



高资源 A/D 型 Flash 单片机

HT66F2350/HT66F2360

HT66F2370/HT66F2390

版本：V1.80 日期：2023-02-23

www.holtek.com

目 录

特性	7
CPU 特性	7
周边特性	7
概述	8
选型表	8
方框图	9
引脚图	9
引脚描述	11
极限参数	27
直流电气特性	27
工作电压特性	27
待机电流特性	28
工作电流特性	29
交流电气特性	30
内部高速振荡器 – HIRC – 频率精确度	30
内部低速振荡器电气特性 – LIRC	31
低速晶体振荡器电气特性 – LXT	31
工作频率电气特性曲线图	31
系统上电时间电气特性	32
输入 / 输出口电气特性	33
存储器电气特性	34
A/D 转换器电气特性	34
比较器电气特性	35
软件控制 LCD 驱动器电气特性	36
I ² C 电气特性	36
LVD/LVR 电气特性	37
上电复位特性	37
系统结构	38
时序和流水线结构	38
程序计数器	39
堆栈	39
算术逻辑单元 – ALU	40
Flash 程序存储器	41
结构	41
特殊向量	42
查表	42
查表范例	42
在线烧录 – ICP	43
片上调试 – OCDS	44
在线应用编程 – IAP	45
数据存储器	54
结构	54

数据存储器寻址	55
通用数据存储器	55
特殊功能数据存储器	55
特殊功能寄存器	59
间接寻址寄存器 – IAR0, IAR1, IAR2	59
存储器指针 – MP0, MP1L/MP1H, MP2L/MP2H	59
程序存储区指针 – PBP	61
累加器 – ACC	62
程序计数器低字节寄存器 – PCL	62
表格寄存器 – TBLP, TBHP, TBLH	62
状态寄存器 – STATUS	62
EEPROM 数据存储器	64
EEPROM 数据存储器结构	64
EEPROM 寄存器	64
从 EEPROM 中读取数据	66
写数据到 EEPROM	66
写保护	66
EEPROM 中断	66
编程注意事项	67
振荡器	68
振荡器概述	68
系统时钟配置	68
外部高频晶体振荡器 – HXT	69
内部 RC 振荡器 – HIRC	69
外部 32.768kHz 晶体振荡器 – LXT	70
内部 32kHz 振荡器 – LIRC	70
工作模式和系统时钟	71
系统时钟	71
系统工作模式	72
控制寄存器	73
工作模式转换	75
待机电流注意事项	79
唤醒	79
看门狗定时器	80
看门狗定时器时钟源	80
看门狗定时器控制寄存器	80
看门狗定时器操作	81
复位和初始化	82
复位功能	82
复位初始状态	86
输入 / 输出端口	93
上拉电阻	94
PA 口唤醒	95
输入 / 输出端口控制寄存器	95
输入 / 输出端口源电流选择	96
输入 / 输出端口电源控制	99
引脚共用功能	99
输入 / 输出引脚结构	110
读端口功能	110

编程注意事项	111
定时器模块 – TM	112
简介	112
TM 操作	112
TM 时钟源	112
TM 中断	112
TM 外部引脚	113
TM 输入 / 输出引脚选择	113
编程注意事项	114
标准型 TM – STM	115
标准型 TM 操作	115
标准型 TM 寄存器介绍	116
标准型 TM 工作模式	120
周期型 TM – PTM	130
周期型 TM 操作	131
周期型 TM 寄存器描述	131
周期型 TM 工作模式	136
A/D 转换器	145
A/D 转换器简介	145
A/D 转换器寄存器介绍	146
A/D 转换器参考电压	150
A/D 转换器输入信号	150
A/D 转换器操作	151
转换率和时序图	152
A/D 转换步骤	153
编程注意事项	154
A/D 转换功能	154
A/D 转换程序范例	155
串行接口模块 – SIM	157
SPI 接口	157
I ² C 接口	163
SPIA 串行接口模块 – SPIA	172
SPIA 接口操作	172
SPIA 寄存器	173
SPIA 通信	175
SPIA 总线使能 / 除能	177
SPIA 操作	178
错误侦测	179
UART 接口	180
UART 外部引脚	181
UART 数据传输方案	181
UART 状态和控制寄存器	181
波特率发生器	185
UART 模块的设置与控制	186
UART 发送器	187
UART 接收器	188
接收错误处理	189
UART 模块中断结构	190
UART 模块暂停和唤醒	191

比较器	192
比较器操作	192
输入失调校准	194
比较器中断	194
编程注意事项	194
软件控制 LCD 驱动	195
LCD 操作	195
LCD 偏压电流控制	195
16 位乘法单元 – MDU	196
MDU 寄存器	196
乘法单元操作	197
循环冗余校验 – CRC	199
CRC 寄存器	199
CRC 操作	200
CRC 计算	200
低电压检测 – LVD	202
LVD 寄存器	202
LVD 操作	203
中断	204
中断寄存器	204
中断操作	212
外部中断	215
比较器中断	215
多功能中断	215
A/D 转换器中断	215
时基中断	216
TM 中断	218
LVD 中断	218
EEPROM 中断	218
串行接口模块中断	218
SPIA 接口中断	218
UART 接口中断	219
中断唤醒功能	219
编程注意事项	219
应用电路	220
指令集	221
简介	221
指令周期	221
数据的传送	221
算术运算	221
逻辑和移位运算	221
分支和控制转换	222
位运算	222
查表运算	222
其它运算	222
指令集概要	223
惯例	223
扩展指令集	226

指令定义	228
扩展指令定义	240
封装信息	250
48-pin LQFP (7mm × 7mm) 外形尺寸	251
64-pin LQFP (7mm × 7mm) 外形尺寸	252

特性

CPU 特性

- 工作电压：
 - ◆ $f_{\text{SYS}}=8\text{MHz}$: 2.2 ~ 5.5V
 - ◆ $f_{\text{SYS}}=12\text{MHz}$: 2.7V ~ 5.5V
 - ◆ $f_{\text{SYS}}=16\text{MHz}$: 3.3V ~ 5.5V
- $V_{\text{DD}}=5\text{V}$, 系统时钟为 16MHz 时, 指令周期为 0.25 μs
- 提供暂停和唤醒功能, 以降低功耗
- 振荡器类型:
 - ◆ 外部高速晶振 – HXT
 - ◆ 内部高速 RC – HIRC
 - ◆ 外部 32.768kHz 晶振 – LXT
 - ◆ 内部 32kHz RC – LIRC
- 完全集成内部 8/12/16MHz 振荡器, 无需外接元器件
- 多种工作模式: 快速、低速、空闲和休眠
- 所有指令都可在 1~3 个指令周期内完成
- 查表指令
- 115 条功能强大的指令系统
- 16 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器: 多达 64K \times 16
- RAM 数据存储器: 多达 4096 \times 8
- True EEPROM 存储器: 多达 1024 \times 8
- 看门狗定时器功能
- 多达 58 个双向 I/O 口
- 可编程 I/O 源电流
- 软件控制的 1/2 bias LCD 驱动功能, 具有 4 个 SCOM 线
- 四个引脚与外部中断口共用
- 多个定时器模块用于时间测量、捕捉输入、比较匹配输出、PWM 输出及单脉冲输出功能
- 串行接口模块 – SIM, 用于 SPI 或 I²C 通信
- 串行外设接口 – SPIA
- 全双工通用异步收发器接口 – UART 接口
- 双时基功能, 可提供固定时间的中断信号
- 两个比较器
- 多达 16 个外部通道 12-bit A/D 转换器
- 内建乘除法单元 – MDU
- 内建 16-bit 循环冗余校验功能 – CRC
- 低电压复位功能
- 低电压检测功能
- 通过欧洲标准 IEC 60730 与美国 UL 60730 认证
- 多种封装类型

概述

该系列单片机是一款 8 位具有高性能精简指令集的 A/D Flash 型单片机。该系列单片机具有一系列功能和特性，其 Flash 存储器可多次编程的特性给客户提供了较大的方便。除了 Flash 程序存储器，还包括 RAM 数据存储器 and 用于存储序列数据、校准数据等非易失性数据的 True EEPROM 存储器。

在模拟特性方面，该单片机包含一个多通道 12-bit A/D 转换器和两个比较器。其具有多个使用灵活的定时器模块，可提供定时功能、脉冲产生功能及 PWM 产生功能。内建完整的 SPI、UART 和 I²C 接口功能，为设计者提供了一个易与外部硬件通信的接口。内部看门狗定时器、低电压复位和低电压检测等内部保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地运行。

该系列单片机提供了 HXT, LXT, HIRC 和 LIRC 振荡器功能选项，且内建完整的系统振荡器，无需外接元器件。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。

包含 I/O 使用灵活、时基功能和其它特性确保了该系列单片机可以广泛应用于各种产品中，例如电子测量仪器、环境监测、手持式测量工具、家庭应用、电子控制工具、马达驱动等多方面。

选型表

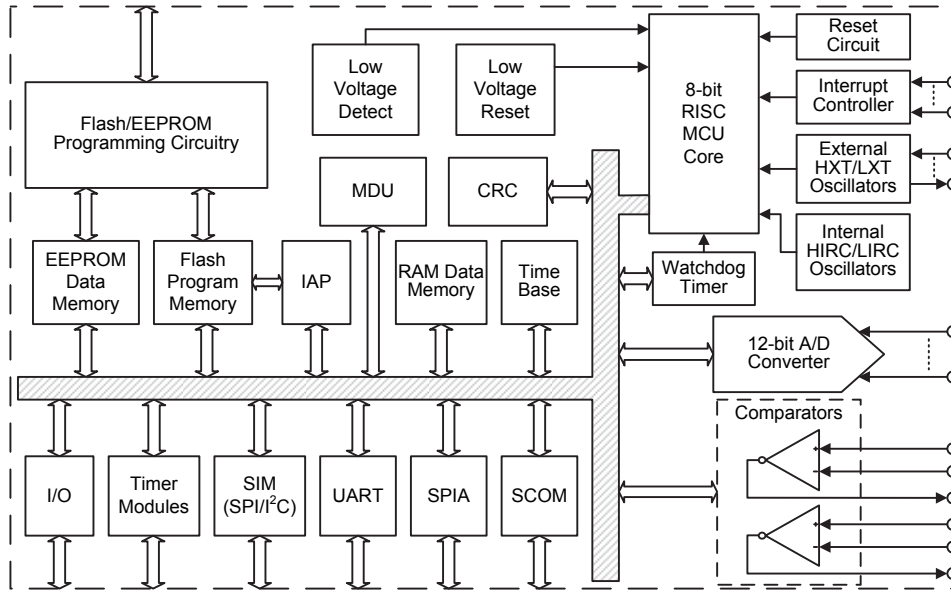
对此系列的单片机而言，大多数的特性参数都是一样的。主要差异在于存储器的容量，I/O 引脚数目和 A/D 转换器通道数量。下表列出了各单片机的主要特性。

单片机型号	ROM	RAM	EEPROM	I/O	外部中断	A/D	TM 模块
HT66F2350	8K×16	768×8	256×8	44	4	12-bit×12	10-bit PTM×2 16-bit PTM×2 16-bit STM×3
HT66F2360	16K×16	1536×8	256×8	58	4	12-bit×16	10-bit PTM×2 16-bit PTM×2 16-bit STM×3
HT66F2370	32K×16	3072×8	512×8	58	4	12-bit×16	10-bit PTM×2 16-bit PTM×2 16-bit STM×3
HT66F2390	64K×16	4096×8	1024×8	58	4	12-bit×16	10-bit PTM×2 16-bit PTM×2 16-bit STM×3

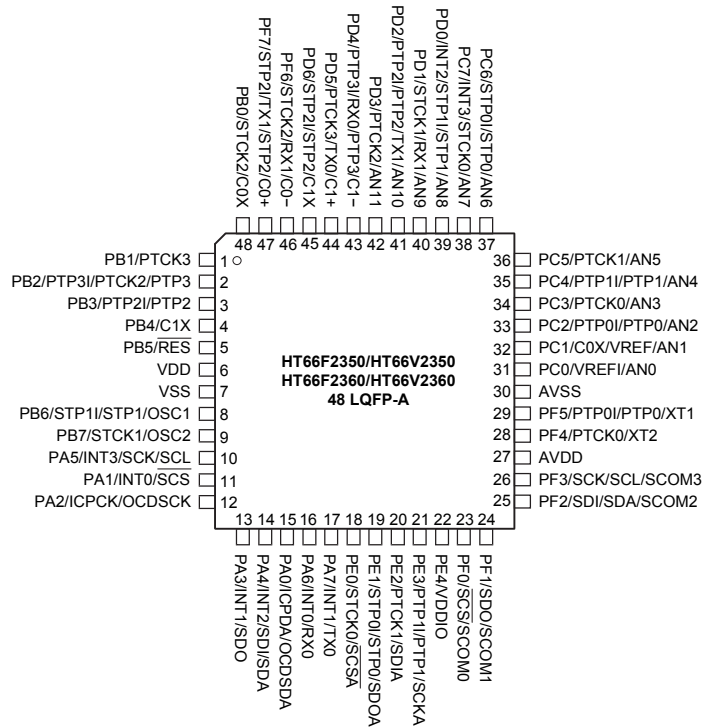
单片机型号	比较器	时基	SCOM	堆栈	SIM	SPIA	UART	MDU	CRC	封装
HT66F2350	2	2	4	16	√	√	2	√	√	48LQFP
HT66F2360	2	2	4	16	√	√	2	√	√	48/64LQFP
HT66F2370	2	2	4	16	√	√	3	√	√	48/64LQFP
HT66F2390	2	2	4	16	√	√	3	√	√	48/64LQFP

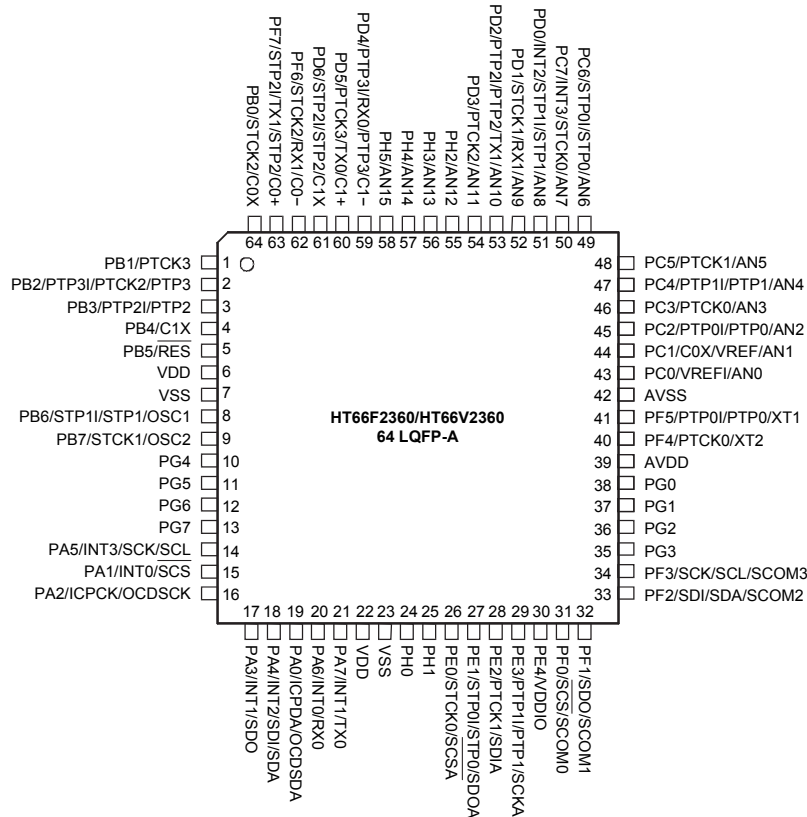
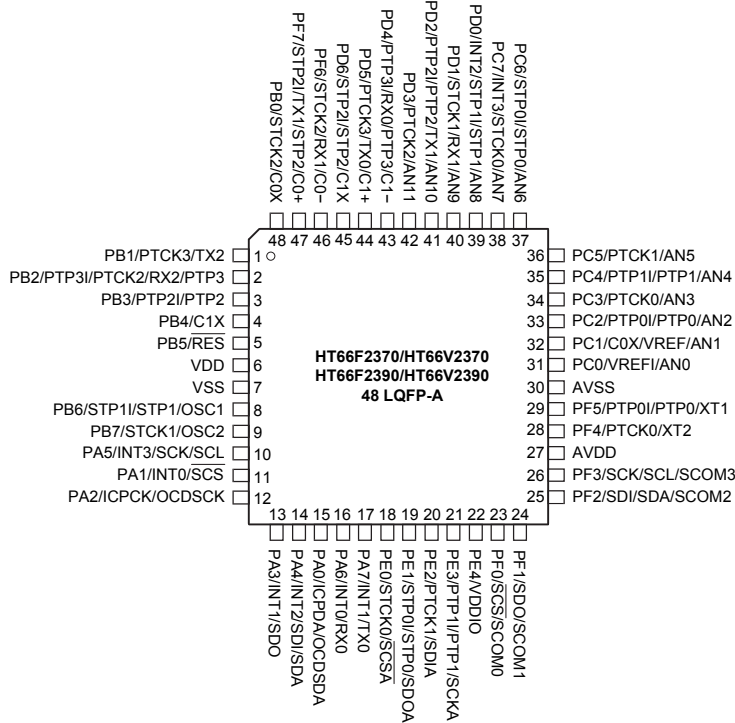
注：对于有不止一种封装形式的单片机，选型表针对较大封装的情况。

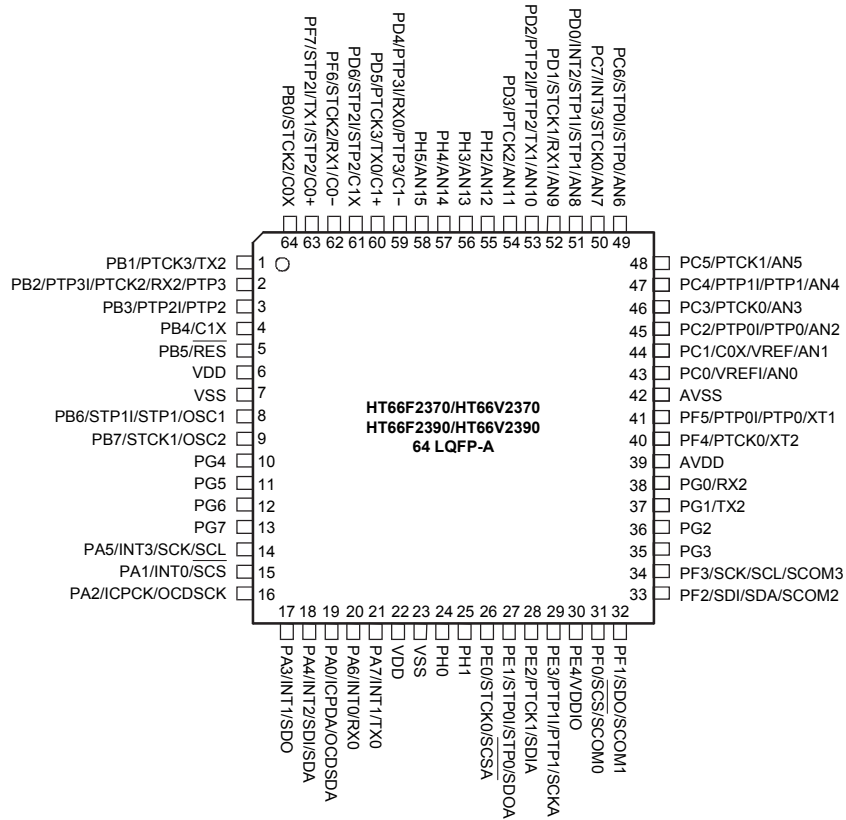
方框图



引脚图







注：OCSDA 和 OCDSCK 是 OCDS 专用引脚，仅存在于 HT66V23x0 中。HT66V23x0 是 HT66F23x0 的 OCDS EV 芯片。

引脚描述

除了电源引脚外，该系列单片机的所有引脚都以它们的端口名称进行标注，例如 PA0、PA1 等，用于描述这些引脚的数字输入 / 输出功能。然而，这些引脚也与其它功能共用，如 A/D 转换器，定时器模块引脚等。每个引脚的功能如下表所述，而引脚配置的详细内容见规格书其它章节。

HT66F2350/HT66F2360

引脚名称	功能	OPT	I/T	O/T	说明
PA0/ICPDA/ OCSDA	PA0	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	ICPDA	—	ST	CMOS	ICP 数据 / 地址引脚
	OCSDA	—	ST	CMOS	OCDS 地址 / 数据引脚，仅适用于 EV 芯片。

引脚名称	功能	OPT	I/T	O/T	说明
PA1/INT0/ $\overline{\text{SCS}}$	PA1	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	INT0	PAS0 INTEG INTC0 IFS2	ST	—	外部中断 0
	$\overline{\text{SCS}}$	PAS0 IFS2	ST	CMOS	SPI 从机选择
PA2/ICPCK/ OCDSCK	PA2	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	ICPCK	—	ST	CMOS	ICP 时钟引脚
	OCDSCK	—	ST	—	OCDS 时钟引脚，仅适用于 EV 芯片。
PA3/INT1/SDO	PA3	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	INT1	PAS0 INTEG INTC0 IFS2	ST	—	外部中断 1
	SDO	PAS0	—	CMOS	SPI 数据输出
PA4/INT2/SDI/ SDA	PA4	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	INT2	PAS1 INTEG INTC3 IFS2	ST	—	外部中断 2
	SDI	PAS1 IFS2	ST	—	SPI 数据输入
	SDA	PAS1 IFS2	ST	NMOS	I ² C 数据线
PA5/INT3/ SCK/SCL	PA5	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	INT3	PAS1 INTEG INTC3 IFS2	ST	—	外部中断 3
	SCK	PAS1 IFS2	ST	CMOS	SPI 串行时钟
	SCL	PAS1 IFS2	ST	NMOS	I ² C 时钟线

引脚名称	功能	OPT	I/T	O/T	说明
PA6/INT0/RX0	PA6	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	INT0	PAS1 INTEG INTC0 IFS2	ST	—	外部中断 0
	RX0	PAS1 IFS3	ST	—	UART0 RX 串行数据输入
PA7/INT1/TX0	PA7	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	INT1	PAS1 INTEG INTC0 IFS2	ST	—	外部中断 1
	TX0	PAS1	—	CMOS	UART0 TX 串行数据输出
PB0/STCK2/ C0X	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STCK2	PBS0 IFS0	ST	—	STM2 时钟输入
	C0X	PBS0	—	CMOS	比较器 0 输出
PB1/PTCK3	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK3	PBS0 IFS0	ST	—	PTM3 时钟输入
PB2/PTP3I/ PTCK2/PTP3	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP3I	PBS0 IFS1	ST	—	PTM3 捕捉输入
	PTCK2	PBS0 IFS0	ST	—	PTM2 时钟输入
	PTP3	PBS0	—	CMOS	PTM3 输出
PB3/PTP2I/ PTP2	PB3	PBPU PBS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP2I	PBS0 IFS1	ST	—	PTM2 捕捉输入
	PTP2	PBS0	—	CMOS	PTM2 输出
PB4/C1X	PB4	PBPU PBS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	C1X	PBS1	—	CMOS	比较器 1 输出
PB5/ $\overline{\text{RES}}$	PB5	PBPU PBS1 RSTC	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	$\overline{\text{RES}}$	RSTC	ST	—	外部复位引脚

引脚名称	功能	OPT	I/T	O/T	说明
PB6/STP1I/ STP1/OSC1	PB6	PBPU PBS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STP1I	PBS1 IFS1	ST	—	STM1 捕捉输入
	STP1	PBS1	—	CMOS	STM1 输出
	OSC1	PBS1	HXT	—	HXT 振荡器引脚
PB7/STCK1/ OSC2	PB7	PBPU PBS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STCK1	PBS1 IFS0	ST	—	STM1 时钟输入
	OSC2	PBS1	—	HXT	HXT 振荡器引脚
PC0/VREFI/ AN0	PC0	PCPU PCS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	VREFI	PCS0	AN	—	A/D 转换器参考电压输入
	AN0	PCS0	AN	—	A/D 转换器模拟输入
PC1/C0X/ VREF/AN1	PC1	PCPU PCS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	C0X	PCS0	—	CMOS	比较器 0 输出
	VREF	PCS0	AN	—	A/D 转换器参考电压输入
	AN1	PCS0	AN	—	A/D 转换器模拟输入
PC2/PTP0I/ PTP0/AN2	PC2	PCPU PCS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP0I	PCS0 IFS1	ST	—	PTM0 捕捉输入
	PTP0	PCS0	—	CMOS	PTM0 输出
	AN2	PCS0	AN	—	A/D 转换器模拟输入
PC3/PTCK0/ AN3	PC3	PCPU PCS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK0	PCS0 IFS0	ST	—	PTM0 时钟输入
	AN3	PCS0	AN	—	A/D 转换器模拟输入
PC4/PTP1I/ PTP1/AN4	PC4	PCPU PCS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP1I	PCS1 IFS1	ST	—	PTM1 捕捉输入
	PTP1	PCS1	—	CMOS	PTM1 输出
	AN4	PCS1	AN	—	A/D 转换器模拟输入
PC5/PTCK1/ AN5	PC5	PCPU PCS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK1	PCS1 IFS0	ST	—	PTM1 时钟输入
	AN5	PCS1	AN	—	A/D 转换器模拟输入

引脚名称	功能	OPT	I/T	O/T	说明
PC6/STP0I/ STP0/AN6	PC6	PCPU PCS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STP0I	PCS1 IFS1	ST	—	STM0 捕捉输入
	STP0	PCS1	—	CMOS	STM0 输出
	AN6	PCS1	AN	—	A/D 转换器模拟输入
PC7/INT3/ STCK0/AN7	PC7	PCPU PCS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	INT3	PCS1 INTEG INTC3 IFS2	ST	—	外部中断 3
	STCK0	PCS1 IFS0	ST	—	STM0 时钟输入
	AN7	PCS1	AN	—	A/D 转换器模拟输入
PD0/INT2/ STP1I/STP1/ AN8	PD0	PDP PDS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	INT2	PDS0 INTEG INTC3 IFS2	ST	—	外部中断 2
	STP1I	PDS0 IFS1	ST	—	STM1 捕捉输入
	STP1	PDS0	—	CMOS	STM1 输出
	AN8	PDS0	AN	—	A/D 转换器模拟输入
PD1/STCK1/ RX1/AN9	PD1	PDP PDS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STCK1	PDS0 IFS0	ST	—	STM1 时钟输入
	RX1	PDS0 IFS3	ST	—	UART1 RX 串行数据输入
	AN9	PDS0	AN	—	A/D 转换器模拟输入
PD2/PTP2I/ PTP2/TX1/ AN10	PD2	PDP PDS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP2I	PDS0 IFS1	ST	—	PTM2 捕捉输入
	PTP2	PDS0	—	CMOS	PTM2 输出
	TX1	PDS0	—	CMOS	UART1 TX 串行数据输出
	AN10	PDS0	AN	—	A/D 转换器模拟输入
PD3/PTCK2/ AN11	PD3	PDP PDS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK2	PDS0 IFS0	ST	—	PTM2 时钟输入
	AN11	PDS0	AN	—	A/D 转换器模拟输入

引脚名称	功能	OPT	I/T	O/T	说明
PD4/PTP3I/ PTP3/RX0/C1-	PD4	PDPU PDS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP3I	PDS1 IFS1	ST	—	PTM3 捕捉输入
	PTP3	PDS1	—	CMOS	PTM3 输出
	RX0	PDS1 IFS3	ST	—	UART0 RX 串行数据输入
	C1-	PDS1	AN	—	比较器 1 负极输入
PD5/PTCK3/ TX0/C1+	PD5	PDPU PDS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK3	PDS1 IFS0	ST	—	PTM3 时钟输入
	TX0	PDS1	—	CMOS	UART0 TX 串行数据输出
	C1+	PDS1	AN	—	比较器 1 正极输入
PD6/STP2I/ STP2/C1X	PD6	PDPU PDS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STP2I	PDS1 IFS1	ST	—	STM2 捕捉输入
	STP2	PDS1	—	CMOS	STM2 输出
	C1X	PDS1	—	CMOS	比较器 1 输出
PE0/STCK0/ SCSA	PE0	PEPU PES0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STCK0	PES0 IFS0	ST	—	STM0 时钟输入
	SCSA	PES0	ST	CMOS	SPIA 从机选择
PE1/STP0I/ STP0/SDOA	PE1	PEPU PES0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STP0I	PES0 IFS	ST	—	STM0 捕捉输入
	STP0	PES0	—	CMOS	STM0 反相输出
	SDOA	PES0	—	CMOS	SPIA 数据输出
PE2/PTCK1/ SDIA	PE2	PEPU PES0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK1	PES0 IFS0	ST	—	PTM1 时钟输入
	SDIA	PES0	ST	—	SPIA 数据输入
PE3/PTP1I/ PTP1/ SCKA	PE3	PEPU PES0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP1I	PES0 IFS1	ST	—	PTM1 捕捉输入
	PTP1	PES0	—	CMOS	PTM1 输出
	SCKA	PES0	ST	CMOS	SPIA 串行时钟

引脚名称	功能	OPT	I/T	O/T	说明
PE4/VDDIO	PE4	PEPU PES1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	VDDIO	PES1 PMPS	PWR	—	PE0~PE3 引脚电源选择
PF0/ $\overline{\text{SCS}}$ / SCOM0	PF0	PFPU PFS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	$\overline{\text{SCS}}$	PFS0 IFS2	ST	CMOS	SPI 从机选择
	SCOM0	PFS0	—	CMOS	软件 LCD COM 输出
PF1/SDO/ SCOM1	PF1	PFPU PFS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	SDO	PFS0	—	CMOS	SPI 数据输出
	SCOM1	PFS0	—	CMOS	软件 LCD COM 输出
PF2/SDI/SDA/ SCOM2	PF2	PFPU PFS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	SDI	PFS0 IFS2	ST	—	SPI 数据输入
	SDA	PFS0 IFS2	ST	NMOS	I ² C 数据线
	SCOM2	PFS0	—	CMOS	软件 LCD COM 输出
PF3/SCK/SCL/ SCOM3	PF3	PFPU PFS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	SCK	PFS0 IFS2	ST	CMOS	SPI 串行时钟
	SCL	PFS0 IFS2	ST	NMOS	I ² C 时钟线
	SCOM3	PFS0	—	CMOS	软件 LCD COM 输出
PF4/PTCK0/ XT2	PF4	PFPU PFS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK0	PFS1 IFS0	ST	—	PTM0 时钟输入
	XT2	PFS1	—	LXT	LXT 振荡器引脚
PF5/PTP0I/ PTP0/XT1	PF5	PFPU PFS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP0I	PFS1 IFS1	ST	—	PTM0 捕捉输入
	PTP0	PFS1	—	CMOS	PTM0 输出
	XT1	PFS1	LXT	—	LXT 振荡器引脚

引脚名称	功能	OPT	I/T	O/T	说明
PF6/STCK2/ RX1/C0-	PF6	PFP PFS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STCK2	PFS1 IFS0	ST	—	STM2 时钟输入
	RX1	PFS1 IFS3	ST	—	UART1 RX 串行数据输入
	C0-	PFS1	AN	—	比较器 0 负极输入
PF7/STP2I/ STP2/TX1/C0+	PF7	PFP PFS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STP2I	PFS1 IFS1	ST	—	STM2 捕捉输入
	STP2	PFS1	—	CMOS	STM2 输出
	TX1	PFS1	—	CMOS	UART1 TX 串行数据输出
C0+	PDS1	AN	—	比较器 0 正极输入	
PG0~PG3	PGn	PGPU PGS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
PG4~PG7	PGn	PGPU PGS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
PH0~PH1	PHn	PHPU PHS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
PH2/AN12	PH2	PHPU PHS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	AN12	PHS0	AN	—	A/D 转换器模拟输入
PH3/AN13	PH3	PHPU PHS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	AN13	PHS0	AN	—	A/D 转换器模拟输入
PH4/AN14	PH4	PHPU PHS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	AN14	PHS1	AN	—	A/D 转换器模拟输入
PH5/AN15	PH5	PHPU PHS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	AN15	PHS1	AN	—	A/D 转换器模拟输入
VDD	VDD	—	PWR	—	正电源
VSS	VSS	—	PWR	—	负电源，接地
AVDD	AVDD	—	PWR	—	模拟正电源
AVSS	AVSS	—	PWR	—	模拟负电源，接地

