPGTA 逼近 PPGTC 操作未完成
1：PPGTA 逼近 PPGTC 操作完成
该位只能通过软件清零，但不能通过软件置位。如果该位为高，在 PPGSAMD=0 且 PPGSAEN 位由 0 变 1 时该位由硬件自动清零。如果 PPGSAMD=1 且 PPGHTMD =0 时，当有 OVPINT 触发时，该位由硬件自动清零。如果 PPGSAMD=1 且 PPGHTMD=1 时，PPGTON 位由 0 变 1 时，该位由硬件自动清零。

Bit 0 **PPGADF**：PPGTA 逼近 PPGTD 操作完成标志位
0：PPGTA 逼近 PPGTD 操作未完成
1：PPGTA 逼近 PPGTD 操作完成
该位只能通过软件清零，但不能通过软件置位。如果该位为高，在 PPGSAMD=0 且 PPGSAEN 位由 0 变 1 时该位由硬件自动清零。如果 PPGSAMD=1 且 PPGHTMD =0 时，当有 OVPINT 触发时，该位由硬件自动清零。如果 PPGSAMD=1 且 PPGHTMD=1 时，PPGTON 位由 0 变 1 时，该位由硬件自动清零。

- **PPGATC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPGHTMD	—	—	PPGCNT1	PPGCNT0	PPGSA2	PPGSA1	PPGSA0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7 **PPGHTMD**：在 H/W 逼近模式下 PPGTIMER 计数器触发源选择
0：OVPINT
1：PPGTON (0→1)

Bit 6~5 未定义，读为“0”

Bit 4~3 **PPGCNT1~PPGCNT0**：PPG 触发次数选择 (变量：M)
00：1
01：2
10：3
11：4

Bit 2~0 **PPGSA2~PPGSA0**：PPGTA 逼近数值选择 (变量：N)
000：±1
001：±2
010：±3
011：±4
100：±5
101：±6
110：±7
111：±8

- **PPGATC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PPGTIMES2	PPGTIMES1	PPGTIMES0	PPGACNT1	PPGACNT0	PPGASA1	PPGASA0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7　未定义，读为“0”

Bit 6~4　**PPGTIMES2~PPGTIMES0**：逼近次数选择 – 改变 PPG 触发次数 (M 值) 与逼近数值 (N 值)
000：1
001：2
010：3
011：4
100：5
101：6
110：7
111：8

Bit 3~2　**PPGACNT1~PPGACNT0**：PPG 触发次数改变选择 (改变 M 值)
00：不变
01：不变
10：+1
11：-1
1. 当 PPGCNT[1:0] 位递增或递减到最大值 11 或最小值 00 时，PPGCNT[1:0] 位会固定于最大值 11 或最小值 00。
2. 应注意在 H/W 逼近模式下当 PPGACNT[1:0]=10 时，在 t1~t2 区间是增加 1，但在 t2~t3 区间则是减 1；PPGACNT[1:0]=11 时，则是在 t1~t2 区间是减 1，但在 t2~t3 区间则是增加 1。

Bit 1~0　**PPGASA1~PPGASA0**：PPGTA 逼近数值改变选择 (改变 N 值)
00：不变
01：不变
10：+1
11：-1
1. 当 PPGSA[2:0] 位递增或递减到最大值 111 或最小值 000 时，PPGSA[2:0] 位会固定于最大值 111 或最小值 000。
2. 应注意在 H/W 逼近模式下当 PPGASA[1:0]=10 时，在 t1~t2 区间是增加 1，但在 t2~t3 区间则是减 1；PPGASA[1:0]=11 时，则是在 t1~t2 区间是减 1，但在 t2~t3 区间则是增加 1。

- **PPGTMC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	PPGTON	PPGTEG	—	PPGTPSC1	PPGTPSC0
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　**PPGTON**：PPGTIMER 计数器使能
0：除能
1：使能
当 PPGSAMD=1 且 PPGHTMD=0 时，该位的写入无效。

Bit 3　**PPGTEG**：OVPINT 触发 PPGTIMER 边沿类型选择
0：上升沿
1：下降沿

Bit 2　　未定义，读为“0”

Bit 1~0　　**PPGTPSC1~PPGTPSC0**：PPGTIMER 计数器预分频比选择
00：$f_H/128$
01：$f_H/256$
10：$f_H/512$
11：$f_H/1024$

- **PPGTMRn 寄存器 (n=1~3)**

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**：PPGTIMER 计数器预载寄存器 Tn Bit 7 ~ Bit 0

- **PPGTMRD 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**：PPGTIMER 计数器计数寄存器 Bit 7 ~ Bit 0

- **PPGRT 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPGRTEN	PPGRT6	PPGRT5	PPGRT4	PPGRT3	PPGRT2	PPGRT1	PPGRT0
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　　**PPGRTEN**：PPG 软件重复触发 on/off 位
当达到设定的触发次数后，此位由硬体自动清除为“0”，并产生中断信号。

Bit 6~0　　**PPGRT6~PPGRT0**：PPG 重复触发周期设定位
该周期可设定为 1~127，$T=(PPGRT + 1)\times 8/f_{SYS}$。
注：禁用 PPGRT[6:0]=0000000

- **PPGRN 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PPGRN6	PPGRN5	PPGRN4	PPGRN3	PPGRN2	PPGRN1	PPGRN0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7　　未定义，读为“0”

Bit 6~0　　**PPGRN6~PPGRN0**：PPG 软件重复触发次数设定位

- **PPG2CT 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPG2CT7	PPG2CT6	PPG2CT5	PPG2CT4	PPG2CT3	PPG2CT2	PPG2CT1	PPG2CT0
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0　　**PPG2CT7~PPG2CT0**：PPG2 脉冲宽度计数器 bit 7 ~ bit 0

当 PPG2M[1:0]=00 或 01 时，PPG2CT[7:0] 不计数，此时读值为“0”。

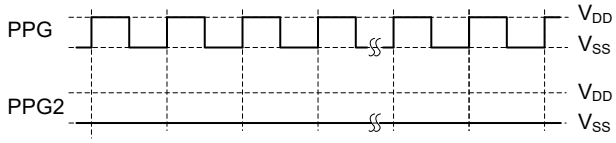
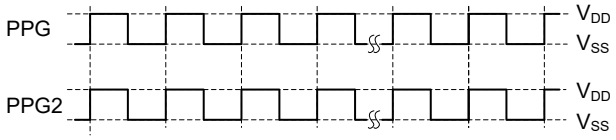
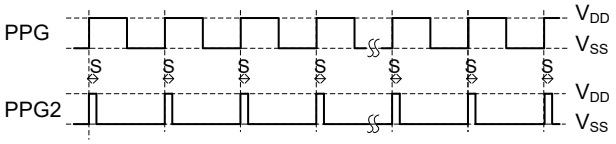
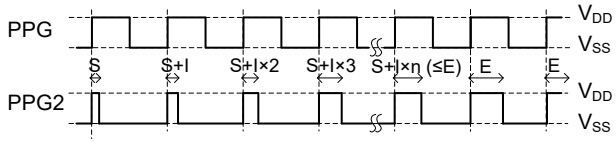
- **PPG2C0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPG2EN	—	—	—	—	—	PPG2M1	PPG2M0
R/W	R/W	—	—	—	—	—	R/W	R/W
POR	0	—	—	—	—	—	0	0

Bit 7 **PPG2EN**：PPG2 计数器使能位
0：除能
1：使能
当 PPG2EN 位除能，PPG2 输出为“0”。

Bit 6~2 未定义，读为“0”

Bit 1~0 **PPG2M1~PPG2M0**：PPG2 输出模式
00：PPG2 输出为 0
01：PPG2 输出脉波宽度与 PPG 相同
10：PPG2 输出脉波宽度固定 (脉波宽度固定值为 PPG2S)
11：PPG2 输出脉波宽度递增 (脉波宽度递增变化由 PPG2S，PPG2E 和 PPG2I 设定)
PPG2 由 PPG 同步触发产生，只有 PPG 发生时，才可能触发 PPG2，不会发生没有 PPG 但有 PPG2 的情况。
PPG2 输出波形如下：

PPG2 输出模式	PPG2 输出波形
PPG2M[1:0]=00	
PPG2M[1:0]=01	
PPG2M[1:0]=10	 注：“S”表示 PPG2S[7:0]。
PPG2M[1:0]=11	 注：“S”表示 PPG2S[7:0]；“E”表示 PPG2E[7:0]；“I”表示 PPG2I[5:0]；“n”表示递增值。

- **PPG2C1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPG2S7	PPG2S6	PPG2S5	PPG2S4	PPG2S3	PPG2S2	PPG2S1	PPG2S0
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 **PPG2S7~PPG2S0**：PPG2 起始值设定 (0~255)
PPG2 起始宽度时间 = PPG2S[7:0]×T~(PPG2S[7:0]+1)×T，(T=1/f_{PPG})。

- **PPG2C2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PPG2E7	PPG2E6	PPG2E5	PPG2E4	PPG2E3	PPG2E2	PPG2E1	PPG2E0
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

Bit 7~0 **PPG2E7~PPG2E0**：PPG2 结束值设定 (0~255)
PPG2 结束宽度时间 = (PPG2E[7:0]+1)×T，(T=1/f_{PPG})。

- **PPG2C3 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PPG2I5	PPG2I4	PPG2I3	PPG2I2	PPG2I1	PPG2I0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 未定义，读为“0”

Bit 5~0 **PPG2I5~PPG2I0**：PPG2 递增值设定 (0~63)

注：1. PPG2S[7:0] 须小于或等于 PPG2E[7:0]。若 PPG2S[7:0] 等于 PPG2E[7:0]，则表示 PPG2 宽度不递增。
2. 若 PPG2[7:0] 递增后大于 PPG2E[7:0]，则 PPG2 宽度保持在 PPG2E[7:0] 且不再递增。
3. 当 PPG2 计数器使能 (PPG2EN=1) 时，不允许写入 PPG2S[7:0]，PPG2E[7:0] 及 PPG2I[5:0]。

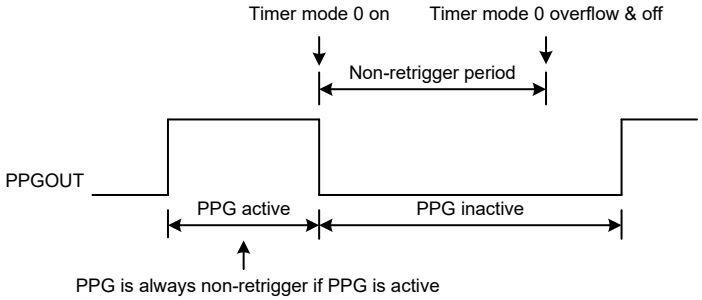
写数据到 PPGTA~PPGTD 寄存器说明

在写数据到 PPGTA/PPGTB/PPGTC/PPGTD 寄存器时，需先写入高字节再写入低字节，也就是要先写 PPGTEX 寄存器中的 PPGTA8/PPGTB8/PPGTC8/PPGTD8 位，高字节写入完成后再写入相应寄存器中的 PPGTA[7:0]/PPGTB[7:0]/PPGTC[7:0]/PPGTD[7:0] 位。如果更新了 PPGTEX 的值，但后续只执行了 PPGTA 寄存器的写入，那 PPGTEX 中的 PPGTB8、PPGTC8 和 PPGTD8 位是不会被更新的。此时读取 PPGTEX 的值，只有 PPGTA8 位会读到更新后的值，PPGTB8、PPGTC8 和 PPGTD8 保留之前所写入的值。

不可重复触发功能

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

- PPG 有效
- 处于不可重复触发期间，定时 / 事件计数器在 PPG 停止时开始计数。仅当定时 / 事件计数器 1 工作于模式 0 有效，不可重复触发周期取决于定时 / 事件计数器 1。PPG 输出从有效到无效转换发生时，计数器开始计算。



