



帶 LED 驅動功能 A/D 型觸控鍵 Flash 單片機

BS84B06A-3

版本 : V1.41 日期 : 2021-11-10

www.holtek.com

目 录

特性	6
CPU 特性	6
周边特性	6
开发工具	6
概述	7
方框图	7
引脚图	8
引脚说明	9
极限参数	10
直流电气特性	10
交流电气特性	12
感应振荡器电气特性	13
A/D 转换器电气特性	17
上电复位特性	18
系统结构	18
时序和流水线结构	18
程序计数器	19
堆栈	20
算术逻辑单元 – ALU	20
Flash 程序存储器	21
结构	21
特殊向量	21
查表	21
查表范例	21
在线烧录	22
片上调试	23
数据存储器	24
结构	24
特殊功能寄存器	26
间接寻址寄存器 – IAR0, IAR1	26
间接寻址指针 – MP0, MP1	26
存储区指针 – BP	26
累加器 – ACC	27
程序计数器低字节寄存器 – PCL	27
表格寄存器 – TBLP, TBHP, TBLH	27
状态寄存器 – STATUS	27
EEPROM 数据寄存器	29
EEPROM 数据寄存器结构	29

EEPROM 寄存器	29
从 EEPROM 中读取数据	30
写数据到 EEPROM	30
写保护	31
EEPROM 中断	31
编程注意事项	31
振荡器	32
振荡器概述	32
系统时钟配置	32
内部高速 RC 振荡器 – HIRC	33
工作模式和系统时钟	33
系统时钟	33
控制寄存器	34
系统工作模式	35
工作模式切换	37
待机电流的注意事项	39
唤醒	39
看门狗定时器	40
看门狗定时器时钟源	40
看门狗定时器控制寄存器	40
看门狗定时器操作	41
复位和初始化	42
复位功能	42
复位初始状态	44
输入 / 输出端口	47
输入 / 输出寄存器列表	47
上拉电阻	47
PA 口唤醒	48
输入 / 输出端口控制寄存器	48
输入 / 输出引脚源电流选择	49
输入 / 输出引脚结构	51
编程注意事项	52
定时 / 计数器	52
配置定时 / 计数器输入时钟源	52
定时 / 计数器寄存器 – TMR	52
定时 / 计数控制寄存器 – TMRC	53
定时器操作	53
预分频器	54
编程注意事项	54
A/D 转换器	54
A/D 简介	54
A/D 转换寄存器介绍	55
A/D 转换器数据寄存器 – ADRL, ADRH	55

A/D 转换控制寄存器 – ADCR0, ADCR1, ACERL	55
A/D 操作	57
A/D 输入引脚	58
A/D 转换步骤	59
编程注意事项	60
A/D 转换功能	60
A/D 转换应用范例	61
触控按键功能	62
触控按键结构	62
触控按键寄存器	64
触控按键操作	69
触控按键中断	70
编程注意事项	70
串行接口模块 – SIM	71
SPI 接口	71
I ² C 接口	76
中断	84
中断寄存器	84
中断操作	86
外部中断	87
时基中断	87
定时 / 计数器中断	88
EEPROM 中断	88
触控按键中断	88
串行接口模块中断	89
A/D 转换器中断	89
中断唤醒功能	89
编程注意事项	89
应用电路	90
指令集	91
简介	91
指令周期	91
数据的传送	91
算术运算	91
逻辑和移位运算	91
分支和控制转换	92
位运算	92
查表运算	92
其它运算	92
指令集概要	93
惯例	93
指令定义	95

封装信息106

 16-pin NSOP (150mil) 外形尺寸 107

 20-pin SOP (300mil) 外形尺寸 108

特性

CPU 特性

- 工作电压
 - ◆ $f_{\text{SYS}}=8\text{MHz}$: 2.7V~5.5V
 - ◆ $f_{\text{SYS}}=12\text{MHz}$: 2.7V~5.5V
 - ◆ $f_{\text{SYS}}=16\text{MHz}$: 4.5V~5.5V
- $V_{\text{DD}}=5\text{V}$, 系统时钟为 16MHz 时, 指令周期为 $0.25\mu\text{s}$
- 提供暂停和唤醒功能, 以降低功耗
- 完全集成高速和低速振荡器:
 - ◆ 高速内部 RC – HIRC: 8/12/16MHz
 - ◆ 低速内部 RC – LIRC: 32kHz
- 多种工作模式: 正常、低速、空闲和休眠
- 所有指令都可在 1 或 2 个指令周期内完成
- 查表指令
- 61 条指令
- 6 层堆栈
- 位操作指令

周边特性

- Flash 程序存储: $3\text{K}\times 16$
- RAM 数据存储: 288×8
- True EEPROM 存储器: 64×8
- 看门狗定时器功能
- 18 个双向 I/O 口
- 外部中断口与 I/O 引脚共用
- 8 位定时 / 计数器
- 单时基功能可提供固定时间的中断信号
- 多通道 12 位分辨精度的 A/D 转换器
- 串行接口模块 – SIM, 用于 SPI 或 I²C 通信
- PMOS 源电流可调节
- 低电压复位功能
- 完全集成 6 个触控按键功能 – 无需外接元件
- 封装类型: 16-pin NSOP, 20-pin SOP

开发工具

为加快产品开发并简化单片机参数设置, Holtek 提供相关开发工具, 用户可通过以下链接下载:

<https://www.holtek.com.cn/esk-bs-210>

概述

该单片机是 A/D 型具有 8 位高性能精简指令集的 Flash 型单片机，具有完全集成的触控按键功能。其内部提供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应用产品中触摸按键功能的执行。

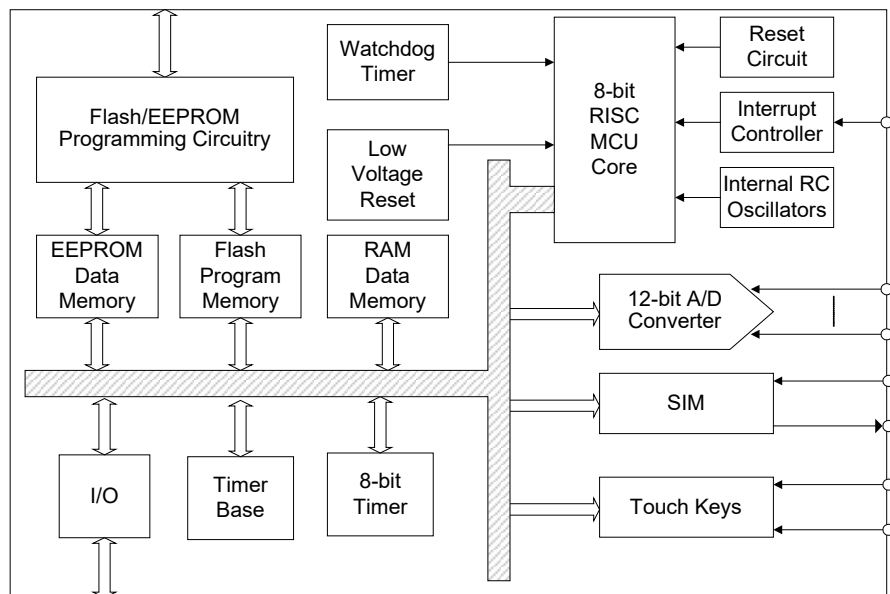
在模拟特性方面，这款单片机包含一个多通道 12 位 A/D 转换器。保护功能方面，包含内部看门狗定时器和低电压复位等内部保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地进行。

触控按键功能完全集成于单片机内部，无需外部元件即可实现触控按键功能。存储器方面，除 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。

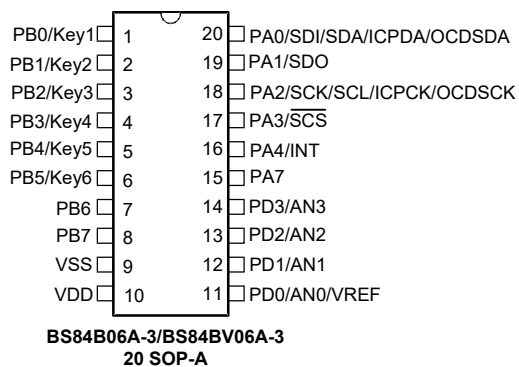
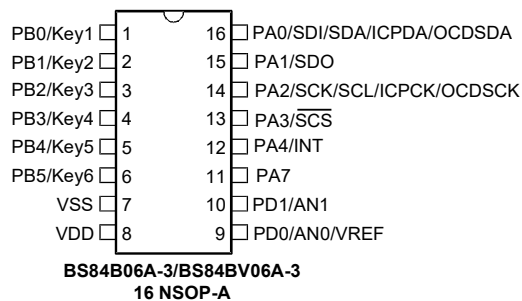
该单片机内建完整的振荡器系统，提供高速和低速振荡器功能选项，无需外围元器件。其不同工作模式之间动态切换的能力，为用户提供了优化单片机操作和减少功耗的手段。这款单片机内建完整的 SPI 和 I²C 功能，为设计者提供了一个易与外部硬件通信的接口。外加 I/O 使用灵活，定时 / 计数器功能等其它特性，进一步增强了设备的功能性和灵活性。

这款触控按键单片机可以广泛应用于各种产品中，例如电子测量仪器、家用电器、电子控制工具等方面。

方框图



引脚图



注：OCSDA 和 OCDSCK 引脚用于 OCDS 功能，仅适用于 BS84B06A-3 的 EV 芯片 BS84BV06A-3。

引脚说明

每个引脚的功能如下表所述，而引脚配置的详细内容见规格书其它章节。

引脚名称	功能	OP	I/T	O/T	引脚说明
PA0/SDI/ SDA/ ICPDA/ OCDSDA	PA0	PAWU PAPU	ST	CMOS	通用 I/O 口，通过寄存器设置上拉电阻和唤醒功能
	SDI	SIMC0	ST	—	SPI 数据输入
	SDA	SIMC0	ST	NMOS	I ² C 数据
	ICPDA	—	ST	CMOS	在线烧录数据 / 地址引脚
	OCDSDA	—	ST	CMOS	片上调试地址 / 数据引脚，仅用于 EV 芯片
PA1/SDO	PA1	PAWU PAPU	ST	CMOS	通用 I/O 口，通过寄存器设置上拉电阻和唤醒功能
	SDO	SIMC0	—	CMOS	SPI 数据输出
PA2/SCK/ SCL/ ICPCK/ OCDCK	PA2	PAWU PAPU	ST	CMOS	通用 I/O 口，通过寄存器设置上拉电阻和唤醒功能
	SCK	SIMC0	ST	CMOS	SPI 串行时钟
	SCL	SIMC0	ST	NMOS	I ² C 时钟
	ICPCK	—	ST	—	在线烧录时钟输入
	OCDCK	—	ST	—	片上调试时钟输入引脚，仅用于 EV 芯片
PA3/ $\overline{\text{SCS}}$	PA3	PAWU PAPU	ST	CMOS	通用 I/O 口，通过寄存器设置上拉电阻和唤醒功能
	$\overline{\text{SCS}}$	SIMC0 SIMC2	ST	CMOS	SPI 从机选择
PA4/INT	PA4	PAWU PAPU	ST	CMOS	通用 I/O 口，通过寄存器设置上拉电阻和唤醒功能
	INT	INTEG	ST	—	外部中断
PA7	PA7	PAWU PAPU	ST	CMOS	通用 I/O 口，通过寄存器设置上拉电阻和唤醒功能
PB0/KEY1~ PB3/KEY4	PB0~PB3	PBPU	ST	CMOS	通用 I/O 口，通过寄存器设置上拉电阻
	KEY1~KEY4	TKM0C1	NSI	—	触控按键输入
PB4/KEY5~ PB5/KEY6	PB4~PB5	PBPU	ST	CMOS	通用 I/O 口，通过寄存器设置上拉电阻
	KEY5~KEY6	TKM1C1	NSI	—	触控按键输入
PB6~PB7	PB6~PB7	PBPU	ST	CMOS	通用 I/O 口，通过寄存器设置上拉电阻
PD0/AN0/ VREF	PD0	PDPU	ST	CMOS	通用 I/O 口，通过寄存器设置上拉电阻
	AN0	ACERL	AN	—	A/D 转换器输入
	VREF	ADCR1	AN	—	A/D 转换器参考电压输入
PD1/AN1~ PD3/AN3	PD1~PD3	PDPU	ST	CMOS	通用 I/O 口，通过寄存器设置上拉电阻
	AN1~AN3	ACERL	AN	—	A/D 转换器输入
VDD	VDD	—	PWR	—	电源电压正极
VSS	VSS	—	PWR	—	电源电压负极

注：I/T：输入类型； O/T：输出类型 OP：通过寄存器设置； AN：模拟输入
PWR：电源； ST：施密特触发输入； NSI：非标准输入
CMOS：CMOS 输出； NMOS：NMOS 输出

极限参数

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

$T_a=25^{\circ}C$

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压 (HIRC)	—	$f_{SYS}=8MHz$	2.7	—	5.5	V
			$f_{SYS}=12MHz$	2.7	—	5.5	V
			$f_{SYS}=16MHz$	4.5	—	5.5	V
I_{DD}	工作电流, 正常模式 (HIRC OSC, $f_{SYS}=f_H$, $f_S=f_{SUB}=f_{LIRC}$)	3V	无负载, $f_H=8MHz$, A/D off,	—	1.2	1.8	mA
		5V	WDT 使能, LVR 使能	—	2.2	3.3	mA
		3V	无负载, $f_H=12MHz$, A/D	—	1.6	2.4	mA
		5V	off, WDT 使能, LVR 使能	—	3.3	5.0	mA
		5V	无负载, $f_H=16MHz$, A/D	—	4.0	6.0	mA
	工作电流, 正常模式 (HIRC OSC, $f_{SYS}=f_L$, $f_S=f_{SUB}=f_{LIRC}$)	3V	无负载, $f_H=12MHz$, $f_L=f_H/2$, A/D off,	—	1.2	2.0	mA
		5V	WDT 使能, LVR 使能	—	2.2	3.3	mA
		3V	无负载, $f_H=12MHz$, $f_L=f_H/64$, A/D off,	—	0.8	1.2	mA
		5V	WDT 使能, LVR 使能	—	1.5	2.3	mA
	工作电流, 低速模式 (LIRC OSC, $f_{SYS}=f_L=f_{LIRC}$, $f_S=f_{SUB}=f_{LIRC}$)	3V	无负载, A/D off,	—	50	100	μA
		5V	WDT 使能, LVR 使能	—	70	150	μA

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
I _{STB}	IDLE1 模式静态电流 (HIRC OSC, f _{SYS} =f _H , f _S =f _{SUB} =f _{LIRC})	3V	无负载, HALT, A/D off, WDT 使能, f _{SYS} =12MHz	—	0.9	1.4	mA
		5V		—	1.4	2.1	mA
	IDLE0 模式静态电流 (HIRC OSC, f _{SYS} =off, f _S =f _{SUB} =f _{LIRC})	3V	无负载, HALT, A/D off, WDT 使能, f _{SYS} =12MHz,	—	1.4	3.0	μA
		5V		—	2.7	5.0	μA
	IDLE1 模式静态电流 (HIRC OSC, f _{SYS} =f _L , f _S =f _{SUB} =f _{LIRC})	3V	无负载, HALT, A/D off, WDT 使能, f _{SYS} =12MHz/64	—	0.7	1.1	mA
		5V		—	1.4	2.1	mA
	IDLE0 模式静态电流 (HIRC OSC, f _{SYS} =off, f _S =f _{SUB} =f _{LIRC})	3V	无负载, HALT, A/D off, WDT 使能, f _{SYS} =12MHz/64	—	1.3	3.0	μA
		5V		—	2.3	5.0	μA
	IDLE1 模式静态电流 (LIRC OSC, f _{SYS} =f _L =f _{LIRC} , f _S =f _{SUB} =f _{LIRC})	3V	无负载, HALT, A/D off, WDT 使能, f _{SYS} =32kHz	—	1.9	4.0	μA
		5V		—	3.3	7.0	μA
	IDLE0 模式静态电流 (LIRC OSC, f _{SYS} =off, f _S =f _{SUB} =f _{LIRC})	3V	无负载, HALT, A/D off, WDT 使能, f _{SYS} =32kHz	—	1.3	3.0	μA
		5V		—	2.4	5.0	μA
	休眠模式静态电流 (HIRC OSC, f _{SYS} =off, f _S =f _{SUB} =f _{LIRC})	3V	无负载, HALT, A/D off, WDT 使能, f _{SYS} =12MHz	—	1.4	5	μA
		5V		—	2.7	10	μA
	休眠模式静态电流 (LIRC OSC, f _{SYS} =off, f _S =f _{SUB} =f _{LIRC})	3V	无负载, HALT, A/D off, WDT 使能, f _{SYS} =32kHz	—	1.3	3.0	μA
		5V		—	2.4	5.0	μA
V _{IL}	输入 / 输出口或其它输入 引脚低电平输入电压	5V	—	0	—	1.5	V
		—	—	0	—	0.2V _{DD}	V
V _{IH}	输入 / 输出口或其它输入 引脚高电平输入电压	5V	—	3.5	—	5.0	V
		—	—	0.8V _{DD}	—	V _{DD}	V
V _{LVR}	低电压复位电压	—	LVR 使能, V _{LVR} 固定为 2.55V	-5%	2.55	+5%	V
I _{OL}	输入 / 输出口灌电流 (PA/PB/PD)	3V	V _{OL} =0.1V _{DD}	16	32	—	mA
		5V	V _{OL} =0.1V _{DD}	32	64	—	mA
I _{OH}	输入 / 输出口源电流 (PA/PB/PD)	3V	V _{OH} =0.9V _{DD} , PxPS=00	-1.0	-2.0	—	mA
		5V		-2.0	-4.0	—	mA
		3V	V _{OH} =0.9V _{DD} , PxPS=01	-1.75	-3.5	—	mA
		5V		-3.5	-7.0	—	mA
		3V	V _{OH} =0.9V _{DD} , PxPS=10	-2.5	-5.0	—	mA
		5V		-5.0	-10	—	mA
		3V	V _{OH} =0.9V _{DD} , PxPS=11	-5.5	-11	—	mA
		5V		-11	-22	—	mA
R _{PH}	输入 / 输出口上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

交流电气特性

Ta=25℃

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
f _{SYS}	系统时钟 (HIRC)	3V/5V	Ta=25℃	-2%	8	+2%	MHz
				-2%	12	+2%	MHz
		5V		-2%	16	+2%	MHz
f _{TIMER}	定时器输入脉宽	2.7V~5.5V	—	—	—	8	MHz
		2.7V~5.5V		—	—	12	MHz
		4.5V~5.5V		—	—	16	MHz
f _{LIRC}	系统时钟 (32K)	5V	Ta=25℃	-10%	32	+10%	kHz
t _{SST}	系统启动时间 (由 HALT 模式唤醒)	—	f _{SYS} = HIRC OSC	16	—	—	t _{SYS}
			f _{SYS} = LIRC OSC	2	—	—	
	系统启动时间 (由 HALT 模式唤醒, HALT 时 f _{SYS} on)	—	—	2	—	—	
t _{INT}	中断脉宽	—	—	1	5	10	μs
t _{LVR}	低电压复位脉宽	—	—	120	240	480	μs
t _{EERD}	EEPROM 读周期	—	—	1	2	4	t _{SYS}
t _{EEWR}	EEPROM 写周期	—	—	1	2	4	ms

注：1. t_{SYS} = 1/f_{SYS}。

2. 为了保持内部 HIRC 振荡器频率的准确性，需要在 VDD 和 VSS 之间接入一个 0.1μF 的去耦电容，并且尽可能地靠近单片机。

感应振荡器电气特性

Ta=25°C

触控按键 RC OSC = 500kHz

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
I _{KEYOSC}	感应 (KEY) 振荡器工作电流	3V	*f _{SENOSC} =500kHz, MnFILEN=0	—	30	60	μA
		5V		—	60	120	μA
		3V	*f _{SENOSC} =500kHz, MnFILEN=1	—	40	80	μA
		5V		—	80	160	μA
I _{REFOSC}	参考振荡器工作电流	3V	*f _{REFOSC} =500kHz, MnTSS=0, MnFILEN=0	—	30	60	μA
		5V		—	60	120	μA
		3V	*f _{REFOSC} =500kHz, MnTSS=0, MnFILEN=1	—	30	60	μA
		5V		—	60	120	μA
		3V	*f _{REFOSC} =500kHz, MnTSS=1, MnFILEN=0	—	30	60	μA
		5V		—	60	120	μA
		3V	*f _{REFOSC} =500kHz, MnTSS=1, MnFILEN=1	—	40	80	μA
		5V		—	80	160	μA
C _{KEYOSC}	感应 (KEY) 振荡器外挂电容值	5V	*f _{SENOSC} =500kHz	5	10	20	pF
C _{REFOSC}	参考振荡器内建电容值	5V	*f _{SENOSC} =500kHz	5	10	20	pF
f _{KEYOSC}	感应 (KEY) 振荡器工作频率	5V	* 外挂电容值 =7,8,9,10,11,12,13,14,15, ... 50pF	100	500	1000	kHz
f _{REFOSC}	参考振荡器工作频率	5V	* 内建电容值 =7,8,9,10,11,12,13,14,15, ... 50pF	100	500	1000	kHz

注：*f_{SENOSC}=500kHz: 调整 KEYn 上的电容值，使得感应振荡器频率为 500kHz。

*f_{REFOSC}=500kHz: 调整参考振荡器内建电容值，使得参考振荡器频率为 500kHz。

触控按键 RC OSC = 1000kHz

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
I _{KEYOSC}	感应 (KEY) 振荡器工作电流	3V	*f _{SENOSC} =1000kHz, M0FILEN=0	—	40	80	μA
		5V	M0FILEN=0	—	80	160	μA
		3V	*f _{SENOSC} =1000kHz, M0FILEN=1	—	60	120	μA
		5V	M0FILEN=1	—	100	200	μA
I _{REFOSC}	参考振荡器工作电流	3V	*f _{REFOSC} =1000kHz, MnTSS=0, MnFILEN=0	—	40	80	μA
		5V	MnTSS=0, MnFILEN=0	—	80	160	μA
		3V	*f _{REFOSC} =1000kHz, MnTSS=0, MnFILEN=1	—	50	100	μA
		5V	MnTSS=0, MnFILEN=1	—	100	200	μA
		3V	*f _{REFOSC} =1000kHz, MnTSS=1, MnFILEN=0	—	40	80	μA
		5V	MnTSS=1, MnFILEN=0	—	80	160	μA
		3V	*f _{REFOSC} =1000kHz, MnTSS=1, MnFILEN=1	—	60	120	μA
		5V	MnTSS=1, MnFILEN=1	—	150	300	μA
C _{KEYOSC}	感应 (KEY) 振荡器 外挂电容值	5V	*f _{SENOSC} =1000kHz	5	10	20	pF
C _{REFOSC}	参考振荡器内建电容 值	5V	*f _{SENOSC} =1000kHz	5	10	20	pF
f _{KEYOSC}	感应 (KEY) 振荡器工作频率	5V	* 外挂电容值 =2,3,4,5,6,7,8,9,10,11,12,13,14, 15, ... 50pF	150	1000	2500	kHz
f _{REFOSC}	参考振荡器工作频率	5V	* 内建电容值 =2,3,4,5,6,7,8,9,10,11,12,13,14, 15, ... 50pF	150	1000	2500	kHz

