



触控型 Flash 单片机
BS83B04C

版本 : V1.10 日期 : 2019-11-08

www.holtek.com

目录

特性	6
CPU 特性	6
周边特性	6
概述	7
方框图	7
引脚图	8
引脚说明	8
极限参数	10
直流电气特性	10
工作电压特性	10
待机电流特性	11
工作电流特性	11
交流电气特性	12
内部高速振荡器 – HIRC – 频率精度	12
内部低速振荡器电气特性 – LIRC	12
系统上电时间电气特性	13
输入 / 输出口电气特性	13
存储器特性	14
LVR 电气特性	14
上电复位特性	15
系统结构	15
时序和流水线结构	15
程序计数器	16
堆栈	16
算术逻辑单元 – ALU	17
Flash 程序存储器	18
结构	18
特殊向量	18
查表	18
查表范例	19
在线烧录 – ICP	19
片上调试 – OCDS	20
数据存储器	21
结构	21
通用数据存储器	21
特殊功能数据存储器	21
特殊功能寄存器	23
间接寻址寄存器 – IAR0, IAR1	23
存储器指针 – MP0, MP1	23
存储区指针 – BP	24

累加器 – ACC	24
程序计数器低字节寄存器 – PCL	24
表格寄存器 – TBLP, TBHP, TBLH	24
状态寄存器 – STATUS	25
EEPROM 数据存储器	27
EEPROM 数据存储器结构	27
EEPROM 寄存器	27
从 EEPROM 中读取数据	28
写数据到 EEPROM	28
写保护	29
EEPROM 写中断	29
编程注意事项	29
振荡器	30
振荡器概述	30
系统时钟配置	30
内部高速 RC 振荡器 – HIRC	31
内部 32kHz 振荡器 – LIRC	31
工作模式和系统时钟	31
系统时钟	31
系统工作模式	32
控制寄存器	33
工作模式切换	34
待机电流的注意事项	37
唤醒	37
看门狗定时器	38
看门狗定时器时钟源	38
看门狗定时器控制寄存器	38
看门狗定时器操作	39
复位和初始化	40
复位功能	40
复位初始状态	43
输入 / 输出端口	46
上拉电阻	46
PA 口唤醒	47
输入 / 输出端口控制寄存器	47
引脚共用功能	47
输入 / 输出引脚结构	49
编程注意事项	50
定时器模块 – TM	51
简介	51
TM 操作	51
TM 时钟源	51
TM 中断	51

TM 外部引脚	51
编程注意事项	52
简易型 TM – CTM.....	53
简易型 TM 操作	53
简易型 TM 寄存器介绍	53
简易型 TM 工作模式	57
触控按键功能	63
触控按键结构	63
触控按键寄存器定义	64
触控按键操作	72
触控按键中断	82
编程注意事项	82
I²C 接口.....	82
I ² C 接口操作	82
I ² C 寄存器	84
I ² C 总线通信	86
I ² C 总线起始信号	87
I ² C 从机地址	87
I ² C 总线读 / 写信号	87
I ² C 总线从机地址应答信号	88
I ² C 总线数据和应答信号	88
I ² C 超时控制	89
中断	91
中断寄存器	91
中断操作	94
外部中断	95
I ² C 中断	95
时基中断	95
EEPROM 中断	96
多功能中断	97
触控按键 TKRCOV 中断	97
触控按键模块 TKTH 中断	97
TM 中断	97
中断唤醒功能	97
编程注意事项	98
配置选项	98
应用电路	99
指令集	100
简介	100
指令周期	100
数据的传送	100
算术运算	100
逻辑和移位运算	100
分支和控制转换	101

位运算	101
查表运算	101
其它运算	101
指令集概要	102
惯例	102
指令定义	105
封装信息	117
8-pin SOP (150mil) 外形尺寸	118
10-pin DFN (3mm×3mm×0.75mm) 外形尺寸	119
10-pin MSOP 外形尺寸	120
16-pin NSOP (150mil) 外形尺寸	121

特性

CPU 特性

- 工作电压
 - ◆ $f_{\text{SYS}}=2/4/8\text{MHz}$: 1.8V~5.5V
- $V_{\text{DD}}=5\text{V}$, 系统时钟为 8MHz 时, 指令周期为 0.5 μs
- 提供暂停和唤醒功能, 以降低功耗
- 振荡器类型:
 - ◆ 内部高速 2/4/8MHz RC – HIRC
 - ◆ 内部低速 32kHz RC – LIRC
- 多种工作模式: 快速、低速、空闲和休眠
- 内部集成振荡器, 无需外部元件
- 所有指令都可在 1~2 个指令周期内完成
- 查表指令
- 63 条功能强大的指令系统
- 4 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器: 2K \times 16
- RAM 数据存储器: 128 \times 8
- 触控按键数据存储器: 16 \times 8
- True EEPROM 存储器: 32 \times 8
- 看门狗定时器功能
- 8 个双向 I/O 口
- 1 个引脚与外部中断口共用
- 1 个 10-bit 简易型定时器模块用于时间测量、比较匹配输出及 PWM 输出
- 1 个时基功能, 可提供固定时间的中断信号
- I²C 接口
- 低电压复位功能
- 4 个触控按键功能
- 封装类型: 8-pin SOP、10-pin DFN/MSOP

概述

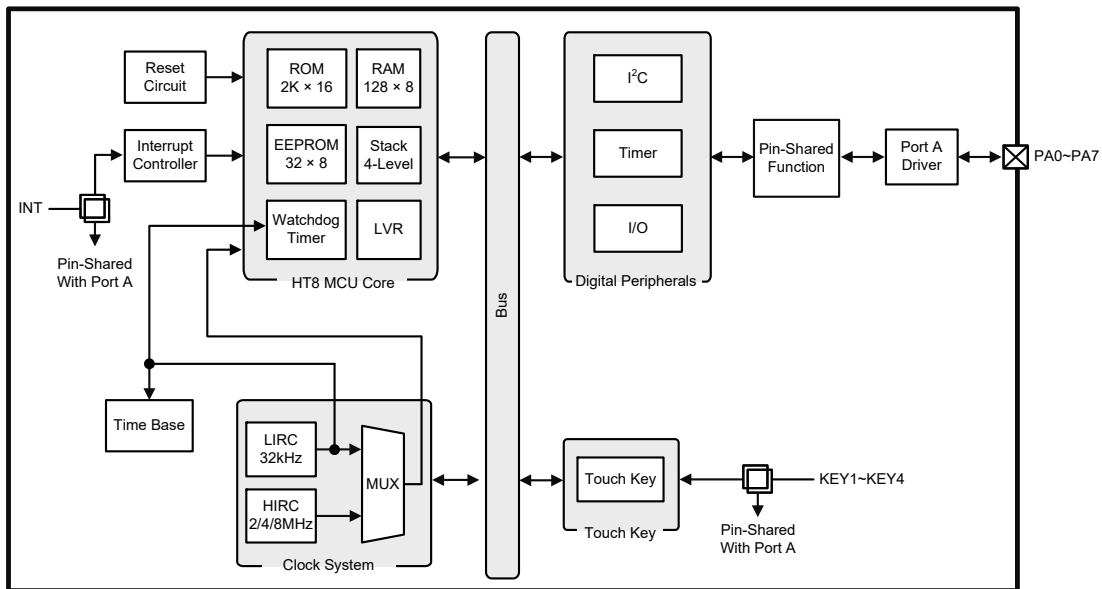
该单片机是一款具有 8 位高性能精简指令集的 Flash 单片机，具有完全集成的触控按键功能。其内置的触控按键功能以及 Flash 存储器可多次编程的特性给用户提供了可靠并易于实现的方式用于带触摸按键功能的应用产品的开发。

触摸按键功能完全集成于单片机内部，无需外部元件。存储器方面，除了 Flash 程序存储器，还包含了一个 RAM 数据存储器和一个可用于存储序列号、校准数据等非易失性数据的 True EEPROM 存储器。保护功能方面，包含内部看门狗定时器、低电压复位等特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地运行。

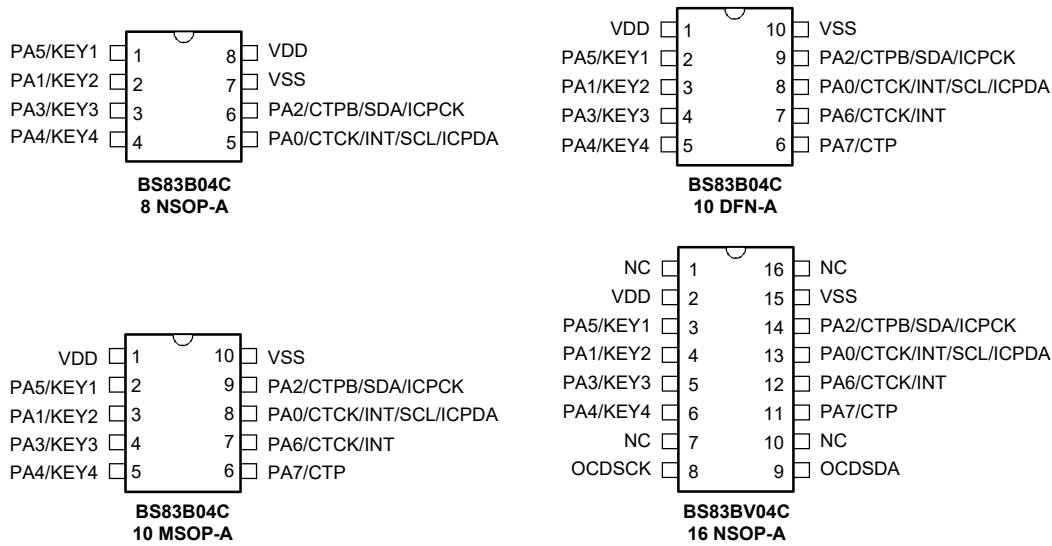
该单片机提供了内部低速和高速振荡器功能选项，且内建完整的系统振荡器，无需外接元件。其不同工作模式之间动态切换的能力，为用户提供了一个优化单片机操作和减少功耗的手段。内建完整的 PC 接口为设计者提供了一个易于与外部硬件通信的接口。外加 I/O 使用灵活、时基功能和定时器模块等其它特性，进一步增强了设备的功能性和灵活性。

该触控按键单片机可以广泛应用于现代触控产品中，例如电子测量仪器、家用电器、电子控制工具等。

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 OCSDSA 引脚为 OCDS 专用引脚。
 3. 在较小封装中可能含有未引出的引脚，需合理设置其状态以避免输入浮空造成额外耗电，详见“待机电流的注意事项”和“输入/输出端口”章节。

引脚说明

除了电源引脚和一些发送器控制相关引脚外，该单片机的所有引脚都以它们的端口名称进行标注，例如 PA0、PA1 等，用于描述这些引脚的数字输入/输出功能。然而，这些引脚也与其它功能共用，如触控按键功能，定时器模块引脚等。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。

该章节的引脚描述是针对最大封装的单片机，这些引脚并非都存在于小封装的单片机内。

引脚名称	功能	OPT	I/T	O/T	说明
PA0/CTCK/INT/ SCL/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
	CTCK	PAS0 IFS	ST	—	CTM 时钟输入
	INT	PAS0 IFS INTEG INTC0	ST	—	外部中断输入
	SCL	PAS0	ST	NMOS	I ² C 时钟线
	ICPDA	—	ST	CMOS	ICP 地址 / 数据

引脚名称	功能	OPT	I/T	O/T	说明
PA1/KEY2	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
	KEY2	PAS0 TKMC1	NSI	—	触控按键输入
PA2/CTPB/SDA/ 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
	CTPB	PAS0	—	CMOS	CTM 反相输出
	SDA	PAS0	ST	NMOS	PC 数据线
	ICPCK	—	ST	—	ICP 时钟
PA3/KEY3	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
	KEY3	PAS0 TKMC1	NSI	—	触控按键输入
PA4/KEY4	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
	KEY4	PAS1 TKMC1	NSI	—	触控按键输入
PA5/KEY1	PA5	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	KEY1	PAS1 TKMC1	NSI	—	触控按键输入
PA6/CTCK/INT	PA6	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	CTCK	IFS	ST	—	CTM 时钟输入
	INT	IFS INTEG INTC0	ST	—	外部中断输入
PA7/CTP	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
	CTP	PAS1	—	CMOS	CTM 输出
OCSDA	OCSDA	—	ST	CMOS	OCDS 地址 / 数据，仅用于 EV 芯片
OCDSCK	OCDSCK	—	ST	—	OCDS 时钟引脚，仅用于 EV 芯片
VDD	VDD	—	PWR	—	正电源
VSS	VSS	—	PWR	—	接地

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；
NSI：非标准输入。

极限参数

电源供应电压	$V_{SS}-0.3V \sim V_{SS}+6.0V$
输入电压	$V_{SS}-0.3V \sim V_{DD}+0.3V$
储存温度	$-50^{\circ}C \sim 125^{\circ}C$
工作温度	$-40^{\circ}C \sim 85^{\circ}C$
I_{OH} 总电流	-80mA
I_{OL} 总电流	80mA
总功耗	500mW

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

直流电气特性

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

工作电压特性

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

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 – HIRC	$f_{SYS}=f_{HIRC}=2MHz$	1.8	—	5.5	V
		$f_{SYS}=f_{HIRC}=4MHz$	1.8	—	5.5	
		$f_{SYS}=f_{HIRC}=8MHz$	1.8	—	5.5	
	工作电压 – LIRC	$f_{SYS}=f_{LIRC}=32kHz$	1.8	—	5.5	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 85°C	单位
		V _{DD}	条件					
I _{STB}	休眠模式	1.8V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3	3.6	
		5V		—	3	5	6	
	空闲模式 0 – LIRC	1.8V	f _{SUB} on	—	2.4	4	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	空闲模式 1 – HIRC	1.8V	f _{SUB} on, f _{SYS} =2MHz	—	72	100	120	μA
		3V		—	90	125	150	
		5V		—	200	300	360	
		1.8V	f _{SUB} on, f _{SYS} =4MHz	—	144	200	240	μA
		3V		—	180	250	300	
		5V		—	400	600	720	
1.8V	f _{SUB} on, f _{SYS} =8MHz	—	288	400	480	μA		
3V		—	360	500	600			
5V		—	600	800	960			

注：当使用该电气特性表格数据时，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. 无直流电流通路。
4. 所有待机电流数值都是在 HALT 指令执行后即所有指令执行停止时测得。

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	1.8V	f _{SYS} =32kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	快速模式 – HIRC	1.8V	f _{SYS} =2MHz	—	0.15	0.25	mA
		3V		—	0.2	0.3	
		5V		—	0.4	0.6	
		1.8V	f _{SYS} =4MHz	—	0.3	0.5	mA
		3V		—	0.4	0.6	
		5V		—	0.8	1.2	
	1.8V	f _{SYS} =8MHz	—	0.6	1.0	mA	
	3V		—	0.8	1.2		
	5V		—	1.6	2.4		

注：当使用该电气特性表格数据时，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. 无直流电流通路。
4. 所有工作电流数据都是通过执行连续 NOP 指令程序回路进行测量。

交流电气特性

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

内部高速振荡器 – HIRC – 频率精度

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

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	温度				
f _{HIRC}	通过烧录器调整后的 2MHz HIRC 频率	3V/5V	25°C	-1%	2	+1%	MHz
			-40°C~85°C	-4%	2	+4%	
		2.2V~5.5V	25°C	-6%	2	+6%	
			-40°C~85°C	-8%	2	+8%	
	通过烧录器调整后的 4MHz HIRC 频率	3V/5V	25°C	-1%	4	+1%	MHz
			-40°C~85°C	-2%	4	+2%	
		2.2V~5.5V	25°C	-2.5%	4	+2.5%	
			-40°C~85°C	-3%	4	+3%	
	通过烧录器调整后的 8MHz HIRC 频率	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	

- 注：1. 烧录器可在 3V/5V 这两个可选的固定电压下对 HIRC 频率进行调整，在此提供 V_{DD}=3V/5V 时的参数值。
2. 3V/5V 表格列下面提供的是全压条件下的参数值。当应用电压范围是 2.2V~3.6V 时，建议烧录器电压固定在 3V，而应用电压范围是 3.3V~5.5V 时，建议烧录器电压固定在 5V。
3. 表格中提供的最小和最大误差值仅在对应的烧录器调整频率下有效。当烧录器已将 HIRC 调整为某一固定频率，此后再通过程序中振荡器控制位将其频率改为其它值时，频率误差范围将增加到 ±20%。

内部低速振荡器电气特性 – LIRC

T_a=25°C, 除非另外规定

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	2.2V~5.5V	25°C	-10%	32	+10%	kHz
			-40°C~85°C	-50%	32	+60%	
t _{START}	LIRC 启动时间	—	—	—	—	100	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 ~f _H /64, f _H =f _{HIRC}	—	16	—	t _{HIRC}
		—	f _{sys} =f _{SUB} =f _{LIRC}	—	2	—	t _{LIRC}
	系统启动时间 (从 f _{sys} on 的状态下唤醒)	—	f _{sys} =f _H ~f _H /64, f _H =f _{HIRC}	—	2	—	t _H
		—	f _{sys} =f _{SUB} =f _{LIRC}	—	2	—	t _{SUB}
	系统速度切换时间 (快速模式 → 低速模式或 低速模式 → 快速模式)	—	f _{HIRC} off → on	—	16	—	t _{HIRC}
t _{RSTD}	系统复位延迟时间 (上电复位或 LVR 硬件复位)	—	RR _{POR} =5V/ms	42	48	54	ms
	系统复位延迟时间 (LVRC/WBTC/RSTC 软件复位)	—	—				
	系统复位延迟时间 (WDT 溢出)	—	—	14	16	18	ms
t _{SRESET}	最小软件复位时间	—	—	45	90	120	μs

- 注：1. 系统启动时间里提到的 f_{sys} on/off 状态取决于工作模式类型以及所选的系统时钟振荡器。更多相关细节请参考系统工作模式章节。
2. t_{HIRC} 等时间单位，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 口低电平输入电压	5V	—	0	—	1.5	V
		—		0	—	0.2V _{DD}	
V _{IH}	I/O 口高电平输入电压	5V	—	3.5	—	5.0	V
		—		0.8V _{DD}	—	V _{DD}	
I _{OL}	I/O 口灌电流	1.8V	V _{OL} =0.1V _{DD}	8	16	—	mA
		3V		16	32	—	
		5V		32	65	—	
I _{OH}	I/O 口源电流	1.8V	V _{OH} =0.9V _{DD}	-2	-4	—	mA
		3V		-4	-8	—	
		5V		-8	-16	—	

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
R _{PH}	I/O 口上拉电阻 (注)	3V	LVPU=0	20	60	100	kΩ
		5V		10	30	50	
		3V	LVPU=1	6.67	15	23	
		5V		3.5	7.5	12	
I _{LEAK}	输入漏电流	5V	V _{IN} =V _{DD} OR V _{IN} =V _{SS}	—	—	±1	μA
t _{TCK}	TM TCK 时钟输入最小脉宽	—	—	0.3	—	—	μs
t _{INT}	中断引脚最小输入脉宽	—	—	10	—	—	μs

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, 用此电压值除此测得的电流值得到此上拉电阻阻值。

存储器特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{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 / 写周期时间 – Flash 程序存储器	—	—	—	2	3	ms
	写周期时间 – 数据 EEPROM 存储器	—	—	—	4	6	ms
I _{DDPGM}	V _{DD} 电压下烧录 / 擦除电流	—	—	—	—	5.0	mA
E _P	电容耐久性 – Flash 程序存储器	—	—	10K	—	—	E/W
	电容耐久性 – 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	—	单片机处于休眠模式	1.0	—	—	V

LVR 电气特性

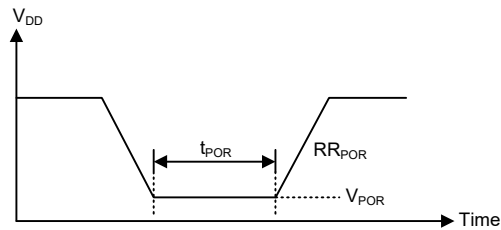
Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{LVR}	低电压复位电压	—	LVR 使能, 电压选择 1.7V	-5%	1.7	+5%	V
			LVR 使能, 电压选择 1.9V	-5%	1.9	+5%	
			LVR 使能, 电压选择 2.55V	-3%	2.55	+3%	
			LVR 使能, 电压选择 3.15V	-3%	3.15	+3%	
			LVR 使能, 电压选择 3.8V	-3%	3.8	+3%	
I _{LVR}	使用 LVR 的额外电流	5V	—	—	—	8	μA
t _{LVR}	复位最小低电压保持时间	—	—	120	240	480	μs

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



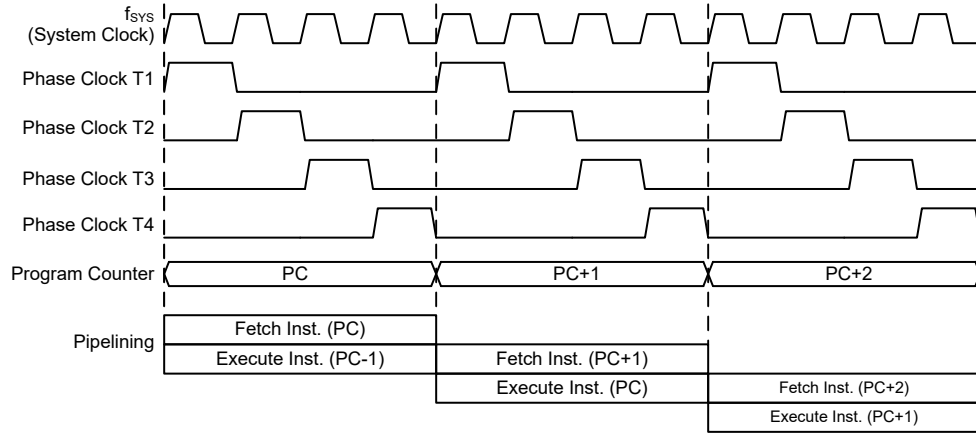
系统结构

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、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性和灵活性的 I/O 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。

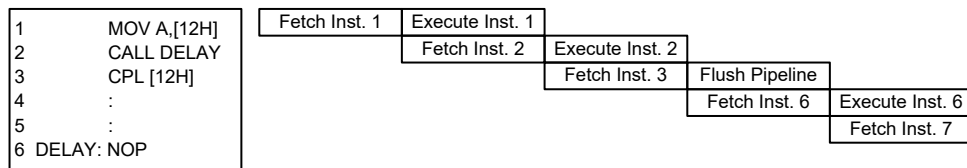
时序和流水线结构

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。

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
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 的使用可能引起程序跳转，因此需要额外的指令周期。

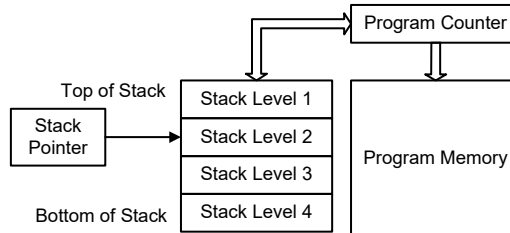
堆栈

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

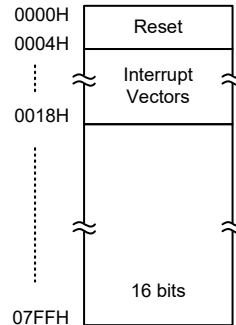
- 算术运算：
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- 逻辑运算：
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- 移位运算：
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- 递增和递减：
INCA, INC, DECA, DEC
- 分支判断：
JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

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

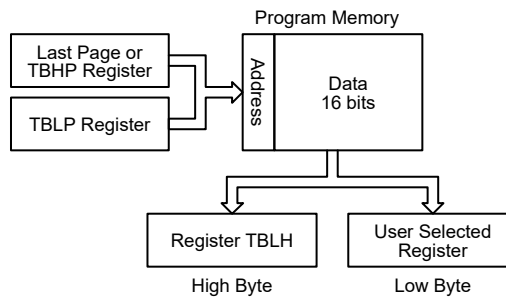
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 0000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

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，表格数据可以使用“TABRD [m]”或“TABRDL [m]”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”指向的地址是单片机 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 TBHP 和 TBLP 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]”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，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