注：1. 定时 / 事件计数器 1 工作于模式 0，T1ON=1 时，INH 信号为高，将使能 PPG 不可重复触发模式以禁止 PPG 被再次触发，直到计数器溢出或 T1ON=0，INH 信号为低时，PPG 才能再次触发，信号正常输出。
2. 在 PPG 不可重复触发模式时，若有 INT00 触发信号发生或软件重复触发启动则不会触发 PPG，但软件 PST 控制位可以触发 PPG。

脉宽限制功能

PPG 模块具有脉宽限制功能，可以停止 PPG 输出。当脉宽达到限值时，PPG 将停止输出。该功能是通过脉宽限制计数器实现的，该计数器在 PPG 触发时开始计数，溢出或 PPG 停止时停止计数。脉宽限值为 (256-PWLT)/(f_{PPG}/2)，其中 PWLT 是脉宽限制计数器寄存器的值。应注意，脉宽限制计数器可能存在一个约 f_{PPG}/2 的误差。

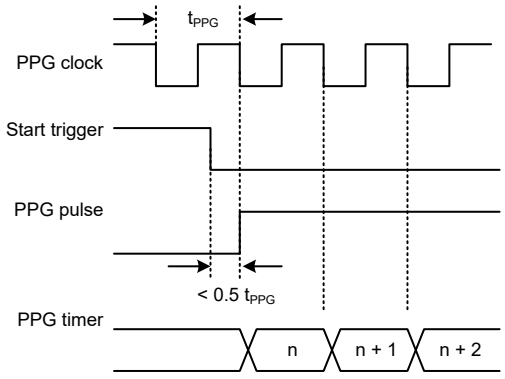
脉宽限制功能重新载入的时机：

- 脉宽限制计数器溢出
- PPG 被触发

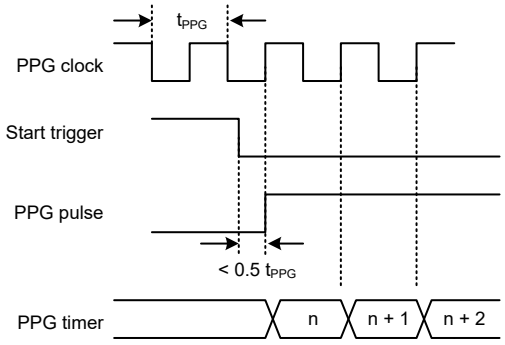
PPG 输出信号说明

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

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



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



停止 PPG 功能

任何停止 PPG 的动作，如 PPG 计数器溢出、C2VOD 或 C3VOD 下降沿触发 (PSPEN=1)、软件设置停止 (PST=1→0) 或达到脉冲限制等，均可导致以下情况的产生：

- PPG 计数器数据重新载入
- PST 位被清 0
- PPG 无效

启动 PPG 功能的操作

- 通过 PPGPC 寄存器设置 PPG 输出状态
- 确定 PPG 计数时钟是否与 PPG 时钟 f_{PPG} 同步
- 通过 PPGC 寄存器中的 PRSEN 和 PSPEN 位设置 PPG 触发启动和停止模式
- 设置 PPG 输出脉宽，写入数据到 PPGTA、PPGTB 和 PPGTEX 寄存器
- 通过 C1RLEN 位设置，确定是否使用 C1VOD 的下降沿来使能 PPGTB 的重新载入功能
- 确定是否使用不可重复触发周期功能 (使用定时 / 事件计数器 1 的模式 0)
- 通过 PWLT 寄存器设置脉宽限制计数器来限制脉宽
- 当 PPG 由 INT00 的下降沿产生、重复触发有效或软件触发位 PST 被设为 1 触发时，PPG 将从预载寄存器的当前值开始计数。当 PPG 计数器溢出或达到脉冲限制时，PPG 软件触发位 PST 被设为 0 或 C2VOD/C3VOD 下降沿发生，PPG 将停止计数。

反压保护功能

反压保护功能有两种：一个是将 PPG 的输出由 PPGTA 的值改为 PPGTB，另一个则是将 PPGTA 的值每隔一段时间就将其递增，以下分别来介绍这两个模式如何使用：

1. 将 PPG 的输出由 PPGTA 改为 PPGTB：将位于 PPGC0 寄存器中的 C1RLEN 位设定为“1”即可，当反压情况发生 (即 C1VOD 信号由高到低的变化产生)，这将产生一个下降沿触发信号，PPG 输出信号将会将从原本的 PPGTA 改为 PPGTB。
2. 减小 PPG 输出宽度：将位于 PPGC0 寄存器中的 C1RLEN 位设定为“0”，设定 PPGC2 寄存器中的 DVS 和 PPGDEC[3:0] 位，当反压情况发生 (C1VOD=0)

时，PPGTA 就会依设定的值每隔 8 个 t_{SYS} 或 16 个 t_{SYS} 时间向上增加，每次增加的设定值由 PPGDEC[3:0] 位确定，PPG 宽度会慢慢的缩减。

反压保护的真值表如下：

PPG 工作模式	C1VOD 信号	C1RLEN	PPGDEC [3:0]	RLBF	说明
0	↓/0	0	0000	0	PPGTA[8:0] 不会自动更新。PPG 宽度由 PPGTA[8:0] 决定。 (此模式不建议使用)
1	↓/0	0	0001~1111	0	PPGTA[8:0] 每隔 $8/f_{SYS}$ 或 $16/f_{SYS}$ 时间增加一个特定值，特定值由 PPGDEC[3:0] 位决定，PPG 宽度由 PPGTA[8:0] 决定。
2	↓/0	1	0000	1 (by H/W)	PPGTA[8:0] 不会自动更新。PPG 宽度由 PPGTB[8:0] 决定。
3	↓/0	1	0001~1111	1 (by H/W)	PPGTA[8:0] 每隔 $8/f_{SYS}$ 或 $16/f_{SYS}$ 时间增加一个特定值，特定值由 PPGDEC[3:0] 位决定。PPG 宽度由 PPGTB[8:0] 决定。

PPGTA 逼近功能

PPGTA 逼近功能只能工作于 C1VOD=1 时，也就是无反压情况发生，当工作于逼近模式时，若 C1VOD=0 时，PPG 会立即切换到 PPG 反压保护模式，而 PPG 逼近模式有 2 种控制方式，分别为软件控制与硬件控制，以下会分别说明其差异和设定步骤。

S/W 逼近模式 (PPGSAMD=0)

用户可以自行设定何时要启动逼近功能，选择逼近 PPGTC 或是 PPGTD，当 PPGSAEN 位为“1”时，PPGADJF 会被硬件设置高。此时 PPGTA 与 PPGATC1 寄存器中的 PPGCNT[1:0] 及 PPGSA[2:0] 位不可以通过软件做任何的修改，PPGACF 与 PPGADF 也会被硬件清零，直到 PPGTA 等于 PPGTC 或 PPGTD，其相应的标志位就会被置位。

在 S/W 逼近模式中，PPGTIMER 工作于一般的计数器模式，计数值从 PPGTMR1 载入，当 PPGTON=1 时，计数器从 PPGTMR1 寄存器的当前值开始计数直到溢出并会触发 PPGTMINT 信号。

下面总结了在 S/W 逼近模式下实现 PPGTA 逼近过程的步骤。

步骤 1. 写入 PPGTA~PPGTD 寄存器的初始值，注意需先写高字节再写低字节，这样 PPGTA~PPGTD 的值才会正确写入。

步骤 2. 设定 PPG 触发次数以及逼近值通过配置 PPGATC1 寄存器。

步骤 3. 选择逼近几次后需调整 PPG 触发次数以及逼近值，通过设置 PPGATC2 寄存器中 PPGTIMES[2:0]。

步骤 4. 选择 PPG 触发次数要如何变动，通过设置 PPGATC2 寄存器中 PPGACNT[1:0] 位。

步骤 5. 选择逼近值要如何变动，通过设置 PPGATC2 寄存器中 PPGASA[1:0] 位。

步骤 6. 将 PPGATC0 寄存器中 PPGSAMD 位清零。

步骤 7. 选择逼近的寄存器，通过设置 PPGATC0 寄存器中 PPGSCD 位。

步骤 8. 设置其它 PPG 相关寄存器，更多详细信息请参考“PPG 寄存器”章节。

步骤 9. 将 PPGATC0 寄存器中 PPGSAEN 位置高。

步骤 10. 可以读取 PPGACF 与 PPGADF 来判别 PPGTA 是否已与 PPGTC 或 PPGTD 相等。

H/W 逼近模式 (PPGSAMD=1)

此模式通过 PPGTIMER 的计时区间来决定 PPGTA 如何改变，而 PPGTIMER 计数器的触发源有两种，其通过 PPGHTMD 位选择，可选择由 OVPINT 有效触发沿或 PPGTON 位由 0 变为 1 时，启动 PPGTIMER。PPG 在四个时间区间内会做不同的动作，第一区间 (t0~t1) 也就是 PPGTIMER 由 PPGTMR1 的值开始计数到计数器溢出，PPG 会输出相同宽度的脉冲，即 PPGTA[8:0] 不会自动调整，为固定值；第二区间 (t1~t2) 也就是 PPGTIMER 由 PPGTMR2 的值开始计数到计数器溢出，PPGTA 会依 PPGCNT[1:0] 与 PPGSA[2:0] 设定的值向 PPGTC 逼近，且到达 PPGTIMES[2:0] 所设定的逼近次数后，PPGCNT[1:0] 与 PPGSA[2:0] 会依 PPGACNT[1:0] 与 PPGASA[1:0] 的设定更动，直到 PPGTA 等于 PPGTC 或者计数器溢出后，PPGTA 的值维持不变；第三区间 (t2~t3) 也就是 PPGTIMER 由 PPGTMR3 的值开始计数到计数器溢出，PPGTA 会依 PPGCNT[1:0] 与 PPGSA[2:0] 设定的值向 PPGTD 逼近，且到达 PPGTIMES[2:0] 所设定的逼近次数后，PPGCNT[1:0] 与 PPGSA[2:0] 会依 PPGACNT[1:0] 与 PPGASA[1:0] 的设定更动，直到 PPGTA 等于 PPGTD 或者计数器溢出后，PPGTA 的值维持不变；第四区间 (t3~t0) 也就是 PPGTIMER 除能，通过软件修改 PPG 相关参数。

应注意，在 t1 点前，须将 PPG 相关寄存器设置好，否则，在 t1~t3 区间，PPGTA[8:0]、PPGCNT[1:0] 与 PPGSA[2:0] 位不能被改变，直到 t3 点后才能修改这些位的值。

在第二区间 (t1~t2)，PPGTA 等于 PPGTC 时，PPGACF 位会被硬件置位，该位可以通过软件清零，但不可通过软件置位。当 PPGACF=1 时，软件未将其清零，则在下一个 OVPINT 下降沿或 PPGTON 位由 0 变 1 时，硬件会将其清零。

在第三区间 (t2~t3)，PPGTA 等于 PPGTD 时，PPGADF 位会被硬件置位，该位可以通过软件清零，但不可通过软件置位。当 PPGADF=1 时，软件未将其清零，则在下一个 OVPINT 下降沿或 PPGTON 位由 0 变 1 时，硬件会将其清零。