注：*f_{SENOSC}=1000kHz：调整 KEYn 上的电容值，使得感应振荡器频率为 1000kHz。

*f_{REFOSC}=1000kHz：调整参考振荡器内建电容值，使得参考振荡器频率为 1000kHz。

触控按键 RC OSC =1500kHz

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
I _{KEYOSC}	感应 (KEY) 振荡器工作电流	3V	*f _{SENOSC} =1500kHz, M0FILEN=0	—	60	120	μA
		5V		—	120	240	μA
		3V	*f _{SENOSC} =1500kHz, M0FILEN=1	—	90	180	μA
		5V		—	150	300	μA
I _{REFOSC}	参考振荡器工作电流	3V	*f _{REFOSC} =1500kHz, MnTSS=0, MnFILEN=0	—	60	120	μA
		5V		—	120	240	μA
		3V	*f _{REFOSC} =1500kHz, MnTSS=0, MnFILEN=1	—	60	120	μA
		5V		—	140	280	μA
		3V	*f _{REFOSC} =1500kHz, MnTSS=1, MnFILEN=0	—	60	120	μA
		5V		—	120	240	μA
		3V	*f _{REFOSC} =1500kHz, MnTSS=1, MnFILEN=1	—	90	180	μA
		5V		—	225	450	μA
C _{KEYOSC}	感应 (KEY) 振荡器外挂电容值	5V	*f _{SENOSC} =1500kHz	5	10	20	pF
C _{REFOSC}	参考振荡器内建电容值	5V	*f _{SENOSC} =1500kHz	5	10	20	pF
f _{KEYOSC}	感应 (KEY) 振荡器工作频率	3V	* 外挂电容值 =2,3,4,5,6,7,8,9,10,11,12,13,14, 15, ... 50pF	150	1500	3000	kHz
		5V		150	1500	3000	kHz
f _{REFYOSC}	参考振荡器工作频率	3V	* 内建电容值 =2,3,4,5,6,7,8,9,10,11,12,13,14, 15, ... 50pF	150	1500	3000	kHz
		5V		150	1500	3000	kHz

注：*f_{SENOSC}=1500kHz：调整 KEYn 上的电容值，使得感应振荡器频率为 1500kHz。

*f_{REFOSC}=1500kHz：调整参考振荡器内建电容值，使得参考振荡器频率为 1500kHz。

触控按键 RC OSC =2000kHz

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
I _{KEYOSC}	感应 (KEY) 振荡器工作电流	3V	*f _{SENOSC} =2000kHz, M0FILEN=0	—	80	160	μA
		5V		—	160	320	μA
		3V	*f _{SENOSC} =2000kHz, M0FILEN=1	—	120	240	μA
		5V		—	200	400	μA
I _{REFOSC}	参考振荡器工作电流	3V	*f _{REFOSC} =2000kHz, MnTSS=0, MnFILEN=0	—	80	160	μA
		5V		—	160	320	μA
		3V	*f _{REFOSC} =2000kHz, MnTSS=0, MnFILEN=1	—	80	160	μA
		5V		—	160	320	μA
		3V	*f _{REFOSC} =2000kHz, MnTSS=1, MnFILEN=0	—	80	160	μA
		5V		—	160	320	μA
		3V	*f _{REFOSC} =2000kHz, MnTSS=1, MnFILEN=1	—	120	240	μA
		5V		—	300	600	μA
C _{KEYOSC}	感应 (KEY) 振荡器外挂电容值	5V	*f _{SENOSC} =2000kHz	5	10	20	pF
C _{REFOSC}	参考振荡器内建电容值	5V	*f _{SENOSC} =2000kHz	5	10	20	pF
f _{KEYOSC}	感应 (KEY) 振荡器工作频率	3V	* 外挂电容值 =2,3,4,5,6,7,8,9,10,11,12,13,14, 15, ... 50pF	150	2000	4000	kHz
		5V		150	2000	4000	kHz
f _{REFOSC}	参考振荡器工作频率	3V	* 内建电容值 =2,3,4,5,6,7,8,9,10,11,12,13,14, 15, ... 50pF	150	2000	4000	kHz
		5V		150	2000	4000	kHz

注：*f_{SENOSC}=2000kHz：调整 KEYn 上的电容值，使得感应振荡器频率为 2000kHz。

*f_{REFOSC}=2000kHz：调整参考振荡器内建电容值，使得参考振荡器频率为 2000kHz。

A/D 转换器电气特性

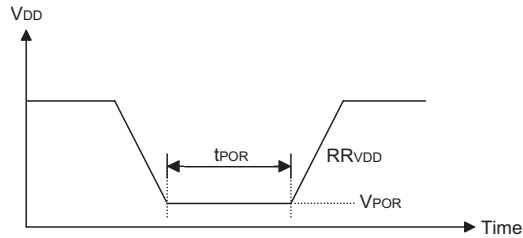
Ta=25°C

符号	参数	测试条件		最小值	典型值	最大值	单位
		V _{DD}	条件				
AV _{DD}	A/D 工作电压	—	—	2.7	—	5.5	V
V _{ADI}	A/D 输入电压	—	—	0	—	V _{REF}	V
V _{REF}	A/D 参考电压	—	—	2.7	—	AV _{DD}	V
V _{BG}	带有缓冲器电压的 Bandgap 参考电压	—	—	-3%	1.19	+3%	V
DNL	A/D 非线性微分误差	2.7V	V _{REF} =AV _{DD} =V _{DD} t _{ADCK} =0.5μs Ta=25°C	-3	—	+3	LSB
		3V					
		5V					
		2.7V	V _{REF} =AV _{DD} =V _{DD} t _{ADCK} =0.5μs Ta=-40°C~85°C	-4	—	+4	LSB
		3V					
		5V					
INL	A/D 非线性积分误差	2.7V	V _{REF} =AV _{DD} =V _{DD} t _{ADCK} =0.5μs Ta=25°C	-4	—	+4	LSB
		3V					
		5V					
		2.7V	V _{REF} =AV _{DD} =V _{DD} t _{ADCK} =0.5μs Ta=-40°C~85°C	-8	—	+8	LSB
		3V					
		5V					
I _{ADC}	打开 A/D 增加的功耗	3V	无负载, t _{ADCK} =0.5μs	—	0.9	1.35	mA
		5V	无负载, t _{ADCK} =0.5μs	—	1.2	1.8	mA
I _{BG}	打开 V _{BG} 参考电压增加的功耗	—	—	—	200	300	μA
t _{ADCK}	A/D 时钟周期	—	—	0.5	—	10	μs
t _{ADC}	A/D 转换时间 (包括 A/D 采样和保持时间)	—	12-bit A/D 转换器	—	16	—	t _{ADCK}
t _{ADS}	A/D 采样时间	—	—	—	4	—	t _{ADCK}
t _{ON2ST}	A/D On-to-Start 时间	—	—	2	—	—	μs
t _{BG}	V _{BG} 稳定时间	—	—	—	—	200	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 _{VDD}	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

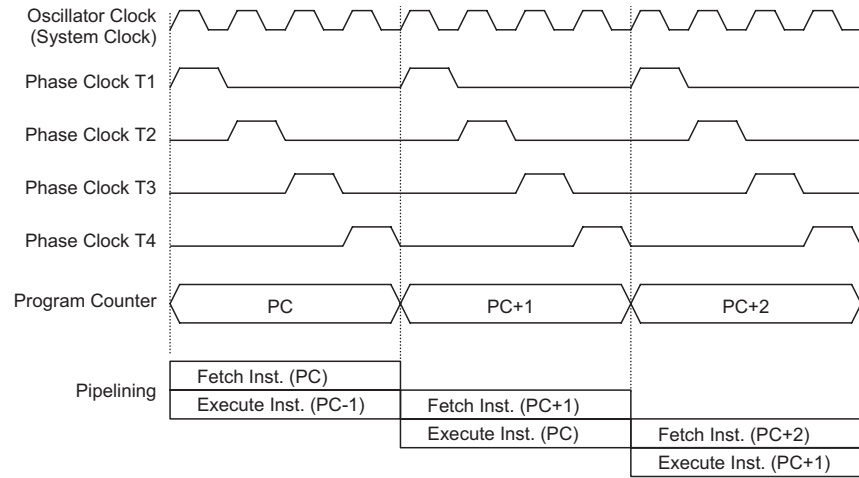


系统结构

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器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供具有较大可靠度和灵活性的 I/O 和 A/D 控制系统时，仅需要少数的外部器件。使得这些单片机适用于低成本和大批量生产控制应用。

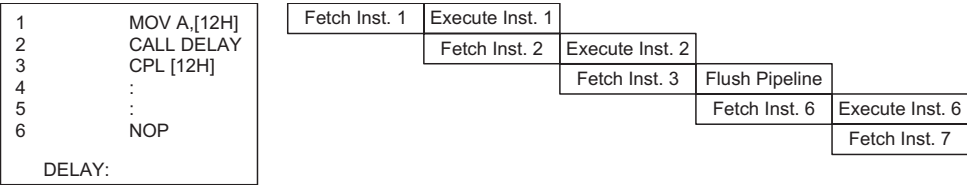
时序和流水线结构

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



系统时序和流水线

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



指令捕捉

程序计数器

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

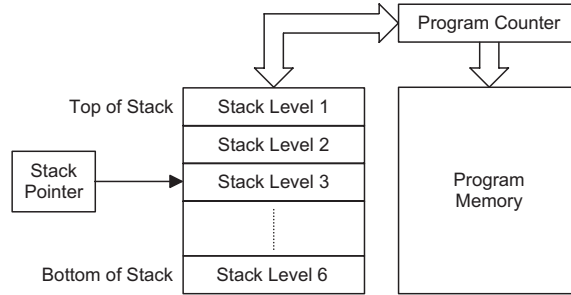
当执行的指令要求跳转到不连续的地址时，如跳转指令、子程序调用、中断或复位等，单片机通过加载所需要的地址到程序寄存器来控制程序，对于条件跳转指令，一旦条件符合，在当前指令执行时取得的下一条指令将会被舍弃，而由一个空指令周期来取代。

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

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

堆栈

堆栈是一个特殊的存储空间，用来存储程序计数器中的内容。此单片机有 6 层堆栈，堆栈既不是数据部分也不是程序空间部分，而且它既不是可读取也不是可写入的。当前层由堆栈指针（SP）加以指示，同样也是不可读写的。在子程序调用或中断响应服务时，程序计数器的内容被压入到堆栈中。当子程序或中断响应结束时，返回指令（RET 或 RETI）使程序计数器从堆栈中重新得到它以前的值。当一个芯片复位后，堆栈指针将指向堆栈顶部。



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

算术逻辑单元 – ALU

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

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

Flash 程序存储器

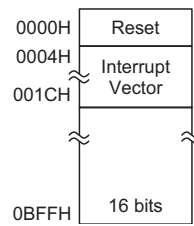
程序存储器用来存放用户代码即储存程序。程序存储器为 Flash 类型意味着可以多次重复编程，方便用户使用同一芯片进行程序的修改。使用适当的单片机编程工具，此单片机提供用户灵活便利的调试方法和项目开发规划及更新。

结构

程序存储器的容量为 $3K \times 16$ 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。

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。



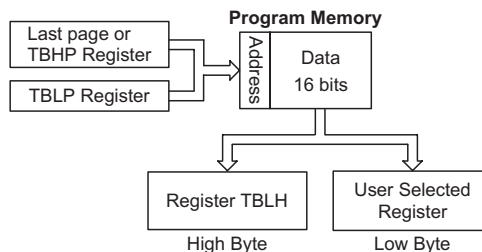
程序存储器结构

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]”指令从程序存储器页来查表读取。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

以下范例说明在该单片机中，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的最后一页。ORG 指令“0B00H”指向的地址是 3K 程序存储器中最后一页的起始地址。表格指针的初始值设为“06H”，这可保证从数据表格读取的第一笔数据位于程序存储器地址 0B06H，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 TBLP 和 TBHP 定义的第一个地址。在这个例子中，表格数据的高字节等于零，而当“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   ; initialize table pointer - note that this address
              ; is referenced

mov tblp, a
mov a, 0Bh   ; initialise high table pointer
mov tbhp, a
:
:
tabrd tempreg1 ; transfers value in table referenced by table pointer,
               ; data at program memory address 0B06H 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 "0B05H" transferred to
               ; tempreg2 and TBLH;
               ; in this example the data "1AH" is transferred to
               ; tempreg1 and data "0FH" to register tempreg2

:
:
org 0B00h    ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

在线烧录

Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。另外，Holtek 单片机提供 4 线接口的在线烧录方式。用户可将进行过烧录或未经过烧录的单片机芯片连同电路板一起制成，最后阶段进行程序的更新和程序的烧录，在无需去除或重新插入芯片的情况下方便地保持程序为最新版。

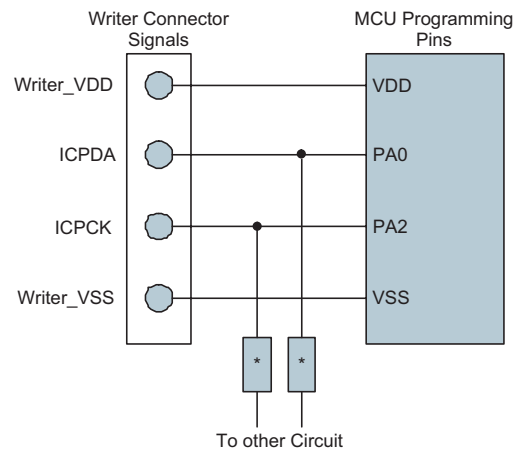
Holtek Flash MCU 与烧录器引脚对应表如下：

Holtek 烧录器引脚名称	MCU 在线烧录引脚名称	功能
ICPDA	PA0	串行数据 / 地址 - 读 / 写
ICPCK	PA2	串行时钟输入
VDD	VDD	电源 (5V)
GND	VSS	地

芯片内部程序存储器可以通过 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，一条用于串行时钟输入输出，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，PA0 和 PA2 引脚用于数据和时钟烧录，用户必须确保这两个引脚没有连接到其它输出脚。



注：“*” 表示电阻或电容，且电阻值必须大于 1kΩ，电容值必须小于 1nF。

片上调试

BS84BV06A-3 EV 芯片用于对单片机 BS84B06A-3 仿真。此 EV 芯片提供片上调试功能（OCDS – On-chip Debug Support）用于开发过程中对单片机的调试。除了片上调试功能方面，EV IC 和实际 MCU 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。当用户用 EV 芯片进行调试时，单片机中 OCSDA 和 OCDSCK 引脚上的其它共用功能无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。关于 OCDS 功能的详细描述，请参考“Holtek e-Link for 8-bit MCU OCDS User’s Guide”手册。

Holtek e-Link 引脚名称	MCU 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
GND	VSS	地

数据存储器

数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。

结构

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

第二部分数据存储器是做一般用途使用，都可在程序控制下进行读取和写入。

该单片机总的数据存储器被分为两个 Bank。大部分特殊功能数据寄存器均可在所有 Bank 被访问，处于“40H”地址的 EEC 寄存器却只能在 Bank 1 中被访问到。切换不同区域可通过设置区域指针（BP）实现。该单片机的数据存储器的起始地址都是“00H”。

容量	Bank 0	Bank 1
288×8	60H~FFH	80H~FFH

通用数据存储器结构

Bank 0, 1		Bank 0	Bank 1
00H	IAR0	30H	ADCR0
01H	MP0	31H	ADCR1
02H	IAR1	32H	ACERL
03H	MP1	33H	Unused
04H	BP	34H	Unused
05H	ACC	35H	PD
06H	PCL	36H	PDC
07H	TBLP	37H	PDPJ
08H	TBLH	38H	Unused
09H	TBHP	39H	Unused
0AH	STATUS	3AH	Unused
0BH	SMOD	3BH	Unused
0CH	CTRL	3CH	Unused
0DH	INTEG	3DH	Unused
0EH	INTC0	3EH	Unused
0FH	INTC1	3FH	Unused
10H	Unused	40H	Unused EEC
11H	Unused	41H	Unused
12H	Unused	42H	Unused
13H	Unused	43H	TKTMR
14H	PA	44H	TKC0
15H	PAC	45H	TK16DL
16H	PAPU	46H	TK16DH
17H	PAWU	47H	TKC1
18H	SLEDC0	48H	TKM016DL
19H	SLEDC1	49H	TKM016DH
1AH	WDTC	4AH	TKM0ROL
1BH	TBC	4BH	TKM0ROH
1CH	TMR	4CH	TKM0C0
1DH	TMRC	4DH	TKM0C1
1EH	EEA	4EH	TKM116DL
1FH	EED	4FH	TKM116DH
20H	PB	50H	TKM1ROL
21H	PBC	51H	TKM1ROH
22H	PBPU	52H	TKM1C0
23H	I2CTOC	53H	TKM1C1
24H	SIMC0	54H	Unused
25H	SIMC1	55H	Unused
26H	SIMD	56H	Unused
27H	SIMC2/SIMA	57H	Unused
28H	Unused	58H	Unused
29H	Unused	59H	Unused
2AH	Unused	5AH	Unused
2BH	Unused	5BH	Unused
2CH	Unused	5CH	Unused
2DH	Unused	5DH	Unused
2EH	ADRL	5EH	Unused
2FH	ADRH	5FH	Unused

特殊功能数据存储器

特殊功能寄存器

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

间接寻址寄存器 – IAR0, IAR1

间接寻址寄存器 IAR0 和 IAR1 的地址虽位于数据存储区，但其并没有实际的物理地址。间接寻址的方法准许使用间接寻址指针做数据操作，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，将对间接寻址指针（MP0 和 MP1）所指定的存储器地址产生对应的读 / 写操作。它们总是成对出现，IAR0 和 MP0 可以访问 Bank 0，而 IAR1 和 MP1 可以访问 Bank 0 和 Bank 1。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。

间接寻址指针 – MP0, MP1

该单片机提供两个间接寻址指针，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间接寻址指针所指定的地址。MP0 和 IAR0 用于访问 Bank 0，而 MP1 和 IAR1 可通过 BP 寄存器访问所有的 Bank。直接寻址仅用于 Bank 0，Bank 1 仅可使用 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。