注：I/T：输入类型；

OPT：通过配置寄存器选项来配置；

NMOS：NMOS 输出；

AN：模拟信号；

HXT：高频晶体振荡器

注：端口 G 和端口 H 仅存在于 HT66F2360 单片机中。

O/T：输出类型；

CMOS：CMOS 输出；

ST：施密特触发输入；

PWR：电源；

LXT：低频晶体振荡器

HT66F2370/HT66F2390

引脚名称	功能	OPT	I/T	O/T	说明
PA0/ICPDA/ OCSDA	PA0	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	ICPDA	—	ST	CMOS	ICP 数据 / 地址引脚
	OCSDA	—	ST	CMOS	OCDS 地址 / 数据引脚, 仅适用于 EV 芯片。
PA1/INT0/ $\overline{\text{SCS}}$	PA1	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	INT0	PAS0 INTEG INTC0 IFS2	ST	—	外部中断 0
	$\overline{\text{SCS}}$	PAS0 IFS2	ST	CMOS	SPI 从机选择
PA2/ICPCK/ OCDSCK	PA2	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	ICPCK	—	ST	CMOS	ICP 时钟引脚
	OCDSCK	—	ST	—	OCDS 时钟引脚, 仅适用于 EV 芯片。
PA3/INT1/SDO	PA3	PAWU PAPU PAS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	INT1	PAS0 INTEG INTC0 IFS2	ST	—	外部中断 1
	SDO	PAS0	—	CMOS	SPI 数据输出
PA4/INT2/SDI/ SDA	PA4	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	INT2	PAS1 INTEG INTC3 IFS2	ST	—	外部中断 2
	SDI	PAS1 IFS2	ST	—	SPI 数据输入
	SDA	PAS1 IFS2	ST	NMOS	I ² C 数据线

引脚名称	功能	OPT	I/T	O/T	说明
PA5/INT3/SCK/ SCL	PA5	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	INT3	PAS1 INTEG INTC3 IFS2	ST	—	外部中断 3
	SCK	PAS1 IFS2	ST	CMOS	SPI 串行时钟
	SCL	PAS1 IFS2	ST	NMOS	I ² C 时钟线
PA6/INT0/RX0	PA6	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	INT0	PAS1 INTEG INTC0 IFS2	ST	—	外部中断 0
	RX0	PAS1 IFS3	ST	—	UART0 RX 串行数据输入
PA7/INT1/TX0	PA7	PAWU PAPU PAS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻和唤醒功能。
	INT1	PAS1 INTEG INTC0 IFS2	ST	—	外部中断 1
	TX0	PAS1	—	CMOS	UART0 TX 串行数据输出
PB0/STCK2/ C0X	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STCK2	PBS0 IFS0	ST	—	STM2 时钟输入
	C0X	PBS0	—	CMOS	比较器 0 输出
PB1/PTCK3/ TX2	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK3	PBS0 IFS0	ST	—	PTM3 时钟输入
	TX2	PBS0	—	CMOS	UART2 TX 串行数据输出

引脚名称	功能	OPT	I/T	O/T	说明
PB2/PTP3I/ PTCK2/PTP3/ RX2	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP3I	PBS0 IFS1	ST	—	PTM3 捕捉输入
	PTCK2	PBS0 IFS0	ST	—	PTM2 时钟输入
	PTP3	PBS0	—	CMOS	PTM3 输出
	RX2	PBS0 IFS3	ST	—	UART2 RX 串行数据输入
PB3/PTP2I/ PTP2	PB3	PBPU PBS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP2I	PBS0 IFS1	ST	—	PTM2 捕捉输入
	PTP2	PBS0	—	CMOS	PTM2 输出
PB4/C1X	PB4	PBPU PBS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	C1X	PBS1	—	CMOS	比较器 1 输出
PB5/ $\overline{\text{RES}}$	PB5	PBPU PBS1 RSTC	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	$\overline{\text{RES}}$	RSTC	ST	—	外部复位引脚
PB6/STP1I/ STP1/OSC1	PB6	PBPU PBS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STP1I	PBS1 IFS1	ST	—	STM1 捕捉输入
	STP1	PBS1	—	CMOS	STM1 输出
	OSC1	PBS1	HXT	—	HXT 振荡器引脚
PB7/STCK1/ OSC2	PB7	PBPU PBS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STCK1	PBS1 IFS0	ST	—	STM1 时钟输入
	OSC2	PBS1	—	HXT	HXT 振荡器引脚
PC0/VREFI/ AN0	PC0	PCPU PCS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	VREFI	PCS0	AN	—	A/D 转换器参考电压输入
	AN0	PCS0	AN	—	A/D 转换器模拟输入
PC1/C0X/ VREF/AN1	PC1	PCPU PCS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	C0X	PCS0	—	CMOS	比较器 0 输出
	VREF	PCS0	AN	—	A/D 转换器参考电压输入
	AN1	PCS0	AN	—	A/D 转换器模拟输入

引脚名称	功能	OPT	I/T	O/T	说明
PC2/PTP0I/ PTP0/AN2	PC2	PCPU PCS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP0I	PCS0 IFS1	ST	—	PTM0 捕捉输入
	PTP0	PCS0	—	CMOS	PTM0 输出
	AN2	PCS0	AN	—	A/D 转换器模拟输入
PC3/PTCK0/ AN3	PC3	PCPU PCS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK0	PCS0 IFS0	ST	—	PTM0 时钟输入
	AN3	PCS0	AN	—	A/D 转换器模拟输入
PC4/PTP1I/ PTP1/AN4	PC4	PCPU PCS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP1I	PCS1 IFS1	ST	—	PTM1 捕捉输入
	PTP1	PCS1	—	CMOS	PTM1 输出
	AN4	PCS1	AN	—	A/D 转换器模拟输入
PC5/PTCK1/ AN5	PC5	PCPU PCS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK1	PCS1 IFS0	ST	—	PTM1 时钟输入
	AN5	PCS1	AN	—	A/D 转换器模拟输入
PC6/STP0I/ STP0/AN6	PC6	PCPU PCS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STP0I	PCS1 IFS1	ST	—	STM0 捕捉输入
	STP0	PCS1	—	CMOS	STM0 输出
	AN6	PCS1	AN	—	A/D 转换器模拟输入
PC7/INT3/ STCK0/AN7	PC7	PCPU PCS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	INT3	PCS1 INTEG INTC3 IFS2	ST	—	外部中断 3
	STCK0	PCS1 IFS0	ST	—	STM0 时钟输入
	AN7	PCS1	AN	—	A/D 转换器模拟输入

引脚名称	功能	OPT	I/T	O/T	说明
PD0/INT2/ STP1I/STP1/ AN8	PD0	PDP PDS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	INT2	PDS0 INTEG INTC3 IFS2	ST	—	外部中断 2
	STP1I	PDS0 IFS1	ST	—	STM1 捕捉输入
	STP1	PDS0	—	CMOS	STM1 输出
	AN8	PDS0	AN	—	A/D 转换器模拟输入
PD1/STCK1/ RX1/AN9	PD1	PDP PDS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STCK1	PDS0 IFS0	ST	—	STM1 时钟输入
	RX1	PDS0 IFS3	ST	—	UART1 RX 串行数据输入
	AN9	PDS0	AN	—	A/D 转换器模拟输入
PD2/PTP2I/ PTP2/TX1/ AN10	PD2	PDP PDS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP2I	PDS0 IFS1	ST	—	PTM2 捕捉输入
	PTP2	PDS0	—	CMOS	PTM2 输出
	TX1	PDS0	—	CMOS	UART1 TX 串行数据输出
	AN10	PDS0	AN	—	A/D 转换器模拟输入
PD3/PTCK2/ AN11	PD3	PDP PDS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK2	PDS0 IFS0	ST	—	PTM2 时钟输入
	AN11	PDS0	AN	—	A/D 转换器模拟输入
PD4/PTP3I/ PTP3/RX0/C1-	PD4	PDP PDS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP3I	PDS1 IFS1	ST	—	PTM3 捕捉输入
	PTP3	PDS1	—	CMOS	PTM3 输出
	RX0	PDS1 IFS3	ST	—	UART0 RX 串行数据输入
	C1-	PDS1	AN	—	比较器 1 负极输入
PD5/PTCK3/ TX0/C1+	PD5	PDP PDS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK3	PDS1 IFS0	ST	—	PTM3 时钟输入
	TX0	PDS1	—	CMOS	UART0 TX 串行数据输出
	C1+	PDS1	AN	—	比较器 1 正极输入

引脚名称	功能	OPT	I/T	O/T	说明
PD6/STP2I/ STP2/C1X	PD6	PDP PDS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STP2I	PDS1 IFS1	ST	—	STM2 捕捉输入
	STP2	PDS1	—	CMOS	STM2 输出
	C1X	PDS1	—	CMOS	比较器 1 输出
PE0/STCK0/ SCSA	PE0	PEPU PES0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STCK0	PES0 IFS0	ST	—	STM0 时钟输入
	SCSA	PES0	ST	CMOS	SPIA 从机选择
PE1/STP0I/ STP0/SDOA	PE1	PEPU PES0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STP0I	PES0 IFS	ST	—	STM0 捕捉输入
	STP0	PES0	—	CMOS	STM0 反相输出
	SDOA	PES0	—	CMOS	SPIA 数据输出
PE2/PTCK1/ SDIA	PE2	PEPU PES0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK1	PES0 IFS0	ST	—	PTM1 时钟输入
	SDIA	PES0	ST	—	SPIA 数据输入
PE3/PTP1I/ PTP1/SCKA	PE3	PEPU PES0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP1I	PES0 IFS1	ST	—	PTM1 捕捉输入
	PTP1	PES0	—	CMOS	PTM1 输出
	SCKA	PES0	ST	CMOS	SPIA 串行时钟
PE4/VDDIO	PE4	PEPU PES1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	VDDIO	PES1 PMPS	PWR	—	PE0~PE3 引脚电源选择
PF0/SCS/ SCOM0	PF0	PFPU PFS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	SCS	PFS0 IFS2	ST	CMOS	SPI 从机选择
	SCOM0	PFS0	—	CMOS	软件 LCD COM 输出
PF1/SDO/ SCOM1	PF1	PFPU PFS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	SDO	PFS0	—	CMOS	SPI 数据输出
	SCOM1	PFS0	—	CMOS	软件 LCD COM 输出

引脚名称	功能	OPT	I/T	O/T	说明
PF2/SDI/SDA/ SCOM2	PF2	PFPU PFS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	SDI	PFS0 IFS2	ST	—	SPI 数据输入
	SDA	PFS0 IFS2	ST	NMOS	I ² C 数据线
	SCOM2	PFS0	—	CMOS	软件 LCD COM 输出
PF3/SCK/SCL/ SCOM3	PF3	PFPU PFS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	SCK	PFS0 IFS2	ST	CMOS	SPI 串行时钟
	SCL	PFS0 IFS2	ST	NMOS	I ² C 时钟线
	SCOM3	PFS0	—	CMOS	软件 LCD COM 输出
PF4/PTCK0/ XT2	PF4	PFPU PFS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTCK0	PFS1 IFS0	ST	—	PTM0 时钟输入
	XT2	PFS1	—	LXT	LXT 振荡器引脚
PF5/PTP0I/ PTP0/XT1	PF5	PFPU PFS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	PTP0I	PFS1 IFS1	ST	—	PTM0 捕捉输入
	PTP0	PFS1	—	CMOS	PTM0 输出
	XT1	PFS1	LXT	—	LXT 振荡器引脚
PF6/STCK2/ RX1/C0-	PF6	PFPU PFS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STCK2	PFS1 IFS0	ST	—	STM2 时钟输入
	RX1	PFS1 IFS3	ST	—	UART1 RX 串行数据输入
	C0-	PFS1	AN	—	比较器 0 负极输入
PF7/STP2I/ STP2/TX1/C0+	PF7	PFPU PFS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	STP2I	PFS1 IFS1	ST	—	STM2 捕捉输入
	STP2	PFS1	—	CMOS	STM2 输出
	TX1	PFS1	—	CMOS	UART1 TX 串行数据输出
	C0+	PDS1	AN	—	比较器 0 正极输入
PG0/RX2	PG0	PGPU PGS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	RX2	PGS0 IFS3	ST	—	UART2 RX 串行数据输入

引脚名称	功能	OPT	I/T	O/T	说明
PG1/TX2	PG1	PGPU PGS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	TX2	PGS0	—	CMOS	UART2 TX 串行数据输出
PG2~PG3	PGn	PGPU PGS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
PG4~PG7	PGn	PGPU PGS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
PH0~PH1	PHn	PHPU PHS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
PH2/AN12	PH2	PHPU PHS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	AN12	PHS0	AN	—	A/D 转换器模拟输入
PH3/AN13	PH3	PHPU PHS0	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	AN13	PHS0	AN	—	A/D 转换器模拟输入
PH4/AN14	PH4	PHPU PHS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	AN14	PHS1	AN	—	A/D 转换器模拟输入
PH5/AN15	PH5	PHPU PHS1	ST	CMOS	通用 I/O 口。 可通过寄存器设置上拉电阻。
	AN15	PHS1	AN	—	A/D 转换器模拟输入
VDD	VDD	—	PWR	—	正电源
VSS	VSS	—	PWR	—	负电源，接地
AVDD	AVDD	—	PWR	—	模拟正电源
AVSS	AVSS	—	PWR	—	模拟负电源，接地

注：I/T：输入类型；

OPT：通过配置寄存器选项来配置；

NMOS：NMOS 输出；

AN：模拟信号；

HXT：高频晶体振荡器

O/T：输出类型；

CMOS：CMOS 输出；

ST：施密特触发输入；

PWR：电源；

LXT：低频晶体振荡器

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	$-60^{\circ}C \sim 150^{\circ}C$
工作温度	$-40^{\circ}C \sim 85^{\circ}C$
I_{OH} 总电流	-80mA
I_{OL} 总电流	80mA
总功耗	500mW

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

直流电气特性

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

工作电压特性

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

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 - HXT	$f_{SYS} = 8MHz$	2.2	—	5.5	V
		$f_{SYS} = 12MHz$	2.7	—	5.5	
		$f_{SYS} = 16MHz$	3.3	—	5.5	
	工作电压 - HIRC	$f_{SYS} = 8MHz$	2.2	—	5.5	V
		$f_{SYS} = 12MHz$	2.7	—	5.5	
		$f_{SYS} = 16MHz$	3.3	—	5.5	
	工作电压 - LXT	$f_{SYS} = 32768Hz$	2.2	—	5.5	V
	工作电压 - LIRC	$f_{SYS} = 32kHz$	2.2	—	5.5	V

待机电流特性

Ta=25°C

符号	待机模式	测试条件		最小	典型	最大	最大 85°C	单位
		V _{DD}	条件					
I _{STB}	休眠模式	2.2V	WDT off	—	0.11	0.15	2.0	μA
		3V		—	0.11	0.15	2.0	
		5V		—	0.18	0.38	2.9	
		2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	—	3	3.6	
		5V		—	—	5	6	
	空闲模式 0	2.2V	f _{SUB} on	—	2.4	4	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	空闲模式 1 – HIRC	2.2V	f _{SUB} on, f _{SYS} = 8MHz	—	0.3	0.6	0.8	mA
		3V		—	0.5	1.0	1.8	
		5V		—	1.0	2.0	2.2	
		2.7V	f _{SUB} on, f _{SYS} = 12MHz	—	0.4	0.8	1.0	mA
		3V		—	0.6	1.2	1.4	
		5V		—	1.2	2.4	2.6	
		3.3V	f _{SUB} on, f _{SYS} = 16MHz	—	1.5	3.0	3.2	mA
		5V		—	2.0	4.0	4.2	
	空闲模式 1 – HXT	2.2V	f _{SUB} on, f _{SYS} = 8MHz	—	0.3	0.6	0.8	mA
		3V		—	0.5	1.0	1.2	
		5V		—	1.0	2.0	2.2	
		2.7V	f _{SUB} on, f _{SYS} = 12MHz	—	0.4	0.8	1.0	mA
		3V		—	0.6	1.2	1.4	
		5V		—	1.2	2.4	2.6	
		3.3V	f _{SUB} on, f _{SYS} = 16MHz	—	1.5	3.0	3.2	mA
5V		—		2.0	4.0	4.2		

注：当使用该表格电气特性数据时，以下几点需注意：

- 任何数字输入都设置为非浮空的状态。
- 所有测量都在无负载且所有外围功能关闭的条件下进行。
- 无直流电流通路。
- 所有待机电流数值都是在 HALT 指令执行后测得，因此 HALT 后停止执行所有指令。

工作电流特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位	
		V _{DD}	条件					
I _{DD}	低速模式 – LXT	2.2V	f _{sys} = 32.768kHz	—	8	16	μA	
		3V		—	10	20		
		5V		—	30	50		
	低速模式 – LIRC	2.2V	f _{sys} = 32kHz	—	8	16	μA	
		3V		—	10	20		
		5V		—	30	50		
	快速模式 – HIRC	2.2V	f _{sys} = 8MHz	—	0.3	0.5	mA	
				3V	—	0.8		1.2
				5V	—	1.6		2.4
		2.7V	f _{sys} = 12MHz	—	0.6	1.0	mA	
				3V	—	1.2		1.8
				5V	—	2.4		3.6
		3.3V	f _{sys} = 16MHz	—	1.0	1.4	mA	
				5V	—	6.0		9.0
				5V	—	6.0		9.0
	快速模式 – HXT	2.2V	f _{sys} = 8MHz	—	0.8	1.2	mA	
				3V	—	1		1.5
				5V	—	2		3
		2.7V	f _{sys} = 12MHz	—	1.2	2.2	mA	
				3V	—	1.5		2.75
				5V	—	3		4.5
3.3V		f _{sys} = 16MHz	—	3.2	4.8	mA		
			5V	—	4.5		7.0	
			5V	—	4.5		7.0	

注：当使用该表格电气特性数据时，以下几点需注意：

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

交流电气特性

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

内部高速振荡器 – HIRC – 频率精确度

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

8/12/16 MHz

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
f _{HIRC}	通过烧录器调整后的 8MHz HIRC 频率	3V/5V	T _a =25°C	-1.0%	8	+1.0%	MHz
			T _a =-40°C ~ 85°C	-2.0%	8	+2.0%	
		2.2V~5.5V	T _a =25°C	-2.5%	8	+2.5%	
			T _a =-40°C ~ 85°C	-3.0%	8	+3.0%	
	通过烧录器调整后的 12MHz HIRC 频率	3V/5V	T _a =25°C	-1.0%	12	+1.0%	MHz
			T _a =-40°C ~ 85°C	-2.0%	12	+2.0%	
		2.7V~5.5V	T _a =25°C	-2.5%	12	+2.5%	
			T _a =-40°C ~ 85°C	-3.0%	12	+3.0%	
	通过烧录器调整后的 16MHz HIRC 频率	5V	T _a =25°C	-1.0%	16	+1.0%	MHz
			T _a =-40°C ~ 85°C	-2.0%	16	+2.0%	
		3.3V~5.5V	T _a =25°C	-2.5%	16	+2.5%	
			T _a =-40°C ~ 85°C	-3.0%	16	+3.0%	

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

内部低速振荡器电气特性 – LIRC

Ta=25°C, 除非另有指定

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
f _{LIRC}	振荡器频率	2.2V~5.5V	Ta=25°C	-2%	32	+2%	kHz
			Ta=-40°C ~ 85°C	-10%	32	+10%	
t _{START}	启动时间	—	—	—	—	100	μs

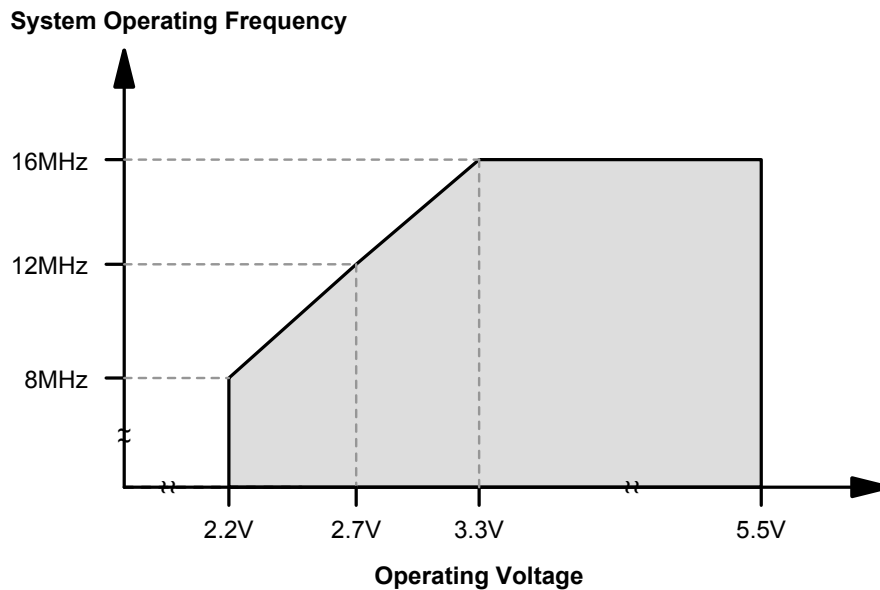
低速晶体振荡器电气特性 – LXT

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
f _{LXT}	振荡器频率	2.2~5.5V	f _{sys} =f _{LXT} =32.768kHz	-10%	32.768	+10%	kHz
Duty Cycle	占空比	—	—	45	50	55	%
t _{START}	启动时间	—	—	—	—	500	ms
R _{NEG}	负阻*	2.2V	—	3×ESR	—	—	Ω

*: C1, C2 和 R_p 为外部元器件。C1=C2=10pF, R_p=10MΩ。C_L=7pF, ESR=30kΩ。

工作频率电气特性曲线图



系统上电时间电气特性

Ta = -40°C ~ 85°C

符号	参数	测试条件	最小	典型	最大	单位
t _{SST}	系统启动时间 (从 f _{sys} off 的状态下唤醒)	f _{sys} = f _H ~ f _H /64, f _H = f _{HXT}	—	128	—	t _{HXT}
		f _{sys} = f _H ~ f _H /64, f _H = f _{HIRC}	—	16	—	t _{HIRC}
		f _{sys} = f _{SUB} = f _{LXT}	—	1024	—	t _{LXT}
		f _{sys} = f _{SUB} = f _{LIRC}	—	2	—	t _{LIRC}
	系统启动时间 (从 f _{sys} on 的状态下唤醒)	f _{sys} = f _H ~ f _H /64, f _H = f _{HXT} 或 f _{HIRC}	—	2	—	t _H
		f _{sys} = f _{SUB} = f _{LXT} 或 f _{LIRC}	—	2	—	t _{SUB}
	系统速度切换时间 (快速模式 → 低速模式或 低速模式 → 快速模式)	f _{HXT} off → on	—	1024	—	t _{HXT}
f _{HIRC} off → on		—	16	—	t _{HIRC}	
f _{LXT} off → on		—	1024	—	t _{LXT}	
t _{RSTD}	系统复位延迟时间 (上电复位或 LVR 硬件复位)	RR _{POR} =5V/ms	42	48	54	ms
	系统复位延迟时间 (LVRC/WDTC/RSTC 软件复位)	—				
	系统复位延迟时间 (WDT 溢出或外部复位引脚复位)	—	14	16	18	

- 注：1. 系统启动时间里提到的 f_{sys} on/off 状态取决于工作模式类型以及所选的系统时钟振荡器。更多相关细节请参考系统工作模式章节。
2. t_{HXT}, t_{HIRC} 等符号所表示的时间单位，是对应频率值的倒数，相关频率值在前面表格有说明。例如，t_{HIRC} = 1/f_{HIRC}, t_{sys} = 1/f_{sys} 等等。
3. LIRC 被选择作为系统时钟源且在休眠模式下 LIRC 关闭，则上面表格中对应 t_{SST} 数值还需加上 LIRC 频率表格里提供的 LIRC 启动时间 t_{START}。
4. 系统速度切换时间实际上是指新使能的振荡器的启动时间。

输入 / 输出口电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{IL}	I/O 口或输入引脚 低电平输入电压	5V	—	0	—	1.5	V
		—	—	0	—	0.2V _{DD}	
V _{IH}	I/O 口或输入引脚 高电平输入电压	5V	—	3.5	—	5.0	V
		—	—	0.8V _{DD}	—	V _{DD}	
I _{OL}	I/O 口灌电流	3V	V _{OL} = 0.1V _{DD}	16	32	—	mA
		5V		32	64	—	
I _{OH}	I/O 口源电流	3V	V _{OH} = 0.9V _{DD} , SLEDCn[m+1:m]=00, n = A, B, C, D 或 E; m=0, 2, 4, 6	-1.0	-2.0	—	mA
		5V		-2.0	-4.0	—	
		3V	V _{OH} = 0.9V _{DD} , SLEDCn[m+1:m]=01, n = A, B, C, D 或 E; m=0, 2, 4, 6	-1.75	-3.5	—	
		5V		-3.5	-7.0	—	
		3V	V _{OH} = 0.9V _{DD} , SLEDCn[m+1:m]=10, n = A, B, C, D 或 E; m=0, 2, 4, 6	-2.5	-5.0	—	
		5V		-5.0	-10.0	—	
		3V	V _{OH} = 0.9V _{DD} , SLEDCn[m+1:m]=11, n = A, B, C, D 或 E; m=0, 2, 4, 6	-5.5	-11.0	—	
		5V		-11.0	-22.0	—	
R _{PH}	I/O 口上拉电阻 (注)	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I _{LEAK}	输入漏电流	5V	V _{IN} = V _{DD} 或 V _{IN} = V _{SS}	—	—	±1	μA
t _{TPI}	TM 捕捉输入最小脉宽	—	—	0.3	—	—	μs
t _{TCK}	TM 时钟输入最小脉宽	—	—	0.3	—	—	μs
t _{INT}	中断引脚最小输入脉宽	—	—	10	—	—	μs

注: R_{PH} 内部上拉电阻值的计算方法是: 将其接地并使能输入引脚的上拉电阻选项, 然后在特定电源电压下测量输入灌电流, 在此基础上测量上拉电阻上的分压从而得到此上拉电阻值。

存储器电气特性

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

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{RW}	读 / 写工作电压	—	—	V _{DDmin}	—	V _{DDmax}	V
Flash 程序存储器 / EEPROM 存储器							
t _{DEW}	擦除 / 写周期时间	—	—	2.2	2.5	2.8	ms
I _{DDPGM}	V _{DD} 电压下烧录 / 擦除电流	—	—	—	—	5.0	mA
E _P	电容耐久性 – Flash 程序存储器	—	—	10K	—	—	E/W
	电容耐久性 – EEPROM 存储器	—	—	100K	—	—	
t _{RETd}	ROM 数据保存时间	—	T _a = 25°C	—	40	—	Year
RAM 数据存储器							
V _{DR}	RAM 数据保存时间	—	单片机处于休眠模式	1.0	—	—	V

A/D 转换器电气特性

 $T_a = 25^{\circ}\text{C}$

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{DD}	A/D 转换器工作电压	—	—	2.2	—	5.5	V
V _{ADI}	A/D 转换器输入电压	—	—	0	—	V _{REF}	V
V _{REF}	A/D 转换器参考电压	—	—	2	—	V _{DD}	V
DNL	非线性微分误差	—	V _{REF} =V _{DD} , t _{ADCK} =0.5μs T _a = -40°C~85°C	-3	—	+3	LSB
INL	非线性积分误差	—	V _{REF} =V _{DD} , t _{ADCK} =0.5μs T _a = -40°C~85°C	-4	—	+4	LSB
I _{ADC}	使能 A/D 转换器增加的电流损耗	2.2V	无负载 (t _{ADCK} =0.5μs)	—	1.0	2.0	mA
		3V		—	1.0	2.0	
		5V		—	1.5	3.0	
t _{ADCK}	A/D 转换器时钟周期	—	—	0.5	—	10	μs
t _{ADS}	A/D 转换器采样时间	—	—	—	4	—	t _{ADCK}
t _{ADC}	A/D 转换时间 (包括采样和保持时间)	—	—	—	16	—	t _{ADCK}
t _{ON2ST}	A/D 转换器 On-to-Start 时间	—	—	4	—	—	μs
t _{START}	OPA 启动稳定时间	—	无外挂负载	—	—	22	μs
I _{PGA}	使能 PGA 增加的电流损耗	3V	无负载	—	300	450	μA
		5V		—	400	550	
V _{CM}	PGA 共模电压范围	3V	—	V _{SS}	—	V _{DD}	V
		5V		-0.3	—	-1.4	
V _{OR}	PGA 最大输出电压范围	3V	—	V _{SS}	—	V _{DD}	V
		5V		+0.1	—	-0.1	
V _{VR}	PGA 固定电压输出	5V	T _a = 25°C	-1%	2	+1%	V
					3		
					4		

比较器电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{DD}	工作电压	—	—	2.7	—	5.5	V
I _{COMP}	使能比较器所增加的电流损耗	3V 5V	CNVTn[1:0] = 00	— —	— 1	3 3	μA
V _{OS}	输入失调电压 (注)	3V/5V	无校准, CNVTn[1:0] = 00B, CnOF[4:0] = 10000	-10	—	10	mV
		3V/5V	校准后, CNVTn[1:0] = 00B	-4	—	4	
V _{CM}	共模电压范围	—	CNVTn[1:0] = 00	V _{SS}	—	V _{DD} -1.4	V
A _{OL}	开环增益	3V	CNVTn[1:0] = 00	60	—	—	dB
		5V	CNVTn[1:0] = 00	60	80	—	
V _{HYS}	迟滞	3V	CNVTn[1:0] = 00	10	—	30	mV
		5V	CNVTn[1:0] = 00	10	24	30	
t _{RP}	响应时间	3V/5V	10mV 过载, C _{LOAD} =3pF, CNVTn[1:0] = 00	—	25	40	μs
			100mV 过载, C _{LOAD} =3pF, CNVTn[1:0] = 00	—	20	40	
		3V/5V	10mV 过载, C _{LOAD} =3pF, CNVTn[1:0] = 01	—	1.5	4	
			100mV 过载, C _{LOAD} =3pF, CNVTn[1:0] = 01	—	1.2	3	
		3V/5V	10mV 过载, C _{LOAD} =3pF, CNVTn[1:0] = 10	—	0.8	2	
			100mV 过载, C _{LOAD} =3pF, CNVTn[1:0] = 10	—	0.5	1.5	
3V/5V	10mV 过载, C _{LOAD} =3pF, CNVTn[1:0] = 11	—	0.7	1.5			
	100mV 过载, C _{LOAD} =3pF, CNVTn[1:0] = 11	—	0.3	1			

注：当比较器的比较阈值电压低于 250mV 时输入失调电压应预先校准，否则输入失调电压将超出特性范围。

软件控制 LCD 驱动器电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
I _{BIAS}	V _{DD} /2 偏压电流	3V	ISEL[1:0]=00	8	—	16	μA
		5V		17.5	25	32.5	
		3V	ISEL[1:0]=01	18	—	32	
		5V		35	50	65	
		3V	ISEL[1:0]=10	35	—	65	
		5V		70	100	130	
		3V	ISEL[1:0]=11	70	—	130	
		5V		140	200	260	
V _{SCOM}	V _{DD} /2 偏压 LCD SCOM 输出	2.2V~5.5V	无负载	0.475V _{DD}	0.5V _{DD}	0.525V _{DD}	V