在 H/W 逼近模式中，若要停止相关功能应将 PPGSAMD 位清零以选择 S/W 逼近模式即可。

C1VOD 信号	PPGSAMD	PPGSAEN	PPGSCD	PPGTMMD [1:0]	说明
1	0	0	x	x	PPGTA[8:0] 不会自动更新。
1	0	1	0	x	PPGTA[8:0] 依 PPGCNT[1:0] 设定的触发次数逼近 PPGTC[8:0]，逼近值由 PPGSA[2:0] 决定。
1	0	1	1	x	PPGTA[8:0] 依 PPGCNT[1:0] 设定的触发次数逼近 PPGTD[8:0]，逼近值由 PPGSA[2:0] 决定。
1	1	0	x	00B	PPGTA[8:0] 不会自动更新。
1	1	1 (by H/W)	x	01B	PPGTA[8:0] 依 PPGCNT[1:0] 设定的触发次数逼近 PPGTC[8:0]，逼近值由 PPGSA[2:0] 决定。
1	1	1 (by H/W)	x	10B	PPGTA[8:0] 依 PPGCNT[1:0] 设定的触发次数逼近 PPGTD[8:0]，逼近值由 PPGSA[2:0] 决定。
1	1	0	x	11B	PPGTA[8:0] 不会自动更新。

“x”：无关

下面总结了在 H/W 逼近模式下实现 PPGTA 逼近过程的步骤。

步骤 1. 写入 PPGTA~PPGTD 寄存器的初始值，注意需先写高字节再写低字节，这样 PPGTA~PPGTD 的值才会正确写入。

步骤 2. 选择 PPG 触发次数和逼近值，通过设置 PPGATC1 寄存器中的 PPGACNT[1:0] 和 PPGASA[2:0] 位。

步骤 3. 选择硬件触发来源，通过设置 PPGATC1 寄存器中的 PPGHTMD 位。

步骤 4. 选择逼近几次后要调整 PPG 触发次数和逼近值，通过设置 PPGATC2 寄存器中的 PPGTIMES[2:0] 位。

步骤 5. 选择 PPG 触发次数要如何变动，通过设置 PPGATC2 寄存器中的 PPGACNT[1:0] 位。

步骤 6. 选择逼近值要如何变动，通过设置 PPGATC2 寄存器中的 PPGASA[1:0] 位。

步骤 7. 设置 PPGTM1、PPGTM2 和 PPGTM3 的计数器值。

步骤 8. 选择硬件逼近功能由 OVPINT 的上升源或下降源触发，通过设置 PPGTMC0 寄存器中的 PPGTEG 位。

步骤 9. 设置计数器的时钟源，通过 PPGTMC0 寄存器中 PPGTPSC 位。

步骤 10. 设置其它 PPG 相关寄存器，更多详细信息请参考“PPG 寄存器”章节。

步骤 11. 将 PPGATC0 寄存器中的 PPGSAMD 位设定为 1。应注意，当 PPGSAMD=1 时，PPGTON 位会被硬件清零，若 PPGHTMD 位为 0，一旦 OVPINT 有效触发信号发生就会触发硬件动作，在设置此位之前，应确保其它相关设置已完成，以避免不可预测的错误。

步骤 12. 读取 PPGACF 与 PPGADF 来判别 PPGTA 是否已与 PPGTC 或 PPGTD 相等。

步骤 13. 读取 PPGTMMD[1:0] 位以确定当前的 PPGTIMER 工作模式。

范例：

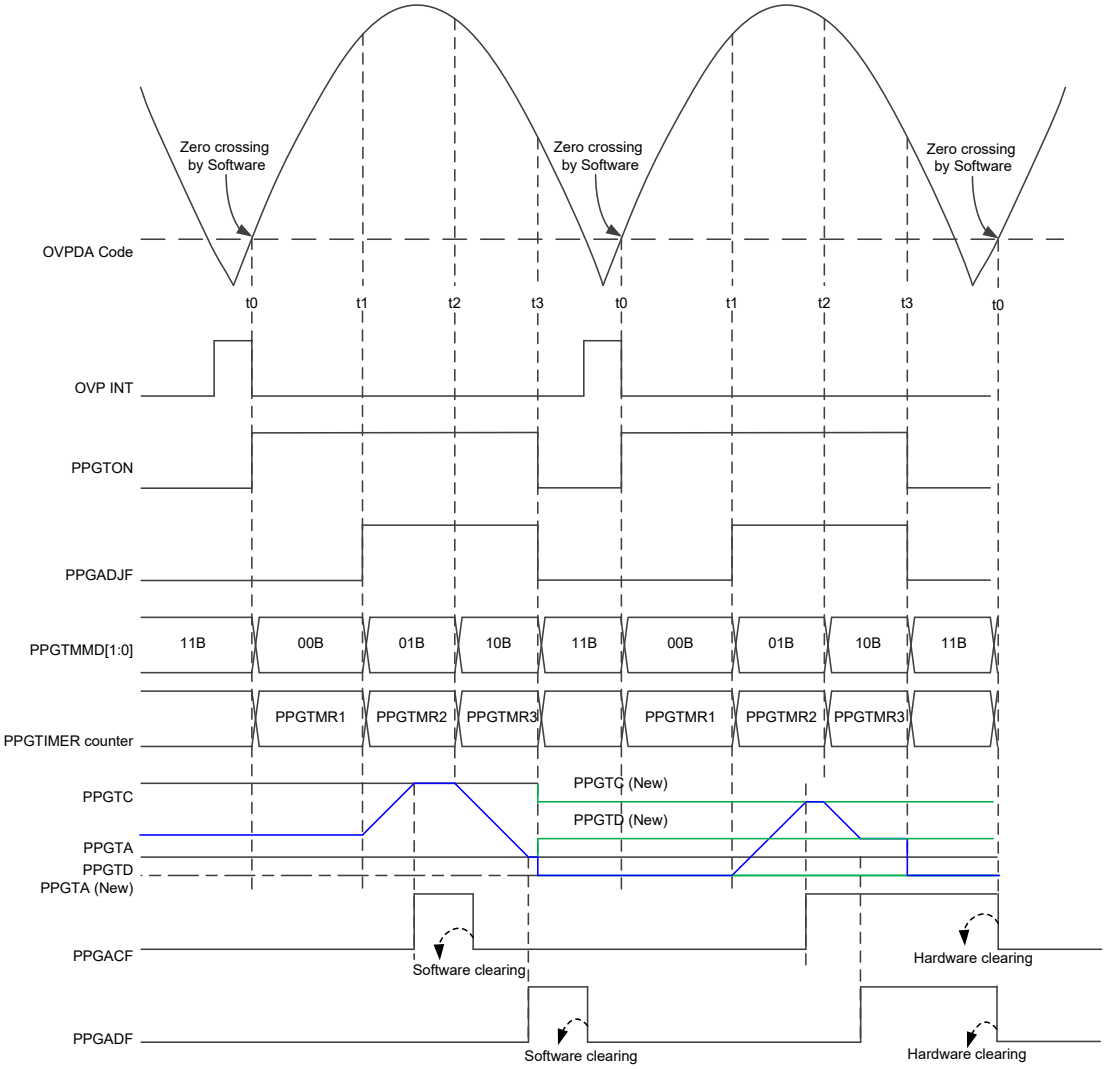
1. PPGSAMD=1；PPGTEG=1；PPGTPSC0=1；PPGHTMD=0
2. PPGCNT[1:0]=10B；PPGSA[2:0]=011B；其表示“每 3 次 PPG 触发，PPGTA 增加 4”
3. PPGTIMES[2:0]=001B；PPGACNT[1:0]=10B；PPGASA[1:0]=00B；其表示“每 2 次逼近后，PPG 触发次数 +1，增加值不变，故变化为每 4 次 PPG 触发，PPGTA 增加 4”
4. PPGTA=400；PPGTC=420；PPGTD=410

信号 / 位	INT00	OVPINT	PPGSAEN	PPGADJF	PPGTMMD [1:0]	PPGTON	PPGTA [8:0]	PPGTC [8:0]	PPGTD [8:0]	PPGACF	PPGADF
t0		0/1/↑	0	0	11	0	400	420	410	0/1	0/1
t0~t1		↓	0	0	00	1	400	420	410	0	0
t1~t2	↓	0	1	1	01	1	400	420	410	0	0
	↓	0	1	1	01	1	400	420	410	0	0
	↓	0	1	1	01	1	404	420	410	0	0
	↓	0	1	1	01	1	404	420	410	0	0
	↓	0	1	1	01	1	404	420	410	0	0
	↓	0	1	1	01	1	408	420	410	0	0
	↓	0	1	1	01	1	408	420	410	0	0
	↓	0	1	1	01	1	408	420	410	0	0
	↓	0	1	1	01	1	408	420	410	0	0
	↓	0	1	1	01	1	412	420	410	0	0
	↓	0	1	1	01	1	412	420	410	0	0
	↓	0	1	1	01	1	412	420	410	0	0
	↓	0	1	1	01	1	412	420	410	0	0
	↓	0	1	1	01	1	416	420	410	0	0
	↓	0	1	1	01	1	416	420	410	0	0
	↓	0	1	1	01	1	416	420	410	0	0
	↓	0	1	1	01	1	416	420	410	0	0
	↓	0	1	1	01	1	420	420	410	1	0
	↓	0	1	1	01	1	420	420	410	1	0
t2~t3	↓	0	1	1	10	1	420	420	410	1	0
	↓	0	1	1	10	1	420	420	410	1	0
	↓	0	1	1	10	1	420	420	410	1	0
	↓	0	1	1	10	1	416	420	410	1	0
	↓	0	1	1	10	1	416	420	410	1	0
	↓	0	1	1	10	1	416	420	410	1	0
	↓	0	1	1	10	1	416	420	410	1	0
	↓	0	1	1	10	1	412	420	410	1	0
	↓	0	1	1	10	1	412	420	410	1	0
	↓	0	1	1	10	1	412	420	410	1	0
	↓	0	1	1	10	1	410	420	410	1	1
	↓	0	1	1	10	1	410	420	410	1	1
t3~t0	↓	0	0	0	11	0	400	440	490	1	1
	↓	1	0	0	11	0	450	440	490	1	1
	↓	1	0	0	11	0	100	440	490	1	1

注：1. PPGTA 向上逼近 (PPGTC/PPGTD 大于 PPGTA)：当 PPGTC–PPGTA<PPGSA，在 PPG 触发后 PPGTA 会先加上 PPGSA 的值，此时 PPGTA>PPGTC，然后在下一次的 PPG 触发才会使 PPGTA=PPGTC。PPGTD 与 PPGTC 同理。

2. PPGTA 向下逼近 (PPGTC/PPGTD 小于 PPGTA)：当 PPGTA–PPGTC<PPGSA，在 PPG 触发后 PPGTA 会先减去 PPGSA 的值，此时 PPGTC > PPGTA，然后在下一次的 PPG 触发才会使 PPGTA=PPGTC。PPGTD 与 PPGTC 同理。

3. 若 (PPGTA+PPGSA)>511 或 (PPGTA-PPGSA)<0 时，PPGTA 会以极限值 511 或 0 填入。


Zero crossing by Software
Zero crossing by Software
Zero crossing by Software
OVPDA Code
t0
t1
t2
t3
t0
t1
t2
t3
t0
OVP INT
PPGTON
PPGADJF
PPGTMMD[1:0]
11B
00B
01B
10B
11B
00B
01B
10B
11B
PPGTIMER counter
PPGTMR1
PPGTMR2
PPGTMR3
PPGTMR1
PPGTMR2
PPGTMR3
PPGTC
PPGTC (New)
PPGTD (New)
PPGTA
PPGTD
PPGTA (New)
PPGACF
Software clearing
Hardware clearing
PPGADF
Software clearing
Hardware clearing