间接寻址程序举例

```
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

数据存储区被分为两个 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区。第 0 位用于选择数据存储区 0~1。

复位后，数据存储区会初始化到 Bank 0，但是在空闲 / 休眠模式下的 WDT 溢出复位，不会改变通用数据存储器的存储区号。应该注意的是，除 Bank 1 中的 EEC 寄存器，其它特殊功能数据存储区不受存储区的影响，也就是说，不论是

在哪个存储区，都能对特殊功能寄存器进行读写操作。而 EEC 寄存器只能通过间接寻址数据存储器进行访问。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 1，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	—	—	—	—	—	—	—	DMBP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 未使用，读为“0”

Bit 0 **DMBP0**: 数据存储区选择位

0: Bank 0

1: Bank 1

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” 为未知

- Bit 7~6 未使用, 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 也受循环移位指令的影响。

EEPROM 数据寄存器

此单片机的一个特性是内建 EEPROM 数据存储器。“Electrically Erasable Programmable Read Only Memory”为电可擦可编程只读存储器，由于其非易失的存储结构，即使在电源掉电的情况下存储器内的数据仍然保存完好。这种存储区扩展了 ROM 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 数据寄存器结构

EEPROM 数据寄存器容量为 64×8 。由于映射方式与程序存储器和数据存储器不同，因此不能像其它类型的存储器一样寻址。使用 Bank 0 中的一个地址寄存器和一个数据寄存器以及 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 数据存储器总的操作。地址寄存器 EEA、数据寄存器 EED 及控制寄存器 EEC。EEA 和 EED 位于 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 被设为“01H”。

EEPROM 寄存器列表

Name	Bit							
	7	6	5	4	3	2	1	0
EEA	—	—	D5	D4	D3	D2	D1	D0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEA 寄存器

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

Bit 7~6 未定义，读为“0”

Bit 5~0 数据 EEPROM 地址

数据 EEPROM 地址 Bit 5~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 EEPROM 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 中将要被读取数据的地址放入 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，写入的数据要存入 EED 寄存器中。写数据至 EEPROM，EEC 寄存器中的写使能位 WREN 先置为高以使能写功能。之后将 WR 位置为高，初始化一个写周期。这两个指令必须连续执行。在执行任何写操作之前，总中断位 EMI 要先清零，写周期开始后，再将 EMI 置为高。需要注意的是若 WR 位已置为高而 WREN 位还未被设置则不能开始写操作。由于控制 EEPROM 写周期的是一个内部时钟，其与单片机的系统时钟异步，所以数据写入 EEPROM 的时间将有所延迟。可通过轮询 EEC 寄存器中的 WR 位或判断 EEPROM 中断以侦测写周期是否完成。若写周期完成，WR 位将自动清除为“0”，通知用户数据已写入 EEPROM。因此，应用程序将轮询 WR 位以确定写周期是否结束。

写保护

防止误写入的写保护有以下几种。单片机上电后控制寄存器中的写使能位将被清除以杜绝任何写入操作。上电后 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 中断

EEPROM 写周期结束后将产生 EEPROM 写中断，当 EEPROM 写周期结束，DEF 请求标志位将被置位。若总中断和 EEPROM 中断使能且堆栈未满的情况下将跳转到相应的 EEPROM 中断向量中执行。当中断被响应，EEPROM 中断标志位 DEF 将自动复位。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。BP 指针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。写数据时，WREN 位置为“1”后，WR 须立即设置为高，以确保正确地执行写周期。写周期执行前总中断位 EMI 应先清零，写周期开始执行后再将此位重新使能。注意，EEPROM 读或写周期彻底完成之前单片机不能进入空闲或休眠模式，否则将导致 EEPROM 读或写操作失败。

程序举例

● 从 EEPROM 中读取数据 – 轮询法

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, 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
```

```
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。振荡器选择是通过寄存器完成的。

振荡器概述

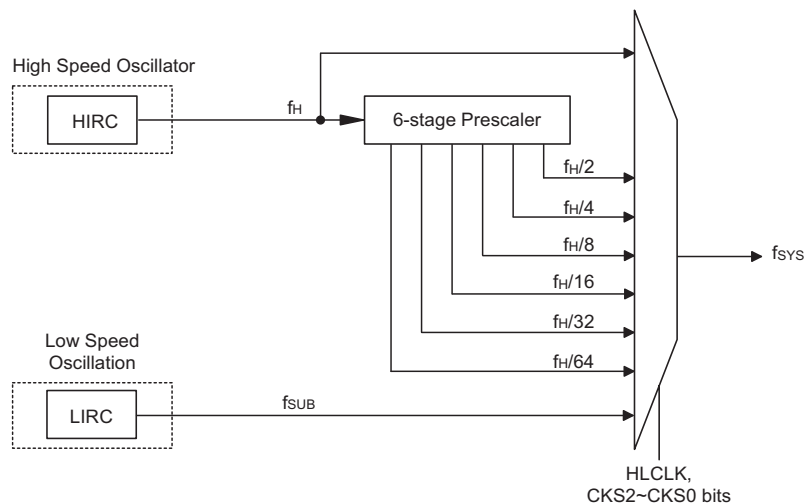
该单片机有两个内部振荡器，一个低速振荡器和一个高速振荡器。它们除了可以作为系统时钟源，还可以作为看门狗定时器和时基功能的时钟源。集成的两个内部振荡器不需要任何外接元件。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

振荡类型	名称	频率范围
内部高速 RC	HIRC	8/12/16MHz
内部低速 RC	LIRC	32kHz

振荡器类型

系统时钟配置

该单片机有两个方式产生系统时钟，一个高速内部时钟源和一个低速内部时钟源。高速振荡器为内部 8MHz、12MHz 或 16MHz RC 振荡器，低速振荡器为内部 32 kHz RC 振荡器。这两个振荡器都是内部全集成的振荡器，无需外接元件。选择高速或低速振荡器作为系统振荡器，是通过 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位进行选择。



系统时钟配置

内部高速 RC 振荡器 – HIRC

内部高速 RC 振荡器是一个完全集成的系统振荡器，不需其它外部元件。内部 RC 振荡器上电默认频率为 8MHz，但可通过 CTRL 寄存器的 HIRCS1 和 HIRCS0 位选择为 8 MHz、12 MHz 或 16 MHz。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因 V_{DD} 、温度以及芯片制成工艺不同的影响较大程度地降低。

内部低速 32kHz 振荡器 – LIRC

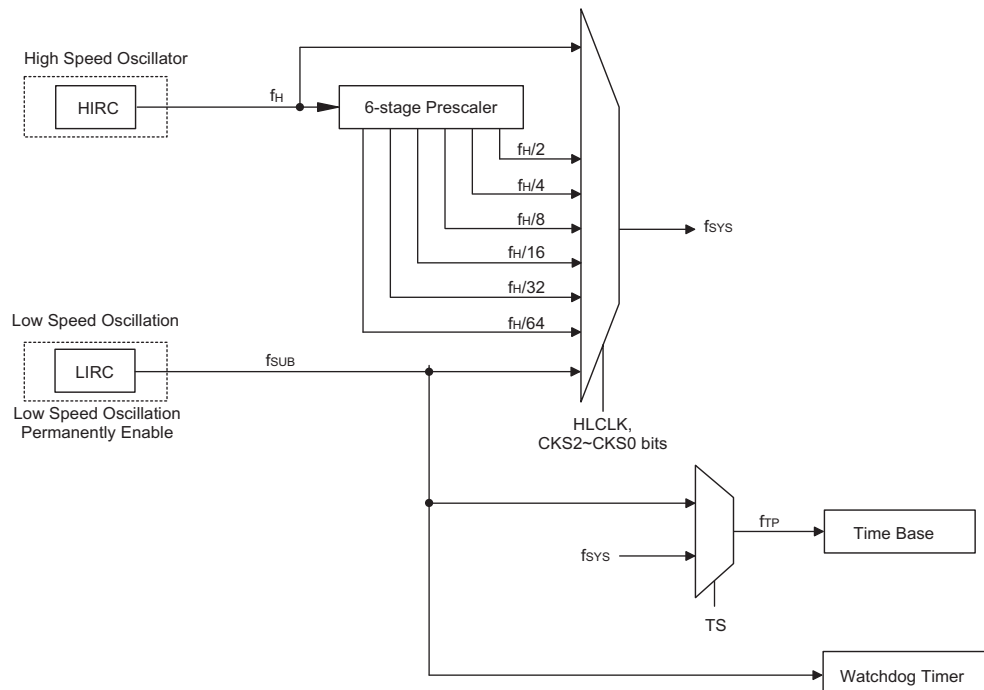
内部 32kHz 系统振荡器是低频振荡器。它是一个完全集成的 RC 振荡器，在 5V 电压下运行的典型频率值为 32kHz 且无需外部元件。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响较大程度地降低。上电后，LIRC 振荡器将被永久使能，此振荡器不能通过寄存器除能。

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

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 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位进行选择。高频时钟和低频时钟都来自内部 RC 振荡器。



系统时钟选项

注：当系统时钟源 f_{SYS} 由 f_H 到 f_{SUB} 转换时，高速振荡器将停止以节省耗电。因此，没有为外围电路提供 $f_H \sim f_H/64$ 。

控制寄存器

寄存器 SMOD 用于控制单片机内部时钟。

SMOD 寄存器

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	—	R	R	R/W	R/W
POR	0	0	0	—	0	0	1	1

Bit 7~5 **CKS2~CKS0**: 当 HLCLK 为“0”时系统时钟选择位

000: f_{SUB} (f_{LIRC})

001: f_{SUB} (f_{LIRC})

010: $f_H/64$

011: $f_H/32$

100: $f_H/16$

101: $f_H/8$

110: $f_H/4$

111: $f_H/2$

这三位用于选择系统时钟源。除了 LIRC 振荡器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 未定义，读为“0”

Bit 3 **LTO**: LIRC 振荡器就绪标志位

0: 未就绪

1: 就绪

此位为低速系统振荡器就绪标志位，用于表明低速系统振荡器在系统上电复位何时稳定下来。系统唤醒后该位转换为高需要 1~2 个时钟周期。

Bit 2 **HTO**: HIRC 振荡器就绪标志位

0: 未就绪

1: 就绪

此位为高速系统振荡器就绪标志位，用于表明高速系统振荡器何时稳定下来。此标志在系统上电后经硬件清零，高速系统振荡器稳定后变为高电平。因此，此位在单片机上电后由应用程序读取的总为“1”。该标志位在休眠模式或空闲模式 0 处于低电平状态，被唤醒后转为高电平状态需 15~16 个时钟周期。

Bit 1 **IDLEN**: 空闲模式控制位

0: 除能

1: 使能

此位为空闲模式控制位，用于决定 HALT 指令执行后发生的动作。若此位为高，当指令 HALT 执行后，单片机进入空闲模式。若 FSYSON 位为高，在空闲模式 1 中 CPU 停止运行，系统时钟将继续工作以保持外围功能继续工作；若 FSYSON 为低，在空闲模式 0 中 CPU 和系统时钟都将停止运行。若此位为低，单片机将在 HALT 指令执行后进入休眠模式。

Bit 0 **HLCLK**: 系统时钟选择位

0: $f_H/2 \sim f_H/64$ 或 f_{SUB}

1: f_H

此位用于选择 f_H 或 $f_H/2 \sim f_H/64$ 还是 f_{SUB} 作为系统时钟。该位为高时选择 f_H 作为系统时钟，为低时则选择 $f_H/2 \sim f_H/64$ 或 f_{SUB} 作为系统时钟。当系统时钟由 f_H 时钟向 f_{SUB} 时钟转换时， f_H 将自动关闭以降低功耗。

CTRL 寄存器

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	HIRCS1	HIRC0	—	LVRF	D1	WRF
R/W	R/W	—	R/W	R/W	—	R/W	R/W	R/W
POR	0	—	0	0	—	x	0	0

“x” 为未知

- Bit 7

FSYSON: 空闲模式下的 f_{SYS} 控制

0: 除能

1: 使能
- Bit 6

未定义, 读为 “0”
- Bit 5~4

HIRCS1~HIRCS0: 高速振荡器频率选择位

00: 8MHz

01: 16MHz

10: 12MHz

11: 8MHz
- Bit 3

未使用, 读为 “0”
- Bit 2

LVRF: 低电压复位标志位

0: 未发生

1: 发生

当发生低电压复位时该位为 “1”, 该位只可通过应用程序清零。
- Bit 1

保留位, 不能被改写
- Bit 0

WRF: WDT 控制寄存器软件复位标志位

0: 未发生

1: 发生

当发生 WDT 控制寄存器软件复位时该位为 “1”, 该位只可通过应用程序清零。

系统工作模式

该单片机有 5 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: 正常模式和低速模式。剩余的 3 种工作模式: 休眠模式、空闲模式 0 和空闲模式 1 用于单片机 CPU 关闭时以节省耗电。

工作模式	说明		
	CPU	f_{SYS}	f_{SUB}
正常模式	On	$f_H \sim f_H/64$	On
低速模式	On	f_{SUB}	On
空闲模式 0	Off	Off	On
空闲模式 1	Off	On	On
休眠模式	Off	Off	On

正常模式

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 SMOD 寄存器中的 CKS2~CKS0 位及 HLCLK 位选择的。单片机使用高速振荡器分频作为系统时钟可减少工作电流。

低速模式

此模式的系统时钟虽为较低速时钟源，但单片机仍能正常工作。该时钟源为 f_{SUB} 。单片机在此模式中运行所耗工作电流较低。在低速模式下， f_H 关闭。

休眠模式

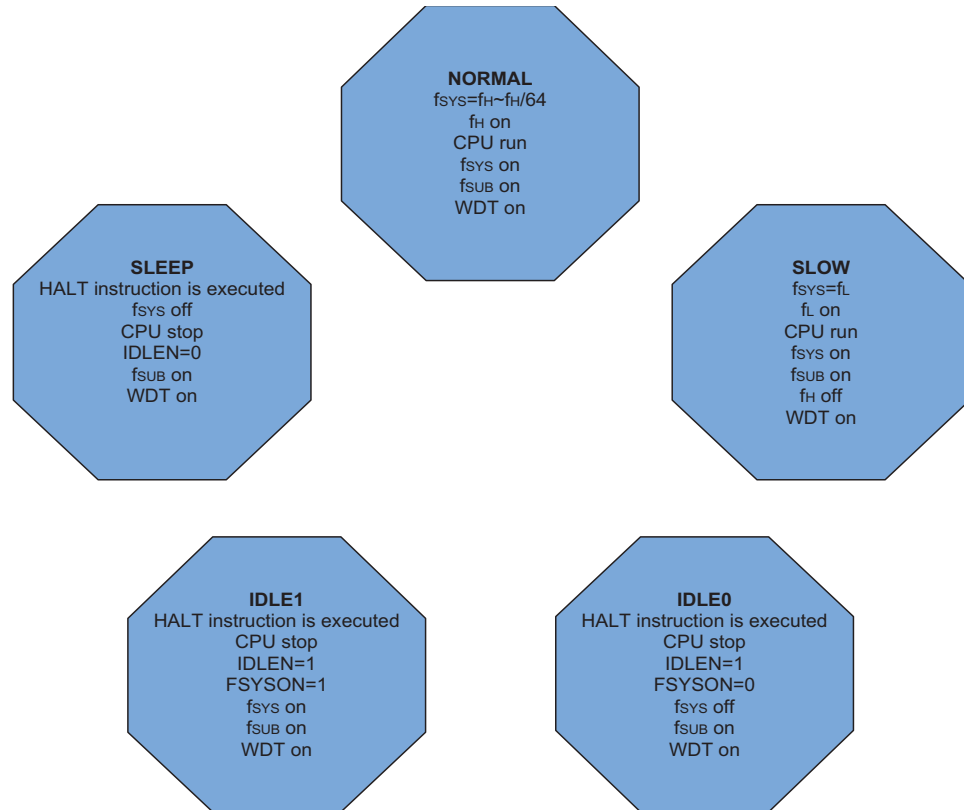
在 HALT 指令执行后且 SMOD 寄存器中 IDLEN 位为低时，系统进入休眠模式。在休眠模式中，CPU 停止运行。然而 f_{SUB} 时钟继续运行，看门狗定时器继续工作。

空闲模式 0

执行 HALT 指令后且 SMOD 寄存器中 IDLEN 位为高，CTRL 寄存器中 FSYSON 位为低时，系统进入空闲模式 0。在空闲模式 0 中，系统振荡器停止，CPU 停止工作。但一些外围功能如看门狗定时器和定时 / 计数器将继续工作。

空闲模式 1

执行 HALT 指令后且 SMOD 寄存器中 IDLEN 位为高，CTRL 寄存器中 FSYSON 位为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，但会提供一个时钟源给一些外围功能。在空闲模式 1 中，系统振荡器继续运行，该系统振荡器可以为高速或低速系统振荡器。在该模式中，看门狗时钟开启。



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。

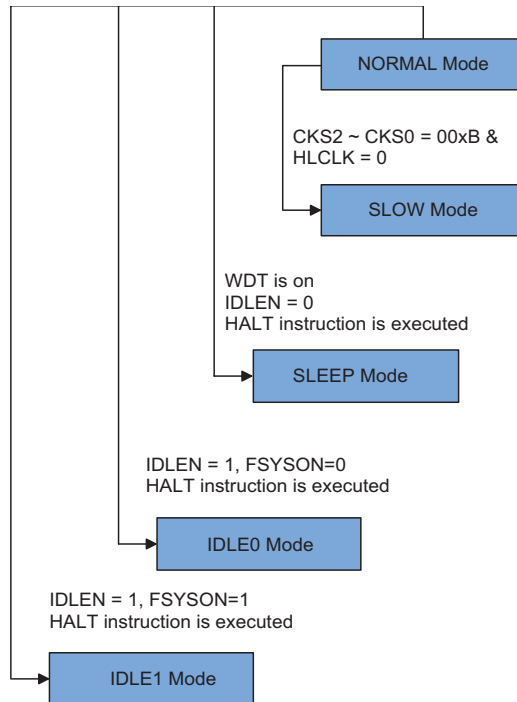
简单来说，正常模式和低速模式间的切换仅需设置 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位即可实现，而正常模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SMOD 寄存器中的 IDLEN 位和 CTRL 寄存器中的 FSYSON 位决定的。

当 HLCLK 位变为低电平时，时钟源将由高速时钟源 f_H 转换成时钟源 $f_H/2 \sim f_H/64$ 或 f_{SUB} 。若时钟源来自 f_{SUB} ，高速时钟源将停止运行以节省耗电。此时须注意， $f_H/16$ 和 $f_H/64$ 时钟源也将停止运行。所附流程图显示了单片机在不同工作模式间切换时的变化。

正常模式切换到低速模式

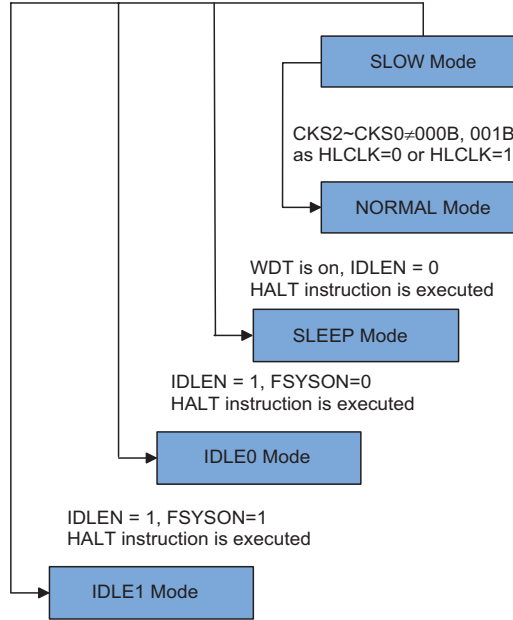
系统运行在正常模式时使用高速系统振荡器，因此较为耗电。可通过设置 SMOD 寄存器中的 HLCLK 位为“0”及 CKS2~CKS0 位为“000”或“001”使系统时钟切换至运行在低速模式下。此时将使用低速系统振荡器以节省耗电。用户可在对性能要求不高的操作中使用此方法以减少耗电。

低速模式的时钟源来自 LIRC 振荡器，因此要求此振荡器在所有模式切换动作发生前稳定下来。该动作由 SMOD 寄存器中 LTO 位检测。



低速模式切换到正常模式

在低速模式系统使用 LIRC 低速振荡器。切换到使用高速系统时钟振荡器的正常模式需设置 HLCLK 位为“1”，也可设置 HLCLK 位为“0”但 CKS2~CKS0 需设为“010”、“011”、“100”、“101”、“110”或“111”。高频时钟需要一定的稳定时间，通过检测 HTO 位的状态可进行判断。



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”指令前需设置寄存器 SMOD 中 IDLEN 位为“0”。在上述条件下执行该指令后，将发生的情况如下：

- 系统时钟和时基时钟停止运行，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。
- WDT 将被清零并重新开始计数。
- 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。

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”指令前需设置寄存器 SMOD 中 IDLEN 位为“1”且 CTRL 寄存器中的 FSYSON 位为“0”。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，时基时钟和 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。
- WDT 将被清零并重新开始计数
- 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。

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”指令前需设置寄存器 SMOD 中 IDLEN 位为“1”且 CTRL 寄存器中的 FSYSON 位为“1”。在上述条件下执行该指令后，将发生的情况如下：

- 系统时钟和 f_{SUB} 开启，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- WDT 将被清零并重新开始计数。
- 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。

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 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。在空闲模式 1 中，系统时钟开启。若系统时钟来自高速系统振荡器，额外的静态电流也可能会有几百微安。

唤醒

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

若由 WDT 溢出唤醒，则会发生看门狗定时器复位。系统上电或执行清除看门狗的指令，会清零 PDF；执行 HALT 指令，PDF 将被置位。看门狗计数器溢出将会置位 TO 标志并唤醒系统，这种复位只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：相关中断使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。

系统振荡器	唤醒时间 (休眠模式)	唤醒时间 (空闲模式 0)	唤醒时间 (空闲模式 1)
HIRC	15~16 个 HIRC 周期		1~2 个 HIRC 周期
LIRC	1~2 个 LIRC 周期		1~2 个 LIRC 周期

唤醒时间

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源来自于内部时钟 f_{SUB} ，而该时钟由 LIRC 振荡器提供。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。电压为 5V 时内部振荡器 LIRC 的频率大约为 32kHz。需要注意的是，这个特殊的内部时钟周期随 V_{DD} 、温度和制成的不同而变化。WDT 一直使能。

看门狗定时器控制寄存器

WDTC 寄存器用于控制 WDT 的使能和溢出周期选择。除了 WDT 在省电模式下的溢出复位外的其它复位发生后 WDTC 寄存器的值为 01010011B。

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 或 10101: 使能
 其它值: MCU 复位 (需要 1~2 个 LIRC 周期响应复位)
 若发生 WDT 软件复位, CTRL 寄存器中的 WRF 位将置高。

Bit 2~0 **WS2~WS0**: 选择看门狗溢出周期
 000: $2^8/f_{SUB}$
 001: $2^{10}/f_{SUB}$
 010: $2^{12}/f_{SUB}$
 011: $2^{14}/f_{SUB}$ (默认)
 100: $2^{15}/f_{SUB}$
 101: $2^{16}/f_{SUB}$
 110: $2^{17}/f_{SUB}$
 111: $2^{18}/f_{SUB}$
 这三位控制看门狗时钟的分频比, 进而控制看门狗的溢出周期

CTRL 寄存器

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	HIRCS1	HIRC0	—	LVRF	D1	WRF
R/W	R/W	—	R/W	R/W	—	R/W	R/W	R/W
POR	0	—	0	0	—	x	0	0

“x” 为未知

Bit 7 **FSYSON**: 空闲模式下的 f_{SYS} 控制
 详见别处描述。
 Bit 6 未定义, 读为 “0”
 Bit 5~4 **HIRCS1~HIRCS0**: 高速振荡器频率选择位
 详见别处描述。
 Bit 3 未使用, 读为 “0”

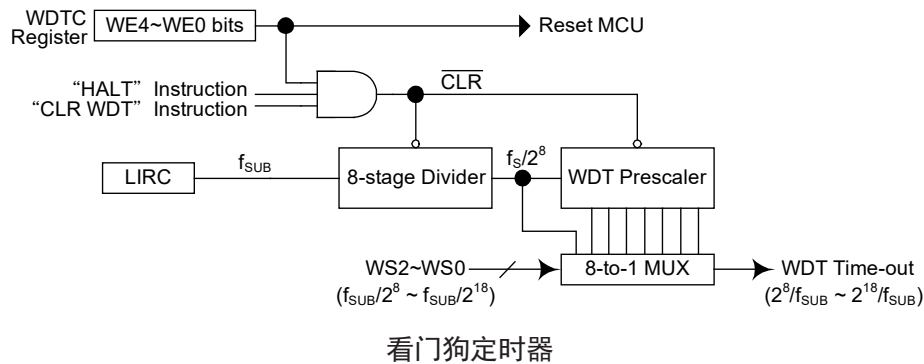
Bit 2	LVRF : 低电压复位标志位 详见别处描述。
Bit1	保留位, 不能被改写
Bit 0	WRF : WDT 控制寄存器软件复位标志位 0: 未发生 1: 发生 当发生 WDT 控制寄存器软件复位时该位为“1”, 该位只可通过应用程序清零。

看门狗定时器操作

该单片机的看门狗定时器时钟源来自 f_{SUB} 振荡器, 因此总是开启的。当 WDT 溢出时, 它产生一个芯片复位的动作。这也就意味着正常工作期间, 用户需在应用程序中看门狗定时器溢出前将看门狗定时器清零以防止其产生复位, 可使用清看门狗指令实现。在程序运行过程由于某些无法预知的原因会使程序跳转到一个未知的地址或进入一个死循环, 此时清除看门狗指令无法被正确执行, 在这种情况下, 看门狗定时器会计数溢出以使单片机复位。WDTC 寄存器中的 WE4~WE0 位可提供使能以及看门狗定时器复位操作。如果 WE4~WE0 位为“10101B”和“01010B”, WDT 功能使能。如果 WE4~WE0 设置为“10101B”和“01010B”以外的其它任意值, 则经过 2~3 个 LIRC 时钟周期后单片机复位。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 标志位会被置位, 且只有程序计数器 PC 和堆栈指针 SP 会被复位。有三种方法可以用来清除 WDT 的内容。第一种是 WDT 软件复位, 即将 WE4~WE0 位设置成除了“10101B”和“01010B”以外的其它任意值; 第二种是通过软件清除指令, 而第三种是通过“HALT”指令。

该单片机只使用一条看门狗清除指令“CLR WDT”。因此只要执行“CLR WDT”便能清除 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 7.8ms。



复位和初始化

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

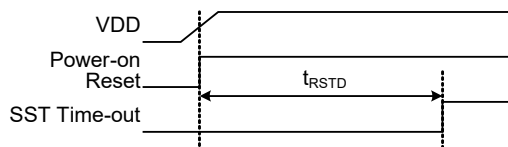
另一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。

复位功能

单片机的几种复位方式将在此处做具体介绍。

上电复位

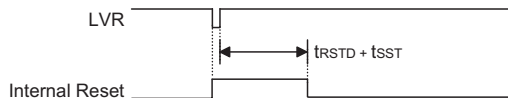
这是最基本且不可避免的复位，发生在单片机上电后。除了保证程序存储器从开始地址执行，上电复位也使得其它寄存器被设定在预设条件。所有的输入/输出端口控制寄存器在上电复位时会保持高电平，以确保上电后所有引脚被设定为输入状态。



上电复位时序图

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。电源复位低电压固定为 2.55V。例如在更换电池的情况下，单片机供应的电压可能会在 0.9V~V_{LVR} 之间，这时 LVR 将会自动复位单片机且 CTRL 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~V_{LVR} 的低电压状态的时间，必须超过交流电气特性中 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。注意当单片机进入空闲或休眠模式，LVR 功能将自动关闭。



低电压复位时序图

• CTRL 寄存器

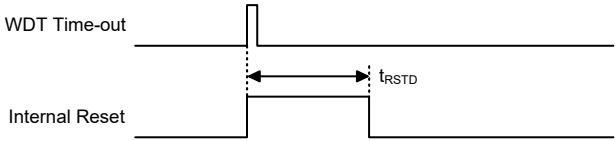
Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	HIRCS1	HIRC0	—	LVRF	D1	WRF
R/W	R/W	—	R/W	R/W	—	R/W	R/W	R/W
POR	0	—	0	0	—	x	0	0

“x” 为未知

- Bit 7 **FSYSON**: 空闲模式下的 f_{sys} 控制
详见别处描述。
- Bit 6 未定义, 读为 “0”
- Bit 5~4 **HIRCS1~HIRCS0**: 高速振荡器频率选择位
详见别处描述。
- Bit 3 未使用, 读为 “0”
- Bit 2 **LVRF**: 低电压复位标志位
0: 未发生
1: 发生
当发生低电压复位时该位为 “1”, 该位只可通过应用程序清零。
- Bit 1 保留位, 不能被改写
- Bit 0 **WRF**: WDT 控制寄存器软件复位标志位
详见别处描述。

正常运行时看门狗溢出复位

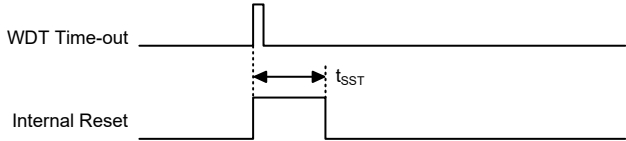
除了看门狗溢出标志位 TO 将被设为 “1” 之外, 正常运行时看门狗溢出复位和 LVR 复位相同。



正常运行时看门狗溢出时序图

休眠或空闲时看门狗溢出复位

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



注: HIRC 为系统时钟源, t_{SST} 为 15 ~ 16 个时钟周期
LIRC 为系统时钟源, t_{SST} 为 1 ~ 2 个时钟周期

休眠或空闲时看门狗溢出复位时序图

复位初始状态

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

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

注：“u”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	WDT 清除并重新计数
定时模块	定时 / 计数器停止
输入 / 输出口	I/O 口设为输入模式，AN0~AN3 作为 A/D 输入脚
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。此芯片有多种封装类型，表格反应较大的封装的情况。

寄存器	上电复位	WDT 溢出 (正常模式)	WDT 溢出 (HALT)
IAR0	----	----	----
MP0	xxxx xxxx	xxxx xxxx	uuuu uuuu
IAR1	----	----	----
MP1	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	---- --0	---- --0	---- --u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	---- xxxx	---- uuuu	---- uuuu
STATUS	--00 xxxx	--1u uuuu	--11 uuuu
SMOD	000- 0011	000- 0011	uuu- uuuu
CTRL	0-00 -x00	0-00 -x00	u-uu -uuu
INTEG	---- --00	---- --00	---- --uu
INTC0	-000 0000	-000 0000	-uuu uuuu

寄存器	上电复位	WDT 溢出 (正常模式)	WDT 溢出 (HALT)
INTC1	0000 0000	0000 0000	uuuu uuuu
PA	1--1 1111	1--1 1111	u--u uuuu
PAC	1--1 1111	1--1 1111	u--u uuuu
PAPU	0--0 0000	0--0 0000	u--u uuuu
PAWU	0--0 0000	0--0 0000	u--u uuuu
SLEDC0	0101 0101	0101 0101	uuuu uuuu
SLEDC1	---- --01	---- --01	---- --uu
WDTC	0101 0011	0101 0011	uuuu uuuu
TBC	--00 ----	--00 ----	--uu ----
TMR	0000 0000	0000 0000	uuuu uuuu
TMRC	--00 -000	--00 -000	--uu -uuu
EEA	--00 0000	--00 0000	--uu uuuu
EED	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	uuuu uuuu
I2CTOC	0000 0000	0000 0000	uuuu uuuu
SIMC0	111- --0-	111- --0-	uuu- --u-
SIMC1	1000 0001	1000 0001	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMC2 (SPI mode)	--00 0000	--00 0000	--uu uuuu
SIMA (I ² C mode)	0000 000-	0000 000-	uuuu uuu-
ADRL (ADRF5=0)	xxxx ----	xxxx ----	uuuu ----
ADRL (ADRF5=1)	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH (ADRF5=0)	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH (ADRF5=1)	---- xxxx	---- xxxx	---- uuuu
ADCR0	0110 --00	0110 --00	uuuu --uu
ADCR1	00-0 -000	00-0 -000	uu-u -uuu
ACERL	---- 1111	---- 1111	---- uuuu
PD	---- 1111	---- 1111	---- uuuu
PDC	---- 1111	---- 1111	---- uuuu
PDPU	---- 0000	---- 0000	---- uuuu
TKTMR	0000 0000	0000 0000	uuuu uuuu
TKC0	-000 0000	-000 0000	-uuu uuuu

寄存器	上电复位	WDT 溢出 (正常模式)	WDT 溢出 (HALT)
TK16DL	0000 0000	0000 0000	uuuu uuuu
TK16DH	0000 0000	0000 0000	uuuu uuuu
TKC1	---- --11	---- --11	---- --uu