I²C 电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
f _{I2C}	I ² C 标准模式下系统频率 (100kHz)	—	无时钟去抖动	2	—	—	MHz
		—	2 个系统时钟去抖动	4	—	—	
		—	4 个系统时钟去抖动	8	—	—	
	I ² C 快速模式下系统频率 (400kHz)	—	无时钟去抖动	5	—	—	MHz
		—	2 个系统时钟去抖动	10	—	—	
		—	4 个系统时钟去抖动	20	—	—	

LVD/LVR 电气特性

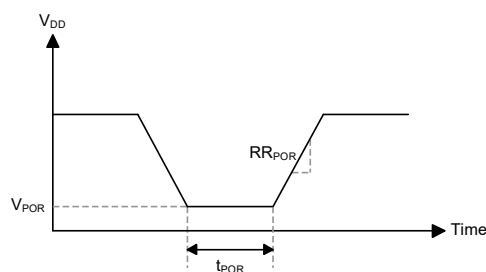
Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V _{DD}	条件				
V _{LVR}	低压复位电压	—	LVR 使能, 电压选择 2.1V	-5%	2.1	+5%	V
			LVR 使能, 电压选择 2.55V		2.55		
			LVR 使能, 电压选择 3.15V		3.15		
			LVR 使能, 电压选择 3.8V		3.8		
V _{LVD}	低压检测电压	—	LVD 使能, 电压选择 2.0V	-5%	2.0	+5%	V
			LVD 使能, 电压选择 2.2V		2.2		
			LVD 使能, 电压选择 2.4V		2.4		
			LVD 使能, 电压选择 2.7V		2.7		
			LVD 使能, 电压选择 3.0V		3.0		
			LVD 使能, 电压选择 3.3V		3.3		
			LVD 使能, 电压选择 3.6V		3.6		
LVD 使能, 电压选择 4.0V	4.0						
I _{LVR}	使能 LVR 所增加的电流损耗	—	LVD 除能, VBGEN=0	—	—	25	μA
I _{LVD}	使能 LVD 所增加的电流损耗	—	LVR 除能, VBGEN=0	—	—	25	μA
t _{LVDS}	LVDO 稳定时间	—	LVR 使能, VBGEN = 0, LVD off → on	—	—	15	μs
		—	LVR 除能, VBGEN = 0, LVD off → on	—	—	150	μs
t _{LVR}	最小低电压复位时间	—	—	120	240	480	μs
t _{BGS}	V _{BG} 启动稳定时间	—	无负载	—	—	200	μs
t _{LVD}	最小低电压中断时间	—	—	60	120	240	μs

上电复位特性

Ta=25°C

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



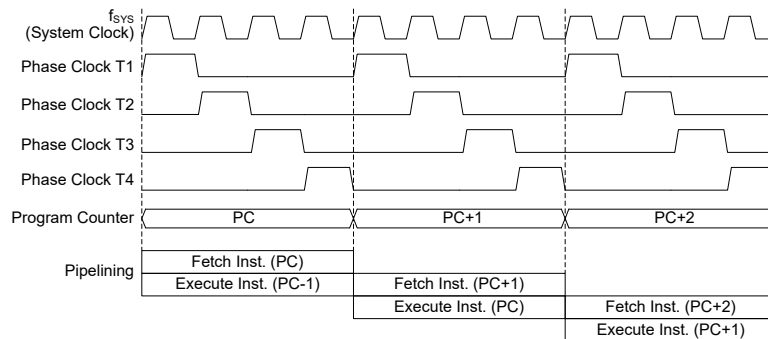
系统结构

内部系统结构是 Holtek 单片机具有良好性能的主要因素。由于采用 RISC 结构，该系列单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的取得和执行同时进行，此举使得除了跳转和调用指令需要多一个指令周期外，大部分的标准指令或扩展指令分别能在一个指令周期或两个指令周期内完成。8-bit 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。这些使得此系列单片机适用于低成本和批量生产的控制应用。

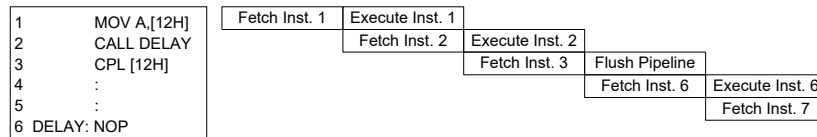
时序和流水线结构

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

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



系统时序和流水线



指令捕捉

程序计数器

在程序执行期间，程序计数器用来指向下一个要执行的指令地址。除了“JMP”和“CALL”指令需要跳转到一个非连续的程序存储器地址之外，它会在每条指令执行完成以后自动加一。对于存储器容量大于 8K 的单片机，其程序存储器地址可能位于某一程序存储区中，可通过程序存储区指针的 PBP 位段来选择。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。

单片机型号	程序计数器	
	高字节	低字节 (PCL)
HT66F2350	PC12~PC8	PC7~PC0
HT66F2360	PBP0, PC12~PC8	PC7~PC0
HT66F2370	PBP1~PBP0, PC12~PC8	PC7~PC0
HT66F2390	PBP2~PBP0, PC12~PC8	PC7~PC0

程序计数器

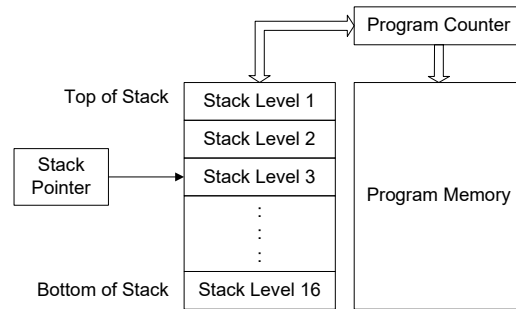
程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内。注意，当这样一个程序跳转要执行时，会插入一个空指令周期。PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

堆栈

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

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

若堆栈溢出，则首个存入堆栈的程序计数器数据将会丢失。



算术逻辑单元 – ALU

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

- 算术运算：
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- 逻辑运算：
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
LAND, LOR, LXOR, LANDM, LORM, LXORM, LCPL, LCPLA
- 移位运算：
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
LRRR, LRR, LRRCA, LRRC, LRLA, LRL, LRLCA, LRLC
- 递增和递减：
INCA, INC, DECA, DEC
LINCA, LINC, LDECA, LDEC
- 分支判断：
JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI
LSZ, LSZA, LSNZ, LSIZ, LSIZA, LSDZ, LSDZA

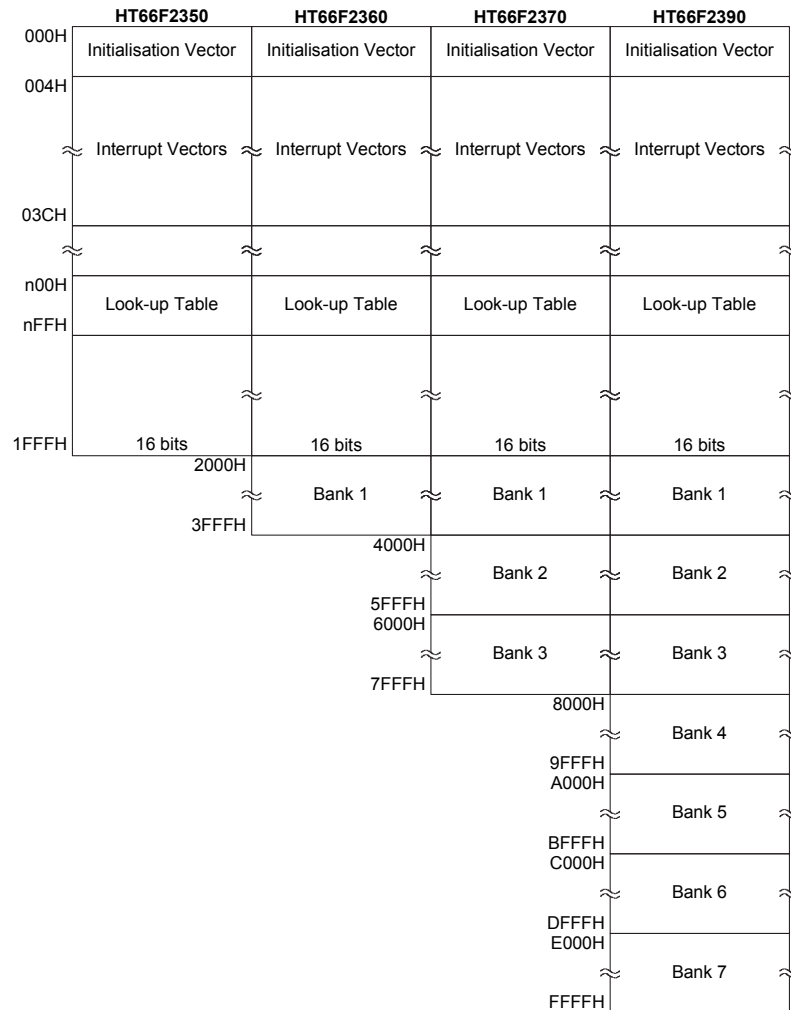
Flash 程序存储器

程序存储器用来存放用户代码及储存程序。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，此系列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。

单片机型号	容量	Bank
HT66F2350	8K×16	—
HT66F2360	16K×16	0~1
HT66F2370	32K×16	0~3
HT66F2390	64K×16	0~7

结构

程序存储器的容量为 8K×16~64K×16 位，程序存储器用程序计数器来寻址，其中也包含数据、表格和中断入口。数据表格可以设定在程序存储器的任何地址，由表格指针来寻址。



程序存储器结构

特殊向量

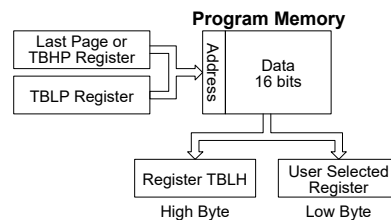
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

查表

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

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

下图是查表中寻址 / 数据流程：



查表范例

以下范例说明表格指针和表格数据如何被定义和执行。这个例子使用的表格数据用 ORG 伪指令储存在存储器中。ORG 指令的值“1F00H”指向的地址是 8K 程序存储器中最后一页的起始地址。表格指针低字节寄存器的初始值设为 06H，这可保证从数据表格读取的第一笔数据位于程序存储器地址“1F06H”，即最后一页起始地址后的第六个地址。值得注意的是，假如“TABRD [m]”指令被使用，则表格指针指向 TBHP 寄存器所指定的当前页的第一个地址。在这个例子中，表格数据的高字节等于零，而当“TABRD [m]”指令被执行时，此值将会自动的被传送到 TBLH 寄存器。

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

表格读取程序范例

```

tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
mov a,06h          ; initialise low table pointer - note that this address
                  ; is referenced
mov tblp,a         ; to the last page or the page that tbhp pointed
mov a,1fh          ; initialise high table pointer
mov tbhp,a
:
tabrd tempreg1     ; transfers value in table referenced by table pointer
                  ; data at program
                  ; memory address "1F06H" transferred to tempreg1 and TBLH
dec tblp           ; reduce value of table pointer by one
tabrd tempreg2     ; transfers value in table referenced by table pointer
                  ; data at program memory address "1F05H" transferred to
                  ; tempreg2 and TBLH in this example the data "1AH" is
                  ; transferred to tempreg1 and data "0FH" to register tempreg2
:
org 1F00h          ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:

```

在线烧录 – ICP

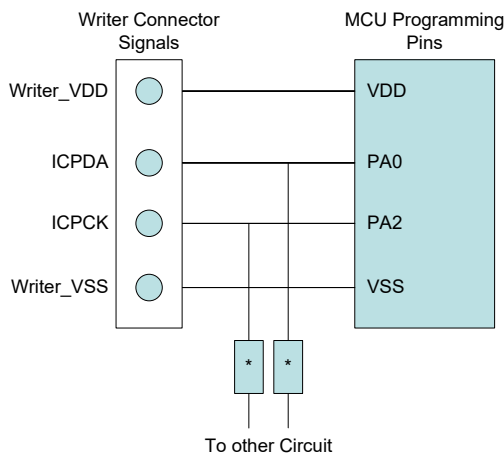
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。

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

Holtek 烧录器引脚	MCU 在线烧录引脚	引脚描述
ICPDA	PA0	串行数据输入 / 输出
ICPCK	PA2	串行时钟
VDD	VDD	电源
VSS	VSS	地

单片机内部程序存储器可以通过 4 线的接口在线进行烧录。其中一条线用于数据串行下载或上传，一条用于串行时钟，剩余两条用于提供电源。芯片在线烧写的详细使用说明超出此文档的描述范围，将由专门的参考文献提供。

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



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

片上调试 – OCDS

EV 芯片 HT66V23x0 分别用于 HT66F23x0 单片机仿真。此 EV 芯片提供片上调试功能 (OCDS – On-Chip Debug Support) 用于开发过程中的单片机调试。除了片上调试功能方面，EV 芯片和实际单片机在功能上几乎是兼容的。用户可将 OCSDA 和 OCDSCK 引脚连接至 Holtek HT-IDE 开发工具，从而实现 EV 芯片对实际单片机的仿真。OCSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户用 EV 芯片进行调试时，实际单片机 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 用户手册”文件。

Holtek e-Link 引脚	EV 芯片引脚	引脚描述
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, AVDD	电源
GND	VSS, AVSS	地

在线应用编程 – IAP

该系列单片机提供 IAP 功能来对 Flash 程序存储器进行数据和程序更新。用户可自行定义 IAP 程序存储器地址，但是用户在使用 IAP 功能时必须注意几个特点。

HT66F2350 配置		HT66F2360/HT66F2370 配置		HT66F2390 配置	
擦除页	32 个字 / 页	擦除页	64 个字 / 页	擦除页	128 个字 / 页
写	32 个字 / 次	写	64 个字 / 次	写	128 个字 / 次
读	1 个字 / 次	读	1 个字 / 次	读	1 个字 / 次

在线应用编程控制寄存器

地址寄存器 FARL/FARH、数据寄存器 FD0L/FD0H、FD1L/FD1H、FD2L/FD2H 和 FD3L/FD3H 以及控制寄存器 FC0、FC1 和 FC2，都是与 IAP 相关的 Flash 存取寄存器，分别位于数据存储器中的 Sector 0 和 Sector 1。如果使用间接寻址方式来存取 FC0、FC1 和 FC2，所有与这些寄存器相关的读写操作必须使用间接寻址寄存器 IAR1 或 IAR2，和存储器指针 MP1L/MP1H 或 MP2L/MP2H 来执行。由于 FC0、FC1 和 FC2 控制寄存器位于数据存储器 Sector 1 地址 43H~45H 中，因此，地址值 43H~45H 必须首先被写入 MP1L 或 MP2L 存储器指针低字节，且“01H”值也被写入 MP1H 或 MP2H 存储器指针高字节。

寄存器名称	位							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	—	CLWB
FARL	A7	A6	A5	A4	A3	A2	A1	A0
FARH (HT66F2350)	—	—	—	A12	A11	A10	A9	A8
FARH (HT66F2360)	—	—	A13	A12	A11	A10	A9	A8
FARH (HT66F2370)	—	A14	A13	A12	A11	A10	A9	A8
FARH (HT66F2390)	A15	A14	A13	A12	A11	A10	A9	A8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP 寄存器列表

● FC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7 **CFWEN**: Flash 程序存储器写使能控制位
 0: Flash 程序存储器写功能除能
 1: Flash 程序存储器写功能已被成功使能
 当此位由应用程序清零后, Flash 程序存储器写功能除能。注意, 此位写“1”无效。此位被用来说明 Flash 程序存储器写功能状态。当此位由硬件置为“1”时, 就意味着 Flash 程序存储器写功能已经成功使能, 否则此位为“0”, 该功能除能。
- Bit 6~4 **FMOD2~FMOD0**: 模式选择
 000: 写程序存储器
 001: 页擦除程序存储器
 010: 保留位
 011: 读程序存储器
 10x: 保留位
 110: FWEN 模式 – Flash 程序存储器写功能使能模式
 111: 保留位
- Bit 3 **FWPEN**: Flash 程序存储器写步骤使能控制
 0: 除能
 1: 使能
 当此位置为“1”且 FMOD2~FMOD0 为“110”时, IAP 控制器将执行“Flash 程序存储器写功能使能”步骤。一旦 Flash 程序存储器写功能成功使能, 无需再设置 FWPEN 位。
- Bit 2 **FWT**: Flash 程序存储器写开始控制位
 0: 不开始 Flash 程序存储器写过程或 Flash 程序存储器写过程已完成
 1: 开始 Flash 程序存储器写过程
 此位由软件置“1”。当 Flash 程序存储器写过程完成, 由硬件清零。
- Bit 1 **FRDEN**: Flash 程序存储器读使能控制位
 0: Flash 程序存储器读除能
 1: Flash 程序存储器读使能
- Bit 0 **FRD**: Flash 程序存储器读开始控制位
 0: 不开始 Flash 程序存储器读或 Flash 程序存储器读过程已完成
 1: 开始 Flash 程序存储器读过程
 此位由软件置“1”。当 Flash 程序存储器读过程完成, 由硬件清零。

● FC1 寄存器

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

- Bit 7~0 **D7~D0**: 整个芯片复位
 当用户写“55H”到该寄存器, 将产生一个复位信号将整个单片机复位。

● FC2 寄存器

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

Bit 7~1 未定义，读为“0”

Bit 0 **CLWB**: Flash 存储器写缓冲清除控制位

0: 不开始写缓冲区清除过程或写缓冲清除过程已完成

1: 开始写缓冲区清除过程

此位由软件置“1”。当写缓冲区清除过程完成，由硬件清零。

● FARL 寄存器

Bit	7	6	5	4	3	2	1	0
Name	A7	A6	A5	A4	A3	A2	A1	A0
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 Flash 程序存储器地址 bit 7 ~ bit 0

● FARH 寄存器 – HT66F2350

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	A12	A11	A10	A9	A8
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 未定义，读为“0”

Bit 4~0 Flash 存储器地址 bit 12 ~ bit 8

● FARH 寄存器 – HT66F2360

Bit	7	6	5	4	3	2	1	0
Name	—	—	A13	A12	A11	A10	A9	A8
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 Flash 存储器地址 bit 13 ~ bit 8

● FARH 寄存器 – HT66F2370

Bit	7	6	5	4	3	2	1	0
Name	—	A14	A13	A12	A11	A10	A9	A8
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 未定义，读为“0”

Bit 6~0 Flash 存储器地址 bit 14 ~ bit 8

● **FARH 寄存器 – HT66F2390**

Bit	7	6	5	4	3	2	1	0
Name	A15	A14	A13	A12	A11	A10	A9	A8
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 Flash 存储器地址 bit 15 ~ bit 8

● **FD0L 寄存器**

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

Bit 7~0 第一个 Flash 程序存储器数据 bit 7 ~ bit 0

应注意写入低字节数据寄存器 FD0L 的数据仅存储在 FD0L 寄存器而不会被载入低 8 位写缓存器。

● **FD0H 寄存器**

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

Bit 7~0 第一个 Flash 程序存储器数据 bit 15 ~ bit 8

应注意当 8-bit 数据写入高字节数据寄存器 FD0H 时，存储在 FD0H 和 FD0L 寄存器中的总的 16-bit 数据将同时被载入 16-bit 写缓存器，之后 Flash 程序存储器地址寄存器对 FARH 和 FARL 的内容将加一。

● **FD1L 寄存器**

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

Bit 7~0 第二个 Flash 程序存储器数据 bit 7 ~ bit 0

● **FD1H 寄存器**

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

Bit 7~0 第二个 Flash 程序存储器数据 bit 15 ~ bit 8

● **FD2L 寄存器**

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

Bit 7~0 第三个 Flash 程序存储器数据 bit 7 ~ bit 0

● **FD2H 寄存器**

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

Bit 7~0 第三个 Flash 程序存储器数据 bit 15 ~ bit 8

● **FD3L 寄存器**

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

Bit 7~0 第四个 Flash 程序存储器数据 bit 7 ~ bit 0

● **FD3H 寄存器**

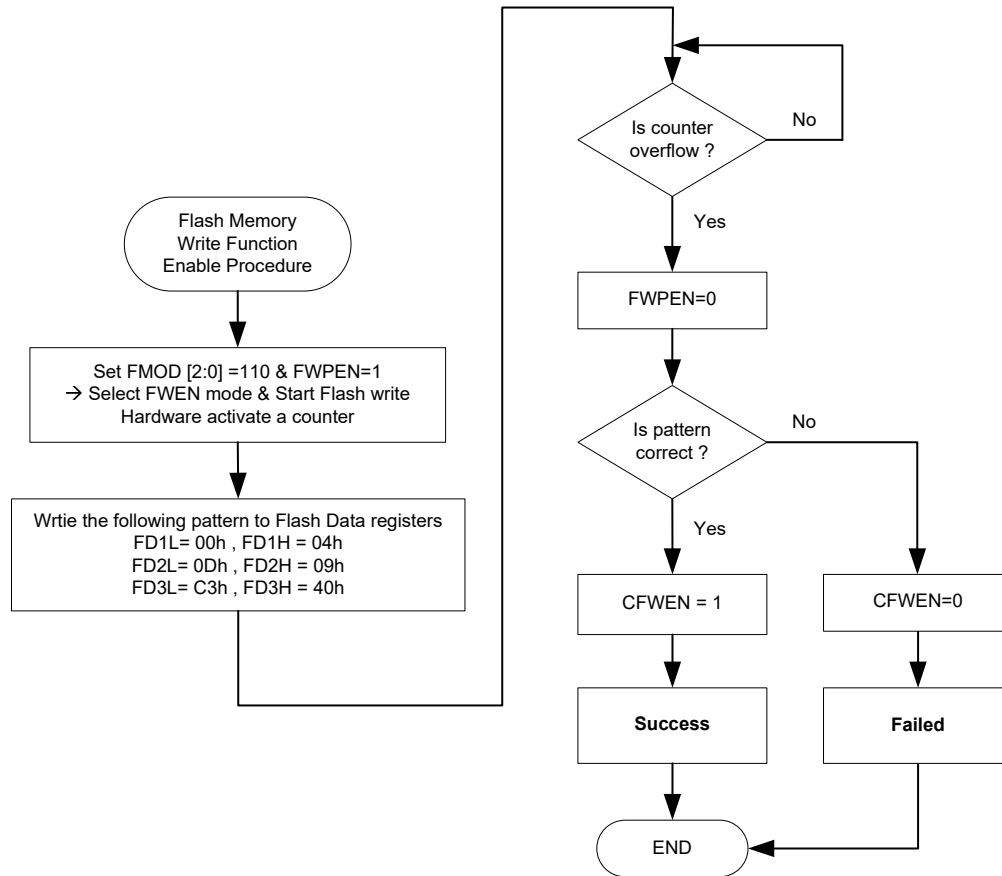
Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 第四个 Flash 程序存储器数据 bit 15 ~ bit 8

Flash 程序存储器写功能使能步骤

为使用户可以通过 IAP 控制寄存器来更改 Flash 程序存储器数据，用户必须首先使能 Flash 程序存储器写操作，步骤如下：

- 步骤 1. 写“110”到 FMOD2~FMOD0 位，选择 FWEN 模式。
- 步骤 2. FWPEN 置为“1”。步骤 1 和步骤 2 可同时执行。
- 步骤 3. 数据序列 00H、04H、0DH、09H、C3H 和 40H 必须分别写入寄存器 FD1L、FD1H、FD2L、FD2H、FD3L 和 FD3H。
- 步骤 4. 溢出周期为 300 μ s 的计数器将进行有效计时，此时允许用户将正确的数据序列写入 FD1L/FD1H~FD3L/FD3H 寄存器对。计数器时钟来自 LIRC。
- 步骤 5. 如果计数器溢出或数据序列不正确，Flash 存储器写操作不被使能且用户必须再次重复以上步骤。FWPEN 位将自动由硬件清零。
- 步骤 6. 如果计数器溢出前数据正确，Flash 程序存储器写操作将被使能且 FWPEN 位将自动由硬件清零。CFWEN 位由硬件置为“1”，表明 Flash 程序存储器写操作成功使能。
- 步骤 7. 一旦 Flash 程序存储器写操作使能，用户可通过 Flash 控制寄存器更改 Flash 程序存储器数据。
- 步骤 8. 用户可以清零 CFWEN 位来除能 Flash 程序存储器写操作。



Flash 程序存储器写功能使能步骤

Flash 程序存储器读 / 写步骤

通过前面的 IAP 步骤成功使能 Flash 程序存储器写功能后，用户必须先擦除相应的 Flash 存储页或存储块，然后启动 Flash 写操作。针对所选中的单片机型号，页擦除操作的数量分别是每页 32，64 和 128 个字，有效的页擦除地址由 FARH 寄存器和 FARL[7:5]，FARL[7:6] 以及 FARL[7] 位段的内容来指定。

擦除页	FARH	FARL[7:5]	FARL[4:0]
0	0000 0000	000	x xxxx
1	0000 0000	001	x xxxx
2	0000 0000	010	x xxxx
3	0000 0000	011	x xxxx
4	0000 0000	100	x xxxx
:	:	:	:
:	:	:	:
254	0001 1111	110	x xxxx
255	0001 1111	111	x xxxx

“x”：无关

HT66F2350 擦除页数量和选择

擦除页	FARH	FARL[7:6]	FARL[5:0]
0	0000 0000	00	XX XXXX
1	0000 0000	01	XX XXXX
2	0000 0000	10	XX XXXX
3	0000 0000	11	XX XXXX
4	0000 0001	00	XX XXXX
:	:	:	:
:	:	:	:
254	0011 1111	10	XX XXXX
255	0011 1111	11	XX XXXX

“x”：无关

HT66F2360 擦除页数量和选择

擦除页	FARH	FARL[7:6]	FARL[4:0]
0	0000 0000	00	XX XXXX
1	0000 0000	01	XX XXXX
2	0000 0000	10	XX XXXX
3	0000 0000	11	XX XXXX
4	0000 0001	00	XX XXXX
:	:	:	:
:	:	:	:
510	0111 1111	10	XX XXXX
511	0111 1111	11	XX XXXX

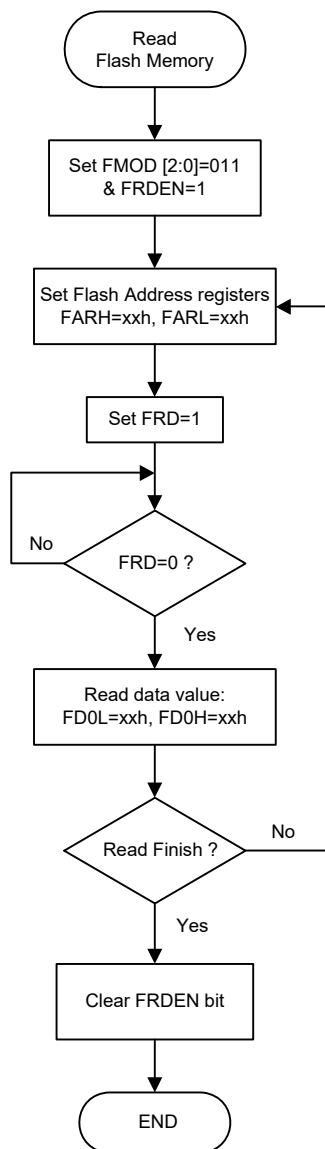
“x”：无关

HT66F2370 擦除页数量和选择

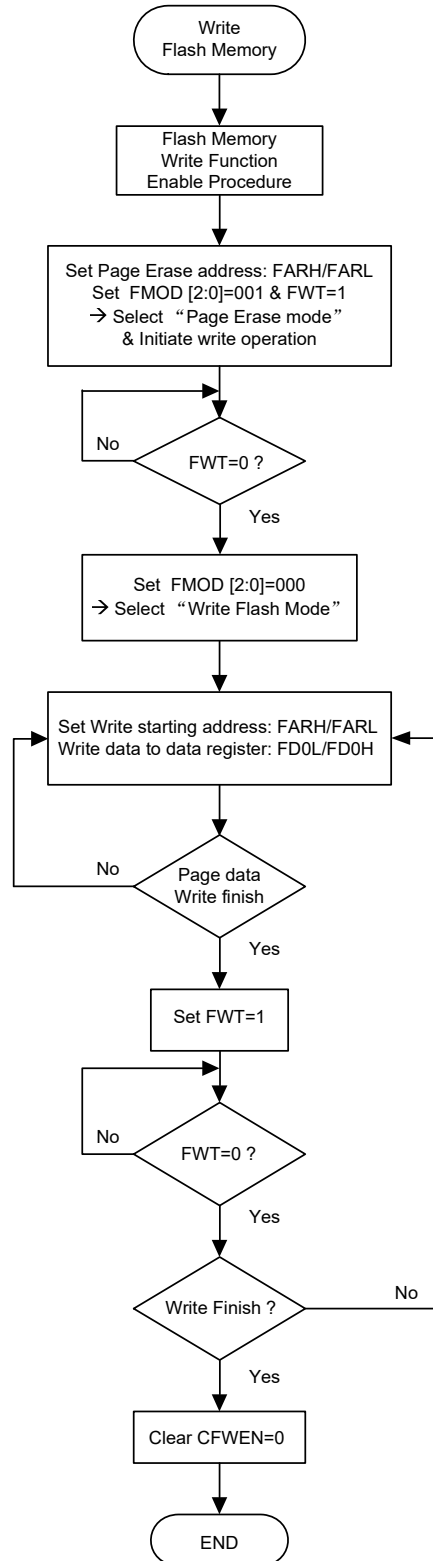
擦除页	FARH	FARL[7]	FARL[5:0]
0	0000 0000	0	XXX XXXX
1	0000 0000	1	XXX XXXX
2	0000 0001	0	XXX XXXX
3	0000 0001	1	XXX XXXX
4	0000 0010	0	XXX XXXX
:	:	:	:
:	:	:	:
510	1111 1111	0	XXX XXXX
511	1111 1111	1	XXX XXXX

“x”：无关

HT66F2390 擦除页数量和选择



读 Flash 程序存储器步骤



写 Flash 程序存储器步骤

注：当 FWT 或 FRD 位置为“1”，单片机停止工作。

数据存储

数据存储是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。

数据存储分为两个部分，第一部分是特殊功能数据存储。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。

切换不同的数据存储 Sector 可通过设置正确的存储器指针值实现。

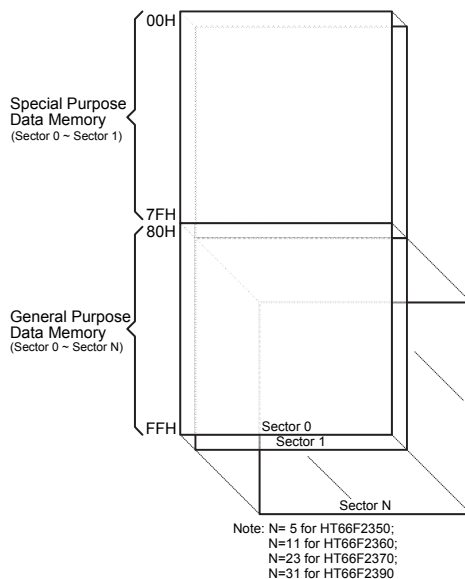
结构

数据存储被分为若干个 Sector，都可在 8-bit 的存储器中实现。每个数据存储 Sector 分为两种类型，即特殊功能数据存储和通用数据存储。

特殊功能数据寄存器地址范围为 00H~7FH 而通用数据存储地址范围为 80H~FFH。

单片机型号	特殊功能数据存储	通用数据存储	
	Sector	容量	Sector: 地址
HT66F2350	0, 1	768×8	0: 80H~FFH 1: 80H~FFH : 5: 80H~FFH
HT66F2360	0, 1	1536×8	0: 80H~FFH 1: 80H~FFH : 11: 80H~FFH
HT66F2370	0, 1	3072×8	0: 80H~FFH 1: 80H~FFH : 23: 80H~FFH
HT66F2390	0, 1	4096×8	0: 80H~FFH 1: 80H~FFH : 31: 80H~FFH

