12V 高电流驱动触控型单片机

BS45F5930

版本 : V1.00　日期 : 2018-02-22

www.holtek.com

目录

特性

CPU 特性

- 工作电压：
  - ♦ f_{SYS}=8MHz：4.75V~5.25V
  - ♦ V_{CC}=6V~12V @ ±10%
- V_{DD}=5V，系统时钟为 8MHz 时，指令周期为 0.5μs
- 提供暂停和唤醒功能，以降低功耗
- 振荡器类型：
  - ♦ 内部高速 8MHz RC – HIRC
  - ♦ 内部低速 32kHz RC – LIRC
- 多种工作模式：快速模式、低速模式、空闲模式和休眠模式
- 内部集成振荡器，无需外部元件
- 所有指令都可在 1 或 2 个指令周期内完成
- 查表指令
- 63 条指令
- 4 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器：2K×16
- 数据存储器：128×8
- 看门狗定时器功能
- 5 个双向 I/O 口
- 1 个与 I/O 口共用的外部中断输入
- 1 个 8-bit 可编程定时 / 事件计数器
- 1 个时基功能，可提供固定时间的中断信号
- 过压保护功能
- 2 个高压驱动器输出引脚
- 4 个触控按键功能
- 2 通道 8-bit PWM 输出功能
- Flash 程序存储器烧录可达 100,000 次
- Flash 程序存储器数据可保存 10 年以上
- 封装类型：8/10-pin SOP

概述

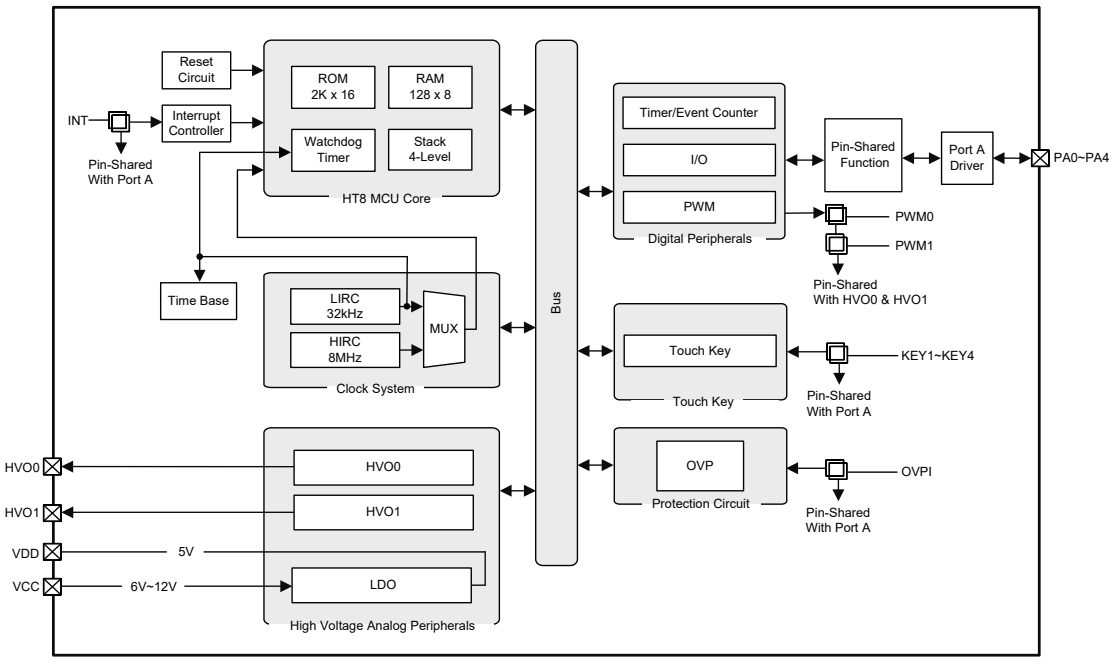
该单片机是一款 8 位具有高性能精简指令集且带有高达 12V 的高压驱动器的 Flash 型单片机。此单片机含有触控按键功能和可多次编程的 Flash 存储器特性，为各种触控按键的应用提供了一种简单而又有效的实现方法。

触控按键功能完全集成于单片机内，使用较少的外部元件便可实现触控按键的应用。该单片机除了 Flash 程序存储器，还包括数据存储器。内部看门狗定时器和过压保护功能具有良好的抗噪声和抗 ESD 保护功能，确保单片机在恶劣的电气环境中仍能保持稳定的操作。

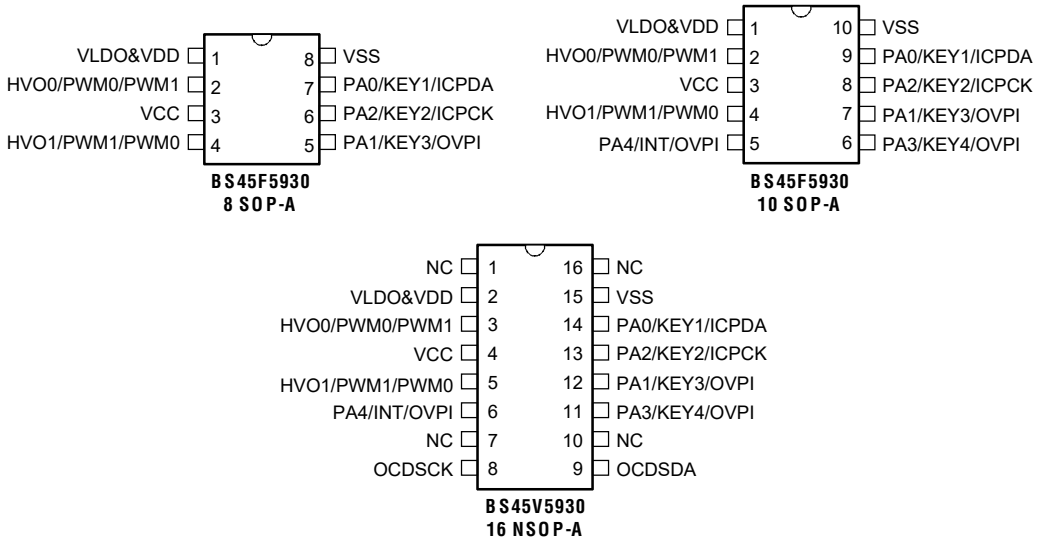
单片机内部集成了高 / 低速振荡器，在应用中不需增加外部元件。动态切换高低系统时钟的能力，为用户提供了优化单片机操作和降低功耗的能力。外加 I/O 灵活，时基、定时 / 事件计数器、脉宽调制器等其它特性，增强了该单片机的功能和灵活性。

该触控按键单片机能广泛应用于各种现代化的触控按键产品中，例如 LED 触摸台灯、触摸美甲光疗机及需求触摸整合 12V 大电流驱动的应用等等。

方框图



引脚图



注：1. 若共用脚有多种输出，所需引脚共用功能通过引脚共用寄存器中相应的软件控制位控制。
2. 16-pin NSOP 封装仅用于 OCDS EV 芯片，OCDSCK 和 OCDSDA 引脚为 OCDS 专用引脚。

引脚说明

引脚名称	功能	OPT	I/T	O/T	说明
PA0/KEY1/ICPDA	PA0	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	KEY1	PAS0 TKMC1	ST	CMOS	触控按键输入
	ICPDA	—	ST	CMOS	ICP 地址 / 数据
PA1/KEY3/OVPI	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
	KEY3	PAS0 TKMC1	ST	CMOS	触控按键输入
	OVPI	PAS0	ST	—	OVP 输入
PA2/KEY2/ICPCK	PA2	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	KEY2	PAS0 TKMC1	ST	CMOS	触控按键输入
	ICPCK	—	ST	CMOS	ICP 时钟
PA3/KEY4/OVPI	PA3	PAPU PAWU PAPS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	KEY4	PAS0 TKMC1	ST	CMOS	触控按键输入
	OVPI	PAS0	ST	—	OVP 输入

引脚名称	功能	OPT	I/T	O/T	说明
PA4/INT/OVPI	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
	INT	INTEG	ST	—	外部中断输入
	OVPI	PAS1	ST	—	OVP 输入
HVO0/PWM0/PWM1	HVO0	HVOC1	—	CMOS	高压驱动输出
	PWM0	HVOC1	—	CMOS	PWM 通道 0 输出
	PWM1	HVOC1	—	CMOS	PWM 通道 1 输出
HVO1/PWM1/PWM0	HVO1	HVOC1	—	CMOS	高压驱动输出
	PWM1	HVOC1	—	CMOS	PWM 通道 1 输出
	PWM0	HVOC1	—	CMOS	PWM 通道 0 输出
VCC	VCC	—	PWR	—	LDO 输入电源、高压驱动输出及电平转换器输入
VLDO/VDD	VLDO	—	PWR	—	LDO 输出数字正电源电压
	VDD	—	PWR	—	数字正电源电压
VSS	VSS	—	PWR	—	数字负电源电压
以下引脚仅用于 BS45V5930					
NC	NC	—	—	—	无连接
OCDSDA	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址，仅用于 EV 芯片
OCDSCK	OCDSCK	—	ST	—	OCDS 时钟，仅用于 EV 芯片

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

极限参数

电源供应电压 ..... V_{SS}-0.3V~V_{SS}+13.2V
输入电压 ..... V_{SS}-0.3V~V_{DD}+0.3V
储存温度 ..... -50°C~125°C
工作温度 ..... -40°C~85°C
I_{OH} 总电流 (for PA) ..... -80mA
I_{OH} 总电流 (for HVO) ..... -1300mA
I_{OL} 总电流 80mA
总功耗 670mW

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=-40˚C~85˚C

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 – HIRC	$f_{SYS}=f_{HIRC}$=8MHz	4.75	—	5.25	V
	工作电压 – LIRC	$f_{SYS}=f_{LIRC}$=32kHz	4.75	—	5.25	V

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	最大	单位
		V_{DD}	条件				**85˚C**	
I_{STB}	休眠模式	5V	WDT on	—	330	560	570	μA
	空闲模式 0 – LIRC	5V	f_{SUB} on	—	320	555	560	μA
	空闲模式 1 – HIRC	5V	f_{SUB} on, f_{SYS}=8MHz	—	600	800	960	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。

工作电流特性

Ta=25˚C

符号	工作模式	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{DD}	低速模式 – LIRC	5V	f_{SYS}=32kHz	—	350	600	μA
	快速模式 – HIRC	5V	f_{SYS}=8MHz	—	2	3	mA

注：当使用该表格电气特性数据时，以下几点需注意：

1. 任何数字输入都设置为非浮空的状态。
2. 所有测量都在无负载且所有外围功能关闭的条件下进行。
3. 无直流电流路径。
4. 所有工作电流数值通过一个连续的 NOP 指令循环程序测得。

交流电气特性

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

内部高速振荡器 – HIRC – 频率精确度

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

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{HIRC}	通过烧录器调整后的 8MHz HIRC 频率	5V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	

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. 当应用电压范围是 4.75V~5.25V 时，建议烧录器电压固定在 5V。
3. 此后再通过程序中振荡器控制位将其频率改为其它值时，频率误差范围将增加到 ±20%。

内部低速振荡器电气特性 – LIRC

Ta=25°C, 除非有特别说明

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	5V~5.5V	25°C	-10%	32	+10%	kHz
			-40°C~85°C	-50%	32	+60%	
t_{START}	LIRC 启动时间	—	—	—	—	500	μs

系统上电时间电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
t_{SST}	系统启动时间 (从 f_{SYS} off 的状态下唤醒)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	系统启动时间 (从 f_{SYS} on 的状态下唤醒)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SUB}
	系统速度切换时间 (快速模式 → 低速模式或 低速模式 → 快速模式)	—	f_{HIRC} off → on	—	16	—	t_{HIRC}
t_{RSTD}	系统复位延迟时间 (上电复位)	—	RR_{POR}=5 V/ms	42	48	54	ms
	系统复位延迟时间 (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_{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 口或输入引脚低电平输入电压	5V	—	3.5	—	5	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OH}	I/O 口源电流	5V	$V_{OH}=0.9V_{DD}$	-8	-16	—	mA
I_{OL}	I/O 口灌电流	5V	$V_{OL}=0.1V_{DD}$	32	65	—	mA
R_{PH}	I/O 口上拉电阻	5V	—	10	30	50	kΩ
	I/O 口上拉电阻 (BS45V5930 中的 OCDSCK 和 OCDSDA)	5V	—	10	30	50	
I_{LEAK}	输入漏电流	5V	$V_{IN}=V_{DD}$ 或 $V_{IN}=V_{SS}$	—	—	±1	μA

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输入灌电流，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。

LDO 电气特性

$V_{IN}=V_{OUT}+1V$, $C_{LOAD}=1\mu F$, Ta=25°C，除非有特别说明

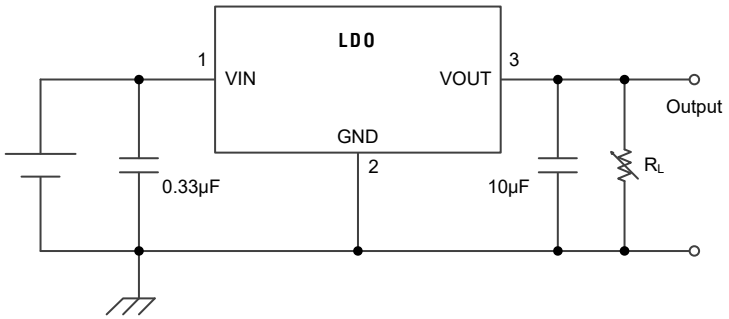
符号	参数	测试条件		最小	典型	最大	单位
		V_{IN}	条件				
V_{IN}	输入电压	—	—	5.1	6	12	V
V_{OUT}	输出电压	—	$I_{LOAD}=1mA$, $V_{OUT}=5.0V$	-3%	5.0	3%	V
		—	Ta=-40°C~85°C, $I_{LOAD}=1mA$, $V_{OUT}=5.0V$	-5%	5.0	5%	V
ΔV_{LOAD}	负载调整率 (1)	—	$1mA \le I_{LOAD} \le 40mA$,	—	0.015	0.033	%/mA
V_{DROP}	压降 (2)	—	$\Delta V_{OUT}=2\%$, $I_{LOAD}=1mA$	—	20	—	mV
		—	$\Delta V_{OUT}=2\%$, $I_{LOAD}=10mA$	—	100	—	mV
		—	$V_{IN}=V_{OUT}+1.5V$, $\Delta V_{OUT}=2\%$, $I_{LOAD}=40mA$	—	400	600	mV
I_{OUT}	输出电流	—	$V_{IN}=V_{OUT}+1V$, $V_{OUT}=5V$ $\Delta V_{OUT}=-3\%$	25	—	—	mA
		—	$V_{IN}=V_{OUT}+2V$, $V_{OUT}=5V$ $\Delta V_{OUT}=-3\%$	40	—	—	mA
I_Q	静态电流	12V	无负载	—	320	—	μA
ΔV_{LINE}	线性调整率	—	$V_{OUT}+1V \le V_{IN} \le 12V$, $I_{LOAD}=1mA$	—	—	0.2	%/V
TC	温度系数	—	Ta=-40°C~85°C, $I_{LOAD}=10mA$	—	±1.5	±2	mV/°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{IN}	条件				
RR	纹波抑制 (3)	—	V_{IN}=10V_{DC}+2$V_{P-P(AC)}$, I_{LOAD} ≤ 40mA, f=120Hz	35	—	—	dB
$t_{LDOSTART}$	LDO 启动时间	6V	I_{LOAD}=1mA, V_{OUT}=5V ± 5%	—	—	10	ms
VCCO	VCCO 电压	—	V_{IN}=5.1V~12V	-5%	0.2×V_{IN}	+5%	V