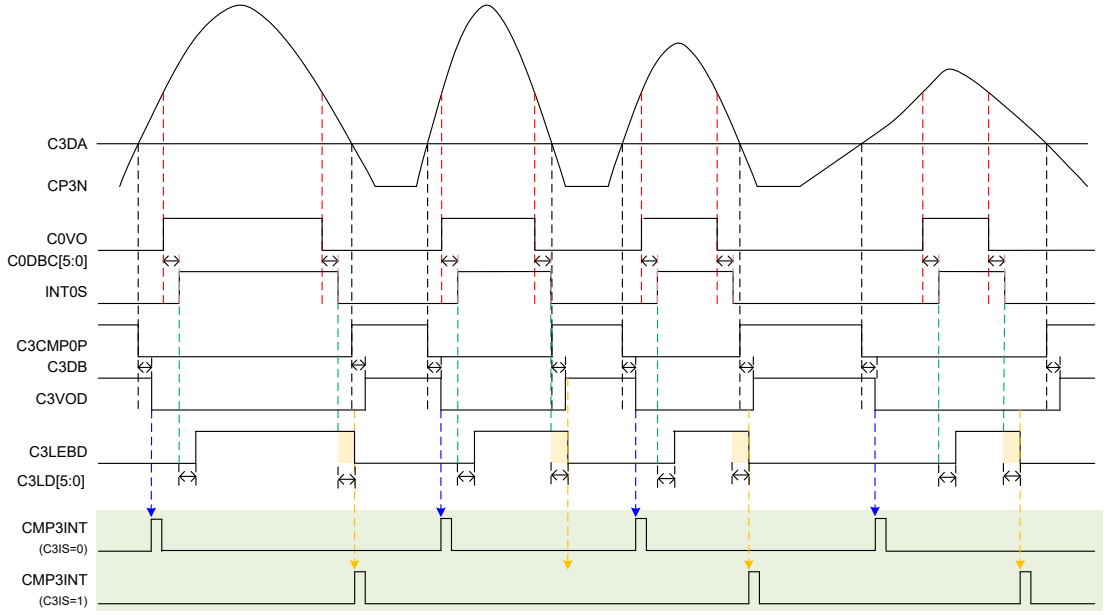
台阶检测功能：

当 C3LEBEN=1 且 INT0S 有一有效触发源信号时，INT0S 经过 C3LD[5:0] 设定的延迟时间后，产生如下图的 C3LEBD 信号，再以 C3LEBD 信号去侦测 C3VOD 的信号，依据侦测的信号去触发中断功能，说明如下：

若侦测 C3VOINV=0，侦测到的 C3VOD 信号为 0，此时会触发 CMP3INT；若侦测到的信号为 1，则不会触发 CMPINT。

若侦测 C3VOINV=1，侦测到的 C3VOD 信号为 1，此时会触发 CMP3INT；若侦测到的信号为 0，则不会触发 CMPINT。

下图以 C3VOINV=0 为例：



注：C3IS 位和 C3LEBEN 位的设定切换，建议在两次 PPG 触发之间 (PPG 上升沿 + 无重复触发区域内)，避免 CMP3 输出产生不预期状态。

IGBT 驱动器

该芯片包含一个 LDO，一个电平转换器和一个电压检测电路。以下将描述它们的工作方式。

LDO

该系统由输入引脚 VCC1 上大约 16V~20V 的高压系统供电。内建 LDO 将高压降至 5V 电平，并供电给输出引脚 VDD。较低的电压电平用于内建逻辑电路，但因为它可供电至 30mA，因此它也可用于外部电路系统。

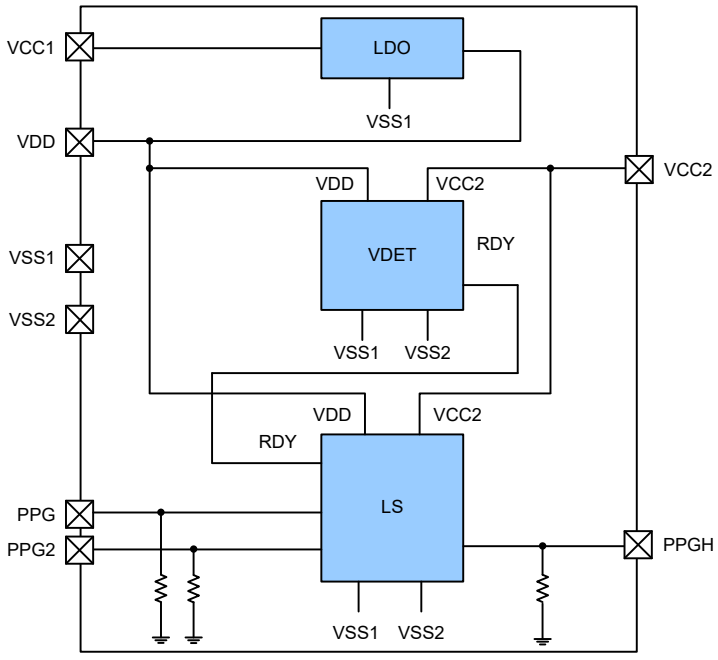
电平转换器

内建电平转换器输入引脚 PPG 和 PPG2 的输入逻辑电平参考值为 V_{DD}。电平转换器输出引脚 PPGH 的高压电平参考值为 V_{CC2}。内部电路必须确保参考电平 V_{DD} 到 V_{CC2} 的转换可靠地进行，以方便应用于 IGBT 驱动进程。电平转换引脚 PPG、PPG2 和引脚 PPGH 可通过下拉电阻连接到地。

输入			输出
“RDY”信号	PPG	PPG2	PPGH
0	x	x	低
1	0	x	低
1	1	0	快速爬升速率
1	1	1	低速爬升速率

电压检测器

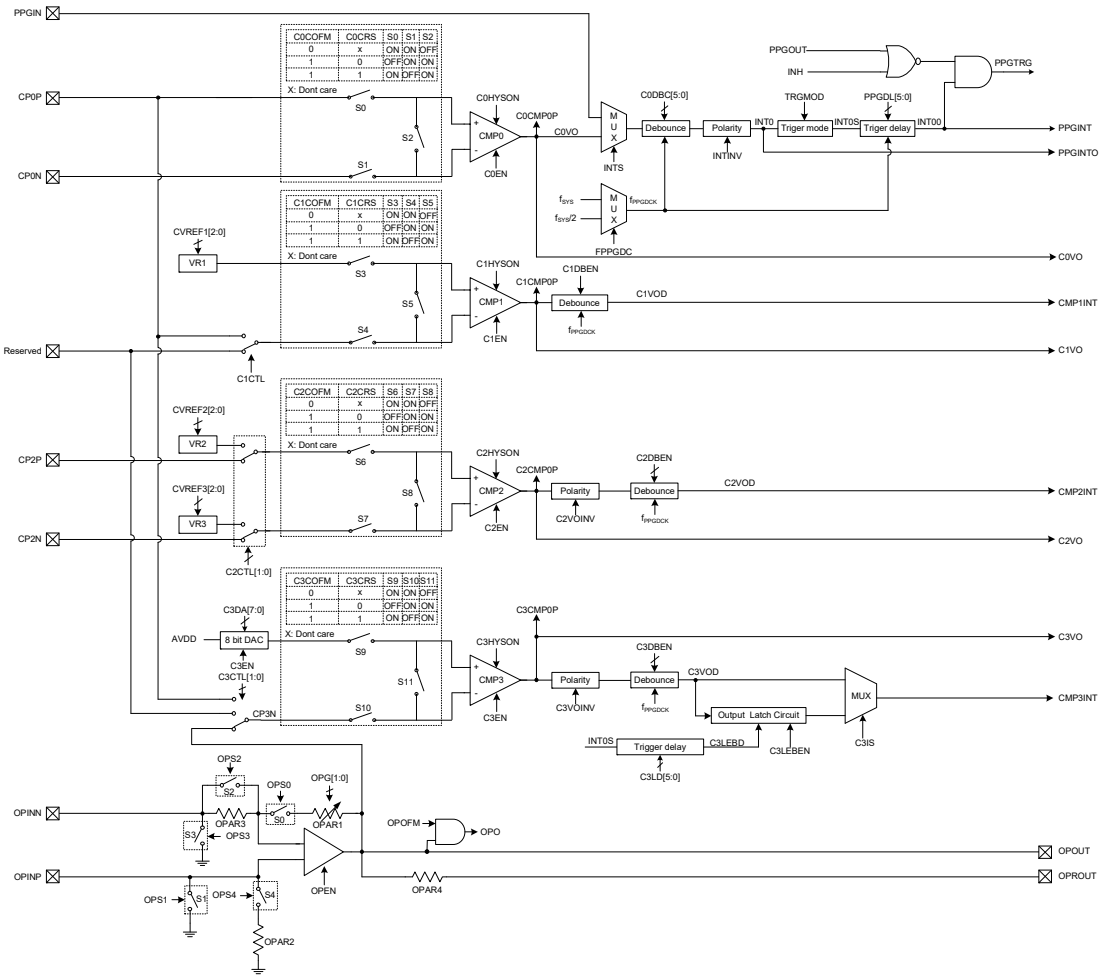
内建电压检测器可监测 V_{DD} 和 V_{CC2} 的电压电平。如果电压电平正常，电平转换器将使能且正常工作。不正常的 V_{DD} 和 V_{CC2} 电压电平将导致电平转换器除能，避免系统故障和可能的电路损害。注意只有当 $9V<V_{CC2}<20V$ 以及 $V_{DD}>3V$ 时，输出信号“RDY”才为高，否则“RDY”信号都为低。



高电压驱动器方框图

比较器 & 运算放大器

该单片机内建四个比较器和一个运算放大器。



比较器 & 运算放大器方框图

比较器

有四个比较器用于同步信号检测、反压保护、浪涌电压检测和过流检测。因比较器功能引脚与其它功能共用引脚，因此在使能比较器功能前，需确保已通过引脚共用功能选择位选择了比较器引脚功能。

比较器寄存器

比较器的功能和操作由一系列寄存器控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
CMP0C	C0COFM	C0CRS	C0COF5	C0COF4	C0COF3	C0COF2	C0COF1	C0COF0
CMPnC (n=1~3)	CnCMPOP	CnCOFM	CnCRS	CnCOF4	CnCOF3	CnCOF2	CnCOF1	CnCOF0
CMPVREF0	—	CVREF22	CVREF21	CVREF20	—	CVREF12	CVREF11	CVREF10
CMPVREF1	—	—	—	—	—	CVREF32	CVREF31	CVREF30
CMPCTL0	C3VOINV	C2VOINV	—	INTINV	C3EN	C2EN	C1EN	C0EN
CMPCTL1	C0CMPOP	C3CTL0	C2CTL1	C2CTL0	—	C1CTL	—	C3CTL1
CMPDBC0	—	—	C0DBC5	C0DBC4	C0DBC3	C0DBC2	C0DBC1	C0DBC0
CMPDBC1	C3DBEN	C2DBEN	C1DBEN	—	—	—	—	—
CMPHYS	—	—	—	—	C3HYSON	C2HYSON	C1HYSON	C0HYSON
C3DA	D7	D6	D5	D4	D3	D2	D1	D0
C3LEBC	C3LEBEN	C3IS	C3LD5	C3LD4	C3LD3	C3LD2	C3LD1	C3LD0

比较器寄存器列表

- **CMP0C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	C0COFM	C0CRS	C0COF5	C0COF4	C0COF3	C0COF2	C0COF1	C0COF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **C0COFM**：比较器 0 输入失调电压校准模式 / 比较器模式选择
0：比较器模式
1：输入失调电压校准模式

Bit 6 **C0CRS**：比较器 0 输入失调电压校准参考输入选择位
0：选择内部 0V 作为参考输入
1：选择比较器正相输入作为参考输入

Bit 5~0 **C0COF5~C0COF0**：比较器 0 输入失调电压校准设置

- **CMPnC 寄存器 (n=1~3)**

Bit	7	6	5	4	3	2	1	0
Name	CnCMPOP	CnCOFM	CnCRS	CnCOF4	CnCOF3	CnCOF2	CnCOF1	CnCOF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