数据存储概要



数据存储器结构

数据存储器寻址

此系列单片机支持扩展指令架构，它并没有可用于数据存储器的存储区指针。存储区指针 PBP 仅适用于程序存储器。对于数据存储器所需的 Sector 是通过 MP1H 或 MP2H 寄存器指定，而所选 Sector 的某一数据存储器地址使用间接寻址访问方式时，通过 MP1L 或 MP2L 寄存器指定。

直接寻址可用于所有 Sector，通过相应的指令可以寻址所有可用的数据存储器空间。所访问的数据存储器位于除 Sector 0 外的任何数据存储器 Sector，扩展指令可代替间接寻址方式用来访问数据存储器。标准指令和扩展指令的主要区别在于扩展指令中的数据存储器地址“m”由 11~13 个有效位组成，具体有效位数取决于所选中的单片机型号，高字节表示 Sector，低字节表示指定的地址。

通用数据存储器

所有的单片机程序需要一个读/写的存储区，让临时数据可以被储存和再使用，该 RAM 区域就是通用数据存储器。使用者可对这个数据存储区进行读取和写入的操作。使用位操作指令可对个别的位做置位或复位的操作，较大地方便了用户在数据存储器内进行位操作。

特殊功能数据存储器

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

	Sector 0	Sector 1		Sector 0	Sector 1
00H	IAR0	CRCCR	40H	LVDC	EEC
01H	MP0	CRCIN	41H	EEA	
02H	IAR1	CRCDL	42H		
03H	MP1L	CRCDH	43H	EED	FC0
04H	MP1H		44H	CP0C	FC1
05H	ACC		45H	CP1C	FC2
06H	PCL	IECC	46H	PTM0C0	
07H	TBLP		47H	PTM0C1	
08H	TBLH		48H	PTM0DL	IFS0
09H	TBHP		49H	PTM0DH	IFS1
0AH	STATUS		4AH	PTM0AL	IFS2
0BH			4BH	PTM0AH	IFS3
0CH	IAR2		4CH	PTM0RPL	
0DH	MP2L		4DH	PTM0RPH	PAS0
0EH	MP2H		4EH	STM0C0	PAS1
0FH	RSTFC		4FH	STM0C1	PBS0
10H	INTC0	U0SR	50H	STM0DL	PBS1
11H	INTC1	U0CR1	51H	STM0DH	PCS0
12H	INTC2	U0CR2	52H	STM0AL	PCS1
13H	INTC3	TXR_RXR0	53H	STM0AH	PDS0
14H	PA	BRG0	54H	STM0RP	PDS1
15H	PAC	U1SR	55H	SLEDC0	PES0
16H	PAPU	U1CR1	56H	SLEDC1	PES1
17H	PAWU	U1CR2	57H	SLEDC2	PFS0
18H	PB	TXR_RXR1	58H		PFS1
19H	PBC	BRG1	59H	MDUWR0	
1AH	PBPU	PTM1C0	5AH	MDUWR1	
1BH	PC	PTM1C1	5BH	MDUWR2	
1CH	PCC	PTM1DL	5CH	MDUWR3	
1DH	PCPU	PTM1DH	5DH	MDUWR4	
1EH	PD	PTM1AL	5EH	MDUWR5	
1FH	PDC	PTM1AH	5FH	MDUWCTRL	
20H	PDPU	PTM1RPL	60H	CP0VOS	
21H	PE	PTM1RPH	61H	CP1VOS	
22H	PEC	PTM2C0	62H		
23H	PEPU	PTM2C1	63H	PSC0R	
24H	PF	PTM2DL	64H	TB0C	
25H	PFC	PTM2DH	65H	TB1C	
26H	PFPU	PTM2AL	66H	PSC1R	
27H		PTM2AH	67H	SADOL	
28H		PTM2RPL	68H	SADOH	
29H		PTM2RPH	69H	SADC0	
2AH		PTM3C0	6AH	SADC1	
2BH		PTM3C1	6BH	SADC2	
2CH		PTM3DL	6CH	SIMC0	
2DH	PMPS	PTM3DH	6DH	SIMC1	
2EH	RSTC	PTM3AL	6EH	SIMD	
2FH	VBGRC	PTM3AH	6FH	SIMA/SIMC2	
30H		PTM3RPL	70H	SIMTOC	
31H		PTM3RPH	71H	SPIAC0	
32H		STM1C0	72H	SPIAC1	
33H	MF10	STM1C1	73H	SPIAD	
34H	MF11	STM1DL	74H	FARL	
35H	MF12	STM1DH	75H	FARH	
36H	MF13	STM1AL	76H	FD0L	
37H	MF14	STM1AH	77H	FD0H	
38H	MF15	STM1RP	78H	FD1L	
39H	INTEG	STM2C0	79H	FD1H	
3AH	SCC	STM2C1	7AH	FD2L	
3BH	HIRCC	STM2DL	7BH	FD2H	
3CH	HXTC	STM2DH	7CH	FD3L	
3DH	LXTC	STM2AL	7DH	FD3H	
3EH	WDTC	STM2AH	7EH		
3FH	LVRC	STM2RP	7FH	SCOMC	

□ : Unused, read as 00H

特殊功能数据存储 - HT66F2350

	Sector 0	Sector 1		Sector 0	Sector 1
00H	IAR0	CRCCR	40H	LVDC	EEC
01H	MP0	CRCIN	41H	EEA	
02H	IAR1	CRCDL	42H		
03H	MP1L	CRCDH	43H	EED	FC0
04H	MP1H		44H	CP0C	FC1
05H	ACC		45H	CP1C	FC2
06H	PCL	IECC	46H	PTM0C0	
07H	TBLP		47H	PTM0C1	
08H	TBLH		48H	PTM0DL	IFS0
09H	TBHP		49H	PTM0DH	IFS1
0AH	STATUS		4AH	PTM0AL	IFS2
0BH	PBP		4BH	PTM0AH	IFS3
0CH	IAR2		4CH	PTM0RPL	
0DH	MP2L		4DH	PTM0RPH	PAS0
0EH	MP2H		4EH	STM0C0	PAS1
0FH	RSTFC		4FH	STM0C1	PBS0
10H	INTC0	U0SR	50H	STM0DL	PBS1
11H	INTC1	U0CR1	51H	STM0DH	PCS0
12H	INTC2	U0CR2	52H	STM0AL	PCS1
13H	INTC3	TXR_RXR0	53H	STM0AH	PDS0
14H	PA	BRG0	54H	STM0RP	PDS1
15H	PAC	U1SR	55H	SLEDC0	PES0
16H	PAPU	U1CR1	56H	SLEDC1	PES1
17H	PAWU	U1CR2	57H	SLEDC2	PFS0
18H	PB	TXR_RXR1	58H	SLEDC3	PFS1
19H	PBC	BRG1	59H	MDUWR0	PGS0
1AH	PBPU	PTM1C0	5AH	MDUWR1	PGS1
1BH	PC	PTM1C1	5BH	MDUWR2	PHS0
1CH	PCC	PTM1DL	5CH	MDUWR3	PHS1
1DH	PCPU	PTM1DH	5DH	MDUWR4	
1EH	PD	PTM1AL	5EH	MDUWR5	
1FH	PDC	PTM1AH	5FH	MDUWCTRL	
20H	PDPU	PTM1RPL	60H	CP0VOS	
21H	PE	PTM1RPH	61H	CP1VOS	
22H	PEC	PTM2C0	62H		
23H	PEPU	PTM2C1	63H	PSC0R	
24H	PF	PTM2DL	64H	TB0C	
25H	PFC	PTM2DH	65H	TB1C	
26H	PFPU	PTM2AL	66H	PSC1R	
27H	PG	PTM2AH	67H	SADOL	
28H	PGC	PTM2RPL	68H	SADOH	
29H	PGPU	PTM2RPH	69H	SADC0	
2AH	PH	PTM3C0	6AH	SADC1	
2BH	PHC	PTM3C1	6BH	SADC2	
2CH	PHPU	PTM3DL	6CH	SIMC0	
2DH	PMPS	PTM3DH	6DH	SIMC1	
2EH	RSTC	PTM3AL	6EH	SIMD	
2FH	VBGRC	PTM3AH	6FH	SIMA/SIMC2	
30H		PTM3RPL	70H	SIMTOC	
31H		PTM3RPH	71H	SPIAC0	
32H		STM1C0	72H	SPIAC1	
33H	MF10	STM1C1	73H	SPIAD	
34H	MF11	STM1DL	74H	FARL	
35H	MF12	STM1DH	75H	FARH	
36H	MF13	STM1AL	76H	FD0L	
37H	MF14	STM1AH	77H	FD0H	
38H	MF15	STM1RP	78H	FD1L	
39H	INTEG	STM2C0	79H	FD1H	
3AH	SCC	STM2C1	7AH	FD2L	
3BH	HIRCC	STM2DL	7BH	FD2H	
3CH	HXTC	STM2DH	7CH	FD3L	
3DH	LXTC	STM2AL	7DH	FD3H	
3EH	WDTC	STM2AH	7EH		
3FH	LVRC	STM2RP	7FH	SCOMC	

□ : Unused, read as 00H

特殊功能数据存储区 – HT66F2360

	Sector 0	Sector 1		Sector 0	Sector 1
00H	IAR0	CRCCR	40H	LVDC	EEC
01H	MP0	CRCIN	41H	EEAL	
02H	IAR1	CRCDL	42H	EEAH	
03H	MP1L	CRCDH	43H	EED	FC0
04H	MP1H		44H	CP0C	FC1
05H	ACC		45H	CP1C	FC2
06H	PCL	IECC	46H	PTM0C0	
07H	TBLP		47H	PTM0C1	
08H	TBLH		48H	PTM0DL	IFS0
09H	TBHP		49H	PTM0DH	IFS1
0AH	STATUS		4AH	PTM0AL	IFS2
0BH	PBP	U2SR	4BH	PTM0AH	IFS3
0CH	IAR2	U2CR1	4CH	PTM0RPL	
0DH	MP2L	U2CR2	4DH	PTM0RPH	PAS0
0EH	MP2H	TXR_RXR2	4EH	STM0C0	PAS1
0FH	RSTFC	BRG2	4FH	STM0C1	PBS0
10H	INTC0	U0SR	50H	STM0DL	PBS1
11H	INTC1	U0CR1	51H	STM0DH	PCS0
12H	INTC2	U0CR2	52H	STM0AL	PCS1
13H	INTC3	TXR_RXR0	53H	STM0AH	PDS0
14H	PA	BRG0	54H	STM0RP	PDS1
15H	PAC	U1SR	55H	SLEDC0	PES0
16H	PAPU	U1CR1	56H	SLEDC1	PES1
17H	PAWU	U1CR2	57H	SLEDC2	PFS0
18H	PB	TXR_RXR1	58H	SLEDC3	PFS1
19H	PBC	BRG1	59H	MDUWR0	PGS0
1AH	PBPU	PTM1C0	5AH	MDUWR1	PGS1
1BH	PC	PTM1C1	5BH	MDUWR2	PHS0
1CH	PCC	PTM1DL	5CH	MDUWR3	PHS1
1DH	PCPU	PTM1DH	5DH	MDUWR4	
1EH	PD	PTM1AL	5EH	MDUWR5	
1FH	PDC	PTM1AH	5FH	MDUWCTRL	
20H	PDPU	PTM1RPL	60H	CP0VOS	
21H	PE	PTM1RPH	61H	CP1VOS	
22H	PEC	PTM2C0	62H		
23H	PEPU	PTM2C1	63H	PSC0R	
24H	PF	PTM2DL	64H	TBOC	
25H	PFC	PTM2DH	65H	TB1C	
26H	PFPU	PTM2AL	66H	PSC1R	
27H	PG	PTM2AH	67H	SADOL	
28H	PGC	PTM2RPL	68H	SADOH	
29H	PGPU	PTM2RPH	69H	SADC0	
2AH	PH	PTM3C0	6AH	SADC1	
2BH	PHC	PTM3C1	6BH	SADC2	
2CH	PHPU	PTM3DL	6CH	SIMC0	
2DH	PMPS	PTM3DH	6DH	SIMC1	
2EH	RSTC	PTM3AL	6EH	SIMD	
2FH	VBGRC	PTM3AH	6FH	SIMA/SIMC2	
30H		PTM3RPL	70H	SIMTOC	
31H		PTM3RPH	71H	SPIAC0	
32H		STM1C0	72H	SPIAC1	
33H	MF10	STM1C1	73H	SPIAD	
34H	MF11	STM1DL	74H	FARL	
35H	MF12	STM1DH	75H	FARH	
36H	MF13	STM1AL	76H	FD0L	
37H	MF14	STM1AH	77H	FD0H	
38H	MF15	STM1RP	78H	FD1L	
39H	INTEG	STM2C0	79H	FD1H	
3AH	SCC	STM2C1	7AH	FD2L	
3BH	HIRCC	STM2DL	7BH	FD2H	
3CH	HXTC	STM2DH	7CH	FD3L	
3DH	LXTC	STM2AL	7DH	FD3H	
3EH	WDTC	STM2AH	7EH		
3FH	LVRC	STM2RP	7FH	SCOMC	

□: Unused, read as 00H

特殊功能数据存储 - HT66F2370/HT66F2390

特殊功能寄存器

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

间接寻址寄存器 – IAR0, IAR1, IAR2

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

存储器指针 – MP0, MP1L/MP1H, MP2L/MP2H

该系列单片机提供五个存储器指针，即 MP0、MP1L、MP1H、MP2L 和 MP2H。由于这些指针在数据存储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。当对相关间接寻址寄存器进行任何操作时，单片机指向的实际地址是由存储器指针 MP0 所指定的地址，此时间接寻址寄存器 IAR0 用于访问 Sector0 中的数据，而 MP1L/MP1H 和 IAR1、MP2L/MP2H 和 IAR2 可根据 MP1H 或 MP2H 寄存器访问所有的 Sector。直接寻址通过相关的数据存储器寻址指令来访问所有可用数据存储空间。

间接寻址程序范例

● 范例 1

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

● 范例 2

```

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

```

需注意，范例中并没有确定 RAM 地址。

使用扩展指令直接寻址程序举例

```

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

```

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

程序存储区指针 – PBP

除了 HT66F2350 外，该系列单片机的程序存储器被分为几个 Bank，可以通过设置程序存储区指针 PBP 来访问不同的程序存储区。PBP 寄存器应在单片机使用“JMP”或“CALL”指令执行“分支”操作前正确地配置。在分支指令执行后会跳转到一个非连续的程序存储器地址，此地址位于程序存储区指针所选 Bank 内。

● PBP 寄存器 – HT66F2360

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	PBP0
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 7~1 **D7~D1**: 通用数据位，可读写。

Bit 0 **PBP0**: 程序存储区指针 bit 0

0: Bank 0

1: Bank 1

● PBP 寄存器 – HT66F2370

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	PBP1	PBP0
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 7~2 **D7~D2**: 通用数据位，可读写。

Bit 1~0 **PBP1~PBP0**: 程序存储区指针 bit 1~bit 0

00: Bank 0

01: Bank 1

10: Bank 2

11: Bank 3

● PBP 寄存器 – HT66F2390

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	PBP2	PBP1	PBP0
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 7~3 **D7~D3**: 通用数据位，可读写。

Bit 2~0 **PBP2~PBP0**: 程序存储区指针 bit 2~bit 0

000: Bank 0

001: Bank 1

010: Bank 2

011: Bank 3

100: Bank 4

101: Bank 5

110: Bank 6

111: Bank 7

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-bit 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

该 8-bit 的状态寄存器由零标志位 (Z)、进位标志位 (C)、辅助进位标志位 (AC)、溢出标志位 (OV)、SC 标志位、CZ 标志位、暂停标志位 (PDF) 和看门狗定时器溢出标志位 (TO) 组成。这些算术 / 逻辑操作和系统运行标志位是用来记录单片机的运行状态。

除了 TO 和 PDF 标志外，状态寄存器中的位像其它大部分寄存器一样可以被改变。任何数据写入到状态寄存器将不会改变 TO 或 PDF 标志位。另外，执行不同的指令后，与状态寄存器有关的运算可能会得到不同的结果。TO 标志位只会受系统上电、看门狗溢出或执行“CLR WDT”或“HALT”指令影响。PDF 标志位只会受执行“HALT”或“CLR WDT”指令或系统上电影响。

Z、OV、AC、C、SC 和 CZ 标志位通常反映最近运算的状态。

- C: 当加法运算的结果产生进位，或减法运算的结果没有产生借位时，则 C 被置位，否则 C 被清零。
- 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。
- SC: 当 OV 与当前指令操作结果 MSB 执行“XOR”所得结果。
- CZ: 不同指令不同标志位的操作结果。详细资料请参考寄存器定义部分。

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

● STATUS 寄存器

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

“x”：未知

- Bit 7 **SC**: 当 OV 与当前指令操作结果 MSB 执行“XOR”所得结果
- Bit 6 **CZ**: 不同指令不同标志位的操作结果。
 对于 SUB/SUBM/LSUB/LSUBM 指令，CZ 等于 Z 标志位。
 对于 SBC/SBCM/LSBC/LSBCM 指令，CZ 等于上一个 CZ 标志位与当前零标志位执行“AND”所得结果。对于其它指令，CZ 标志位无影响。
- Bit 5 **TO**: 看门狗溢出标志位
 0: 系统上电或执行“CLR WDT”或“HALT”指令后
 1: 看门狗溢出发生
- Bit 4 **PDF**: 暂停标志位
 0: 系统上电或执行“CLR WDT”指令后
 1: 执行“HALT”指令
- Bit 3 **OV**: 溢出标志位
 0: 无溢出
 1: 运算结果高两位的进位状态异或结果为 1
- Bit 2 **Z**: 零标志位
 0: 算术或逻辑运算结果不为 0
 1: 算术或逻辑运算结果为 0
- Bit 1 **AC**: 辅助进位标志位
 0: 无辅助进位
 1: 在加法运算中低四位产生了向高四位进位，或减法运算中低四位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 数据存储

此系列单片机的一个特性是内建 EEPROM 数据存储。 “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。

单片机型号	容量	地址
HT66F2350	256 × 8	00H ~ FFH
HT66F2360		
HT66F2370	512 × 8	000H ~ 1FFH
HT66F2390	1024 × 8	000H ~ 3FFH

EEPROM 数据存储结构

该系列单片机的 EEPROM 数据存储容量高达 1024×8 位。由于映射方式与程序存储器和数据存储器不同，因此不能像其它类型的存储器一样寻址。使用 Sector 0 中的一个地址和数据寄存器以及 Sector 1 中的一个控制寄存器，可以实现对 EEPROM 的单字节读写操作。

EEPROM 寄存器

有三个寄存器控制内部 EEPROM 数据存储总的操作，地址寄存器 EEA、数据寄存器 EED 及控制寄存器 EEC。EEA 和 EED 位于 Sector 0 中，它们能像其它特殊功能寄存器一样直接被访问。EEC 位于 Sector 1 中，不能被直接访问，仅能通过 MP1L/MP1H 和 IAR1 或 MP2L/MP2H 和 IAR2 进行间接读取或写入。由于 EEC 控制寄存器位于 Sector 1 中的“40H”，在 EEC 寄存器上的任何操作被执行前，MP1L 或 MP2L 必须先设为“40H”，MP1H 或 MP2H 被设为“01H”。

寄存器名称	位							
	7	6	5	4	3	2	1	0
EEA(HT66F2350/60)	EEA7	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EEAL(HT66F2370/90)	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
EEAH(HT66F2370)	—	—	—	—	—	—	—	EEAH0
EEAH(HT66F2390)	—	—	—	—	—	—	EEAH1	EEAH0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM 控制寄存器列表

- EEA 寄存器 – HT66F2350/HT66F2360

Bit	7	6	5	4	3	2	1	0
Name	EEA7	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
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 EEA7~EEA0: 数据 EEPROM 地址 bit 7~bit 0

● **EEAL 寄存器 – HT66F2370/HT66F2390**

Bit	7	6	5	4	3	2	1	0
Name	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **EEAL7~EEAL0**: 数据 EEPROM 低字节地址 bit 7~bit 0

● **EEAH 寄存器 – HT66F2370**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	EEAH0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 未定义, 读为“0”

Bit 0 **EEAH0**: 数据 EEPROM 高字节地址 bit 0

● **EEAH 寄存器 – HT66F2390**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	EEAH1	EEAH0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义, 读为“0”

Bit 1~0 **EEAH1~EEAH0**: 数据 EEPROM 高字节地址 bit 1~bit 0

● **EED 寄存器**

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

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

● **EEC 寄存器**

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

Bit 7~4 未定义, 读为“0”

Bit 3 **WREN**: 数据 EEPROM 写使能位

0: 除能

1: 使能

此位为数据 EEPROM 写使能位, 向数据 EEPROM 写操作之前需将此位置高。将此位清零时, 则禁止向数据 EEPROM 写操作。

Bit 2 **WR**: EEPROM 写控制位

0: 写周期结束

1: 写周期有效

此位为数据 EEPROM 写控制位, 由应用程序将此位置高将激活写周期。写周期结束后, 硬件自动将此位清零。当 WREN 未先置高时, 此位置高无效。

- Bit 1 **RDEN**: 数据 EEPROM 读使能位
 0: 除能
 1: 使能
 此位为数据 EEPROM 读使能位, 向数据 EEPROM 读操作之前需将此位置高。将此位清零时, 则禁止向数据 EEPROM 读操作。
- Bit 0 **RD**: EEPROM 读控制位
 0: 读周期结束
 1: 读周期有效
 此位为数据 EEPROM 读控制位, 由应用程序将此位置高将激活读周期。读周期结束后, 硬件自动将此位清零。当 RDEN 未首先置高时, 此位置高无效。
- 注: 在同一条指令中 WREN、WR、RDEN 和 RD 不能同时置为“1”。WR 和 RD 不能同时置为“1”。

从 EEPROM 中读取数据

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

写保护

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

EEPROM 中断

EEPROM 写周期结束后将产生 EEPROM 写中断。EEPROM 中断需先通过设置相关中断寄存器的 DEE 位使能 EEPROM 中断。但由于 EEPROM 中断属于多功能中断, 其相关的多功能中断使能位也需被置位。当 EEPROM 写周期结束, DEF 请求标志位将被置位。若总中断, EEPROM 中断和相关的多功能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, 仅多功能中断标志位会被自动复位, EEPROM 中断标志位需通过应用程序手动复位。

编程注意事项

必须注意的是数据不会无意写入 EEPROM。在没有写动作时写使能位被正常清零可以增强保护功能。存储器指针高字节寄存器也可以正常清零以阻止进入 EEPROM 控制寄存器存在的 Sector 1。尽管没有必要，写一个简单的读回程序以检查新写入的数据是否正确还是应该考虑的。WREN 位置位后，EEC 寄存器中的 WR 位需立即置位，以确保写周期正确地执行。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，单片机不应在 EEPROM 读或写操作完全完成之前进入空闲或休眠模式，否则 EEPROM 读或写操作将失败。