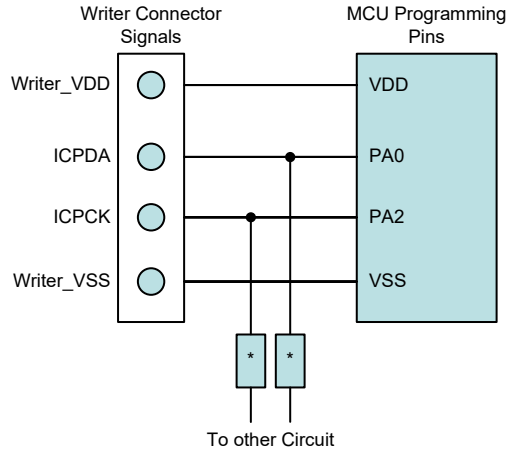
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。

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	串行时钟
VDD	VDD	电源
VSS	VSS	地

芯片内部程序存储器和 EEPROM 数据存储都可以通过 4 线的接口在线进行烧录。其中一条用于数据串行下载或上传，一条用于串行时钟，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 ICPCCK 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 BS83BV04C 用于 BS83B04C 单片机仿真。此 EV 芯片提供片上调试功能 (OCDS—On-Chip Debug Support) 用于开发过程中的单片机调试。除了片上调试功能和封装类型方面，EV 芯片和实际单片机在功能上几乎是兼容的。用户可将 OCSDA 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。OCSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。关于 OCDS 功能的详细描述，请参考“Holtek e-Link for 8-bit MCU OCDS 使用手册”文件。

Holtek e-Link 引脚名称	EV 芯片引脚名称	功能
OCSDA	OCSDA	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

数据存储是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。

结构

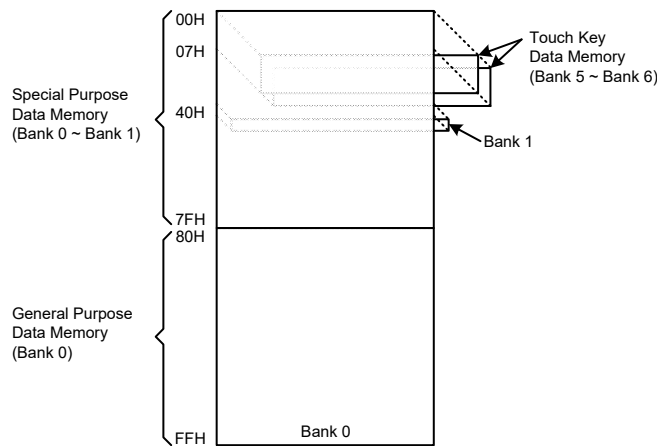
数据存储分为两个部分，第一部分是特殊功能数据存储，其中包括。这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二部分数据存储是做一般用途使用，都可在程序控制下进行读取和写入。

另一部分是为触控按键数据存储预留的。

总的存储器被分为多个 Bank。特殊功能数据存储寄存器均可在 Bank 0 被访问，处于“40H”地址的 EEC 寄存器却只能在 Bank 1 中被访问到。触控按键数据存储位于 Bank 5 和 Bank 6 中。切换不同区域可通过设置区域指针 (BP) 实现。单片机数据存储的起始地址是“00H”。

特殊功能数据存储	通用数据存储		触控按键数据存储	
有效 Banks	容量	Banks	容量	Banks
0, 1	128×8	Bank 0: 80H~FFH	16×8	Bank 5: 00H~07H Bank 6: 00H~07H

数据存储概要



数据存储结构

通用数据存储

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。这个数据存储区可让使用者进行读取和写入的操作。使用位操作指令可对个别的位做置位或复位的操作，较大地方便了用户在数据存储区内进行位操作。

特殊功能数据存储

这个区域的数据存储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	Bank 1		Bank 0	Bank 1
00H	IAR0		40H		EEC
01H	MP0		41H	EEA	
02H	IAR1		42H	EED	
03H	MP1		43H	TKTMR	
04H	BP		44H	TKC0	
05H	ACC		45H	TK16DL	
06H	PCL		46H	TK16DH	
07H	TBLP		47H	TKC1	
08H	TBLH		48H	TKM16DL	
09H	TBHP		49H	TKM16DH	
0AH	STATUS		4AH	TKMROL	
0BH	SCC		4BH	TKMROH	
0CH	HIRCC		4CH	TKMC0	
0DH	INTEG		4DH	TKMC1	
0EH	INTC0		4EH	TKMC2	
0FH	RSTFC		4FH	TKC2	
10H	IFS		50H	TK1MTH16L	
11H	INTC1		51H	TK1MTH16H	
12H	LVPUC		52H	TK2MTH16L	
13H	LVRC		53H	TK2MTH16H	
14H	PA		54H	TK3MTH16L	
15H	PAC		55H	TK3MTH16H	
16H	PAPU		56H	TK4MTH16L	
17H	PAWU		57H	TK4MTH16H	
18H	MFIO		58H	TKMTHS	
19H	MF11		59H		
1AH	WDTC		5AH		
1BH	TBC		5BH		
1CH	PSCR		5CH		
1DH	PAS0		5DH		
1EH	PAS1		5EH		
1FH			5FH		
20H			60H		
21H			61H		
22H			62H		
23H			63H		
24H			64H		
25H			65H		
26H			66H		
27H			67H		
28H			68H		
29H			69H		
2AH			6AH		
2BH			6BH		
2CH			6CH		
2DH			6DH		
2EH			6EH		
2FH			6FH		
30H	CTMC0		70H		
31H	CTMC1		71H		
32H	CTMDL		72H		
33H	CTMDH		73H		
34H	CTMAL		74H		
35H	CTMAH		75H		
36H			76H		
37H			77H		
38H			78H		
39H			79H		
3AH	IICC0		7AH		
3BH	IICC1		7BH		
3CH	IICD		7CH		
3DH	IICA		7DH		
3EH	IICTOC		7EH		
3FH	RSTC		7FH		

□: Unused, read as 00H.

特殊功能数据存储结构

特殊功能寄存器

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

间接寻址寄存器 – IAR0, IAR1

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 只可以访问 Bank 0，而 IAR1 和 MP1 可以访问所有 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区中能像普通的寄存器一般被操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由存储器指针所指定的地址。MP0、IAR0 用于访问 Bank 0，而 MP1 和 IAR1 可通过 BP 寄存器访问所有的 Bank。直接寻址仅可以用在 Bank 0 中，所有 Bank 都可使用 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。

存储区指针 – BP

该单片机数据存储器被分为多个 Bank。可以通过设置存储区指针 (Bank Pointer) 值来访问不同的数据存储器。

复位后，数据存储器会初始化到 Bank 0，但是在休眠模式或空闲模式下的 WDT 溢出复位，选择的数据存储器 Bank 不会改变。数据存储器的直接寻址总是访问 Bank 0，不影响存储区指针的值。要访问 Bank 0 之外的存储区，则必须要使用间接寻址方式。

• BP 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	DMBP2	DMBP1	DMBP0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 未定义，读为“0”

Bit 2~0 **DMBP2~DMBP0**: 数据存储器 Bank 选择位

000: Bank 0

001: Bank 1

010: 未定义

011: 未定义

100: 未定义

101: Bank 5

110: Bank 6

111: 未定义

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。它们的值必须在任何表格读取指令执行前加以设定，由于它们的值可以被如“INC”或“DEC”的指令所改变，这就提供了一种简单的方法对表格数据进行读取。表格读取数据指令执行之后，表格数据高字节存储在 TBLH 中。其中要注意的是，表格数据低字节会被传送到使用者指定的地址。

状态寄存器 – STATUS

这 8 位的状态寄存器由零标志位 (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 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
“C”位也受循环移位指令的影响。

EEPROM 数据存储

此单片机的一个特性是内建 EEPROM 数据存储。由于其非易失的存储结构，即使在电源掉电的情况下存储器内的数据仍然保存完好。这种存储区扩展了数据存储空间，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 数据存储结构

该单片机的 EEPROM 数据存储容量为 32×8 位。由于映射方式与程序存储器和数据存储器不同，因此不能像其它类型的存储器一样寻址。使用 Bank 0 中的一个地址和数据寄存器以及 Bank 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 Bank 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 Bank 1 中，不能被直接访问，仅能通过 MP1 和 IAR1 进行间接读取或写入。由于 EEC 控制寄存器位于 Bank 1 中的“40H”，在 EEC 寄存器上的任何操作被执行前，MP1 必须先设为“40H”，BP 设为“01”。

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	—	—	—	—	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 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 Bit 7~Bit 0

● EEC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	WREN	WR	RDEN	RD
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 **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: 写周期有效

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: 读周期有效

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：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”。

从 EEPROM 中读取数据

从 EEPROM 中读取数据，EEC 寄存器中的读使能位 RDEN 先置为高以使能读功能，EEPROM 中读取数据的地址要先放入 EEA 寄存器中。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，写入的数据需存入 EED 寄存器中。EEC 寄存器中的写使能位 WREN 先置为高以使能写功能，然后 EEC 寄存器中的 WR 位需立即置高以开始写操作，这两条指令必须连续执行。总中断位 EMI 在写周期开始前应当被清零，写周期开始后再将其使能。若 WR 位已置为高而 WREN 位还未被设置则不能开始写操作。由于控制 EEPROM 写周期是一个内部时钟，与单片机的系统时钟异步，所以数据写入 EEPROM 的时间将有所延迟。可通过轮询 EEC 寄存器中的 WR 位或判断 EEPROM 写中断以侦测写周期是否完成。若写周期完成，WR 位将自动清除为“0”，通知用户数据已写入 EEPROM。因此，应用程序将轮询 WR 位以确定写周期是否结束。

写保护

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 BP 将重置为 0，这意味着数据存储区 Bank 0 被选中。由于 EEPROM 控制寄存器位于 Bank 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 写中断

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 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 中断请求标志位 DEF 会自动清零。EMI 位也会被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 BP 也可以正常清零以阻止进入 EEPROM 控制寄存器存在的 Bank 1。尽管没有必要，写一个简单的读回程序以检查新写入的数据是否正确还是应该考虑的。

WREN 位置位后，EEC 寄存器中的 WR 位需立即置位，以确保写周期正确地执行。写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，在 EEPROM 读或写操作完全完成之前单片机不能进入空闲或休眠模式，否则会导致 EEPROM 读或写操作失败。

程序范例

从 EEPROM 中读取数据 — 轮询法

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, 040H              ; setup memory pointer MP1
MOV MP1, A               ; MP1 points to EEC register
MOV A, 01H               ; setup Bank Pointer
MOV BP, 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/write
CLR BP
MOV A, EED                ; move read data to register
MOV READ_DATA, A
```

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, 040H              ; setup memory pointer MP1
MOV MP1, A               ; MP1 points to EEC register
MOV A, 01H               ; setup Bank Pointer
MOV BP, 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 IAR1             ; disable EEPROM read/write
CLR BP
    
```

振荡器

不同的振荡器选择可以让使用者在不同的应用需求中实现更大范围的功能。振荡器的灵活性使得在速度和功耗方面可以达到较佳的优化。振荡器选择和操作是通过配置选项和寄存器共同完成的。

振荡器概述

振荡器除了作为系统时钟源，还作为看门狗定时器和时基功能的时钟源。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。

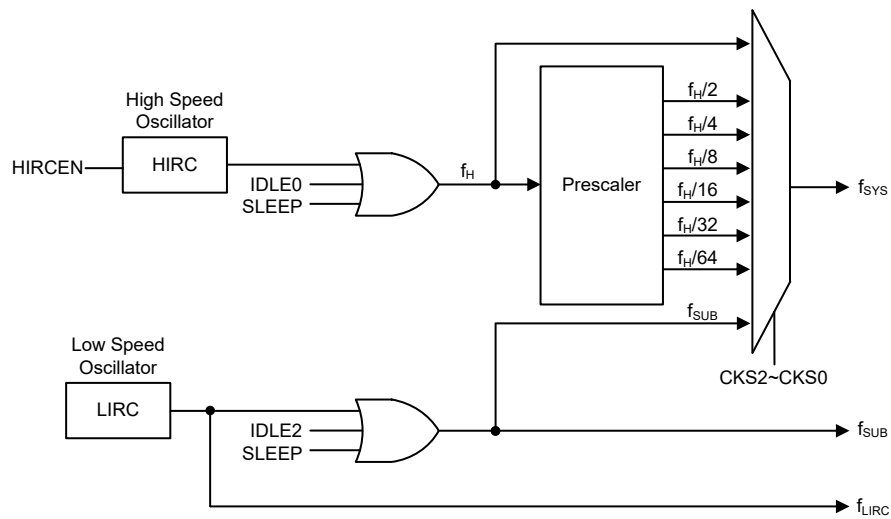
类型	名称	频率
内部高速 RC	HIRC	2/4/8MHz
内部低速 RC	LIRC	32kHz

振荡器类型

系统时钟配置

该单片机有两个系统振荡器，包括一个高速振荡器和一个低速振荡器。高速振荡器为内部 2/4/8MHz RC 振荡器，HIRC。低速振荡器为内部 32kHz 振荡器，LIRC。

低速或高速系统时钟频率由 SCC 寄存器中的 CKS2~CKS0 位决定的。



系统时钟配置

内部高速 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，不需其它外部器件。内部 RC 振荡器具有三种固定的频率：2MHz、4MHz 和 8MHz，可通过配置选项选择。HIRCC 寄存器的 HIRCS1~HIRCS0 位必须先设置来匹配配置选项所选择的频率。设置 HIRCS1~HIRCS0 位是有必要的，以确保交流特性中指定的 HIRC 频率精确度。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较大程度地降低。

内部 32kHz 振荡器 – LIRC

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

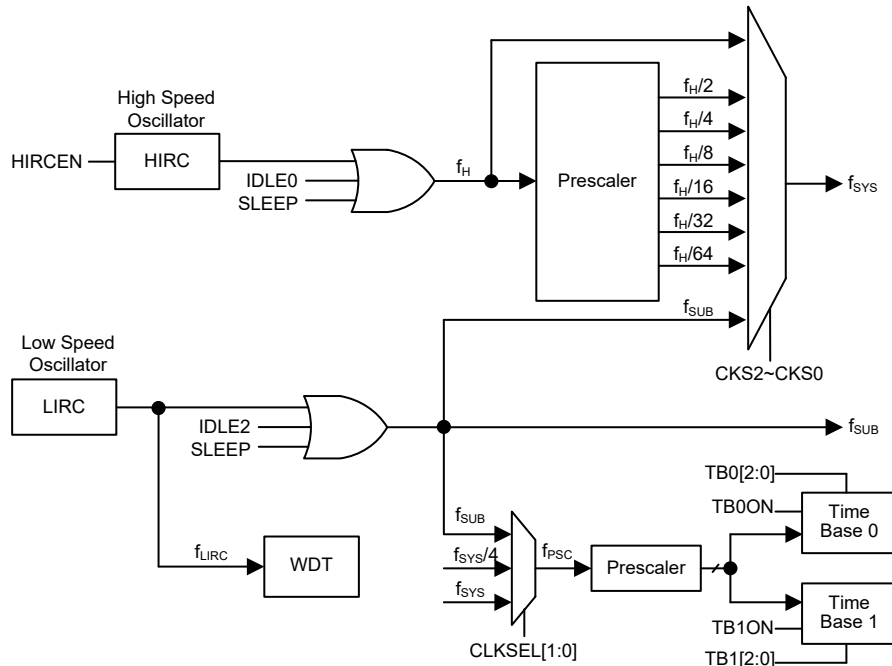
工作模式和系统时钟

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

系统时钟

单片机为 CPU 和外围功能操作提供了两种不同的时钟源。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 f_{SUB} ，若 f_{SUB} 被选择，低频时钟来自 LIRC 振荡器。其它系统时钟还有高速系统振荡器的分频 $f_H/2 \sim f_H/64$ 。



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 $f_H \sim f_H/64$ 频率的时钟源。

系统工作模式

单片机有 6 种不同的工作模式，每种有它自身的特性，根据应用中不同的性能和功耗要求可选择不同的工作模式。单片机正常工作有两种模式：快速模式和低速模式。剩余的 4 种工作模式：休眠模式、空闲模式 0、空闲模式 1 和空闲模式 2 用于单片机 CPU 关闭时以节省耗电。

工作模式	CPU	寄存器设置			f_{SYS}	f_H	f_{SUB}	f_{LIRC}
		FHIDEN	FSIDEN	CKS2~CKS0				
快速模式	On	x	x	000~110	$f_H \sim f_H/64$	On	On	On
低速模式	On	x	x	111	f_{SUB}	On/Off ⁽¹⁾	On	On
空闲模式 0	Off	0	1	000~110	Off	Off	On	On
				111	On			
空闲模式 1	Off	1	1	xxx	On	On	On	On
空闲模式 2	Off	1	0	000~110	On	On	Off	On
				111	Off			
休眠模式	Off	0	0	xxx	Off	Off	Off	On ⁽²⁾

“x”：无关

注：1. 在低速模式中， f_H 开启或关闭由相应的振荡器使能位控制。

2. 在休眠模式中，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。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。单片机在此模式中运行所耗工作电流较低。

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	—	—	—	—	HIRCS1	HIRCS0	HIRCF	HIRCEN

系统工作模式控制寄存器列表

• SCC 寄存器

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

Bit 7~5 **CKS2~CKS0**: 系统时钟选择位

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

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

Bit 4~2 未定义，读为“0”

Bit 1 **FHIDEN**: CPU 关闭时高频振荡器控制位

0: 除能
 1: 使能

此位用来控制在 CPU 执行 HALT 指令关闭后高速振荡器是被激活还是停止。

Bit 0 **FSIDEN**: CPU 关闭时低频振荡器控制位

0: 除能
 1: 使能

此位用来控制在 CPU 执行 HALT 指令关闭后低速振荡器是被激活还是停止。

• HIRCC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HIRCS1	HIRCS0	HIRCF	HIRCEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	1

Bit 7~4 未定义，读为“0”

Bit 3~2 **HIRCS1~HIRCS0**: HIRC 频率选择位

00: 2MHz

01: 4MHz

10: 8MHz

11: 2MHz

当 HIRC 振荡器使能或通过应用程序改变 HIRC 频率选择位时，在 HIRCF 标志位置高后时钟频率会自动改变。建议由这两个位选择的 HIRC 频率应该与配置选项所选择的频率相同，以确保交流特性中指定的 HIRC 频率精确度。

Bit 1 **HIRCF**: HIRC 振荡器稳定标志位

0: HIRC 未稳定

1: HIRC 稳定

此位用于表明 HIRC 振荡器是否稳定。HIRCEN 位置高使能 HIRC 振荡器或通过应用程序改变 HIRC 频率选择位，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

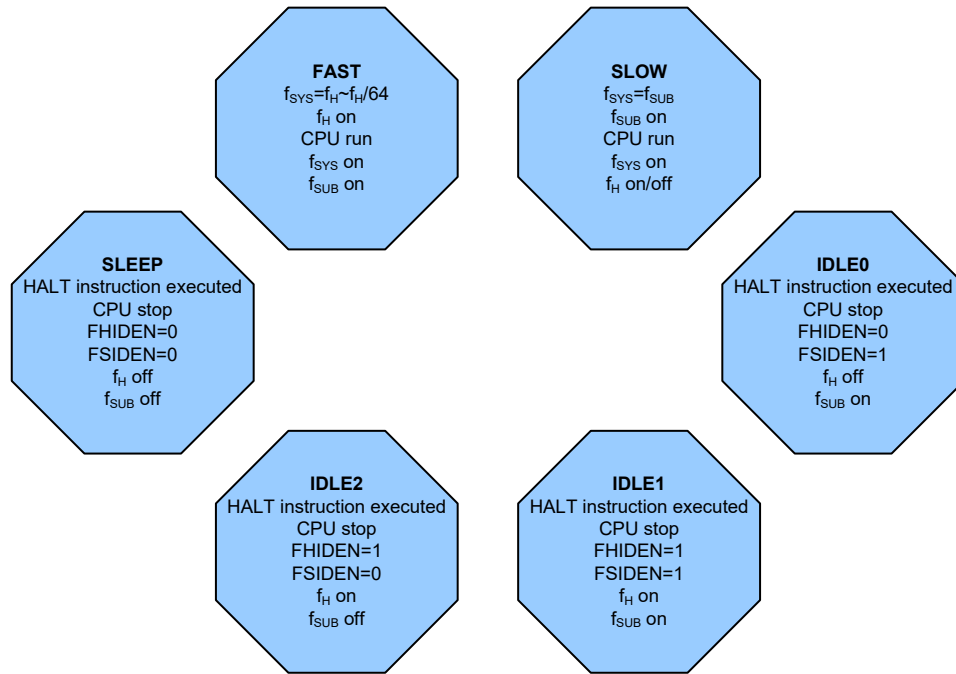
0: 除能

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。

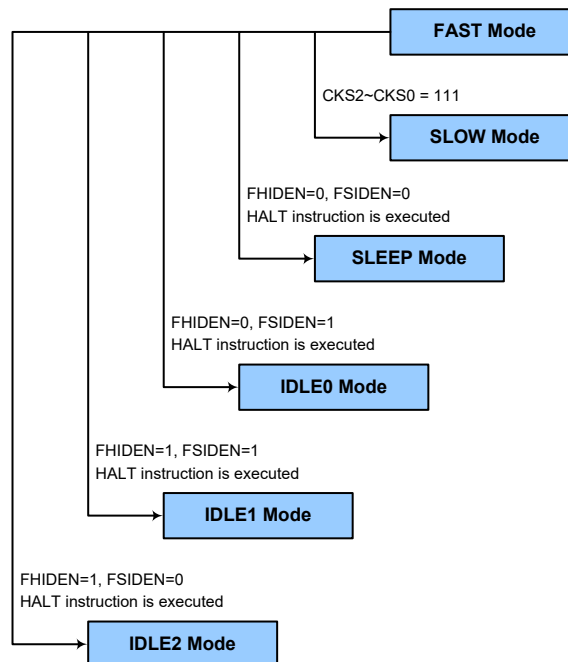
简单来说，快速模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



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。

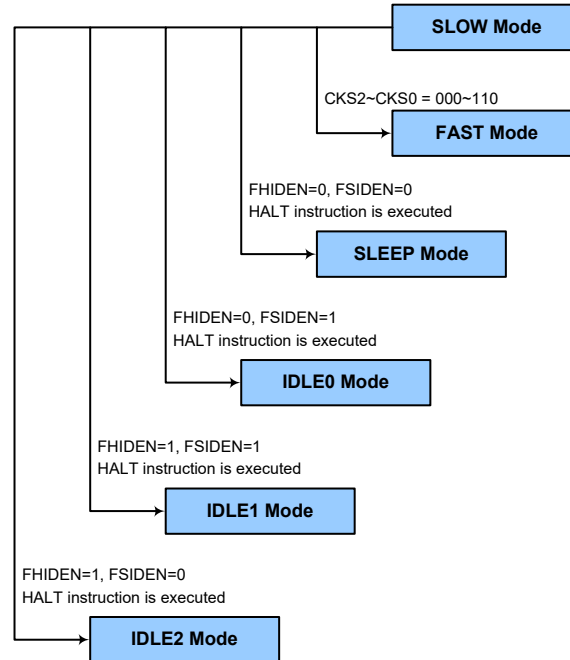
低速模式的时钟源来自 LIRC 振荡器，因此要求这些振荡器在所有模式切换动作发生前稳定下来。



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

然而，如果在低速模式下 f_H 因未使用而关闭，那么从低速模式切换到快速模式时，它需要一定的时间来重新起振和稳定，可通过检测 HIRCC 寄存器中的 HIRCF 位进行判断，所需的高速系统振荡器稳定时间在系统上电时间电气特性中有说明。



进入休眠模式

进入休眠模式的方法仅有一种——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。
- 数据存储器和寄存器的内容和寄存器将保持当前值。
- 输入 / 输出将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 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。
- 数据存储器和寄存器的内容和寄存器将保持当前值。
- 输入 / 输出将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 因 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，可能到只有几个微安的级别（空闲模式 1 和空闲模式 2 除外），所以如果要将电路的电流进一步降低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入 / 输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。