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

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

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

Bit 4~0 **CnCOF4~CnCOF0**：比较器 n 输入失调电压校准设置

- **CMPVREF0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	CVREF22	CVREF21	CVREF20	—	CVREF12	CVREF11	CVREF10
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7　未定义，读为“0”

Bit 6~4　**CVREF22~CVREF20**：比较器 2 内部参考电压 V_{R2} 选择
000：0.600V_{DD}
001：0.625V_{DD}
010：0.650V_{DD}
011：0.675V_{DD}
100：0.700V_{DD}
101：0.725V_{DD}
110：0.750V_{DD}
111：0.775V_{DD}

Bit 3　未定义，读为“0”

Bit 2~0　**CVREF12~CVREF10**：比较器 1 内部参考电压 V_{R1} 选择
000：0.600V_{DD}
001：0.625V_{DD}
010：0.650V_{DD}
011：0.675V_{DD}
100：0.700V_{DD}
101：0.725V_{DD}
110：0.750V_{DD}
111：0.775V_{DD}

- **CMPVREF1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	CVREF32	CVREF31	CVREF30
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3　未定义，读为“0”

Bit 2~0　**CVREF32~CVREF30**：比较器 2 内部参考电压 V_{R3} 选择
000：0.075V_{DD}
001：0.100V_{DD}
010：0.125V_{DD}
011：0.150V_{DD}
100：0.175V_{DD}
101：0.200V_{DD}
110：0.225V_{DD}
111：0.250V_{DD}

- **CMPCTL0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	C3VOINV	C2VOINV	—	INTINV	C3EN	C2EN	C1EN	C0EN
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 **C3VOINV**：比较器 3 输出信号反相控制
0：同相
1：反相

Bit 6 **C2VOINV**：比较器 2 输出信号反相控制
0：同相
1：反相

Bit 5 未定义，读为“0”

Bit 4 **INTINV**：去抖外部中断输入信号反相控制
0：同相
1：反相

Bit 3 **C3EN**：CMP3 使能控制位
0：除能
1：使能
如果该位清零，比较器 3 将除能并且不会产生功耗。如果该位置位，比较器 3 的电源开启。

Bit 2 **C2EN**：CMP2 使能控制位
0：除能
1：使能
如果该位清零，比较器 2 将除能并且不会产生功耗。如果该位置位，比较器 2 的电源开启。

Bit 1 **C1EN**：CMP1 使能控制位
0：除能
1：使能
如果该位清零，比较器 1 将除能并且不会产生功耗。如果该位置位，比较器 1 的电源开启。

Bit 0 **C0EN**：CMP0 使能控制位
0：除能
1：使能
如果该位清零，比较器 0 将除能并且不会产生功耗。如果该位置位，比较器 0 的电源开启。

- **CMPCTL1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	C0CMPOP	C3CTL0	C2CTL1	C2CTL0	—	C1CTL	—	C3CTL1
R/W	R	R/W	R/W	R/W	—	R/W	—	R/W
POR	0	0	0	0	—	0	—	0

Bit 7 **C0CMPOP**：比较器 0 数字输出
0：正相输入电压 < 反相输入电压
1：正相输入电压 > 反相输入电压

Bit 6, 0 **C3CTL1~C3CTL0**：CMP3 反相引脚输入选择
C3CTL[1:0]=
00：CP0P
01：OPOUT
10/11：保留，不能使用

Bit 5~4 **C2CTL1~C2CTL0**：CMP2 反相引脚输入选择
00：选择 V_{R3} 作为反相输入，CP2P 作为同相输入
01：选择 CP2N 作为反相输入，V_{R2} 作为同相输入
10/11：保留

Bit 3 未定义，读为“0”

Bit 2 **C1CTL**：CMP1 反相引脚输入选择
0：CP0P
1：保留，不能使用

Bit 1 未定义，读为“0”

- **CMPDBC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	C0DBC5	C0DBC4	C0DBC3	C0DBC2	C0DBC1	C0DBC0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 未定义，读为“0”

Bit 5~0 **C0DBC5~C0DBC0**：外部中断输入去抖时间选择 (f_{PPGDCK}=8MHz)
000000：无去抖
000001：0~1/f_{PPGDCK}，约 0.125μs
000010：1/f_{PPGDCK}~2/f_{PPGDCK}，约 0.25μs
:
101111：46/f_{PPGDCK}~47/f_{PPGDCK}，约 5.875μs
110000~111111：47/f_{PPGDCK}~48/f_{PPGDCK}，约 6μs
设定去抖时，需要在 C0VO=0 (INTS=1) 或 PPGIN=0 (INTS=0) 时进行切换 (C0VO =0：CMP0 除能或者 CMP0 使能且 CMP0 正相输入信号 < 反相输入信号)，否则会产生 INT0 信号。

- **CMPDBC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	C3DBEN	C2DBEN	C1DBEN	—	—	—	—	—
R/W	R/W	R/W	R/W	—	—	—	—	—
POR	0	0	0	—	—	—	—	—

Bit 7 **C3DBEN**：比较器 3 去抖使能控制位
0：除能
1：使能
f_{PPGDCK}=8MHz，3/f_{PPGDCK}~4/f_{PPGDCK}，约 0.5μs
设定去抖时，需要在 C3VO=0 时进行切换 (C3VO=0：CMP3 除能或者 CMP3 使能且 CMP3 正相输入信号 < 反相输入信号)，否则会产生 CMP3INT 信号。

Bit 6 **C2DBEN**：比较器 2 去抖使能控制位
0：除能
1：使能
f_{PPGDCK}=8MHz，3/f_{PPGDCK}~4/f_{PPGDCK}，约 0.5μs
设定去抖时，需要在 C2VO=0 时进行切换 (C2VO=0：CMP2 除能或者 CMP2 使能且 CMP2 正相输入信号 < 反相输入信号)，否则会产生 CMP2INT 信号。

Bit 5 **C1DBEN**：比较器 1 去抖使能控制位
0：除能
1：使能
f_{PPGDCK}=8MHz，3/f_{PPGDCK}~4/f_{PPGDCK}，约 0.5μs
设定去抖时，需要在 C1VO=0 时进行切换 (C1VO=0：CMP1 除能或者 CMP1 使能且 CMP2 正相输入信号 < 反相输入信号)，否则会产生 CMP1INT 信号。

Bit 4~0 未定义，读为“0”

- **CMPHYS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	C3HYSON	C2HYSON	C1HYSON	C0HYSON
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 **C3HYSON**：比较器 3 迟滞使能控制位
0：除能
1：使能

Bit 2 **C2HYSON**：比较器 2 迟滞使能控制位
0：除能
1：使能

Bit 1 **C1HYSON**：比较器 1 迟滞使能控制位
0：除能
1：使能

Bit 0 **C0HYSON**：比较器 0 迟滞使能控制位
0：除能
1：使能

- **C3DA 寄存器**

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

Bit 7 ~ 0 **D7 ~ D0**：比较器 3 D/A 转换器输出电压控制位
D/A 转换器 V_{OUT}=(D/A 转换器 V_{REF}/256)×D[7:0]

- **C3LEBC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	C3LEBEN	C3IS	C3LD5	C3LD4	C3LD3	C3LD2	C3LD1	C3LD0
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 **C3LEBEN**：台阶检测启动位
0：除能
1：使能

Bit 6 **C3IS**：比较器 3 中断信号选择位
0：C3VOD
1：C3VL

Bit 5 ~ 0 **C3LD5 ~ C3LD0**：C3LEBEN 使能 LEB 的延迟时间
f_{SYS}=16MHz 时的数字延迟时间：
000000：旁路数字延迟电路
000001：0~2/f_{SYS}，大约 0.125μs
000010：2/f_{SYS}~4/f_{SYS}，大约 0.25μs
:
101111：92/f_{SYS}~94/f_{SYS}，大约 5.875μs
11xxxx：94/f_{SYS}~96/f_{SYS}，大约 6μs

比较器 n 输入失调校准功能 (n=0~3；若 n=0, m=5；若 n=1~3, m=4)

比较器具有输入失调校准功能。比较器输入失调电压校准的数据存储在 CnCOF[m:0] 位中。CnCOFM 是校准模式控制位，CnCRS 用于表明校准模式下的输入参考电压源。CnEN 用于使能 / 除能比较器。

比较器 n 校准步骤

比较器 n 失调电压校准的步骤如下。注意，在执行比较器 n 失调校准之前，应通过设置 CnHYSON=0 来除能迟滞电压。

1. 设置 CnEN=1，CnCOFM=1 和 CnCRS=1，比较器 n 进入失调电压校准模式。了确保校准后的 V_{CS} 尽可能小，校准模式下的输入参考电压应该跟正常模式下的输入直流工作电压相同。
2. 设置 CnCOF[m:0]=000000 或 =00000，在一定延迟时间后读取 CnCMPOP 位。
3. 将 CnCOF[m:0] 值加一，在一定延迟时间后再读取 CnCMPOP 位。

   若 CnCMPOP 位状态未改变，重复步骤 3，直到 CnCMPOP 位发生改变

   若 CnCMPOP 位状态改变，则记录下 CnCOF[m:0] 值为 V_{CS1}，前往步骤 4。
4. 设置 CnCOF[m:0]=111111 或 11111，在一定延迟时间后读取 CnCMPOP 位。
5. 将 CnCOF[m:0] 值减一，在一定延迟时间后再读取 CnCMPOP 位。

   若 CnCMPOP 位状态未改变，重复步骤 5 直到 CnCMPOP 位状态改变。

   若 CnCMPOP 位状态改变，则记录下 CnCOF[m:0] 值为 V_{CS2}，前往步骤 6。
6. 将比较器输入失调校准值 V_{CS} 重新存入 CnCOF[m:0] 位。此时失调校准步骤完成。

   $V_{CS}=(V_{CS1}+V_{CS2})/2$。若 $(V_{CS1}+V_{CS2})/2$ 的值不是整数，则去掉小数部分。

运算放大器

该单片机内部集成了一个运算放大器，可用于放大微小的模拟输入信号。OPINP 是 OPAMP 的正相输入，OPINN 是 OPAMP 的反相输入。OPOUT 和 OPROUT 是 OPAMP 模拟电压输出引脚。OPEN 用于使能或除能 OPAMP。因 OPAMP 功能引脚与其它功能共用引脚，因此在使能 OPAMP 功能前，需确保已通过引脚共用功能选择位选择了 OPAMP 引脚功能。

运算放大器寄存器

运算放大器的功能和操作由三个寄存器控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
OPC	OPO	OPEN	—	—	OPG1	OPG0	—	—
OPVOS	OOFM	ORSP	OOF5	OOF4	OOF3	OOF2	OOF1	OOF0
OPS	—	—	—	OPS4	OPS3	OPS2	OPS1	OPS0

运算放大器寄存器列表

- **OPC 寄存器**

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

Bit 7 **OPO**：输入失调电压校准模式下的 OPAMP 数字输出
0：正相输入电压 < 反相输入电压
1：正相输入电压 > 反相输入电压

Bit 6 **OPEN**：OPAMP 使能 / 除能选择位
0：除能
1：使能

Bit 5~4 未定义，读为“0”

Bit 3~2 **OPG1~OPG0**：R2/R1 比值选择位
00：R2/R1=20
01：R2/R1=30
10：R2/R1=40
11：R2/R1=60
因这些位对增益的选择是通过电阻 R1 和 R2 来实现。因此在使用这些位选择增益时，如果选择 OPINN 输入，需保证 SW0 开关为 ON。否则，无法保证增益精度。(R2=OPAR1；R1=OPAR3)

Bit 1~0 未定义，读为“0”

- **OPVOS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OOFM	ORSP	OOF5	OOF4	OOF3	OOF2	OOF1	OOF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