程序范例 – HT66F2350

● 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 low byte MP1L
MOV MP1L, A                   ; MP1L points to EEC register
MOV A, 01H                    ; setup Memory Pointer high byte MP1H
MOV MP1H, A
SET IAR1.1                    ; set RDEN bit, enable read operations
SET IAR1.0                    ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                     ; check for read cycle end
JMP BACK
CLR IAR1                      ; disable EEPROM write/read
CLR MP1H
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 low byte MP1L
MOV MP1L, A                   ; MP1L points to EEC register
MOV A, 01H                    ; setup Memory Pointer high byte MP1H
MOV MP1H, A
CLR EMI
SET IAR1.3                    ; set WREN bit, enable write operations
SET IAR1.2                    ; start Write Cycle - set WR bit
SET EMI
BACK:
SZ IAR1.2                     ; check for write cycle end
JMP BACK
CLR IAR1                      ; disable EEPROM write
CLR MP1H
```

振荡器

不同的振荡器选择可以让使用者在不同的应用需求中实现更大范围的功能。振荡器的灵活性使得在速度和功耗方面可以达到较佳的优化。振荡器选择是通过应用程序和相关控制寄存器寄存器共同完成的。

振荡器概述

振荡器除了作为系统时钟源，还作为看门狗定时器、时基功能和定时器模块的时钟源。外部振荡器需要一些外围器件，而集成的内部振荡器不需要任何外围器件。它们提供的高速和低速系统振荡器具有较宽的频率范围。振荡器类型通过寄存器选择。较高频率的振荡器提供更高的性能，但要求有更高的功率，反之亦然。动态切换快慢系统时钟的能力使单片机具有灵活而优化的性能 / 功耗比，此特性对功耗敏感的应用领域尤为重要。

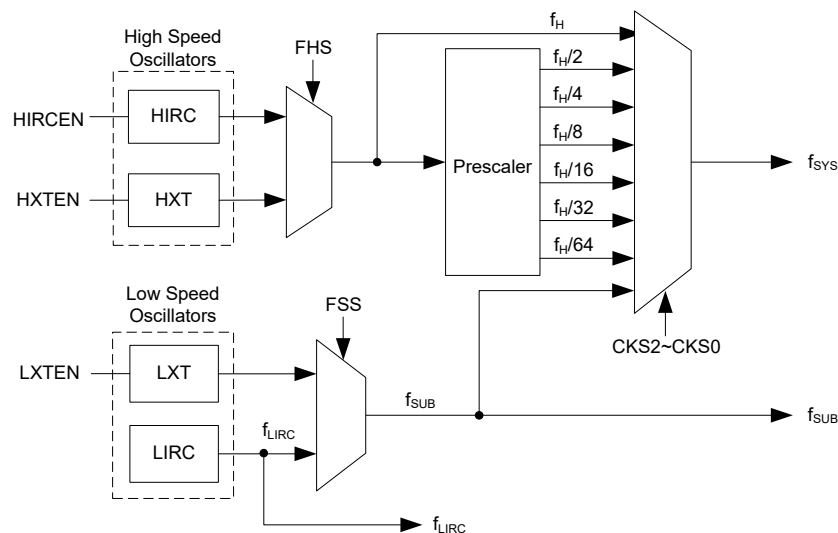
类型	名称	频率	引脚
外部高速晶振	HXT	400kHz~16MHz	OSC1/OSC2
内部高速 RC	HIRC	8/12/16MHz	—
外部低速晶振	LXT	32.768kHz	XT1/XT2
内部低速 RC	LIRC	32kHz	—

振荡器类型

系统时钟配置

此系列单片机有四个系统振荡器，包括两个高速振荡器和两个低速振荡器。高速振荡器有外部晶体 / 陶瓷振荡器 HXT 和内部 8/12/16MHz RC 振荡器 HIRC，低速振荡器有内部 32kHz 低速振荡器 LIRC 和外部 32.768kHz 晶振 LXT。使用高速或低速振荡器作为系统时钟的选择是通过设置 SCC 寄存器中的 CKS2~CKS0 位决定的，系统时钟可动态选择。

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

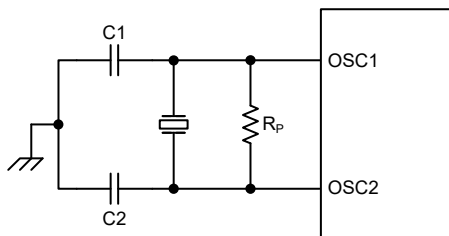


系统时钟配置

外部高频晶体振荡器 – HXT

外部高频晶体 / 陶瓷振荡器是一个高频振荡器，为上电后的默认振荡器时钟源。对于晶体振荡器，只要简单地将晶体连接至 OSC1 和 OSC2，则会产生振荡所需的相移及反馈，而不需其它外部电容。为保证某些低频率的晶体振荡和陶瓷谐振器的振荡频率更精准，建议连接两个小容量电容 C1 和 C2 到 VSS，具体数值与客户选择的晶体 / 陶瓷晶振有关。

为了确保振荡器的稳定性及减少噪声和串扰的影响，晶体振荡器及其相关的电阻和电容以及它们之间的连线都应尽可能的接近单片机。



Note: 1. R_p is normally not required. C1 and C2 are required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

晶体 / 陶瓷振荡器 – HXT

HXT 振荡器 C1 和 C2 值		
晶振频率	C1	C2
12MHz	0pF	0pF
8MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	100pF	100pF

注：C1 和 C2 数值仅作参考

晶体振荡器电容推荐值

内部 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，不需其它外部器件。内部 RC 振荡器具有三种固定的频率：8MHz，12MHz 或 16MHz。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡频率因电源电压、温度以及芯片制成工艺不同的影响较大程度地降低。因此，当电源为 3V/5V 且温度为 25°C 时，所选中的调整后的振荡器频率误差将在 1% 以内。应注意若系统时钟选中选择该内部振荡器，则无需外部引脚，此时的引脚将用作正常 I/O 功能或其它引脚共用功能。

外部 32.768kHz 晶体振荡器 – LXT

外部 32.768kHz 晶体振荡器是一个低频振荡器，经由软件控制位 FSS 选择。时钟频率固定为 32.768kHz，此时 XT1 和 XT2 间引脚必须连接 32.768kHz 的晶体振荡器，需要外部电阻和电容连接到 32.768kHz 晶振以帮助起振。对于那些要求精确频率的场合中，可能需要这些元件来对由制程产生的误差提供频率补偿。在置位 LXTEN 使能 LXT 振荡器后，LXT 振荡器启动需要一定的延时。

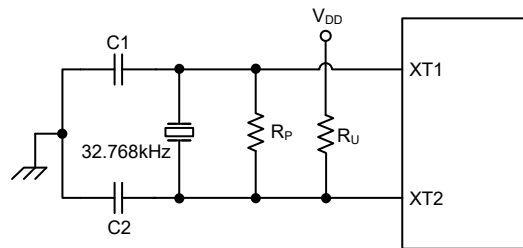
当系统进入空闲 / 休眠模式，系统时钟关闭以降低功耗。然而在某些应用，比如空闲 / 休眠模式下要保持内部定时器功能，必须提供额外的时钟，且与系统时钟无关。

然而，对于一些晶体，为了保证系统频率的启动与精度要求，需要外接两个小容量电容 C1 和 C2，具体数值与客户选择的晶体规格有关。外部并联的反馈电阻 R_P 和上拉电阻 R_U 是必需的。

引脚共用软件控制位决定 XT1/XT2 脚是用于 LXT 还是作为普通 I/O 口或者其它共用功能使用。

- 若 LXT 振荡器未被用于任何时钟源，XT1/XT2 脚能被用作一般 I/O 口或者其它共用功能使用。
- 若 LXT 振荡器被用于一些时钟源，32.768kHz 晶体应被连接至 XT1/XT2 脚。

为了确保振荡器的稳定性及减少噪声和串扰的影响，晶体振荡器及其相关的电阻和电容以及它们之间的连线都应尽可能地接近单片机。



Note: 1. R_P , R_U , C1 and C2 are required.
2. Although not shown XT1/XT2 pins have a parasitic capacitance of around 7pF.

外部 LXT 振荡器

LXT 振荡器 C1 和 C2 值		
晶体频率	C1	C2
32.768kHz	10pF	22~24pF

注：1. C1 和 C2 数值仅作参考。
2. R_P 的建议值为 $5M\Omega\sim 10M\Omega$ 。
3. R_U 的建议值为 $5.1M\Omega$ 。

32.768kHz 晶体电容推荐值

内部 32kHz 振荡器 – LIRC

内部 32kHz 系统振荡器是一个低频振荡器，经由软件控制位 FSS 选择。该系列单片机具有完全集成 RC 振荡器，它在 5V 电压下运行的典型频率值为 32kHz 且无需外部元件。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响较大程度地降低。因此，在电源为 5V 且温度为 25°C 时，32kHz 的固定振荡频率误差将在 2% 以内。

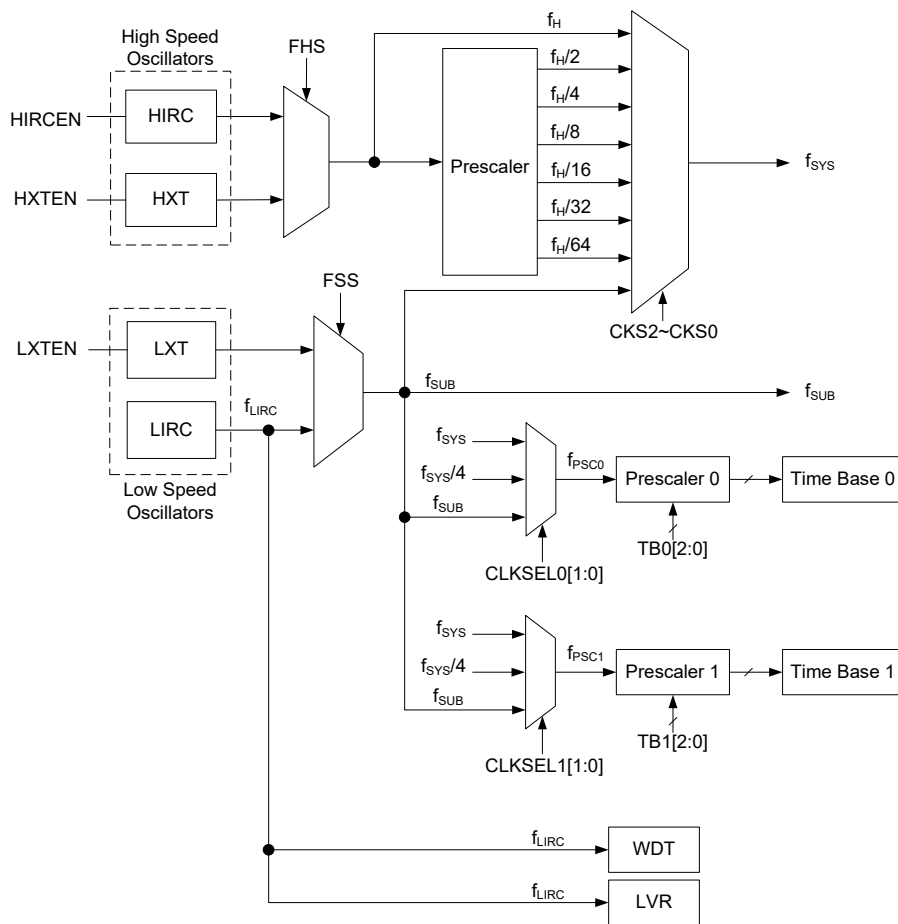
工作模式和系统时钟

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。此系列单片机提供高、低速两种时钟源，它们之间可以动态切换，用户可通过优化单片机操作来获得较佳性能 / 功耗比。

系统时钟

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

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



单片机时钟选项

注：当系统时钟源 f_{SYS} 由 f_H 到 f_{SUB} 转换时，高速振荡器可通过程序设置关闭将停止以节省耗电，或通过设置对应高频振荡器使能控制位继续为外围电路提供 $f_H \sim f_H/64$ 的频率。

系统工作模式

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

工作模式	CPU	寄存器设定			f _{sys}	f _H	f _{sub}	f _{LIRC}
		FHIDEN	FSIDEN	CKS2~CKS0				
快速模式	On	x	x	000~110	f _H ~f _H /64	On	On	On
低速模式	On	x	x	111	f _{sub}	On/Off ⁽¹⁾	On	On
空闲模式 0	Off	0	1	000~110	Off	Off	On	On
				111	On			
空闲模式 1	Off	1	1	xxx	On	On	On	On
空闲模式 2	Off	1	0	000~110	On	On	Off	On
				111	Off			
休眠模式	Off	0	0	xxx	Off	Off	Off	On ⁽²⁾

注：1. 在低速模式下，f_H 时钟开启或关闭由对应高频振荡器使能位控制。

2. 若 WDT 功能使能则 f_{LIRC} 时钟将开启。

快速模式

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

低速模式

此模式的系统时钟虽为较低速时钟源，但单片机仍能正常工作。该低速时钟源来自 f_{sub}。该时钟来源于 LIRC 或 LXT 振荡器。

休眠模式

在 HALT 指令执行后且 FHIDEN 和 FSIDEN 位为低时，系统进入休眠模式。在休眠模式中，CPU 停止运行且高速和低速振荡器都将被关闭，若看门狗定时器功能由 WDTC 寄存器设定为使能，则 f_{LIRC} 时钟将继续运行。

空闲模式 0

执行 HALT 指令后且 SCC 寄存器中 FHIDEN 位为低，FSIDEN 位为高时，系统进入空闲模式 0。在空闲模式 0 中，CPU 将被关闭但低速振荡器将开启以驱动一些外围功能。

空闲模式 1

执行 HALT 指令后且 SCC 寄存器中 FHIDEN 位为高，FSIDEN 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以驱动一些外围功能，该系统振荡器可以为高速或低速系统振荡器。

空闲模式 2

执行 HALT 指令后且 SCC 寄存器中 FHIDEN 位为低，FSIDEN 位为低时，系统进入空闲模式 2。在空闲模式 2 中，CPU 和低速振荡器将被关闭，但高速振荡器将开启以驱动一些外围功能。

控制寄存器

寄存器 SCC，HIRCC，HXTC 和 LXTC 用于控制单片机内部时钟。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	FHS	FSS	FHIDEN	FSIDEN
HIRCC	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
HXTC	—	—	—	—	—	HXTM	HXTF	HXTEN
LXTC	—	—	—	—	—	—	LXTF	LXTEN

系统工作模式控制寄存器列表

• SCC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	FHS	FSS	FHIDEN	FSIDEN
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 7~5 **CKS2~CKS0**: 系统时钟选择

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

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

Bit 4 未定义，读为“0”

Bit 3 **FHS**: 高频时钟选择位

0: HIRC
1: HXT

Bit 2 **FSS**: 低频时钟选择

0: LIRC
1: LXT

Bit 1 **FHIDEN**: CPU 关闭时的高频振荡器控制位

0: 除能
1: 使能

此位用来控制在执行 HALT 指令 CPU 关闭后高速振荡器是否停止。

Bit 0 **FSIDEN**: CPU 关闭时低频振荡器控制位

0: 除能
1: 使能

此位用来控制在执行 HALT 指令 CPU 关闭后低速振荡器是否停止。若选择 LIRC 作为低速振荡器时钟源，则 LIRC 振荡器是由该位与 WDT 功能控制位共同控制的。若 WDT 功能使能，即使该位被清零，LIRC 振荡器也会继续运行。

● HIRCC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	1

Bit 7~4 未定义，读为“0”

Bit 3~2 **HIRC1~HIRC0**: HIRC 频率选择位

00: 8MHz
01: 12MHz
10: 16MHz
11: 8MHz

当 HIRC 振荡器使能或通过应用程序改变 HIRC 频率选择位时，在 HIRCF 标志位置高后时钟频率会自动改变。

Bit 1 **HIRCF**: HIRC 振荡器稳定标志位

0: HIRC 不稳定
1: HIRC 稳定

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

Bit 0 **HIRCEN**: HIRC 振荡器使能控制位

0: 除能
1: 使能

● HXTC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	HXTM	HXTF	HXTEN
R/W	—	—	—	—	—	R/W	R	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 未定义，读为“0”

Bit 2 **HXTM**: HXT 模式选择位

0: HXT 频率 ≤10MHz
1: HXT 频率 >10MHz

此位用于选择 HXT 振荡器的工作模式。注意，此位必须在 HXT 使能前正确地配置，在 HXTEN 位置高使能 HXT 振荡器后改变此位的值将无效。

Bit 1 **HXTF**: HXT 振荡器稳定标志位

0: HXT 不稳定
1: HXT 稳定

此位用于表明 HXT 振荡器是否稳定。HXTEN 位置高使能 HXT 振荡器后，HXTF 位会先被清零，在 HXT 稳定后会被置高。

Bit 0 **HXTEN**: HXT 振荡器使能控制位

0: 除能
1: 使能

• LXTC 寄存器

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

Bit 7~2 未定义，读为“0”

Bit 1 **LXTF**: LXT 振荡器稳定标志位

0: LXT 不稳定

1: LXT 稳定

此位用于表明 LXT 振荡器是否稳定。LXTEN 位置高使能 LXT 振荡器后，LXTF 位会先被清零，在 LXT 稳定后会被置高。

Bit 0 **LXTEN**: LXT 振荡器使能控制位

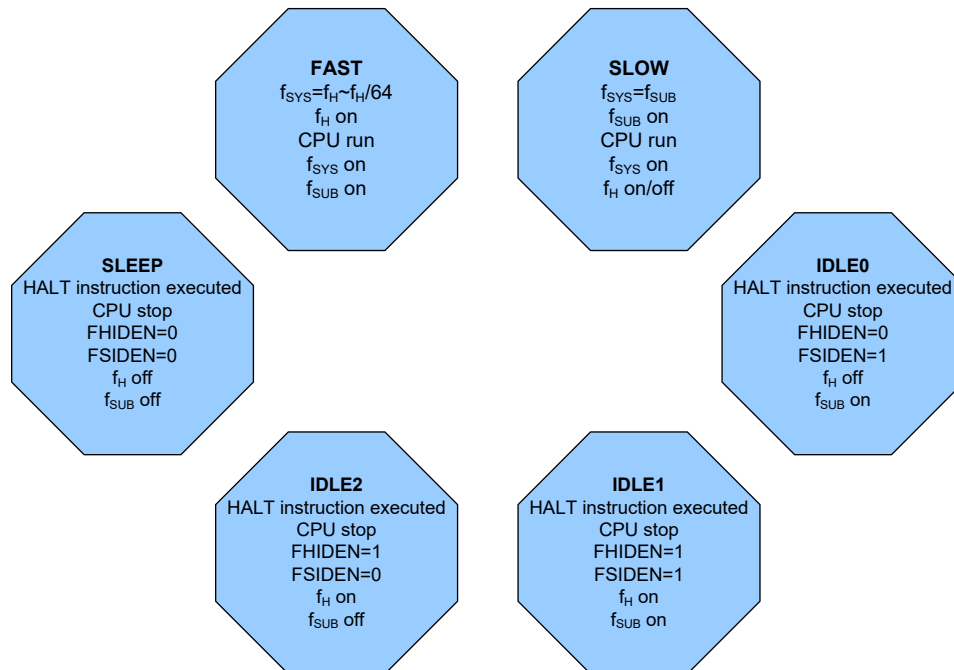
0: 除能

1: 使能

工作模式转换

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

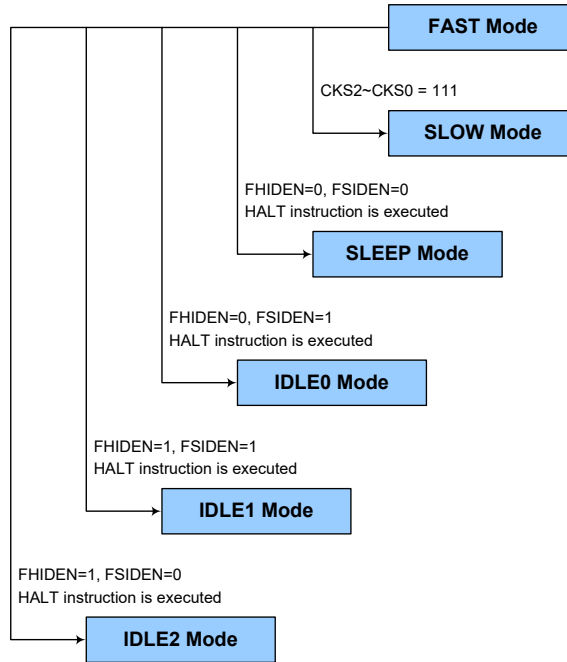
简单来说，快速模式和低速模式间的切换仅需设置 SCC 寄存器中的 CKS2~CKS0 位即可实现，而快速模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SCC 寄存器中的 FHIDEN 和 FSIDEN 位决定的。



快速模式切换到低速模式

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

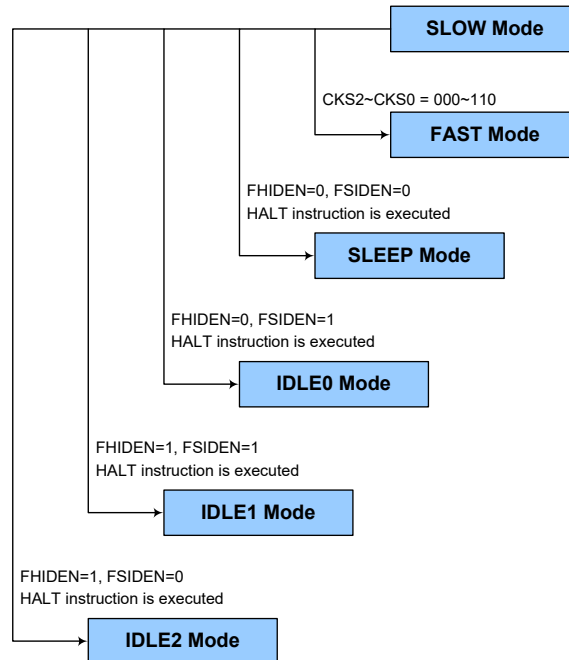
低速模式的时钟源来自 LXT 或 LIRC 振荡器，由 SCC 寄存器中的 FSS 位确定，因此要求这些振荡器在所有模式切换动作发生前稳定下来。



低速模式切换到快速模式

在低速模式时系统时钟来自 f_{SUB} 。切换回快速模式时，需设置 $CKS2\sim CKS0$ 位为“000”~“110”使系统时钟从 f_{SUB} 切换到 $f_H\sim f_H/64$ 。

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



进入休眠模式

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

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

进入空闲模式 0

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

- f_H 时钟停止运行，应用程序停止在“HALT”指令处， f_{SUB} 时钟将继续运行。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 若 WDT 功能由 WDTC 寄存器使能，则 WDT 将被清零并重新开始计数。

进入空闲模式 1

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

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

进入空闲模式 2

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

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

待机电流注意事项

由于单片机进入休眠或空闲模式的主要原因是将MCU的电流降低到尽可能低，可能到只有几个微安的级别（空闲模式1和空闲模式2除外），所以如果要将电路的电流进一步降低，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。

另外应注意单片机设为输出的I/O引脚上的负载。应将它们设置在有最小拉电流的状态或将它们和其它的CMOS输入一样接到没有拉电流的外部电路上。还需注意的是LIRC振荡器的使能也会消耗额外的待机电流。

在空闲模式1和空闲模式2中高速振荡器开启，若外围功能时钟源来自高速系统振荡器，额外的待机电流也可能会有几百微安。

唤醒

单片机进入休眠模式或空闲模式后，系统时钟将停止以降低功耗。然而当单片机再次唤醒，原来的系统时钟重新起振、稳定且恢复正常工作需要一定的时间。系统进入休眠或空闲模式之后，可以通过以下几种方式唤醒：

- PA 口下降沿
- 外部复位
- 系统中断
- WDT 溢出

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

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

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源来自于内部时钟 f_{LIRC} 。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} 、温度和制程的不同而变化。看门狗定时器的时钟源可分频为 $2^8 \sim 2^{18}$ 以提供更大的溢出周期，分频由 WDTC 寄存器中的 WS2~WS0 位来决定。

看门狗定时器控制寄存器

WDTC 寄存器用于控制 WDT 功能的使能 / 除能控制及溢出周期选择。该寄存器控制 WDT 的所有操作。

• WDTC 寄存器

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

Bit 7~3 **WE4~WE0**: WDT 软件控制位

10101: 除能
01010: 使能
其它值: MCU 复位

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

Bit 2~0 **WS2~WS0**: WDT 溢出周期选择位

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

这三位控制 WDT 时钟源的分频比，从而实现了对 WDT 溢出周期的控制。

• RSTFC 寄存器

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

“x”: 未知

Bit 7~4 未定义，读为“0”

Bit 3 **RSTF**: RSTC 寄存器软件复位标志位
详见其它章节

Bit 2 **LVRF**: LVR 复位标志位
详见其它章节

- Bit 1 **LRF**: LVR 控制寄存器软件复位标志位
 详见其它章节
- Bit 0 **WRF**: WDT 控制寄存器软件复位标志位
 0: 未发生
 1: 发生
 当 WDT 控制寄存器软件复位发生时，此位被置为“1”。因注意此位只能通过应用程序清零。

看门狗定时器操作

当 WDT 溢出时，它产生一个芯片复位的动作。这也就意味着正常工作期间，用户需在应用程序中看门狗溢出前有策略地清看门狗定时器以防止其产生复位，可使用清除看门狗指令实现。无论什么原因，程序失常跳转到一个未知的地址或进入一个死循环，这些清除指令都不能被正确执行，此种情况下，看门狗将溢出以使单片机复位。看门狗定时器控制寄存器 WDT_C 中的 WE4~WE0 位可提供使能/除能控制以及控制看门狗定时器复位操作。当 WE4~WE0 设置为“10101B”时除能 WDT 功能，而当设置为“01010B”时使能 WDT 功能。如果 WE4~WE0 设置为除“01010B”和“10101B”以外的值时，单片机将在一段延迟时间 t_{SRESET} 后复位。上电后这些位的值为“01010B”。

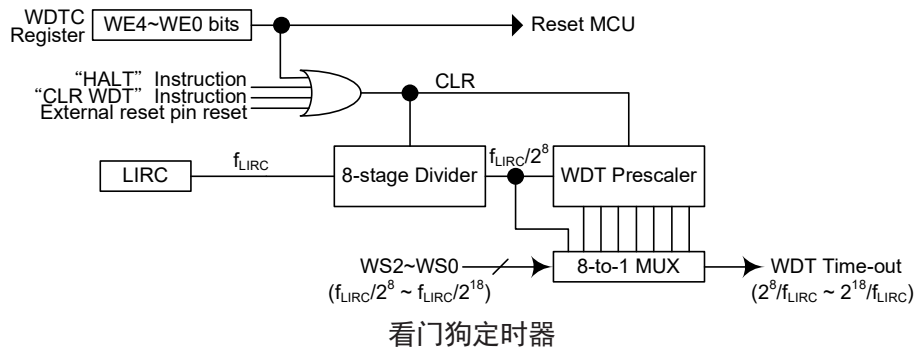
WE4 ~ WE0	WDT 功能
10101B	除能
01010B	使能
其它值	复位 MCU

看门狗定时器使能 / 除能控制

程序正常运行时，WDT 溢出将导致芯片复位，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。有四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“01010B”和“10101B”外的任意值；第二种是通过看门狗定时器软件清除指令，而第三种是通过“HALT”指令。最后一种为外部硬件复位，即外部复位引脚上出现低电平所引起的复位，外部复位引脚是否存在由 RSTC 寄存器设置。

该系列单片机只有一种软件指令清除 WDT 的方式，即看门狗指令“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 8ms。



复位和初始化

复位功能是整个单片机中基本的部分，使得单片机可以设置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置为预先设置的状态。程序计数器就是其中之一，它会被清除为零，使得单片机从最低的程序存储器地址开始执行程序。

除上电复位以外，即使单片机处于正常工作状态，有些情况的发生也会迫使单片机复位。譬如当单片机上电后已经开始执行程序， $\overline{\text{RES}}$ 强制拉为低电平。这种复位为正常操作复位，单片机中只有一些寄存器受影响，而大部分寄存器不会改变，在复位引脚恢复至高电平后，单片机可以正常运行。

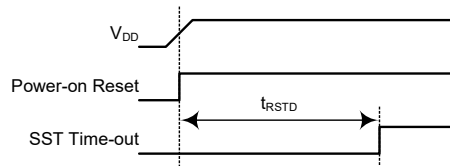
看门狗定时器溢出也是单片机复位方式之一。另一种复位为低电压复位即 LVR 复位，该复位方式与 $\overline{\text{RES}}$ 引脚复位相似，在电源供应电压低于一定阈值的时候，系统会产生完全复位。此外还有一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。

复位功能

通过内部和外部事件触发复位，单片机共有以下五种复位方式：

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置在预设条件。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置为输入状态。

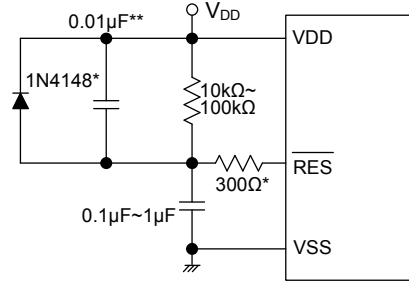


注： t_{RSTD} 为上电延迟时间，典型值为 48ms。

上电复位时序图

$\overline{\text{RES}}$ 引脚复位

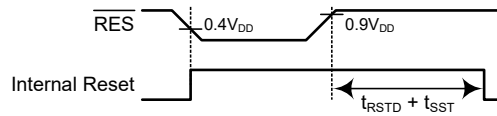
由于 $\overline{\text{RES}}$ 复位引脚与 I/O 引脚共用， $\overline{\text{RES}}$ 复位功能必须通过 RSTC 控制寄存器选择。虽然单片机有一个内部 RC 复位功能，如果电源上升缓慢或上电时电源不稳定，内部 RC 振荡可能导致芯片复位不良，所以推荐使用和 $\overline{\text{RES}}$ 引脚连接的外部 RC 电路，由 RC 电路所造成的时间延迟使得 $\overline{\text{RES}}$ 引脚在电源供应稳定前的一段延长周期内保持在低电平。在这段时间内，单片机的正常操作是被禁止的。 $\overline{\text{RES}}$ 引脚达到一定电压值后，再经过延迟时间 t_{RSTD} 单片机可以开始进行正常操作。下图中 SST 是系统延迟周期“System Start-up Timer”的缩写。在许多应用场合，可以在 VDD 和 $\overline{\text{RES}}$ 之间接入一个电阻，在 VSS 与 $\overline{\text{RES}}$ 之间接入一个电容作为外部复位电路。与 $\overline{\text{RES}}$ 脚上所有相连接的线段必须尽量短以减少噪声干扰。当系统在较强干扰的场合工作时，建议使用增强型的复位电路，如下图所示。



注：“*”表示建议加上此元件以加强静电保护。
“**”表示建议在电源有较强干扰场合加上此元件。

外部 RES 电路

RES 引脚通过外部硬件强迫拉至低电平时，此种复位形式即会发生。这种复位方式和其它的复位方式一样，程序计数器会被清除为零且程序从头开始执行。



注： t_{RSTD} 为上电延迟时间，其经典值为 16ms。

RES 复位时序图

内部复位控制寄存器 RSTC 用于选择外部 RES 引脚功能，并且在单片机操作环境噪声干扰不能正常运行时复位。如果 RSTC 寄存器设置为 01010101B 或 10101010B 外的其它任意值，则经过延迟时间 t_{SRESET} 后单片机复位。上电后这几位的值为 01010101B。

RSTC7 ~ RSTC0 位	复位功能
01010101B	I/O
10101010B	RES
其它任意值	复位 MCU

内部复位功能控制

● RSTC 寄存器

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

Bit 7~0 **RSTC7~RSTC0**: 复位功能控制位

01010101: I/O 引脚
10101010: RES 引脚
其它: MCU 复位

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

除了 WDT 溢出硬件热复位之外的复位方式都会将此寄存器复位为 POR 值。应注意的是若该寄存器设置为“10101010”以选择 RES 引脚，则该配置优先等级高于其它的相关引脚共用控制。

● RSTFC 寄存器

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

“x”：未知

Bit 7~4 未定义，读为“0”

Bit 3 **RSTF**：复位控制寄存器软件复位标志位

0：未发生

1：发生

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

Bit 2 **LVRF**：LVR 复位标志位

详见其它章节

Bit 1 **LRF**：LVR 控制寄存器软件复位标志位

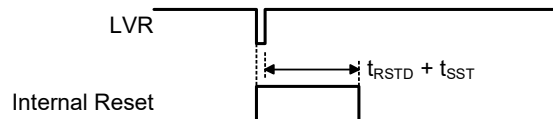
详见其它章节

Bit 0 **WRF**：WDT 控制寄存器软件复位标志位

详见其它章节

低电压复位 – LVR

单片机具有低电压复位电路，用来监测它的电源电压。LVR 始终使能，并会设置一个电源复位低电压， V_{LVR} 。例如在更换电池的情况下，单片机供应的电压可能会在 $0.9V \sim V_{LVR}$ 之间，这时 LVR 将会自动复位单片机且 RSTFC 寄存器中的 LVRF 标志位置位。LVR 包含以下的规格：有效的 LVR 信号，即在 $0.9V \sim V_{LVR}$ 的低电压状态的时间，必须超过 LVD/LVR 电气特性中 t_{LVR} 参数的值。如果低电压存在不超过 t_{LVR} 参数的值，则 LVR 将会忽略它且不会执行复位功能。 V_{LVR} 参数值通过 LVRC 寄存器中的 LVS 位段选择。若 LVS7~LVS0 位段的值由于不利的环境因素如噪声而发生改变，单片机将在一段延迟时间 t_{SRESET} 后复位，此时 RSTFC 寄存器中的 LRF 位将被置为 1。上电后该寄存器的默认值为 01010101B。注意当单片机进入空闲或休眠模式，LVR 功能将自动关闭。



注： t_{RSTD} 为上电延迟时间，典型值为 48ms。

低电压复位时序图

• LVRC 寄存器

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

Bit 7~0 **LVS7~LVS0**: LVR 电压选择

01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V

其它值: 单片机复位 – 寄存器复位为 POR 值

若低电压情况发生且满足以上定义的低电压复位值, 则单片机复位。当低电压状态保持时间大于 t_{LVR} 后, 响应复位。此时复位后的寄存器内容保持不变。

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

• RSTFC 寄存器

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

“x”: 未知

Bit 7~4 未定义, 读为 “0”

Bit 3 **RSTF**: 复位控制寄存器软件复位标志位
详见其它章节

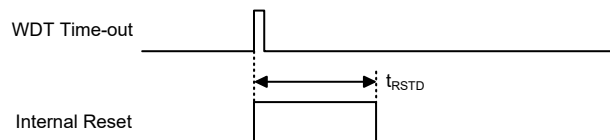
Bit 2 **LVRF**: LVR 复位标志位
0: 未发生
1: 发生
此位在出现指定低压复位情况时被置位且仅能由应用程序清零。

Bit 1 **LRF**: LVR 控制寄存器 LVRC 软件复位标志位
0: 未发生
1: 发生
当 LVRC 寄存器的值不属于任何具体定义的 LVR 电压寄存器值时此位被置位。该复位方式与软件复位功能很相似。此位只能由应用程序清零。

Bit 0 **WRF**: WDT 控制寄存器软件复位标志位
详见其它章节

正常运行时的看门狗溢出复位

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。

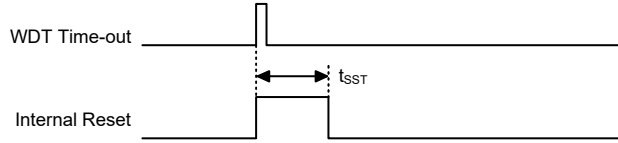


注: t_{RSTD} 为上电延迟时间, 典型时间为 16ms。

正常运行时看门狗溢出时序图

休眠或空闲时的看门狗溢出复位

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_{SS1} 的详细说明请参考交流电气特性。



休眠或空闲时的看门狗溢出复位时序图

复位初始状态

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

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

注：“u”表示未改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

项目	复位后情况
程序计数器	清除为零
中断	所有中断被除能
WDT, 时基	复位后清零，且 WDT 开始计数
定时模块	所有定时模块停止
输入 / 输出口	I/O 口设为输入模式
堆栈指针	堆栈指针指向堆栈顶端

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

寄存器	HT66F2350	HT66F2360	HT66F2370	HT66F2390	上电复位	RES 复位 (正常运行)	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (休眠/空闲)
IAR0	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP0	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1L	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1H	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	●	●	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	●	●	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	●	●	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	●				--- x xxxx	--- u uuuu	--- u uuuu	--- u uuuu	--- u uuuu
TBHP		●			-- xx xxxx	-- uu uuuu	-- uu uuuu	-- uu uuuu	-- uu uuuu
TBHP			●		- xxx xxxx	- uuu uuuu	- uuu uuuu	- uuu uuuu	- uuu uuuu
TBHP				●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
STATUS	●	●	●	●	xx00 xxxx	uuuu uuuu	uuuu uuuu	xx1u uuuu	uu1l uuuu