另外还需注意单片机设为输出的 I/O 引脚上的负载。应将它们设置在有最小拉电流的状态或将它们和其它的 CMOS 输入一样接到没有拉电流的外部电路上。还应注意的是，如果选择 LIRC 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，额外的静态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，系统进入休眠或空闲模式，PDF 将被置位。系统上电或执行清除看门狗的指令，PDF 将被清零。若系统由看门狗定时器溢出唤醒，会发生看门狗定时器复位，TO 将被置位。看门狗定时器溢出将会置位 TO 标志并唤醒系统，这种复位会重置程序计数器和堆栈指针，其它标志保持原有状态。PA 口中的每个引脚都可以通过 PAWU 寄存器使能下降沿唤醒功能。PA 端口唤醒后，程序将在“HALT”指令后继续执行。如果系统是通过中断唤醒，则有两种可能发生。第一种情况是：相关中断除能或是中断使能且堆栈已满，则程序会在“HALT”指令之后继续执行。这种情况下，唤醒系统的中断会等到相关中断使能或有堆栈层可以使用之后才执行。第二种情况是：相关中断使能且堆栈未满，则中断可以马上执行。如果在进入休眠或空闲模式之前中断标志位已经被设置为“1”，则相关中断的唤醒功能将无效。

看门狗定时器

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

看门狗定时器时钟源

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

看门狗定时器控制寄存器

WDTC 寄存器用于控制 WDT 功能的使能、选择溢出周期以及软件复位单片机。

- **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 软件控制位

01010/10101: 使能
其它值: 单片机复位

如果因外部环境干扰使这些位发生改变，则单片机将在一段延迟时间 t_{SRESET} 后复位，且在复位后 RSTFC 寄存器中的 WRF 标志位会被置位。

Bit 2~0 **WS2~WS0**: WDT 溢出周期选择位

000: $2^8/f_{LIRC}$
001: $2^{10}/f_{LIRC}$
010: $2^{12}/f_{LIRC}$
011: $2^{14}/f_{LIRC}$
100: $2^{15}/f_{LIRC}$
101: $2^{16}/f_{LIRC}$
110: $2^{17}/f_{LIRC}$
111: $2^{18}/f_{LIRC}$

这三位控制 WDT 时钟源的分频比，从而实现了对 WDT 溢出周期的控制。

● RSTFC 寄存器

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

“x”：未知

- Bit 7~4 未定义，读为“0”
- Bit 3 **RSTF**: 复位控制寄存器软件复位标志位
详见别处的描述。
- Bit 2 **LVRF**: LVR 复位标志位
详见别处的描述。
- Bit 1 **LRF**: LVR 控制寄存器软件复位标志位
详见别处的描述。
- 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除看门狗定时器以防止其产生复位，可使用清除看门狗指令实现。无论什么原因，程序失常跳转到一个未知的地址或进入一个死循环，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 WDTIC 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。当设置为“01010B”或“10101B”时使能 WDT 功能。如果 WE4~WE0 设置为除“01010B”和“10101B”以外的值时，单片机将在一段延迟时间 t_{SRESET} 后复位。上电后这些位初始化为“01010B”。

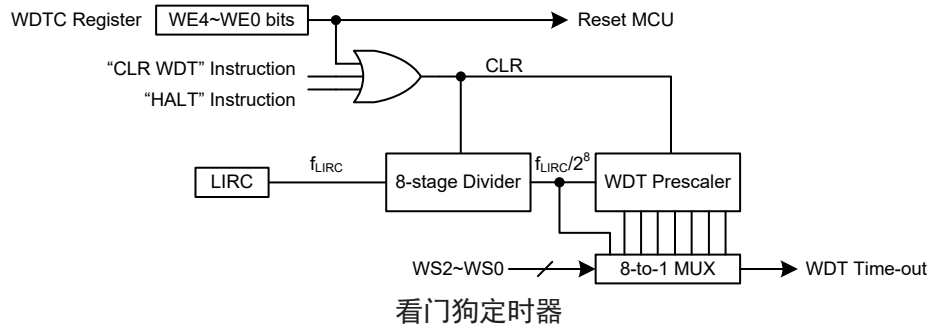
WE4~WE0 位	WDT 功能
01010B/10101B	使能
其它值	单片机复位

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 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，即写入除 01010B 和 10101B 外任何值到 WE4~WE0 位，第二种是通过软件清除指令，而第三种是通过“HALT”指令。

只有一种软件指令用于清除看门狗寄存器。因此只要执行“CLR WDT”便清除 WDT。

当设置分频比为 2^{18} 时，溢出周期最大。例如，时钟源为 32kHz LIRC 振荡器，分频比为 2^{18} 时最大溢出周期约 8s，分频比为 2^8 时最小溢出周期约 8ms。



复位和初始化

复位功能是在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。

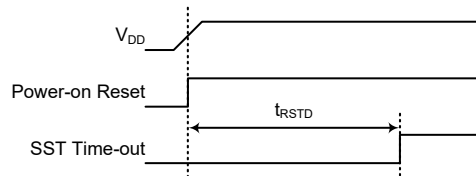
除了上电复位外，另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。还有一种复位为看门狗溢出单片机复位，不同方式的复位操作会对寄存器产生不同的影响。

复位功能

单片机的几种内部复位方式将在此处做具体介绍。

上电复位

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



上电复位时序图

内部复位控制

内部复位控制寄存器 RSTC 用于为单片机在受到环境噪声干扰而异常工作时提供复位。如果 RSTC 寄存器的内容被设置为除 01010101B 或 10101010B 以外的任何值，单片机会在一段延迟时间 t_{SRESET} 后发生复位。上电后寄存器的值为 01010101B。

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

内部复位功能控制

● RSTC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	RSTC7	RSTC6	RSTC5	RSTC4	RSTC3	RSTC2	RSTC1	RSTC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7~0 **RSTC7~RSTC0**: 复位功能控制位

01010101: 无操作

10101010: 无操作

其它值: 单片机复位

如果由于不利的环境因素使这些位发生改变, 单片机将复位。复位动作发生在一段延迟时间 t_{SRESET} 后, 且 RSTFC 寄存器的 RSTF 位将置为“1”。

● RSTFC 寄存器

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

“x”: 未知

Bit 7~4 未定义, 读为“0”

Bit 3 **RSTF**: 复位控制寄存器软件复位标志位

0: 未发生

1: 发生

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

Bit 2 **LVRF**: LVR 复位标志位

详见别处的描述。

Bit 1 **LRF**: LVR 控制寄存器软件复位标志位

详见别处的描述。

Bit 0 **WRF**: WDT 控制寄存器软件复位标志位

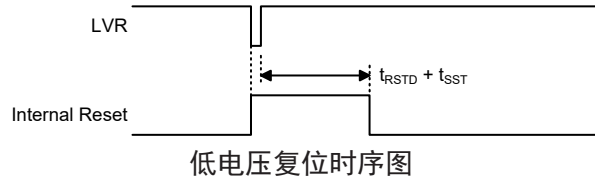
详见别处的描述。

低电压复位 – LVR

单片机具有低电压复位电路, 用来监测它的电源电压。当电源电压低于预设值时产生单片机复位。

通过 LVRC 控制寄存器可使能或除能 LVR 功能。若 LVRC 控制寄存器被配置为使能 LVR 功能, 除了单片机进入休眠 / 空闲模式, 其它情况 LVR 功能始终使能。例如在更换电池的情况下, 单片机供应的电压可能会在 $0.9V \sim V_{LVR}$ 之间, 这时 LVR 将会自动复位单片机且 RSTFC 寄存器中的 LVRF 标志位置位。LVR 包含以下的规格: 有效的 LVR 信号, 即在 $0.9V \sim V_{LVR}$ 的低电压状态的时间, 必须超过 LVR 电气特性中 t_{LVR} 参数的值。如果低电压存在不超过 t_{LVR} 参数的值, 则 LVR 将会忽略它且不会执行复位功能。 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，需经过一段延迟时间 t_{SRESET} 响应复位。此时 RSTFC 寄存器的 LRF 位被置位。上电后寄存器的值为 01100110B。LVR 会于休眠或空闲模式时自动除能关闭。



● LVRC 寄存器

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

Bit 7~0 **LVS7~LVS0**: LVR 电压选择

01100110: 1.7V
01010101: 1.9V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
11110000: LVR 除能

其它值: 单片机复位 – 寄存器复位为 POR 值

若低电压情况发生且满足以上定义的低电压复位值，则单片机复位。保持低电压状态的时间需超过 t_{LVR} 才能响应复位。此时复位后的寄存器内容保持不变。

除了以上定义的低电压复位值外，其它值也能导致单片机复位。需要经过一段延迟时间 t_{SRESET} 响应复位。但此时寄存器内容将复位为 POR 值。

● RSTFC 寄存器

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

“x”: 未知

Bit 7~4 未定义，读为“0”

Bit 3 **RSTF**: 复位控制寄存器软件复位标志位
详见别处的描述。

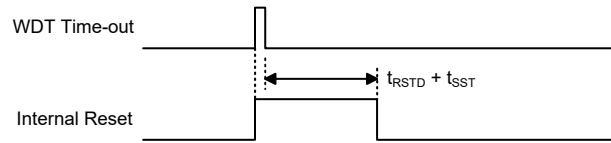
Bit 2 **LVRF**: LVR 复位标志位
0: 未发生
1: 发生
当特定的低电压复位条件发生时，此位被置为“1”，且只能通过应用程序清零。

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
详见别处的描述。

正常运行时看门狗溢出复位

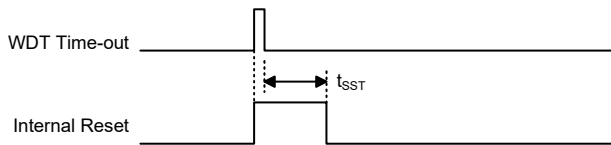
除了看门狗溢出标志位 TO 将被设为“1”之外，在快速模式或低速模式时看门狗溢出复位和 LVR 复位相同。



正常运行时看门狗溢出复位时序图

休眠或空闲时看门狗溢出复位

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



休眠或空闲时看门狗溢出复位时序图

复位初始状态

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

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

“u”代表不改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

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

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

寄存器	上电复位	LVR 复位 (正常操作)	WDT 溢出 (正常操作)	WDT 溢出 (空闲 / 休眠)
IAR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
IAR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	---- -000	---- -000	---- -000	---- -uuu
ACC	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	---- -xxx	---- -xxx	---- -uuu	---- -uuu
STATUS	--00 xxxx	--00 xxxx	--1u uuuu	--11 uuuu
SCC	000- --00	000- --00	000- --00	uuu- --uu
HIRC	---- 0001	---- 0001	---- 0001	---- uuuu
INTEG	---- --00	---- --00	---- --00	---- --uu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
RSTFC	---- 0x00	---- u1uu	---- uuuu	---- uuuu
IFS	---- --00	---- --00	---- --00	---- --uu
INTC1	-000 -000	-000 -000	-000 -000	-uuu -uuu
LVPUC	---- ---0	---- ---0	---- ---0	---- ---u
LVRC	0110 0110	0110 0110	0110 0110	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFIO	--00 --00	--00 --00	--00 --00	--uu --uu
MFII	--00 --00	--00 --00	--00 --00	--uu --uu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
TBC	0--- -000	0--- -000	0--- -000	u--- -uuu
PSCR	---- --00	---- --00	---- --00	---- --uu
PAS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	00-- 0000	00-- 0000	00-- 0000	uu-- uuuu
CTMC0	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMDH	---- --00	---- --00	---- --00	---- --uu
CTMAL	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMAH	---- --00	---- --00	---- --00	---- --uu
IICC0	---- 000-	---- 000-	---- 000-	---- uuu-
IICC1	1000 0001	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

寄存器	上电复位	LVR 复位 (正常操作)	WDT 溢出 (正常操作)	WDT 溢出 (空闲 / 休眠)
IICA	0000 000-	0000 000-	0000 000-	uuuu uuu-
IICTOC	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTC	0101 0101	0101 0101	0101 0101	uuuu uuuu
EEA	---0 0000	---0 0000	---0 0000	---u uu u
EED	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKTMR	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC0	0000 0010	0000 0010	0000 0010	uuuu uuuu
TK16DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK16DH	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC1	0000 0011	0000 0011	0000 0011	uuuu uuuu
TKM16DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM16DH	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKMROL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKMROH	---- --00	---- --00	---- --00	---- --uu
TKMC0	--00 0000	--00 0000	--00 0000	--uu uuuu
TKMC1	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKMC2	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKC2	---- --01	---- --01	---- --01	---- --uu
TK1MTH16L	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK1MTH16H	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK2MTH16L	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK2MTH16H	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK3MTH16L	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK3MTH16H	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK4MTH16L	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK4MTH16H	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKMTHS	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEC	---- 0000	---- 0000	---- 0000	---- 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。

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0

I/O 逻辑功能寄存器列表

上拉电阻

许多产品应用在端口处于输入状态时需要外加一个上拉电阻来实现上拉的功能。为了免去外部上拉电阻，当引脚规划为输入时，可由内部连接到一个上拉电阻。这些上拉电阻可通过相应的上拉控制寄存器 PAPU 和 LVPUC 来设置，它用一个 PMOS 晶体管来实现上拉电阻功能。寄存器 PAPU 用来决定上拉功能是否使能，而寄存器 LVPUC 则用来选择低电源电压应用所需的上拉电阻值。

需注意的是，当 I/O 引脚设为逻辑输入或 NMOS 输出时，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。

● PAPU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
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 **PAPU7~PAPU0**: PA7~PA0 上拉电阻控制位

0: 除能
1: 使能

● LVPUC 寄存器

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

Bit 7~1 未定义，读为“0”

Bit 0 **LVPU**: 低电压上拉电阻控制位

0: 所有引脚上拉电阻为 30kΩ @ 5V
1: 所有引脚上拉电阻为 7.5kΩ @ 5V

该位用来选择低电源电压应用的上拉电阻值，当通过设置相应的上拉电阻控制

位为高来使能相应的引脚上拉功能时，LVPU 才有效，当上拉功能除能时，该位不受影响。

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，此功能可由唤醒控制寄存器控制。

• PAWU 寄存器

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

Bit 7~0 **PAWU7~PAWU0**: PA7~PA0 唤醒功能控制位

0: 除能
1: 使能

输入 / 输出端口控制寄存器

每一个输入 / 输出口都具有各自的控制寄存器，即 PAC，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，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	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
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 **PAC7~PAC0**: PA7~PA0 引脚类型选择位

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一系列寄存器进行设定。

引脚共用功能选择寄存器

封装中有限的引脚个数会对某些单片机功能造成影响。然而，引脚功能共用和引脚功能选择，使得小封装单片机具有更多不同的功能。单片机包含端口“A”

输出功能选择寄存器“n”，记为 PASn，和输入功能选择寄存器，记为 IFS，这些寄存器可以用来选择多功能共用引脚上的特定功能。

当要使用引脚上的输入功能，对应的输入和输出功能要进行合理设定。例如，若要使用 I²C SDA 引脚，对应的输出引脚共用功能要通过寄存器 PASn 设置为 SDI/SDA 功能且需 SDA 输入源需通过 IFS 寄存器合理选择。如果要选择外部中断功能，相关的输出引脚共用功能应选择作为 I/O 功能，且也要选择。

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字段时，一些数字输入引脚如 INT、CTCK 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	PAS17	PAS16	—	—	PAS13	PAS12	PAS11	PAS10
IFS	—	—	—	—	—	—	IFS1	IFS0

引脚共用功能选择寄存器列表

● PAS0 寄存器

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

Bit 7~6 **PAS07~PAS06**: PA3 引脚共用功能选择
00/01/10: PA3
11: KEY3

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

Bit 3~2 **PAS03~PAS02**: PA1 引脚共用功能选择
00/01/10: PA1
11: KEY2

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

● PAS1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAS17	PAS16	—	—	PAS13	PAS12	PAS11	PAS10
R/W	R/W	R/W	—	—	R/W	R/W	R/W	R/W
POR	0	0	—	—	0	0	0	0

Bit 7~6 **PAS17~PAS16:** PA7 引脚共用功能选择
00/01/10: PA7
11: CTP

Bit 5~4 未定义, 读为“0”

Bit 3~2 **PAS13~PAS12:** PA5 引脚共用功能选择
00/01/10: PA5
11: KEY1

Bit 1~0 **PAS11~PAS10:** PA4 引脚共用功能选择
00/01/10: PA4
11: KEY4

● IFS 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	IFS1	IFS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

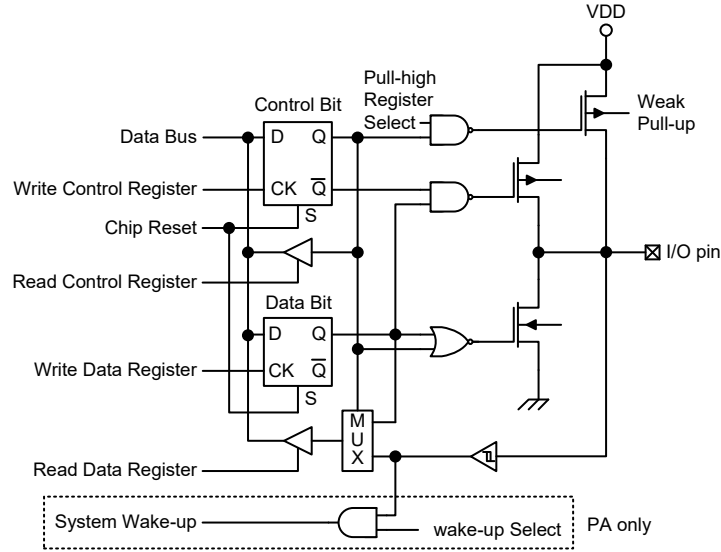
Bit 7~2 未定义, 读为“0”

Bit 1 **IFS1:** CTCK 输入源引脚选择
0: PA6
1: PA0

Bit 0 **IFS0:** INT 输入源引脚选择
0: PA0
1: PA6

输入 / 输出引脚结构

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



逻辑功能输入 / 输出端口结构

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。

定时器模块 – TM

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

这里只简单介绍简易型 TM 的基本特性，更多详细资料请参考简易型 TM 章节。

简介

该单片机包含一个简易型 TM 单元，即 CTM，其主要特性见下表。

功能	CTM
定时 / 计数器	√
比较匹配输出	√
PWM 通道数	1
PWM 对齐方式	边沿对齐
PWM 调节周期 & 占空比	占空比或周期

CTM 功能概要

TM 操作

简易型 TM 提供从简单的定时操作到 PWM 信号产生等多种功能。理解 TM 操作的关键是比较 TM 内独立运行的计数器的值与内部比较器的预置值。当计数器的值与比较器的预置值相同时，则比较匹配，TM 中断信号产生，清零计数器并改变 TM 输出引脚的状态。用户选择内部时钟或外部引脚来驱动内部 TM 计数器。

TM 时钟源

驱动 TM 计数器的时钟源很多。通过设置 CTM 控制寄存器的 CTCK2~CTCK0 位可选择所需的时钟源。该时钟源来自系统时钟 f_{SYS} 或内部高速时钟 f_H 或 f_{SUB} 时钟源或外部 CTCK 引脚的分频比。CTCK 引脚时钟源用于允许外部信号作为 TM 时钟源或用于事件计数。

TM 中断

简易型 TM 有两个内部中断，即内部比较器 A 和比较器 P，当发生比较匹配时比较器将产生一个 TM 中断。中断产生时将清除计数器，并改变 TM 输出引脚的状态。

TM 外部引脚

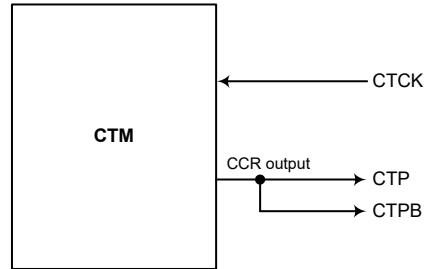
简易型 TM 有一个输入引脚 CTCK。CTM 输入引脚也作为 CTM 的时钟源输入脚，通过设置 CTMC0 寄存器中的 CTCK2~CTCK0 位进行选择。外部时钟源可通过该引脚来驱动内部 TM。CTM 输入引脚可以选择上升沿或下降沿。

简易型 TM 有两个输出引脚 CTP 和 CTPB。当 CTM 工作在比较匹配输出模式且比较匹配发生时，这些引脚会由 TM 控制切换到高电平或低电平或翻转。外部 CTP 和 CTPB 输出引脚也被 TM 用来产生 PWM 输出波形。

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

CTM	
输入	输出
CTCK	CTP, CTPB

CTM 外部引脚

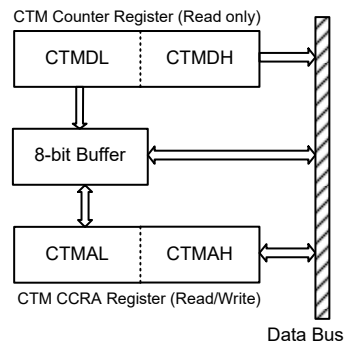


CTM 功能引脚控制方框图

编程注意事项

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

由于 CCRA 寄存器按照下图方式执行，具体存取这些寄存器对的方式如上所述，建议使用“MOV”指令，通过以下步骤访问 CCRA 低字节寄存器 CTMAL。若不采用以下步骤访问 CCRA 低字节寄存器将导致不可预期的结果。

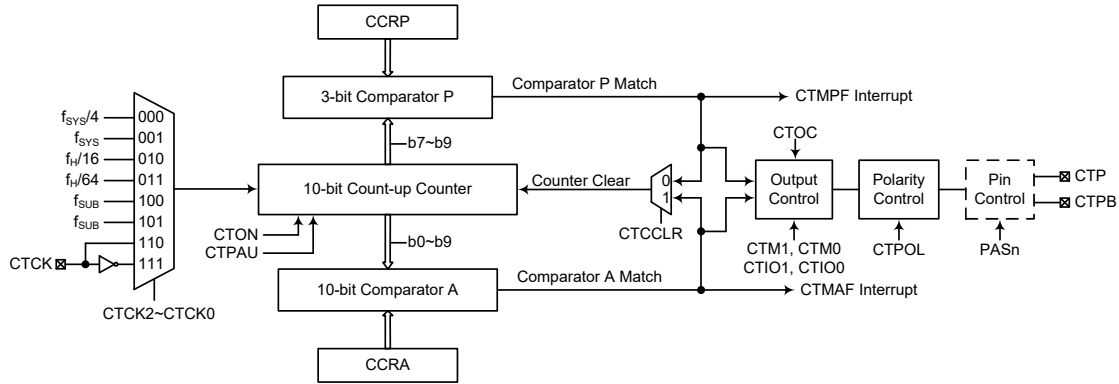


读写流程如下步骤所示：

- 写数据至 CCRA
 - ◆ 步骤 1. 写数据至低字节寄存器 CTMAL
 - 注意，此时数据仅写入 8-bit 缓存器。
 - ◆ 步骤 2. 写数据至高字节寄存器 CTMAH
 - 注意，此时数据直接写入高字节寄存器，同时锁存在 8-bit 缓存器中的数据写入低字节寄存器。
- 从计数器寄存器和 CCRA 中读取数据
 - ◆ 步骤 1. 从高字节寄存器 CTMDH 和 CTMAH 中读取数据
 - 注意，此时高字节寄存器中的数据直接读取，同时由低字节寄存器读取的数据锁存至 8-bit 缓存器中。
 - ◆ 步骤 2. 从低字节寄存器 CTMDL 和 CTMAL 中读取数据
 - 注意，此时读取 8-bit 缓存器中的数据。

简易型 TM – CTM

简易型 TM 包括三种工作模式，即比较匹配输出、定时 / 事件计数器和 PWM 输出模式。简易型 TM 也由一个外部输入脚控制并驱动两个外部输出脚。



注：CTPB 为 CTP 的反相输出。

简易型 TM 方框图

简易型 TM 操作

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

通过应用程序改变 10 位计数器值的唯一方法是使 CTON 位发生上升沿跳变清除计数器。此外，计数器溢出或比较匹配也会自动清除计数器。上述条件发生时，通常情况下会产生 TM 中断信号。简易型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，也可以控制两个输出脚。所有工作模式的设定都是通过设置相关内部寄存器来实现的。

简易型 TM 寄存器介绍

简易型 TM 的所有操作由一系列寄存器控制。包含一对只读寄存器用来存放 10 位计数器的值，一对读 / 写寄存器存放 10 位 CCRA 的值，剩下两个控制寄存器设置不同的操作和控制模式以及 3 位的 CCRP 值。

寄存器名称	位							
	7	6	5	4	3	2	1	0
CTMC0	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
CTMC1	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
CTMDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMDH	—	—	—	—	—	—	D9	D8
CTMAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMAH	—	—	—	—	—	—	D9	D8

10-bit 简易型 TM 寄存器列表

• CTMC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	CTPAU	CTCK2	CTCK1	CTCK0	CTON	CTRP2	CTRP1	CTRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CTPAU**: CTM 计数器暂停控制位