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

Bit 6 **ORSP**：OPAMP 输入失调电压校准参考选择位
0：反相输入端作为参考输入
1：正相输入端作为参考输入

Bit 5~0 **OOF5~OOF0**：OPAMP 输入失调电压校准控制位

- **OPS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	OPS4	OPS3	OPS2	OPS1	OPS0
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 **OPS4**：OPAMP 开关 SW4 On/Off 控制
0：Off
1：On

Bit 3 **OPS3**：OPAMP 开关 SW3 On/Off 控制
0：Off
1：On

Bit 2 **OPS2**：OPAMP 开关 SW2 On/Off 控制
0：Off
1：On

Bit 1 **OPS1**：OPAMP 开关 SW1 On/Off 控制
0：Off
1：On

Bit 0 **OPS0**：OPAMP 开关 SW0 On/Off 控制
0：Off
1：On

运算放大器输入失调校准

运算放大器具有输入失调校准功能。运算放大器输入失调电压校准的数据存储在 OOF[5:0] 位中。OOFM 是校准模式控制位，ORSP 用于表明校准模式下的输入参考电压源。运算放大器数字输出标志是 OPO，用于 OPAMP 输入失调电压校准模式。

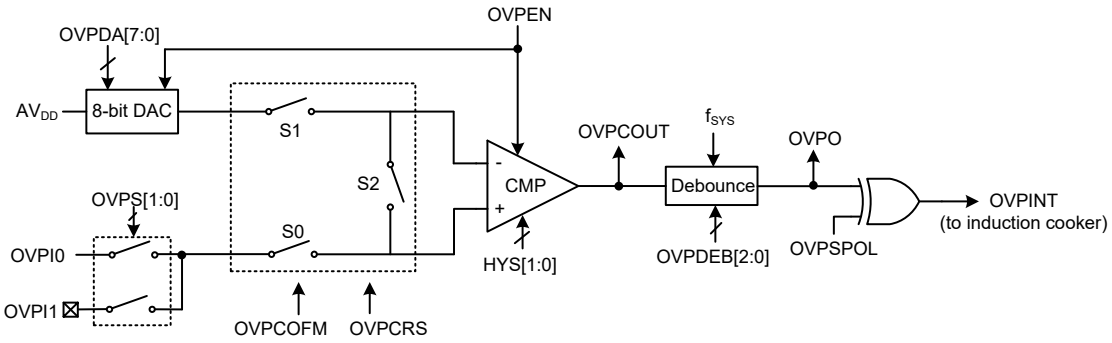
运算放大器校准步骤

运算放大器失调电压校准的步骤如下。

1. 设置 OOFM=1 且 ORSP=1，OPAMP 进入失调电压校准模式。为了确保校准后的 V_{OS} 尽可能小，校准模式下的输入参考电压应该跟正常模式下的输入直流工作电压相同。
2. 设置 OOF[5:0]=000000，在一定延迟时间后读取 OPO 位。
3. 将 OOF[5:0] 值加一，在一定延迟时间后再读取 OPO 位
   若 OPO 位状态不变，重复步骤 3，直到 OPO 位发生改变
   若 OPO 位状态改变，则记录下 OOF[5:0] 值为 V_{OS1}，前往步骤 4。
4. 设置 OOF[5:0]=111111，在一定延迟时间后读取 OPO 位。
5. 将 OOF[5:0] 值减一，在一定延迟时间后再读取 OPO 位
   若 OPO 位状态不变，重复步骤 3，直到 OPO 位发生改变
   若 OPO 位状态改变，则记录下 OOF[5:0] 值为 V_{OS2}，前往步骤 6。
6. 将运算放大器输入失调校准值 V_{OS} 重新存入 OOF[5:0] 位。此时失调校准步骤完成。
   $V_{OS}=(V_{OS1}+V_{OS2})/2$，若 $(V_{OS1}+V_{OS2})/2$ 的值不是整数，则去掉小数部分。

过压保护 – OVP

此单片机包含一个过压保护功能，即 OVP，为应用提供保护机制或产生输出信号用于过零检测应用。为了防止工作电压超过特定值，可将 OVP 输入的电压与 8-bit D/A 转换器产生的参考电压进行比较。当有超过预设电压的情况发生时，OVP 的输出会发生翻转。



过压保护电路

注：1. 如果由 OVPI1 引脚提供输入源，由于 OVPI1 引脚与其它功能共用引脚，因此在使能 OVP 功能前，需确保已通过引脚共用功能选择位选择了 OVPI1 引脚功能。
2. OVPI0 输入源来自 CP2N 引脚。
3. S0、S1 和 S2 开关的 On/Off 控制，如下表所示：

OVPCOFM	OVPCRS	S0	S1	S2
0	x	On	On	Off
1	0	Off	On	On
1	1	On	Off	On
“x”：无关				

过压保护操作

输入电压由 OVPI0 或 OVPI1 提供，连接到比较器的正相输入端。D/A 转换器用于产生参考电压。比较器将参考电压与输入电压进行比较，以产生 OVPCOUT 信号。

过压保护控制寄存器

过压保护的控制由几个寄存器控制。其中一个寄存器为过压保护电路提供参考电压。剩余三个寄存器用于 OVP 功能的控制、D/A 转换器参考电压选择、比较器去抖时间选择、比较器迟滞控制、OVP 输入选择以及比较器输入失调校准控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
OVPC0	OVPO	OVPSPOL	OVPEN	—	—	OVPDEB2	OVPDEB1	OVPDEB0
OVPC1	OVPCOUT	OVPCOFM	OVPCRS	OVPCOF4	OVPCOF3	OVPCOF2	OVPCOF1	OVPCOF0
OVPC2	—	—	—	—	HYS1	HYS0	OVPS1	OVPS0
OVPDA	D7	D6	D5	D4	D3	D2	D1	D0

OVP 寄存器控制

- **OVPC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OVPO	OVPSPOL	OVPEN	—	—	OVPDEB2	OVPDEB1	OVPDEB0
R/W	R	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

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

Bit 6 **OVPSPOL**：OVPO 极性控制位
0：同相
1：反相

Bit 5 **OVPEN**：OVP 功能控制
0：除能
1：使能
若 OVPEN 位清零，过压保护功能除能不产生功耗。此时，过电压保护功能中的比较器和 D/A 转换器都关闭。

Bit 4~3 未定义，读为“0”

Bit 2~0 **OVPDEB2~OVPDEB0**：OVP 比较器去抖时间控制位
000：无去抖
001：(1~2) × t_{DEB}
010：(3~4) × t_{DEB}
011：(7~8) × t_{DEB}
100：(15~16) × t_{DEB}
101：(31~32) × t_{DEB}
110：(63~64) × t_{DEB}
111：(127~128) × t_{DEB}
注：$t_{DEB}=1/f_{SYS}$。

- **OVPC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OVPCOUT	OVPCOFM	OVPCRS	OVPCOF4	OVPCOF3	OVPCOF2	OVPCOF1	OVPCOF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **OVPCOUT**：OVP 比较器输出位 (无去抖)
0：正端输入电压 < 负端输入电压
1：正端输入电压 > 负端输入电压

Bit 6 **OVPCOFM**：OVP 比较器正常操作或输入失调电压校准模式选择
0：正常操作
1：输入失调电压校准模式

Bit 5 **OVPCRS**：OVP 比较器输入失调电压校准参考电压选择
0：输入参考电压来自负端输入
1：输入参考电压来自正端输入

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

- **OVPC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HYS1	HYS0	OVPS1	OVPS0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未定义，读为“0”

Bit 3~2　**HYS1~HYS0**：OVP 比较器迟滞电压窗口控制位
具体参考“过电压保护电气特性”表章节。

Bit 1~0　**OVPS1~OVPS0**：OVP 输入选择位
00：保留
01：OVPI0
10：OVPI1
11：保留
OVPI0 输入源来自 CP2N 引脚。

- **OVPDA 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　OVP D/A 转换器输出电压控制位
D/A 转换器输出：D/A 转换器 V_{OUT}=(D/A 转换器参考电压 / 256) × OVPDA[7:0]

输入失调校准

OVPC1 寄存器的 OVPCOFM 位用于选择 OVP 比较器的工作模式，即正常操作或输入失调校准模式。若此位置高，比较器进入失调电压校准模式。在进入校准模式之前，应通过设置 HYS[1:0]=00B 以确保无迟滞电压。由于 OVPI 引脚与 I/O 口或其它引脚功能共用，需事前将其设置为比较器输入功能。OVP 比较器输入失调电压校准步骤如下所示。

步骤 1. 设置 OVPCOFM=1，OVPCRS=1，使 OVP 比较器工作于校准模式，开关 S0 和 S2 都 On。为了确保校准后的 V_{OS} 尽可能小，校准模式下的输入参考电压应该跟正常模式下的输入直流工作电压相同。

步骤 2. 设置 OVPCOF[4:0]=00000，延迟一段时间后再读取 OVPCOUT 位的状态。

步骤 3. 使 OVPCOF[4:0]=OVPCOF[4:0]+1，延迟一段时间后再读取位的状态。
若 OVPCOUT 位状态不变，重复步骤 3，直到 OVPCOUT 位发生改变
若 OVPCOUT 位状态改变，则记录下 OVPCOF[4:0] 值为 V_{CS1}，前往步骤 4。

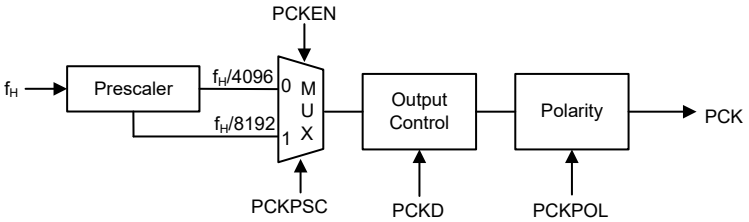
步骤 4. 设置 OVPCOF[4:0]=11111，延迟一段时间后再读取 OVPCOUT 位的状态。

步骤 5. 使 OVPCOF[4:0]=OVPCOF[4:0]-1，延迟一段时间后再读取 OVPCOUT 位的状态。
若 OVPCOUT 位状态不变，重复步骤 5，直到 OVPCOUT 位发生改变
若 OVPCOUT 位状态改变，则记录下 OVPCOF[4:0] 值为 V_{CS2}，前往步骤 6。

步骤 6. 将 $V_{OS}=(V_{CS1}+V_{CS2})/2$ 存入 OVPCOF[4:0] 位中，校准结束。
若 $(V_{CS1}+V_{CS2})/2$ 不是整数，忽略小数部分。

外围时钟输出

外围时钟输出功能使单片机能够为外部硬件提供和单片机时钟同步的时钟信号。



外围时钟输出方框图

外围时钟输出操作

外围时钟输出引脚 PCK 与 PB1 引脚共用，可以通过相关引脚共用控制位选择该引脚功能。外围时钟输出功能由 PCKC 寄存器控制。设置完 PCKEN 和 PCKPOL 位后，将 PCKD 位置 1 会使能 PCK 输出功能，将 PCKD 位清零会除能 PCK 输出功能，且根据 PCKPOL 位的设置将 PCK 引脚输出强制拉低或拉高。

外围时钟输出的时钟源来自 f_H 分频，分频比由 PCKC 寄存器中的 PCKPSC 位选择。

PCK 输出真值表如下：

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

外围时钟输出寄存器

外围时钟输出功能由 PCKC 寄存器控制。

- **PCKC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PCKD	PCKPOL	PCKEN	—	—	—	PCKPSC
R/W	—	R/W	R/W	R/W	—	—	—	R/W
POR	—	0	0	0	—	—	—	0

Bit 7　未定义，读为“0”