PBP		●			---- --- 0	---- --- 0	---- --- 0	---- --- 0	---- --- u
PBP			●		---- -- 00	---- -- 00	---- -- 00	---- -- 00	---- -- uu
PBP				●	---- - 000	---- - 000	---- - 000	---- - 000	---- - uuu
IAR2	●	●	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP2L	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	●	●	●	●	---- 0x00	---- uuuu	---- u1uu	---- uuuu	---- uuuu
INTC0	●	●	●	●	- 000 0000	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
INTC1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC3	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PA	●	●	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	●	●	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	●	●	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	●	●	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	●	●	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	●	●	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD	●	●	●	●	- 111 1111	- 111 1111	- 111 1111	- 111 1111	- uuu uuuu
PDC	●	●	●	●	- 111 1111	- 111 1111	- 111 1111	- 111 1111	- uuu uuuu
PDPU	●	●	●	●	- 000 0000	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
PE	●	●	●	●	--- 1 1111	--- 1 1111	--- 1 1111	--- 1 1111	--- u uuuu

寄存器	HT66F2350	HT66F2360	HT66F2370	HT66F2390	上电复位	RES 复位 (正常运行)	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (休眠/空闲)
PEC	●	●	●	●	--- 1 1111	--- 1 1111	--- 1 1111	--- 1 1111	--- u uuuu
PEPU	●	●	●	●	--- 0 0000	--- 0 0000	--- 0 0000	--- 0 0000	--- u uuuu
PF	●	●	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFC	●	●	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFPU	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PG		●	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PGC		●	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PGPU		●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PH		●	●	●	-- 11 1111	-- 11 1111	-- 11 1111	-- 11 1111	-- uu uuuu
PHC		●	●	●	-- 11 1111	-- 11 1111	-- 11 1111	-- 11 1111	-- uu uuuu
PHPU		●	●	●	-- 00 0000	-- 00 0000	-- 00 0000	-- 00 0000	-- uu uuuu
PMPS	●	●	●	●	---- -- 00	---- -- 00	---- -- 00	---- -- 00	---- -- uu
RSTC	●	●	●	●	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu
VBGRC	●	●	●	●	---- -- 00	---- -- 00	---- -- 00	---- -- 00	---- -- u
MF10	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MF11	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MF12	●	●	●	●	-- 00 -- 00	-- 00 -- 00	-- 00 -- 00	-- 00 -- 00	-- uu -- uu
MF13	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MF14	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MF15	●	●			-- 00 -- 00	-- 00 -- 00	-- 00 -- 00	-- 00 -- 00	-- uu -- uu
MF15			●	●	- 000 - 000	- 000 - 000	- 000 - 000	- 000 - 000	- uuu - uuu
INTEG	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SCC	●	●	●	●	000- 0000	000- 0000	000- 0000	000- 0000	uuu- uuuu
HIRCC	●	●	●	●	---- 0001	---- 0001	---- 0001	---- 0001	---- uuuu
HXTC	●	●	●	●	---- -000	---- -000	---- -000	---- -000	---- -uuu
LXTC	●	●	●	●	---- -- 00	---- -- 00	---- -- 00	---- -- 00	---- -- uu
WDTC	●	●	●	●	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu
LVRC	●	●	●	●	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu
LVDC	●	●	●	●	-- 00 0000	-- 00 0000	-- 00 0000	-- 00 0000	-- uu uuuu
EEA	●	●			0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEAL			●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEAH			●		---- --- 0	---- --- 0	---- --- 0	---- --- 0	---- --- u
EEAH				●	---- -- 00	---- -- 00	---- -- 00	---- -- 00	---- -- uu
EED	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CP0C	●	●	●	●	- 000 00--	- 000 00--	- 000 00--	- 000 00--	- uuu uu--
CP1C	●	●	●	●	- 000 00--	- 000 00--	- 000 00--	- 000 00--	- uuu uu--
PTM0C0	●	●	●	●	0000 0---	0000 0---	0000 0---	0000 0---	uuuu u---
PTM0C1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

寄存器	HT66F2350	HT66F2360	HT66F2370	HT66F2390	上电复位	RES 复位 (正常运行)	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (休眠/空闲)
PTM0DH	●	●	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu
PTM0AL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0AH	●	●	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu
PTM0RPL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	●	●	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu
STM0C0	●	●	●	●	0000 0---	0000 0---	0000 0---	0000 0---	uuuu u---
STM0C1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0DL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0DH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0AL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0AH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0RP	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC0	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC2	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC3		●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MDUWR0	●	●	●	●	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu
MDUWR1	●	●	●	●	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu
MDUWR2	●	●	●	●	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu
MDUWR3	●	●	●	●	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu
MDUWR4	●	●	●	●	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu
MDUWR5	●	●	●	●	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu
MDUWC-TRL	●	●	●	●	00-- ----	00-- ----	00-- ----	00-- ----	uu-- ----
CP0VOS	●	●	●	●	-001 0000	-001 0000	-001 0000	-001 0000	-uuu uuuu
CP1VOS	●	●	●	●	-001 0000	-001 0000	-001 0000	-001 0000	-uuu uuuu
PSCR0	●	●	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu
TB0C	●	●	●	●	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu
TB1C	●	●	●	●	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu
PSCR1	●	●	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu
SADOL	●	●	●	●	xxxx ----	xxxx ----	xxxx ----	xxxx ----	uuuu ---- (ADRF=0)
									uuuu uuuu (ADRF=0)
SADOH	●	●	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu (ADRF=0)
									---- uuuu (ADRF=0)
SADC0	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

寄存器	HT66F2350	HT66F2360	HT66F2370	HT66F2390	上电复位	RES 复位 (正常运行)	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (休眠/空闲)
SADC1	●	●	●	●	0000 - 000	0000 - 000	0000 - 000	0000 - 000	uuuu - uu
SADC2	●	●	●	●	0- 00 0000	0- 00 0000	0- 00 0000	0- 00 0000	u- uu uuuu
SIMC0	●	●	●	●	111- 0000	111- 0000	111- 0000	111- 0000	uuu- uuuu
SIMC1	●	●	●	●	1000 0001	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	●	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/ SIMC2	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMTOC	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SPIAC0	●	●	●	●	111- -- 00	111- -- 00	111- -- 00	111- -- 00	uuu- -- uu
SPIAC1	●	●	●	●	-- 00 0000	-- 00 0000	-- 00 0000	-- 00 0000	-- uu uuuu
SPIAD	●	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
FARL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARH	●				--- 0 0000	--- 0 0000	--- 0 0000	--- 0 0000	--- u uuuu
FARH		●			-- 00 0000	-- 00 0000	-- 00 0000	-- 00 0000	-- uu uuuu
FARH			●		- 000 0000	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
FARH				●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0L	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0H	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1L	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1H	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2L	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2H	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3L	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3H	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SCOMC	●	●	●	●	- 000 ----	- 000 ----	- 000 ----	- 000 ----	- uuu ----
CRCCR	●	●	●	●	---- -- 0	---- -- 0	---- -- 0	---- -- 0	---- -- u
CRCIN	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CRCDL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CRCDH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
IECC	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
U0SR	●	●	●	●	0000 1011	0000 1011	0000 1011	0000 1011	uuuu uuuu
U0CR1	●	●	●	●	0000 00x0	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
U0CR2	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TXR_RXR0	●	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG0	●	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
U1SR	●	●	●	●	0000 1011	0000 1011	0000 1011	0000 1011	uuuu uuuu
U1CR1	●	●	●	●	0000 00x0	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
U1CR2	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TXR_RXR1	●	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

寄存器	HT66F2350	HT66F2360	HT66F2370	HT66F2390	上电复位	RES 复位 (正常运行)	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (休眠/空闲)
BRG1	●	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
U2SR			●	●	0000 1011	0000 1011	0000 1011	0000 1011	uuuu uuuu
U2CR1			●	●	0000 00x0	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
U2CR2			●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TXR_RXR2			●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG2			●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PTM1C0	●	●	●	●	0000 0- - -	0000 0- - -	0000 0- - -	0000 0- - -	uuuu u- - -
PTM1C1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DH	●	●	●	●	- - - - - 00	- - - - - 00	- - - - - 00	- - - - - 00	- - - - - uu
PTM1AL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1AH	●	●	●	●	- - - - - 00	- - - - - 00	- - - - - 00	- - - - - 00	- - - - - uu
PTM1RPL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	●	●	●	●	- - - - - 00	- - - - - 00	- - - - - 00	- - - - - 00	- - - - - uu
PTM2C0	●	●	●	●	0000 0- - -	0000 0- - -	0000 0- - -	0000 0- - -	uuuu u- - -
PTM2C1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2AL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2AH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2RPL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3C0	●	●	●	●	0000 0- - -	0000 0- - -	0000 0- - -	0000 0- - -	uuuu u- - -
PTM3C1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3DL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3DH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3AL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2AH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3RPL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3RPH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1C0	●	●	●	●	0000 0- - -	0000 0- - -	0000 0- - -	0000 0- - -	uuuu u- - -
STM1C1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1DL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1DH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1AL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1AH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1RP	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2C0	●	●	●	●	0000 0- - -	0000 0- - -	0000 0- - -	0000 0- - -	uuuu u- - -
STM2C1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

寄存器	HT66F2350	HT66F2360	HT66F2370	HT66F2390	上电复位	RES 复位 (正常运行)	LVR 复位 (正常运行)	WDT 溢出 (正常运行)	WDT 溢出 (休眠/空闲)
STM2DL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2DH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2AL	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2AH	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2RP	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEC	●	●	●	●	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
FC0	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC2	●	●	●	●	---- -- 0	---- -- 0	---- -- 0	---- -- 0	---- -- u
IFS0	●	●	●	●	- 000 0000	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
IFS1	●	●	●	●	- 000 0000	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
IFS2	●	●	●	●	- 000 0000	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
IFS3	●	●			---- -- 00	---- -- 00	---- -- 00	---- -- 00	---- -- uu
IFS3			●	●	---- - 000	---- - 000	---- - 000	---- - 000	---- - uuu
PAS0	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS0	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS0	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS1	●	●	●	●	-- 00 0000	-- 00 0000	-- 00 0000	-- 00 0000	-- uu uuuu
PES0	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PES1	●	●	●	●	---- -- 00	---- -- 00	---- -- 00	---- -- 00	---- -- uu
PFS0	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PFS1	●	●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PGS0		●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PGS1		●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PHS0		●	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PHS1		●	●	●	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu

注：“u”表示未改变
“x”表示未知
“-”表示未定义

输入 / 输出端口

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

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	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
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	—	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	—	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	—	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PE	—	—	—	PE4	PE3	PE2	PE1	PE0
PEC	—	—	—	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	—	—	—	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PF	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFC	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0

“—”：未定义，读为“0”

输入 / 输出逻辑功能寄存器列表 – HT66F2350

寄存器名称	位							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	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
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	—	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	—	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	—	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PE	—	—	—	PE4	PE3	PE2	PE1	PE0
PEC	—	—	—	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	—	—	—	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PF	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFC	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0
PG	PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0
PGC	PGC7	PGC6	PGC5	PGC4	PGC3	PGC2	PGC1	PGC0
PGPU	PGPU7	PGPU6	PGPU5	PGPU4	PGPU3	PGPU2	PGPU1	PGPU0
PH	—	—	PH5	PH4	PH3	PH2	PH1	PH0
PHC	—	—	PHC5	PHC4	PHC3	PHC2	PHC1	PHC0
PHPU	—	—	PHPU5	PHPU4	PHPU3	PHPU2	PHPU1	PHPU0

“—”：未定义，读为“0”

输入 / 输出逻辑功能寄存器列表 – HT66F2360/HT66F2370/HT66F2390

上拉电阻

许多产品应用在端口处于输入状态时需要外加一个上拉电阻来实现上拉的功能。为了免去外部上拉电阻，当引脚规划为输入时，可由内部连接到一个上拉电阻。这些上拉电阻可通过相关上拉控制寄存器来设置，它用一个 PMOS 晶体管来实现上拉电阻功能。应注意只有在引脚共用功能用脚配置为输入或 NMOS 输出时，可通过相关上拉控制寄存器控制上拉电阻，否则，上拉电阻无法被使能。

● PxPU 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PxPU7	PxPU6	PxPU5	PxPU4	PxPU3	PxPU2	PxPU1	PxPU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

PxPUn: I/O 端口 x 引脚上拉功能控制

- 0: 除能
- 1: 使能

PxPUn 用于控制引脚上拉功能。根据不同的单片机型号，这里的 x 可以是 A, B, C, D, E, F, G 和 H。但是，每个 I/O 端口的实际有效位可能不同。

PA 口唤醒

当使用“HALT”指令迫使单片机进入休眠或空闲模式，单片机的系统时钟将会停止以降低功耗，此功能对于电池及低功耗应用很重要。唤醒单片机有很多种方法，其中之一就是使 PA 口的其中一个引脚从高电平转为低电平。这个功能特别适合于通过外部开关来唤醒的应用。PA 口的每个引脚可以通过设置 PAWU 寄存器来单独选择是否具有唤醒功能。应注意只有在引脚共用功能选择为通用输入/输出且单片机进入暂停模式时，此功能可由唤醒控制寄存器控制。

● PAWU 寄存器

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

Bit 7~0 PAWU7~PAWU0: PA 端口引脚唤醒功能控制

- 0: 除能
- 1: 使能

输入 / 输出端口控制寄存器

每一个输入 / 输出都具有各自的控制寄存器，用来控制输入 / 输出状态。从而每个 I/O 引脚都可以通过软件控制，动态的设置为 CMOS 输出或输入。所有的 I/O 端口的引脚都各自对应于 I/O 端口控制的某一位。若 I/O 引脚要实现输入功能，则对应的控制寄存器的位需要设置为“1”。这时程序指令可以直接读取输入脚的逻辑状态。若控制寄存器相应的位被设定为“0”，则此引脚被设置为 CMOS 输出。当引脚设置为输出状态时，程序指令读取的是输出端口寄存器的内容。注意，如果对输出口做读取动作时，程序读取到的是内部输出数据锁存器中的状态，而不是输出引脚上实际的逻辑状态。

注意，如果对输出口做读取动作时，程序读取到的是内部输出数据锁存器中的状态，而不是输出引脚上实际的逻辑状态。

• PxC 寄存器

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

PxCn: I/O 端口 x 引脚类型选择

0: 输出

1: 输入

PxCn 用于控制引脚类型选择。根据不同的单片机型号，这里的 x 可以是 A, B, C, D, E, F, G 和 H。但是，每个 I/O 端口的实际有效位可能不同。

输入 / 输出端口源电流选择

该系列单片机的每个 I/O 口都支持不同的源电流驱动能力。通过配置相应的选择寄存器 SLEDCn，特定的 I/O 端口可支持 4 个 Level 的源电流驱动能力。用户可参考输入 / 输出电气特性章节为不同应用选择所需的源电流。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SLEDC0	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
SLEDC1	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
SLEDC2	SLEDC27	SLEDC26	SLEDC25	SLEDC24	SLEDC23	SLEDC22	SLEDC21	SLEDC20