0: 运行
1: 暂停

通过设置此位为高可暂停计数器。此位清零可恢复正常计数器操作。在暂停情况下 CTM 将保持上电并继续产生功耗。当此位由低到高转换，计数器将保留其剩余值，当其电平再变为低时从此位开始继续计数。

Bit 6~4 **CTCK2~CTCK0**: CTM 计数器时钟选择位

000: $f_{SYS}/4$

001: f_{SYS}

010: $f_H/16$

011: $f_H/64$

100: f_{SUB}

101: f_{SUB}

110: CTCK 上升沿时钟

111: CTCK 下降沿时钟

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

Bit 3 **CTON**: CTM 计数器 On/Off 控制位

0: Off

1: On

此位控制 CTM 的总开关功能。设置此位为高可使能计数器使其运行，清零此位则除能 CTM。清零此位将停止计数器并关闭 CTM 减少功耗。当此位经由低到高转换时，内部计数器将复位清零，当此位由高到低转换时，内部计数器将保持其剩余值直到此位再次转变为高。

若 CTM 处于比较匹配输出模式或 PWM 输出模式，当 CTON 位经由低到高转换时，CTM 输出脚将复位至 CTC0 位指定的初始值。

Bit 2~0 **CTRP2~CTRP0**: CTM CCRP 3-bit 寄存器，与 CTM 计数器 bit 9~bit 7 比较

比较器 P 匹配周期

000: 1024 个 CTM 时钟

001: 128 个 CTM 时钟

010: 256 个 CTM 时钟

011: 384 个 CTM 时钟

100: 512 个 CTM 时钟

101: 640 个 CTM 时钟

110: 768 个 CTM 时钟

111: 896 个 CTM 时钟

此三位设定内部 CCRP 3-bit 寄存器的值，然后与内部计数器的高三位进行比较。如果 CTCCLR 位设定为 0 时，选中该比较结果清除内部计数器。CTCCLR 位设定为 0，内部计数器在比较器 P 比较匹配发生时被重置；由于 CCRP 只与计数器高三位比较，比较结果是 128 时钟周期的倍数。CCRP 被清零时，会使得计数器在最大值溢出。

• CTMC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	CTM1	CTM0	CTIO1	CTIO0	CTOC	CTPOL	CTDPX	CTCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **CTM1~CTM0**: CTM 工作模式选择位

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

这两个位为 CTM 设置所需的工作模式。为保证操作的可靠性，在此位段发生任何改变之前需先关闭 CTM。在定时 / 计数器模式下，CTM 输出引脚状态未定义。

Bit 5~4 **CTIO1~CTIO0**: CTP 输出功能选择位

比较匹配输出模式

- 00: 未改变
- 01: 输出低
- 10: 输出高
- 11: 输出翻转

PWM 输出模式

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

定时 / 计数器模式

未使用

这两位用于确定在达到某些条件时 CTM 输出引脚状态应如何改变。这些位的功能选择取决于当下 CTM 的运行模式。

在比较匹配输出模式下，CTIO1 和 CTIO0 位决定当比较器 A 比较匹配输出发生时 CTM 输出脚如何改变状态。当比较器 A 比较匹配输出发生时 CTM 输出脚设为切换高、切换低或翻转当前状态。若此两位同时为 0 时，输出将不会改变。CTM 输出脚的初始值通过 CTOC 位设置取得。注意，由 CTIO1 和 CTIO0 位得到的输出电平必须与通过 CTOC 位设置的初始值不同，否则当比较匹配发生时，CTM 输出脚将不会发生变化。在 CTM 输出脚改变状态后，通过 CTON 位由低到高电平的转换可复位至初始值。

在 PWM 输出模式，CTIO1 和 CTIO0 用于决定比较匹配条件发生时怎样改变 CTM 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。仅在 CTM 关闭时改变 CTIO1 和 CTIO0 的值是很有必要的。若在 CTM 运行时改变 CTIO1 和 CTIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **CTOC**: CTP 输出控制位

比较匹配输出模式

- 0: 初始低
- 1: 初始高

PWM 输出模式

- 0: 低有效
- 1: 高有效

此位为 CTM 输出脚的输出控制位。它取决于 CTM 此时正运行于比较匹配输出模式还是 PWM 输出模式。若 CTM 处于定时 / 计数器模式，则其不受影响。在比较匹配输出模式时，比较匹配发生前其决定 CTM 输出脚的逻辑电平值。在 PWM 输出模式时，其决定 PWM 信号是高有效还是低有效。

Bit 2 **CTPOL**: CTP 输出极性控制位

- 0: 同相
- 1: 反相

此位控制 CTP 输出脚的极性。此位为高时 CTM 输出脚反相，为低时 CTM 输出脚同相。若 CTM 处于定时 / 计数器模式时其不受影响。

- Bit 1 **CTDPX**: CTM PWM 周期 / 占空比控制位
 0: CCRP - 周期; CCRA - 占空比
 1: CCRP - 占空比; CCRA - 周期
 此位决定 CCRA 与 CCRP 寄存器哪个被用于 PWM 波形的周期和占空比控制。
- Bit 0 **CTCCLR**: 选择 CTM 计数器清零条件位
 0: CTM 比较器 P 匹配
 1: CTM 比较器 A 匹配
 此位用于选择清除计数器的方法。简易型 TM 包括两个比较器 - 比较器 A 和比较器 P。这两个比较器每个都可以用于清除内部计数器。CTCCLR 位设为高, 计数器在比较器 A 比较匹配发生时被清除; 此位设为低, 计数器在比较器 P 比较匹配发生或计数器溢出时被清除。计数器溢出清除的方法仅在 CCRP 的位都被清除为 0 时才能生效。CTCCLR 位在 PWM 输出模式时未使用。

● **CTMDL 寄存器**

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

- Bit 7~0 **D7~D0**: CTM 计数器低字节寄存器 bit 7~bit 0
 CTM 10-bit 计数器 bit 7~bit 0

● **CTMDH 寄存器**

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

- Bit 7~2 未定义, 读为 “0”
 Bit 1~0 **D9~D8**: CTM 计数器高字节寄存器 bit 1~bit 0
 CTM 10-bit 计数器 bit 9~bit 8

● **CTMAL 寄存器**

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

- Bit 7~0 **D7~D0**: CTM CCRA 低字节寄存器 bit 7~bit 0
 CTM 10-bit CCRA bit 7~bit 0

● **CTMAH 寄存器**

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

- Bit 7~2 未定义, 读为 “0”
 Bit 1~0 **D9~D8**: CTM CCRA 高字节寄存器 bit 1~bit 0
 CTM 10-bit CCRA bit 9~bit 8

简易型 TM 工作模式

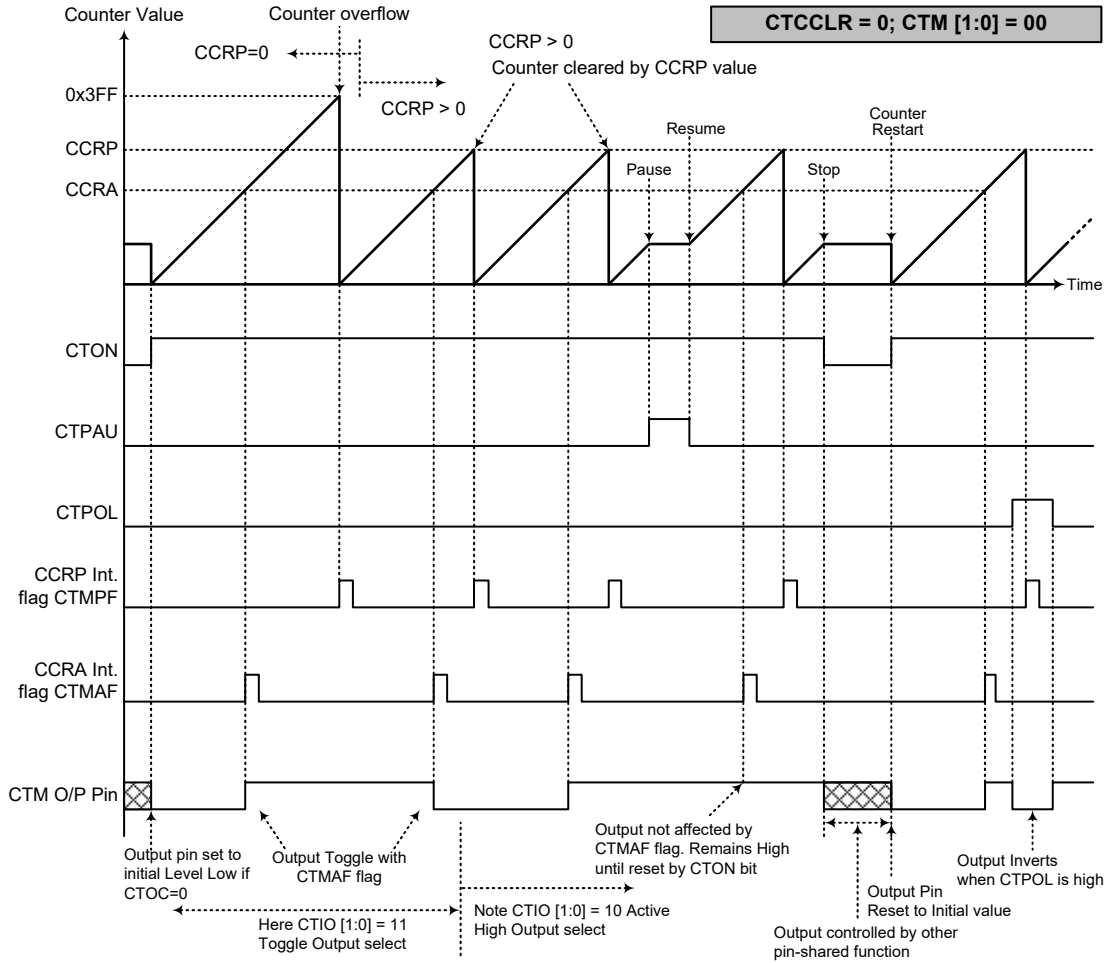
简易型 TM 有三种工作模式，即比较匹配输出模式、PWM 输出模式或定时 / 计数器模式。通过设置 CTMC1 寄存器的 CTM1 和 CTM0 位选择任意工作模式。

比较匹配输出模式

要工作在此模式，CTMC1 寄存器中的 CTM1 和 CTM0 位需要设置为“00”。当工作在该模式，一旦计数器使能并开始计数，有三种方法来清零，分别是：计数器溢出，比较器 A 比较匹配发生和比较器 P 比较匹配发生。当 CTCCLR 位为低，有两种方法清除计数器。一种是比较器 P 比较匹配发生，另一种是 CCRP 所有位设置为零并使得计数器溢出。此时，比较器 A 和比较器 P 的请求标志位 CTMAF 和 CTMPF 将分别置起。

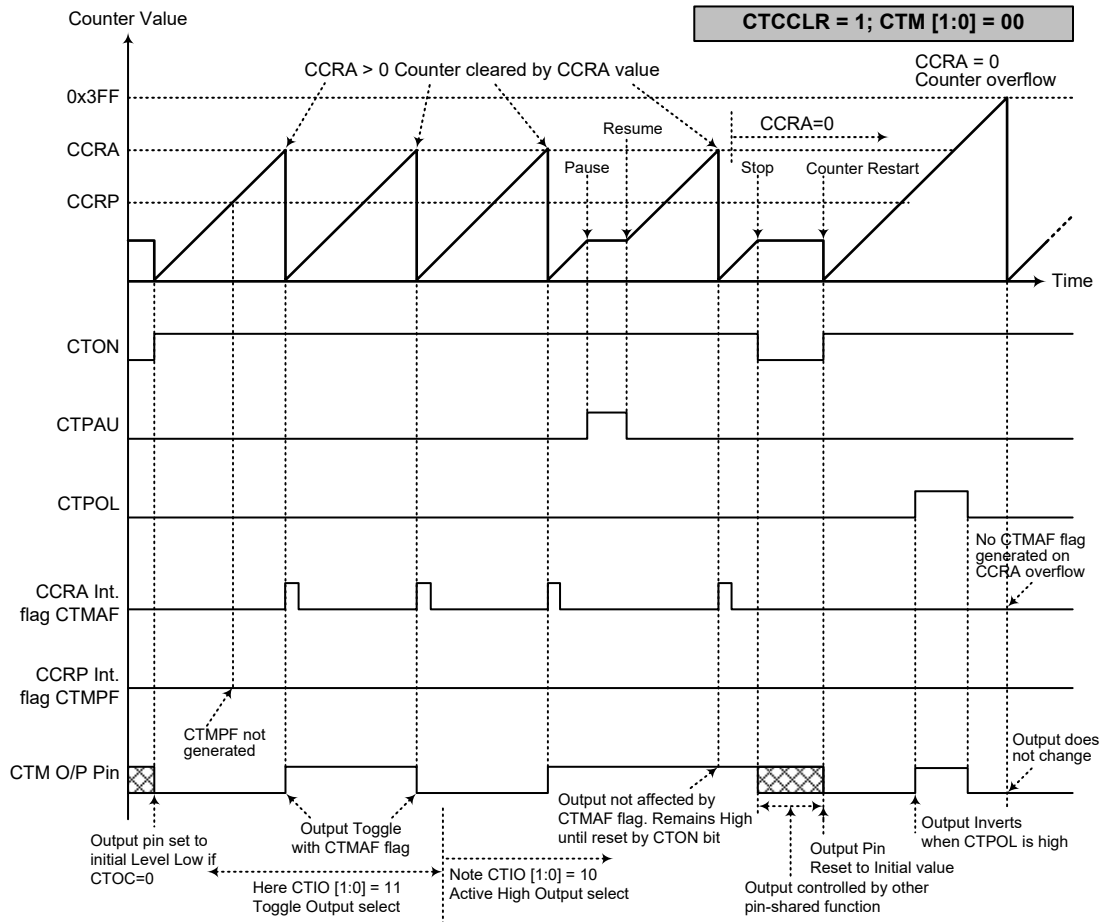
如果 CTMC1 寄存器的 CTCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 CTMAF 中断请求标志产生。所以当 CTCCLR 为高时，不产生 CTMPF 中断请求标志。如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 CTMAF 请求标志。

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



比较匹配输出模式 – CTCCLR=0

- 注：1. CTCCLR=0 时，比较器 P 匹配将清除计数器
2. CTM 输出引脚仅由 CTMAF 标志位控制
3. 输出引脚通过 CTON 位上升沿复位为初始值



比较匹配输出模式 – CTCCLR=1

- 注：1. CTCCLR=1 时，比较器 A 匹配将清除计数器
2. CTM 输出引脚控制仅由 CTMAF 标志位控制
3. 输出引脚通过 CTON 上升沿复位至初始值
4. 当 CTCCLR=1 时不产生 CTMPF 标志位

定时 / 计数器模式

要工作在此模式，CTMC1 寄存器中的 CTM1 和 CTM0 位需要设置为“11”。定时 / 计数器模式与比较输出模式操作方式相同，并产生同样的中断请求标志。不同的是，在定时 / 计数器模式下 CTM 输出脚未使用。因此，比较匹配输出模式中的描述和时序图可以帮助理解此功能。该模式中未使用的 CTM 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

要工作在此模式，CTMC1 寄存器中的 CTM1 和 CTM0 位需要设置为“10”。TM 的 PWM 功能在马达控制，加热控制，照明控制等方面十分有用。给 TM 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可控，其波形的选择就较为灵活。在 PWM 输出模式中，CTCCLR 位不影响 PWM 操作。CCRA 和 CCRP 寄存器决定 PWM 波形，一个用来清除内部计数器并控制 PWM 波形的频率，另一个用来控制占空比。哪个寄存器控制频率或占空比取决于 CTMC1 寄存器的 CTD PX 位。所以 PWM 波形频率和占空比由 CCRA 和 CCRP 寄存器共同决定。

当比较器 A 或比较器 P 比较匹配发生时，将产生 CCRA 或 CCRP 中断标志。CTMC1 寄存器中的 CTOC 位决定 PWM 波形的极性，CTIO1 和 CTIO0 位使能 PWM 输出或将 CTM 输出脚置为逻辑高或逻辑低。CTPOL 位对 PWM 输出波形的极性取反。

- 10-bit CTM, PWM 输出模式，边沿对齐模式，CTDPX=0

CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	128	256	384	512	640	768	896	1024
Duty	CCRA							

若 $f_{sys}=16\text{MHz}$ ，CTM 时钟源选择 $f_{sys}/4$ ，CCRP=100b，CCRA=128，

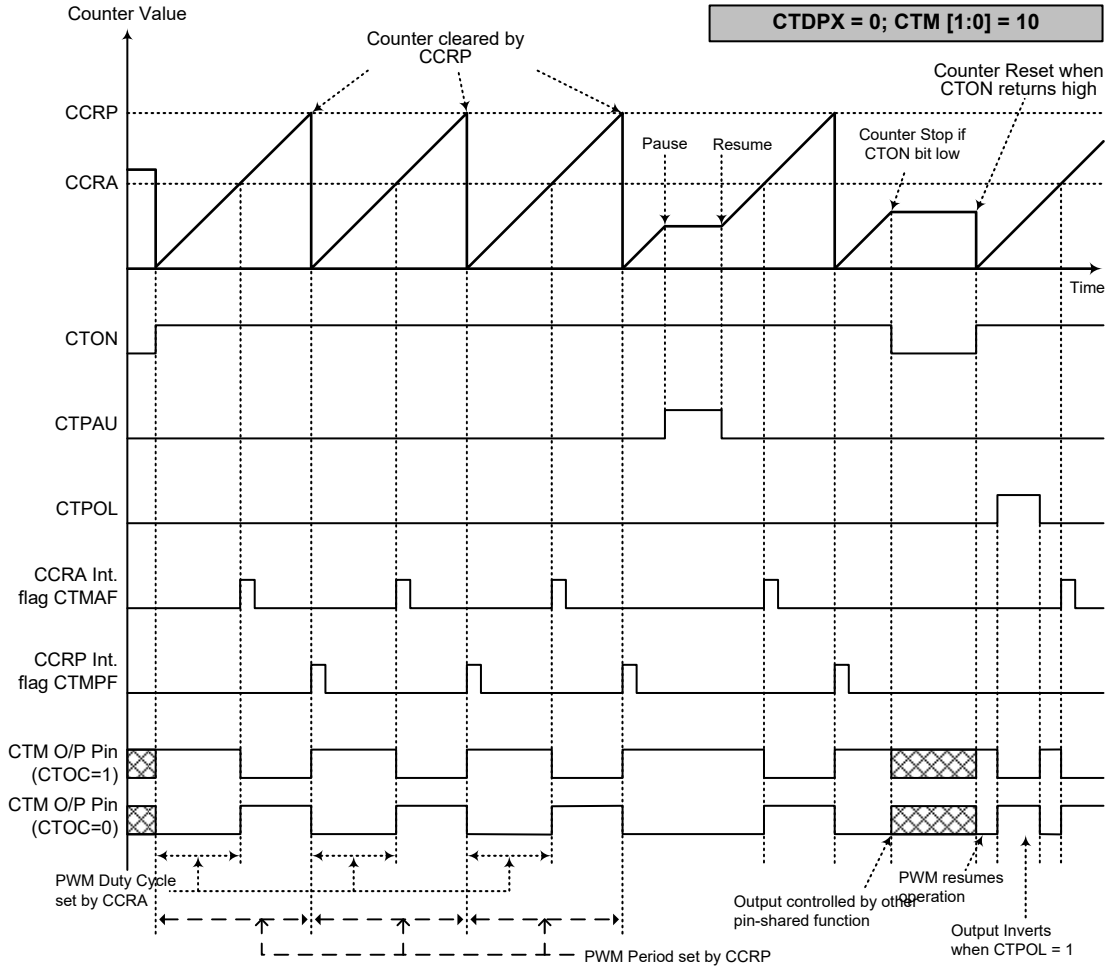
CTM PWM 输出频率 = $(f_{sys}/4)/512=f_{sys}/2048=8\text{kHz}$ ， $duty=128/512=25\%$ 。

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

- 10-bit CTM, PWM 输出模式，边沿对齐模式，CTDPX=1

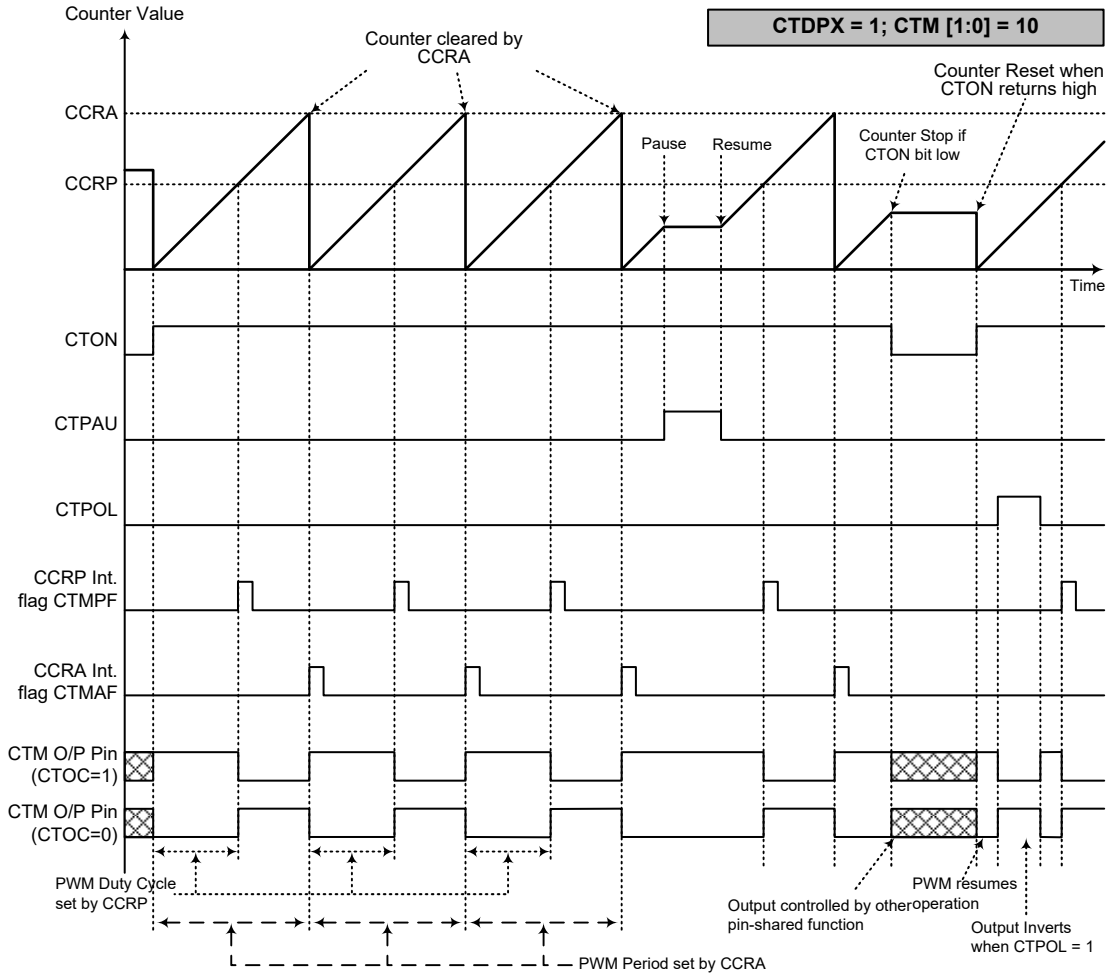
CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

PWM 的输出周期由 CCRA 寄存器的值与 CTM 的时钟共同决定，PWM 的占空比由 CCRP 寄存器的值决定。



PWM 输出模式 - CTD PX=0

- 注：1. 这里的 CTD PX=0 - 计数器由 CCRP 清除
 2. 计数器清除设置 PWM 周期
 3. 即使在 CTIO[1:0]=00 或 01 时，内部 PWM 功能继续运行
 4. CTCCLR 位对 PWM 操作没有影响



PWM 输出模式 - CTD PX=1

- 注：1. 这里的 CTD PX=1 - 计数器由 CCRA 清除
 2. 计数器清除设置 PWM 周期
 3. 即使在 CTIO[1:0]=00 或 01 时，内部 PWM 功能继续运行
 4. CTCCLR 位对 PWM 操作无影响