TKM016DL	0000 0000	0000 0000	uuuu uuuu
TKM016DH	0000 0000	0000 0000	uuuu uuuu
TKM0ROL	0000 0000	0000 0000	uuuu uuuu
TKM0ROH	---- --00	---- --00	---- --uu
TKM0C0	0000 0000	0000 0000	uuuu uuuu
TKM0C1	0-00 0000	0-00 0000	u-uu uuuu
TKM116DL	0000 0000	0000 0000	uuuu uuuu
TKM116DH	0000 0000	0000 0000	uuuu uuuu
TKM1ROL	0000 0000	0000 0000	uuuu uuuu
TKM1ROH	---- --00	---- --00	---- --uu
TKM1C0	-000 0000	-000 0000	-uuu uuuu
TKM1C1	0-00 --00	0-00 --00	u-uu --uu
EEC	---- 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。大部分引脚都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、PB 和 PD 双向输入 / 输出口。这些寄存器在数据存储器有特定的地址。所有 I/O 口用于输入输出操作。作为输入操作，输入引脚无锁存功能，也就是说输入数据必须在执行“MOV A, [m]”，T2 的上升沿准备好，m 为端口地址。对于输出操作，所有数据都是被锁存的，且保持不变直到输出锁存被重写。

输入 / 输出寄存器列表

寄存器名称	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	—	—	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	—	—	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	—	—	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	—	—	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PD	—	—	—	—	PD3	PD2	PD1	PD0
PDC	—	—	—	—	PDC3	PDC2	PDC1	PDC0
PDPU	—	—	—	—	PDPU3	PDPU2	PDPU1	PDPU0

上拉电阻

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

PAPU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAPU7	—	—	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7 PA 口 bit 7 上拉电阻控制

0: 除能

1: 使能

Bit 6~5 未使用，读为“0”

Bit 4~0 PA 口 bit 4~bit 0 上拉电阻控制

0: 除能

1: 使能

PBPU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
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 PB 口 bit 7~bit 0 上拉电阻控制
0: 除能
1: 使能

PDPU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PDPU3	PDPU2	PDPU1	PDPU0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 未使用，读为“0”
Bit 3~0 PD 口 bit 3~bit 0 上拉电阻控制
0: 除能
1: 使能

PA 口唤醒

当使用暂停指令“HALT”迫使单片机进入空闲或休眠模式状态，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 口上的每个引脚是可以通过设置 PAWU 寄存器来单独选择是否具有唤醒功能。

PAWU 寄存器

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

Bit 7 PA 口 bit 7 唤醒控制
0: 除能
1: 使能
Bit 6~5 未使用，读为“0”
Bit 4~0 PA 口 bit 4~bit 0 唤醒控制
0: 除能
1: 使能

输入 / 输出端口控制寄存器

每一个输入 / 输出端口都具有各自的控制寄存器，即 PAC、PBC 和 PDC，用来控制输入 / 输出状态。通过这些控制寄存器，每个 CMOS 输出或输入都可以通过软件动态控制。所有的 I/O 端口的引脚都各自对应于 I/O 端口控制寄存器的某一位。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，如果对输出端口做读取动作时，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	—	—	PAC4	PAC3	PAC2	PAC1	PAC0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	1	—	—	1	1	1	1	1

- Bit 7 PA 口 bit 7 输入 / 输出控制
0: 输出
1: 输入
- Bit 6~5 未使用，读为 “0”
- Bit 4~0 PA 口 bit 4~bit 0 输入 / 输出控制
0: 输出
1: 输入

PBC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
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 PB 口 bit 7~bit 0 输入 / 输出控制
0: 输出
1: 输入

PDC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PDC3	PDC2	PDC1	PDC0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	1

- Bit 7~4 未使用，读为 “0”
- Bit 3~0 PD 口 bit 3~bit 0 输入 / 输出控制
0: 输出
1: 输入

输入 / 输出引脚源电流选择

对于此款单片机，每个引脚的源电流可以通过寄存器相应位设置，但前提是引脚已被设置为 CMOS 输出模式，否则源电流选择位无效。用户可以参考直流电气特性部分 I_{OH} 在不同设置下的精确值。

SLEDC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PBPS3	PBPS2	PBPS1	PBPS0	PAPS3	PAPS2	PAPS1	PAPS0
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 7~6 **PBPS3~PBPS2**: PB7~PB4 引脚源电流选择位
00: 源电流 = Level 0 (最小)
01: 源电流 = Level 1

- 10: 源电流 = Level 2
11: 源电流 = Level 3 (最大)
PB7~PB4 引脚为 CMOS 输出时, 相应源电流选择位的设置才有效
- Bit 5~4 **PBPS1~PBPS0**: PB3~PB0 引脚源电流选择位
00: 源电流 = Level 0 (最小)
01: 源电流 = Level 1
10: 源电流 = Level 2
11: 源电流 = Level 3 (最大)
PB3~PB0 引脚为 CMOS 输出时, 相应源电流选择位的设置才有效
- Bit 3~2 **PAPS3~PAPS2**: PA7 和 PA4 引脚源电流选择位
00: 源电流 = Level 0 (最小)
01: 源电流 = Level 1
10: 源电流 = Level 2
11: 源电流 = Level 3 (最大)
PA7 和 PA4 引脚为 CMOS 输出时, 相应源电流选择位的设置才有效
- Bit 1~0 **PAPS1~PAPS0**: PA3~PA0 引脚源电流选择位
00: 源电流 = Level 0 (最小)
01: 源电流 = Level 1
10: 源电流 = Level 2
11: 源电流 = Level 3 (最大)
PA3~PA0 引脚为 CMOS 输出时, 相应源电流选择位的设置才有效

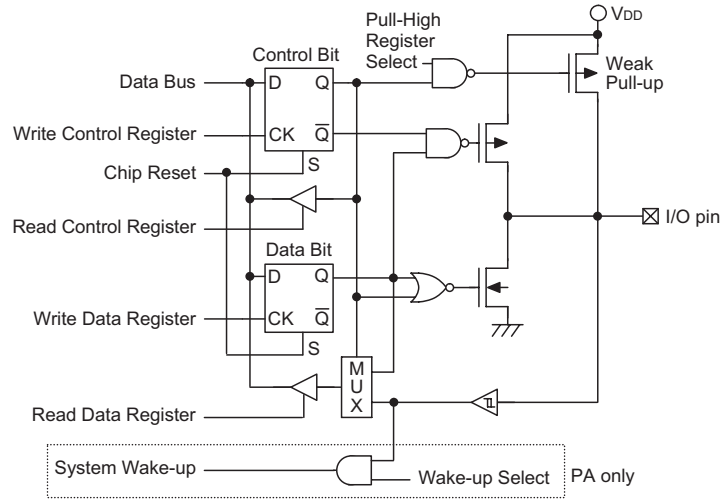
SLEDC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PDPS1	PDPS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	1

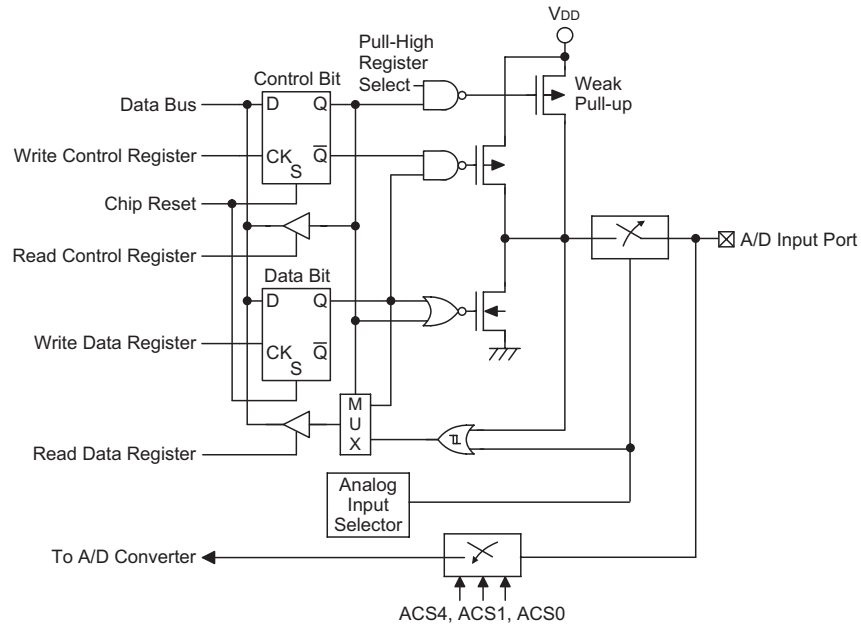
- Bit 7~2 未定义, 读为 “0”
- Bit 1~0 **PDPS1~PDPS0**: PD3~PD0 引脚源电流选择位
00: 源电流 = Level 0 (最小)
01: 源电流 = Level 1
10: 源电流 = Level 2
11: 源电流 = Level 3 (最大)
PD3~PD0 引脚为 CMOS 输出时, 相应源电流选择位的设置才有效

输入 / 输出引脚结构

下图为输入 / 输出引脚的内部结构图。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功能的理解提供的一个参考。图中引脚共用并非针对所有单片机。



通用输入 / 输出端口结构



A/D 输入 / 输出结构

编程注意事项

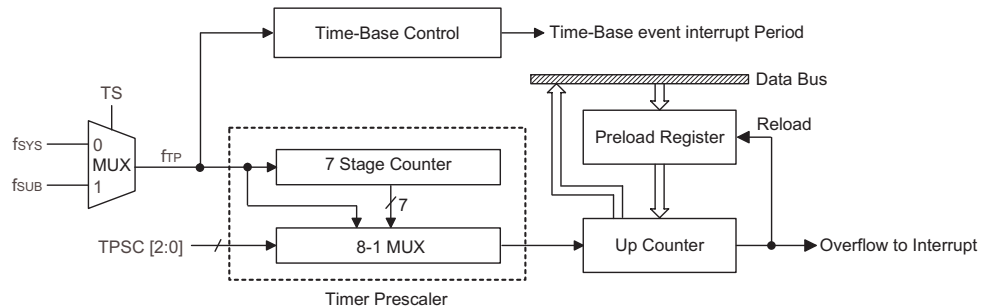
在编程中，最先要考虑的是端口的初始化。复位之后，所有的输入 / 输出数据及端口控制寄存器都将被设为逻辑高。所有输入 / 输出引脚默认为输入状态，而其电平则取决于其它相连接电路以及是否选择了上拉电阻。如果端口控制寄存器 PAC、PBC 和 PDC，某些引脚位被设定输出状态，这些输出引脚会有初始高电平输出，除非数据寄存器端口 PA、PB 和 PD 在程序中被预先设定。设置哪些引脚是输入及哪些引脚是输出，可通过设置正确的值到适当的端口控制寄存器，或使用指令“SET [m].i”及“CLR [m].i”来设定端口控制寄存器中个别的位。注意，当使用这些位控制指令时，系统即将产生一个读 - 修改 - 写的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

定时 / 计数器在任何单片机中都是一个很重要的部分，提供程序设计者一种实现和时间有关功能的方法。该单片机具有 1 个 8 位的计数器。这个计数器是可以作为一个普通定时器，另外提供了一个内部时钟分频器功能，以扩大定时器的范围。

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



定时 / 计数器

配置定时 / 计数器输入时钟源

定时 / 计数器的时钟源来自内部时钟。内部时钟源可以由 TMRC 寄存器 TS 位选择为来自系统时钟 f_{SYS} 或者 f_{SUB} 时钟。内部时钟首先由分频器分频，分频比由 TMRC 寄存器的 TPSC2~TPSC0 位来确定。

定时 / 计数器寄存器 – TMR

定时 / 计数寄存器 TMR，是位于特殊数据存储体内的特殊功能寄存器，用于储存定时器的当前值。在每收到一个内部计数脉冲，此寄存器的值将会加一。定时器将从预置寄存器所载入的值开始计数，到 FFH 时定时器溢出且会产生一个内部中断信号。定时器的值随后被预置寄存器的值重新载入并继续计数。

为了得到定时器的最大计算范围 FFH，预置寄存器需要先清为零。注意，上电

后预置寄存器处于未知状态。定时 / 计数器在关闭条件下，写数据到预置寄存器，会立即写入实际的定时器。而如果定时 / 计数器已经打开且正在计数，在这个周期内写入到预置寄存器的任何新数据将保留在预置寄存器，直到溢出发生后才被写入实际定时器。

定时 / 计数控制寄存器 – TMRC

定时 / 计数控制寄存器 TMRC，配合相应的定时寄存器控制定时 / 计数器的整个操作。在使用定时器之前，需要先正确地设定定时 / 计数控制寄存器，以便保证定时器能正确操作，而这个过程通常在程序初始化期间完成。

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

TMRC 寄存器

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

Bit 7~6 未定义，读为“0”

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

Bit 4 **TON**: 定时 / 计数器使能
0: 除能
1: 使能

Bit 3 未定义，读为“0”

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

定时器操作

在这个模式下，定时器可以用来测量固定时间间隔，当定时器发生溢出时，就会产生一个内部中断信号。 f_{SYS} 或 f_{SUB} 被用来当定时器的输入时钟源。然而，该定时器时钟源被预分频器进一步分频，分频比是由定时器控制寄存器的 TPSC2~TPSC0 位来确定。定时器控制寄存器第 4 位，即 TON 位需要设为逻辑高，使能定时器工作。每次内部时钟由高到低的电平转换都会使定时器值增加一；当定时器计数已满即溢出时，会产生中断信号且定时器会重新载入预置寄存器的值，然后继续计数。定时器溢出以及相应的内部中断产生也是唤醒暂停模式的一种方法。通过设置中断寄存器中的定时器中断使能位为“0”，可以禁止定时 / 计数器中断。

预分频器

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

编程注意事项

当读取定时 / 计数器值或写数据到预置寄存器时，计数时钟会被禁止以避免发生错误，但这样做可能会导致计数错误，所以程序设计者应该考虑到这点。在第一次使用定时 / 计数器之前，要仔细确认有没有正确地设定初始值。中断控制寄存器中的定时器使能位需要正确的设置，否则相应定时 / 计数器内部中断仍然无效。在定时 / 计数器打开之前，需要确保先载入定时 / 计数器寄存器的初始值。上电后，定时 / 计数器寄存器中的初始值是零。

定时 / 计数器初始化后，可以使用定时 / 计数器控制寄存器中的使能位来打开或关闭定时器。当定时 / 计数器产生溢出，中断控制寄存器中相应的中断请求标志将置位。若中断允许，将会依次产生一个中断信号。不管中断是否允许，在省电状态下，定时 / 计数器的溢出也会产生唤醒。若在空闲 / 休眠模式下，不需要定时器中断唤醒系统，可以在执行“HALT”指令之前将相应中断请求标志位置位。

A/D 转换器

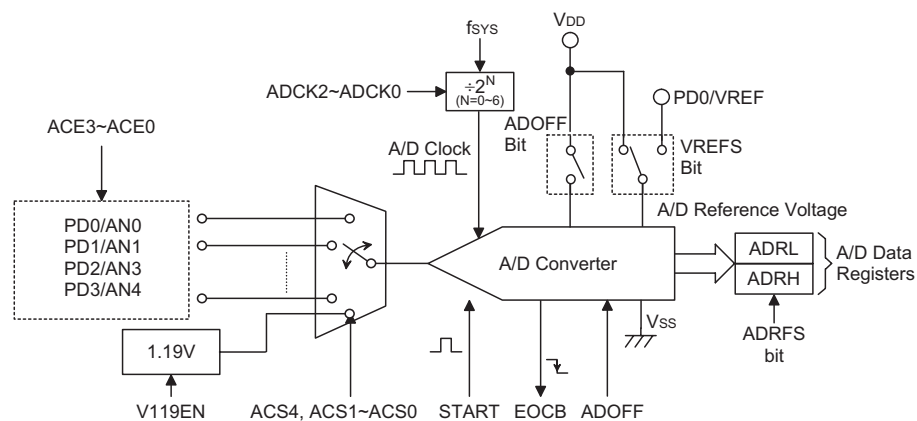
对于大多数电子系统而言，处理现实世界的模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先需要通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

A/D 简介

此单片机都包含一个 4 通道的 A/D 转换器，它们可以直接接入外部模拟信号（来自传感器或其它控制信号），并直接将这些信号转换成 12 位的数字量。

输入通道数	A/D 通道选择位	输入引脚
4	ACS4, ACS1~ACS0	AN0~AN3

下图显示了 A/D 转换器内部结构和相关的寄存器。



A/D 转换寄存器介绍

A/D 转换器的所有工作由五个寄存器控制。一对只读寄存器来存放 12 位 ADC 数据的值。剩下三个控制寄存器设置 A/D 转换器的操作和控制功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
ADRL (ADRFs=0)	D3	D2	D1	D0	—	—	—	—
ADRL (ADRFs=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADRH (ADRFs=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADRH (ADRFs=1)	—	—	—	—	D11	D10	D9	D8
ADCR0	START	EOCB	ADOFF	ADRFs	—	—	ACS1	ACS0
ADCR1	ACS4	V119EN	—	VREFS	—	ADCK2	ADCK1	ADCK0
ACERL	—	—	—	—	ACE3	ACE2	ACE1	ACE0

A/D 转换器寄存器列表

A/D 转换器数据寄存器 – ADRL, ADRH

对于具有 12 位 A/D 转换器的芯片，需要两个数据寄存器存放转换结果，一个高字节寄存器 ADRH 和一个低字节寄存器 ADRL。在 A/D 转换完毕后，单片机可以直接读取这些寄存器以获得转换结果。由于寄存器只使用了 16 位中的 12 位，其数据存储格式由 ADCR0 寄存器的 ADRFS 位控制，如下表所示。D0~D11 是 A/D 转换数据结果位。未使用的位读为“0”。

ADRFs	ADRH								ADRL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D 数据寄存器

A/D 转换控制寄存器 – ADCR0, ADCR1, ACERL

寄存器 ADCR0, ADCR1 和 ACERL 用来控制 A/D 转换器的功能和操作。这些 8 位的寄存器定义包括选择连接至内部 A/D 转换器的模拟通道，数字化数据格式，A/D 时钟源，并控制和监视 A/D 转换器的开始和转换结束状态。寄存器 ADCR0 的 ACS1~ACS0 位和 ADCR1 的 ACS4 位定义 ADC 输入通道编号。由于单片机只包含一个实际的模数转换电路，因此这 4 个模拟输入中的每一个都需要分别被发送到转换器。ACS4 和 ACS1~ACS0 位的功能决定选择哪个模拟输入通道或内部 1.19V 电路是否被连接到内部 A/D 转换器。

ACERL 控制寄存器中的 ACE3~ACE0 位，用来定义 PD 端口中的哪些引脚为 A/D 转换器的模拟输入，哪些引脚不作为 A/D 转换输入。相应位设为高将选择 A/D 输入功能，清零将选择 I/O 或其它引脚共用功能。当引脚作为 A/D 输入时，其原来的 I/O 或其它引脚共用功能消失，此外，其内部上拉电阻也将自动断开。

ADCR0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	START	EOCB	ADOFF	ADRF5	—	—	ACS1	ACS0
R/W	R/W	R	R/W	R/W	—	—	R/W	R/W
POR	0	1	1	0	—	—	0	0

- Bit 7 START:** 启动 A/D 转换位
 0 → 1 → 0: 启动
 0 → 1: 重置 A/D 转换, 并且设置 EOCB 为“1”
 此位用于初始化 A/D 转换过程。通常此位为低, 但如果设为高再被清零, 将初始化 A/D 转换过程。当此位为高, 将重置 A/D 转换器。
- Bit 6 EOCB:** A/D 转换结束标志
 0: A/D 转换结束
 1: A/D 转换中
 此位用于表明 A/D 转换过程的完成。当转换正在进行时, 此位为高。
- Bit 5 ADOFF:** ADC 模块电源开 / 关控制位
 0: ADC 模块电源开
 1: ADC 模块电源关
 此位控制 A/D 内部功能的电源。该位被清零将使能 A/D 转换器。如果该位设为高将关闭 A/D 转换器以降低功耗。由于 A/D 转换器即使在不执行转换动作时都会产生一定的功耗, 所以这在电源敏感的电池应用中需要多加注意。
 注: 1. 建议在进入空闲 / 休眠模式前, 设置 ADOFF=1 以减少功耗。
 2. ADOFF=1 将关闭 ADC 模块的电源。
- Bit 4 ADRF5:** ADC 数据格式控制位
 0: ADC 数据高八位是 ADRH 的 bit 7~bit 0, 低四位是 ADRL 的 bit 7~bit 4
 1: ADC 数据高四位是 ADRH 的 bit 3~bit 0, 低八位是 ADRL 的 bit 7~bit 0
 此位控制存放在两个 A/D 数据寄存器中的 12 位 A/D 转换结果的格式。细节方面请参考 A/D 数据寄存器章节。
- Bit 3~2** 未定义, 读为“0”
- Bit 1~0 ACS1~ACS0:** 选择 A/D 通道位 (ACS4 为“0”)
 00: AN0
 01: AN1
 10: AN2
 11: AN3
 这两位是 A/D 通道选择控制位。由于只包含一个内部 A/D 转换电路, 因此通过这些位将 4 个 A/D 输入连接到转换器。如果 ADCR1 寄存器中的 ACS4 设为高, 内部 1.19V 电路将被连接到内部 A/D 转换器。

ADCR1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ACS4	V119EN	—	VREFS	—	ADCK2	ADCK1	ADCK0
R/W	R/W	R/W	—	R/W	—	R/W	R/W	R/W
POR	0	0	—	0	—	0	0	0

- Bit 7 ACS4:** 选择内部 1.19V 作为 A/D 输入控制位
 0: 除能
 1: 使能
 此位使能 1.19V 连接到 A/D 转换器。V119EN 位必须先被置位使能 1.19V Bandgap 电路被用于 A/D 转换器。当 ACS4 设为高, 1.19V Bandgap 电压将连接到 A/D 转换器, 其它 A/D 输入通道断开。

- Bit 6 **V119EN**: 内部 1.19V 控制位
0: 除能
1: 使能
此位控制连接到 A/D 转换器的内部 Bandgap 电路开 / 关功能。当此位设为高, Bandgap 电压 1.19V 连接至 A/D 转换器。如果 1.19V 未连接至 A/D 转换器且 LVR 除能, Bandgap 参考电压电路自动关闭以减少功耗。当 1.19V 打开连接至 A/D 转换器, 在 A/D 转换动作执行前, Bandgap 电路稳定需一段时间 t_{BG} 。
- Bit 5 未定义, 读为 “0”
- Bit 4 **VREFS**: 选择 ADC 参考电压
0: 内部 ADC 电源
1: VREF 引脚
此位用于选择 A/D 转换器的参考电压。如果该位设为高, A/D 转换器参考电压来源于外部 VREF 引脚。如果该位设为低, 内部参考电压来源于电源电压 V_{DD} 。
- Bit 3 未定义, 读为 “0”
- Bit 2~0 **ADCK2~ADCK0**: 选择 ADC 时钟源
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: 未定义
这三位用于选择 A/D 转换器的时钟源。

ACERL 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	ACE3	ACE2	ACE1	ACE0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	1

- Bit 7~4 未定义, 读为 “0”
- Bit 3 **ACE3**: 定义 PD3 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN3
- Bit 2 **ACE2**: 定义 PD2 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN2
- Bit 1 **ACE1**: 定义 PD1 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN1
- Bit 0 **ACE0**: 定义 PD0 是否为 A/D 输入
0: 不是 A/D 输入
1: A/D 输入, AN0

A/D 操作

ADCR0 寄存器中的 START 位, 用于打开和复位 A/D 转换器。当单片机设定此位从逻辑低到逻辑高, 然后再到逻辑低, 就会开始一个模数转换周期。当 START 位从逻辑低到逻辑高, 但不再回到逻辑低时, ADCR0 寄存器中的 EOCB 位置 “1”, 复位模数转换器。START 位用于控制内部模数转换器的开

启动作。

ADCR0 寄存器中的 EOCB 位用于表明模数转换过程的完成。在转换周期结束后，EOCB 位会被单片机自动地置为“0”。此外，也会置位中断控制寄存器内相应的 A/D 中断请求标志位，如果中断使能，就会产生对应的内部中断信号。A/D 内部中断信号将引导程序到相应的 A/D 内部中断入口。如果 A/D 内部中断被禁止，可以让单片机轮询 ADCR0 寄存器中的 EOCB 位，检查此位是否被清除，以作为另一种侦测 A/D 转换周期结束的方法。

A/D 转换器的时钟源为系统时钟 f_{SYS} 分频，而分频系数由 ADCR1 寄存器中的 ADCK2~ADCK0 位决定。

虽然 A/D 时钟源是由系统时钟 f_{SYS} ，ADCK2~ADCK0 位决定，但可选择的最大 A/D 时钟源则有一些限制。允许的 A/D 时钟周期 t_{ADCK} 的范围为 $0.5\mu\text{s}$ ~ $10\mu\text{s}$ ，所以选择系统时钟速度时必须小心。例如，如果系统时钟速度为 4MHz 时，ADCK2~ADCK0 位不能设为“000”和“110”。必须保证设定的 A/D 转换时钟周期不小于时钟周期的最小值或大于时钟周期的最大值，否则将会产生不准确的 A/D 转换值。使用者可以参考下面的表格，被标上星号 * 的数值是不允许的，因为它们 A/D 转换时钟周期小于规定的最小值。

f_{SYS}	A/D 时钟周期 (t_{ADCK})							
	ADCK2, ADCK1, ADCK0 =000 (f_{SYS})	ADCK2, ADCK1, ADCK0 =001 ($f_{\text{SYS}}/2$)	ADCK2, ADCK1, ADCK0 =010 ($f_{\text{SYS}}/4$)	ADCK2, ADCK1, ADCK0 =011 ($f_{\text{SYS}}/8$)	ADCK2, ADCK1, ADCK0 =100 ($f_{\text{SYS}}/16$)	ADCK2, ADCK1, ADCK0 =101 ($f_{\text{SYS}}/32$)	ADCK2, ADCK1, ADCK0 =110 ($f_{\text{SYS}}/64$)	ADCK2, ADCK1, ADCK0 =111
1MHz	1 μs	2 μs	4 μs	8 μs	16 μs *	32 μs *	64 μs *	未定义
2MHz	500ns	1 μs	2 μs	4 μs	8 μs	16 μs *	32 μs *	未定义
4MHz	250ns*	500ns	1 μs	2 μs	4 μs	8 μs	16 μs *	未定义
8MHz	125ns*	250ns*	500ns	1 μs	2 μs	4 μs	8 μs	未定义
12MHz	83ns*	167ns*	333ns*	667ns	1.33 μs	2.67 μs	5.33 μs	未定义
16MHz	62.5ns*	125ns*	250ns*	500ns	1 μs	2 μs	4 μs	未定义

A/D 时钟周期范例

ADCR0 寄存器的 ADOFF 位用于控制 A/D 转换电路电源的开 / 关。该位必须清零以开启 A/D 转换器电源。即使通过清除 ACERL 寄存器的 ACE3~ACE0 位，选择无引脚作为 A/D 输入，如果 ADOFF 设为“0”，那么仍然会产生功耗。因此当未使用 A/D 转换器功能时，在功耗敏感的应用中建议设置 ADOFF 为高以减少功耗。

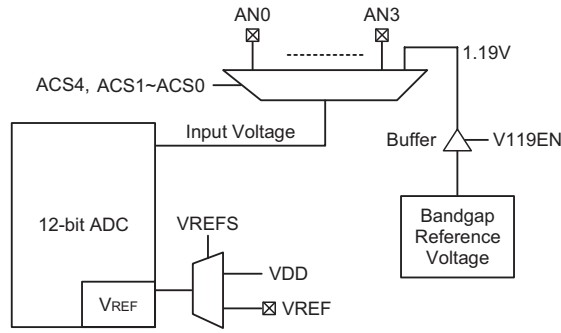
A/D 转换器参考电压来自正电源电压引脚 VDD 或外部参考源引脚 VREF，可通过 VREFS 位来选择。由于 VREF 引脚与其它功能共用，当 VREFS 设为高，选择 VREF 引脚功能且其它引脚功能将自动除能。

A/D 输入引脚

所有的 A/D 模拟输入引脚都与 PD0~PD3 引脚及其它功能共用。使用 ACERL 寄存器中的 ACE3~ACE0 位，可以将它们设置为 A/D 转换器模拟输入脚或具有其它功能。如果引脚的对应位 ACE3~ACE0 设为高，那么该引脚作为 A/D 转换输入且原引脚功能除能。通过这种方式，引脚的功能可由程序来控制，灵活地切换引脚功能。如果将引脚设为 A/D 输入，则通过寄存器编程设置的所有上拉电阻会自动断开。请注意，PAC 端口控制寄存器不需要为使能 A/D 输入而先设定为输入模式，当 ACE3~ACE0 位使能 A/D 输入时，端口控制寄存器的状态将

被重置。

A/D 转换器有自己的参考电压引脚 VREF，而通过设置 ADCR1 寄存器的 VREFS 位，参考电压也可以选择来自电源电压引脚。模拟输入值一定不能超过 V_{REF} 值。



A/D 输入结构

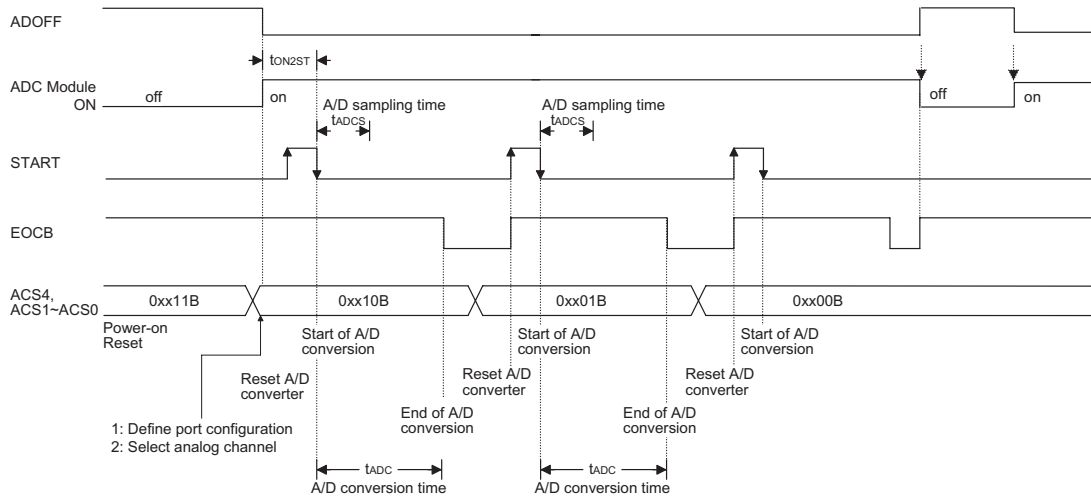
A/D 转换步骤

下面概述实现 A/D 转换过程的各个步骤。

- 步骤 1
通过 ADCR1 寄存器中的 ADCK2~ADCK0 位，选择所需的 A/D 转换时钟。
- 步骤 2
清零 ADCR0 寄存器中的 ADOFF 位使能 A/D。
- 步骤 3
通过 ADCR1 和 ADCR0 寄存器中的 ACS4 和 ACS1~ACS0 位，选择连接至内部 A/D 转换器的通道。
- 步骤 4
通过 ACERL 寄存器中的 ACE3~ACE0 位，选择哪些引脚规划为 A/D 输入引脚。
- 步骤 5
如果要使用中断，则中断控制寄存器需要正确地设置，以确保 A/D 转换功能是激活的。总中断控制位 EMI 需要置位为“1”，以及 A/D 转换器中断位 ADE 也需要置位为“1”。
- 步骤 6
现在可以通过设定 ADCR0 寄存器中的 START 位从“0”到“1”再回到“0”，开始模数转换的过程。注意，该位需初始化为“0”。
- 步骤 7
可以轮询 ADCR0 寄存器中的 EOCB 位，检查模数转换过程是否完成。当此位成为逻辑低时，表示转换过程已经完成。转换完成后，可读取 A/D 数据寄存器 ADRL 和 ADRH 获得转换后的值。另一种方法是，若中断使能且堆栈未满，则程序等待 A/D 中断发生。

注：若使用轮询 ADCR0 寄存器中 EOCB 位的状态的方法来检查转换过程是否结束时，则中断使能的步骤可以省略。

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



A/D 转换时序图

编程注意事项

在编程时，如果 A/D 转换器未使用，通过设置 ADCR0 寄存器中的 ADOFF 为高，关闭 A/D 内部电路以减少电源功耗。此时，无论输入脚的模拟电压为何值，内部 A/D 转换器电路不产生功耗。如果 A/D 转换器输入脚用作普通 I/O 引脚，必须特别注意，输入电压为无效逻辑电平也可能增加功耗。

A/D 转换功能

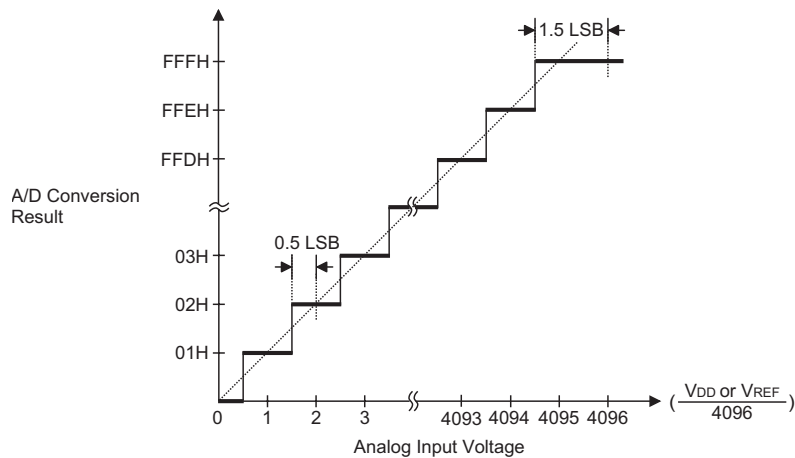
单片机含有一组 12 位的 A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于 V_{DD} 或 V_{REF} 的电压值，因此每一位可表示 $(V_{DD} \text{ 或 } V_{REF}) / 4096$ 的模拟输入值。

$$1 \text{ LSB} = (V_{DD} \text{ 或 } V_{REF}) \div 4096$$

通过下面的等式可估算 A/D 转换器输入电压值：

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

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



理想的 A/D 转换功能

A/D 转换应用范例

下面两个范例程序用来说明怎样使用 A/D 转换。第一个范例是轮询 ADCR0 寄存器中的 EOCB 位来判断 A/D 转换是否完成；第二个范例则使用中断的方式判断。

范例：使用查询 EOCB 的方式来检测转换结束