输入 / 输出端口源电流选择寄存器列表 – HT66F2350

寄存器名称	位							
	7	6	5	4	3	2	1	0
SLEDC0	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
SLEDC1	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
SLEDC2	SLEDC27	SLEDC26	SLEDC25	SLEDC24	SLEDC23	SLEDC22	SLEDC21	SLEDC20
SLEDC3	SLEDC37	SLEDC36	SLEDC35	SLEDC34	SLEDC33	SLEDC32	SLEDC31	SLEDC30

输入 / 输出端口源电流选择寄存器列表 – HT66F2360/HT66F2370/HT66F2390

• SLEDC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
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 **SLEDC07~SLEDC06**: PB7~PB4 源电流选择

00: Level 0 (最小)

01: Level 1

10: Level 2

11: Level 3 (最大)

Bit 5~4 **SLEDC05~SLEDC04**: PB3~PB0 源电流选择

00: Level 0 (最小)

01: Level 1

10: Level 2

11: Level 3 (最大)

- Bit 3~2 **SLEDC03~SLEDC02**: PA7~PA4 源电流选择
 00: Level 0 (最小)
 01: Level 1
 10: Level 2
 11: Level 3 (最大)
- Bit 1~0 **SLEDC01~SLEDC00**: PA3~PA0 源电流选择
 00: Level 0 (最小)
 01: Level 1
 10: Level 2
 11: Level 3 (最大)

• **SLEDC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
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 **SLEDC17~SLEDC16**: PD6~PD4 源电流选择
 00: Level 0 (最小)
 01: Level 1
 10: Level 2
 11: Level 3 (最大)
- Bit 5~4 **SLEDC15~SLEDC14**: PD3~PD0 源电流选择
 00: Level 0 (最小)
 01: Level 1
 10: Level 2
 11: Level 3 (最大)
- Bit 3~2 **SLEDC13~SLEDC12**: PC7~PC4 源电流选择
 00: Level 0 (最小)
 01: Level 1
 10: Level 2
 11: Level 3 (最大)
- Bit 1~0 **SLEDC11~SLEDC10**: PC3~PC0 源电流选择
 00: Level 0 (最小)
 01: Level 1
 10: Level 2
 11: Level 3 (最大)

• **SLEDC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC27	SLEDC26	SLEDC25	SLEDC24	SLEDC23	SLEDC22	SLEDC21	SLEDC20
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 **SLEDC27~SLEDC26**: PF7~PF4 源电流选择
 00: Level 0 (最小)
 01: Level 1
 10: Level 2
 11: Level 3 (最大)

- Bit 5~4 **SLEDC25~SLEDC24**: PF3~PF0 源电流选择
00: Level 0 (最小)
01: Level 1
10: Level 2
11: Level 3 (最大)
- Bit 3~2 **SLEDC23~SLEDC22**: PE4 源电流选择
00: Level 0 (最小)
01: Level 1
10: Level 2
11: Level 3 (最大)
- Bit 1~0 **SLEDC21~SLEDC20**: PE3~PE0 源电流选择
00: Level 0 (最小)
01: Level 1
10: Level 2
11: Level 3 (最大)

● **SLEDC3 寄存器 – HT66F2360/HT66F2370/HT66F2390**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC37	SLEDC36	SLEDC35	SLEDC34	SLEDC33	SLEDC32	SLEDC31	SLEDC30
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 **SLEDC37~SLEDC36**: PH5~PH4 源电流选择
00: Level 0 (最小)
01: Level 1
10: Level 2
11: Level 3 (最大)
- Bit 5~4 **SLEDC35~SLEDC34**: PH3~PH0 源电流选择
00: Level 0 (最小)
01: Level 1
10: Level 2
11: Level 3 (最大)
- Bit 3~2 **SLEDC33~SLEDC32**: PG7~PG4 源电流选择
00: Level 0 (最小)
01: Level 1
10: Level 2
11: Level 3 (最大)
- Bit 1~0 **SLEDC31~SLEDC30**: PG3~PG0 源电流选择
00: Level 0 (最小)
01: Level 1
10: Level 2
11: Level 3 (最大)

输入 / 输出端口电源控制

此系列单片机为 PE3~PE0 输入 / 输出端口提供了不同的端口电源选择。通过设定 PMPS 寄存器中的 PMPS1~PMPS0 位段可确定端口电源是来自电源引脚 VDD 或是 VDDIO 引脚。若来自 VDDIO 引脚则该引脚功能必须通过相应的引脚共用功能选择位预先设定。必须注意的是若 VDDIO 引脚被选作端口电源引脚，则该引脚上的输入电源电压应等于或小于单片机供电电压。

● PMPS 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PMPS1	PMPS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

Bit 1~0 **PMPS1~PMPS0**: PE3~PE0 引脚电源选择

0x: VDD

1x: VDDIO

引脚共用功能

引脚的多功能可以增加单片机应用的灵活性。有限的引脚个数将会限制设计者，而引脚的多功能将会解决很多此类问题。此外，这些引脚功能可以通过应用程序控制一系列寄存器进行设定。

引脚共用功能选择寄存器

封装中有限的引脚个数会对单片机某些功能造成影响。然而，引脚功能共用功能通过引脚功能选择，使得小封装单片机具有更多不同的功能。该单片机具有端口“x”输出功能选择寄存器“n”，即寄存器 P_xS_n，和输入功能选择寄存器 IFS_n，用于选择多功能共用引脚上的所需功能。

当选择引脚共用输入功能，对应的输入和输出功能要进行合理设定。例如，I²C SDA 端口被使用，对应的输出引脚共用功能则要通过寄存器 P_xS_n 设置为 SDI/SDA 功能同时通过 IFS_n 寄存器选择 SDA 信号输入引脚。但是，如果选择外部中断功能，相关的输出引脚共用功能应选择作为 I/O 功能引脚，且也要选择中断输入信号。

需要注意的一点是要确保所需引脚共用功能被正确地选中和取消。对于多数引脚共用功能，要正确地选择所需引脚共用功能，需通过对相应的引脚共用控制寄存器正确配置来实现。接着配置相应的外围功能设定从而使能这些外围功能。但是，在设置相关引脚控制字段时，一些数字输入引脚如 INT_n、xTCK_n、xTPnI 等，与对应的通用 I/O 口共用同一个引脚共用设置选项。要选择这些引脚功能，除了上述的必要的引脚共用控制和外围功能设置外，还必须将其对应的端口控制寄存器位设置为输入。要正确地取消选择的引脚共用功能，应先除能外围功能，接着修改相应的引脚共用功能控制寄存器以选择其它引脚共用功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
PDS1	—	—	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
PES0	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
PES1	—	—	—	—	—	—	PES11	PES10
PFS0	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
PFS1	PFS17	PFS16	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10
PGS0 (HT66F2360/70/90)	PGS07	PGS06	PGS05	PGS04	PGS03	PGS02	PGS01	PGS00
PGS1 (HT66F2360/70/90)	PGS17	PGS16	PGS15	PGS14	PGS13	PGS12	PGS11	PGS10
PHS0 (HT66F2360/70/90)	PHS07	PHS06	PHS05	PHS04	PHS03	PHS02	PHS01	PHS00
PHS1 (HT66F2360/70/90)	—	—	—	—	PHS13	PHS12	PHS11	PHS10
IFS0	—	PTCK3PS	PTCK2PS	PTCK1PS	PTCK0PS	STCK2PS	STCK1PS	STCK0PS
IFS1	—	PTP3IPS	PTP2IPS	PTP1IPS	PTP0IPS	STP2IPS	STP1IPS	STP0IPS
IFS2	—	SCSBPS	SDISDAPS	SCKSCLPS	INT3PS	INT2PS	INT1PS	INT0PS
IFS3 (HT66F2350/60)	—	—	—	—	—	—	RX1PS	RX0PS
IFS3 (HT66F2370/90)	—	—	—	—	—	RX2PS	RX1PS	RX0PS

引脚共用功能选择寄存器列表

● PAS0 寄存器

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

- Bit 7~6 **PAS07~PAS06**: PA3 引脚功能选择
00/01/10: PA3/INT1
11: SDO
- Bit 5~4 **PAS05~PAS04**: PA2 引脚功能选择
00/01/10/11: PA2
- Bit 3~2 **PAS03~PAS02**: PA1 引脚功能选择
00/01/10: PA1/INT0
11: SCS
- Bit 1~0 **PAS01~PAS00**: PA0 引脚功能选择
00/01/10/11: PA0

● PAS1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PAS17~PAS16:** PA7 引脚功能选择
 00/01/10: PA7/INT1
 11: TX0

Bit 5~4 **PAS15~PAS14:** PA6 引脚功能选择
 00/01/10: PA6/INT0
 11: RX0

Bit 3~2 **PAS13~PAS12:** PA5 引脚功能选择
 00/01/10: PA5/INT3
 11: SCK/SCL

Bit 1~0 **PAS11~PAS10:** PA4 引脚功能选择
 00/01/10: PA4/INT2
 11: SDI/SDA

● PBS0 寄存器 – HT66F2350/HT66F2360

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

Bit 7~6 **PBS07~PBS06:** PB3 引脚功能选择
 00/01/10: PB3/PTP21
 11: PTP2

Bit 5~4 **PBS05~PBS04:** PB2 引脚功能选择
 00/01/10: PB2/PTP31/PTCK2
 11: PTP3

Bit 3~2 **PBS03~PBS02:** PB1 引脚功能选择
 00/01/10/11: PB1/PTCK3

Bit 1~0 **PBS01~PBS00:** PB0 引脚功能选择
 00/01/10: PB0/STCK2
 11: C0X

● PBS0 寄存器 – HT66F2370/HT66F2390

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

Bit 7~6 **PBS07~PBS06:** PB3 引脚功能选择
 00/01/10: PB3/PTP21
 11: PTP2

Bit 5~4 **PBS05~PBS04:** PB2 引脚功能选择
 00/01: PB2/PTP31/PTCK2
 10: RX2
 11: PTP3

Bit 3~2 **PBS03~PBS02**: PB1 引脚功能选择
00/01/10: PB1/PTCK3
11: TX2

Bit 1~0 **PBS01~PBS00**: PB0 引脚功能选择
00/01/10: PB0/STCK2
11: C0X

● **PBS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
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 **PBS17~PBS16**: PB7 引脚功能选择
00/01/10: PB7/STCK1
11: OSC2

Bit 5~4 **PBS15~PBS14**: PB6 引脚功能选择
00/01: PB6/STP1I
10: STP1
11: OSC1

Bit 3~2 **PBS13~PBS12**: PB5 引脚功能选择
00/01/10/11: PB5/RES

Bit 1~0 **PBS11~PBS10**: PB4 引脚功能选择
00/01/10: PB4
11: C1X

● **PCS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
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 **PCS07~PCS06**: PC3 引脚功能选择
00/01/10: PC3/PTCK0
11: AN3

Bit 5~4 **PCS05~PCS04**: PC2 引脚功能选择
00/01: PC2/PTP0I
10: PTP0
11: AN2

Bit 3~2 **PCS03~PCS02**: PC1 引脚功能选择
00: PC1
01: C0X
10: VREF
11: AN1

Bit 1~0 **PCS01~PCS00**: PC0 引脚功能选择
00/01: PC0
10: VREFI
11: AN0

● PCS1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
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 **PCS17~PCS16:** PC7 引脚功能选择
 00/01/10: PC7/INT3/STCK0
 11: AN7

Bit 5~4 **PCS15~PCS14:** PC6 引脚功能选择
 00/01: PC6/STP0I
 10: STP0
 11: AN6

Bit 3~2 **PCS13~PCS12:** PC5 引脚功能选择
 00/01/10: PC5/PTCK1
 11: AN5

Bit 1~0 **PCS11~PCS10:** PC4 引脚功能选择
 00/01: PC4/PTP1I
 10: PTP1
 11: AN4

● PDS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
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 **PDS07~PDS06:** PD3 引脚功能选择
 00/01/10: PD3/PTCK2
 11: AN11

Bit 5~4 **PDS05~PDS04:** PD2 引脚功能选择
 00: PD2/PTP2I
 01: PTP2
 10: TX1
 11: AN10

Bit 3~2 **PDS03~PDS02:** PD1 引脚功能选择
 00/01: PD1/STCK1
 10: RX1
 11: AN9

Bit 1~0 **PDS01~PDS00:** PD0 引脚功能选择
 00/01: PD0/INT2/STP1I
 10: STP1
 11: AN8

● PDS1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
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 5~4 **PDS15~PDS14**: PD6 引脚功能选择
00/01: PD6/STP2I
10: STP2
11: C1X
- Bit 3~2 **PDS13~PDS12**: PD5 引脚功能选择
00/01: PD5/PTCK3
10: TX0
11: C1+
- Bit 1~0 **PDS11~PDS10**: PD4 引脚功能选择
00: PD4/PTP3I
01: RX0
10: PTP3
11: C1-

● PES0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
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 **PES07~PES06**: PE3 引脚功能选择
00/01: PE3/PTP1I
10: PTP1
11: SCKA
- Bit 5~4 **PES05~PES04**: PE2 引脚功能选择
00/01/10: PE2/PTCK1
11: SDIA
- Bit 3~2 **PES03~PES02**: PE1 引脚功能选择
00/01: PE1/STP0I
10: STP0
11: SDOA
- Bit 1~0 **PES01~PES00**: PE0 引脚功能选择
00/01/10: PE0/STCK0
11: SCSA

● PES1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PES11	PES10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

- Bit 7~2 未定义, 读为“0”
- Bit 1~0 **PES11~PES10**: PE4 引脚功能选择
00/01/10: PE4
11: VDDIO

● PFS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
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 **PFS07~PFS06**: PF3 引脚功能选择
 00/01: PF3
 10: SCK/SCL
 11: SCOM3
- Bit 5~4 **PFS05~PFS04**: PF2 引脚功能选择
 00/01: PF2
 10: SDI/SDA
 11: SCOM2
- Bit 3~2 **PFS03~PFS02**: PF1 引脚功能选择
 00/01: PF1
 10: SDO
 11: SCOM1
- Bit 1~0 **PFS01~PFS00**: PF0 引脚功能选择
 00/01: PF0
 10: SCS
 11: SCOM0

● PFS1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PFS17	PFS16	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10
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 **PFS17~PFS16**: PF7 引脚功能选择
 00: PF7/STP2I
 01: TX1
 10: STP2
 11: C0+
- Bit 5~4 **PFS15~PFS14**: PF6 引脚功能选择
 00,01: PF6/STCK2
 10: RX1
 11: C0-
- Bit 3~2 **PFS13~PFS12**: PF5 引脚功能选择
 00,01: PF5/PTP0I
 10: PTP0
 11: XT1
- Bit 1~0 **PFS11~PFS10**: PF4 引脚功能选择
 00/01/10: PF4/PTCK0
 11: XT2

● **PGS0 寄存器 – HT66F2360**

Bit	7	6	5	4	3	2	1	0
Name	PGS07	PGS06	PGS05	PGS04	PGS03	PGS02	PGS01	PGS00
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 **PGS07~PGS06:** PG3 引脚功能选择
 00/01/10/11: PG3
- Bit 5~4 **PGS05~PGS04:** PG2 引脚功能选择
 00/01/10/11: PG2
- Bit 3~2 **PGS03~PGS02:** PG1 引脚功能选择
 00/01/10/11: PG1
- Bit 1~0 **PGS01~PGS00:** PG0 引脚功能选择
 00/01/10/11: PG0

● **PGS0 寄存器 – HT66F2370/HT66F2390**

Bit	7	6	5	4	3	2	1	0
Name	PGS07	PGS06	PGS05	PGS04	PGS03	PGS02	PGS01	PGS00
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 **PGS07~PGS06:** PG3 引脚功能选择
 00/01/10/11: PG3
- Bit 5~4 **PGS05~PGS04:** PG2 引脚功能选择
 00/01/10/11: PG2
- Bit 3~2 **PGS03~PGS02:** PG1 引脚功能选择
 00/01/10: PG1
 11: TX2
- Bit 1~0 **PGS01~PGS00:** PG0 引脚功能选择
 00/01/10: PG0
 11: RX2

● **PGS1 寄存器 – HT66F2360/HT66F2370/HT66F2390**

Bit	7	6	5	4	3	2	1	0
Name	PGS17	PGS16	PGS15	PGS14	PGS13	PGS12	PGS11	PGS10
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 **PGS17~PGS16:** PG7 引脚功能选择
 00/01/10/11: PG7
- Bit 5~4 **PGS15~PGS14:** PG6 引脚功能选择
 00/01/10/11: PG6
- Bit 3~2 **PGS13~PGS12:** PG5 引脚功能选择
 00/01/10/11: PG5
- Bit 1~0 **PGS11~PGS10:** PG4 引脚功能选择
 00/01/10/11: PG4

● PHS0 寄存器 – HT66F2360/HT66F2370/HT66F2390

Bit	7	6	5	4	3	2	1	0
Name	PHS07	PHS06	PHS05	PHS04	PHS03	PHS02	PHS01	PHS00
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 **PHS07~PHS06**: PH3 引脚功能选择
00/01/10: PH3
11: AN13
- Bit 5~4 **PHS05~PHS04**: PH2 引脚功能选择
00/01/10: PH2
11: AN12
- Bit 3~2 **PHS03~PHS02**: PH1 引脚功能选择
00/01/10/11: PH1
- Bit 1~0 **PHS01~PHS00**: PH0 引脚功能选择
00/01/10/11: PH0

● PHS1 寄存器 – HT66F2360/HT66F2370/HT66F2390

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PHS13	PHS12	PHS11	PHS10
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

- Bit 7~4 未定义, 读为“0”
- Bit 3~2 **PHS13~PHS12**: PH5 引脚功能选择
00/01/10: PH5
11: AN15
- Bit 1~0 **PHS11~PHS10**: PH4 引脚功能选择
00/01/10: PH4
11: AN14

● IFS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	PTCK3PS	PTCK2PS	PTCK1PS	PTCK0PS	STCK2PS	STCK1PS	STCK0PS
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

- Bit 7 未定义, 读为“0”
- Bit 6 **PTCK3PS**: PTCK3 输入源引脚选择
0: PD5
1: PB1
- Bit 5 **PTCK2PS**: PTCK2 输入源引脚选择
0: PD3
1: PB2
- Bit 4 **PTCK1PS**: PTCK1 输入源引脚选择
0: PC5
1: PE2
- Bit 3 **PTCK0PS**: PTCK0 输入源引脚选择
0: PC3
1: PF4

- Bit 2 **STCK2PS**: STCK2 输入源引脚选择
 0: PF6
 1: PB0
- Bit 1 **STCK1PS**: STCK1 输入源引脚选择
 0: PD1
 1: PB7
- Bit 0 **STCK0PS**: STCK0 输入源引脚选择
 0: PC7
 1: PE0

● **IFS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	PTP3IPS	PTP2IPS	PTP1IPS	PTP0IPS	STP2IPS	STP1IPS	STP0IPS
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

- Bit 7 未定义，读为“0”
- Bit 6 **PTP3IPS**: PTP3I 输入源引脚选择
 0: PD4
 1: PB2
- Bit 5 **PTP2IPS**: PTP2I 输入源引脚选择
 0: PD2
 1: PB3
- Bit 4 **PTP1IPS**: PTP1I 输入源引脚选择
 0: PC4
 1: PE3
- Bit 3 **PTP0IPS**: PTP0I 输入源引脚选择
 0: PC2
 1: PF5
- Bit 2 **STP2IPS**: STP2I 输入源引脚选择
 0: PD6
 1: PF7
- Bit 1 **STP1IPS**: STP1I 输入源引脚选择
 0: PD0
 1: PB6
- Bit 0 **STP0IPS**: STP0I 输入源引脚选择
 0: PC6
 1: PE1

● **IFS2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	SCSBPS	SDISDAPS	SCKSCLPS	INT3PS	INT2PS	INT1PS	INT0PS
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

- Bit 7 未定义，读为“0”
- Bit 6 **SCSBPS**: SCS 输入源引脚选择
 0: PA1
 1: PF0

- Bit 5 **SDISDAPS:** SDI/SDA 输入源引脚选择
 0: PA4
 1: PF2
- Bit 4 **SCKSCLPS:** SCK/SCL 输入源引脚选择
 0: PA5
 1: PF3
- Bit 3 **INT3PS:** INT3 输入源引脚选择
 0: PA5
 1: PC7
- Bit 2 **INT2PS:** INT2 输入源引脚选择
 0: PA4
 1: PD0
- Bit 1 **INT1PS:** INT1 输入源引脚选择
 0: PA3
 1: PA7
- Bit 0 **INT0PS:** INT0 输入源引脚选择
 0: PA1
 1: PA6

● **IFS3 寄存器 – HT66F2350/HT66F2360**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	RX1PS	RX0PS
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

- Bit 7~2 未定义，读为“0”
- Bit 1 **RX1PS:** RX1 输入源引脚选择
 0: PD1
 1: PF6
- Bit 0 **RX0PS:** RX0 输入源引脚选择
 0: PA6
 1: PD4

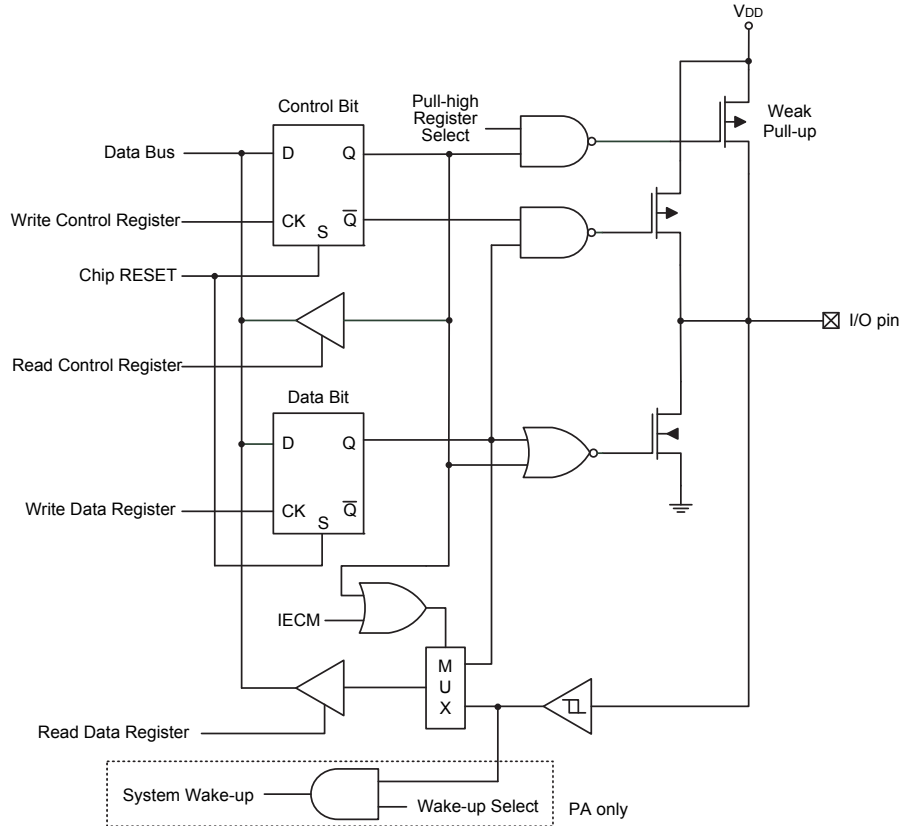
● **IFS3 寄存器 – HT66F2370/HT66F2390**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	RX2PS	RX1PS	RX0PS
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

- Bit 7~3 未定义，读为“0”
- Bit 2 **RX2PS:** RX2 输入源引脚选择
 0: PB2
 1: PG0
- Bit 1 **RX1PS:** RX1 输入源引脚选择
 0: PD1
 1: PF6
- Bit 0 **RX0PS:** RX0 输入源引脚选择
 0: PA6
 1: PD4

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 I/O 引脚功能的理解提供的一个参考。图中的引脚共用结构并非针对所有单片机。



逻辑功能输入 / 输出端口结构

读端口功能

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

• IECC 寄存器

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

Bit 7~0 IECS7~IECS0: 读端口功能使能控制 bit 7~bit 0

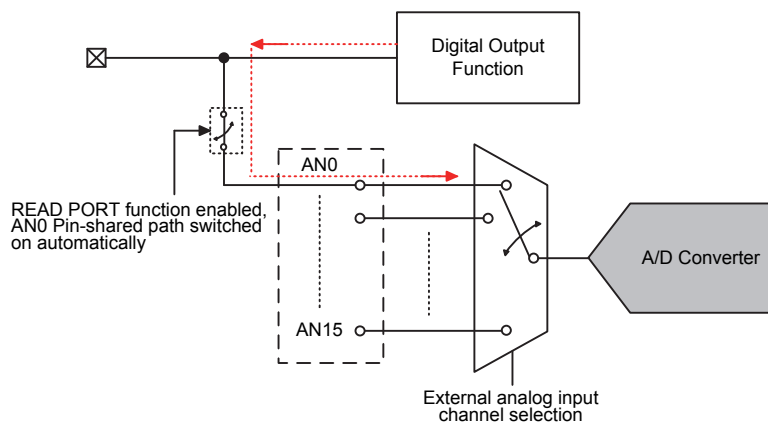
11001010: IECM=1 – 读端口功能使能

其它值: IECM=0 – 读端口功能除能

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

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

读端口模式的另一个功能是检查 A/D 通道。当读端口功能除能，若相应的选择位没有选中 A/D 输入引脚功能，则从外部引脚到内部模拟输入的 A/D 通道将会被关闭。对于带 A/D 转换通道的单片机，如 A/D AN15~AN0，通过适当配置 A/D 控制寄存器中的外部模拟输入通道选择位并选中相应的模拟输入引脚功能，所需的 A/D 转换通道将被开启。而读端口模式的功能则是强制开启 A/D 通道。例如，无论 AN0 是否选择作为模拟输入引脚使用，只要读端口功能使能，AN0 模拟输入通道都将开启。通过这种方式，AN0 模拟输入路径可与其共用引脚上的数字输出内部连接，然后在无外接模拟输入电压的情况下对相应的数字数据进行转换，从而实现 AN0 模拟输入通道检测。



A/D 通道输入路径内部连接

编程注意事项

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

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

定时器模块 – TM

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

这里只介绍各种 TM 的共性，更多详细资料请参考标准型和周期型定时器章节。

简介

该系列单片机包含了七个 TM 且每个 TM 可分为特定类型，即标准型 TM 或周期型 TM。虽然性质相似，但不同 TM 特性复杂度不同。本章介绍标准型和周期型 TM 的共性，更多详细资料分别见后面各章。两种类型 TM 的特性和区别见下表。

TM 功能	STM	PTM
定时 / 计数器	√	√
捕捉输入	√	√
比较匹配输出	√	√
PWM 通道数	1	1
单脉冲输出	1	1
PWM 对齐方式	边沿对齐	边沿对齐
PWM 调节周期 & 占空比	占空比或周期	占空比或周期

TM 功能概要

TM 操作

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

TM 时钟源

驱动 TM 计数器的时钟源很多。通过设置 xTMn 控制寄存器的 xTnCK2~xTnCK0 位，选择所需的时钟源。其中“x”代表“S”或“P”型 TM，“n”代表指定 TM 的序列号。该时钟源来自系统时钟 f_{SYS} 或内部高速时钟 f_H 或 f_{SUB} 时钟源或外部 xTCKn 引脚。xTCKn 引脚时钟源用于允许外部信号作为 TM 时钟源或用于事件计数。

TM 中断

标准型和周期型 TM 都有两个内部中断，分别是内部比较器 A 或比较器 P，当比较匹配发生时产生 TM 中断。当 TM 中断产生时，计数器清零并改变 TM 输出引脚的状态。

TM 外部引脚

无论哪种类型的 TM，都有两个 TM 输入引脚，分别为 xTCKn 和 xTPnI。TM 输入引脚 xTCKn 作为 TM 时钟源输入脚，通过设置 xTMnC0 寄存器中的 xTnCK2~xTnCK0 位进行选择，外部时钟源可通过该引脚来驱动内部 TM。xTCKn 输入引脚可以选择上升沿或下降沿有效。STCKn 和 PTCKn 引脚还可分别用作 STMn 和 PTMn 单脉冲输出模式的外部触发输入引脚。

另一种 xTM 输入引脚 STPnI 或 PTPnI 作为捕捉输入脚，其有效边沿有上升沿、下降沿和双沿，可分别通过设置 STMnC1 和 PTMnC1 寄存器中的 STnIO1~STnIO0 和 PTnIO1~PTnIO0 位来选择有效边沿类型。除 PTPnI 引脚外，PTCKn 也可用作 PTMn 捕捉输入模式的外部触发输入引脚。

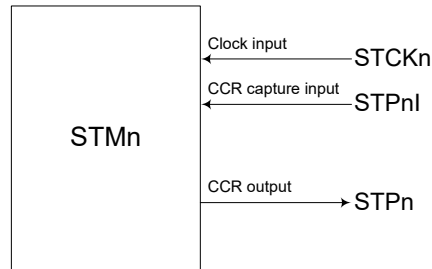
每个 TM 各有一个输出引脚，xTPn。TM 输出引脚可通过相应的引脚共用功能选择位选择，详细内容在引脚共用功能章节有所描述。当 TM 工作在比较匹配输出模式且比较匹配发生时，这些引脚会由 TM 控制切换到高电平或低电平或翻转。外部 xTPn 输出引脚也被 TM 用来产生 PWM 输出波形。当 TM 输出引脚与其它功能共用时，TM 输出功能需要通过寄存器先被设置。

单片机型号	STM		PTM	
	输入	输出	输入	输出
HT66F2350	STCK0, STP0I	STP0	PTCK0, PTP0I	PTP0
HT66F2360	STCK1, STP1I	STP1	PTCK1, PTP1I	PTP1