触控按键功能

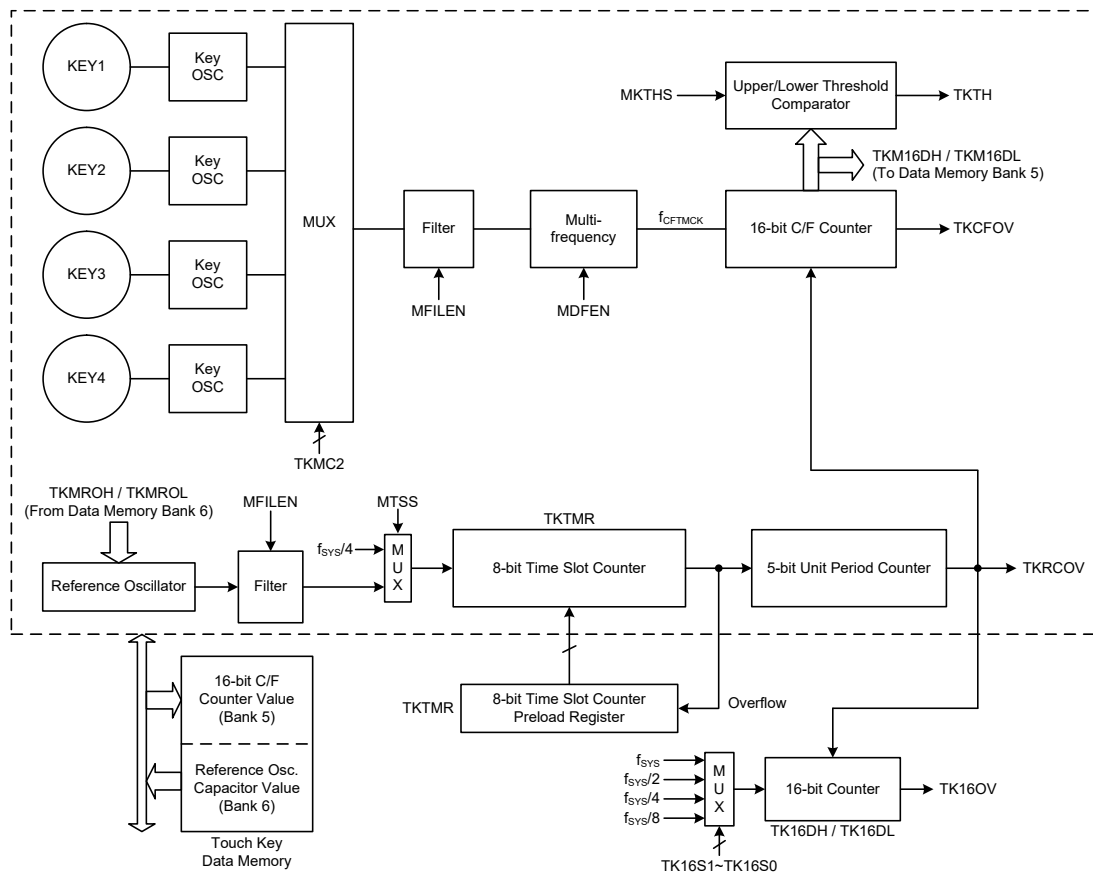
该单片机提供多个触控按键功能。该触控按键功能完全内部集成不需外接元件，通过内部寄存器对其进行简单的操作。

触控按键结构

触控按键引脚与 I/O 引脚共用。通过对应的引脚共用功能控制寄存器的位来选择此功能。按键被分成一个组，称为一个模块。该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。该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	PA5, PA1, PA3, PA4

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

触控按键寄存器定义

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
TKC2	触控按键功能控制寄存器 2
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
TKMC2	触控按键模块控制寄存器 2
TK1MTH16L	触控按键模块 KEY1 16-bit 阈值低字节
TK1MTH16H	触控按键模块 KEY1 16-bit 阈值高字节
TK2MTH16L	触控按键模块 KEY2 16-bit 阈值低字节
TK2MTH16H	触控按键模块 KEY2 16-bit 阈值高字节
TK3MTH16L	触控按键模块 KEY3 16-bit 阈值低字节
TK3MTH16H	触控按键模块 KEY3 16-bit 阈值高字节
TK4MTH16L	触控按键模块 KEY4 16-bit 阈值低字节
TK4MTH16H	触控按键模块 KEY4 16-bit 阈值高字节
TKMTHS	触控按键模块阈值比较标志位

触控按键功能寄存器定义

寄存器名称	位							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	TKRAMC	TKRCOV	TKST	TKCFOV	TK16OV	TKMOD1	TKMOD0	TKBUSY
TKC1	D7	D6	D5	D4	TK16S1	TK16S0	TKFS1	TKFS0
TKC2	—	—	—	—	—	—	ASMP1	ASMP0
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	—	—	MDFEN	MFILEN	MSOFC	MSOF2	MSOF1	MSOF0
TKMC1	MTSS	—	MROEN	MKOEN	MK4EN	MK3EN	MK2EN	MK1EN
TKMC2	MSK31	MSK30	MSK21	MSK20	MSK11	MSK10	MSK01	MSK00
TK1MTH16L	D7	D6	D5	D4	D3	D2	D1	D0
TK1MTH16H	D15	D14	D13	D12	D11	D10	D9	D8
TK2MTH16L	D7	D6	D5	D4	D3	D2	D1	D0
TK2MTH16H	D15	D14	D13	D12	D11	D10	D9	D8
TK3MTH16L	D7	D6	D5	D4	D3	D2	D1	D0
TK3MTH16H	D15	D14	D13	D12	D11	D10	D9	D8
TK4MTH16L	D7	D6	D5	D4	D3	D2	D1	D0
TK4MTH16H	D15	D14	D13	D12	D11	D10	D9	D8
TKMTHS	MK4THF	MK3THF	MK2THF	MK1THF	MK4THS	MK3THS	MK2THS	MK1THS

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 = $(256 - \text{TKTMR}[7:0]) \times 32t_{\text{rsc}}$ ，此处的 t_{rsc} 为时隙计数器时钟周期。

• TKC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TKRAMC	TKRCOV	TKST	TKCFOV	TK16OV	TKMOD1	TKMOD0	TKBUSY
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	1	0

Bit 7 **TKRAMC**: 触控按键 RAM 存取控制位

- 0: 由 MCU 读 / 写
- 1: 由触控按键模块读 / 写

该位用于控制触控按键模块 RAM 是由 MCU 还是触控按键模块使用。但若触控按键模块工作在自动扫描模式或周期性自动扫描模式 (当 TKMOD1~TKMOD0 位为 00, 10 或 11 时, TKST 位由 0 转为 1 可进入自动扫描模式或周期性自动扫描模式), 由触控按键模块可优先存取 RAM 内容。当结束自动扫描或周期性自动扫描后 (例如 TKBUSY 位状态由 1 转为 0), 触控按键模块 RAM 存取才可通过 TKRAMC 位控制。因此建议当触控按键模块工作在自动扫描模式或周期性自动扫描模式时, 将 TKRAMC 位置为“1”。否则, 若选择 MCU 存取 RAM, 在自动扫描或周期性自动扫描阶段, 触控按键的 RAM 内容可能会被改动。

Bit 6 **TKRCOV**: 触控按键时隙计数器溢出标志位

- 0: 无溢出
- 1: 溢出

此位可通过应用程序读 / 写。当触控按键时隙计数器溢出将此位置为“1”时, 相应的触控按键 TKRCOV 中断请求标志位也会同时置位。然而若是通过应用程序将此位设置为“1”时, 相应的触控按键 TKRCOV 中断请求标志位不会受到影响。因此, 该位不能通过应用程序置位, 但必须通过应用程序清零。

在自动扫描模式时, 如果时隙计数器溢出, 但自动扫描还未完成, TKRCOV 位将不会被置位, 同时触控按键模块 16 位 C/F 计数器、触控按键功能 16 位计数器、5 位时隙单位周期计数器将会自动清零, 但 8 位时隙定时计数器将会从 8 位时隙定时计数器预置寄存器 (TKTMR 寄存器) 重新加载数据。当自动扫描工作结束, TKRCOV 位及触控按键 TKRCOV 中断请求标志位 TKRCOVF 将被置位同时所有模块按键振荡器和参考振荡器自动停止。触控按键模块的 16 位 C/F 计数器、触控按键功能 16 位计数器、5 位时隙单位周期计数器和 8 位时隙定时计数器都会自动关闭。

在周期性自动扫描模式时, 在自动扫描工作周期内, TKRCOV 位将被清零。仅在 WDT 溢出周期内的最后一次扫描结束时, 16 位 C/F 计数器内容将被写入相应的触控按键 RAM, TKRCOV 位将被硬件电路置高。除了以上提到的, 该模式下的其它动作和自动扫描模式一样。

在手动扫描模式时, 如果时隙计数器溢出, TKRCOV 位及触控按键 TKRCOV 中断请求标志位 TKRCOVF 将被置位同时所有模块按键振荡器和参考振荡器自动停止。触控按键模块的 16 位 C/F 计数器、触控按键功能 16 位计数器、5 位时隙单位周期计数器和 8 位时隙定时计数器也都会自动关闭。

Bit 5 **TKST**: 触控按键检测开启控制位

- 0: 检测停止或无操作
- 0→1: 启动检测

当该位为“0”时, 触控按键模块的 16 位 C/F 计数器、触控按键功能 16 位计数器和 5 位时隙单位周期计数器会自动清零但 8 位可编程时隙定时计数器不会被清零。当该位由 0→1 时, 16 位 C/F 计数器、触控按键功能 16 位计数器、5 位时隙单位周期计数器和 8 位时隙定时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 位 C/F 计数器溢出标志位

- 0: 无溢出
- 1: 溢出

该位由触控按键模块 16 位 C/F 计数器溢出置位, 必须通过应用程序清零。

- Bit 3 TK16OV:** 触控按键功能 16 位计数器溢出标志位
0: 无溢出
1: 溢出
该位由触控按键功能 16 位计数器溢出置位，必须通过应用程序清零。
- Bit 2~1 TKMOD1~TKMOD0:** 触控按键扫描模式选择位
00: 自动扫描模式
01: 手动扫描模式
1x: 周期性自动扫描模式
在手动扫描模式时，应在扫描开始之前，合理的设置参考振荡器电容值，并在扫描结束后通过应用程序读取触控按键模块 16 位 C/F 计数器值。
在自动扫描模式时，上面所说的数据的读取或写入是通过硬件完成。有一个专用的触控按键模块数据存储区可存放所有被扫描按键的参考振荡器电容值及 16 位 C/F 计数器值。自动扫描按键的顺序是由 TKMC2 寄存器的 MSK3[1:0]~MSK0[1:0] 位设置。直至扫描完所有计划扫描的按键后，自动扫描工作才会停止。
在周期性自动扫描模式时，触控按键扫描工作将自动周期性的进行，可由寄存器 TKC2 的 ASMP1~ASMP0 位决定。仅在 WDT 溢出周期内的最后一次扫描结束时，16 位 C/F 计数器内容将被写入相应的触控按键 RAM。另外，当 MKnTHS 位为 0 时，任何一个按键 C/F 计数器值小于下限阈值，或 MKnTHS 为 1 时，任何一个按键 C/F 计数器值大于上限阈值，TKTH 信号都将被置高。除了以上提到的，该模式下的其它动作和自动扫描模式一样。
- Bit 0 TKBUSY:** 触控按键扫描忙碌标志位
0: 空闲 – 没有在执行按键扫描或按键扫描已完成
1: 忙碌 – 正在扫描中
该位用于指示触控按键扫描工作是否在执行中。当 TKST 位置高启动扫描工作，该位被置“1”。
在手动扫描模式中，当触控时隙计数器溢出时，该位会自动清零。在自动扫描模式中，当触控按键扫描工作完成时，该位也会自动清零。在周期性自动扫描模式时，若在 WDT 溢出周期内完成最后一次扫描工作，或当 MKnTHS 位为 0，任何一个按键 C/F 计数器值小于下限阈值时，或当 MKnTHS 为 1，任何一个按键 C/F 计数器值大于上限阈值时，这些情况下，该位都会被清零。

• **TKC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	TK16S1	TK16S0	TKFS1	TKFS0
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	1	1

- Bit 7~5 D7~D5:** 测试数据位
这些位仅供内部测试使用，正常工作时，要保持 D7~D5 为“000”。
- Bit 4 D4:** 保留位，不可被用。
- Bit 3~2 TK16S1~TK16S0:** 触控按键功能 16 位计数器时钟选择位
00: f_{sys}
01: f_{sys}/2
10: f_{sys}/4
11: f_{sys}/8
- 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

• TKC2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	ASMP1	ASMP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	1

Bit 7~2 未定义，读为“0”

Bit 1~0 **ASMP1~ASMP0**: 周期性自动扫描模式周期选择位

00: $2^{14}/f_{LIRC}$

01: $2^{13}/f_{LIRC}$

10: $2^{12}/f_{LIRC}$

11: $2^{11}/f_{LIRC}$

这些位用来决定触控按键扫描周期，仅当触控按键功能被配置为工作在周期性自动扫描模式时有效。触控按键扫描次数可通过 WDT 溢出周期得到，周期性自动扫描模式周期 t_{KEY} 可通过等式 $N=t_{WDT}/t_{KEY}$ 得到。

例如：若将 WS[2:0] 位设为 100，选择 WDT 溢出周期为 $2^{15}/f_{LIRC}$ ， t_{WDT} 等于 1.024s。因此，在一个 WDT 溢出周期内，当 ASMP[1:0] 位分别设为 00/01/10/11 时，触控按键扫描次数为 2/4/8/16 次。在应用上确保周期性自动扫描模式周期 t_{KEY} 不会超过 WDT 溢出周期 t_{WDT} 是非常重要的。

• 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 计数器用于校准参考振荡器或按键振荡器频率。如果触控按键时隙计数器溢出，此 16-bit 计数器将停止，但不会改变当前值。在自动扫描模式或周期性自动扫描模式时，时隙 0、时隙 1 和时隙 2 扫描结束时，该 16-bit 计数器值会被清零，但在时隙 3 结束时，该 16-bit 计数器值不会改变。当 TKST 位为“0”时，该寄存器对被清零。

• 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。在手动扫描模式时，如果触控按键时隙计数器溢出，该 16-bit C/F 计数器将停止，但不会改变当前值。在自动扫描模式或周期性自动扫描模式时，时隙 0、时隙 1 和时隙 2 扫描结束时，该 16-bit C/F 计数器值会被清零，但在时隙 3 结束时，16-bit C/F 计数器值不会改变。当 TKST 位为“0”时，该寄存器对被清零。

● **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。在自动扫描模式或周期性自动扫描模式时，当前时隙扫描结束后，此寄存器会从专用的触控按键模块数据存储区加载下一时隙要扫描按键对应的参考振荡器电容值。
参考振荡器内接电容值 = (TKMRO[9:0]×50pF)/1024。

● **TKMCO 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	MDFEN	MFILEN	MSOFC	MSOF2	MSOF1	MSOF0
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 **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: 1.125MHz
 001: 1.111MHz
 010: 1.099MHz
 011: 1.085MHz
 100: 1.074MHz
 101: 1.059MHz
 110: 1.040MHz
 111: 1.020MHz

这些位用于为跳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。

● TKMCI1 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。在自动扫描模式或周期性自动扫描模式时, 若选择使用参考振荡器作为时隙时钟源, 当 TKST 位由低到高变化时, 必须通过将 MROEN 位置高使能参考振荡器。MTSS 位和 MK4EN~MK1EN 位共同决定了参考振荡器是否被用。当 TKBUSY 位从高变为低时, MROEN 位将被自动清零来除能参考振荡器。

在手动扫描时, 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, 当 TKBUSY 位从高变为低时, 参考振荡器将除能。

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。在自动扫描模式或周期性自动扫描模式时, 当 TKST 位由低到高变化时, 必须通过将 MKOEN 位置高使能按键振荡器。当 TKBUSY 位由高到低变化时, MKOEN 位将被自动清零来除能按键振荡器。

在手动扫描模式时, 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。当 TKBUSY 位从高变为低时, 参考振荡器将除能。

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

● **TKMC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MSK31	MSK30	MSK21	MSK20	MSK11	MSK10	MSK01	MSK00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	1	0	0

Bit 7~6 **MSK31~MSK30**: 触控按键模块时隙 3 扫描按键选择位

- 00: KEY1
01: KEY2
10: KEY3
11: KEY4

这两位用于选择在时隙 3 时要被扫描的按键，仅在自动扫描模式或周期性自动扫描模式时有效。

Bit 5~4 **MSK21~MSK20**: 触控按键模块时隙 2 扫描按键选择位

- 00: KEY1
01: KEY2
10: KEY3
11: KEY4

这两位用于选择在时隙 2 时要被扫描的按键，仅在自动扫描模式或周期性自动扫描模式时有效。

Bit 3~2 **MSK11~MSK10**: 触控按键模块时隙 1 扫描按键选择位

- 00: KEY1
01: KEY2
10: KEY3
11: KEY4

这两位用于选择在时隙 1 时要被扫描的按键，仅在自动扫描模式或周期性自动扫描模式时有效。

Bit 1~0 **MSK01~MSK00**: 触控按键模块时隙 0 扫描按键选择位

- 00: KEY1
01: KEY2
10: KEY3
11: KEY4

这两位用于选择在自动扫描模式或周期性自动扫描模式下时隙 0 时要被扫描的按键或在手动模式时选择要扫描的按键。

● **TKnMTH16L 寄存器 (n=1~4)**

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 KEYn 16-bit 阈值低字节内容

● **TKnMTH16H 寄存器 (n=1~4)**

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**: 触控按键模块 KEYn 16-bit 阈值高字节内容
 这四个寄存器对分别用来存储触控按键模块 KEY1~KEY4 16-bit 上限 / 下限阈值。当触控按键模块 KEYn 完成扫描工作后, 16-bit C/F 计数器内容 TKM16DH/TKM16DL 将和 TKnMTH16H/TKnMTH16L 值将通过硬件进行比较。当 MKnTHS=0, 该值小于下限阈值时或当 MKnTHS=1, 该值大于上限阈值, MKnTHF 标志位将置高, 中断信号将产生。

● **TKMTHS 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MK4THF	MK3THF	MK2THF	MK1THF	MK4THS	MK3THS	MK2THS	MK1THS
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 **MK4THF**: 触控按键模块 KEY4 上限 / 下限阈值比较标志位
 0: 不小于下限阈值或不大于上限阈值
 1: 小于下限阈值或大于上限阈值

Bit 6 **MK3THF**: 触控按键模块 KEY3 上限 / 下限阈值比较标志位
 0: 不小于下限阈值或不大于上限阈值
 1: 小于下限阈值或大于上限阈值

Bit 5 **MK2THF**: 触控按键模块 KEY2 上限 / 下限阈值比较标志位
 0: 不小于下限阈值或不大于上限阈值
 1: 小于下限阈值或大于上限阈值

Bit 4 **MK1THF**: 触控按键模块 KEY1 上限 / 下限阈值比较标志位
 0: 不小于下限阈值或不大于上限阈值
 1: 小于下限阈值或大于上限阈值

Bit 3 **MK4THS**: 触控按键模块 KEY4 上限 / 下限阈值比较选择位
 0: 下限阈值比较
 1: 上限阈值比较

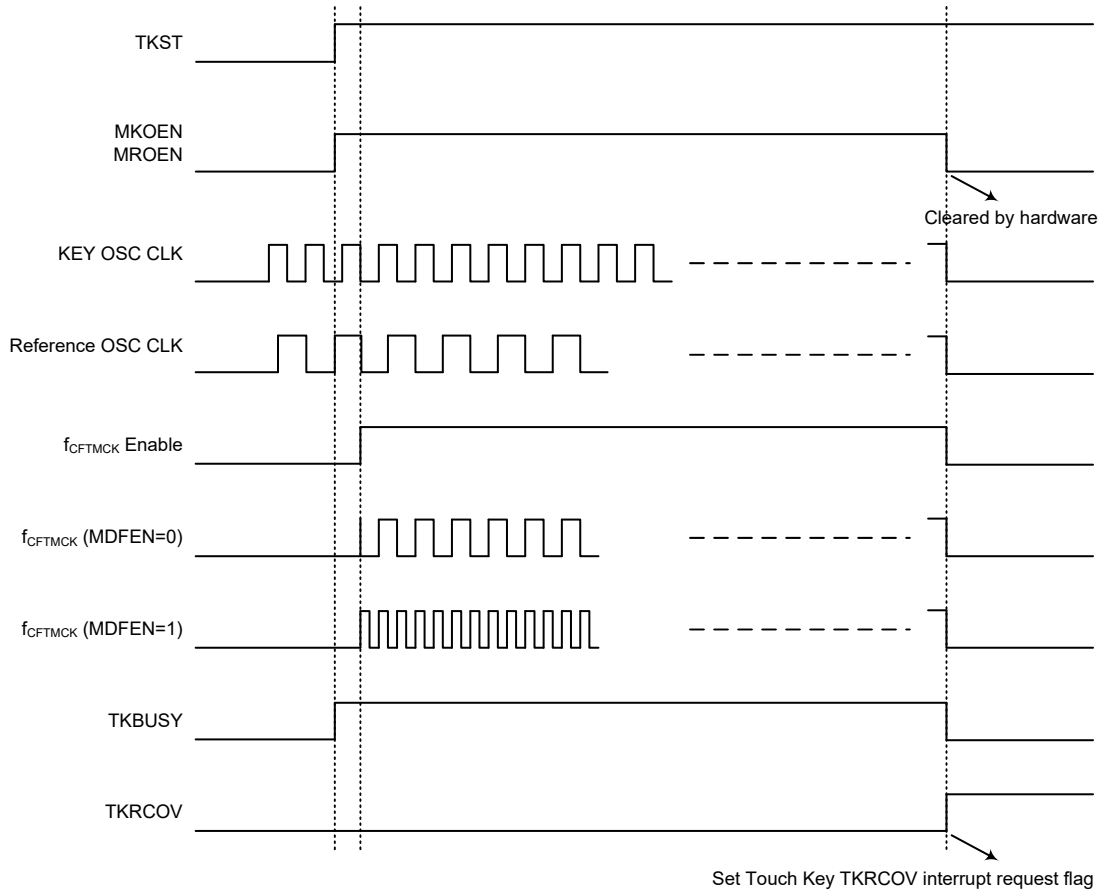
Bit 2 **MK3THS**: 触控按键模块 KEY3 上限 / 下限阈值比较选择位
 0: 下限阈值比较
 1: 上限阈值比较

Bit 1 **MK2THS**: 触控按键模块 KEY2 上限 / 下限阈值比较选择位
 0: 下限阈值比较
 1: 上限阈值比较

Bit 0 **MK1THS**: 触控按键模块 KEY1 上限 / 下限阈值比较选择位
 0: 下限阈值比较
 1: 上限阈值比较

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。



触控按键手动扫描模式时序图

触控按键模块包含四个与 I/O 引脚共用的触控按键 KEY1~KEY4。通过相关的引脚共用功能控制寄存器位可选择触控按键引脚功能。触控按键具有独立的感应振荡器，因此触控按键模块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。在手动扫描模式时，在此固定的时间间隔最后，会产生一个触控按键 TKRCOV 中断信号。