注：1. 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=($T_{J(MAX)}$–Ta)/θ_{JA}。

2. 在 V_{IN}=V_{OUT}+1V 与一个固定负载条件下使输出电压下降 2%，此时的输入电压减去输出电压就是 Dropout 电压。

3. 纹波抑制比测量电路。RR=20×log(ΔV_{IN}/ΔV_{OUT})。



高压驱动器输出电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{IN}	输入电压	—	—	V_{DD}	—	12	V
I_{OH}	HVO0/HVO1 引脚源电流	—	V_{OH}=0.9×V_{IN}, V_{IN}=6V	-250	-300	—	mA
		—	V_{OH}=0.9×V_{IN}, V_{IN}=12V	-400	-450	—	mA
I_{OL}	HVO0/HVO1 引脚灌电流	—	V_{OL}=0.1×V_{IN}, V_{IN}=6V	40	50	—	mA
			V_{OL}=0.1×V_{IN}, V_{IN}=12V	60	80	—	mA

OVP 电气特性

Ta=25°C

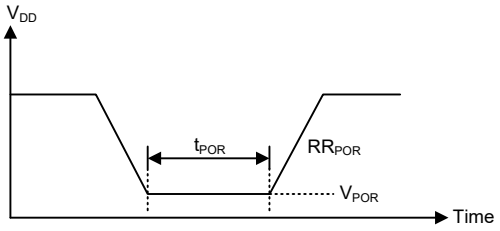
符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	4.75	—	5.25	V
I_{OVP}	工作电流	5V	OVPEN=1, {OVPDAH[3:0], OVPDAL[7:0]} =1000 0000 0000	—	145	180	μA
V_{OS}	输入失调电压	5V	校准后	-2	—	2	mV
V_{HYS}	迟滞	5V	—	20	40	60	mV
V_{CM}	共模电压范围	5V	—	V_{SS}	—	V_{DD}-1.4	V

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
DNL	非线性微分误差	5V	DAC V_{REF}=V_{DD}	—	—	±3	LSB
INL	非线性积分误差	5V	DAC V_{REF}=V_{DD}	—	—	±4	LSB

上电复位特性

Ta=25°C

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

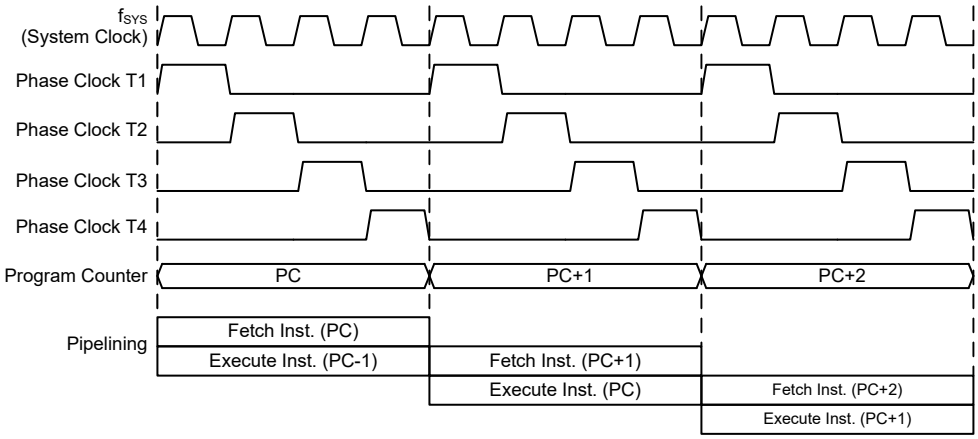


系统结构

内部系统结构是 Holtek 单片机具有良好性能的主要因素。由于采用 RISC 结构该单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的取得和执行同时进行，此举使得除了跳转和调用指令外，其它指令都能在一个指令周期内完成。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、移位元、递增、递减和分支等功能，而内部的数据路径则是以通过累加器和 ALU 的方式加以简化。有些寄存器在数据存储器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供具有最大可靠度和灵活性的 I/O 控制系统时，仅需要少数的外部器件。使得该单片机适用于低成本和批量生产的控制应用。

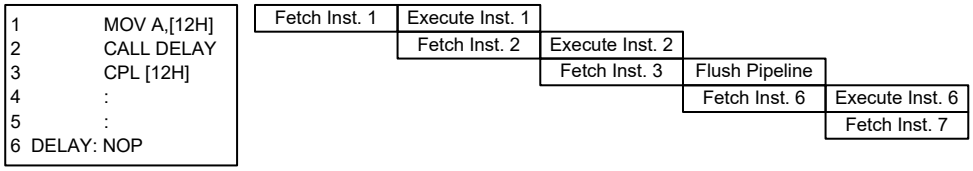
时序和流水线结构

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



系统时序和流水线

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



指令捕捉

程序计数器

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

当执行的指令要求跳转到不连续的地址时，如跳转指令、子程序调用、中断或复位等，单片机通过加载所需要的位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
PC10~PC8	PCL7~PCL0

程序计数器

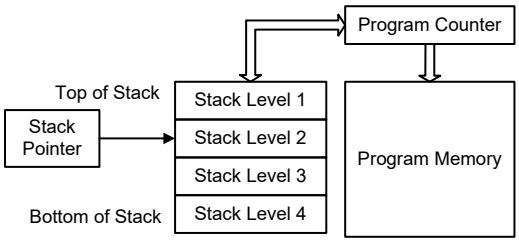
程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。PCL 寄存器的使用可能引起程序跳转，因此需要额外的指令周期。

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

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

若堆栈溢出，则首个存入堆栈的程序计数器数据将会丢失。



算术逻辑单元 – ALU

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

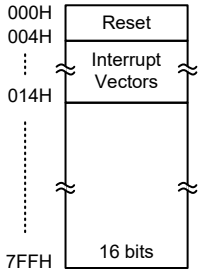
- 算术运算：ADD，ADDM，ADC，ADCM，SUB，SUBM，SBC，SBCM，DAA
- 逻辑运算：AND，OR，XOR，ANDM，ORM，XORM，CPL，CPLA
- 移位运算：RRA，RR，RRCA，RRC，RLA，RL，RLCA，RLC
- 递增和递减：INCA，INC，DECA，DEC
- 分支判断：JMP，SZ，SZA，SNZ，SIZ，SDZ，SIZA，SDZA，CALL，RET，RETI

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 Flash 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 2K×16 位，程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口。数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。



程序存储器结构

特殊向量

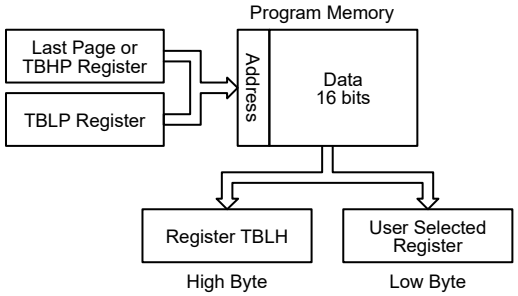
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

查表

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

在设定完表格指针后，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。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“0700H”指向的地址是 2K 程序存储器中最后一页的起始地址。表格指针低字节寄存器的初始值设为“06H”，这可保证从数据表格读取的第一笔数据位于程序存储器地址“0706H”，即最后一页起始地址后的第六个地址。值得注意的是，假如“TABRD [m]”指令被使用，则表格指针指向页的第一个地址。在这个例子中，表格数据的高字节等于零，而当“TABRDL [m]”指令被执行时，此值将会自动的被传送到 TBLH 寄存器。