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

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

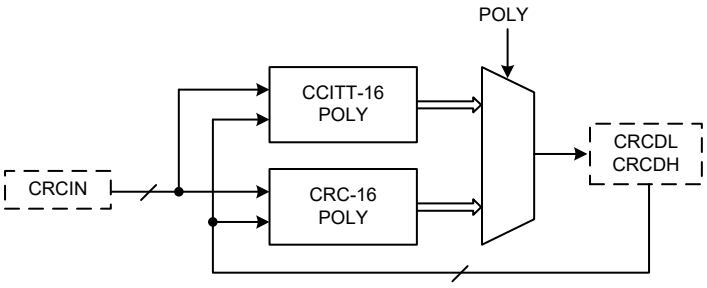
Bit 4 **PCKEN**：PCK 功能控制
0：除能
1：使能

Bit 3~1 未定义，读为“0”

Bit 0 **PCKPSC**：外围时钟 f_{PCK} 分频比选择
0：f_H/4096
1：f_H/8192

循环冗余校验 – CRC

循环冗余校验 (CRC) 计算单元是一种错误检测技术测试算法，用于验证数据传输或存储数据的正确性。CRC 计算将数据流或数据块作为输入，并生成一个 16-bit 的输出余数。通常情况下，一个数据流带有 CRC 后缀码，当被发送或存储时该数据流可用作校验和。因此，被接收或重新存储的数据流都是通过上述相同的生成多项式计算得到的，详情见下方章节。



CRC 方框图

CRC 寄存器

CRC 发生器包含了一个 8-bit CRC 数据输入寄存器 CRCIN 和 CRC 校验和寄存器对 CRCDH 和 CRCDL。CRCIN 寄存器用于输入新数据，而 CRCDH 和 CRCDL 寄存器用于保持前一个 CRC 计算结果。CRCCR 控制寄存器用于选择使用哪一个 CRC 生成多项式。

寄存器名称	位							
	7	6	5	4	3	2	1	0
CRCCR	—	—	—	—	—	—	—	POLY
CRCIN	D7	D6	D5	D4	D3	D2	D1	D0
CRCDL	D7	D6	D5	D4	D3	D2	D1	D0
CRCDH	D15	D14	D13	D12	D11	D10	D9	D8

CRC 寄存器列表

- **CRCCR 寄存器**

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

Bit 7~1　未定义，读为“0”

Bit 0　**POLY**：16-bit CRC 生成多项式选择
0：CRC-CCITT：$X^{16} + X^{12} + X^5 + 1$
1：CRC-16：$X^{16} + X^{15} + X^2 + 1$

- **CRCIN 寄存器**

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

Bit 7~0　**D7~D0**：CRC 输入数据寄存器

- **CRCDL 寄存器**

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

Bit 7~0　**D7~D0**：16-bit CRC 校验和低字节数据寄存器

- **CRCDH 寄存器**

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

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

CRC 操作

CRC 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。

低电压检测 – LVD

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

LVD 寄存器

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

- **LVDC 寄存器**

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

Bit 7~6　未定义，读为“0”

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

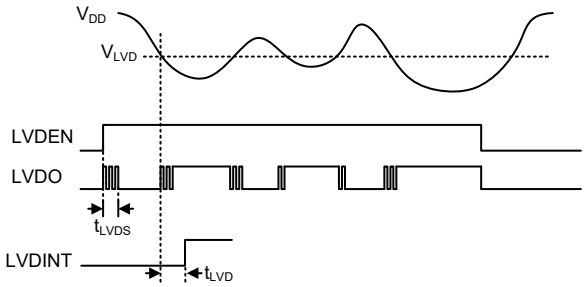
Bit 4　**LVDEN**：低电压检测功能控制位
0：除能
1：使能

Bit 3　**VBGEN**：Bandgap 缓冲器控制
0：除能
1：使能
当 LVD 或 LVR 使能或者 VBGEN 位置高时，Bandgap 电路使能。

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

LVD 操作

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



LVD 操作

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

中断

中断是单片机一个重要功能。当内部功能如定时器或 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。此单片机提供多个内部中断功能，如定时器、比较器、LVD、EEPROM 和 A/D 转换器等产生。

中断寄存器

中断控制基本上是在一定单片机条件发生时设置请求标志位，应用程序中中断使能位的设置是通过位于特殊功能数据存储器中的 INTC0~INTC3 寄存器控制的。

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，紧接着是中断号 (可选)，最后的字母“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	—	—
OVP		OVPE	OVPF	—
比较器		CPnE	CPnF	n=1~3
PPG	PPGINT 中断	PPGINTF	PPGINTE	—
	PPGTIMER 中断	PPGTMF	PPGTME	—
	PPGATCD 中断	PPGATCDF	PPGATCDE	—
	PPG 重复触发中断	PPGRNE	PPGRNF	—
A/D 转换器		ADE	ADF	—
EEPROM		DEE	DEF	—
LVD		LVE	LVF	—
定时 / 事件计数器		TnE	TnF	n=0~2
I^2C		IICE	IICF	—

中断寄存器位命名模式

寄存器名称	位							
	7	6	5	4	3	2	1	0
INTC0	—	PPGINTF	CP1F	OVPF	PPGINTE	CP1E	OVPE	EMI
INTC1	CP3F	CP2F	ADF	T0F	CP3E	CP2E	ADE	T0E
INTC2	DEF	T2F	T1F	LVF	DEE	T2E	T1E	LVE
INTC3	IICF	PPGRNF	PPGATCDF	PPGTMF	IICE	PPGRNE	PPGATCDE	PPGTME

中断寄存器列表

- **INTC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PPGINTF	CP1F	OVPF	PPGINTE	CP1E	OVPE	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　**PPGINTF**：PPGINT 中断请求标志位
0：无请求
1：中断请求

Bit 5　**CP1F**：比较器 1 中断请求标志位
0：无请求
1：中断请求

Bit 4　**OVPF**：OVP 中断请求标志位
0：无请求
1：中断请求

Bit 3　**PPGINTE**：PPGINT 中断控制位
0：除能
1：使能

Bit 2　**CP1E**：比较器 1 中断控制位
0：除能
1：使能

Bit 1　**OVPE**：OVP 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	CP3F	CP2F	ADF	T0F	CP3E	CP2E	ADE	T0E
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　**CP3F**：比较器 3 中断请求标志位
0：无请求
1：中断请求

Bit 6　**CP2F**：比较器 2 中断请求标志位
0：无请求
1：中断请求

Bit 5 **ADF**：A/D 转换器中断请求标志位
0：无请求
1：中断请求

Bit 4 **T0F**：定时 / 事件计数器 0 转换器中断请求标志位
0：无请求
1：中断请求

Bit 3 **CP3E**：比较器 3 中断控制位
0：除能
1：使能

Bit 2 **CP2E**：比较器 2 中断控制位
0：除能
1：使能

Bit 1 **ADE**：A/D 转换器中断控制位
0：除能
1：使能

Bit 0 **T0E**：定时 / 事件计数器 0 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	DEF	T2F	T1F	LVF	DEE	T2E	T1E	LVE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **DEF**：数字 EEPROM 中断请求标志位
0：无请求
1：中断请求

Bit 6 **T2F**：定时 / 事件计数器 2 请求标志位
0：无请求
1：中断请求

Bit 5 **T1F**：定时 / 事件计数器 1 请求标志位
0：无请求
1：中断请求

Bit 4 **LVF**：LVD 中断请求标志位
0：无请求
1：中断请求

Bit 3 **DEE**：数字 EEPROM 中断控制位
0：除能
1：使能

Bit 2 **T2E**：定时 / 事件计数器 2 控制位
0：除能
1：使能

Bit 1 **T1E**：定时 / 事件计数器 1 控制位
0：除能
1：使能

Bit 0 **LVE**：LVD 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	IICF	PPGRNF	PPGATCDF	PPGTMF	IICE	PPGRNE	PPGATCDE	PPGTME
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 **IICF**：I^2C 中断请求标志位
0：无请求
1：中断请求

Bit 6 **PPGRNF**：PPG 重复触发中断请求标志位
0：无请求
1：中断请求

Bit 5 **PPGATCDF**：PPGATCD 中断请求标志位
0：无请求
1：中断请求

Bit 4 **PPGTMF**：PPGTIMER 中断请求标志位
0：无请求
1：中断请求

Bit 3 **IICE**：I^2C 中断控制位
0：除能
1：使能

Bit 2 **PPGRNE**：PPG 重复触发中断控制位
0：除能
1：使能

Bit 1 **PPGATCDE**：PPGATCD 中断控制位
0：除能
1：使能

Bit 0 **PPGTME**：PPGTIMER 中断控制位
0：除能
1：使能

中断操作

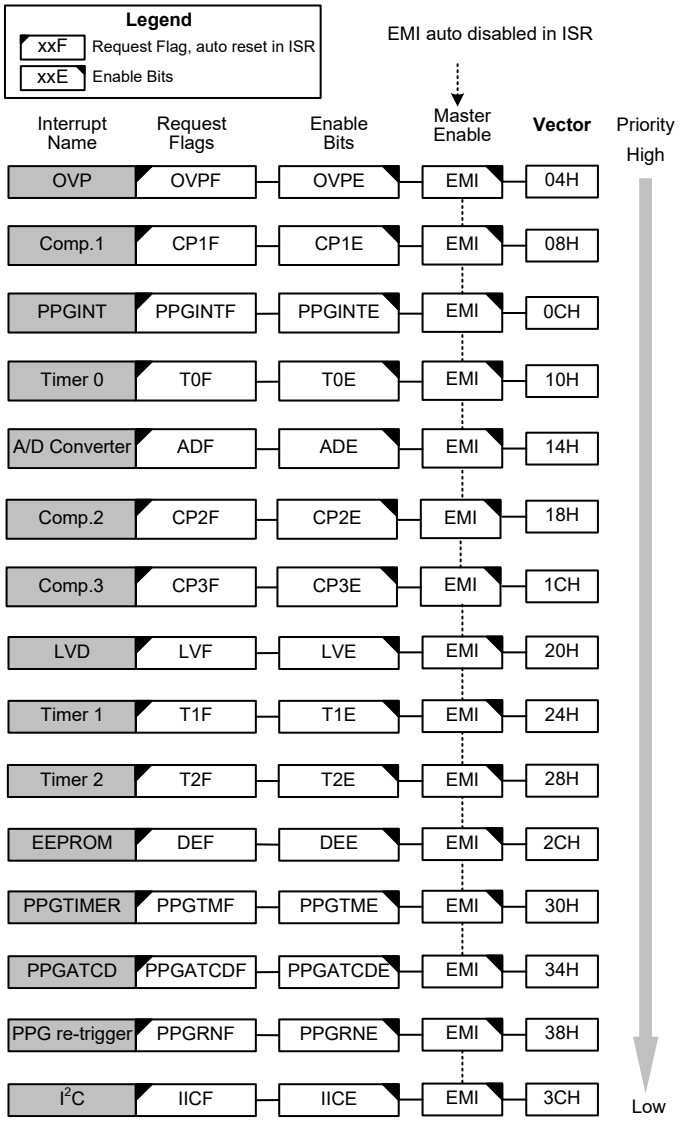
若中断事件条件产生，如一个定时 / 事件计数器 n 溢出或 A/D 转换结束等等，相关中断请求标志将置起。中断标志产生后程序是否会跳转至相关中断向量执行是由中断使能位的条件决定的。若使能位为“1”，程序将跳至相关中断向量中执行；若使能位为“0”，即使中断请求标志置起中断也不会发生，程序也不会跳转至相关中断向量执行。若总中断使能位为“0”，所有中断都将除能。

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

各个中断使能位以及相应的请求标志位，以优先级的次序显示在下图。所有的中断源都有自己的向量。一旦中断子程序被响应，系统将自动清除 EMI 位，所有其它的中断将被屏蔽，这个方式可以防止任何进一步的中断嵌套。其它中断请求可能发生在此期间，虽然中断不会立即响应，但是中断请求标志位会被记录。