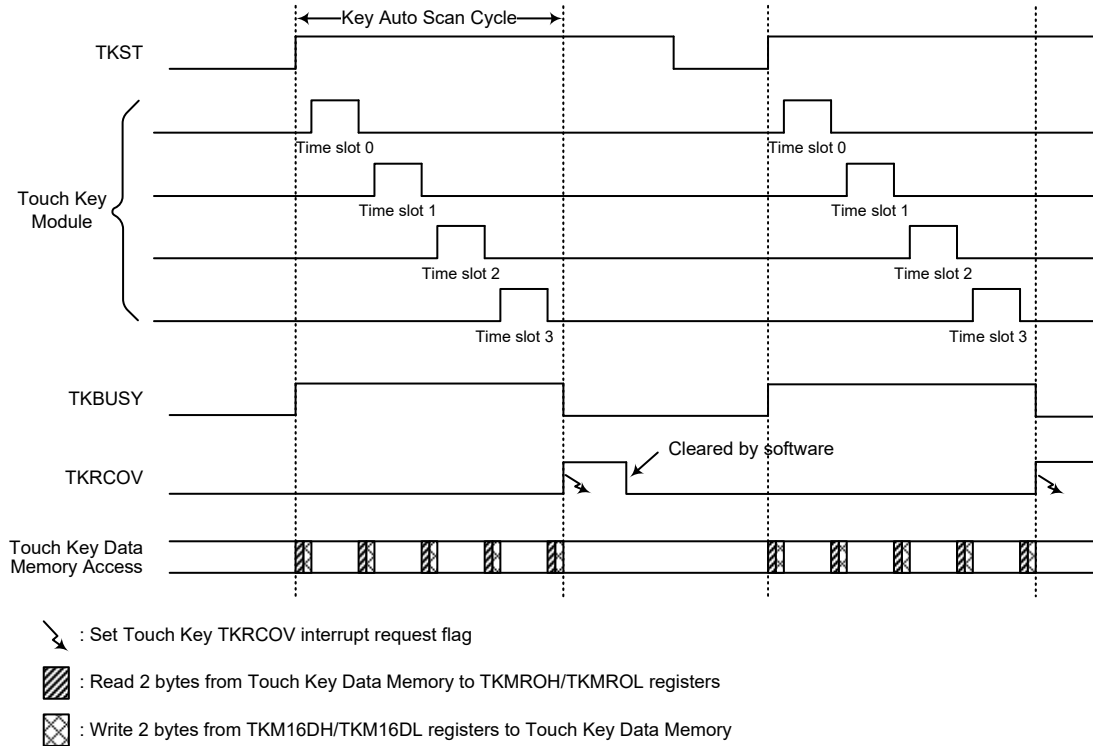
当 TKST 位被清零时，触控按键模块的 16 位 C/F 计数器、16 位计数器和 5 位时隙单位周期计数器会自动清零，而 8 位可编程时隙计数器不清零，由用户设置溢出时间。当 TKST 位置高时，16 位 C/F 计数器、16 位计数器、5 位时隙单位周期计数器和 8 位时隙定时计数器会自动开启。

如果时隙计数器溢出，触控按键模块的按键振荡器和参考振荡器都会自动停止且 16 位 C/F 计数器、16 位计数器、5 位时隙单位周期计数器和 8 位时隙定时计数器会自动停止。时隙计数器时钟源可通过 TKMC1 寄存器 MTSS 位选择来自参考振荡器或 $f_{sys}/4$ 。通过设置 TKMC1 寄存器中的 MROEN 位和 MKOEN 为“1”，可启用参考振荡器和按键振荡器。

当触控按键模块的时隙计数器溢出，产生触控按键 TKRCOV 中断。这里所有的触控按键是指已使能的触控按键。

自动扫描模式

触控按键功能包含三种按键扫描模式，即自动扫描模式、周期性自动扫描和手动扫描模式，可通过寄存器 TKC0 的 TKMOD1~TKMOD0 位选择。自动扫描模式可以最大程度的减少程序负担并能提高按键扫描执行效率。设置 TKMOD1~TKMOD0 位为 00 可选择自动扫描模式，在此模式下以指定顺序自动扫描模块按键。扫描顺序及按键由 TKMC2 寄存器的 MSK3[1:0]~MSK0[1:0] 位决定。

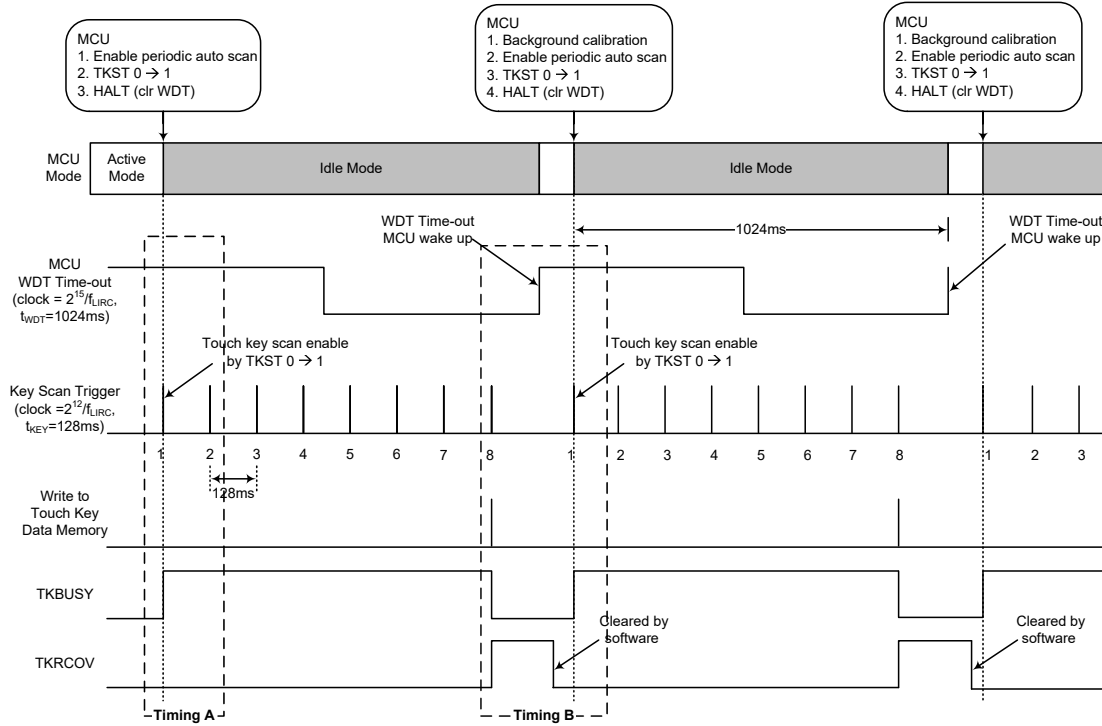


触控按键自动扫描模式时序图

在自动扫描模式时，按键振荡器和参考振荡器在 TKST 位由低变为高时，自动使能，在 TKBUSY 位由高到低时，自动除能。在自动扫描模式时，若已设置 TKST 位由低到高，硬件会自动从专用触控按键数据存储区指定位置读取时隙 0 选择扫描的按键对应的参考振荡器的电容值，并写入相应的 TKMROH/TKMROL 寄存器对中。并将 16-bit C/F 计数器的值写回到专用触控按键数据存储区时隙 3 扫描的按键对应的位置。之后开始扫描选择的时隙 0 的按键。时隙 0 按键扫描结束时，硬件会自动从专用触控按键数据存储区读取下一个要扫描按键对应的参考振荡器电容值并写入相应的 TKMROH/TKMROL 寄存器对中。并将当前 16-bit C/F 计数器的值写回到专用触控按键数据存储区对应位置。整个自动扫描操作会按上述指定的方式从时隙 0 到时隙 3 有序的执行。时隙 3 按键扫描结束时，硬件会再次从专用触控按键数据存储区读取时隙 0 选择扫描的按键对应的参考振荡器的电容值，并写入相应的 TKMROH/TKMROL 寄存器对中。16 位 C/F 计数器值也会被再次写回到专用触控按键数据存储区时隙 3 扫描的按键对应的位置。四个按键全部扫描后，TKRCOV 位将被置高同时 TKBUSY 位拉低，意味着自动扫描模式下的扫描工作已完成。

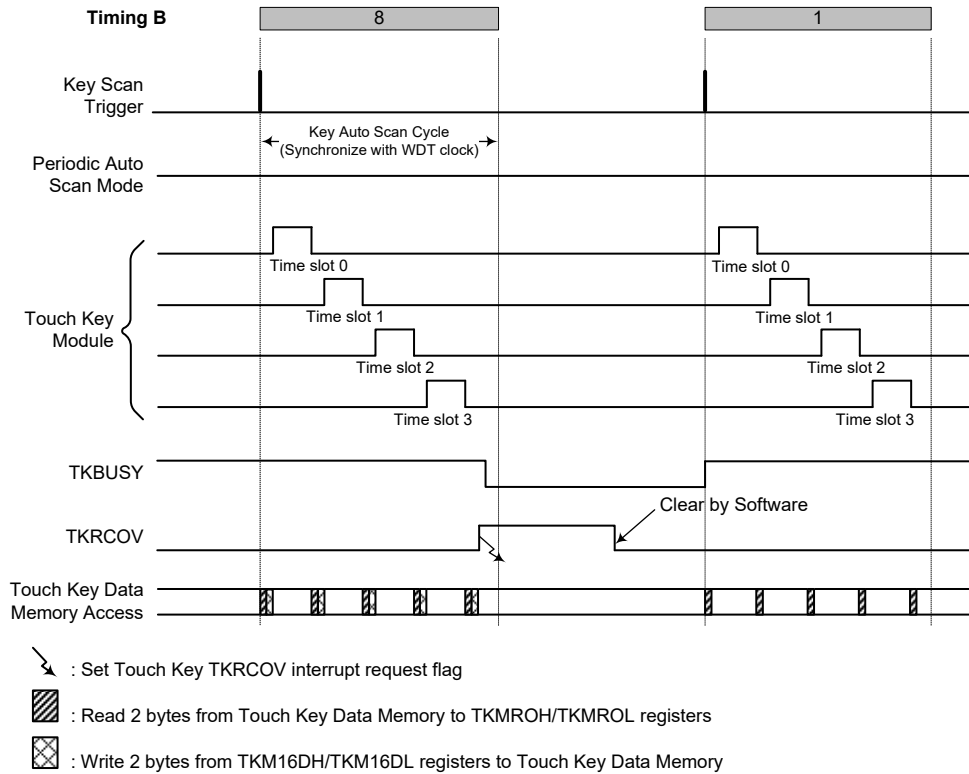
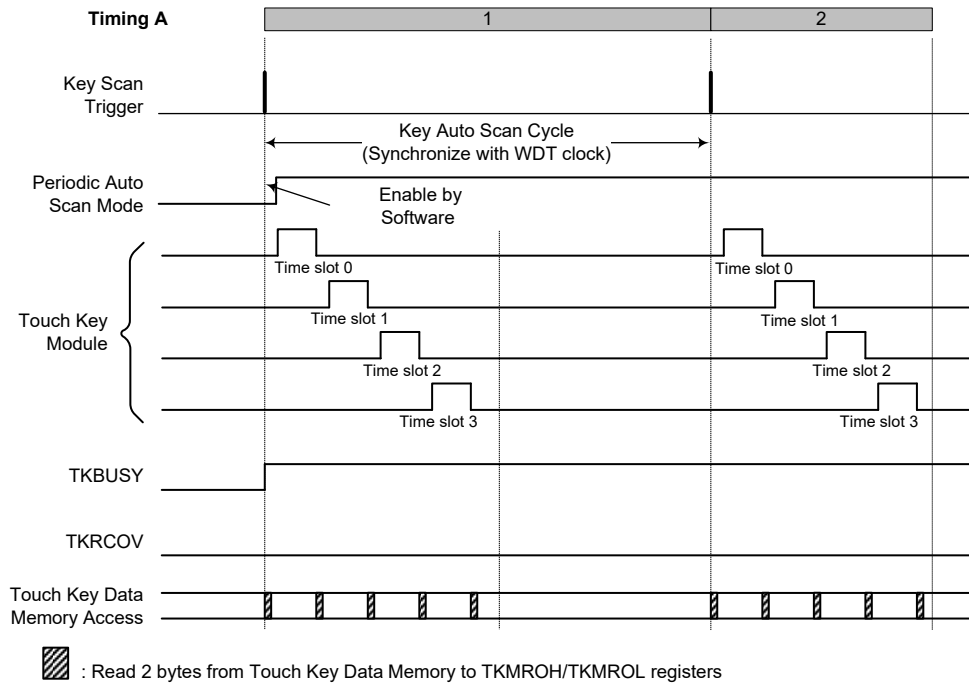
周期性自动扫描模式

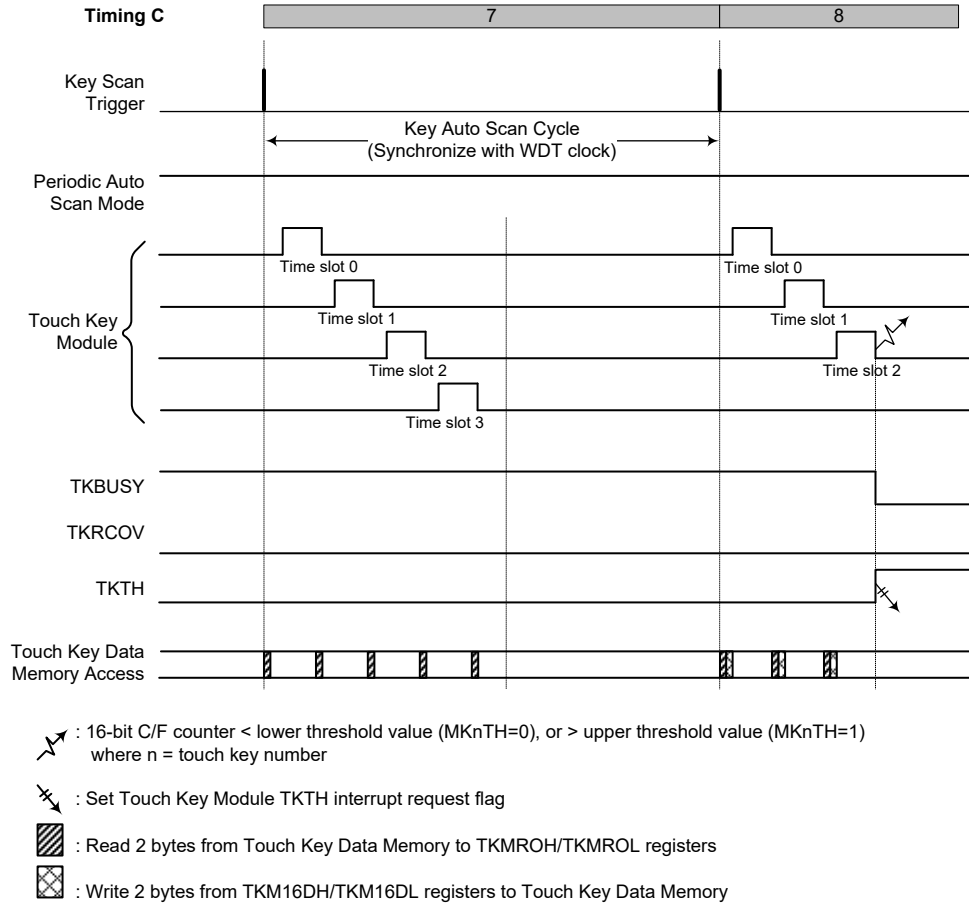
除了在自动扫描模式中提到的动作，周期性自动扫描模式还提供了周期自动扫描和 C/F 计数器上限 / 下限阈值比较功能。设置 TKMOD1~TKMOD0 位为 10 或 11，可选择周期性自动扫描模式用来自动和周期性地扫描模块按键。须注意的是，该模式通常在空闲模式下使用，用来监测触控按键状态，并尽量减少功耗。



注：触控按键周期性自动扫描模式时序图显示了特殊的时序 A 和时序 B。

空闲模式下的触控按键周期性自动扫描功能





注：1. 时序 A 和时序 B 为空闲模式下的触控按键周期性自动扫描功能中特定的时序图。
2. 时序 C 为某个触控按键阈值比较状态发生时的时序图。

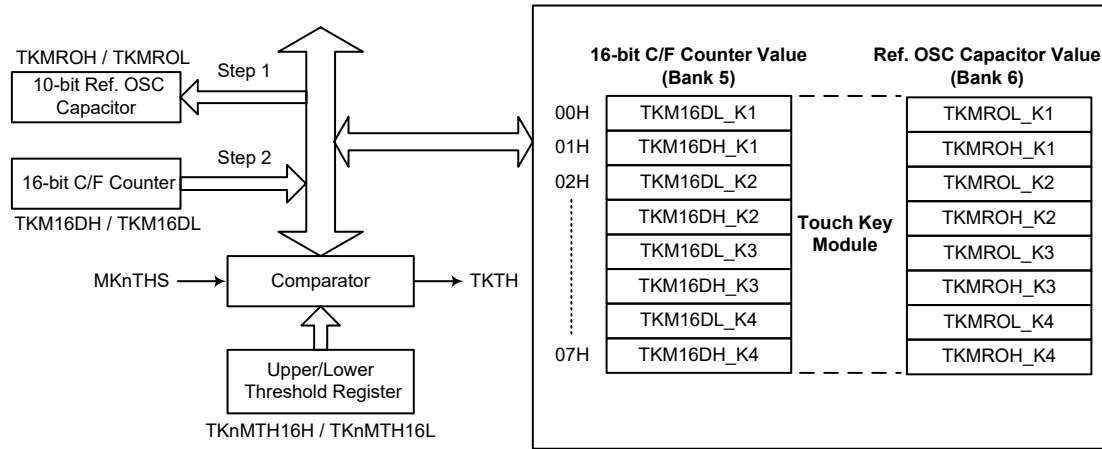
触控按键周期性自动扫描模式时序图

在周期性自动扫描模式时，触控按键扫描工作将周期性地自动执行，可由寄存器 TKC2 的 ASMP1~ASMP0 位决定。触控按键扫描次数取决于 WDT 溢出周期和周期性自动扫描模式周期。整个周期性自动扫描操作会像自动扫描模式一样按指定的方式从时隙 0 到时隙 3 有序的执行。每个时隙选择扫描的按键对应的参考振荡器的电容值将会被自动从专用触控按键数据存储区指定位置读取出来，并写入相应的 TKMROH/TKMROL 寄存器对中。然而，仅在 WDT 溢出周期内最后一次扫描工作结束时，16-bit C/F 计数器的值才会被写回到专用触控按键数据存储区所有扫描的按键对应的位置。

另外，每个触控按键都有自己独立的上限 / 下限阈值比较器。在周期性自动扫描模式时，上限 / 下限阈值比较功能将自动使能。当 MKnTHS=0，任何按键 C/F 计数器值小于下限阈值，或 MKnTHS=1，按键 C/F 计数器值大于上限阈值，这表明触控按键状态发生了变化，MKnTHF 标志位将被硬件置高，中断信号产生。使用 WDT 计数器时钟进行周期性自动扫描可减少功耗，当 WDT 被清除时，WDT 计数器将复位，周期性自动扫描工作时间将会受到影响，但触控按键次数却不受影响。

触控按键模块数据存储器

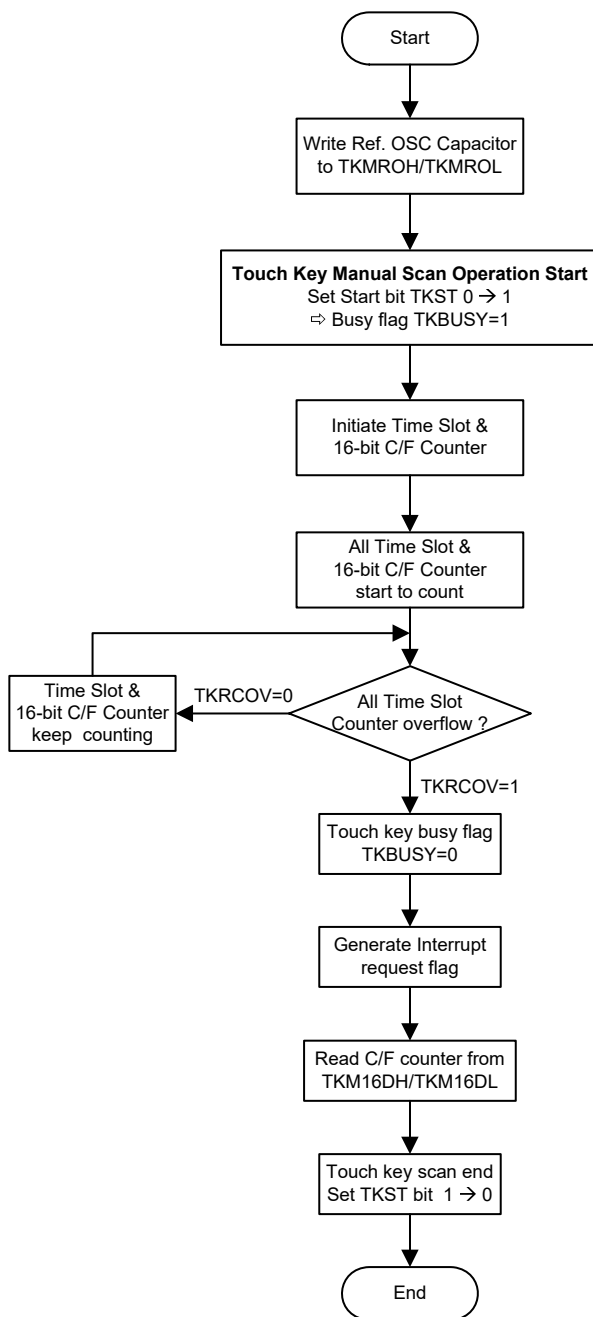
该单片机为触控按键模块自动扫描模式提供 2 个专用的数据存储器区域。一个位于数据存储区 5 用于存储触控按键模块 16 位 C/F 计数器值，另一个位于数据存储区 6 用于存储触控按键模块参考振荡器内部电容值。