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

表格读取程序范例

```
tempreg1 db ?       ; temporary register #1
tempreg2 db ?       ; temporary register #2
:
:
mov a,06h           ; initialise low table pointer - note that this address
                    ; is referenced
mov tblp,a          ; to the last page or the page that tbhp pointed
mov a,07h           ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1      ; transfers value in table referenced by table pointer
                    ; data at program memory address "0706H" 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 "0705H" transferred to
                    ; tempreg2 and TBLH
                    ; in this example the data "1AH" is transferred to
                    ; tempreg1 and data "0FH" to register tempreg2
:
:
org 0700h           ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

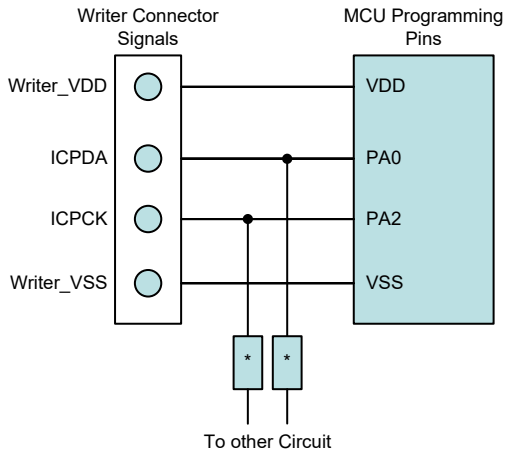
在线烧录 – ICP

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写，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 烧录器引脚名称	MCU 在线烧录引脚名称	引脚说明
ICPDA	PA0	串行数据 / 地址烧录
ICPCK	PA2	时钟烧录
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，一条用于串行时钟，剩余两条用于提供电源和地。芯片在线烧写的详细使用说明超出此文档的描述范围，将由专门的参考文献提供。

烧录过程中，用户必须确保 ICPDA 和 ICPCK 这两个引脚没有连接至其它输出脚。



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

片上调试 – OCDS

EV 芯片 BS45V5930 用于 BS45F5930 单片机仿真。此 EV 芯片提供片上调试功能 (OCDS – On-Chip Debug) 用于开发过程中的单片机调试。除了片上调试功能方面和封装类型，EV 芯片和实际单片机在功能上几乎是兼容的。用户可将 OCDSDA 和 OCDSCK 引脚连接至 Holtek HT-IDE 开发工具，从而实现 EV 芯片对实际单片机的仿真。OCDSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户用 EV 芯片进行调试时，实际单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。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。

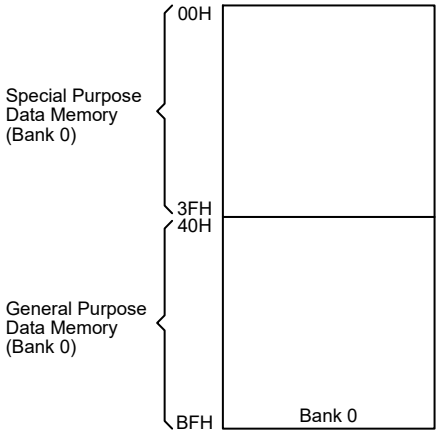
结构

数据存储器分为两个部分，第一部分是 RAM 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 RAM 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。

数据存储器只有一个 Bank，是 8 位存储器。数据存储器 Bank 分为两类，特殊功能数据存储器和通用数据存储器。特殊功能数据存储器地址范围为 00H~3FH，而通用数据存储器地址范围为 40H~BFH。

特殊功能数据存储器		通用数据存储器	
所在 **Bank**	**Bank**：地址	容量	**Bank**：地址
0	Bank 0：00H~3FH	128×8	0：40H~BFH

数据存储器概要



数据存储器结构

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。使用者可对这个数据存储区进行读取和写入的操作。使用位操作指令可对个别的位做置位或复位的操作，极大地方便了用户在数据存储器内进行位操作。

特殊功能数据存储器

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

	Bank 0
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	SCC
0CH	HIRCC
0DH	INTEG
0EH	INTC0
0FH	RSTFC
10H	
11H	INTC1
12H	
13H	
14H	PA
15H	PAC
16H	PAPU
17H	PAWU
18H	
19H	
1AH	WDTC
1BH	TBC
1CH	PSCR
1DH	PAS0
1EH	PAS1
1FH	HVOC

	Bank 0
20H	HVOC1
21H	DIVOC
22H	
23H	
24H	
25H	
26H	
27H	
28H	
29H	
2AH	
2BH	PWMC
2CH	PWM0DATA
2DH	
2EH	PWM1DATA
2FH	TMRC
30H	TMR
31H	OVPC0
32H	OVPC1
33H	OVPDAL
34H	OVPDAH
35H	TKTMR
36H	TKC0
37H	TK16DL
38H	TK16DH
39H	TKC1
3AH	TKM16DL
3BH	TKM16DH
3CH	TKMROL
3DH	TKMROH
3EH	TKMC0
3FH	TKMC1

: Unused, read as “00”

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

间接寻址寄存器 IAR0 和 IAR1 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寄存器和存储器指针做数据操作，以取代定义实际存储器地址的直接存储器寻址方法。在间接寻址寄存器 IAR0 和 IAR1 上的任何动作，将对相应的存储器指针 MP0 和 MP1 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 Bank 0，而 IAR1 和 MP1 可以访问任何 Bank。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，MP1

该单片机提供两个存储器指针，即 MP0 和 MP1。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 和 IAR0 用于访问 Bank 0，而 MP1 和 IAR1 可访问任何 Bank。直接寻址仅可以用在 Bank 0 中，其它所有 Bank 只能使用 MP1 和 IAR1 进行间接寻址。

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

间接寻址程序范例

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

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。

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。

表格寄存器 – TBLP，TBHP，TBLH

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

状态寄存器 – STATUS

该 8-bit 的状态寄存器由零标志位 (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 TO 和 PDF 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 和 C 标志位通常反映最近运算的状态。

- C：当加法运算的结果产生进位，或减法运算的结果没有产生借位时，则 C 被置位，否则 C 被清零，同时 C 也会被带进位的移位指令所影响。
- AC：当低半字节加法运算的结果产生进位，或低半字节减法运算的结果没有产生借位时，AC 被置位，否则 AC 被清零。
- Z：当算术或逻辑运算结果是零时，Z 被置位，否则 Z 被清零。
- OV：当运算结果高两位的进位状态异或结果为 1 时，OV 被置位，否则 OV 被清零。
- PDF：系统上电或执行“CLR WDT”指令会清零 PDF，而执行“HALT”指令则会置位 PDF。
- TO：系统上电或执行“CLR WDT”或“HALT”指令会清零 TO，而当 WDT 溢出则会置位 TO。

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	—	—	TO	PDF	OV	Z	AC	C
R/W	—	—	R	R	R/W	R/W	R/W	R/W
POR	—	—	0	0	x	x	x	x

"x"：未知

Bit 7~6　未定义，读为"0"

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

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

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

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

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

Bit 0　**C**：进位标志位
0：无进位
1：如果在加法运算中结果产生了进位，或在减法运算中结果不发生借位

标志位 C 也受循环移位指令的影响。

振荡器

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

振荡器概述

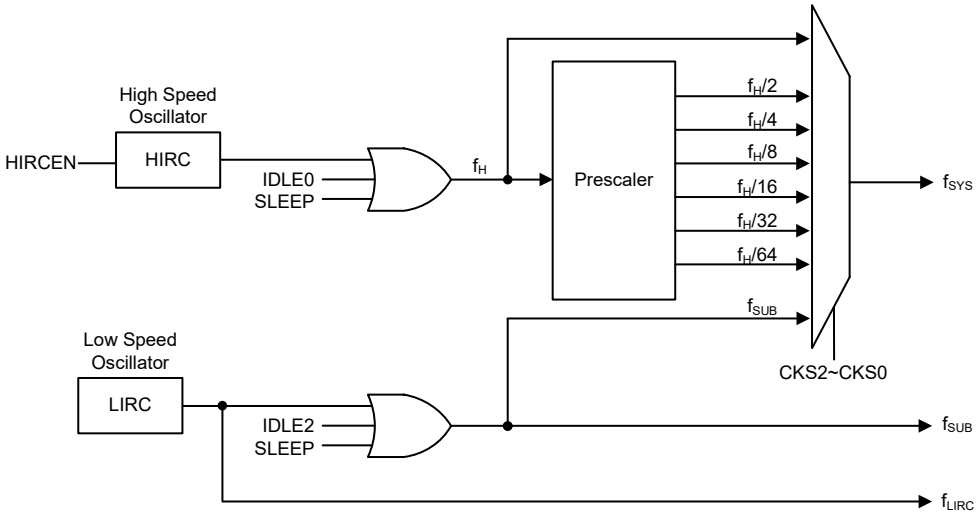
振荡器除了作为系统时钟源，还作为看门狗定时器和时基功能的时钟源。两个完全集成的内部振荡器不需要任何外围器件。它们提供的高速和低速系统振荡器具有较宽的频率范围。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

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

振荡器类型

系统时钟配置

该单片机有两个系统振荡器，一个高速振荡器和一个低速振荡器。高速振荡器为内部 8MHz RC 振荡器 HIRC。低速振荡器为内部 32kHz RC 振荡器 LIRC。使用高速或低速振荡器作为系统时钟的选择是通过设置 SCC 寄存器中的 CKS2~CKS0 位决定的，系统时钟可动态选择。



系统时钟配置

内部高速 RC 振荡器 – HIRC

内部高速 RC 振荡器是一个集成的系统振荡器，不需其它外部器件。该内部 RC 振荡器具有固定频率 8MHz。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因电源电压、温度以及芯片制成工艺不同的影响减至最低程度。

内部 32kHz 振荡器 – LIRC

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

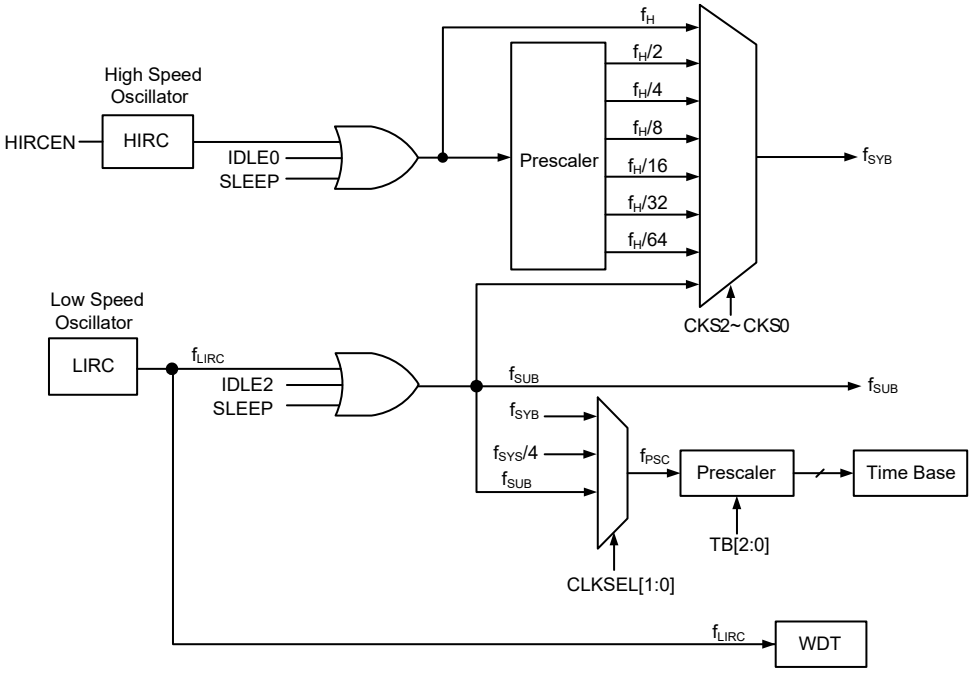
工作模式和系统时钟

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

系统时钟

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

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



单片机时钟配置

注：当系统时钟源 f_{SYS} 由 f_H 到 f_{SUB} 转换时，可以通过设置相应的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	相关的寄存器值			f_{SYS}	f_H	f_{SUB}	f_{LIRC}
		FHIDEN	FSIDEN	CKS[2:0]				
快速模式	On	x	x	000~110	On	On	On	On
低速模式	On	x	x	111	On	On/Off(1)	On	On
空闲模式 0	Off	0	1	000~110	Off	Off	On	On
				111	On			
空闲模式 1	Off	1	1	xxx	On	On	On	On
空闲模式 2	Off	1	0	000~110	On	On	Off	On
				111	Off			
休眠模式	Off	0	0	xxx	Off	Off	Off	On(2)

“x”表示无关

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} 将保持开启。

快速模式

顾名思义，这是主要的工作模式之一，单片机的所有功能均可在此模式中实现且系统时钟由一个高速振荡器提供。该模式下单片机正常工作的时钟源来自 HIRC 振荡器。高速振荡器频率可被分为 1~64 的不等比率，实际的比率由 SCC 寄存器中的 CKS2~CKS0 位选择的。单片机使用高速振荡器分频作为系统时钟可减少工作电流。

低速模式

此模式的系统时钟虽为较低速时钟源，但单片机仍能正常工作。该低速时钟源可来自 f_{SUB}，而 f_{SUB} 来自于 LIRC 振荡器。

休眠模式

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为低时，系统进入休眠模式。在休眠模式中，CPU 停止运行，f_{SUB} 停止为外围功能提供时钟。然而因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相应的振荡器配置。

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

系统工作模式控制寄存器列表

- **SCC 寄存器**

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

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

Bit 4~2 未定义，读为“0”

Bit 1 **FHIDEN**：CPU 关闭时高频振荡器控制位
0：除能
1：使能
此位用来控制在执行 HALT 指令 CPU 关闭后高速振荡器是否停止。

Bit 0 **FSIDEN**：CPU 关闭时低频振荡器控制位
0：除能
1：使能
此位用来控制在执行 HALT 指令 CPU 关闭后低速振荡器是否停止。因 WDT 功能始终使能，即使该位被清零，LIRC 振荡器也会继续运行。

- **HIRCC 寄存器**

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

Bit 7~2 未定义，读为“0”

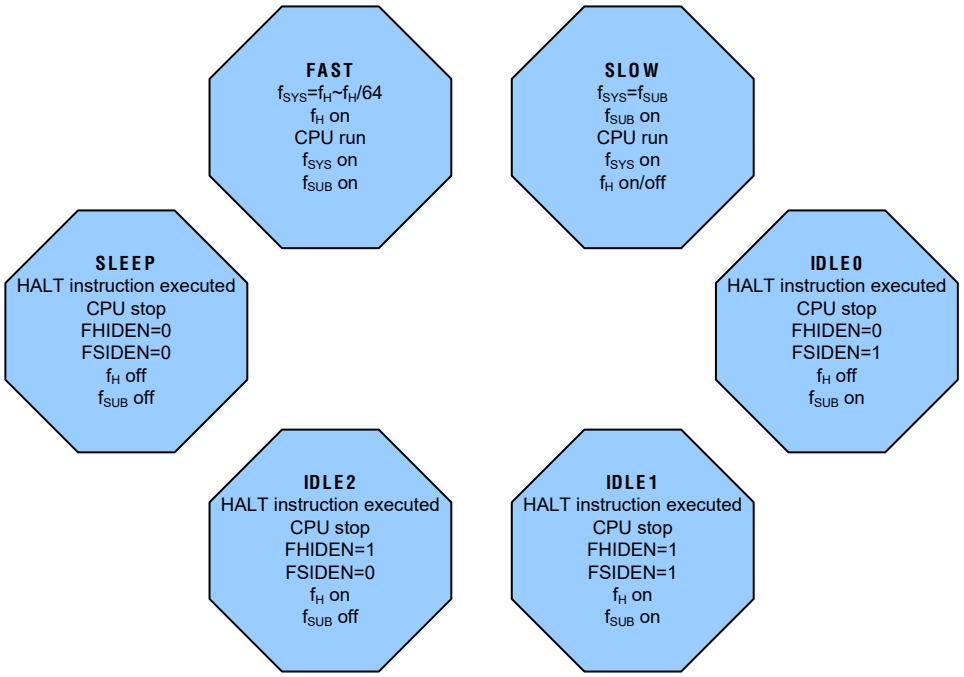
Bit 1 **HIRCF**：HIRC 振荡器稳定标志位
0：HIRC 不稳定
1：HIRC 稳定
此位用于表明 HIRC 振荡器是否稳定。HIRCEN 位置高使能 HIRC 振荡器，或者通过应用程序改变 HIRC 频率选择位时，HIRCF 位会先被清零，在 HIRC 稳定后会被置高。

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

工作模式切换

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

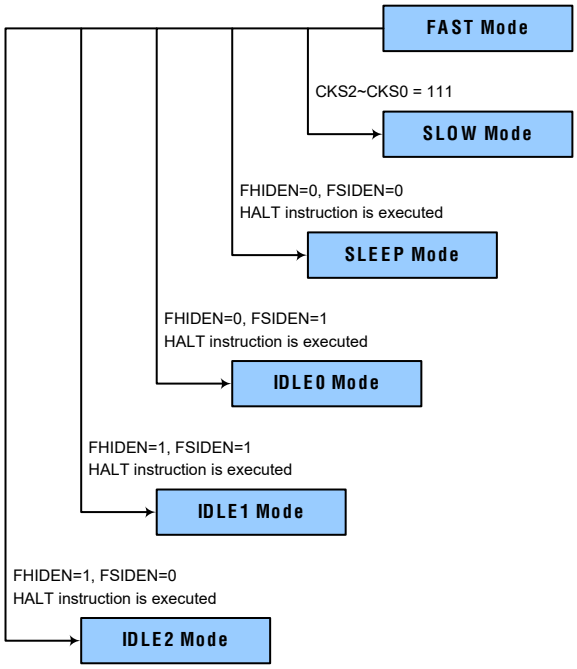
简单来说，快速模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