如果某个中断服务子程序正在执行时，有另一个中断要求立即响应，那么 EMI 位应在程序进入中断子程序后置位，以允许此中断嵌套。如果堆栈已满，即使此中断使能，中断请求也不会被响应，直到 SP 减少为止。如果要求立刻动作，则堆栈必须避免成为储满状态。请求同时发生时，执行优先级如下流程图所示。

所有被置起的中断请求标志都可把单片机从休眠或空闲模式中唤醒，若要防止唤醒动作发生，在单片机进入休眠或空闲模式前应将相应的标志置起。



中断结构

OVP 中断

OVP 中断，即过电压保护中断。当有过电压事件发生时，OVP 中断请求标志位 OVPF 被置位，OVP 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 OVP 中断使能位 OVPE 需先被置位。当中断使能，堆栈未满并且发生过电压输入时，将调用相应的 OVP 中断向量子程序。当响应 OVP 中断服务子程序时，OVP 中断的中断请求标志位会自动复位且 EMI 位会被清零以除能其它中断。

比较器中断

该单片机有三个比较器中断，由内部比较器 CMP1~CMP3 控制。当相应的比较器输出位的状态改变，相应的比较器中断请求标志 CPnF 被置位，比较器中断

请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和比较器中断使能位 CPnE 需先被置位。当中断使能，堆栈未满并且以上任何一种情况发生使得比较器输出位的状态发生改变时，将调用相应的比较器中断向量子程序。当中断服务子程序响应时，比较器中断请求标志位 CPnF 会自动复位且 EMI 位会被清零以除能其它中断。

A/D 转换器中断

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

EEPROM 中断

当写周期结束，EEPROM 中断请求标志 DEF 被置位，EEPROM 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和 EEPROM 中断使能位 DEE 需先被置位。当中断使能，堆栈未满且 EEPROM 写周期结束时，可跳转至 EEPROM 中断向量子程序中执行。当 EEPROM 中断响应，相应的中断请求标志位 DEF 会自动复位且 EMI 位会被清零以除能其它中断。

LVD 中断

当低电压检测功能检测到一个低电压时，LVD 中断请求标志 LVF 被置位，LVD 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和低电压中断使能位 LVE 需先被置位。当中断使能，堆栈未满且低电压条件发生时，可跳转至 LVD 中断向量子程序中执行。当 LVD 中断响应，相应的中断请求标志位 LVF 会自动复位且 EMI 位会被清零以除能其它中断。

定时 / 事件计数器中断

当定时 / 事件计数器 n 溢出，相应的中断请求标志位 TnF 将置位并触发定时 / 事件计数器 n 中断。要产生定时 / 事件计数器 n 中断，总中断使能位 EMI 和相应的定时 / 事件计数器中断使能位 TnE 需先被置位。若中断使能，堆栈未满，当定时 / 事件计数器 n 溢出时，将调用相应定时器中断向量子程序。当定时 / 事件计数器 n 中断被响应时，相应的中断请求标志位 TnF 会自动复位且 EMI 位会自动清零以除能其它中断。

PPGINT 中断

当 PPG INT00 信号下降沿产生时，相应的中断请求标志位 PPGINTF 将置位并触发 PPGINT 中断。要产生 PPGINT 中断，总中断使能位 EMI 和相应的 PPGINT 中断使能位 PPGINTE 需先被置位。若中断使能，堆栈未满，当 PPG INT00 信号下降沿产生时，将调用 PPGINT 中断向量子程序。当 PPGINT 中断被响应时，中断请求标志位 PPGINTF 会自动复位且 EMI 位会自动清零以除能其它中断。

PPGTIMER 中断

当 PPGTIMER 溢出时，相应的中断请求标志位 PPGTMF 将置位并触发 PPGTIMER 中断。要产生 PPGTIMER 中断，总中断使能位 EMI 和相应的 PPGTIMER 中断使能位 PPGTME 需先被置位。若中断使能，堆栈未满，当 PPGTIMER 溢出时，

将调用 PPGTIMER 中断向量子程序。当 PPGTIMER 中断被响应时，中断请求标志位 PPGTMF 会自动复位且 EMI 位会自动清零以除能其它中断。

PPGATCD 中断

当 PPGTA 逼近 PPGTC/PPGTD 完成时，相应的中断请求标志位 PPGATCDF 将置位并触发 PPGATCD 中断。要产生 PPGATCD 中断，总中断使能位 EMI 和相应的 PPGATCD 中断使能位 PPGATCDE 需先被置位。若中断使能，堆栈未满，当 PPGTA 逼近 PPGTC/PPGTD 时，将调用 PPGATCD 中断向量子程序。当 PPGATCD 中断被响应时，中断请求标志位 PPGATCDF 会自动复位且 EMI 位会自动清零以除能其它中断。

PPG 重复触发中断

当达到 PPG 软件设定时间时，相应的中断请求标志位 PPGRNF 将置位并触发 PPG 重复触发中断。要产生 PPG 重复触发中断，总中断使能位 EMI 和相应的 PPG 重复触发中断使能位 PPGRNE 需先被置位。若中断使能，堆栈未满，当 PPG 软件重复触发使能时，将调用 PPG 重复触发中断向量子程序。当 PPG 重复触发中断被响应时，中断请求标志位 PPGRNF 会自动复位且 EMI 位会自动清零以除能其它中断。

I^2C 中断

当一个字节数据已由 I^2C 接口接收或发送完，或 I^2C 从机地址匹配，或 I^2C 超时，中断请求标志 IICF 被置位，I^2C 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 IICE 需先被置位。当中断使能，堆栈未满且以上任一种情况发生时，可跳转至相关中断向量子程序中执行。当响应中断服务子程序时，串行接口中断标志位 IICF 会自动复位且 EMI 将被自动清零以除能其它中断。

中断唤醒功能

每个中断都具有将处于休眠或空闲模式的单片机唤醒的能力。当中断请求标志由低到高转换时唤醒动作产生，其与中断是否使能无关。因此，尽管单片机处于休眠或空闲模式且系统振荡器停止工作，如有比较器输入改变或 A/D 转换完成都可能导致其相应的中断标志被置位，由此产生中断，因此必须注意避免伪唤醒情况的发生。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。

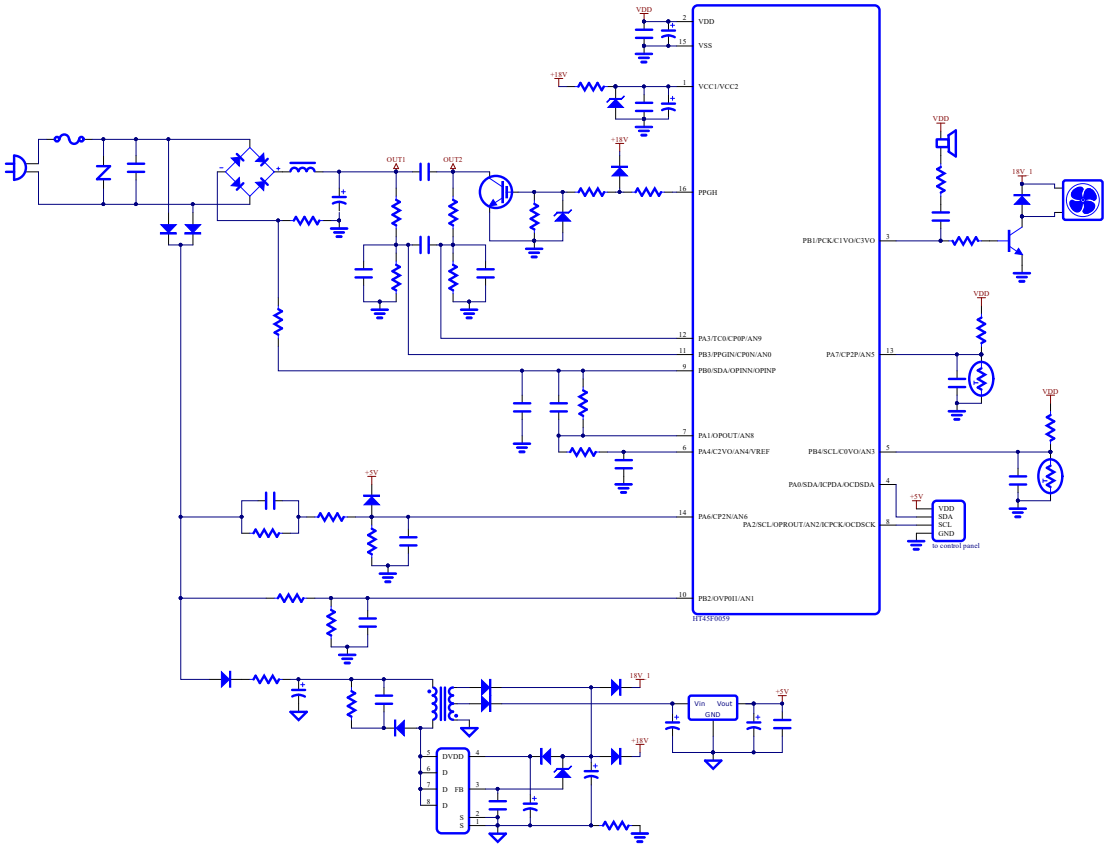
建议在中断服务子程序中不要使用“CALL 子程序”指令。中断通常发生在不可预料的情况或是需要立刻执行的某些应用。假如只剩下一层堆栈且没有控制好中断，当“CALL 子程序”在中断服务子程序中执行时，将破坏原来的控制序列。

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



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 $[m] \leftarrow ACC - [m] - \overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

LSDZ [m] Skip if Decrement Data Memory is 0

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

功能表示 [m] ← [m] – 1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

LSDZA [m] Skip if decrement Data Memory is zero with result in ACC

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

功能表示 ACC ← [m] – 1，如果 ACC=0 跳过下一条指令执行

影响标志位 无

LSET [m] Set Data Memory

指令说明 将指定数据存储器的每一个位置位为 1。

功能表示 [m] ← FFH

影响标志位 无

LSET [m].i Set bit of Data Memory

指令说明 将指定数据存储器的第 i 位置位为 1。

功能表示 [m].i ← 1

影响标志位 无

LSIZ [m] Skip if increment Data Memory is 0

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

功能表示 [m] ←[m] + 1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

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

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

功能表示 ACC ←[m] + 1，如果 ACC=0 跳过下一条指令执行

影响标志位 无

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

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

功能表示 如果 [m].i≠0，跳过下一条指令执行

影响标志位 无

LSNZ [m] Skip if Data Memory is not 0

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

功能表示 如果 [m]≠0，跳过下一条指令执行

影响标志位 无

LSUB A, [m] Subtract Data Memory from ACC

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

功能表示 ACC ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

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

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

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

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

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

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

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

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

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

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

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

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

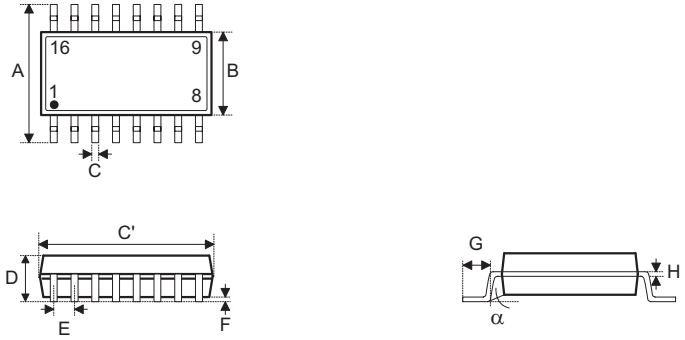
封装信息

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

封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

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

16-pin NSOP (150mil) 外形尺寸



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

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	—	6.00 BSC	—
B	—	3.90 BSC	—
C	0.31	—	0.51
C’	—	9.90 BSC	—
D	—	—	1.75
E	—	1.27 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°