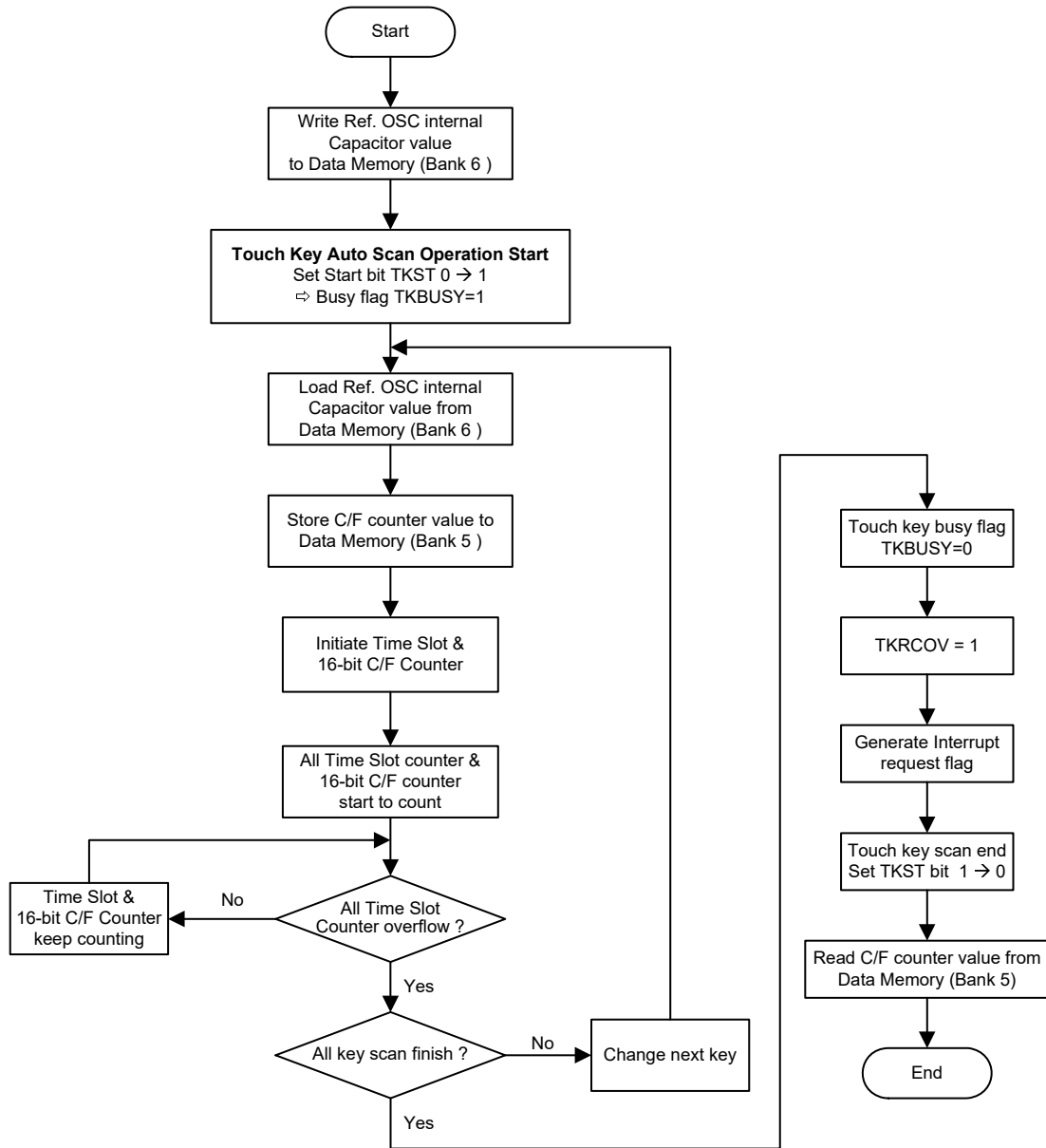
注：触控按键数 n=1~4。

触控按键模块数据存储器分布

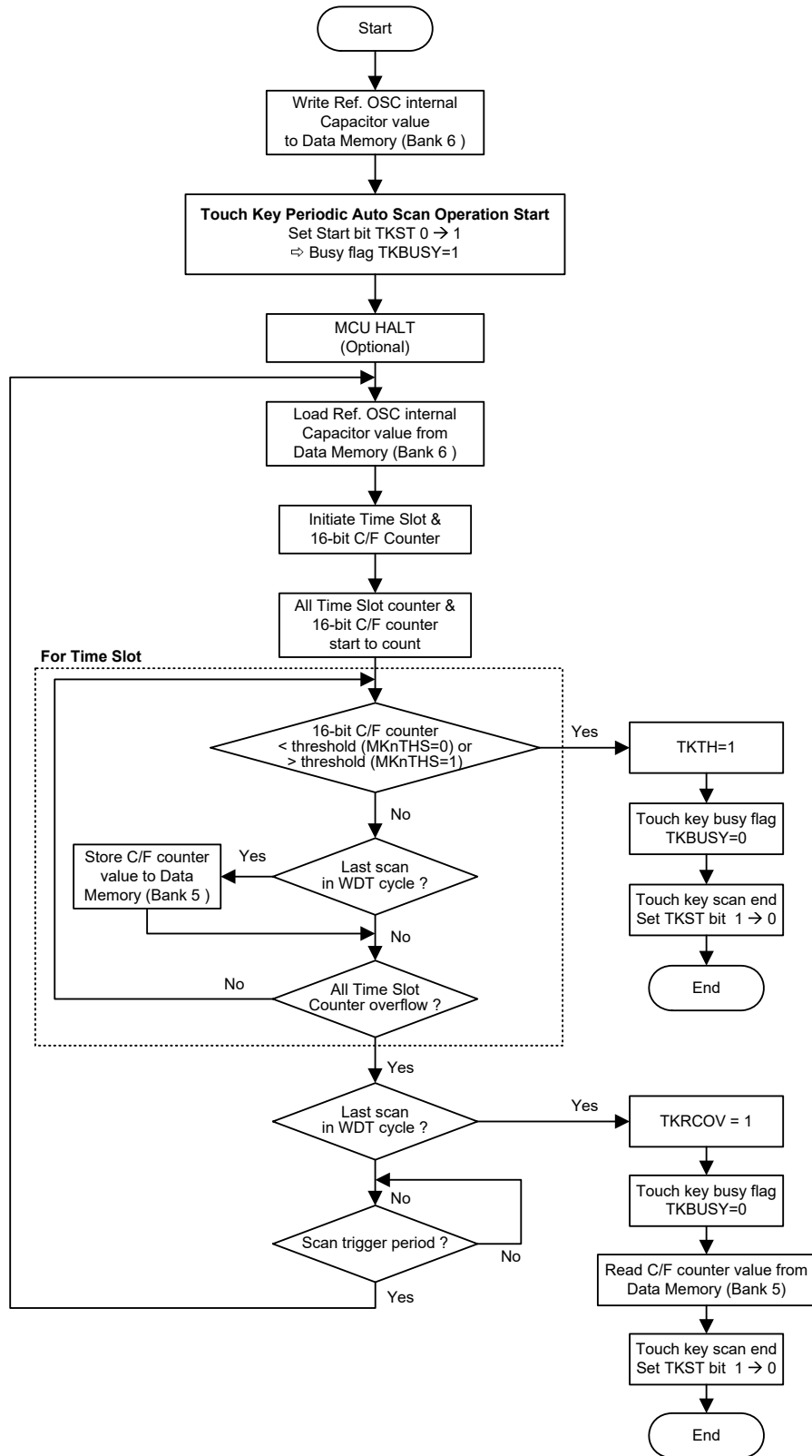
触控按键扫描流程图



触控按键手动扫描模式流程图



触控按键自动扫描模式流程图



触控按键周期性自动扫描模式流程图

触控按键中断

触控按键有两个独立的中断，分别为触控按键 TKRCOV 中断和触控按键模块 TKTH 中断。当触控按键模块的时隙计数器溢出时，才会产生触控按键 TKRCOV 中断，注意，此处提到的触控按键是指已被使能的触控按键。此时 16 位 C/F 计数器、16 位计数器，5 位时隙单位周期计数器和 8 位时隙定时计数器会自动清零。当 $MKnTHS=0$ ，任何按键 C/F 计数器值小于下限阈值，或 $MKnTHS=1$ ，按键 C/F 计数器值大于上限阈值，将产生触控按键模块 TKTH 中断。

若触控按键模块的 16 位 C/F 计数器溢出，16 位 C/F 计数器溢出标志位 TKCFOV 将置高，此中断标志位不会自动复位，必须通过应用程序将其清零。

若 16 位计数器溢出，16 位计数器溢出标志位 TK16OV 将置高，此中断标志位不会自动复位，必须通过应用程序将其清零。触控按键中断的详细说明见中断部分。

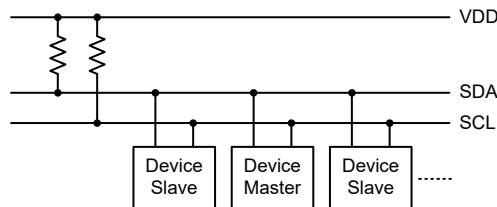
编程注意事项

相关寄存器设置后，将 TKST 位由低电平变为高电平会启动触控按键检测程序初始化。此时所有相关的振荡器将同步使能。当计数器溢出时，时隙计数器标志位 TKRCOV 将变为高电平。计数器溢出发生时，会产生一个中断信号。当某一阈值比较条件发生时，阈值比较指示信号 TKTH 信号将变为高电平。某一阈值比较条件发生时将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。

I²C 接口

I²C 可以和传感器、EEPROM 存储器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。

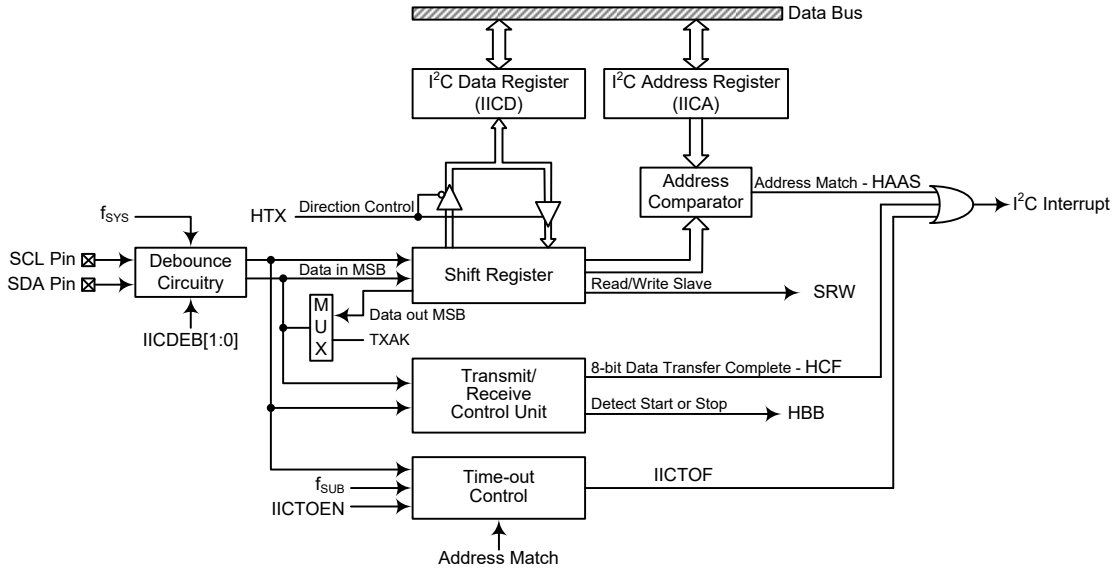


I²C 主从总线连接图

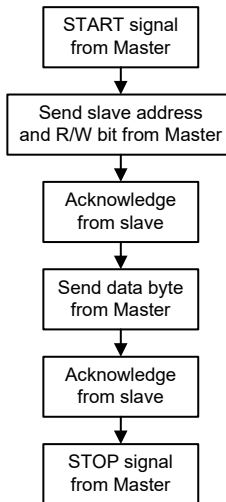
I²C 接口操作

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。因此应在这些输出上都应加上拉电阻。应注意的是，I²C 总线上的每个设备都没有选择线，但分别与唯一的地址一一对应，用于 I²C 通信。

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



I²C 方框图



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。

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

I²C 最小 f_{SYS} 频率要求

I²C 寄存器

I²C 总线有三个控制寄存器 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 7~1 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 IICA 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 超时章节介绍。

● 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
- 1x: 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_{hi} 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

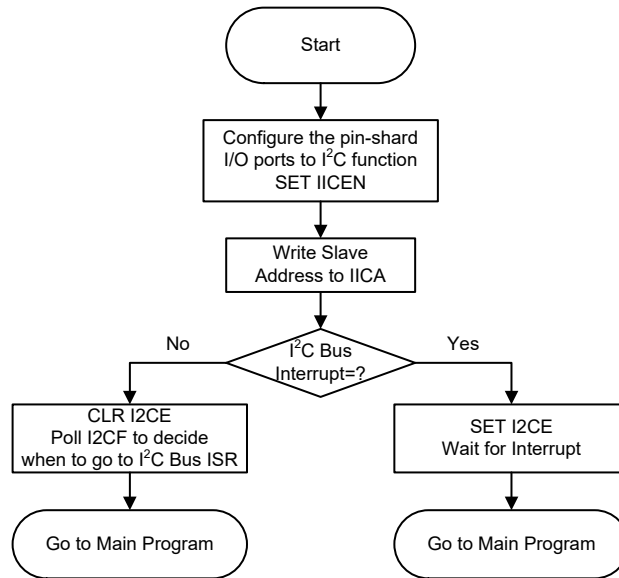
此位是 I²C 总线发送应答标志位。从机接收完 8 位数据之后，该位将在第九个从机时钟时被传到总线上。如果从机想要接收更多的数据，则应在接收数据之前将此位设置为“0”。

Bit 2	<p>SRW: I²C 从机读 / 写位</p> <p>0: 从机应处于接收模式</p> <p>1: 从机应处于发送模式</p> <p>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。</p>
Bit 1	<p>IAMWU: I²C 地址匹配唤醒控制位</p> <p>0: 除能</p> <p>1: 使能</p> <p>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。</p>
Bit 0	<p>RXAK: I²C 总线接收应答标志位</p> <p>0: 从机接收到应答标志</p> <p>1: 从机没有接收到应答标志</p> <p>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。</p>

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 I2CE 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。

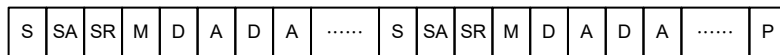
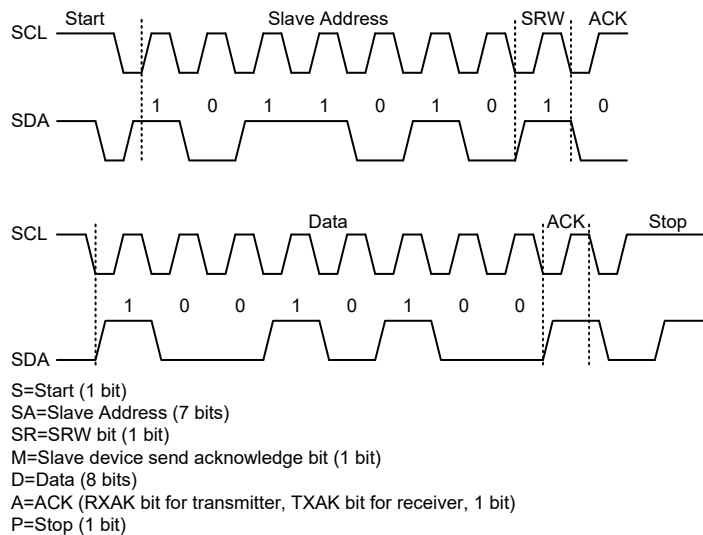
I²C 总线从机地址应答信号

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

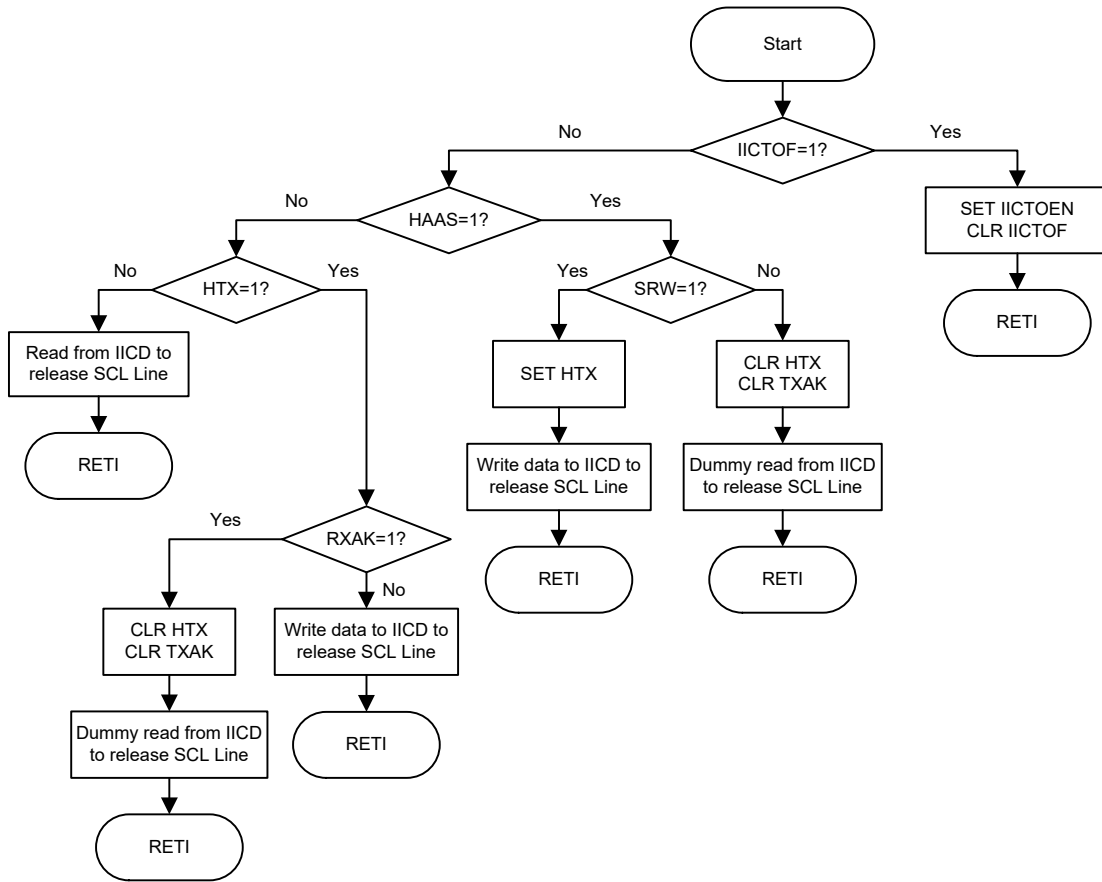
在从机确认接收到从地址后，会进行 8 位宽度的数据传输。这个数据传输顺序是高位在前，低位在后。接收方在接收到 8 位数据后必须发出一个应答信号 (“0”) 以继续接收下一个数据。如果从机发送方没接收到来自主机接收方的应答信号，发送方将释放 SDA 线，此时主机方可发出 STOP 信号以释放 I²C 总线。所传送的数据存储在 IICD 寄存器中。如果设置成发送方，从机必须先将欲传输的数据写到 IICD 寄存器中；如果设置成接收方，从机必须从 IICD 寄存器读取数据。

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。



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。

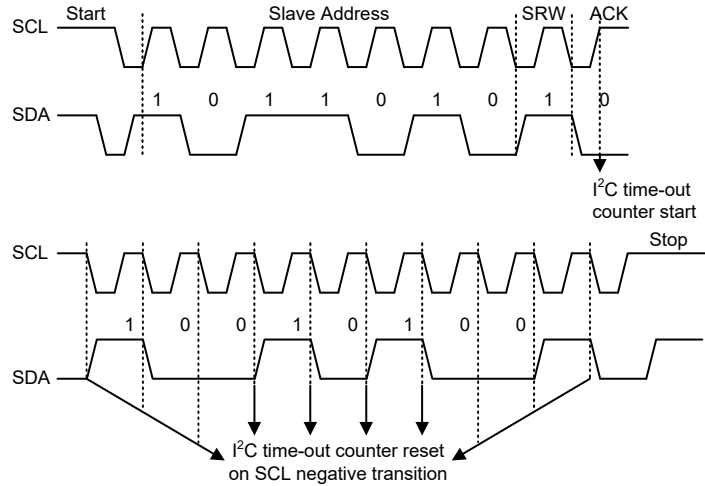
I²C 通信时序图



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，IIC_{TOEN} 位被清零，且 IIC_{TOF} 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	I ² C 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

IIC_{TOF} 标志位由应用程序清零。共有 64 个超时周期，可通过 IIC_{TOC} 寄存器的 IIC_{TOS} 位进行选择。超时周期可通过公式计算： $((1\sim 64)\times 32)/f_{SUB}$ 。由此可得超时周期范围为 1ms~64ms。

● IIC_{TOC} 寄存器

Bit	7	6	5	4	3	2	1	0
Name	IIC _{TOEN}	IIC _{TOF}	IIC _{TOS5}	IIC _{TOS4}	IIC _{TOS3}	IIC _{TOS2}	IIC _{TOS1}	IIC _{TOS0}
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 IIC_{TOEN}: I²C 超时控制位

- 0: 除能
- 1: 使能

Bit 6 IIC_{TOF}: I²C 超时标志位

- 0: 超时未发生
- 1: 超时发生

Bit 5~0 IIC_{TOS5}~IIC_{TOS0}: 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： $((IIC_{TOS}[5:0]+1)\times(32/f_{SUB}))$ 。

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控动作或定时器模块溢出有效，并且产生中断时，系统会暂时中止当前的程序而转到执行相对应的中断服务程序。此单片机提供多个外部中断和内部中断功能，外部中断由 INT 引脚动作产生，而内部中断由各种内部功能，如触控按键、定时器模块、时基等产生。

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专用数据存储中的一系列寄存器控制的。寄存器总的分为三类。第一类是 INTC0~INTC1 寄存器，用于设置基本的中断；第二类是 MF10~MF11 寄存器，用于设置多功能中断；最后一种有 INTEG 寄存器，用于设置外部中断边沿触发类型。

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

功能	使能位	请求标志位	注释
总中断	EMI	—	—
INT 脚	INTE	INTF	—
I ² C	I2CE	I2CF	—
时基	TBE	TBF	—
EEPROM	DEE	DEF	—
多功能	MF _n E	MF _n F	n=0~1
触控按键 TKRCOV	TKRCOVE	TKRCOVF	—
触控按键模块 TKTH	TKTHE	TKTHF	—
CTM	CTMPE	CTMPF	—
	CTMAE	CTMAF	—

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	—	MF1F	MF0F	INTF	MF1E	MF0E	INTE	EMI
INTC1	—	DEF	TBF	I2CF	—	DEE	TBE	I2CE
MF10	—	—	TKTHF	TKRCOVF	—	—	TKTHE	TKRCOVE
MF11	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE

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	—	MF1F	MF0F	INTF	MF1E	MF0E	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 **MF1F**: 多功能中断 1 请求标志位
 0: 无请求
 1: 中断请求

Bit 5 **MF0F**: 多功能中断 0 请求标志位
 0: 无请求
 1: 中断请求

Bit 4 **INTF**: INT 中断请求标志位
 0: 无请求
 1: 中断请求

Bit 3 **MF1E**: 多功能中断 1 控制位
 0: 除能
 1: 使能

Bit 2 **MF0E**: 多功能中断 0 控制位
 0: 除能
 1: 使能

Bit 1 **INTE**: INT 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	—	DEF	TBF	I2CF	—	DEE	TBE	I2CE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

- Bit 7 未定义，读为“0”
- Bit 6 **DEF**: 数据 EEPROM 中断请求标志位
0: 无请求
1: 中断请求
- Bit 5 **TBF**: 时基中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **I2CF**: I²C 中断请求标志位
0: 无请求
1: 中断请求
- Bit 3 未定义，读为“0”
- Bit 2 **DEE**: 数据 EEPROM 中断控制位
0: 除能
1: 使能
- Bit 1 **TBE**: 时基中断控制位
0: 除能
1: 使能
- Bit 0 **I2CE**: I²C 中断控制位
0: 除能
1: 使能

• MFIO 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	TKTHF	TKRCOVF	—	—	TKTHE	TKRCOVE
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 **TKTHF**: 触控按键模块 TKTH 中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **TKRCOVF**: 触控按键 TKRCOV 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 **TKTHE**: 触控按键模块 TKTH 中断控制位
0: 除能
1: 使能
- Bit 0 **TKRCOVE**: 触控按键 TKRCOV 中断控制位
0: 除能
1: 使能

● MF11 寄存器

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

Bit 7~6 未定义，读为“0”

Bit 5 **CTMAF**: CTM 比较器 A 匹配中断请求标志位
 0: 无请求
 1: 中断请求

Bit 4 **CTMPF**: CTM 比较器 P 匹配中断请求标志位
 0: 无请求
 1: 中断请求

Bit 3~2 未定义，读为“0”

Bit 1 **CTMAE**: CTM 比较器 A 匹配中断控制位
 0: 除能
 1: 使能

Bit 0 **CTMPE**: CTM 比较器 P 匹配中断控制位
 0: 除能
 1: 使能

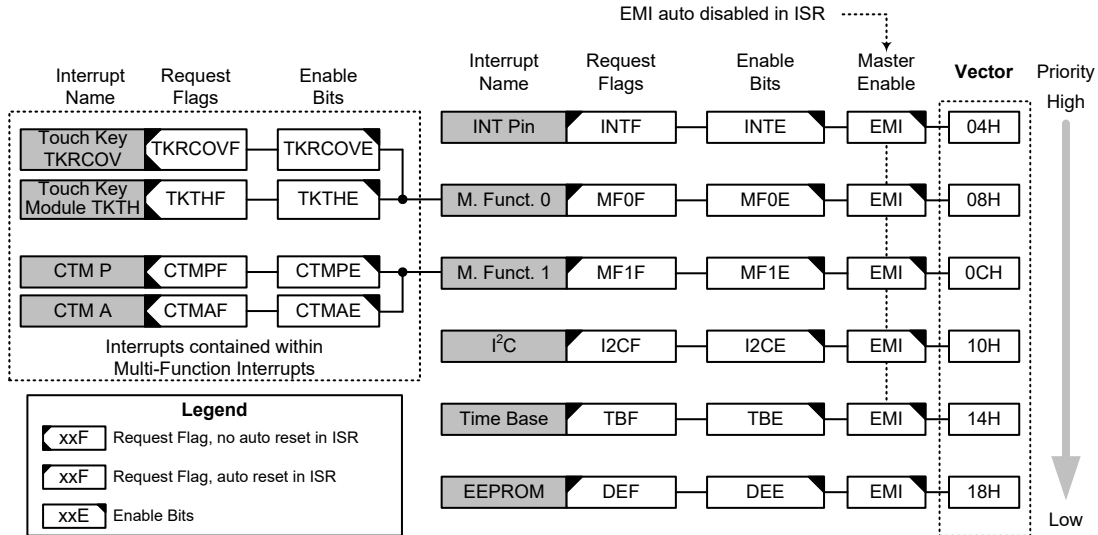
中断操作

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

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

各个中断使能位以及相应的请求标志位，以优先级的次序显示在下图。一些中断源有自己的向量，但是有些中断却共用多功能中断向量。一旦中断子程序被响应，系统将自动清除 EMI 位，所有其它的中断将被屏蔽，这个方式可以防止任何进一步的中断嵌套。其它中断请求可能发生在此期间，虽然中断不会立即响应，但是中断请求标志位会被记录。