```

clr ADE                ; disable A/D interrupt
mov a,03H
mov ADCR1,a            ; select fsys/8 as A/D clock and switch off 1.19V
clr ADOFF
mov a,0Fh              ; setup ACERL to configure pins AN0~AN3
mov ACERL,a
mov a,00h
mov ADCR0,a            ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr START              ; high pulse on start bit to initiate conversion
set START              ; reset A/D
clr START              ; start A/D
polling_EOC:
sz EOCB                ; poll the ADCR0 register EOCB bit to detect end
                        ; of A/D conversion
jmp polling_EOC        ; continue polling
mov a,ADRL              ; read low byte conversion result value
mov ADRL_buffer,a      ; save result to user defined register
mov a,ADRH              ; read high byte conversion result value
mov ADRH_buffer,a      ; save result to user defined register
:
:
jmp start_conversion   ; start next A/D conversion

```

范例：使用中断的方式来检测转换结束

```

clr ADE                ; disable A/D interrupt
mov a,03H
mov ADCR1,a            ; select fsys/8 as A/D clock and switch off 1.19V
clr ADOFF
mov a,0Fh              ; setup ACERL to configure pins AN0~AN3
mov ACERL,a
mov a,00h
mov ADCR0,a            ; enable and connect AN0 channel to A/D converter
Start_conversion:
clr START              ; high pulse on start bit to initiate conversion
set START              ; reset A/D
clr START              ; start A/D
clr ADF                ; clear A/D interrupt request flag
set ADE                ; enable A/D interrupt
set EMI                ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov acc_stack,a        ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a     ; save STATUS to user defined memory
:
:

```