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。

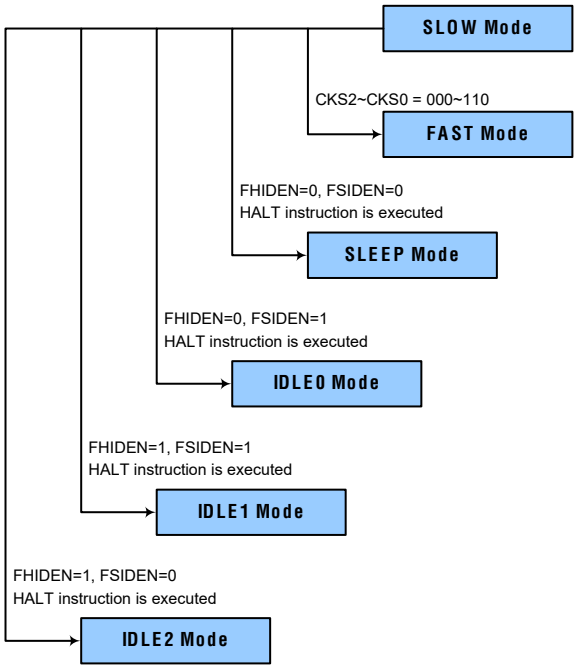
低速模式的时钟源来自 LIRC 振荡器，因此要求这些振荡器在所有模式切换动作发生前稳定下来。



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指定在交流电气特性中。



进入休眠模式

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

- 系统时钟停止运行，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 因 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——应用程序中执行“HALT”指令前需设置 SCC 寄存器中的 FHIDEN 位为“0”且 FSIDEN 位为“1”。在上述条件下执行该指令后，将发生的情况如下：

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

- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 因 WDT 功能始终使能，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——应用程序中执行“HALT”指令前需设置 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为“1”。在上述条件下执行该指令后，将发生的情况如下：

- f_H 和 f_{SUB} 时钟开启，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 因 WDT 功能始终使能，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——应用程序中执行“HALT”指令前需设置 SCC 寄存器中的 FHIDEN 位为“1”且 FSIDEN 位为“0”。在上述条件下执行该指令后，将发生的情况如下：

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

待机电流注意事项

由于单片机进入休眠或空闲模式的主要原因是将 MCU 的电流降低到尽可能低，可能到只有几个微安的级别 (空闲模式 1 和空闲模式 2 除外)，所以如果要将电路的电流降到最低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入 / 输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这些引脚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

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

系统进入休眠或空闲模式之后，可以通过以下几种方式唤醒：

- PA 口下降沿
- 系统中断
- WDT 溢出

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

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

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源 f_{LIRC} 由内部低速振荡器 LIRC 提供。内部振荡器 LIRC 的频率大约为 32kHz，这个特殊的内部时钟周期随 V_{DD}、温度和制成的不同而变化。看门狗定时器的时钟源可分频为 2^8~2^{18} 以提供更大的溢出周期，分频比由 WDTC 寄存器中的 WS2~WS0 位来决定。

看门狗定时器控制寄存器

WDTC 寄存器用于控制看门狗定时器功能的使能、复位单片机及选择溢出周期。

- **WDTC 寄存器**

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

Bit 7~3 **WE4~WE0**：WDT 软件控制
01010B/10101B：使能
其它值：MCU 复位
如果由于不利的环境因素使这些位变为其它值，单片机将复位。复位动作发生在 t_{SRESET} 延迟时间后，且 RSTFC 寄存器的 WRF 位将置为“1”。

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

- **RSTFC 寄存器**

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

Bit 7~1　未定义，读为“0”

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

看门狗定时器操作

当 WDT 溢出时，它产生一个芯片复位的动作。这也就意味着正常工作期间，用户需在应用程序中看门狗溢出前有策略地清看门狗定时器以防止其产生复位，可使用清除看门狗指令实现。无论什么原因，程序失常跳转到一个未知的地址或进入一个死循环，这些清除指令都不能被正确执行，此种情况下，看门狗将溢出以使单片机复位。看门狗定时器控制寄存器 WDTC 中有五位 WE4~WE0 可提供使能控制以及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”。

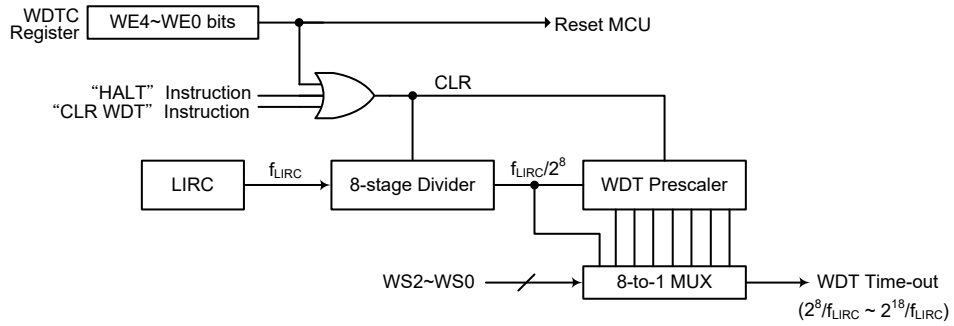
WE4~WE0 位	WDT 功能
10101B/01010B	使能
其它值	复位单片机

看门狗定时器功能控制

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

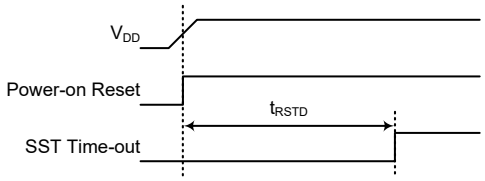
有一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。

复位功能

单片机的三种内部复位方式将在此处做具体介绍。

上电复位

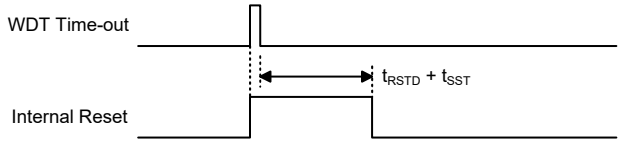
这是最基本且不可避免的复位，发生在单片机上电后。除了保证程序存储器从开始地址执行，上电复位也使得其它寄存器被设定在预设条件。所有的输入 / 输出端口控制寄存器在上电复位时会保持高电平，以确保上电后所有引脚被设定为输入状态。



上电复位时序图

正常运行时看门狗溢出复位

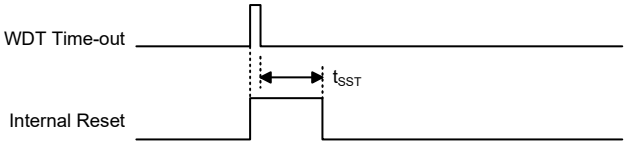
除了看门狗溢出标志位 TO 将被设为“1”之外，在快速模式或低速模式时看门狗溢出复位和上电复位相同。



正常运行时看门狗溢出时序图

休眠或空闲时看门狗溢出复位

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

注：“u”表示不改变

在单片机上电复位之后，各功能单元初始化的情况，列于下表。

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

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

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
IAR0	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP0	xxxx xxxx	xxxx xxxx	uuuu uuuu
IAR1	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	---- -xxx	---- -uuu	---- -uuu
STATUS	--00 xxxx	--1u uuuu	--11 uuuu
SCC	000- --00	000- --00	uuu- --uu
HIRCC	---- --01	---- --01	---- --uu
INTEG	---- --00	---- --00	---- --uu
INTC0	-000 0000	-000 0000	-uuu uuuu
RSTFC	---- ---0	---- ---u	---- ---u
INTC1	--00 --00	--00 --00	--uu --uu
PA	---1 1111	---1 1111	---u uuuu
PAC	---1 1111	---1 1111	---u uuuu
PAPU	---0 0000	---0 0000	---u uuuu
PAWU	---0 0000	---0 0000	---u uuuu

寄存器	上电复位	WDT 溢出 (正常运行)	WDT 溢出 (空闲 / 休眠)
WDTC	0101 0011	0101 0011	uuuu uuuu
TBC	0--- -000	0--- -000	u--- -uuu
PSCR	---- --00	---- --00	---- --uu
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	---- --00	---- --00	---- --uu
HVOC	---- 0000	---- 0000	---- uuuu
HVOC1	0000 --00	0000 --00	uuuu --uu
DIVOC	---- ---0	---- ---0	---- ---u
PWMC	000- --00	000- --00	uuu- --uu
PWM0DATA	0000 0000	0000 0000	uuuu uuuu
PWM1DATA	0000 0000	0000 0000	uuuu uuuu
TMRC	--00 -000	--00 -000	--uu -uuu
TMR	0000 0000	0000 0000	uuuu uuuu
OVPC0	0000 -000	0000 -000	uuuu -uuu
OVPC1	0001 0000	0001 0000	uuuu uuuu
OVPDAL	0000 0000	0000 0000	uuuu uuuu
OVPDAH	---- 0000	---- 0000	---- uuuu
TKTMR	0000 0000	0000 0000	uuuu uuuu
TKC0	-000 0000	-000 0000	-uuu uuuu
TK16DL	0000 0000	0000 0000	uuuu uuuu
TK16DH	0000 0000	0000 0000	uuuu uuuu
TKC1	---- --11	---- --11	---- --uu
TKM16DL	0000 0000	0000 0000	uuuu uuuu
TKM16DH	0000 0000	0000 0000	uuuu uuuu
TKMROL	0000 0000	0000 0000	uuuu uuuu
TKMROH	---- --00	---- --00	---- --uu
TKMC0	0000 0000	0000 0000	uuuu uuuu
TKMC1	0-00 0000	0-00 0000	u-uu 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。每个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。

此单片机提供 PA 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	—	—	—	PA4	PA3	PA2	PA1	PA0
PAC	—	—	—	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	—	—	—	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	—	—	—	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0

“—”：未定义，读为“0”

输入 / 输出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，通常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 来设置，它用一个 PMOS 晶体管来实现上拉电阻功能。

需要注意的是，当 I/O 引脚设为输入或 NMOS 输出时，上拉功能才会受 PAPU 控制开启，其它状态下上拉功能不可用。

- **PAPU 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
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　**PAPU4~PAPU0**：PA 口 PA4~PA0 引脚上拉电阻控制位
0：除能
1：使能

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。此功能特别适合于通过外部开关来唤醒的应用。PA 口的每个引脚可以通过设置 PAWU 寄存器来单独选择是否具有唤醒功能。

需要注意的是，只有当引脚功能为通用 I/O 功能且单片机处于休眠或空闲模式时，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	—	—	—	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
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　**PAWU4~PAWU0**：PA4~PA0 引脚唤醒功能控制位
0：除能
1：使能

输入 / 输出端口控制

每一个输入 / 输出口都具有各自的控制寄存器，即 PAC，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。应注意，如果对输出口做读取动作时，程序读取到的是内部输出数据锁存器中的状态，而不是输出引脚上实际的逻辑状态。

- **PAC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	PAC4	PAC3	PAC2	PAC1	PAC0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	1	1	1	1	1

Bit 7~5　未定义，读为“0”

Bit 4~0　**PAC4~PAC0**：I/O 口 PA4~PA0 输入 / 输出控制
0：输出
1：输入

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应用程序控制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。单片机包含端口“A”引脚共用功能选择寄存器，记为 PASn。这些寄存器可以用来选择共用引脚的特定功能。

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	—	—	—	—	—	—	PAS11	PAS10

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
01：PA3
10：OVPI
11：KEY4

Bit 5~4 **PAS05~PAS04**：PA2 引脚共用功能选择
00：PA2
01：PA2
10：PA2
11：KEY2

Bit 3~2 **PAS03~PAS02**：PA1 引脚共用功能选择
00：PA1
01：PA1
10：OVPI
11：KEY3

Bit 1~0 **PAS01~PAS00**：PA0 引脚共用功能选择
00：PA0
01：PA0
10：PA0
11：KEY1

- **PAS1 寄存器**

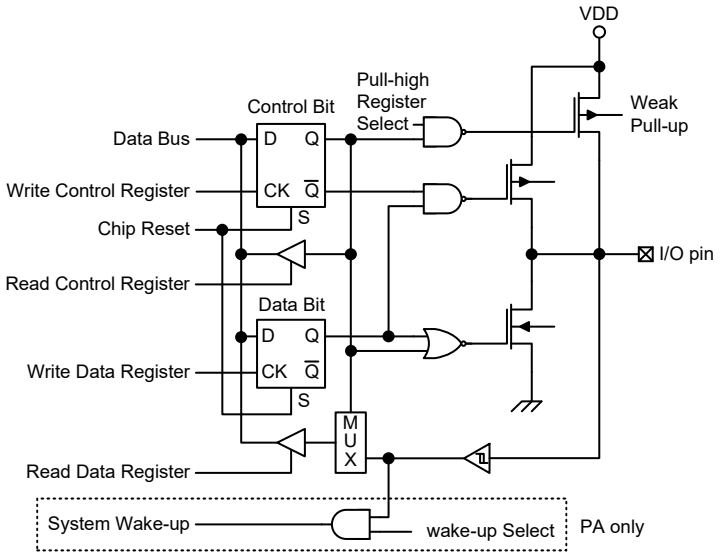
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PAS11	PAS10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

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

输入 / 输出引脚结构

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



逻辑功能输入 / 输出结构

编程注意事项

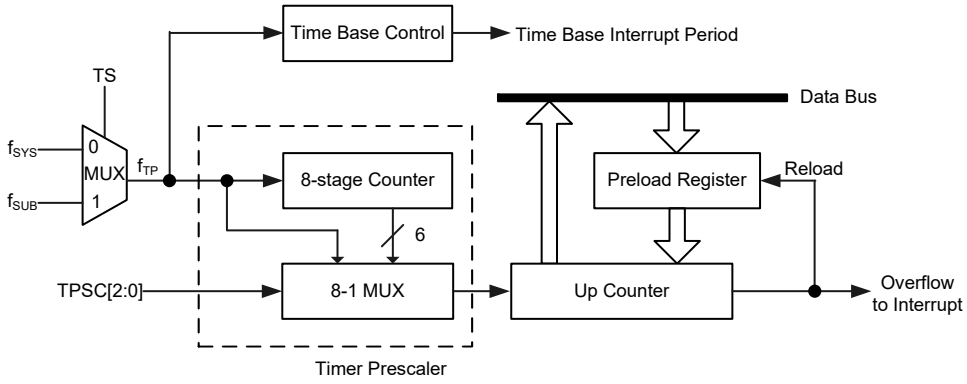