如果某个中断服务子程序正在执行时，有另一个中断要求立即响应，那么 EMI 位应在程序进入中断子程序后置位，以允许此中断嵌套。如果堆栈已满，即使此中断使能，中断请求也不会被响应，直到 SP 减少为止。如果要求立刻动作，则堆栈必须避免成为储满状态。请求同时发生时，执行优先级如下流程图所示。所有被置起的中断请求标志都可把单片机从休眠或空闲模式中唤醒，若要防止唤醒动作发生，在单片机进入休眠或空闲模式前应将相应的标志置起。



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端口控制寄存器，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，其上拉电阻选项仍保持有效。

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。

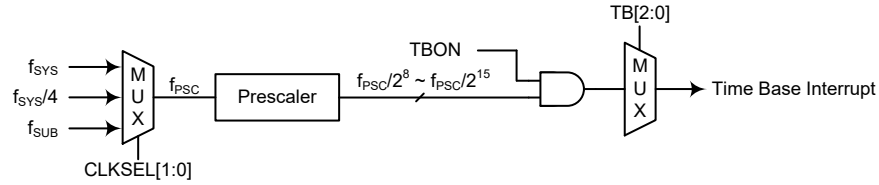
I²C 中断

当一个字节数据已由 I²C 接口接收或发送完，或 I²C 从机地址匹配，或 I²C 超时，中断请求标志 I2CF 被置位，I²C 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 I2CE 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 I2CF 会自动复位且 EMI 将被自动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。当时基中断请求标志 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} 。 f_{PSC} 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 和 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**: 选择时基溢出周期位
 000: $2^8/f_{PSC}$
 001: $2^9/f_{PSC}$
 010: $2^{10}/f_{PSC}$
 011: $2^{11}/f_{PSC}$
 100: $2^{12}/f_{PSC}$
 101: $2^{13}/f_{PSC}$
 110: $2^{14}/f_{PSC}$
 111: $2^{15}/f_{PSC}$

EEPROM 中断

当 EEPROM 写周期结束，EEPROM 中断请求标志 DEF 被置位，EEPROM 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI、EEPROM 中断使能位 DEE 需先被置位。当中断使能，堆栈未滿且 EEPROM 写周期结束时，可跳转至 EEPROM 中断向量程序执行。当 EEPROM 中断响应，EEPROM 中断标志位 DEF 会自动复位且 EMI 将被自动清零以除能其它中断。

多功能中断

此单片机中有两个多功能中断，与其它中断不同，它没有独立源，但由其它现有的中断源构成，即触控按键 TKRCOV 中断、触控按键模块 TKTH 中断和 TM 中断。

当多功能中断中任何一种中断请求标志 MFnF 被置位，多功能中断请求产生。当所包含的任一功能产生中断请求标志，多功能中断标志将置位。若要跳转到相应的中断向量地址，当多功能中断使能，堆栈未满，包括在多功能中断中的任意一个中断发生时，将调用多功能中断向量中的一个子程序。当响应中断服务子程序时，相关的多功能请求标志位会自动复位且 EMI 位会自动清零以除能其它中断。

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

触控按键 TKRCOV 中断

触控按键 TKRCOV 中断属于多功能中断。当触控按键时隙计数器溢出，触控按键 TKRCOV 中断请求标志位 TKRCOVF 被置位，触控按键 TKRCOV 中断请求发生。若要跳转到相应中断向量地址，总中断使能位 EMI、触控按键 TKRCOV 中断使能位 TKRCOVE 和相关多功能中断使能位必须先被置位。当中断使能，堆栈未满且触控按键时隙计数器溢出时，将调用位于多功能中断向量处的子程序。当触控按键 TKRCOV 中断响应，EMI 将被自动清零以除能其它中断，相关多功能中断请求标志也可自动清除，但 TKRCOV 中断请求标志 TKRCOVF 需在应用程序中手动清除。

触控按键模块 TKTH 中断

触控按键模块 TKTH 中断属于多功能中断。当 MKnTHS=0，触控按键模块 16 位 C/F 计数器值小于下限阈值时，或当 MKnTHS=1，触控按键模块 16 位 C/F 计数器值大于上限阈值时，触控按键模块 TKTH 中断请求标志位 TKTHF 被置位，触控按键模块 TKTH 中断请求发生。若要跳转到相应中断向量地址，总中断使能位 EMI、触控按键模块 TKTH 中断使能位 TKTHE 和相关多功能中断使能位必须先被置位。当中断使能，堆栈未满且任何上述的阈值比较条件发生时，将调用位于多功能中断向量处的子程序。当触控按键模块 TKTH 中断响应，EMI 将被自动清零以除能其它中断，相关多功能中断请求标志也可自动清除，但 TKTH 中断请求标志 TKTHF 需在应用程序中手动清除。

TM 中断

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

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

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软件指令清除。

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

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

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

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

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

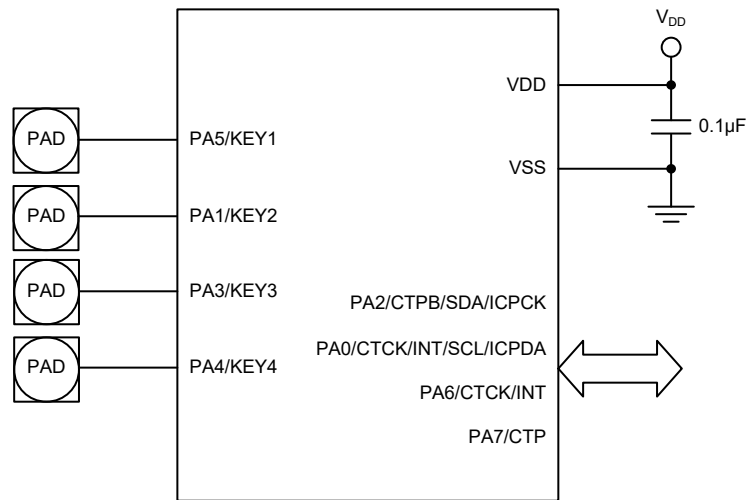
配置选项

配置选项在烧写程序时写入芯片。通过 HT-IDE 的软件开发环境，使用者在开发过程中可以选择配置选项。由于使用的是硬件工具将配置选项烧入单片机，之后无法再通过应用程序修改。所有的选项必须按系统的需要定义，具体内容可参考下表：

序号	选项
振荡器选择	
1	HIRC 频率选择： 1. 2MHz 2. 4MHz 3. 8MHz

注：当 HIRC 被配置为上表中的频率时，HIRCS1~HIRCS0 位必须设置为和选项一样的频率以实现交流电气特性中指定的 HIRC 频率精确度。

应用电路



指令集

简介

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

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

指令周期

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

数据的传送

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

算术运算

算术运算和数据处理是大部分单片机应用所必需具备的能力，在 Holttek 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 Holttek 单片机内部的指令集中。大多数牵涉到数据运算的指令，数据的传送必须通过累加器。在所有逻辑数据运算中，如果运算结果为零，则零标志位将被置位，另外逻辑数据运用形式还有移位指令，例如 RR、RL、RRC 和 RLC 提供了向左或向右移动一位的方法。不同的移位指令可满足不同的应用需要。移位指令常用于串行端口的程序应用，数据可从内部寄存器转移至进位标志位，而此位则可被检验，移位运算还可应用在乘法与除法的运算组成中。

分支和控制转换

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

位运算

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

查表运算

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

其它运算

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

指令集概要

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 and TBLH	2 ^注	无
TABRDC [m]	读取当前页的 ROM 内容, 并送至数据存储器 and TBLH	2 ^注	无
TABRDL [m]	读取最后页的 ROM 内容, 并送至数据存储器 and TBLH	2 ^注	无
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	$ACC \leftarrow ACC + [m] + C$
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	$[m] \leftarrow ACC + [m] + C$
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	$ACC \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC + x$
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	$[m] \leftarrow ACC + [m]$
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	$ACC \leftarrow ACC \text{ "AND" } [m]$
影响标志位	Z

AND A, x	Logical AND immediate data to ACC
指令说明	将累加器中的数据和立即数做逻辑与，结果存放到累加器。
功能表示	$ACC \leftarrow ACC \text{ “AND” } x$
影响标志位	Z
ANDM A, [m]	Logical AND ACC to Data Memory
指令说明	将指定数据存储器内容和累加器中的数据做逻辑与，结果存放到数据存储器。
功能表示	$[m] \leftarrow ACC \text{ “AND” } [m]$
影响标志位	Z
CALL addr	Subroutine call
指令说明	无条件地调用指定地址的子程序，此时程序计数器先加 1 获得下一个要执行的指令地址并压入堆栈，接着载入指定地址并从新地址继续执行程序，由于此指令需要额外的运算，所以为一个 2 周期的指令。
功能表示	$Stack \leftarrow Program\ Counter + 1$ $Program\ Counter \leftarrow addr$
影响标志位	无
CLR [m]	Clear Data Memory
指令说明	将指定数据存储器的内容清零。
功能表示	$[m] \leftarrow 00H$
影响标志位	无
CLR [m].i	Clear bit of Data Memory
指令说明	将指定数据存储器的 i 位内容清零。
功能表示	$[m].i \leftarrow 0$
影响标志位	无
CLR WDT	Clear Watchdog Timer
指令说明	WDT 计数器、暂停标志位 PDF 和看门狗溢出标志位 TO 清零。
功能表示	WDT cleared $TO \ \& \ PDF \leftarrow 0$
影响标志位	TO、PDF

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 \leftarrow 00H TO & PDF \leftarrow 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 \leftarrow 00H TO & PDF \leftarrow 0
影响标志位	TO、PDF
CPL [m]	Complement Data Memory
指令说明	将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1。
功能表示	[m] \leftarrow $\overline{[m]}$
影响标志位	Z
CPLA [m]	Complement Data Memory with result in ACC
指令说明	将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1，而结果被储存回累加器且数据存储器中的内容不变。
功能表示	ACC \leftarrow $\overline{[m]}$
影响标志位	Z

<p>DAA [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Decimal-Adjust ACC for addition with result in Data Memory 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, 结果存放和数据存储器。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。</p> <p>$[m] \leftarrow ACC + 00H$ 或 $[m] \leftarrow ACC + 06H$ 或 $[m] \leftarrow ACC + 60H$ 或 $[m] \leftarrow ACC + 66H$</p> <p>C</p>
<p>DEC [m] 指令说明 功能表示 影响标志位</p>	<p>Decrement Data Memory 将指定数据存储器内容减 1。</p> <p>$[m] \leftarrow [m] - 1$</p> <p>Z</p>
<p>DECA [m] 指令说明 功能表示 影响标志位</p>	<p>Decrement Data Memory with result in ACC 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。</p> <p>$ACC \leftarrow [m] - 1$</p> <p>Z</p>
<p>HALT 指令说明 功能表示 影响标志位</p>	<p>Enter power down mode 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。</p> <p>$TO \leftarrow 0$ $PDF \leftarrow 1$</p> <p>TO、PDF</p>
<p>INC [m] 指令说明 功能表示 影响标志位</p>	<p>Increment Data Memory 将指定数据存储器的内容加 1。</p> <p>$[m] \leftarrow [m] + 1$</p> <p>Z</p>

INCA [m] 指令说明 功能表示 影响标志位	Increment Data Memory with result in ACC 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的内容不变。 $ACC \leftarrow [m] + 1$ Z
JMP addr 指令说明 功能表示 影响标志位	Jump unconditionally 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。 $Program Counter \leftarrow addr$ 无
MOV A, [m] 指令说明 功能表示 影响标志位	Move Data Memory to ACC 将指定数据存储器的内容复制到累加器。 $ACC \leftarrow [m]$ 无
MOV A, x 指令说明 功能表示 影响标志位	Move immediate data to ACC 将 8 位立即数载入累加器。 $ACC \leftarrow x$ 无
MOV [m], A 指令说明 功能表示 影响标志位	Move ACC to Data Memory 将累加器的内容复制到指定的数据存储器。 $[m] \leftarrow ACC$ 无
NOP 指令说明 功能表示 影响标志位	No operation 空操作，接下来顺序执行下一条指令。 $PC \leftarrow PC + 1$ 无
OR A, [m] 指令说明 功能表示 影响标志位	Logical OR Data Memory to ACC 将累加器中的数据和指定的数据存储器内容逻辑或，结果存放到累加器。 $ACC \leftarrow ACC \text{ "OR" } [m]$ Z

<p>ORA, x 指令说明 功能表示 影响标志位</p>	<p>Logical OR immediate data to ACC 将累加器中的数据和立即数逻辑或，结果存放到累加器。 $ACC \leftarrow ACC \text{ "OR" } x$ Z</p>
<p>ORM A, [m] 指令说明 功能表示 影响标志位</p>	<p>Logical OR ACC to Data Memory 将存在指定数据存储器的数据和累加器逻辑或，结果放到数据存储器。 $[m] \leftarrow ACC \text{ "OR" } [m]$ Z</p>
<p>RET 指令说明 功能表示 影响标志位</p>	<p>Return from subroutine 将堆栈寄存器中的程序计数器值恢复，程序由取回的地址继续执行。 $Program\ Counter \leftarrow Stack$ 无</p>
<p>RET A, x 指令说明 功能表示 影响标志位</p>	<p>Return from subroutine and load immediate data to ACC 将堆栈寄存器中的程序计数器值恢复且累加器载入指定的立即数，程序由取回的地址继续执行。 $Program\ Counter \leftarrow Stack$ $ACC \leftarrow x$ 无</p>
<p>RETI 指令说明 功能表示 影响标志位</p>	<p>Return from interrupt 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。 $Program\ Counter \leftarrow Stack$ $EMI \leftarrow 1$ 无</p>
<p>RL [m] 指令说明 功能表示 影响标志位</p>	<p>Rotate Data Memory left 将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。 $[m].(i+1) \leftarrow [m].i \ (i=0\sim6)$ $[m].0 \leftarrow [m].7$ 无</p>

RLA [m]	Rotate Data Memory left with result in ACC
指令说明	将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位，结果送到累加器，而指定数据存储器的内容保持不变。
功能表示	$ACC.(i+1) \leftarrow [m].i \ (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
影响标志位	无
RLC [m]	Rotate Data Memory Left through Carry
指令说明	将指定数据存储器的内容连同进位标志左移 1 位，第 7 位取代进位标志且原本的进位标志移到第 0 位。
功能表示	$[m].(i+1) \leftarrow [m].i \ (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位	C
RLC A [m]	Rotate Data Memory left through Carry with result in ACC
指令说明	将指定数据存储器的内容连同进位标志左移 1 位，第 7 位取代进位标志且原本的进位标志移到第 0 位，移位结果送回累加器，但是指定数据寄存器的内容保持不变。
功能表示	$ACC.(i+1) \leftarrow [m].i \ (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位	C
RR [m]	Rotate Data Memory right
指令说明	将指定数据存储器的内容循环右移 1 位且第 0 位移到第 7 位。
功能表示	$[m].i \leftarrow [m].(i+1) \ (i=0\sim6)$ $[m].7 \leftarrow [m].0$
影响标志位	无
RRA [m]	Rotate Data Memory right with result in ACC
指令说明	将指定数据存储器的内容循环右移 1 位，第 0 位移到第 7 位，移位结果存放到累加器，而指定数据存储器的内容保持不变。
功能表示	$ACC.i \leftarrow [m].(i+1) \ (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
影响标志位	无

RRC [m]	Rotate Data Memory right through Carry
指令说明	将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位。
功能表示	$[m].i \leftarrow [m].(i+1) (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位	C
RRCA [m]	Rotate Data Memory right through Carry with result in ACC
指令说明	将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位，移位结果送回累加器，但是指定数据寄存器的内容保持不变。
功能表示	$ACC.i \leftarrow [m].(i+1) (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位	C
SBC A, [m]	Subtract Data Memory from ACC with Carry
指令说明	将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$ACC \leftarrow ACC - [m] - \bar{C}$
影响标志位	OV、Z、AC、C、SC、CZ
SBCM A, [m]	Subtract Data Memory from ACC with Carry and result in Data Memory
指令说明	将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$[m] \leftarrow ACC - [m] - \bar{C}$
影响标志位	OV、Z、AC、C、SC、CZ
SDZ [m]	Skip if Decrement Data Memory is 0
指令说明	将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	$[m] \leftarrow [m] - 1$ ，如果 $[m]=0$ 跳过下一条指令执行
影响标志位	无

SDZA [m] 指令说明	Decrement data memory and place result in ACC,skip if 0 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，此结果将存放累加器，但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	$ACC \leftarrow [m] - 1$ ，如果 $ACC=0$ 跳过下一条指令执行
影响标志位	无
SET [m] 指令说明	Set Data Memory 将指定数据存储器的每一位设置为 1。
功能表示	$[m] \leftarrow FFH$
影响标志位	无
SET [m].i 指令说明	Set bit of Data Memory 将指定数据存储器的第 i 位置位为 1。
功能表示	$[m].i \leftarrow 1$
影响标志位	无
SIZ [m] 指令说明	Skip if increment Data Memory is 0 将指定的数据存储器内容加 1，判断是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	$[m] \leftarrow [m] + 1$ ，如果 $[m]=0$ 跳过下一条指令执行
影响标志位	无
SIZA [m] 指令说明	Skip if increment Data Memory is zero with result in ACC 将指定数据存储器内容加 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果会被存放累加器，但是指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	$ACC \leftarrow [m] + 1$ ，如果 $ACC=0$ 跳过下一条指令执行
影响标志位	无

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	$ACC \leftarrow ACC - [m]$
影响标志位	OV、Z、AC、C、SC、CZ
SUBM A, [m]	Subtract Data Memory from ACC with result in Data Memory
指令说明	将累加器的内容减去指定数据存储器的数据，结果存放到指定的数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$[m] \leftarrow ACC - [m]$
影响标志位	OV、Z、AC、C、SC、CZ
SUB A, x	Subtract immediate Data from ACC
指令说明	将累加器的内容减去立即数，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$ACC \leftarrow ACC - x$
影响标志位	OV、Z、AC、C、SC、CZ
SWAP [m]	Swap nibbles of Data Memory
指令说明	将指定数据存储器的低 4 位和高 4 位互相交换。
功能表示	$[m].3\sim[m].0 \leftrightarrow [m].7\sim[m].4$
影响标志位	无
SWAPA [m]	Swap nibbles of Data Memory with result in ACC
指令说明	将指定数据存储器的低 4 位与高 4 位互相交换，再将结果存放到累加器且指定数据寄存器的数据保持不变。
功能表示	$ACC.3\sim ACC.0 \leftarrow [m].7\sim[m].4$ $ACC.7\sim ACC.4 \leftarrow [m].3\sim[m].0$
影响标志位	无

SZ [m]	Skip if Data Memory is 0
指令说明	判断指定数据存储器的内容是否为 0，若为 0，则程序跳下一条指令执行。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	$[m] \leftarrow$ 程序代码 (低字节) $TBLH \leftarrow$ 程序代码 (高字节)
影响标志位	无
XORA, [m]	Logical XOR Data Memory to ACC
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或, 结果存放到累加器。
功能表示	$ACC \leftarrow ACC \text{ "XOR" } [m]$
影响标志位	Z
XORM A, [m]	Logical XOR ACC to Data Memory
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或, 结果放到数据存储器。
功能表示	$[m] \leftarrow ACC \text{ "XOR" } [m]$
影响标志位	Z
XORA, x	Logical XOR immediate data to ACC
指令说明	将累加器的数据与立即数逻辑异或, 结果存放到累加器。
功能表示	$ACC \leftarrow ACC \text{ "XOR" } x$
影响标志位	Z

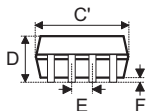
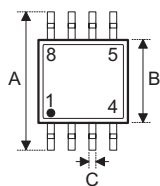
封装信息

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

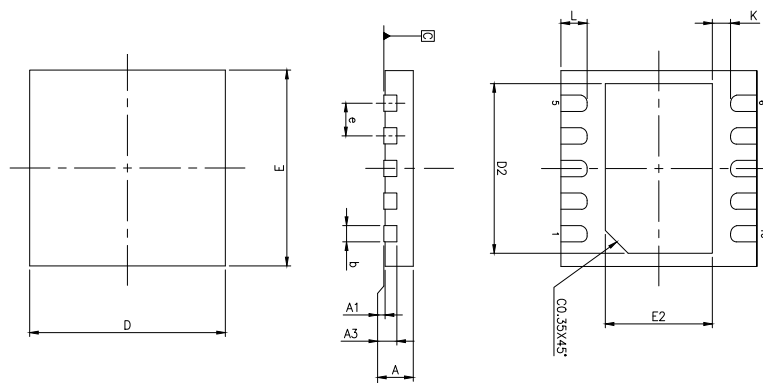
8-pin SOP (150mil) 外形尺寸



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°

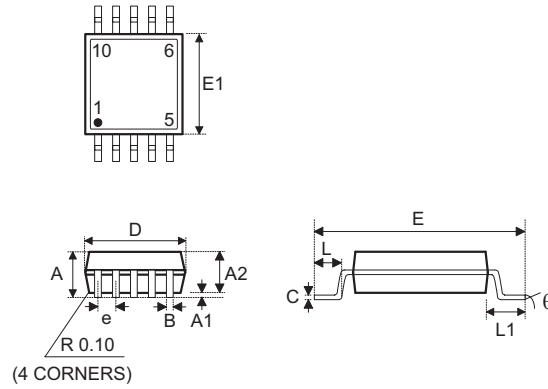
10-pin DFN (3mm×3mm×0.75mm) 外形尺寸



符号	尺寸 (单位: inch)		
	最小值	典型值	最大值
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3	—	0.080 BSC	—
b	0.007	0.010	0.012
D	—	0.118 BSC	—
E	—	0.118 BSC	—
e	—	0.020 BSC	—
D2	0.087	0.091	0.093
E2	0.061	0.065	0.067
L	0.012	0.016	0.020
K	0.008	—	—

符号	尺寸 (单位: mm)		
	最小值	典型值	最大值
A	0.700	0.750	0.800
A1	0.000	0.020	0.050
A3	—	0.203 BSC	—
b	0.180	0.250	0.300
D	—	3.000 BSC	—
E	—	3.000 BSC	—
e	—	0.500 BSC	—
D2	2.200	2.300	2.350
E2	1.550	1.650	1.700
L	0.300	0.400	0.450
K	0.200	—	—

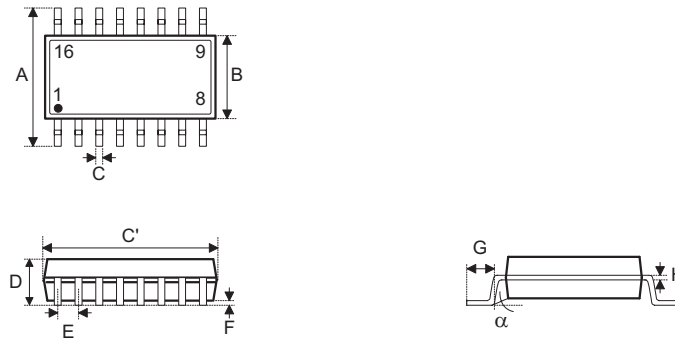
10-pin MSOP 外形尺寸



符号	尺寸 (单位: inch)		
	最小值	典型值	最大值
A	—	—	0.043
A1	0.000	—	0.006
A2	0.030	0.033	0.037
B	0.007	—	0.013
C	0.003	—	0.009
D	—	0.118 BSC	—
E	—	0.193 BSC	—
E1	—	0.118 BSC	—
e	—	0.020 BSC	—
L	0.016	0.024	0.031
L1	—	0.037 BSC	—
y	—	0.004	—
θ	0°	—	8°

符号	尺寸 (单位: mm)		
	最小值	典型值	最大值
A	—	—	1.10
A1	0.00	—	0.15
A2	0.75	0.85	0.95
B	0.17	—	0.33
C	0.08	—	0.23
D	—	3.00 BSC	—
E	—	4.90 BSC	—
E1	—	3.00 BSC	—
e	—	0.50 BSC	—
L	0.40	0.60	0.80
L1	—	0.95 BSC	—
y	—	0.10	—
θ	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°

Copyright® 2019 by HOLTEK SEMICONDUCTOR INC.

使用指南中所出现的信息在出版当时相信是正确的，然而 **Holtek** 对于说明书的使用不负任何责任。文中提到的应用目的仅仅是用来做说明，**Holtek** 不保证或表示这些没有进一步修改的应用将是适当的，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。**Holtek** 产品不授权用于救生、维生从机或系统中做为关键从机。**Holtek** 拥有不事先通知而修改产品的权利，对于最新的信息，请参考我们的网址 <http://www.holtek.com/zh/>。