```

mov a,ADRL                ; read low byte conversion result value
mov adrl_buffer,a         ; save result to user defined register
mov a,ADRH                ; read high byte conversion result value
mov adrh_buffer,a         ; save result to user defined register
:
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a              ; restore STATUS from user defined memory
mov a, acc_stack          ; restore ACC from user defined memory
reti

```

触控按键功能

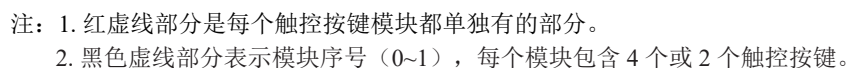
该单片机提供多触控按键功能。该触控按键功能完全内部集成不需外接元件，通过内部寄存器对其进行简单的操作。

触控按键结构

触控按键与 PB 的 I/O 引脚共用。通过寄存器的位来选择此功能。按键被分成两组，即 M0~M1 两个模块。每个模块为独立的一组包含四个触控按键（M1 模块只有两个按键）且每个按键有各自的振荡器。每个模块具有单独的控制逻辑电路和寄存器设置。寄存器的名称和它相关的模块编号相对应。

按键个数	触控按键模块	触控按键	共用 I/O 口
6	M0	KEY1~KEY4	PB0~PB3
	M1	KEY5~KEY6	PB4~PB5

触控按键结构



触控开关模块方框图

触控按键寄存器

每个触控按键模块包含四个或两个触控按键功能，且都有配套的寄存器系列。以下表格显示了每个触控按键模块的寄存器系列。寄存器名称里的 Mn 对照触控按键模块的序号 M0~M1。

寄存器名称	Bit							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	—	TKRCOV	TKST	TKCFOV	TK16OV	TSCS	TK16S1	TK16S0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKC1	—	—	—	—	—	—	TKFS1	TKFS0
TKMn16DL	D7	D6	D5	D4	D3	D2	D1	D0
TKMn16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKMnROL	D7	D6	D5	D4	D3	D2	D1	D0
TKMnROH	—	—	—	—	—	—	D9	D8
TKM0C0	M0MXS1	M0MXS0	M0DFEN	M0FILEN	M0SOFC	M0SOF2	M0SOF1	M0SOF0
TKM0C1	M0TSS	—	M0ROEN	M0KOEN	M0K4IO	M0K3IO	M0K2IO	M0K1IO
TKM1C0	—	M1MXS0	M1DFEN	M1FILEN	M1SOFC	M1SOF2	M1SOF1	M1SOF0
TKM1C1	M1TSS	—	M1ROEN	M1KOEN	—	—	M1K2IO	M1K1IO

触控按键寄存器列表 (n=0~1)

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 触控按键 8 位定时 / 计数器寄存器
时隙计数器溢出设置的时间为 $(256 - \text{TKTMR}[7:0]) \times 32$

TKC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	TKRCOV	TKST	TKCFOV	TK16OV	TSCS	TK16S1	TK16S0
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 未使用，读为“0”

Bit 6 **TKRCOV**: 时隙计数器溢出标志位

0: 无溢出

1: 溢出

如果模块 0 或所有模块时隙计数器（通过 TSCS 位选择）溢出，则触控按键中断请求标志位 (TKMF) 将会被置位且所有模块按键振荡器和参考振荡器自动停止。所有模块的 16 位 C/F 计数器、16 位计数器、5 位时隙计数器和 8 位时隙时钟计数器都会自动关闭。

注意：该位可通过应用程序置高，但触控按键中断标志位不能通过应用程序置高。该位必须通过应用程序清零。

- Bit 5 **TKST**: 开启触控按键检测控制位
0: 停止
0 → 1: 开启
当该位为“0”时, 所有模块的 16 位 C/F 计数器、16 位计数器和 5 位时隙计数器会自动清零 (8 位可编程时隙计数器不清, 由用户设置溢出时间)。当该位由低转为高时, 16 位 C/F 计数器、16 位计数器、5 位时隙计数器和 8 位时隙时钟计数器都会自动开启, 并使能按键振荡器和参考振荡器输出时钟输入到这些计数器。
- 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
该位必须通过应用程序清零。
- Bit 3 **TK16OV**: 触控按键模块 16 位计数器溢出标志位
0: 无溢出
1: 溢出
该位必须通过应用程序清零。
- Bit 2 **TSCS**: 触控按键时隙计数器选择位
0: 每个模块使用自己的时隙计数器
1: 所有触控按键模块使用模块 0 的时隙计数器
- Bit 1~0 **TK16S1~TK16S0**: 触控按键模块 16 位计数器时钟选择位
00: f_{SYS}
01: $f_{SYS}/2$
10: $f_{SYS}/4$
11: $f_{SYS}/8$

TKC1 寄存器

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

Bit 7~2 未使用, 读为“0”

- Bit 1~0 **TKFS1~TKFS0**: 触控按键振荡器频率选择位
00: 500kHz
01: 1000 kHz
10: 1500 kHz
11: 2000 kHz

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**: 触控按键模块 16 位计数器低字节内容

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:** 触控按键模块 16 位计数器高字节内容

TKMn16DL 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:** 模块 n 16 位计数器低字节内容

TKMn16DH 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:** 模块 n 16 位计数器高字节内容

TKMnROL 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:** 参考振荡器内建电容选择
振荡器内建电容选择为 (TKMnRO[9:0] x 50pF)/1024

TKMnROH 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 未使用，读为“0”

Bit 1~0 **D9~D8:** 参考振荡器内建电容选择
振荡器内建电容选择为 (TKMnRO[9:0] x 50pF)/1024

TKM0C0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	M0MXS1	M0MXS0	M0DFEN	M0FILEN	M0SOFC	M0SOF2	M0SOF1	M0SOF0
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 **M0MXS1~M0MXS0**: 模块 0 复用按键选择
00: KEY1
01: KEY2
10: KEY3
11: KEY4
- Bit 5 **M0DFEN**: 模块 0 倍频功能控制
0: 除能
1: 使能
- Bit 4 **M0FILEN**: 模块 0 滤波器功能控制
0: 除能
1: 使能
- Bit 3 **M0SOFC**: 模块 0 C-F 振荡器跳频功能选择
0: 由软件处理跳频功能, M0SOF2~M0SOF0 位决定 C-F 振荡器微调频率
1: 由硬件处理跳频功能, M0SOF2~M0SOF0 位无作用
- Bit 2~0 **M0SOF2~M0SOF0**: 由软件选择按键振荡器或参考振荡器频率作为 C-F 振荡器频率
000: 1380kHz
001: 1500kHz
010: 1670kHz
011: 1830 kHz
100: 2000 kHz
101: 2230 kHz
110: 2460 kHz
111: 2740 kHz
上述频率会依外挂或内建电容值的不同而变化。按键振荡器频率选择为 2MHz, 用户选择其它频率时可依比例调整。

TKM0C1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	M0TSS	—	M0ROEN	M0KOEN	M0K4IO	M0K3IO	M0K2IO	M0K1IO
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 **M0TSS**: 模块 0 时隙计数器选择
0: 参考振荡器
1: $f_{\text{SYS}}/4$
- Bit 6 未使用, 读为“0”
- Bit 5 **M0ROEN**: 模块 0 参考振荡器控制
0: 除能
1: 使能
- Bit 4 **M0KOEN**: 模块 0 按键振荡器控制
0: 除能
1: 使能

- Bit 3 **M0K4IO**: 触控按键 4 功能控制
0: 除能
1: 使能
- Bit 2 **M0K3IO**: 触控按键 3 功能控制
0: 除能
1: 使能
- Bit 1 **M0K2IO**: 触控按键 2 功能控制
0: 除能
1: 使能
- Bit 0 **M0K1IO**: 触控按键 1 功能控制
0: 除能
1: 使能

TKM1C0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	M1MXS0	M1DFEN	M1FILEN	M1SOFC	M1SOF2	M1SOF1	M1SOF0
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 未使用，读为“0”
- Bit 6 **M1MXS0**: 模块 1 复用按键选择
0: KEY5
1: KEY6
- Bit 5 **M1DFEN**: 模块 1 倍频功能控制
0: 除能
1: 使能
- Bit 4 **M1FILEN**: 模块 1 滤波器功能控制
0: 除能
1: 使能
- Bit 3 **M1SOFC**: 模块 1 C-F 振荡器跳频功能选择
0: 由软件处理跳频功能，M1SOF2~M1SOF0 位决定 C-F 振荡器微调频率
1: 由硬件处理跳频功能，M1SOF2~M1SOF0 位无作用
- Bit 2~0 **M1SOF2~M1SOF0**: 由软件选择按键振荡器或参考振荡器频率作为 C-F 振荡器频率
000: 1380kHz
001: 1500kHz
010: 1670kHz
011: 1830 kHz
100: 2000 kHz
101: 2230 kHz
110: 2460 kHz
111: 2740 kHz

上述频率会依外挂或内建电容值的不同而变化。按键振荡器频率选择为 2MHz，用户选择其它频率时可依比例调整。

TKM1C1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	M1TSS	—	M1ROEN	M1KOEN	—	—	M1K2IO	M1K1IO
R/W	R/W	—	R/W	R/W	—	—	R/W	R/W
POR	0	—	0	0	—	—	0	0

- Bit 7 **M1TSS**: 模块 1 时限计数器选择
 0: 参考振荡器
 1: $f_{sys}/4$
- Bit 6 未使用, 读为 “0”
- Bit 5 **M1ROEN**: 模块 1 参考振荡器控制
 0: 除能
 1: 使能
- Bit 4 **M1KOEN**: 模块 1 按键振荡器控制
 0: 除能
 1: 使能
- Bit 3~2 未使用, 读为 “0”
- Bit 1 **M1K2IO**: 触控按键 6 功能控制
 0: 除能
 1: 使能
- Bit 0 **M1K1IO**: 触控按键 5 功能控制
 0: 除能
 1: 使能

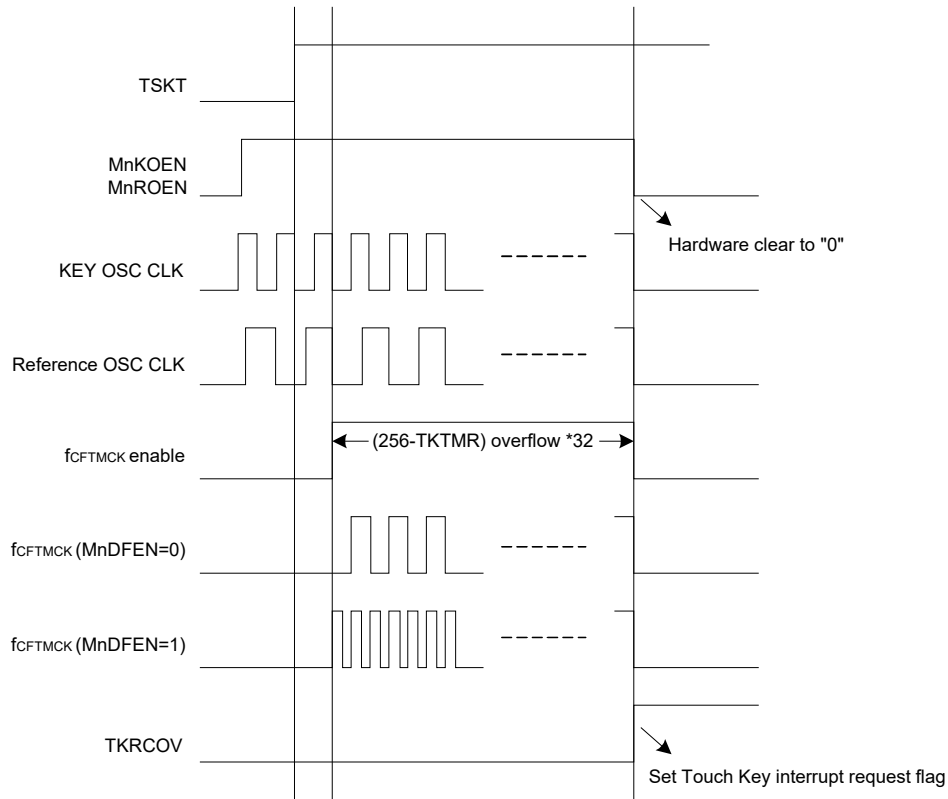
触控按键操作

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

每个触控按键模块包含四个或两个与 I/O 引脚共用的触控按键。通过寄存器可设置相应引脚功能。每个触控按键具有独立的感应振荡器, 因此每个模块包含四个或两个感应振荡器。

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发生。在最后一个时间间隔后, 会产生一个触控按键中断信号。

触控按键感应振荡器和参考振荡器的时序如下图所示。



触控按键时序方框图

触控按键中断

触控按键只有一个中断，所有触控按键模块的时隙计数器都溢出时，或模块 0 时隙计数器溢出时，才产生中断，此时所有模块的 16 位 C/F 计数器、16 位计数器，5 位时隙计数器和 8 位时隙计数器会自动清零。这里所有的触控按键是指已使能的触控按键。

任何一个触控按键模块的 16 位 C/F 计数器溢出就会把 16 位 C/F 计数器溢出标志位 TKCFOV 设为“1”，此标志位不会自动复位，必须通过应用程序将其清零。

模块 0 中只包含一组 16 位计数器，16 位计数器溢出就会把 16 位计数器溢出标志位 TK16OV 设为“1”，此标志位不会自动复位，必须通过应用程序将其复位。触控按键中断的详细说明见中断部分。更多细节详见本 datasheet 中断章节的触控按键中断。

编程注意事项

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

当外部触控按键的大小和布局确定时，其相关的电容将决定感应振荡器的频率。

串行接口模块 – SIM

该单片机内有一个串行接口模块，包括两种易与外部设备通信的串行接口：四线 SPI 或两线 I²C 接口。这两种接口具有相当简单的通信协议，单片机可以通过这些接口与传感器、闪存或 EEPROM 内存等硬件设备通信。SIM 接口的引脚与其它的 I/O 引脚共用，所以要先通过设置 SIMC0 寄存器的 SIMEN 位选择使用 SIM 功能引脚。因为这两种接口共用引脚和寄存器，所以要通过 SIMC0 寄存器中的 SIM2~SIM0 位来选择哪一种通信接口。

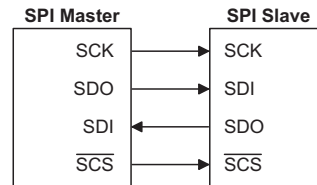
SPI 接口

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

SPI 通信模式为全双工模式，且能以主 / 从模式的工作方式进行通信，单片机既可以做为主机，也可以做为从机。虽然 SPI 接口理论上允许一个主机控制多个从机，但此处的 SPI 中只有一个片选信号引脚 $\overline{SCS}$ 。若主机需要控制多个从机，可使用输入 / 输出引脚选择从机。

SPI 接口操作

SPI 接口是一个全双工串行数据传输器。SPI 接口的四线为：SDI、SDO、SCK 和 $\overline{SCS}$ 。SDI 和 SDO 是数据的输入和输出线。SCK 是串行时钟线， $\overline{SCS}$ 是从机的选择线。SPI 的接口引脚与普通 I/O 口和 I²C 的功能脚共用。通过设定 SIMC0/SIMC2 寄存器的对应位，来使能 SPI 接口。连接到 SPI 接口的单片机以从主 / 从模式进行通信，且主机完成所有的数据传输初始化，并控制时钟信号。由于单片机只有一个 $\overline{SCS}$ 引脚，所以只能拥有一个从机设备。可通过软件控制 $\overline{SCS}$ 引脚使能与除能，设置 CSEN 位为“1”使能 $\overline{SCS}$ 功能，设置 CSEN 位为“0” $\overline{SCS}$ 引脚将作为普通 I/O 口功能。

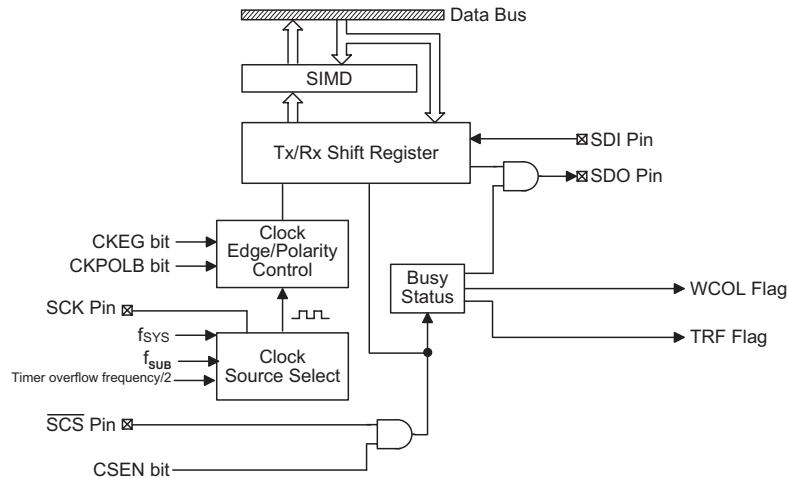


SPI 主 / 从机连接方式

该单片机的 SPI 功能具有以下特点：

- 全双工同步数据传输
- 主从模式
- 最低有效位先传或最高有效位先传的数据传输模式
- 传输完成标志位
- 时钟源上升沿或下降沿有效

SPI 接口状态受很多因素的影响，如单片机处于主机或从机的工作模式以及 CSEN 和 SIMEN 位的状态。



SPI 方框图

SPI 寄存器

有三个内部寄存器用于控制 SPI 接口的所有操作，其中有一个数据寄存器 SIMD、两个控制寄存器 SIMC0 和 SIMC2。注意，SIMC1 寄存器仅用于 I²C 接口。

寄存器名称	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	—	—	SIMEN	—
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMC2	—	—	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF

SIM 寄存器列表

SIMD 用于存储发送和接收的数据。这个寄存器由 SPI 和 I²C 功能所共用。在单片机尚未将数据写入到 SPI 总线中时，要传输的数据应先存在 SIMD 中。SPI 总线接收到数据之后，单片机就可以从 SIMD 数据寄存器中读取。所有通过 SPI 传输或接收的数据都必须通过 SIMD 实现。

• SIMD 寄存器

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

“x” 为未知

单片机中也有两个控制 SPI 接口功能的寄存器，SIMC0 和 SIMC2。应注意的是 SIMC2 与 I²C 接口功能中的寄存器 SIMA 是同一个寄存器。SPI 功能不会用到寄存器 SIMC1，SIMC1 只适用于 I²C 中。寄存器 SIMC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。虽然 SIMC0 与 SPI 功能关，但是也用于控制外部时钟分频。寄存器 SIMC2 用于其它的控制功能如 LSB/MSB 选择，写冲突标志位等。

● SIMC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	—	—	SIMEN	—
R/W	R/W	R/W	R/W	—	—	—	R/W	—
POR	1	1	1	—	—	—	0	—

Bit 7~5 **SIM2~SIM0: SIM 工作模式控制位**

- 000: SPI 主机模式; SPI 时钟为 $f_{SYS}/4$
- 001: SPI 主机模式; SPI 时钟为 $f_{SYS}/16$
- 010: SPI 主机模式; SPI 时钟为 $f_{SYS}/64$
- 011: SPI 主机模式; SPI 时钟为 f_{SUB}
- 100: SPI 主机模式; SPI 时钟为 TIMER 溢出频率 /2
- 101: SPI 从机模式
- 110: I²C 从机模式
- 111: 保留

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

Bit 4~2 未使用, 读为 “0”

Bit 1 **SIMEN: SIM 控制位**

- 0: 除能
- 1: 使能

此位为 SIM 接口的开 / 关控制位。此位为 “0” 时, SIM 接口除能, SDI、SDO、SCK 和 SCS 或 SDA 和 SCL 脚将失去 SPI 或 I²C 功能, SIM 工作电流减小到最小值。此位为 “1” 时, SIM 接口使能。

若 SIM 经由 SIM2~SIM0 位设置为工作在 SPI 接口, 当 SIMEN 位由低到高转变时, SPI 控制寄存器中的设置不会发生变化, 其首先应在应用程序中初始化。若 SIM 经由 SIM2~SIM0 位设置为工作在 I²C 接口, 当 SIMEN 位由低到高转变时, I²C 控制寄存器中的设置, 如 HXT 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 未使用, 读为 “0”

● SIMC2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
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 **CKPOLB: 时钟线的基础状态位**

- 0: 当时钟无效时, SCK 口为高电平
- 1: 当时钟无效时, SCK 口为低电平

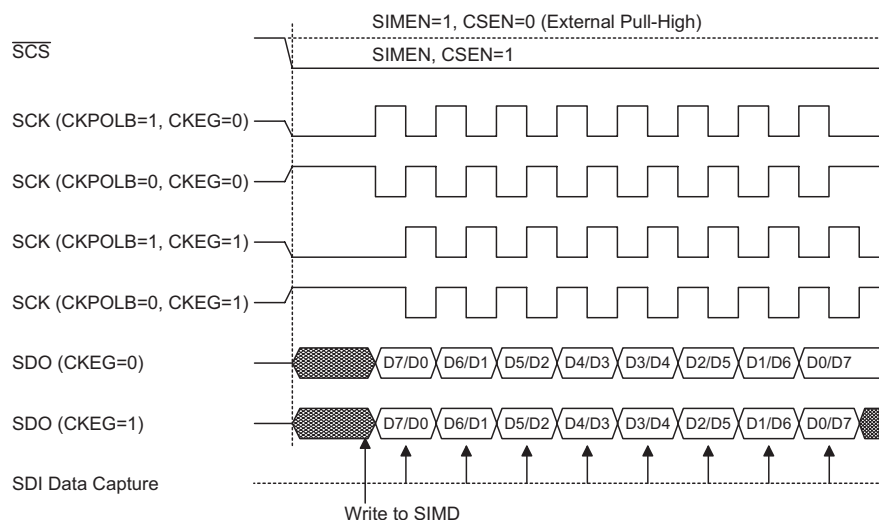
此位决定了时钟线的基础状态。当时钟无效时, 若此位为高, SCK 为低电平, 若此位为低, SCK 为高电平。

Bit 4	CKEG: SPI 的 SCK 有效时钟边沿类型位 CKPOLB=0 0: SCK 为高电平且在 SCK 上升沿抓取数据 1: SCK 为高电平且在 SCK 下降沿抓取数据 CKPOLB=1 0: SCK 为低电平且在 SCK 下降沿抓取数据 1: SCK 为低电平且在 SCK 上升沿抓取数据 CKEG 和 CKPOLB 位用于设置 SPI 总线上时钟信号输入和输出方式。在执行数据传输前, 这两位必须被设置, 否则将产生错误的时钟边沿信号。CKPOLB 位决定时钟线的基本状态, 若时钟无效且此位为高, 则 SCK 为低电平, 若时钟无效且此位为低, 则 SCK 为高电平。CKEG 位根据 CKPOLB 位的状态决定有效时钟边沿类型。
Bit 3	MLS: SPI 数据移位命令位 0: LSB 优先 1: MSB 优先 数据移位选择位, 用于选择数据传输时优先传输还是低位优先传输。此位设置为高时高位优先传输, 为低时低位优先传输。
Bit 2	CSEN: SPI $\overline{\text{SCS}}$ 引脚控制位 0: 除能 1: 使能 CSEN 位用于 $\overline{\text{SCS}}$ 引脚的使能 / 除能控制。此位为低时, $\overline{\text{SCS}}$ 除能并作为普通 I/O 口或其他功能。此位为高时, SCS 使能并作为选择脚。
Bit 1	WCOL: SPI 写冲突标志位 0: 无冲突 1: 冲突 WCOL 标志位用于监测数据冲突的发生。此位为高时, 表示数据传输过程中有新数据写入 SIMD 寄存器。若数据正在被传输, 此操作无效。此位可被应用程序清零。