在编程中，最先要考虑的是端口的初始化。复位之后，所有的输入 / 输出数据及端口控制寄存器都将被设为逻辑高。所有输入 / 输出引脚默认为输入状态，而其电平则取决于其它相连接电路以及是否选择了上拉电阻。如果端口控制寄存器 PAC 将某些引脚设定为输出状态，这些输出引脚会有初始高电平输出，除非端口数据寄存器 PA 在程序中被预先设定。设置哪些引脚是输入及哪些引脚是输出，可通过设置正确的值到对应的端口控制寄存器，或使用指令“SET [m].i”及“CLR [m].i”来设定端口控制寄存器中个别的位。注意，当使用这些位控制指令时，系统即将产生一个读 - 修改 - 写的操作。单片机需要先读入整个端口上的数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

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

定时 / 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。该单片机包含一个 8 位的定时 / 事件计数器。并且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 / 事件计数器的定时设置以及决定定时器如何运作。



定时 / 事件计数器结构

配置定时 / 事件计数器输入时钟源

定时 / 事件计数器的时钟源可以来自系统时钟 f_{SYS} 或 f_{SUB} 振荡器，由寄存器 TMRC 的 TS 位选择使用哪种时钟源。内部时钟首先由分频器分频，分频比由定时器控制寄存器的 TPSC2~TPSC0 位来确定。

定时 / 计数寄存器 – TMR

定时 / 计数寄存器 TMR，是位于特殊数据存储器内的特殊功能寄存器，用于储存定时器的当前值。当收到一个内部计数脉冲，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，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。注意，如果定时 / 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 / 事件计数器已经使能且正在计数，在这个周期内写入到预置数据寄存器的任何新数据将保留在预置寄存器，直到溢出发生时才被写入实际定时器。

定时 / 计数控制寄存器 – TMRC

定时 / 计数控制寄存器为 TMRC，配合相应的 TMR 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 / 计数控制寄存器，以便保证定时器能正确操作，而这个过程通常在程序初始化期间完成。

定时 / 计数控制寄存器的第 4 位即 TON，用于定时器开关控制，设定为逻辑高时，计数器开始计数，而清零时则停止计数。定时 / 计数控制寄存器的第 0~2 位用来控制输入时钟预分频器。TS 位用来选择内部时钟源。

- **TMRC 寄存器**

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

Bit 7~6　未定义，读为“0”

Bit 5　**TS**：定时 / 事件计数器时钟源选择
0：f_{SYS}
1：f_{SUB}

Bit 4　**TON**：定时 / 事件计数器计数使能位
0：除能
1：使能

Bit 3　未定义，读为“0”

Bit 2~0　**TPSC2~TPSC0**：定时器预分频比选择位
定时器内部时钟 =
000：f_{TP}
001：$f_{TP}/2$
010：$f_{TP}/4$
011：$f_{TP}/8$
100：$f_{TP}/16$
101：$f_{TP}/32$
110：$f_{TP}/64$
111：$f_{TP}/128$

定时 / 事件计数器操作

定时 / 事件计数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。f_{SYS} 或 f_{SUB} 振荡器被用来当定时器的输入时钟源。然而，该定时器时钟源被预分频器进一步分频，分频比是由定时器控制寄存器的 TPSC0~TPSC2 位来确定。定时器使能位，即 TON 位需要设为逻辑高，才能令定时器工作。每次内部时钟由高到低的电平转换，定时器会重新载入预置寄存器的值，然后继续计数。定时器溢出以及相应的内部中断产生也是唤醒暂停模式的一种方法，然而，通过设置中断寄存器 INTC0 中的定时器中断使能位 TE 为 0，可以禁止计数器中断。

预分频器

寄存器 TMRC 的 TPSC0~TPSC2 位用来确定定时 / 事件计数器的内部时钟的分频比，从而能够设置更长的定时器溢出周期。

编程注意事项

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定时器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。定时器控制寄存器中的边沿选择，定时器模式，时钟源控制位需要正确地设置以确保定时器能正确配置为所需的应用。在定时 / 事件计数器打开之前，需要确保先载入定时 / 计数寄存器的初始值，这是因为在上电后，定时 / 计数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 / 事件计数器产生溢出，中断控制寄存器中相应的中断请求标志将置位。若定时 / 事件计数器中断允许，将会依次产生一个中断信号。不管中断是否使能，在省电状态下，定时 / 事件计数器的溢出也会产生唤醒。若定时 / 事件计数器处于事件计数模式，并且外部信号继续改变状态，则可能会发生以下这种情况，在这种情况下，定时 / 事件计数器将继续对这些外部事件计数，若发生溢出，单片机将从省电状态中唤醒。为了防止这种唤醒，可以在执行“HALT”指令进入空闲 / 休眠模式之前将相应中断请求标志位置位。

触控按键功能

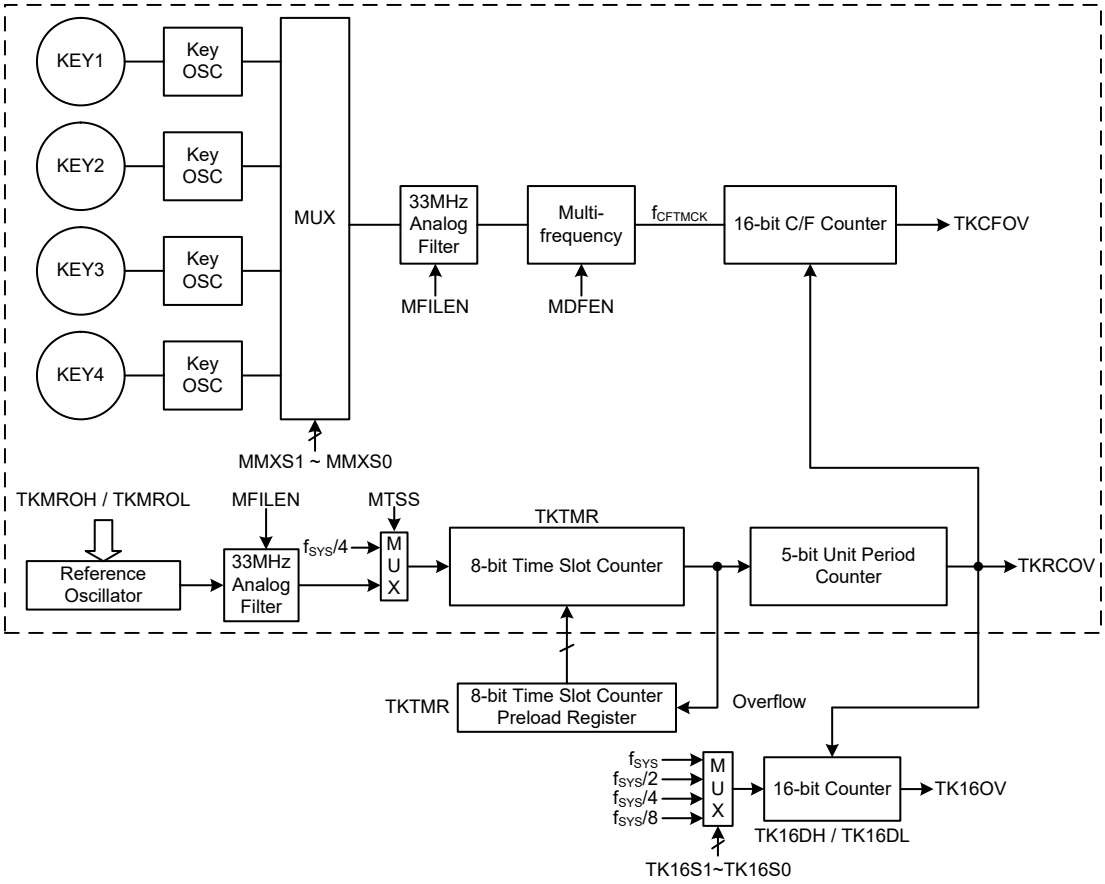
该单片机提供了 4 个触控按键功能。该按键功能内建于单片机内，不需外接元件，通过内部寄存器对其进行简单的操作。

触控按键结构

触控按键与 I/O 引脚共用。通过引脚共用选择寄存器的位来选择所需功能。按键被分成一个组，称为一个模块。每个模块为独立的一组，包含四个触控按键且每个按键有各自的振荡器。每个模块具有单独的控制逻辑电路和配套的寄存器系列。

按键总数	触控按键	共用 I/O 引脚
4	KEY1~KEY4	PA0, PA2, PA1, PA3

触控按键结构



注：1. 虚线中的结构适用于具有四个触控按键的触控按键模块。
2. 当 MTSS=0 和 MROEN=1，或当 MTSS=1，触控按键功能 16-bit 计数器可以正常地工作。

触控按键功能方框图

触控按键寄存器描述

触控按键模块包含 4 个触控按键功能，且都有其配套的寄存器。以下表格显示了触控按键模块的寄存器系列。

寄存器名称	说明
TKTMR	触控按键时隙 8-bit 计数器预载寄存器
TKC0	触控按键功能控制寄存器 0
TKC1	触控按键功能控制寄存器 1
TK16DL	触控按键功能 16-bit 计数器低字节
TK16DH	触控按键功能 16-bit 计数器高字节
TKM16DL	触控按键模块 16-bit C/F 计数器低字节
TKM16DH	触控按键模块 16-bit C/F 计数器高字节
TKMROL	触控按键模块参考振荡器电容选择低字节
TKMROH	触控按键模块参考振荡器电容选择高字节
TKMC0	触控按键模块控制寄存器 0
TKMC1	触控按键模块控制寄存器 1

触控按键模块寄存器定义

寄存器名称	位							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	—	TKRCOV	TKST	TKCFOV	TK16OV	—	TK16S1	TK16S0
TKC1	—	—	—	—	—	—	TKFS1	TKFS0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKM16DL	D7	D6	D5	D4	D3	D2	D1	D0
TKM16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKMROL	D7	D6	D5	D4	D3	D2	D1	D0
TKMROH	—	—	—	—	—	—	D9	D8
TKMC0	MMXS1	MMXS0	MDFEN	MFILEN	MSOFC	MSOF2	MSOF1	MSOF0
TKMC1	MTSS	—	MROEN	MKOEN	MK4EN	MK3EN	MK2EN	MK1EN

触控按键功能寄存器列表

- **TKTMR 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**：触控按键时隙 8-bit 计数器预载寄存器

触控按键时隙计数器预载寄存器用于确定触控按键时隙溢出时间。时隙单元周期为 32 个时隙时钟周期，通过一个 5-bit 计数器获得。因此，时隙计数器溢出时间可由下面的等式算出。

时隙计数器溢出时间 = (256-TKTMR[7:0])×32t_{TSC}

此处的 t_{TSC} 为时隙计数器时钟周期

- **TKC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	TKRCOV	TKST	TKCFOV	TK16OV	—	TK16S1	TK16S0
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 **TKRCOV**：触控按键时隙计数器溢出标志位

0：无溢出

1：溢出

该位可由应用程序访问。当通过触控按键时隙计数器溢出置位该位时，相应的触控按键中断请求标志位将被置位。然而，若该位由应用程序置位，则触控按键中断请求标志位不会受到影响。因此，该位不能由应用程序置位，但必须通过应用程序清零。

若时隙计数器溢出，则 TKRCOV 位和触控按键中断请求标志位 TKMF 将会被置位且所有模块按键振荡器和参考振荡器将自动停止。此时触控按键模块 16-bit C/F 计数器、触控按键功能 16-bit 计数器、5-bit 时隙单元周期计数器和 8-bit 时隙计数器都会自动关闭。

Bit 5 **TKST**：触控按键检测开启控制位
0：停止或无操作
0 → 1：开始检测
当该位为“0”时，触控按键模块的 16-bit C/F 计数器、触控按键功能 16-bit 计数器和 5-bit 时隙计数器会自动清零。但 8-bit 可编程时隙计数器不清零。当该位由 0 到 1 转变时，触控按键模块的 16-bit C/F 计数器、触控按键功能 16-bit 计数器、5-bit 时隙计数器和 8-bit 时隙计数器都会自动开启，并使能按键振荡器和参考振荡器以驱动相应的计数器。

Bit 4 **TKCFOV**：触控按键模块 16-bit C/F 计数器溢出标志位
0：无溢出
1：溢出
当触控按键模块 16-bit C/F 计数器溢出时，该位将被置高，但该位必须通过应用程序清零。

Bit 3 **TK16OV**：触控按键功能 16-bit 计数器溢出标志位
0：无溢出
1：溢出
当触控按键功能 16-bit 计数器溢出时，该位将被置高，但该位必须通过应用程序清零。

Bit 2 未定义，读为“0”

Bit 1~0 **TK16S1~TK16S0**：触控按键功能 16-bit 计数器时钟源选择
00：f_{SYS}
01：$f_{SYS}/2$
10：$f_{SYS}/4$
11：$f_{SYS}/8$

- **TKC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	TKFS1	TKFS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	1	1

Bit 7~2 未定义，读为“0”

Bit 1~0 **TKFS1~TKFS0**：触控按键振荡器和参考振荡器频率选择位
00：1MHz
01：3MHz
10：7MHz
11：11MHz

- **TK16DL 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**：触控按键功能 16-bit 计数器低字节内容

- **TK16DH 寄存器**

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

Bit 7~0 **D15~D8**：触控按键功能 16-bit 计数器高字节内容
该寄存器对用于存储触控按键功能 16-bit 计数器值。该 16-bit 计数器可用于校准参考振荡器或按键振荡器频率。当触控按键时隙计数器溢出，此 16-bit 计数器将停止，计数器内容保持不变。当 TKST 位置低，该寄存器对将被清零。

- **TKM16DL 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**：触控按键模块 16-bit C/F 计数器低字节内容

- **TKM16DH 寄存器**

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

Bit 7~0 **D15~D8**：触控按键模块 16-bit C/F 计数器高字节内容
该寄存器对用于存储触控按键模块 16-bit C/F 计数器值。当触控按键时隙计数器溢出，该 16-bit C/F 计数器将被停止，其内容保持不变。当 TKST 位置低，该寄存器对将被清零。

- **TKMROL 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**：触控按键模块参考振荡器内部电容选择

- **TKMROH 寄存器**

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

Bit 7~2 未定义，读为“0”
Bit 1~0 **D9~D8**：触控按键模块参考振荡器内部电容选择
该寄存器对用于存储触控按键模块参考振荡器电容值。
参考振荡器内部电容值 = (TKMRO[9:0]×50pF)/1024

- **TKMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MMXS1	MMXS0	MDFEN	MFILEN	MSOFC	MSOF2	MSOF1	MSOF0
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 **MMXS1~MMXS0**：触控按键模块多路复用按键选择
00：KEY1
01：KEY2
10：KEY3
11：KEY4

Bit 5 **MDFEN**：触控按键模块倍频功能控制位
0：除能
1：使能
此位用于控制触控按键振荡器频率的倍频功能。当此位置“1”，按键振荡器频率将为原来的倍频。

Bit 4 **MFILEN**：触控按键模块过滤功能控制
0：除能
1：使能

Bit 3 **MSOFC**：触控按键模块 C/F 振荡器跳频功能选择位
0：软件处理跳频功能，由 MSOF2~MSOF0 位决定
1：硬件处理跳频功能，MSOF2~MSOF0 位无作用
该位用来选择触控按键振荡器跳频功能控制方式，当此位置 1，按键振荡器跳频功能由硬件电路控制，而不受 MSOF2~MSOF0 位影响。

Bit 2~0 **MSOF2~MSOF0**：触控按键模块参考和按键振荡器跳频选择位
000：1.020MHz
001：1.040MHz
010：1.059MHz
011：1.074MHz
100：1.085MHz
101：1.099MHz
110：1.111MHz
111：1.125MHz
这些位用于为调频功能选择触控按键振荡器频率。应注意的是这些位仅在 MSOFC 位清零时有效。
上述频率会随着外部或内部电容值的不同而变化。若触控按键振荡器频率选择 1MHz，用户选择其它频率时，可依此比例调整。

- **TKMC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MTSS	—	MROEN	MKOEN	MK4EN	MK3EN	MK2EN	MK1EN
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 **MTSS**：触控按键模块时隙计数器时钟源选择位
0：触控按键模块参考振荡器
1：$f_{SYS}/4$

Bit 6 未定义，读为“0”

Bit 5 **MROEN**：触控按键模块参考振荡器使能控制位
0：除能
1：使能
该位用于使能触控按键模块参考振荡器。若选择使用参考振荡器，在将 TKST 位设置为低到高前，必须先使能参考振荡器。

Bit 4 **MKOEN**：触控按键模块按键振荡器使能控制位
0：除能
1：使能
该位用于使能触控按键模块按键振荡器。若使能相应的按键进行扫描时，在将 TKST 位设置为低到高前，必须先使能按键振荡器。

Bit 3 **MK4EN**：触控按键模块 KEY 4 使能控制
0：除能
1：使能

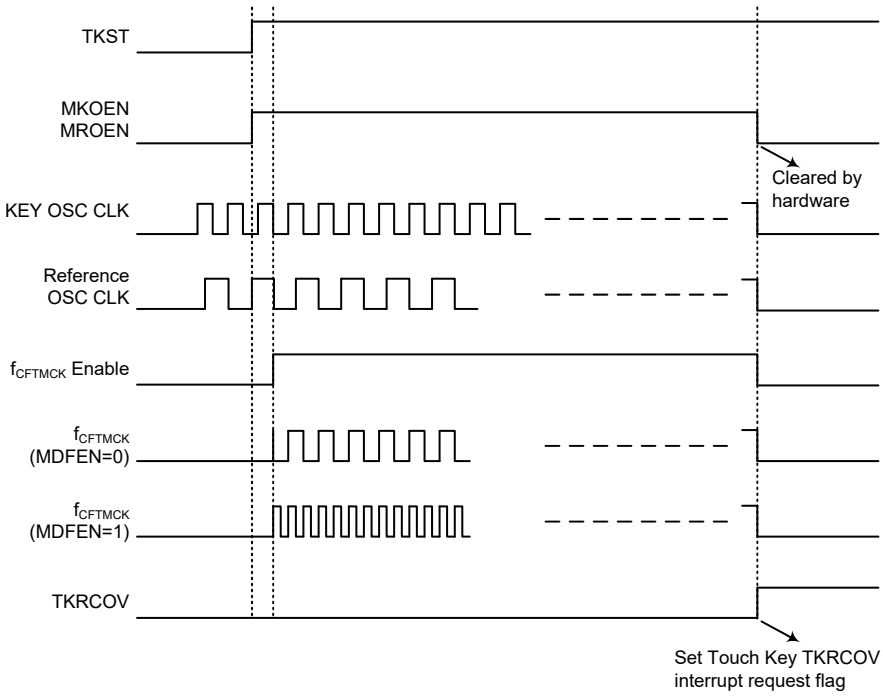
Bit 2 **MK3EN**：触控按键模块 KEY 3 使能控制
0：除能
1：使能

Bit 1 **MK2EN**：触控按键模块 KEY 2 使能控制
0：除能
1：使能

Bit 0 **MK1EN**：触控按键模块 KEY 1 使能控制
0：除能
1：使能

触控按键操作

手指接近或接触到触控面板时，面板的电容量会增大，电容量的变化会轻微改变内部感应振荡器的频率，通过测量频率的变化可以感知触控动作。参考时钟通过内部可编程分频器能够产生一个固定的时间周期。在这个时间周期内，通过在此固定时间周期内对感应振荡器产生的时钟周期计数，可确定触控按键的动作。



触控按键模块时序图

触控按键模块包含四个与 I/O 引脚共用的触控按键输入 KEY1~KEY4，通过寄存器可设置相应引脚功能。触控按键具有独立的感应振荡器，因此模块中包含四个感应振荡器。

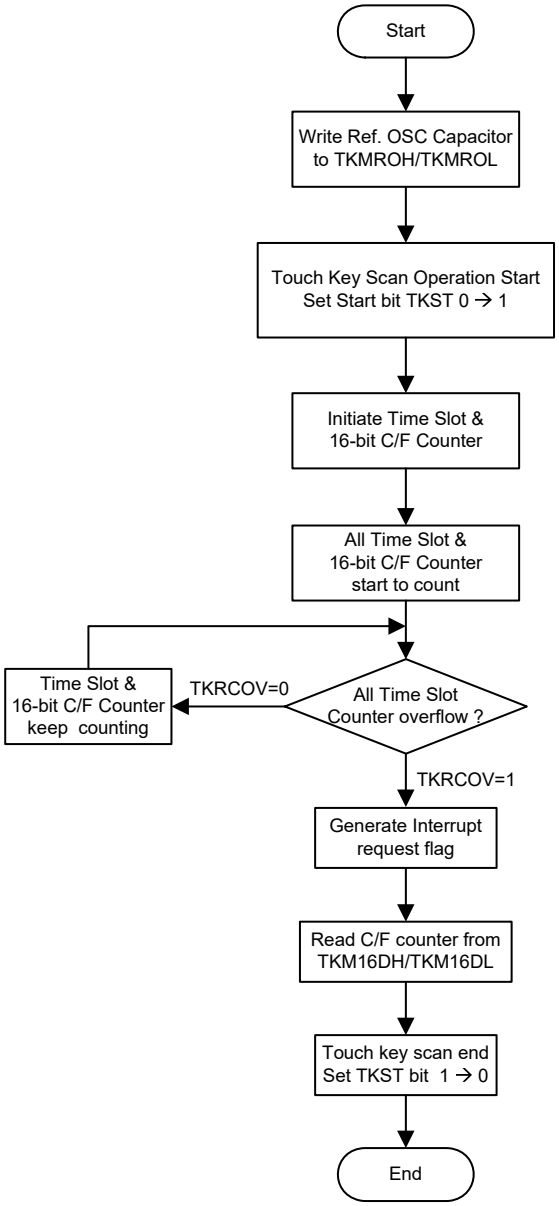
在参考时钟固定的时间间隔内，感应振荡器产生的时钟周期数是可以测量的。测到的周期数可以用于判断触控动作是否有效发生。在最后一个时间间隔后，会产生一个触控按键中断信号。

在 TKST 位清零时，触控按键模块的 16-bit C/F 计数器、16-bit 计数器和 5-bit 时隙计数器会自动清零，而 8-bit 可编程时隙计数器不清零，由用户设置溢出时间。在 TKST 上升沿时，16-bit C/F 计数器、16-bit 计数器、5-bit 时隙计数器和 8-bit 时隙计数器会自动开启。

当时隙计数器溢出，模块中的按键振荡器和参考振荡器都会自动停止且 16-bit C/F 计数器、16-bit 计数器、5-bit 时隙计数器和 8-bit 时隙计数器会自动停止。通过 TKMC1 寄存器中的 MTSS 位来选择时隙计数器时钟来自参考振荡器或 $f_{SYS}/4$。当 TKMC1 寄存器中的 MROEN 位和 MKOEN 位为“1”时，选到的参考振荡器和按键振荡器就使能。

当触控按键模块的时隙计数器溢出时，将产生实际的触控按键中断。这里所有的触控按键是指已使能的触控按键。

触控按键扫描工作流程图



触控按键扫描工作流程图

触控按键中断

触控按键只有一个中断，触控按键模块的时隙计数器溢出时，将产生实际的触控按键中断，这里所有的触控按键是指已使能的触控按键。此时模块中的 16-bit C/F 计数器、16-bit 计数器，5-bit 时隙计数器和 8-bit 时隙计数器会自动清零。

当触控按键模块 16-bit C/F 计数器溢出时，16-bit C/F 计数器溢出标志位 TKCFOV 将置高。由于此标志位无法被自动清零，需通过应用程序将此位清零。

当 16-bit 计数器溢出时，其溢出标志位 TK16OV 将置高。由于此标志位无法被自动清零，需通过应用程序将此位清零。更多详细内容见规格书中断章节中的“触控按键中断”。

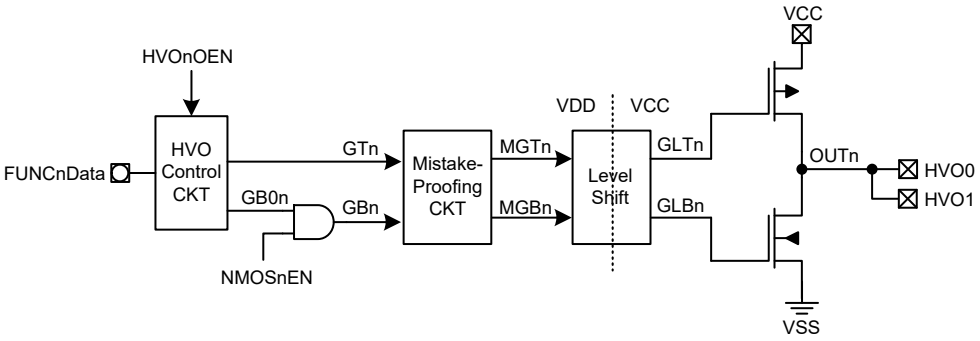
编程注意事项

相关寄存器设置后，TKST 位由低电平变为高电平，触控按键检测程序启动。此时所有相关的振荡器将使能并同步。时隙计数器标志位 TKRCOV 将变为高电平直到计数器溢出。计数器溢出发生时，会产生一个中断信号。

当外部触控按键的大小和布局确定时，其相关的电容将决定感应振荡器的频率。

高压驱动器输出

该单片机的高压驱动电路包含两组高压输出。每组高压驱动电路由两个主要电路组成即防呆电路及电平转换器电路。高压驱动电路内建两条 12V 高压制程的 Level Shift 电路，提供两条输出 HVO0~HVO1，可通过 PMOS 驱动 LED。



注：1. n=0 或 1。

2. FUNC0Data 来源于 PWM0/PWM1/HVO0 引脚，FUNC1Data 来源于 PWM1/PWM0/HVO1 引脚，作为控制信号输出高 / 低数据源。

3. 驱动 LED 时，建议将 NMOSnEN 位清零。

高压驱动输出方框图

高压驱动输出寄存器

高压驱动输出的所有操作是通过寄存器 HVOC 和 HVOC1 来控制的。HVOC 寄存器控制高压驱动输出使能 / 除能操作，HVOC1 寄存器用来选择 HVOn 引脚来源及高压驱动输出数据。

- **HVOC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	NMOS1EN	HVO1OEN	NMOS0EN	HVO0OEN
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　**NMOS1EN**：NMOS1 输出使能控制位
0：NMOS1 始终除能
1：NMOS1 由 GB01 信号控制
注：若 PWM0 或 PWM1 引脚被选为高压输出，NMOS1EN 位必须清零。

Bit 2　**HVO1OEN**：高压驱动输出 1 使能控制位
0：除能
1：使能
若 HVO1OEN 位被清零，HVO1 引脚将处于浮空状态。

Bit 1 **NMOS0EN**：NMOS0 输出使能控制位
0：NMOS0 始终除能
1：NMOS0 由 GB00 信号控制
注：若 PWM0 或 PWM1 引脚被选为高压输出，NMOS0EN 位必须清零。

Bit 0 **HVO0OEN**：高压驱动输出 0 使能控制位
0：除能
1：使能
若 HVO0OEN 位被清零，HVO0 引脚将处于浮空状态。

- **HVOC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	HVO1S1	HVO1S0	HVO0S1	HVO0S0	—	—	HVOD1	HVOD0
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7~6 **HVO1S1~HVO1S0**：高压驱动输出 1 选择
00：HVO1
01：HVO1
10：PWM0
11：PWM1

Bit 5~4 **HVO0S1~HVO0S0**：高压驱动输出 0 选择
00：HVO0
01：HVO0
10：PWM0
11：PWM1

Bit 3~2 未定义，读为 “0”

Bit 1 **HVOD1**：高压驱动输出 1 数据
1：HVO1=VCC
0：HVO1=VSS

Bit 0 **HVOD0**：高压驱动输出 0 数据
1：HVO0=VCC
0：HVO0=VSS

功能说明

高压驱动器从 HVOn 引脚输出，并通过 HVOnOEN 位驱动外部功率 MOS，或使用 FUNCnData 输出来控制功率 MOS 负载。

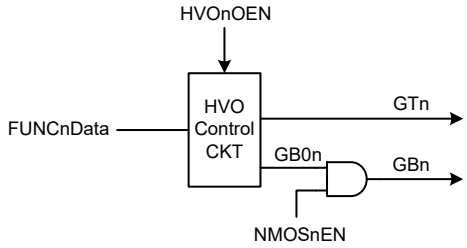
高压驱动器 Mistake-Proofing 电路

当软件有误写动作发生或是因外部因素，如 ESD 发生时，造成外部驱动晶体管对上臂与下臂的输出 MOS 同时开启的状态。Mistake-Proofing 电路设计的目的为了避免这种情况。

GTn	GBn	MGTn	MGBn	HVOn
0	0	0	0	VCC
0	1	1	1	VSS
1	0	1	0	浮空
1	1	1	1	VSS

注：0：MOS 关闭
1：MOS 开启

高压驱动输出控制电路

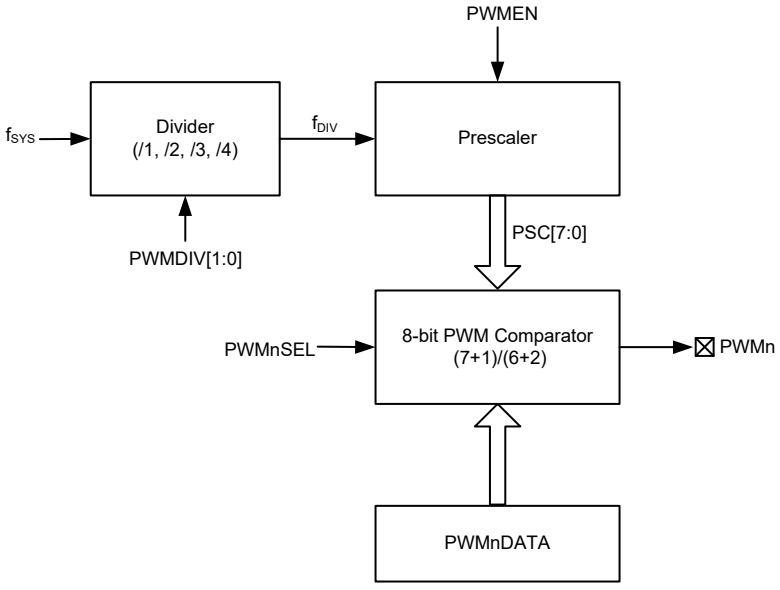


HVOnOEN	HVO 功能数据	NMOSnEN	GTn	GBn	HVOn
1	0	1	1	1	VSS
1	1	1	0	0	VCC
1	0	0	1	0	浮空
1	1	0	0	0	VCC
0	x	x	1	0	浮空

“x”表示无关

脉冲宽度调制 – PWM

该单片机提供两路 8 位的脉冲宽度调制 (PWM) 输出。这在马达速率控制应用方面十分有用，通过给相应的 PWMnDATA 寄存器设定一定数值，PWM 功能可提供占空比可调但频率固定的 PWM 信号输出。



PWM 方框图

PWM 寄存器说明

脉宽调制的所有操作是通过两个寄存器来控制的，一个数据寄存器 PWMnDATA 和一个控制寄存器 PWMC。PWM 计数器的频率来源于预分频器的输出信号。

- **PWMC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PWMEN	PWMDIV1	PWMDIV0	—	—	—	PWM1SEL	PWM0SEL
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	0	0	0	—	—	—	0	0

Bit 7 **PWMEN**：PWM 使能控制位
0：除能
1：使能

Bit 6~5 **PWMDIV1~PWMDIV0**：f_{DIV} 频率选择
00：$f_{DIV}=f_{SYS}$
01：$f_{DIV}=f_{SYS}/2$
10：$f_{DIV}=f_{SYS}/3$
11：$f_{DIV}=f_{SYS}/4$

Bit 4~2 未定义，读为“0”

Bit 1 **PWM1SEL**：PWM1 类型选择
0：(6+2) 模式
1：(7+1) 模式

Bit 0 **PWM0SEL**：PWM0 类型选择
0：(6+2) 模式
1：(7+1) 模式

- **PWMnDATA 寄存器 (n=0, 1)**

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**：PWM 数据位

PWM 工作模式

在数据存储器中，单片机为每一个 PWM 都指定了对应的寄存器，称为 PWMC 寄存器和 PWMnDATA 寄存器。PWMnDATA 寄存器表示输出波形中每个调制周期的占空比。为了提高 PWM 调制频率，每个调制周期被调制成两个或四个独立的调制子区段，即分别是 7+1 模式或 6+2 模式。可以通过设置 PWMC 寄存器来选择每个 PWM 通道所需的模式和使能 / 除能控制。注意，当使用 PWM 时，只要将所需的值写入相应的 PWMnDATA 寄存器并通过 PWMC 寄存器设置所需模式和使能 / 除能控制，单片机内部电路即自动完成 PWM 各子调制周期的划分输出 PWM 信号。PWM 的时钟源为系统时钟 f_{SYS}。

将原始调制周期分成 2 个或 4 个子周期的方法，使产生更高的 PWM 频率成为可能，这样可以提供更广泛的应用。使用者需要理解 PWM 频率与 PWM 调制频率的不同之处。PWM 时钟为系统时钟 f_{SYS}，当 PWM 值为 8 位时，整个 PWM 周期的频率为 $f_{SYS}/256$。在 7+1 模式，PWM 调制频率将会是 $f_{SYS}/128$，在 6+2 模式，PWM 调制频率将会是 $f_{SYS}/64$。

PWM 输出信号的调制频率，周期频率以及周期占空比详见下表。

PWM 调制频率	PWM 周期频率	PWM 占空比
$f_{DIV}/64$ 用于 (6+2) 模式	$f_{DIV}/256$	(PWMnDATA 寄存器值)/256
$f_{DIV}/128$ 用于 (7+1) 模式		

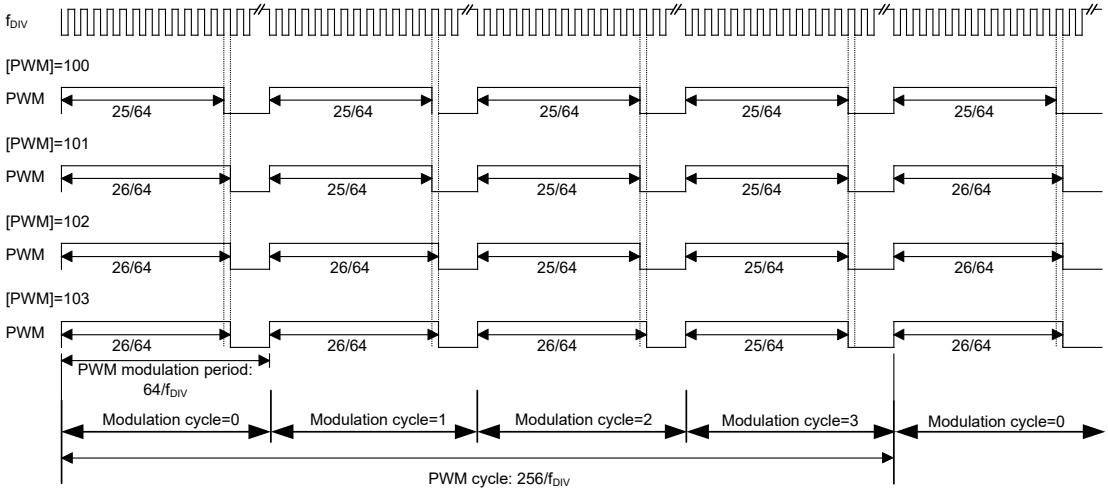
6+2 PWM 模式调制

在 6+2 PWM 模式中，每个 PWM 周期又被分成四个独立的子周期，称为调制周期 0~ 调制周期 3，在表格中以 “i” 表示。四个子周期各包含 64 个时钟周期。在这个模式下，8 位的 PWMnDATA 寄存器被分成两个部分，这个寄存器的值表明整个 PWM 波形的占空比。第一部分包括 PWMnDATA 寄存器的 D7~D2 位，表示 DC 值，第二部分为 PWMnDATA 寄存器的 D1~D0 位，表示 AC 值。在 6+2 PWM 模式中，四个调制子周期的占空比，分别如下表所示。

参数	AC (0~3)	DC (占空比)
调制周期 i (i=0~3)	i < AC	(DC+1)/64
	i ≥ AC	DC/64

6+2 模式调制周期值

下图表示在 6+2 模式下 PWM 输出的波形。请特别注意单个的 PWM 周期是如何被划分为四个单独的调制周期 0 ～ 3 以及 AC 值与 PWM 值之间的关系。PWM 输出波形如下所示。



6+2 PWM 模式调制波形

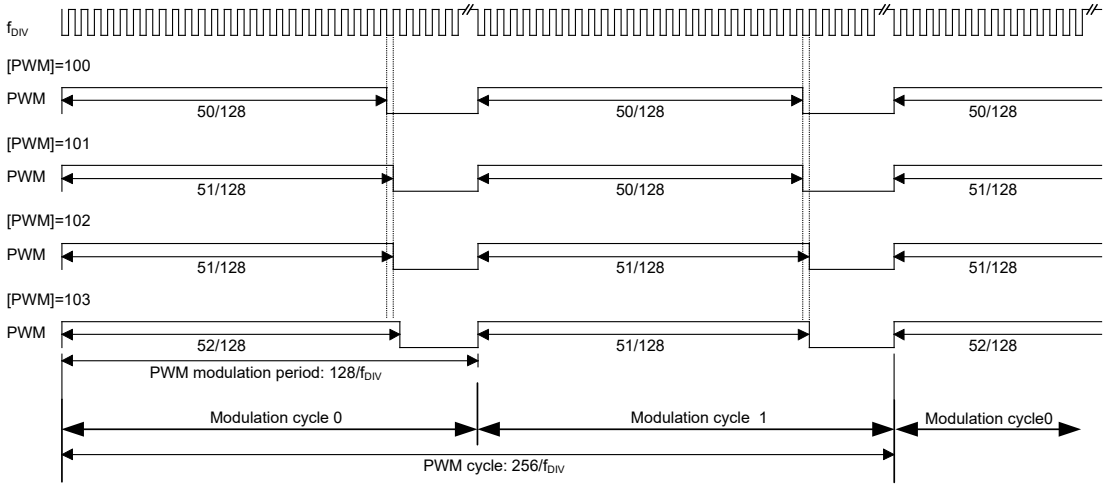
7+1 PWM 模式调制

在 7+1 PWM 模式中，每个 PWM 周期又被分成两个独立的子周期，称为调制周期 0~ 调制周期 1，在表格中以 “i” 表示。两个子周期各包含 128 个时钟周期。在这个模式下，8 位的 PWMnDATA 寄存器被分成两个部分，这个寄存器的值表明整个 PWM 波形的占空比。第一部分包括 PWMnDATA 寄存器的 D7~D1 位，表示 DC 值，第二部分为 PWMnDATA 寄存器的 D0 位，表示 AC 值。在 7+1 PWM 模式中，两个调制子周期的占空比，分别如下表所示。

参数	AC (0~1)	DC (占空比)
调制周期 i (i=0~1)	i < AC	(DC+1)/128
	i ≥ AC	DC/128

7+1 模式调制周期值

下图表示在 7+1 模式下 PWM 输出的波形。请特别注意单个的 PWM 周期是如何被划分为两个单独的调制周期 0 ～ 1 以及 AC 值与 PWM 值之间的关系。PWM 输出波形如下所示。



7+1 PWM 模式调制波形

PWM 输出控制

PWM 输出与 HVOn 引脚共用。通过正确地设置 HVOC1 寄存器可使该引脚作为一个 PWM 输出而不是一个高压输出。

- **HVOC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	HVO1S1	HVO1S0	HVO0S1	HVO0S0	—	—	HVOD1	HVOD0
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7~6 **HVO1S1~HVO1S0**：高压驱动输出 1 选择
00：HVO1
01：HVO1
10：PWM0
11：PWM1

Bit 5~4 **HVO0S1~HVO0S0**：高压驱动输出 0 选择
00：HVO0
01：HVO0
10：PWM0
11：PWM1

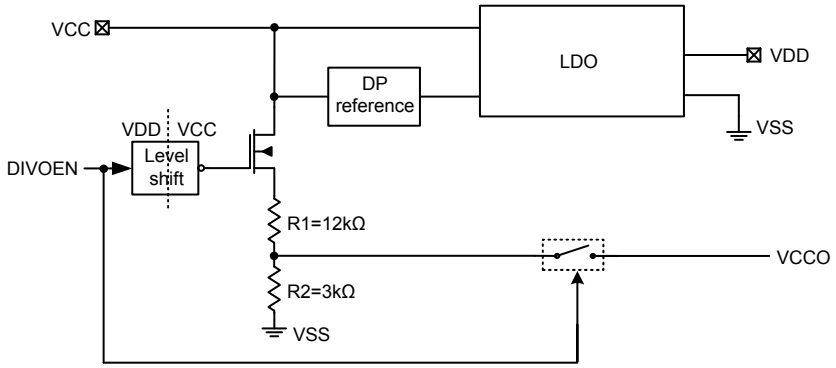
Bit 3~2 未定义，读为“0”

Bit 1 **HVOD1**：高压驱动输出 1 数据
详见其它章节。

Bit 0 **HVOD0**：高压驱动输出 0 数据
详见其它章节。

低压降稳压器 – LDO

该系统由输入引脚 VCC 上大约 6V~12V 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 40mA，因此它也可用于外部电路。



注：分压器：R1:R2=12kΩ:3kΩ=4:1，V_{CCO}=R2/(R1+R2)×V_{CC}=0.2×V_{CC}。

LDO 控制寄存器

- **DIVOC 寄存器**

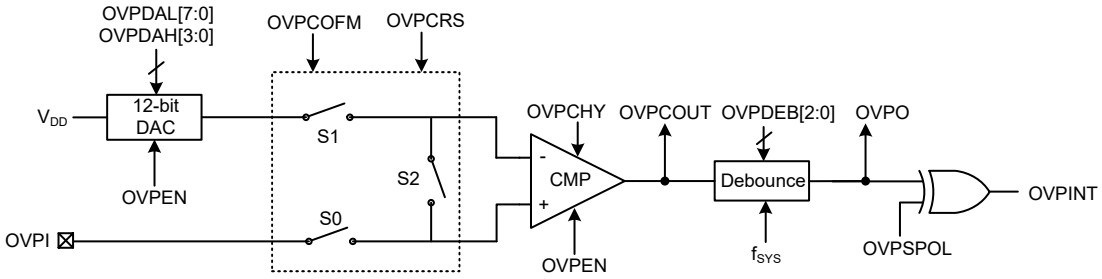
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	DIVOEN
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1　未定义，读为“0”

Bit 0　**DIVOEN**：V_{CC} 除法器输出使能控制
0：除能
1：使能
当 DIVOEN 位被清零，VCCO 将浮空。
当 DIVOEN 位被置高，VCCO 电压为 V_{CC}/5。

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。为了防止工作电压超过特定值，可将 OVPI 引脚的电压与 12-bit DAC 产生的参考电压进行比较。当发生过压情况时，若相应中断已使能，则产生 OVP 中断。



过电压保护电路

开关 S0~S2 的 ON/OFF 状态控制如下表所示。

OVPCOFM	OVPCRS	S0	S1	S2
0	x	ON	ON	OFF
1	0	OFF	ON	ON
1	1	ON	OFF	ON

“x”：无关

OVP 操作

电压源供给 OVPI 引脚，并连接至比较器的其中一个输入。D/A 转换器用于产生一个参考电压。比较器将参考电压和输入电压进行比较产生 OVPO 信号。

OVP 控制寄存器

过电压保护的所有操作是通过一系列寄存器来控制的。一个寄存器用于为过电压保护电路提供参考电压。剩下的两个控制寄存器用于控制 OVP 功能、D/A 转换器参考电压选择、比较器去抖时间、比较器迟滞功能和比较器输入失调校准等功能。

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

OVP 控制寄存器列表

- **OVPC0 寄存器**

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

Bit 7 **OVPCOUT**：OVP 比较器输出
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