Bit 0	TRF: SPI 发送 / 接收结束标志位 0: 数据正在发送 1: 数据发送结束 TRF 位为发送 / 接收结束标志位, 当 SPI 数据传输结束时, 此位自动置为高, 但须通过应用程序设置为“0”。此位也可用于产生中断。

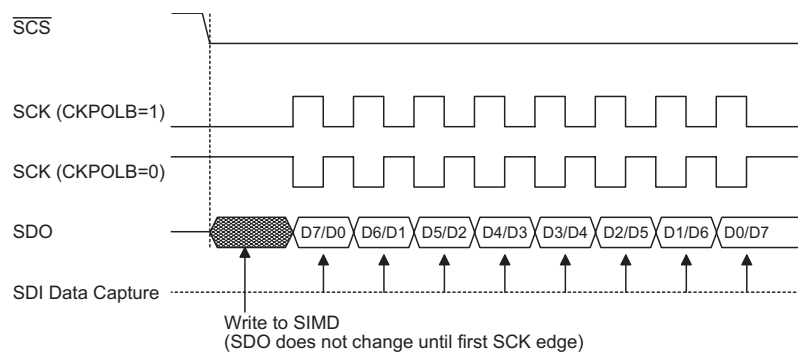
SPI 通信

将 SIMEN 设置为高, 使能 SPI 功能之后, 单片机处于主机模式, 当数据写入到寄存器 SIMD 的同时传输 / 接收开始进行。数据传输完成时, TRF 位将自动被置位但清除只能通过应用程序完成。单片机处于从机模式时, 收到主机发来的信号之后, 会传输 SIMD 中的数据, 而且在 SDI 引脚上的数据也会被移位到 SIMD 寄存器中。主机应在输出时钟信号之前先输出一个 $\overline{\text{SCS}}$ 信号以使能从机, 从机的数据传输功能也应在与 $\overline{\text{SCS}}$ 信号相关的适当时候准备就绪, 这由 CKPOLB 和 CKEG 位决定。所附时序图表明了 CKPOLB 和 CKEG 位各种设置情况下从机数据与 $\overline{\text{SCS}}$ 信号的关系。

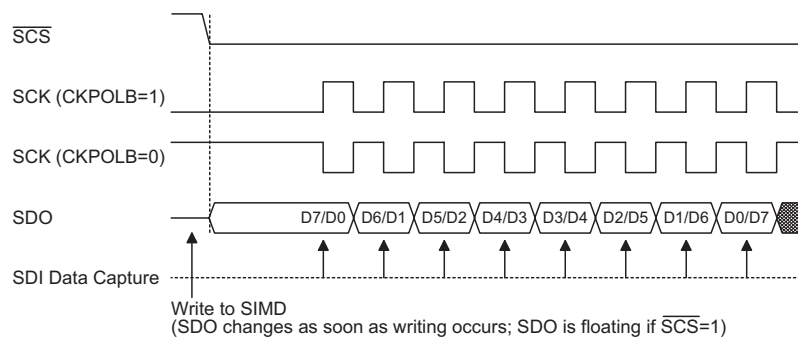
即使在单片机处于空闲模式, SPI 功能仍将继续执行。



SPI 主机模式时序

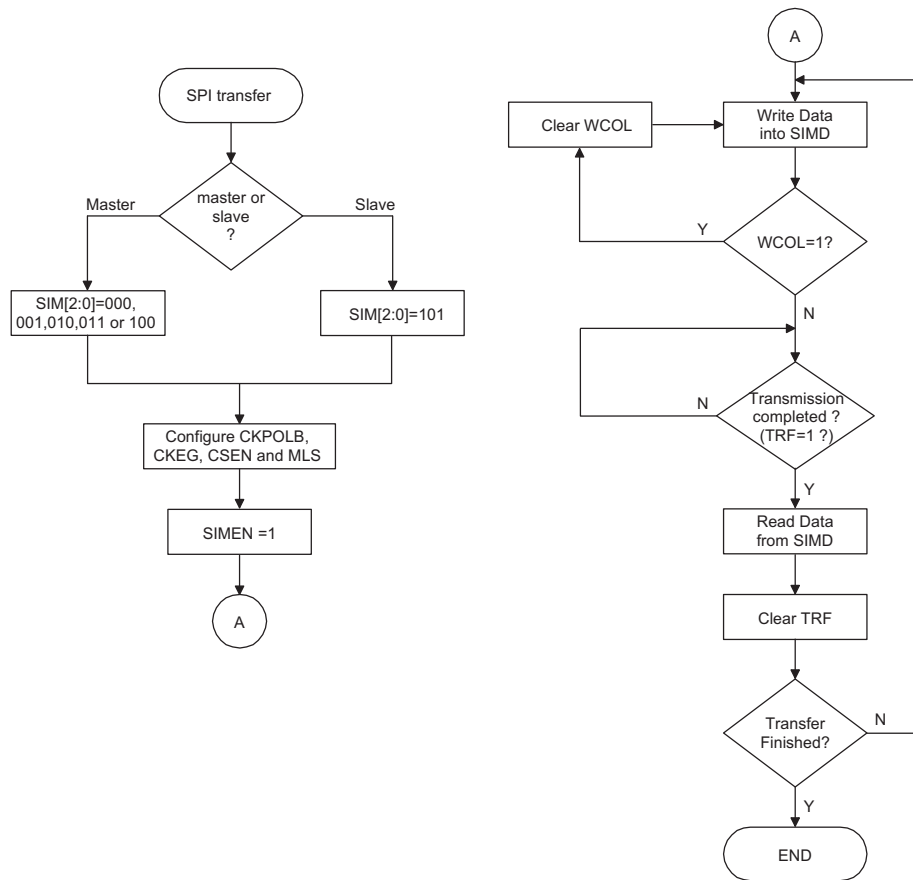


SPI 从机模式时序 – CKEG=0



Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always enabled and ignores the $\overline{SCS}$ level.

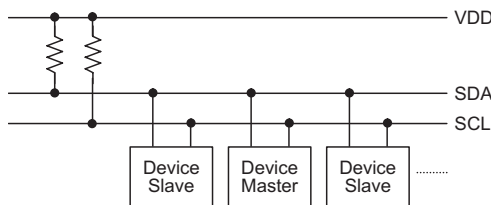
SPI 从机模式时序 – CKEG=1



SPI 传输控制流程图

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

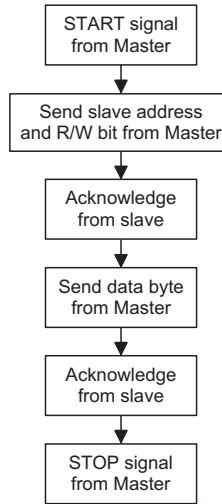
I²C 串行接口是一个双线的接口，有一条串行数据线 SDA 和一条串行时钟线 SCL。由于可能有多个设备在同一条总线上相互连接，所以这些设备的输出都是开漏型输出。因此应在这些输出上都应加上拉电阻。应注意的是：I²C 总线上的每个设备都没有选择线，但分别与唯一的地址一一对应，用于 I²C 通信。

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。

I²C 接口的去抖时间使用内部时钟在外部时钟上增加一个去抖间隔，会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，去抖时间为 2 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)
2 个系统时钟去抖时间	$f_{\text{SYS}} > 4\text{MHz}$	$f_{\text{SYS}} > 10\text{MHz}$

I²C 最小 f_{SYS} 频率



I²C 寄存器

I²C 总线的四个控制寄存器是 SIMC0, SIMC1, SIMA 和 I2CTOC 及一个数据寄存器 SIMD。SIMD 寄存器，SPI 章节中已有介绍，用于存储正在传输和接收的数据，当单片机将数据写入 I²C 总线之前，实际将被传输的数据存放在寄存器 SIMD 中。从 I²C 总线接收到数据之后，单片机就可以从寄存器 SIMD 中得到这个数据。I²C 总线上的所有传输或接收到的数据都必须通过 SIMD。SIM 引脚与 I/O 口共用，通过 SIMC0 寄存器中的 SIMEN 位来使能。

应注意的是 SIMA 也有另外一个名字，SIMC2，使用 SPI 功能时会用到。I²C 接口会用到寄存器 SIMC0 中的 SIMEN 位和 SIM0~SIM2 位。

寄存器名称	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	—	—	SIMEN	—
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	RNIC	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	A6	A5	A4	A3	A2	A1	A0	—
I2CTOC	I2CTOEN	I2CTOF	I2CTOS5	I2CTOS4	I2CTOS3	I2CTOS2	I2CTOS1	I2CTOS0

I²C 寄存器列表

● SIMC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	—	—	SIMEN	—
R/W	R/W	R/W	R/W	—	—	—	R/W	—
POR	1	1	1	—	—	—	0	—

Bit 7~5 **SIM2~SIM0: SIM 工作模式控制位**

- 000: SPI 主机模式; SPI 时钟为 $f_{sys}/4$
- 001: SPI 主机模式; SPI 时钟为 $f_{sys}/16$
- 010: SPI 主机模式; SPI 时钟为 $f_{sys}/64$
- 011: SPI 主机模式; SPI 时钟为 f_{sub}
- 100: SPI 主机模式; SPI 时钟为 TIMER 溢出频率 /2
- 101: SPI 从机模式
- 110: I²C 从机模式
- 111: 保留

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

Bit 4~2 未使用, 读为 “0”

Bit 1 **SIMEN: SIM 控制位**

- 0: 除能
- 1: 使能

此位为 SIM 接口的开 / 关控制位。此位为 “0” 时, SIM 接口除能, SDI、SDO、SCK 和 SCS 或 SDA 和 SCL 脚将作为普通 I/O 口功能, SIM 工作电流减小到最小值。此位为 “1” 时, SIM 接口使能。

若 SIM 经由 SIM2~SIM0 位设置为工作在 SPI 接口, 当 SIMEN 位由低到高转变时, SPI 控制寄存器中的设置不会发生变化, 其首先应在应用程序中初始化。若 SIM 经由 SIM2~SIM0 位设置为工作在 I²C 接口, 当 SIMEN 位由低到高转变时, I²C 控制寄存器中的设置, 如 HXT 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 未使用, 读为 “0”

● SIMC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	HCF	HAAS	HBB	HTX	TXAK	SRW	RNIC	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

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

此标志位用于决定从机地址是否与主机发送地址相同。若地址匹配此位为高, 否则此位为低。

- Bit 5 **HBB**: I²C 总线忙标志位
0: I²C 总线闲
1: I²C 总线忙
当检测到 START 信号时 I²C 忙, 此位变为高电平。当检测到 STOP 信号时 I²C 总线停止, 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 总线发送确认标志位
0: 从机发送确认标志
1: 从机没有发送确认标志
单片机接收 8 位数据之后会将该位在第九个时钟传到总线上。如果单片机想要接收更多的数据, 则应在接收数据之前将此位设置为“0”。
- Bit 2 **SRW**: I²C 从机读 / 写位
0: 从机应处于接收模式
1: 从机应处于发送模式
SRW 位是从机读写位。决定主机是否希望传输或接收来自 I²C 总线的的数据。当传输地址和从机的地址相同时, HAAS 位会被设置为高, 主机将检测 SRW 位来决定进入发送模式还是接收模式。如果 SRW 位为“1”时, 主机会请求从总线上读数据, 此时设备处于传输模式。当 SRW 位为“0”时, 主机往总线上写数据, 设备处于接收模式以读取该数据。
- Bit 1 **RNIC**: I²C 内部时钟使用控制位
0: I²C 使用内部时钟
1: I²C 未使用内部时钟
I²C 模块不需要内部时钟也可运行, 如果 SIM 中断使能会产生中断, 该中断可用于休眠模式、空闲模式和正常 (低速) 模式。如果此位为“1”且单片机处于 HALT 状态, 则从机接收方可以正常工作, 而从机发送方不能正常工作, 因为它需要系统时钟。
- Bit 0 **RXAK**: I²C 总线接收确认标志位
0: 从机接收到确认标志
1: 从机没有接收到确认标志
RXAK 位是接收确认标志位。如果 RXAK 位被重设为“0”即 8 位数据传输之后, 设备在第九个时钟有接受到一个正确的确认位。如果单片机处于发送状态, 发送方会检查 RXAK 位来判断接收方是否愿意继续接收下一个字节。因此直到 RXAK 为“1”时, 传输方停止发送数据。这时, 传输方将释放 SDA 线, 主机发出停止信号。

SIMD 用于存储发送和接收的数据。这个寄存器由 SPI 和 I²C 功能所共用。在单片机尚未将数据写入到 I²C 总线中时, 要传输的数据应存在 SIMD 中。I²C 总线接收到数据之后, 单片机就可以从 SIMD 数据寄存器中读取。所有通过 I²C 传输或接收的数据都必须通过 SIMD 实现。

• SIMD 寄存器

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

“x” 为未知

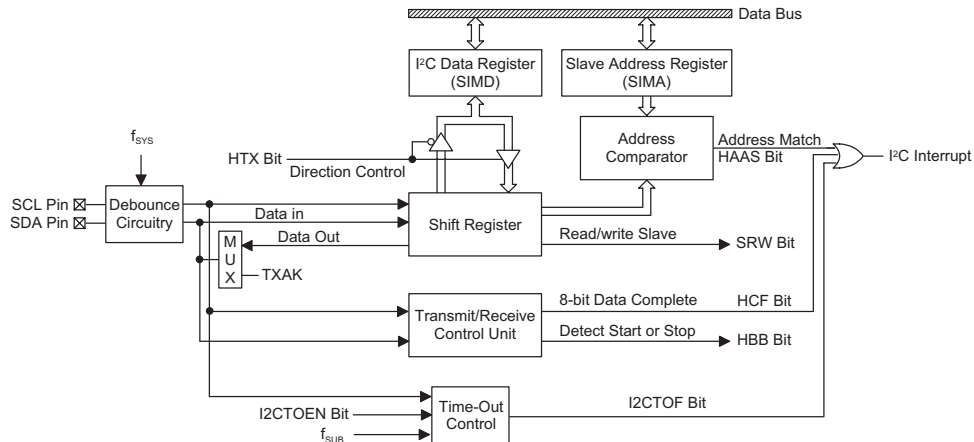
• SIMA 寄存器

Bit	7	6	5	4	3	2	1	0
Name	A6	A5	A4	A3	A2	A1	A0	—
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 **A6~A0: I²C 从机地址位**

A6~A0 是从机地址对应的 6~0 位。此寄存器也在 SPI 接口功能中使用，但其名称为 SIMC2。SIMA 寄存器用于存放 7 位从机地址，寄存器 SIMA 中的第 7~1 位是单片机的从机地址，位 0 未定义。如果接至 I²C 的主机发送处的地址和寄存器 SIMA 中存储的地址相符，那么就选中了这个从机。应注意的是寄存器 SIMA 和 SPI 接口使用的寄存器 SIMC2 是同一个寄存器。

Bit 0 未定义，读为“0”

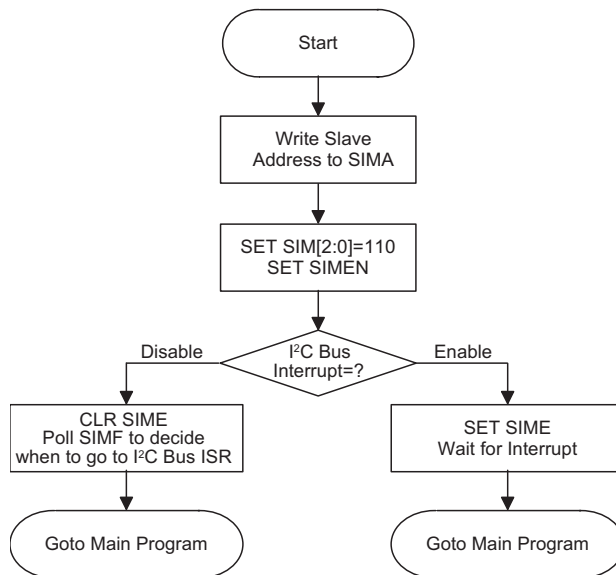


I²C 方框图

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会即将有数据到达。数据的前 7 位是从机地址，高位在前，低位在后。如果发出的地址和从机地址匹配，SIMC1 寄存器的 HAAS 位会被置位，同时产生 I²C 中断。进入中断服务程序后，系统要检测 HAAS 位和 I2CTOF 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。在数据传输中，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 位以确定主控制器是要进入发送模式还是接收模式。在 I²C 总线开始传送数据前，需要先初始化 I²C 总线，初始化 I²C 总线步骤如下：

- 步骤 1
设置 SIMC0 寄存器中 SIM2~SIM0 为“110”，SIMEN 位为“1”，以使能 I²C 总线。
- 步骤 2
向 I²C 总线地址寄存器 SIMA 写入从机地址。
- 步骤 3
设置中断控制寄存器中的 SIM 中断使能位 SIME，以使能 SIM 中断。



I²C 总线初始化流程图

I²C 总线起始信号

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

从机地址

总线上的所有从机都会侦测由主机发出的起始信号。发送起始信号后，紧接着主机将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 SIMC1 寄存器的 SRW 位，随后发出一个低电平应答信号（即第 9 位）。当单片机从机的地址匹配时，会将状态标志位 HAAS 置位。

I²C 总线有三个中断源，当程序运行至中断服务子程序时，通过检测 HAAS 位和 I2CTOF 位以确定 I²C 总线中断是来自从机地址匹配，还是来自 8 位数据传输完毕，或是来自 I²C 超时。当是从机地址匹配发生中断时，则从机或是用于发送模式并将数据写进 SIMD 寄存器，或是用于接收模式并从 SIMD 寄存器中读取空值以释放 SCL 线。

I²C 总线读 / 写信号

SIMC1 寄存器的 SRW 位用来表示主机是要从 I²C 总线上读取数据还是要将数据写到 I²C 总线上。从机则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。

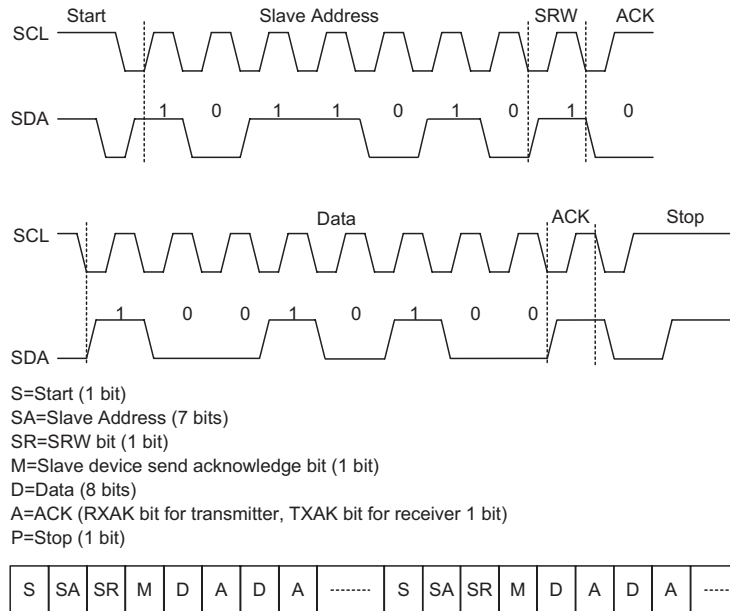
I²C 总线从机地址确认信号

主机发送呼叫地址后，当 I²C 总线上的任何从机内部地址与其匹配时，会发送一个应答信号。此应答信号会通知主机有从机已经接收到了呼叫地址。如果主机没有收到应答信号，则主机必须发送停止（STOP）信号以结束通信。当 HAAS 为高时，表示从机接收到的地址与自己内部地址匹配，则从机需检查 SRW 位，以确定自己是作为发送方还是作为接收方。如果 SRW 位为高，从机须设置成发送方，这样会置位 SIMC1 寄存器的 HTX 位。如果 SRW 位为低，从机须设置成接收方，这样会清零 SIMC1 寄存器的 HTX 位。

I²C 总线数据和确认信号

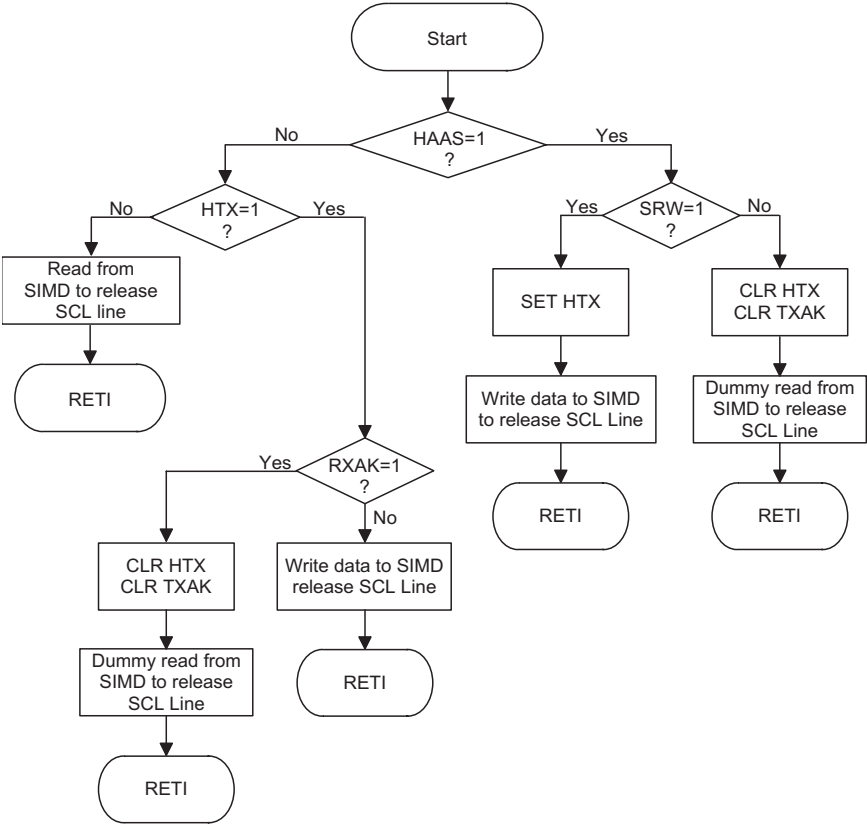
在从机确认接收到地址后，会进行 8 位宽度的数据传输。这个数据传输顺序是的高位在前，低位在后。接收方在接收到 8 位数据后必须发出一个应答信号（“0”）以继续接收下一个数据。如果发送方没接收到应答信号，发送方将释放 SDA 线，同时，主机将发出 STOP 信号以释放 I²C 总线。所传送的数据存储在 SIMD 寄存器中。如果设置成发送方，从机必须先将欲传输的数据写到 SIMD 寄存器中；如果设置成接收方，从机必须从 SIMD 寄存器读取数据。

当接收器想要继续接收下一个数据时，必须在第 9 个时钟发出应答信号（TXAK）。被设为发送方的从机将检测寄存器 SIMC1 中的 RXAK 位以判断是否传输下一个字节的数据，如果单片机不传输下一个字节，那么它将释放 SDA 线并等待接收主机的停止信号。



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

I²C 通信时序图



I²C 总线 ISR 流程图

I²C 超时控制

超时功能可减少 I²C 接收错误的时钟源而引起的锁死问题。如果连接到 I²C 总线的时钟源经过一会儿时间还未接收到，则在一定的周期后，I²C 电路和寄存器将复位。超时计数器在 I²C 总线“起始”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，如果此时间大于 I2CTOC 寄存器指定的周期，则超时状况发生。当 I²C “停止”条件发生时超时功能终止。当 I²C 计数器溢出时，计数器停止、I2CTOEN 位清零且 I2CTOF 位置为“1”，标志着超时发生。发生超时将产生一个中断且共用 I²C 中断向量。当 I²C 超时发生时，I²C 内部电路和寄存器都会复位，如下表所示：

寄存器	I ² C 超时发生后
SIMD, SIMA, SIMC0	无变化
SIMC1	复位到 POR 状态

I2CTOF 标志位由应用程序清零。共有 64 个超时周期，可通过 I2CTOC 寄存器的 I2CTOS5~I2CTOS0 位进行选择。超时周期可通过公式计算： $((1\sim64)\times(32/f_{SUB}))$ 。由此可得溢出周期范围为 1ms~64ms。要注意的是 LIRC 振荡器是一直使能的。

• I2CTOC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	I2CTOEN	I2CTOF	I2CTOS5	I2CTOS4	I2CTOS3	I2CTOS2	I2CTOS1	I2CTOS0
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 **I2CTOEN:** I²C 超时控制位

0: 除能

1: 使能

Bit 6 **I2CTOF:** I²C 超时标志位

0: 超时未发生

1: 超时发生

Bit 5~0 **I2CTOS5~I2CTOS0:** I²C 超时时间选择位

I²C 超时时钟源是 $f_{SUB}/32$

I²C 超时时间计算公式: $(I2CTOS[5:0]+1) \times (32/f_{SUB})$

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检测到触摸按键动作或定时 / 计数器溢出, 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 / 计数器、时基和 SIM 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器中的一系列寄存器控制的。寄存器总的分为两类。第一类是 INTC0~INTC1 寄存器, 用于设置基本的中断; 第二类是 INTEG 寄存器, 用于设置外部中断边沿触发类型。

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

功能	使能位	请求标志
总中断	EMI	—
INT 脚	INTE	INTF
触控按键模块	TKME	TKMF
定时 / 计数器	TE	TF
SIM	SIME	SIMF
时基	TBE	TBF
EEPROM	DEE	DEF
A/D 转换器	ADE	ADF

中断寄存器位命名模式

中断寄存器内容

Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	TF	TKMF	INTF	TE	TKME	INTE	EMI
INTC1	ADF	DEF	TBF	SIMF	ADE	DEE	TBE	SIME

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 未使用，读为“0”

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

INTC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	TF	TKMF	INTF	TE	TKME	INTE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 未使用，读为“0”

Bit 6 **TF**: 定时 / 计数器中断请求标志位
 0: 无请求
 1: 中断请求

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

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

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

Bit 2 **TKME**: 触控按键模块中断控制位
 0: 除能
 1: 使能

Bit 1 **INTE**: 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	ADF	DEF	TBF	SIMF	ADE	DEE	TBE	SIME
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 **ADF**: A/D 转换器中断请求标志位
 0: 无请求
 1: 中断请求
- 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 **SIMF**: SIM 中断请求标志位
 0: 无请求
 1: 中断请求
- Bit 3 **ADE**: A/D 转换器中断控制位
 0: 除能
 1: 使能
- Bit 2 **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 **SIME**: SIM 中断控制位
 0: 除能
 1: 使能

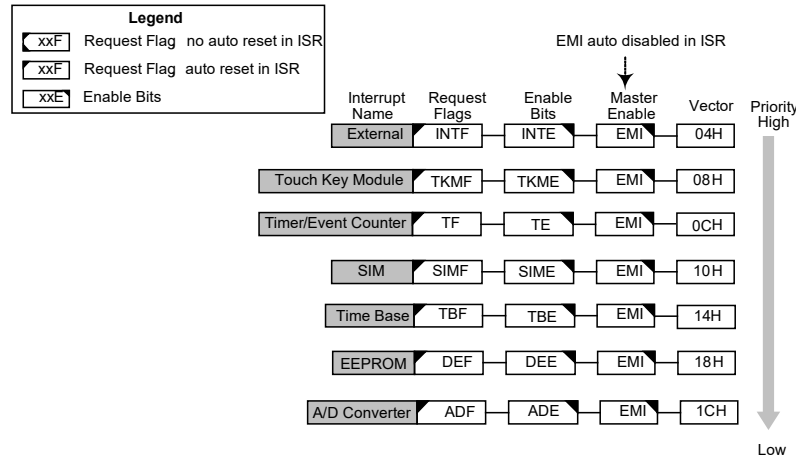
中断操作

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

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

各个中断使能位以及相应的请求标志位，以优先级的次序显示在下图。这里的所有中断源都有自己的向量中断向量。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，此引脚将被作为外部中断脚使用。此时该引脚必须通过设置控制寄存器，将该引脚设置为输入口。当中断使能，堆栈未满并且外部中断脚状态改变，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，其上拉电阻选择仍保持有效。

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。

时基中断

时基中断提供一个固定周期的中断信号，由定时器功能产生溢出信号控制。当中断请求标志 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，允许程序跳转到其中断向量地址。当中断使能，堆栈未满且时基溢出时，将调用相应的中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 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，时钟源来自内部时钟源 f_{TP} 。该 f_{TP} 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。控制时基中断频率 f_{TP} 的时钟源有几种，如在系统工作模式章节所示。

TBC 寄存器

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

Bit 7~6 未使用，读为“0”