0：同相
1：反相

Bit 5 **OVPEN**：OVP 功能控制位
0：除能
1：使能
若 OVPEN 位清零，过电压保护功能将除能以节省功耗。此时 OVP 中的比较器和 D/A 转换器也全部关闭。

Bit 4 **OVPCHY**：OVP 比较器迟滞功能控制位
0：除能
1：使能

Bit 3 未定义，读为“0”

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

- **OVPC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OVPO	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 **OVPO**：OVP 比较器去抖输出
比较器输出 OVPCOUT 去抖后将得到 OVPO。

Bit 6 **OVPCOFM**：OVP 比较器正常工作模式或输入失调电压校准模式选择
0：正常工作模式
1：输入失调电压校准模式

Bit 5 **OVPCRS**：OVP 比较器输入失调校准参考输入选择
0：选择负输入作为参考输入
1：选择正输入作为参考输入

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

- **OVPDAL 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**：OVP DAC 输出电压控制

注：写入到OVPDAL寄存器的数据只写入阴影缓冲区，直到数据被写入OVPDAH寄存器，数据才会写入到 OVPDAL 寄存器。

- **OVPDAH 寄存器**

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

Bit 7~4 未定义，读为 “0”

Bit 3~0 **D3~D0**：OVP DAC 输出电压控制

DAC $V_{OUT}=(V_{DD}/4096)\times\{OVPDAH[3:0], OVPDAL[7:0]\}$

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 OVPCHY 位为零确保无迟滞电压。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。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，读取 OVPO 位的状态。

步骤 3：使 OVPCOF[4:0]=OVPCOF[4:0]+1，读取 OVPO 位的状态。若 OVPO 位状态改变，记录此时 OVPCOF[4:0] 的数据为 V_{OS1}。

步骤 4：设置 OVPCOF[4:0]=11111，读取 OVPO 位的状态。

步骤 5：使 OVPCOF[4:0]=OVPCOF[4:0] - 1，读取 OVPO 位的状态。若 OVPO 位状态改变，记录此时 OVPCOF[4:0] 的数据为 V_{OS2}。

步骤 6：将 $V_{OS}=(V_{OS1}+V_{OS2})/2$ 存入 OVPCOF[4:0] 位中，校准结束。

若 $(V_{OS1}+V_{OS2})/2$ 不是整数，忽略小数部分。$V_{OS}=V_{OUT} - V_{IN}$

中断

中断是单片机一个重要功能。当外部事件或内部功能如触摸动作或定时 / 事件计数器溢出有效，并且产生中断时，系统会暂停当前的程序而转到执行相对应的中断服务程序。此单片机提供一个外部中断和一些内部中断功能，外部中断由 INT 引脚动作产生，而内部中断由各种内部功能，如触控按键、定时 / 事件计数器、时基和过压保护等产生。

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数据存储器中的一系列寄存器控制的，如表所示。寄存器总的分为两类。第一类是 INTC0~INTC1 寄存器，用于设置基本的中断；第二类是 INTEG 寄存器，用于设置外部中断边沿触发类型。

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

功能	使能位	请求标志
总中断	EMI	—
外部中断	INTE	INTF
触控按键	TKME	TKMF
定时 / 事件计数器	TE	TF
过压保护	OVPE	OVPF
时基	TBE	TBF

中断寄存器位命名模式

寄存器名称	位							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	TF	TKMF	INTF	TE	TKME	INTE	EMI
INTC1	—	—	TBF	OVPF	—	—	TBE	OVPE

中断寄存器列表

- **INTEG 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INTS1	INTS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未定义，读为“0”

Bit 1~0　**INTS1~INTS0**：INT 脚中断边沿控制位
00：除能
01：上升沿
10：下降沿
11：双沿

- **INTC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	TF	TKMF	INTF	TE	TKME	INTE	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　**TF**：定时 / 事件计数器中断请求标志位
0：无请求
1：中断请求

Bit 5　**TKMF**：触控按键模块中断请求标志位
0：无请求
1：中断请求

Bit 4　**INTF**：外部中断请求标志位
0：无请求
1：中断请求

Bit 3　**TE**：定时 / 事件计数器中断控制位
0：除能
1：使能

Bit 2　**TKME**：触控按键模块中断控制位
0：除能
1：使能

Bit 1　**INTE**：外部中断控制位
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	—	—	TBF	OVPF	—	—	TBE	OVPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6　未定义，读为“0”

Bit 5　**TBF**：时基中断请求标志位
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 3~2　未定义，读为“0

Bit 1　**TBE**：时基中断控制位
0：除能
1：使能

Bit 0　**OVPE**：OVP 中断控制位
0：除能
1：使能

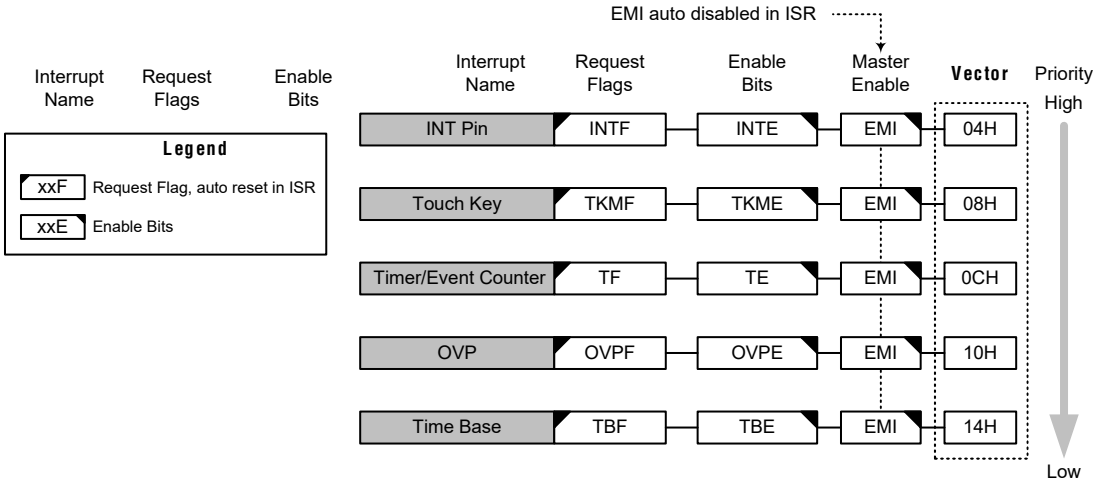
中断操作

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

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

各个中断使能位以及相应的请求标志位，以优先级的次序显示在下图。一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

外部中断

通过 INT 引脚上的信号变化可控制外部中断。当触发沿选择位设置好触发类型，INT 引脚的状态发生变化，外部中断请求标志 INTF 被置位时外部中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和相应中断使能位 INTE 需先被置位。此外，必须使用 INTEG 寄存器使能外部中断功能并选择触发沿类型。外部中断引脚和普通 I/O 口共用，如果相应寄存器中的中断使能位被置位和通过相关引脚共用功能选择位选择此引脚被作为外部中断脚使用。此时该引脚必须通过设置相关控制寄存器，将该引脚设置为输入口。当中断使能，堆栈未满并且外部中断脚状态改变，将调用外部中断向量子程序。当响应外部中断服务子程序时，中断请求标志位 INTF 会自动复位且 EMI 位会被清零以除能其它中断。注意，即使此引脚被用作外部中断输入，其配置选项中的上拉电阻仍保持有效。

寄存器 INTEG 被用来选择有效的边沿类型，来触发外部中断。可以选择上升沿还是下降沿或双沿触发都产生外部中断。注意 INTEG 也可以用来除能外部中断功能。

触控按键模块中断

当触控按键时隙计数器溢出，触控按键模块中断请求标志位 TKMF 将置位，触控按键中断请求产生。当总中断使能位 EMI 和触控按键模块中断使能位 TKME 被置位，允许程序跳转到各自的中断向量地址。中断使能，堆栈未满，当触控按键时隙计数器溢出发生中断时，将调用触控按键中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 TKMF 会被自动复位且 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 / 事件计数器溢出，定时 / 事件计数器中断请求标志位 TF 将置位，定时 / 事件计数器中断请求产生。当总中断使能位 EMI 和定时 / 事件计数器中断使能位 TE 被置位，允许程序跳转到各自的中断向量地址。中断使能，堆栈未满，当定时 / 事件计数器溢出发生中断时，将调用定时 / 事件计数器中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 TF 会被自动复位且 EMI 位会被清零以除能其它中断。

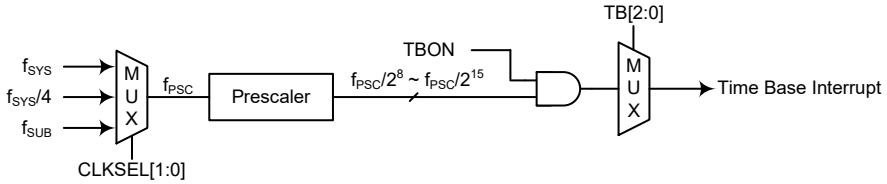
过压保护中断

通过检测 OVP 输入电压决定是否产生 OVP 中断。当 OVP 中断请求标志 OVPF 被置位，即检测到过压情况时，OVP 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 和过压保护中断使能位 OVPE 需先被置位。当中断使能，检测到过压情况时，将调用 OVP 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 OVPF 会自动复位且 EMI 位会被清零以除能其它中断。

时基中断

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

时基中断的目的是提供一个固定周期的中断信号。其时钟源 f_{PSC} 来自内部时钟源 f_{SYS}、$f_{SYS}/4$ 或 f_{SUB}。输入时钟首先经过分频器，分频率由程序设置 TBC 寄存器相关位获取合适的分频值以提供更长的时基中断周期。相应的控制时基中断周期的时钟源可通过 PSCR 寄存器的 CLKSEL1~CLKSEL0 位选择。



时基中断

- **PSCR 寄存器**

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

Bit 7~2　未定义，读为“0”

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

- **TBC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TBON	—	—	—	—	TB2	TB1	TB0
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7　**TBON**：时基控制位
0：除能
1：使能

Bit 6~3　未定义，读为“0”

Bit 2~0 **TB2~TB0**：时基溢出周期选择位
000：$2^8/f_{PSC}$
001：$2^9/f_{PSC}$
010：$2^{10}/f_{PSC}$
011：$2^{11}/f_{PSC}$
100：$2^{12}/f_{PSC}$
101：$2^{13}/f_{PSC}$
110：$2^{14}/f_{PSC}$
111：$2^{15}/f_{PSC}$

中断唤醒功能

每个中断都具有将处于休眠或空闲模式的单片机唤醒的能力。当中断请求标志由低到高转换时唤醒动作产生，其与中断是否使能无关。因此，尽管单片机处于休眠或空闲模式且系统振荡器停止工作，如有外部中断脚上产生外部边沿跳变可能导致其相应的中断标志被置位，由此产生中断。

因此必须注意避免伪唤醒情况的发生。若中断唤醒功能被除能，单片机进入休眠或空闲模式前相应中断请求标志应被置起。中断唤醒功能不受中断使能位的影响。

编程注意事项

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

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高。

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

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

应用说明 – 调光调色触控台灯

简介

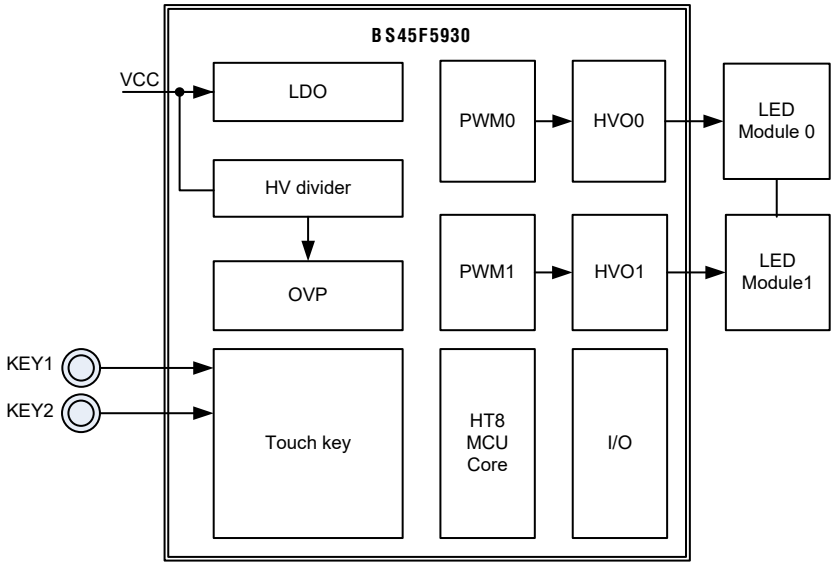
调光调色触控台灯主要包括触控侦测、PWM 调制、高压驱动三项功能。当输入 12V 工作电压时，若 MCU 侦测到按键即会改变亮度。当两路 LED 分别使用黄光及白光时，不同的黄与白光比例即可产生不同的光色，其功能原理介绍如下。

功能说明

本系统电源电压为 12V。每路 LED 为 3 颗 LED 串联，每路使用一个限流电阻限制流过 LED 的最大电流，保护 LED 不会因电流过大而烧毁。另外该单片机还提供过电压保护 OVP 功能，搭配高压分压器可侦测 VCC 电压，当 VCC 电压过高时可将高压驱动输出关闭，以避免大电流将 LED 烧毁。本范例 LED 模块 0 使用白光 LED，LED 模块 1 采用暖白光 LED，利用调整两路 PWM 占空比以不同的亮度比例混和出不同色温。

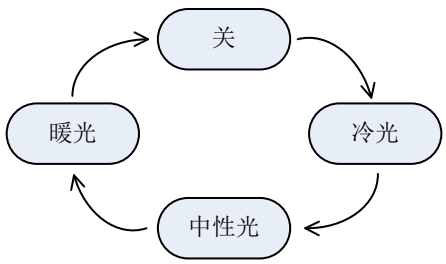
灯光控制的部分使用两个触控按键，KEY1 用来调整色温，KEY2 用来调整亮度。

硬件方框图



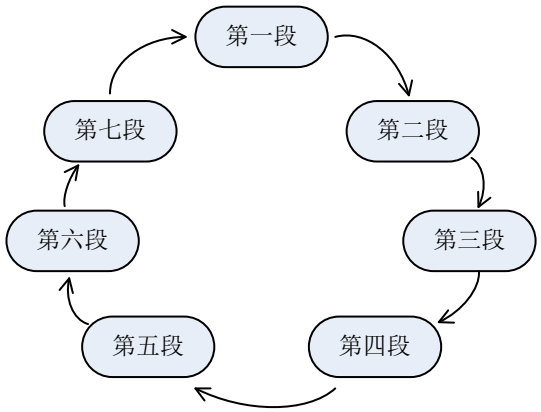
色温调节键 (KEY1)

采用循环式设定，上电预设为关闭，轻触 KEY1 则切换到冷光模式，再轻触 KEY1 则切为中性光，以此类推，如下图。



调光键 (KEY2)

调光键采用循环式设定，上电预设为第一段，轻触 KEY2 则切换到第二段，再轻触 KEY2 则切换到第三段，以此类推，本范例有七段亮度选择，如下图。



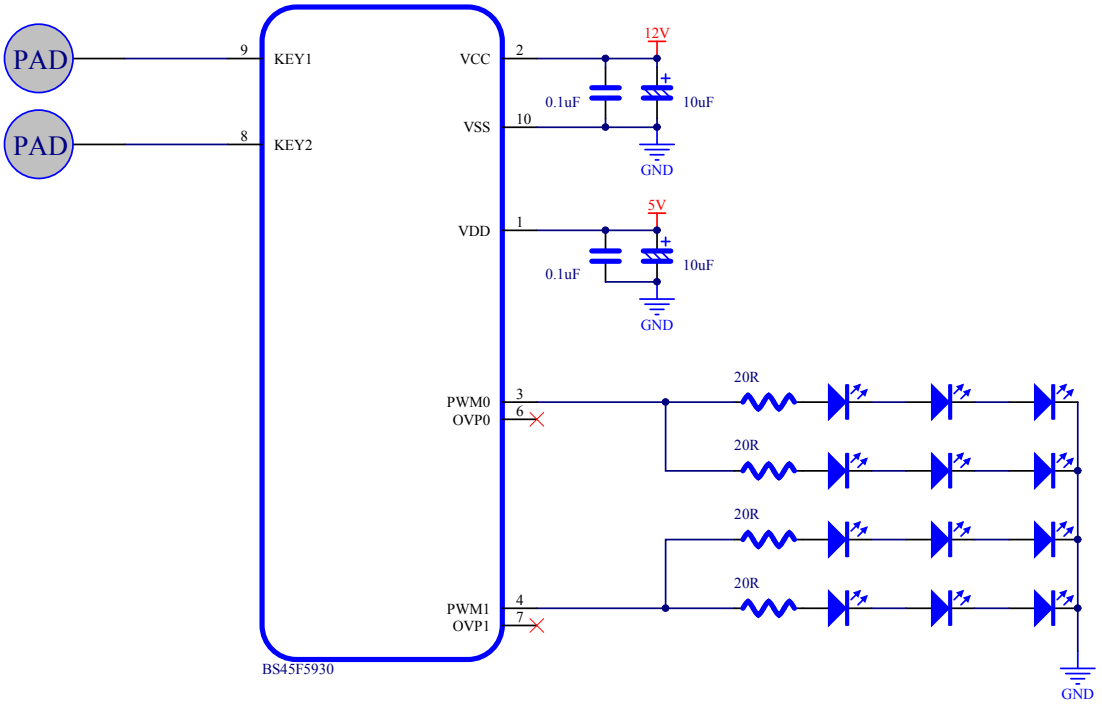
色温定义

本范例定义如下表，表中的数字标示为 PWM 的占空比，实现在同一个色温模式下改变亮度不影响色温，所以在亮度增加时两路 LED 的占空比并不是等比增加。

PWM 占空比	暖光		中性光		冷光	
	暖白	正白	暖白	正白	暖白	正白
第一级	9%	2%	5%	5%	2%	9%
第二级	23%	3%	12%	12%	3%	23%
第三级	36%	4%	20%	20%	4%	36%
第四级	50%	6%	28%	28%	6%	50%
第五级	63%	7%	35%	35%	7%	63%
第六级	77%	9%	43%	43%	9%	77%
第七级	90%	10%	50%	50%	10%	90%

注：表中数值仅供参考，不同 LED 灯会有不同设定。

硬件电路图


PAD
PAD
9
KEY1
8
KEY2
VCC
2
VSS
10
VDD
1
12V
0.1uF
10uF
GND
5V
0.1uF
10uF
GND
PWM0
OVP0
3
6
PWM1
OVP1
4
7
20R
20R
20R
20R
BS45F5930
GND

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

下表中说明了按功能分类的指令集，用户可以将该表作为基本的指令参考。

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
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
ADD A, x	ACC 与立即数相加，结果放入 ACC	1	Z, C, AC, OV
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
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
SUB A, x	ACC 与立即数相减，结果放入 ACC	1	Z, C, AC, OV
SUB A,[m]	ACC 与数据存储器相减，结果放入 ACC	1	Z, C, AC, OV
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