Bit 5~4 **TB1~TB0**: 选择时基溢出周期位

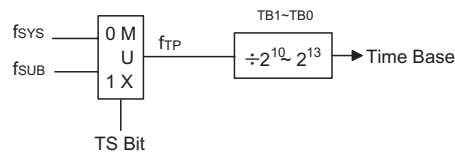
00: $1024/f_{TB}$

01: $2048/f_{TB}$

10: $4096/f_{TB}$

11: $8192/f_{TB}$

Bit 3~0 未使用，读为“0”



时基中断

定时 / 计数器中断

要使定时 / 计数器中断发生，总中断控制位 EMI 和相应的内部中断使能位 TE 必须先被置位。当定时 / 计数器溢出，相应的中断请求标志位 TF 将置位并触发定时 / 计数器中断。中断使能，堆栈未满，当定时 / 计数器溢出发生中断时，将调用位于计数器中断向量处的子程序。当响应中断服务子程序时，中断请求标志位 TF 会被自动复位且 EMI 位会被清零以除能其它中断。

EEPROM 中断

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

触控按键中断

要使触控按键中断发生，总中断控制位 EMI 和相应的时隙计数器的中断使能位 TKME 必须先被置位。当触控按键时隙计数器溢出时，时隙计数器中断请求标志 TKMF 将被置为“1”，触控按键中断产生。中断使能，堆栈未满，当触控按键时隙计数器溢出发生中断时，将调用相应中断向量处的子程序。当响应中断服务子程序时，中断请求标志位 TKMF 会被自动复位且 EMI 位会被清零以除能其它中断。

任何一个触控按键模块的 16 位 C/F 计数器溢出就会把 16 位 C/F 计数器溢出标志位 TKCFOV 设为“1”，此标志位不会自动复位，必须通过应用程序将其复位。模块 0 中只包含一组 16 位计数器，16 位计数器溢出就会把 16 位计数器溢出标志位 TK16OV 设为“1”，此标志位不会自动复位，必须通过应用程序将其复位。

串行接口模块中断

串行接口模块中断，即 SIM 中断。当一个字节数据已由 SIM 接口接收或发送完，或发生 I²C 地址匹配，或发生 I²C 超时，中断请求标志 SIMF 被置位，SIM 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI、串行接口中断使能位 SIME 需先被置位。当中断使能，堆栈未满且以上任何情况时，可跳转至相应中断向量子程序中执行。当串行接口中断响应，SIMF 标志自动清除，EMI 也将被自动清零以除能其它中断。

A/D 转换器中断

A/D 转换器中断由 A/D 转换动作的结束来控制。当 A/D 转换器中断请求标志被置位，即 A/D 转换过程完成时，中断请求发生。当总中断使能位 EMI 和 A/D 中断使能位 ADE 被置位，允许程序跳转到各自的中断向量地址。当中断使能，堆栈未满且 A/D 转换动作结束时，将调用相关的中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 ADF 会自动清零。EMI 位也会被清零以除能其它中断。

中断唤醒功能

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

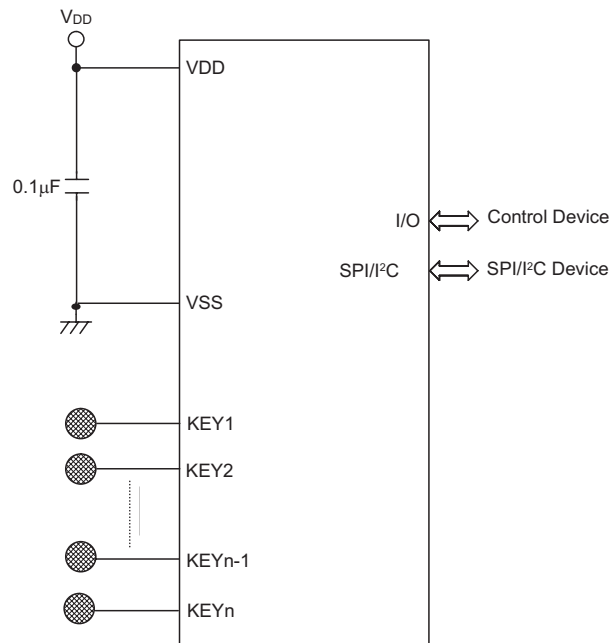
建议在中断服务子程序中不要使用“CALL 子程序”指令。中断通常发生在不可预料的情况或是需要立刻执行的某些应用。假如只剩下一层堆栈且没有控制好中断，当“CALL 子程序”在中断服务子程序中执行时，将破坏原来的控制序列。

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

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

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

应用电路



指令集

简介

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

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

指令周期

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

数据的传送

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

算术运算

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

逻辑和移位运算

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

分支和控制转换

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

位运算

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

查表运算

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

其它运算

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

指令集概要

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
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

助记符	说明	指令周期	影响标志位
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
CALL addr	子程序调用	2	无
RET	从子程序返回	2	无
RET A, x	从子程序返回，并将立即数放入 ACC	2	无
RETI	从中断返回	2	无
查表			
TABRD [m]	读取特定页或当前页的 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
SWAP [m]	交换数据存储器的高低字节，结果放入数据存储器	1 ^注	无
SWAPA [m]	交换数据存储器的高低字节，结果放入 ACC	1	无
HALT	进入暂停模式	1	TO, PDF

注: 1. 对跳转指令而言，如果比较的结果牵涉到跳转即需 2 个周期，如果没有发生跳转，则只需一个周期。

2. 任何指令若要改变 PCL 的内容将需要 2 个周期来执行。

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第 i 位内容清零。
功能表示	$[m].i \leftarrow 0$
影响标志位	无
CLR WDT	Clear Watchdog Timer
指令说明	WDT 计数器、暂停标志位 PDF 和看门狗溢出标志位 TO 清零。
功能表示	WDT cleared $TO \ \& \ PDF \leftarrow 0$
影响标志位	TO、PDF

CPL [m]	Complement Data Memory
指令说明	将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1。
功能表示	$[m] \leftarrow \overline{[m]}$
影响标志位	Z
CPLA [m]	Complement Data Memory with result in ACC
指令说明	将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1，而结果被储存回累加器且数据存储器中的内容不变。
功能表示	$ACC \leftarrow \overline{[m]}$
影响标志位	Z
DAA [m]	Decimal-Adjust ACC for addition with result in Data Memory
指令说明	将累加器中的内容转换为 BCD (二进制转成十进制) 码。如果低四位的值大于“9”或 AC=1，那么 BCD 调整就执行对原值加“6”，否则原值保持不变；如果高四位的值大于“9”或 C=1，那么 BCD 调整就执行对原值加“6”。BCD 转换实质上是根据累加器和标志位执行 00H, 06H, 60H 或 66H 的加法运算，结果存放到数据存储器。只有进位标志位 C 受影响，用来指示原始 BCD 的和是否大于 100，并可以进行双精度十进制数的加法运算。
功能表示	$[m] \leftarrow ACC + 00H$ 或 $[m] \leftarrow ACC + 06H$ 或 $[m] \leftarrow ACC + 60H$ 或 $[m] \leftarrow ACC + 66H$
影响标志位	C
DEC [m]	Decrement Data Memory
指令说明	将指定数据存储器内容减 1。
功能表示	$[m] \leftarrow [m] - 1$
影响标志位	Z
DECA [m]	Decrement Data Memory with result in ACC
指令说明	将指定数据存储器的内容减 1，把结果存放回累加器并保持指定数据存储器的内容不变。
功能表示	$ACC \leftarrow [m] - 1$
影响标志位	Z

HALT	Enter power down mode
指令说明	此指令终止程序执行并关掉系统时钟，RAM 和寄存器的内容保持原状态，WDT 计数器和分频器被清“0”，暂停标志位 PDF 被置位 1，WDT 溢出标志位 TO 被清 0。
功能表示	$TO \leftarrow 0$ $PDF \leftarrow 1$
影响标志位	TO、PDF
INC [m]	Increment Data Memory
指令说明	将指定数据存储器的内容加 1。
功能表示	$[m] \leftarrow [m] + 1$
影响标志位	Z
INCA [m]	Increment Data Memory with result in ACC
指令说明	将指定数据存储器的内容加 1，结果存放回累加器并保持指定的数据存储器内容不变。
功能表示	$ACC \leftarrow [m] + 1$
影响标志位	Z
JMP addr	Jump unconditionally
指令说明	程序计数器的内容无条件地由被指定的地址取代，程序由新的地址继续执行。当新的地址被加载时，必须插入一个空指令周期，所以此指令为 2 个周期的指令。
功能表示	$Program\ Counter \leftarrow addr$
影响标志位	无
MOV A, [m]	Move Data Memory to ACC
指令说明	将指定数据存储器的内容复制到累加器。
功能表示	$ACC \leftarrow [m]$
影响标志位	无
MOV A, x	Move immediate data to ACC
指令说明	将 8 位立即数载入累加器。
功能表示	$ACC \leftarrow x$
影响标志位	无
MOV [m], A	Move ACC to Data Memory
指令说明	将累加器的内容复制到指定的数据存储器。
功能表示	$[m] \leftarrow ACC$
影响标志位	无

NOP	No operation
指令说明	空操作，接下来顺序执行下一条指令。
功能表示	无操作
影响标志位	无
ORA, [m]	Logical OR Data Memory to ACC
指令说明	将累加器中的数据和指定的数据存储器内容逻辑或，结果存放到累加器。
功能表示	$ACC \leftarrow ACC \text{ "OR" } [m]$
影响标志位	Z
ORA, x	Logical OR immediate data to ACC
指令说明	将累加器中的数据和立即数逻辑或，结果存放到累加器。
功能表示	$ACC \leftarrow ACC \text{ "OR" } x$
影响标志位	Z
ORM A, [m]	Logical OR ACC to Data Memory
指令说明	将存在指定数据存储器中的数据和累加器逻辑或，结果放到数据存储器。
功能表示	$[m] \leftarrow ACC \text{ "OR" } [m]$
影响标志位	Z
RET	Return from subroutine
指令说明	将堆栈寄存器中的程序计数器值恢复，程序由取回的地址继续执行。
功能表示	Program Counter ← Stack
影响标志位	无
RET A, x	Return from subroutine and load immediate data to ACC
指令说明	将堆栈寄存器中的程序计数器值恢复且累加器载入指定的立即数，程序由取回的地址继续执行。
功能表示	Program Counter ← Stack $ACC \leftarrow x$
影响标志位	无

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

RR [m]	Rotate Data Memory right
指令说明	将指定数据存储器的内容循环右移 1 位且第 0 位移到第 7 位。
功能表示	$[m].i \leftarrow [m].(i+1) \ (i=0\sim6)$ $[m].7 \leftarrow [m].0$
影响标志位	无
RRA [m]	Rotate Data Memory right with result in ACC
指令说明	将指定数据存储器的内容循环右移 1 位，第 0 位移到第 7 位，移位结果存放到累加器，而指定数据存储器的内容保持不变。
功能表示	$ACC.i \leftarrow [m].(i+1) \ (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
影响标志位	无
RRC [m]	Rotate Data Memory right through Carry
指令说明	将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位。
功能表示	$[m].i \leftarrow [m].(i+1) \ (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位	C
RRCA [m]	Rotate Data Memory right through Carry with result in ACC
指令说明	将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位，移位结果送回累加器，但是指定数据寄存器的内容保持不变。
功能表示	$ACC.i \leftarrow [m].(i+1) \ (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位	C
SBC A, [m]	Subtract Data Memory from ACC with Carry
指令说明	将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$ACC \leftarrow ACC - [m] - \bar{C}$
影响标志位	OV、Z、AC、C

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

SIZ [m] 指令说明	Skip if increment Data Memory is 0 将指定的数据存储器的内容加 1，判断是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 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 将指定数据存储器的内容加 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果会被存放到累加器，但是指定数据存储器的内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	$ACC \leftarrow [m] + 1$ ，如果 $ACC=0$ 跳过下一条指令执行
影响标志位	无
SNZ [m].i 指令说明	Skip if bit i of Data Memory is not 0 判断指定数据存储器的第 i 位，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果为 0，则程序继续执行下一条指令。
功能表示	如果 $[m].i \neq 0$ ，跳过下一条指令执行
影响标志位	无
SUB A, [m] 指令说明	Subtract Data Memory from ACC 将累加器的内容减去指定的数据存储器的数据，把结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$ACC \leftarrow ACC - [m]$
影响标志位	OV、Z、AC、C
SUBM A, [m] 指令说明	Subtract Data Memory from ACC with result in Data Memory 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

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
SWAP [m]	Swap nibbles of Data Memory
指令说明	将指定数据存储器的低 4 位和高 4 位互相交换。
功能表示	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
影响标志位	无
SWAPA [m]	Swap nibbles of Data Memory with result in ACC
指令说明	将指定数据存储器的低 4 位与高 4 位互相交换，再将结果存放到累加器且指定数据寄存器的数据保持不变。
功能表示	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
影响标志位	无
SZ [m]	Skip if Data Memory is 0
指令说明	指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定数据存储器的内容是否为 0，若为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	如果 $[m]=0$ ，跳过下一条指令执行
影响标志位	无
SZA [m]	Skip if Data Memory is 0 with data movement to ACC
指令说明	将指定数据存储器内容复制到累加器，并判断指定数据存储器的内容是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	$ACC \leftarrow [m]$ ，如果 $[m]=0$ ，跳过下一条指令执行
影响标志位	无

SZ [m].i	Skip if bit i of Data Memory is 0
指令说明	判断指定数据存储器的第 i 位是否为 0，若为 0，则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	如果 [m].i=0，跳过下一条指令执行
影响标志位	无
TABRD [m]	Read table (specific page or current page) to TBLH and Data Memory
指令说明	将表格指针 (TBHP 和 TBLP，若无 TBHP 则仅 TBLP) 所指的程序代码低字节移至指定数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
TABRDL [m]	Read table (last page) to TBLH and Data Memory
指令说明	将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
XOR A, [m]	Logical XOR Data Memory to ACC
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。
功能表示	ACC ← ACC “XOR” [m]
影响标志位	Z
XORM A, [m]	Logical XOR ACC to Data Memory
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果放到数据存储器。
功能表示	[m] ← ACC “XOR” [m]
影响标志位	Z
XOR A, x	Logical XOR immediate data to ACC
指令说明	将累加器的数据与立即数逻辑异或，结果存放到累加器。
功能表示	ACC ← ACC “XOR” x
影响标志位	Z

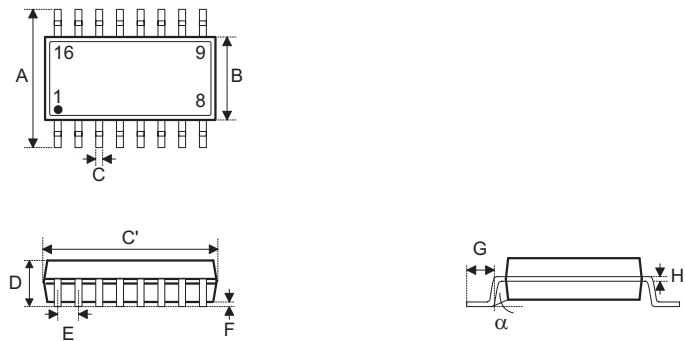
封装信息

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

封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

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

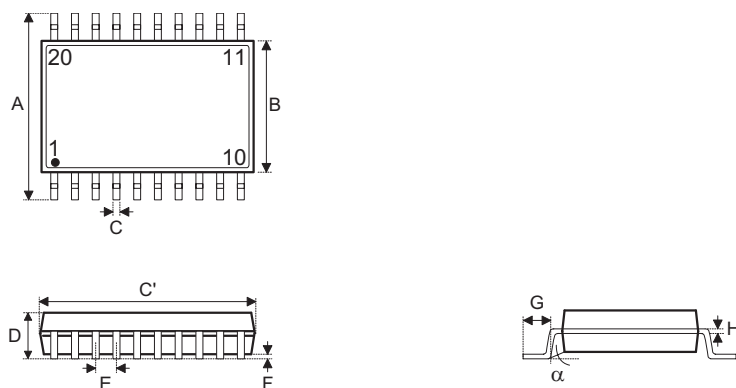
16-pin NSOP (150mil) 外形尺寸



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 BSC	—
B	—	3.9 BSC	—
C	0.31	—	0.51
C'	—	9.9 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°

20-pin SOP (300mil) 外形尺寸



符号	尺寸 (单位: inch)		
	最小值	典型值	最大值
A	—	0.406 BSC	—
B	—	0.295 BSC	—
C	0.012	—	0.020
C'	—	0.504 BSC	—
D	—	—	0.104
E	—	0.050 BSC	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

符号	尺寸 (单位: mm)		
	最小值	典型值	最大值
A	—	10.30 BSC	—
B	—	7.50 BSC	—
C	0.31	—	0.51
C'	—	12.80 BSC	—
D	—	—	2.65
E	—	1.27 BSC	—
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

Copyright© 2021 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** 产品不授权使用于救生、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/>.