SBC A,[m]	ACC 与数据存储器、进位标志的反相减，结果放入 ACC	1	Z, C, AC, OV
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
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
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
CALL0 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
TABRDC [m]	读取当前页的 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
其它指令			
NOP	空指令	1	无
CLR [m]	清除数据存储器	1 注	无
SET [m]	置位数据存储器	1 注	无
CLR WDT	清除看门狗定时器	1	TO, PDF
CLR WDT1	预清除看门狗定时器	1	TO, PDF
CLR WDT2	预清除看门狗定时器	1	TO, PDF

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

注：1. 对跳转指令而言，如果比较的结果牵涉到跳转即需多达 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。

3. 对于“CLR WDT1”或“CLR WDT2”指令而言，TO 和 PDF 标志位也许会受执行结果影响，“CLR WDT1”和“CLR WDT2”被连续地执行后，TO 和 PDF 标志位会被清除，否则 TO 和 PDF 标志位保持不变

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

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

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

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

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

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

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

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

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

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

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

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

CLR WDT1 Preclear Watchdog Timer

指令说明 PDF 和 TO 标志位都被清 0。必须配合 CLR WDT2 一起使用清除 WDT 计时器。当程序仅执行 CLR WDT1，而没有执行 CLR WDT2 时，PDF 与 TO 保留原状态不变。

功能表示 WDT ← 00H
TO & PDF ← 0

影响标志位 TO、PDF

CLR WDT2 Preclear Watchdog Timer

指令说明 PDF 和 TO 标志位都被清 0。必须配合 CLR WDT1 一起使用清除 WDT 计时器。当程序仅执行 CLR WDT2，而没有执行 CLR WDT1 时，PDF 与 TO 保留原状态不变。

功能表示 WDT ← 00H
TO & PDF ← 0

影响标志位 TO、PDF

CPL [m] Complement Data Memory

指令说明 将指定数据存储器中的每一位取逻辑反，
相当于从 1 变 0 或 0 变 1。

功能表示 [m] ← $[\overline{m}]$

影响标志位 Z

CPLA [m] Complement Data Memory with result in ACC

指令说明 将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1，而结果被储存回累加器且数据存储器中的内容不变。

功能表示 ACC ← $[\overline{m}]$

影响标志位 Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

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

功能表示 [m] ← ACC+00H 或
[m] ← ACC+06H 或
[m] ← ACC+60H 或
[m] ← ACC+66H

影响标志位 C

DEC [m] Decrement Data Memory

指令说明 将指定数据存储器内容减 1。

功能表示 [m] ← [m] – 1

影响标志位 Z

DECA [m] Decrement Data Memory with result in ACC

指令说明 将指定数据存储器的内容减 1，把结果存放回累加器并保持指定数据存储器的内容不变。

功能表示 ACC ← [m] – 1

影响标志位 Z

HALT Enter power down mode

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

功能表示 TO ← 0
PDF ← 1

影响标志位 TO、PDF

INC [m] Increment Data Memory

指令说明 将指定数据存储器的内容加 1。

功能表示 [m] ← [m]+1

影响标志位 Z

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

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

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

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

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

NOP	No operation
指令说明	空操作，接下来顺序执行下一条指令。
功能表示	PC ← PC+1
影响标志位	无

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

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

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

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

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

RETI	Return from interrupt
指令说明	将堆栈寄存器中的程序计数器值恢复且中断功能通过设置 EMI 位重新使能。EMI 是控制中断使能的主控制位。如果 在执行 RETI 指令之前还有中断未被相应，则这个中断将 在返回主程序之前被相应。
功能表示	Program Counter ← Stack EMI ← 1
影响标志位	无

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

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

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

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

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

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

RRC [m] Rotate Data Memory right through Carry

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

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

影响标志位 C

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

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

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

影响标志位 C

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

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

功能表示 ACC ← ACC – [m] – $\overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

SBCM A, [m] Subtract Data Memory from ACC with Carry and result in Data Memory

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

功能表示 [m] ← ACC – [m] – $\overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

SDZ [m] Skip if Decrement Data Memory is 0

指令说明 将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。

功能表示 [m] ← [m] -1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

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

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

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

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

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

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

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

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

SUB A, x Subtract immediate Data from ACC
指令说明 将累加器的内容减去立即数，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – x
影响标志位 OV、Z、AC、C、SC、CZ

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

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

SZ [m] Skip if Data Memory is 0

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

功能表示 如果 [m]=0, 跳过下一条指令执行

影响标志位 无

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

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

功能表示 ACC ← [m]，如果 [m]=0，跳过下一条指令执行

影响标志位 无

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

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

功能表示 如果 [m].i=0，跳过下一条指令执行

影响标志位 无

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

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

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

影响标志位 无

TABRDC [m] Read table (current 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 (当前页) 移至指定的数据存储器且将高字节移至 TBLH。

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

影响标志位 无

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

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

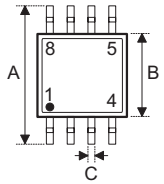
封装信息

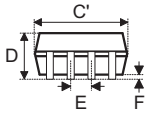
请注意，这里提供的封装信息仅作为参考。由于这个信息经常更新，提醒用户咨询 Holtek 网站 以获取最新版本的封装信息。

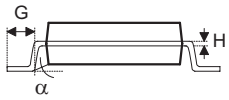
封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

- 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

8-pin SOP (150mil) 外形尺寸



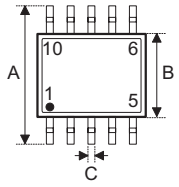


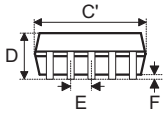


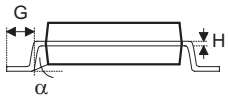
符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C′	—	0.193 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

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′	—	4.90 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°

10-pin SOP (150mil) 外形尺寸



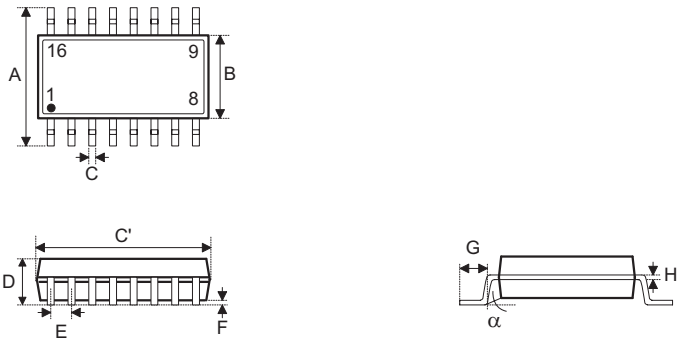




符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.018
C′	—	0.193 BSC	—
D	—	—	0.069
E	—	0.039 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

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	0.30	—	0.45
C′	—	4.90 BSC	—
D	—	—	1.75
E	—	1.00 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

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°