HT66F2370	STCK2, STP2I	STP2	PTCK2, PTP2I	PTP2
HT66F2390			PTCK3, PTP3I	PTP3

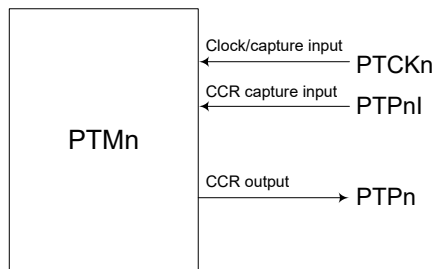
TM 外部引脚

TM 输入 / 输出引脚选择

通过设置与 TM 输入 / 输出引脚相关的引脚共用功能控制寄存器位，选择作为 TM 输入 / 输出功能或其它共用功能。正确设置选择位将相应的引脚用作 TM 输入 / 输出。更多引脚共用功能控制详见引脚共用功能章节。



STM 功能引脚控制方框图 (n=0~2)

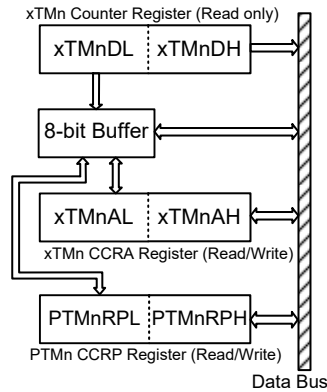


PTM 功能引脚控制方框图 (n=0~3)

编程注意事项

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

CCRA 和 CCRP 寄存器访问方式如下图所示，读写这些成对的寄存器需通过特殊的方式。建议使用“MOV”指令按照以下步骤访问 CCRA 或 CCRP 低字节寄存器，xTMnAL 或 PTMnRPL，否则可能导致无法预期的结果。



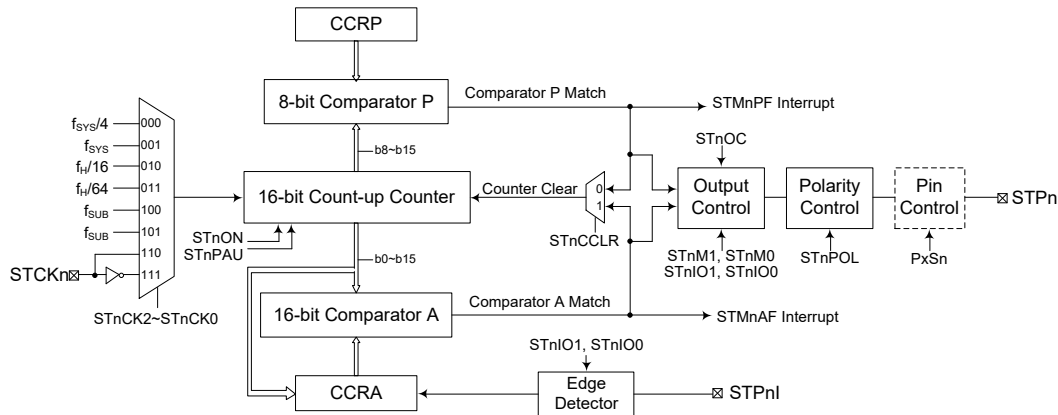
读写流程如下步骤所示：

- 写数据至 CCRA 或 CCRP
 - ◆ 步骤 1. 写数据至低字节寄存器 xTMnAL 或 PTMnRPL
 - 注意，此时数据仅写入 8-bit 缓存器。
 - ◆ 步骤 2. 写数据至高字节寄存器 xTMnAH 或 PTMnRPH
 - 注意，此时数据直接写入高字节寄存器，同时锁存在 8-bit 缓存器中的数据写入低字节寄存器。
- 从计数器寄存器和 CCRA 或 CCRP 中读取数据
 - ◆ 步骤 1. 由高字节寄存器 xTMnDH、xTMnAH 或 PTMnRPH 读取数据
 - 注意，此时高字节寄存器中的数据直接读取，同时由低字节寄存器读取的数据锁存至 8-bit 缓存器中。
 - ◆ 步骤 2. 由低字节寄存器 xTMnDL、xTMnAL 或 PTMnRPL 读取数据
 - 注意，此时读取 8-bit 缓存器中的数据。

标准型 TM – STM

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

单片机型号	STM 核心	STM 输入引脚	STM 输出引脚
HT66F2350 HT66F2360 HT66F2370 HT66F2390	16-bit STM (STM0, STM1, STM2)	STCK0, STP0I STCK1, STP1I STCK2, STP2I	STP0 STP1 STP2



标准型 TM 方框图 (n=0~2)

标准型 TM 操作

标准型 TM 核心是一个由用户选择的内部或外部时钟源驱动的 16-bit 向上计数器，它还包括两个内部比较器即比较器 A 和比较器 P。这两个比较器将计数器的值与 CCRP 和 CCRA 寄存器中的值进行比较。CCRP 是 8 位宽度，与计数器的高 8 位比较；而 CCRA 是 16 位的，与计数器的所有位比较。

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

标准型 TM 寄存器介绍

标准型 TM 的所有工作模式由一系列寄存器控制。一对只读寄存器用来存放 16 位计数器的值，一对读 / 写寄存器存放 16 位 CCRA 的值，STMnRP 寄存器存放 8 位 CCRP 的值。剩下两个控制寄存器设置不同的操作和控制模式。

寄存器名称	位							
	7	6	5	4	3	2	1	0
STMnC0	STnPAU	STnCK2	STnCK1	STnCK0	STnON	—	—	—
STMnC1	STnM1	STnM0	STnIO1	STnIO0	STnOC	STnPOL	STnDPX	STnCCLR
STMnDL	D7	D6	D5	D4	D3	D2	D1	D0
STMnDH	D15	D14	D13	D12	D11	D10	D9	D8
STMnAL	D7	D6	D5	D4	D3	D2	D1	D0
STMnAH	D15	D14	D13	D12	D11	D10	D9	D8
STMnRP	STnRP7	STnRP6	STnRP5	STnRP4	STnRP3	STnRP2	STnRP1	STnRP0

16-bit 标准型 TM 寄存器列表 (n=0~2)

• STMnDL 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 STMn 计数器低字节寄存器 bit 7~bit 0
STMn 16-bit 计数器 bit 7~bit 0

• STMnDH 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 STMn 计数器高字节寄存器 bit 7~bit 0
STMn 16-bit 计数器 bit 15~bit 8

• STMnAL 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 STMn CCRA 低字节寄存器 bit 7~bit 0
STMn 16-bit CCRA bit 7~bit 0

• **STMnAH 寄存器**

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

Bit 7~0 STMn CCRA 高字节寄存器 bit 7~bit 0
STMn 16-bit CCRA bit 15~bit 8

• **STMnC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	STnPAU	STnCK2	STnCK1	STnCK0	STnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **STnPAU**: STMn 计数器暂停控制位

0: 运行
1: 暂停

通过设置此位为高可使计数器暂停，清零此位恢复正常计数器操作。当处于暂停条件时，STMn 保持上电状态并继续耗电。当此位由低到高转换时，计数器将保留其剩余值，直到此位再次改变为低电平，并从此值开始继续计数。

Bit 6~4 **STnCK2~STnCK0**: 选择 STMn 计数时钟位

000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: STCKn 上升沿时钟
111: STCKn 下降沿时钟

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

Bit 3 **STnON**: STMn 计数器 On/Off 控制位

0: Off
1: On

此位控制 STMn 的总开关功能。设置此位为高则使能计数器使其运行，清零此位则除能 STMn。清零此位将停止计数器并关闭 STMn 减少耗电。当此位经由高到低转换时，内部计数器将保持其剩余值，直到此位再次改变为高电平。若 STMn 处于比较匹配输出模式时，当 STnON 位经由低到高的转换时，STMn 输出脚将复位至 STnOC 位指定的初始值。

Bit 2~0 未定义，读为“0”

• STMnC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	STnM1	STnM0	STnIO1	STnIO0	STnOC	STnPOL	STnDPX	STnCCLR
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 **STnM1~STnM0**: 选择 STMn 工作模式位

- 00: 比较匹配输出模式
- 01: 捕捉输入模式
- 10: PWM 输出模式或单脉冲输出模式
- 11: 定时 / 计数器模式

这两位设置 STMn 需要的工作模式。为了确保操作可靠，STMn 应在 STnM1 和 STnM0 位有任何改变前先关掉。在定时 / 计数器模式，STMn 输出脚控制必须除能。

Bit 5~4 **STnIO1~STnIO0**: 选择 STMn 外部引脚 STPn 功能

比较匹配输出模式

- 00: 无变化
- 01: 输出低
- 10: 输出高
- 11: 输出翻转

PWM 输出模式 / 单脉冲输出模式

- 00: PWM 输出无效状态
- 01: PWM 输出有效状态
- 10: PWM 输出
- 11: 单脉冲输出

捕捉输入模式

- 00: 在 STPnI 上升沿输入捕捉
- 01: 在 STPnI 下降沿输入捕捉
- 10: 在 STPnI 双沿输入捕捉
- 11: 输入捕捉除能

定时 / 计数器模式

未使用

此两位用于决定在一定条件达到时 STMn 输出引脚如何改变状态。这两位值的选择取决于 STMn 运行在何种模式下。

在比较匹配输出模式下，STnIO1 和 STnIO0 位决定当从比较器 A 比较匹配输出发生时 STMn 输出脚 STPn 如何改变状态。当从比较器 A 比较匹配输出发生时 STPn 输出脚能设为切换高、切换低或翻转当前状态。若此两位同时为 0 时，这个输出将不会改变。STPn 输出脚的初始值通过 STMnC1 寄存器的 STnOC 位设置取得。注意，由 STnIO1 和 STnIO0 位得到的输出电平必须与通过 STnOC 位设置的初始值不同，否则当比较匹配发生时，STPn 输出脚将不会发生变化。在 STPn 输出脚改变状态后，通过 STnON 位由低到高电平的转换复位至初始值。

在 PWM 输出模式，STnIO1 和 STnIO0 决定比较匹配条件发生时怎样改变 STPn 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。仅在 STMn 关闭时改变 STnIO1 和 STnIO0 位的值是很有必要的。若在 STMn 运行时改变 STnIO1 和 STnIO0 的值，PWM 输出的值将无法预料。

- Bit 3 STnOC:** STMn 输出脚 STPn 输出控制位
比较匹配输出模式
0: 初始低
1: 初始高
PWM 输出模式 / 单脉冲输出模式
0: 低有效
1: 高有效
这是 STMn 输出脚输出控制位。它取决于 STMn 此时正运行于比较匹配输出模式还是 PWM 输出模式 / 单脉冲输出模式。若 STMn 处于定时 / 计数器模式, 则其不受影响。在比较匹配输出模式时, 比较匹配发生前其决定 STMn 输出脚 STPn 的逻辑电平值。在 PWM 输出模式 / 单脉冲输出模式时, 其决定 PWM 信号是高有效还是低有效。
- Bit 2 STnPOL:** STPn 输出极性控制位
0: 同相
1: 反相
此位控制 STPn 输出脚的极性。此位为高时 STPn 输出脚反相, 为低时 STPn 输出脚同相。若 STMn 处于定时 / 计数器模式时其不受影响。
- Bit 1 STnDPX:** STMn PWM 周期 / 占空比控制位
0: CCRP – 周期; CCRA – 占空比
1: CCRP – 占空比; CCRA – 周期
此位决定 CCRA 与 CCRP 寄存器哪个被用于 PWM 波形的周期和占空比控制。
- Bit 0 STnCCLR:** 选择 STMn 计数器清零条件位
0: STMn 比较器 P 匹配
1: STMn 比较器 A 匹配
此位用于选择清除计数器的方法。标准型 TM 包括两个比较器 - 比较器 A 和比较器 P。这两个比较器每个都可以用作清除内部计数器。STnCCLR 位设为高, 计数器在比较器 A 比较匹配发生时被清除; 此位设为低, 计数器在比较器 P 比较匹配发生或计数器溢出时被清除。计数器溢出清除的方法仅在 CCRP 被清除为 0 时才能生效。STnCCLR 位在 PWM 输出, 单脉冲输出或输入捕捉模式时未使用。

● **STMnRP 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	STnRP7	STnRP6	STnRP5	STnRP4	STnRP3	STnRP2	STnRP1	STnRP0
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 STnRP7~STnRP0:** STMn CCRP 8-bit 寄存器, 与 STMn 计数器 bit 15~bit 8 比较器 P 匹配周期
0: 65536 个 STMn 时钟周期
1~255: 256×(1~255) 个 STMn 时钟周期
此八位设定内部 CCRP 8-bit 寄存器的值, 然后与内部计数器的高八位进行比较。如果 STnCCLR 位设为 0 时, 选择 CCRP 比较匹配后清除内部计数器。STnCCLR 位设为低, CCRP 比较匹配结果将重置内部计数器。由于 CCRP 只与计数器高八位比较, 比较结果是 256 时钟周期的倍数。CCRP 被清零时, 实际上会使得计数器在最大值溢出。

标准型 TM 工作模式

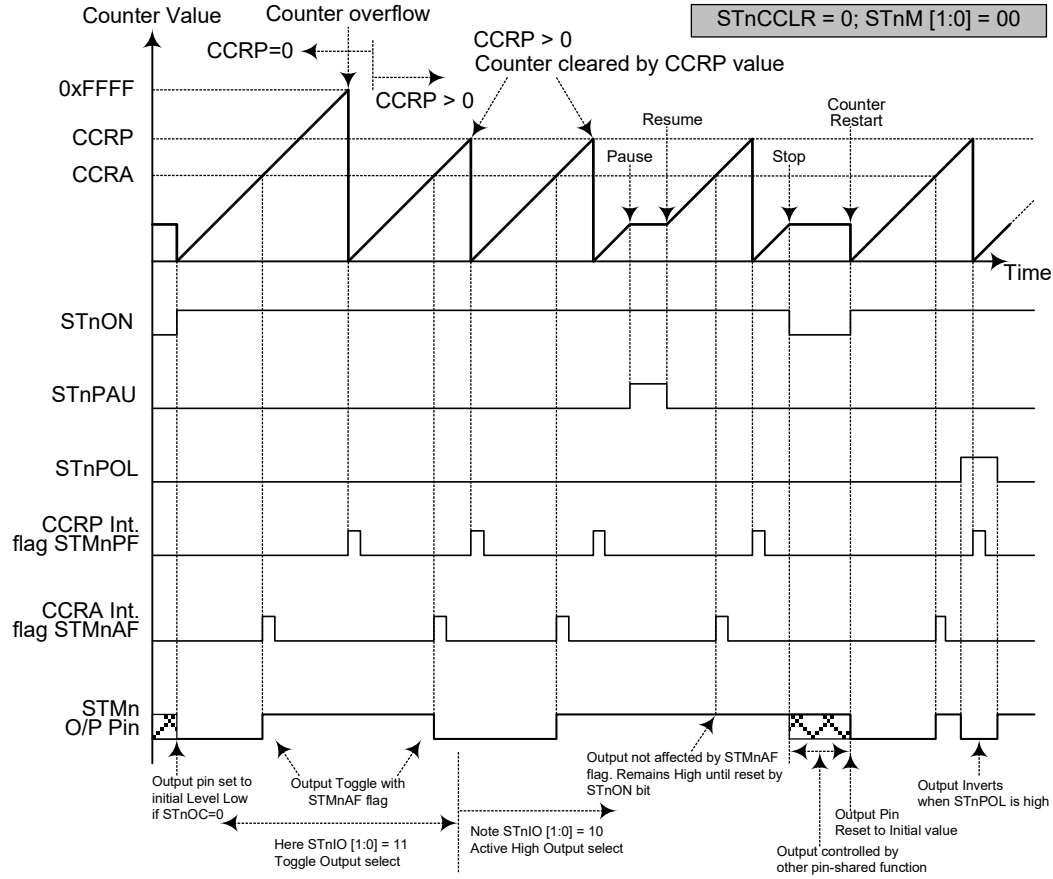
标准型 TM 有五种工作模式，即比较匹配输出模式，PWM 输出模式，单脉冲输出模式，捕捉输入模式或定时 / 计数器模式。通过设置 STMnC1 寄存器的 STnM1 和 STnM0 位选择任意模式。

比较匹配输出模式

为使 TM 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 位需要设置为“00”。当工作在该模式，一旦计数器使能并开始计数，有三种方法来清零，分别是：计数器溢出，比较器 A 比较匹配发生和比较器 P 比较匹配发生。当 STnCCLR 位为低，有两种方法清除计数器。一种是比较器 P 比较匹配发生，另一种是 CCRP 所有位设置为零并使得计数器溢出。此时，比较器 A 和比较器 P 的请求标志位 STMnAF 和 STMnPF 将分别置位。

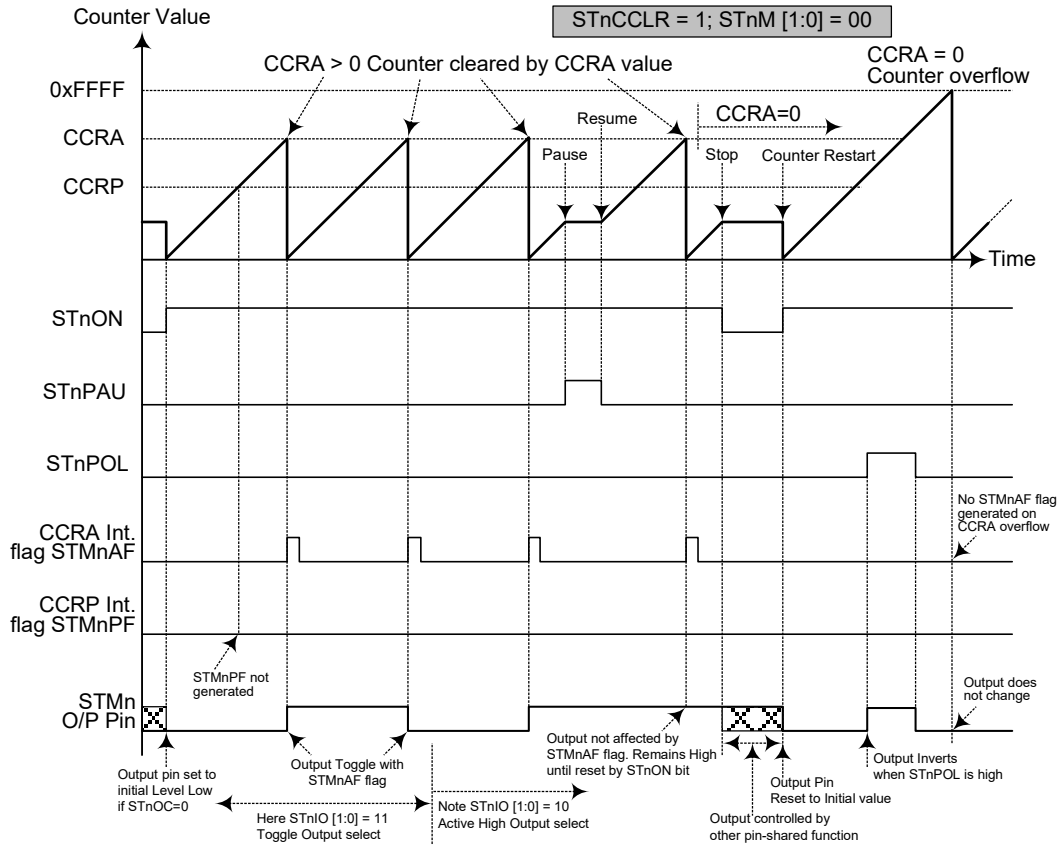
如果 STMnC1 寄存器的 STnCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅产生 STMnAF 中断请求标志。所以当 STnCCLR 为高时，不会产生 STMnPF 中断请求标志。在比较匹配输出模式下，CCRA 不能设为“0”。

正如该模式名所言，当比较匹配发生后，TM 输出脚状态改变。当比较器 A 比较匹配发生后 STMnAF 标志产生时，TM 输出脚状态改变。比较器 P 比较匹配发生时产生的 STMPF 标志不影响 TM 输出脚。TM 输出脚状态改变方式由 STMnC1 寄存器中 STnIO1 和 STnIO0 位决定。当比较器 A 比较匹配发生时，STnIO1 和 STnIO0 位决定 TM 输出脚输出高，低或翻转当前状态。STMn 输出脚初始值，在 STnON 位由低到高电平的变化后通过 STnOC 位设置。注意，若 STnIO1 和 STnIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 – STnCCLR=0

- 注：1. STnCCLR=0，比较器 P 匹配将清除计数器
 2. STMn 输出脚仅由 STMnAF 标志位控制
 3. 在 STnON 上升沿 TM 输出脚复位至初始值
 4. n=0~2



比较匹配输出模式 – STnCCLR=1

- 注：1. STnCCLR=1，比较器 A 匹配将清除计数器
2. STMn 输出脚仅由 STMnAF 标志位控制
3. 在 STnON 上升沿 TM 输出脚复位至初始值
4. 当 STnCCLR=1 时，不会产生 STMnPF 标志位
5. n=0~2

定时 / 计数器模式

为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 位需要设置为“11”。定时 / 计数器模式与比较输出模式操作方式相同，并产生同样的中断请求标志。不同的是，在定时 / 计数器模式下 STMn 输出脚未使用。因此，比较匹配输出模式中的描述和时序图可以适用于此功能。该模式中未使用的 STMn 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 位需要设置为“10”，且 STnIO1 和 STnIO0 位也需要设置为“10”。STMn 的 PWM 功能在马达控制，加热控制，照明控制等方面十分有用。给 STMn 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就较为灵活。在 PWM 输出模式中，STCCLR 位不影响 PWM 周期。CCRA 和 CCRP 寄存器决定 PWM 波形，一个用来清除内部计数器并控制 PWM 波形的频率，另一个用来控制占空比。哪个寄存器控制频率或占空比取决于 STMnC1 寄存器的 STnDPX 位。所以 PWM 波形由 CCRA 和 CCRP 寄存器共同决定。

当比较器 A 或比较器 P 比较匹配发生时，将产生 CCRA 或 CCRP 中断标志。STMnC1 寄存器中的 STnOC 位决定 PWM 波形的极性，STnIO1 和 STnIO0 位使能 PWM 输出或将 TM 输出脚置为逻辑高或逻辑低。STnPOL 位对 PWM 输出波形的极性取反。

- 16-bit STMn, PWM 输出模式, 边沿对齐模式, STnDPX=0

CCRP	1~255	0
Period	CCRP × 256	65536
Duty	CCRA	

若 $f_{sys}=16\text{MHz}$, STMn 时钟源选择 $f_{sys}/4$, CCRP=2, CCRA=128,

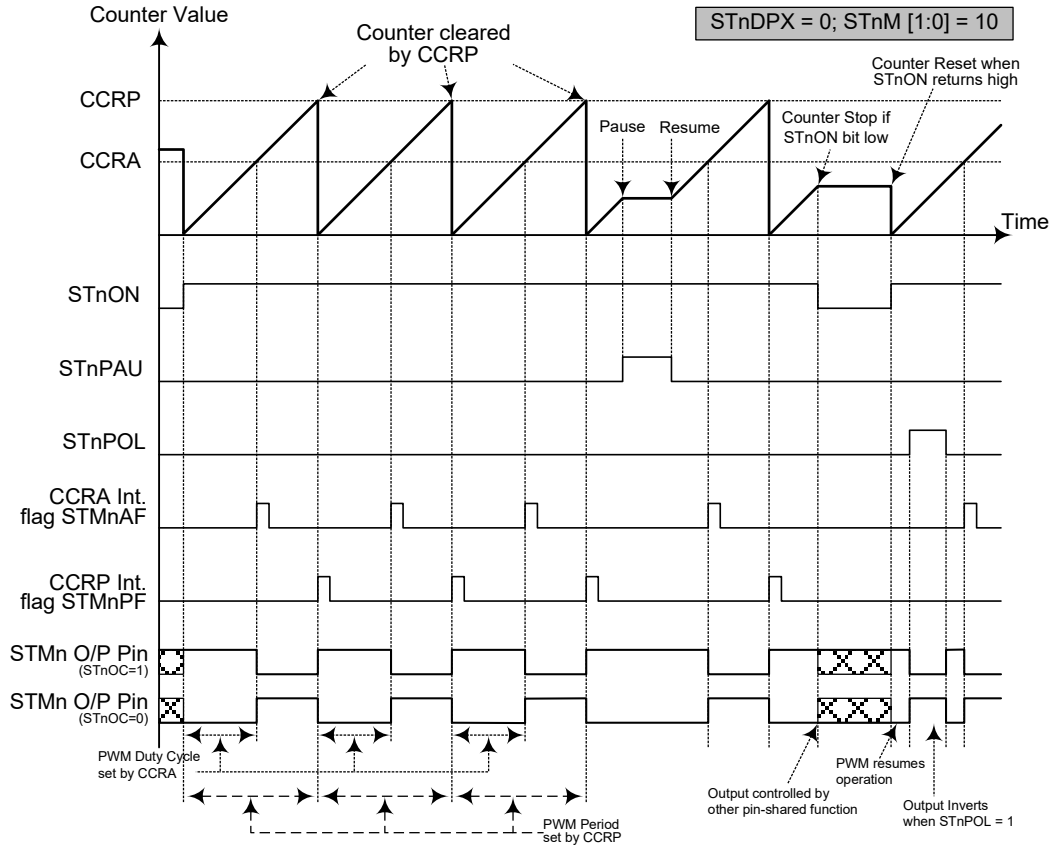
STMn PWM 输出频率 = $(f_{sys}/4)/(2 \times 256) = f_{sys}/2048 = 8\text{kHz}$, $duty = 128/(2 \times 256) = 25\%$

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

- 16-bit STMn, PWM 输出模式, 边沿对齐模式, STnDPX=1

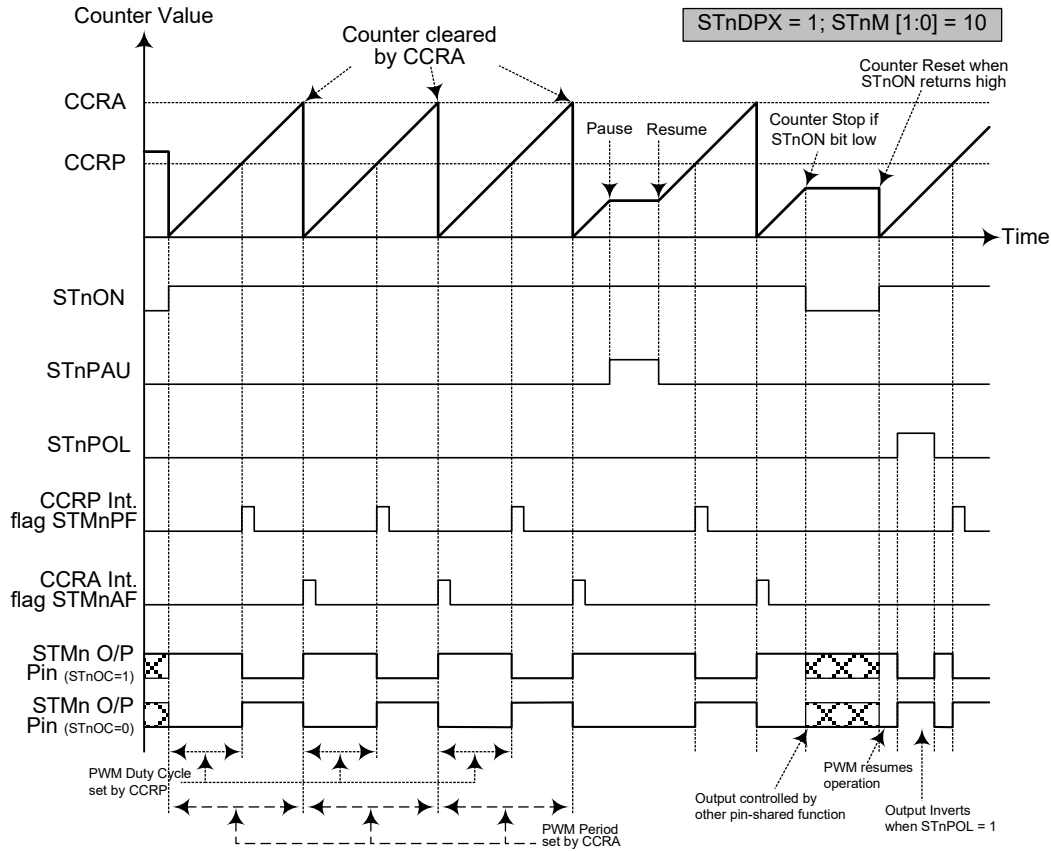
CCRP	1~255	0
Period	CCRA	
Duty	CCRP × 256	65536

PWM 的输出周期由 CCRA 寄存器的值与 STM 的时钟共同决定, PWM 的占空比由 CCRP × 256 (除了 CCRP 为“0”外) 的值决定。



PWM 输出模式 – STnDPX=0

- 注：1. STnDPX=0，CCRP 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 STnIO[1:0]=00 或 01，PWM 功能不变
4. STnCCLR 位不影响 PWM 操作
5. n=0~2



PWM 输出模式 – STnDPX=1

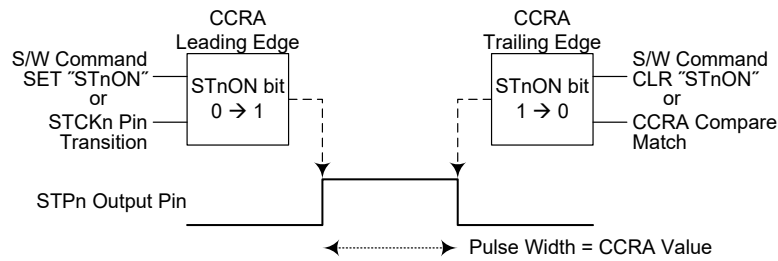
- 注: 1. STnDPX=1, CCRA 清除计数器
 2. 计数器清零并设置 PWM 周期
 3. 当 STnIO[1:0]=00 或 01, PWM 功能不变
 4. STnCCLR 位不影响 PWM 操作
 5. n=0~2

单脉冲输出模式

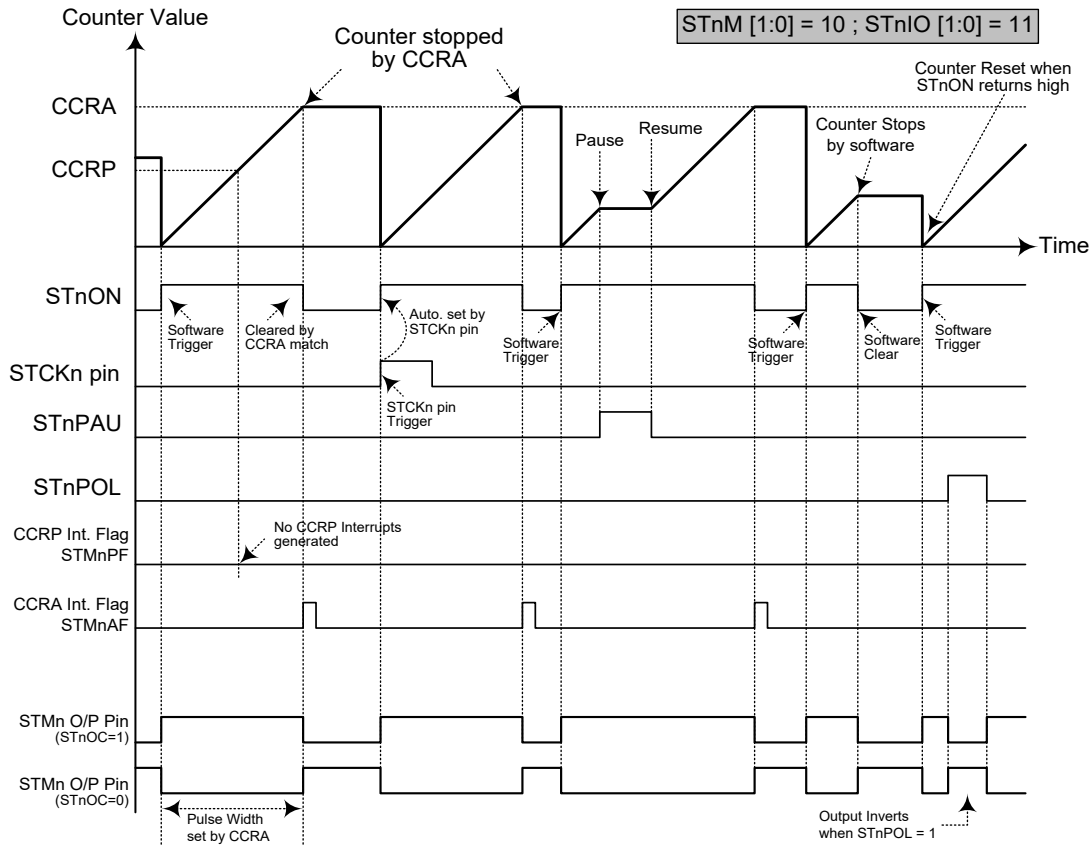
为使 TM 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 位需要设置为“10”，同时 STnIO1 和 STnIO0 位需要设置为“11”。正如模式名所言，单脉冲输出模式，在 STMn 输出脚将产生一个脉冲输出。

脉冲输出可以通过应用程序控制 STnON 位由低到高的转变来触发。而处于单脉冲输出模式时，STnON 位可由 STnCK 脚自动由低转变为高，进而初始化单脉冲输出状态。当 STnON 位转变为高电平时，计数器将开始运行，并产生脉冲前沿。当脉冲有效时 STnON 位保持高电平。通过应用程序使 STnON 位清零或比较器 A 比较匹配发生时，产生脉冲下降沿。

然而，比较器 A 比较匹配发生时，会自动清除 STnON 位并产生单脉冲输出下降沿。CCRA 的值通过这种方式控制脉冲宽度。比较器 A 比较匹配发生时，也会产生 STMn 中断。STnON 位在计数器重启时会发生由低到高的转变，此时计数器才复位至零。在单脉冲输出模式中，CCRP 寄存器，STnCCLR 和 STnDPX 位未使用。



单脉冲产生示意图



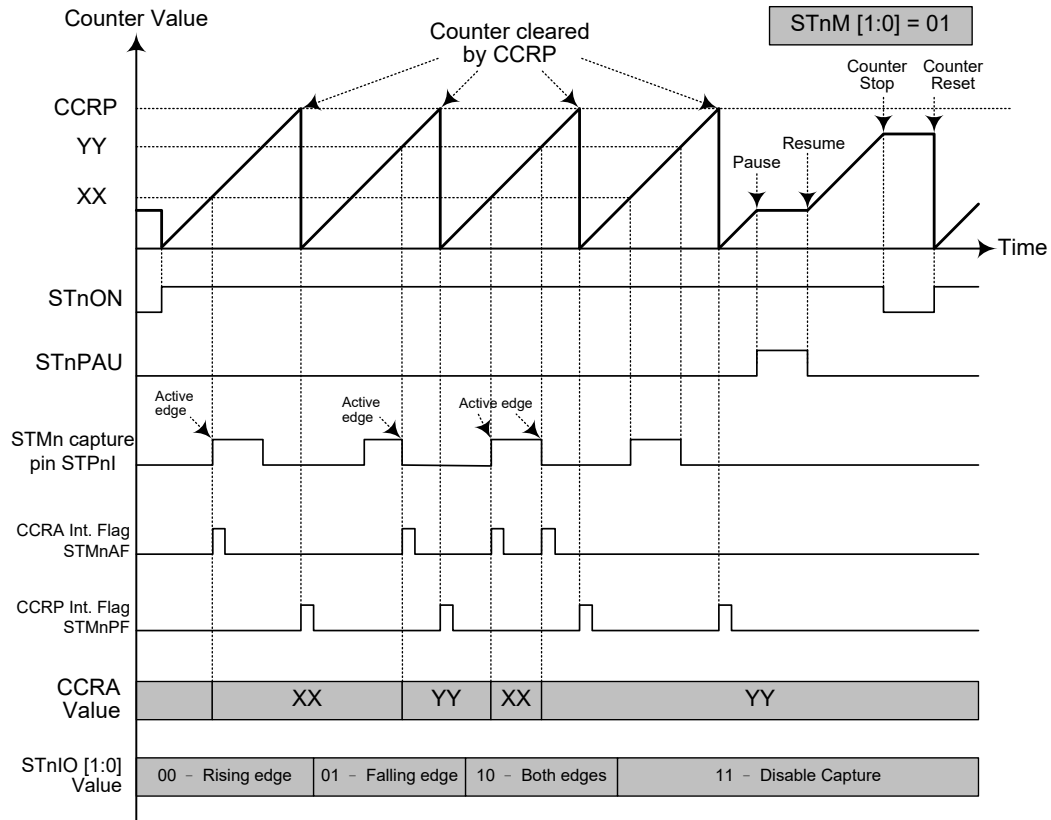
单脉冲输出模式

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

捕捉输入模式

为使 TM 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 位需要设置为“01”。此模式使能外部信号捕捉并保存内部计数器当前值，因此被用于诸如脉冲宽度测量的应用中。STPnI 脚上的外部信号，通过设置 STMnC1 寄存器的 STnIO1 和 STnIO0 位选择有效边沿类型，即上升沿，下降沿或双沿有效。通过应用程序将 STnON 位由低置为高时，计数器启动。

当 STPnI 脚出现有效边沿转换时，计数器当前值被锁存到 CCRA 寄存器，并产生 STMn 中断。不考虑 STPnI 引脚事件，计数器继续工作直到 STnON 位发生下降沿跳变。当 CCRP 比较匹配发生时计数器复位至零；CCRP 的值通过这种方式控制计数器的最大值。当比较器 P CCRP 比较匹配发生时，也会产生 STMn 中断。记录 CCRP 溢出中断信号的值可以测量长脉宽。通过设置 STnIO1 和 STnIO0 位选择 STPnI 引脚为上升沿，下降沿或双沿有效。如果 STnIO1 和 STnIO0 都设置为高，无论 STPnI 引脚发生哪种边沿转换都不会产生捕捉操作，但计数器仍会继续运行。当 STPnI 引脚与其它功能共用，STMn 工作在输入捕捉模式时需多加注意。这是因为如果引脚被设为输出，那么该引脚上的任何电平转变都可能执行输入捕捉操作。STnCCLR 和 STnDPX 位在此模式中未使用。



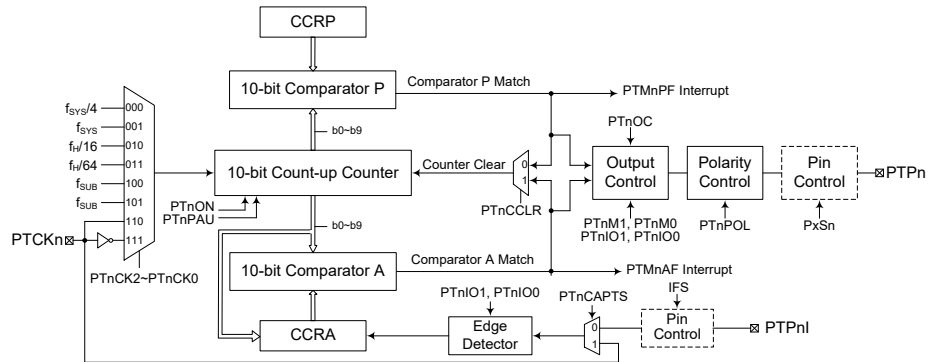
捕捉输入模式

- 注：1. STnM[1:0]=01 并通过 STnIO1 和 STnIO0 位设置有效边沿
 2. STMn 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中
 3. STnCCLR 位未使用
 4. 无输出功能 – STnOC 和 STnPOL 位未使用
 5. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大
 6. n=0~2

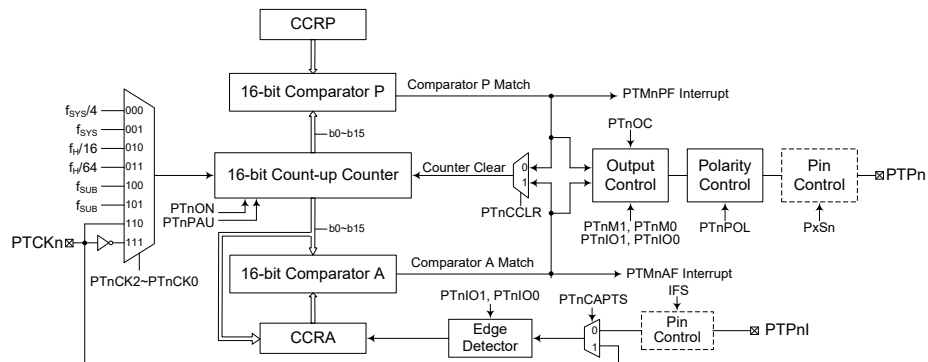
周期型 TM – PTM

周期型 TM 包括 5 种工作模式，即比较匹配输出、定时 / 事件计数器、捕捉输入、单脉冲输出和 PWM 输出模式。周期型 TM 由两个外部输入脚控制并驱动两个外部输出脚。

单片机型号	PTM 核心	PTM 输入引脚	PTM 输出引脚
HT66F2350	10-bit PTM (PTM0, PTM1)	PTCK0, PTP0I	PTP0
HT66F2360		PTCK1, PTP1I	PTP1
HT66F2370	16-bit PTM (PTM2, PTM3)	PTCK2, PTP2I	PTP2
HT66F2390		PTCK3, PTP3I	PTP3



10-bit 周期型 TM 方框图 (n=0~1)



16-bit 周期型 TM 方框图 (n=2~3)

周期型 TM 操作

周期型 TM 的核心是一个由用户选择的内部或外部时钟源驱动的 10-/16-bit 向上计数器，它还包括两个内部比较器即比较器 A 和比较器 P。这两个比较器将计数器的值与 CCRP 和 CCRA 寄存器中的值进行比较。CCRA 和 CCRP 比较器有 10 或 16 宽度。

通过应用程序改变 10-/16-bit 计数器值的唯一方法是使 PTnON 位发生上升沿跳变清除计数器。此外，计数器溢出或比较匹配也会自动清除计数器。上述条件发生时，通常情况会产生 PTM 中断信号。周期型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，也可以控制输出脚。所有工作模式的设置都是通过设置相关寄存器来实现的。

周期型 TM 寄存器描述

周期型 TM 的所有操作由一系列寄存器控制。一对只读寄存器用来存放 10-/16-bit 计数器的值，两对读/写寄存器存放 10-/16-bit CCRA 和 CCRP 的值。剩下两个控制寄存器用来设置不同的操作和工作模式。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnRPL	PTnRP7	PTnRP6	PTnRP5	PTnRP4	PTnRP3	PTnRP2	PTnRP1	PTnRP0
PTMnRPH	—	—	—	—	—	—	PTnRP9	PTnRP8

10-bit 周期型 TM 寄存器列表 (n=0~1)

寄存器名称	位							
	7	6	5	4	3	2	1	0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	D15	D14	D13	D12	D11	D10	D9	D8
PTMnRPL	PTnRP7	PTnRP6	PTnRP5	PTnRP4	PTnRP3	PTnRP2	PTnRP1	PTnRP0
PTMnRPH	PTnRP15	PTnRP14	PTnRP13	PTnRP12	PTnRP11	PTnRP10	PTnRP9	PTnRP8

16-bit 周期型 TM 寄存器列表 (n=2~3)

● PTMnDL 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 PTMn 计数器低字节寄存器 bit 7~bit 0
PTMn 10-/16-bit 计数器 bit 7~bit 0

PTMnDH 寄存器 (n=0~1)

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

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

● PTMnDH 寄存器 (n=2~3)

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 PTMn 计数器高字节寄存器 bit 7~bit 0
PTMn 16-bit 计数器 bit 15~bit 8

● PTMnAL 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 PTMn CCRA 低字节寄存器 bit 7~bit 0
PTMn 10-/16-bit CCRA bit 7~bit 0

● PTMnAH 寄存器 (n=0~1)

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

Bit 7~2 未定义, 读为“0”
Bit 1~0 PTMn CCRA 高字节寄存器 bit 1~bit 0
PTMn 10-bit CCRA bit 9~bit 8

• **PTMnAH 寄存器 (n=2~3)**

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

Bit 7~0 PTMn CCRA 高字节寄存器 bit 7~bit 0
 PTMn 16-bit CCRA bit 15~bit 8

• **PTMnRPL 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PTnRP7	PTnRP6	PTnRP5	PTnRP4	PTnRP3	PTnRP2	PTnRP1	PTnRP0
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 **PTnRP7~PTnRP0**: PTMn CCRP 低字节寄存器 bit 7~bit 0
 PTMn 10-/16-bit CCRP bit 7~bit 0

• **PTMnRPH 寄存器 (n=0~1)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PTnRP9	PTnRP8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义, 读为“0”

Bit 1~0 **PTnRP9~PTnRP8**: PTMn CCRP 高字节寄存器 bit 1~bit 0
 PTMn 10-bit CCRP bit 9~bit 8

• **PTMnRPH 寄存器 (n=2~3)**

Bit	7	6	5	4	3	2	1	0
Name	PTnRP15	PTnRP14	PTnRP13	PTnRP12	PTnRP11	PTnRP10	PTnRP9	PTnRP8
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 **PTnRP15~PTnRP8**: PTMn CCRP 高字节寄存器 bit 7~bit 0
 PTMn 16-bit CCRP bit 15~bit 8

• **PTMnC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **PTnPAU**: PTMn 计数器暂停控制位

0: 运行
 1: 暂停

通过设置此位为高可使计数器暂停, 清零此位恢复正常计数器操作。当处于暂停条件时, PTMn 保持上电状态并继续耗电。当此位由低到高转变时, 计数器将保留其剩余值, 直到此位再次改变为低电平, 并从此值开始继续计数。

- Bit 6~4 **PTnCK2~PTnCK1**: 选择 PTMn 计数时钟位
- 000: $f_{SYS}/4$
 - 001: f_{SYS}
 - 010: $f_H/16$
 - 011: $f_H/64$
 - 100: f_{SUB}
 - 101: f_{SUB}
 - 110: PTCKn 上升沿
 - 111: PTCKn 下降沿

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

- Bit 3 **PTnON**: PTMn 计数器 On/Off 控制位
- 0: Off
 - 1: On

此位控制 PTMn 的总开关功能。设置此位为高则使能计数器使其运行，清零此位则除能 PTMn。清零此位将停止计数器并关闭 PTMn 减少耗电。当此位经由低到高转变时，内部计数器将复位清零；当此位经由高到低转换时，内部计数器将保持其剩余值，直到此位再次改变为高电平。

若 PTMn 处于比较匹配输出模式时，当 PTnON 位经由低到高转换时，PTMn 输出脚将复位至 PTnOC 位指定的初始值。

- Bit 2~0 未定义，读为“0”

● PTMnC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCCLR
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 **PTnM1~PTnM0**: 选择 PTMn 工作模式位
- 00: 比较匹配输出模式
 - 01: 捕捉输入模式
 - 10: PWM 输出模式或单脉冲输出模式
 - 11: 定时 / 计数器模式

这两位设置 PTMn 需要的工作模式。为了确保操作可靠，PTMn 应在 PTnM1 和 PTnM0 位有任何改变前先关掉。在定时 / 计数器模式，PTMn 输出脚控制必须除能。

- Bit 5~4 **PTnIO1~PTnIO0**: 选择 PTPn 或 PTPnI 引脚功能位

比较匹配输出模式

- 00: 无变化
- 01: 输出低
- 10: 输出高
- 11: 输出翻转

PWM 输出模式 / 单脉冲输出模式

- 00: PWM 输出无效状态
- 01: PWM 输出有效状态
- 10: PWM 输出
- 11: 单脉冲输出

捕捉输入模式

- 00: 在 PTCKn 和 PTPnI 上升沿输入捕捉
- 01: 在 PTCKn 和 PTPnI 下降沿输入捕捉
- 10: 在 PTCKn 和 PTPnI 双沿输入捕捉
- 11: 输入捕捉除能

定时 / 计数器模式

未使用

此两位用于决定在一定条件达到时 PTM 输出脚如何改变状态。这两位值的选择取决于 PTM 运行在何种模式下。

在比较匹配输出模式下，PTnIO1 和 PTnIO0 位决定当从比较器 A 比较匹配输出发生时 PTM 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 PTM 输出脚能设为切换高、切换低或翻转当前状态。若此两位同时为 0 时，这个输出将不会改变。PTM 输出脚的初始值通过 PTMnC1 寄存器的 PTnOC 位设置取得。注意，由 PTnIO1 和 PTnIO0 位得到的输出电平必须与通过 PTnOC 位设置的初始值不同，否则当比较匹配发生时，PTM 输出脚将不会发生变化。在 PTM 输出脚改变状态后，通过 PTnON 位由低到高电平的转换复位至初始值。

在 PWM 输出模式，PTnIO1 和 PTnIO0 用于决定比较匹配条件发生时怎样改变 PTM 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。仅在 PTM 关闭时改变 PTnIO1 和 PTnIO0 位的值是很有必要的。若在 PTM 运行时改变 PTnIO1 和 PTnIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **PTnOC**: PTMn PTPn 输出控制位

比较匹配输出模式

0: 初始低

1: 初始高

PWM 输出模式 / 单脉冲输出模式

0: 低有效

1: 高有效

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

Bit 2 **PTnPOL**: PTMn PTPn 输出极性控制位

0: 同相

1: 反相

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

Bit 1 **PTnCAPTS**: 选择 PTMn 捕捉触发源

0: 来自 PTPnI 引脚

1: 来自 PTCKn 引脚

Bit 0 **PTnCCLR**: 选择 PTMn 计数器清零条件位

0: PTM 比较器 P 匹配

1: PTM 比较器 A 匹配

此位用于选择清除计数器的方法。周期型 TM 包括两个比较器 – 比较器 A 和比较器 P，两者都可以用作清除内部计数器。PTnCCLR 位设为高，计数器在比较器 A 比较匹配发生时被清除；此位设为低，计数器在比较器 P 比较匹配发生或计数器溢出时被清除。计数器溢出清除的方法仅在 CCRP 被清除为 0 时才能生效。PTnCCLR 位在 PWM 输出、单脉冲输出或输入捕捉模式时未使用。

周期型 TM 工作模式

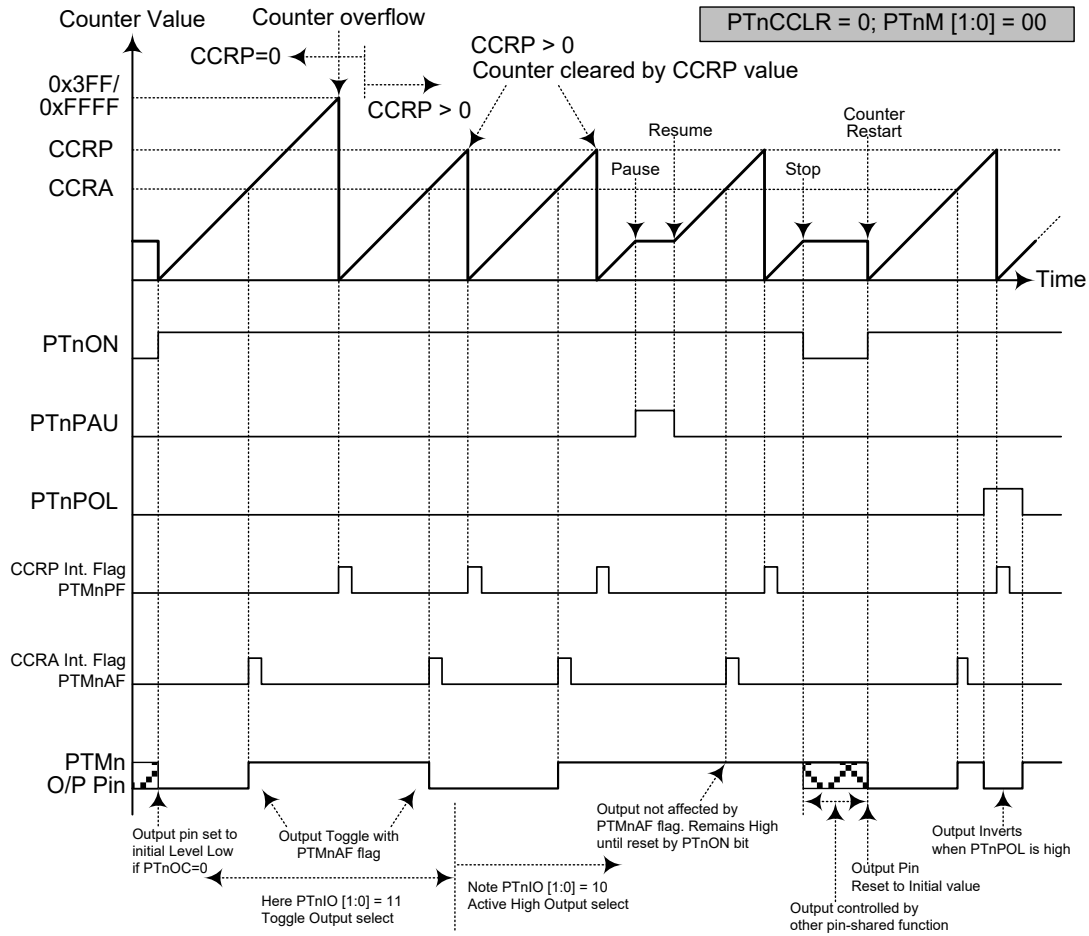
周期型 TM 有五种工作模式，即比较匹配输出模式、PWM 输出模式、单脉冲输出模式、捕捉输入模式或定时 / 计数器模式。通过设置 PTMnC1 寄存器的 PTnM1 和 PTnM0 位选择任意模式。

比较匹配输出模式

为使 PTMn 工作在此模式，PTMnC1 寄存器中的 PTnM1 和 PTnM0 位需要设置为“00”。当工作在该模式，一旦计数器使能并开始计数，有三种方法来清零，分别是：计数器溢出，比较器 A 比较匹配发生和比较器 P 比较匹配发生。当 PTnCCLR 位为低，有两种方法清除计数器。一种是比较器 P 比较匹配发生，另一种是 CCRP 所有位设置为零并使得计数器溢出。此时，比较器 A 和比较器 P 的请求标志位 PTMnAF 和 PTMnPF 将分别置起。

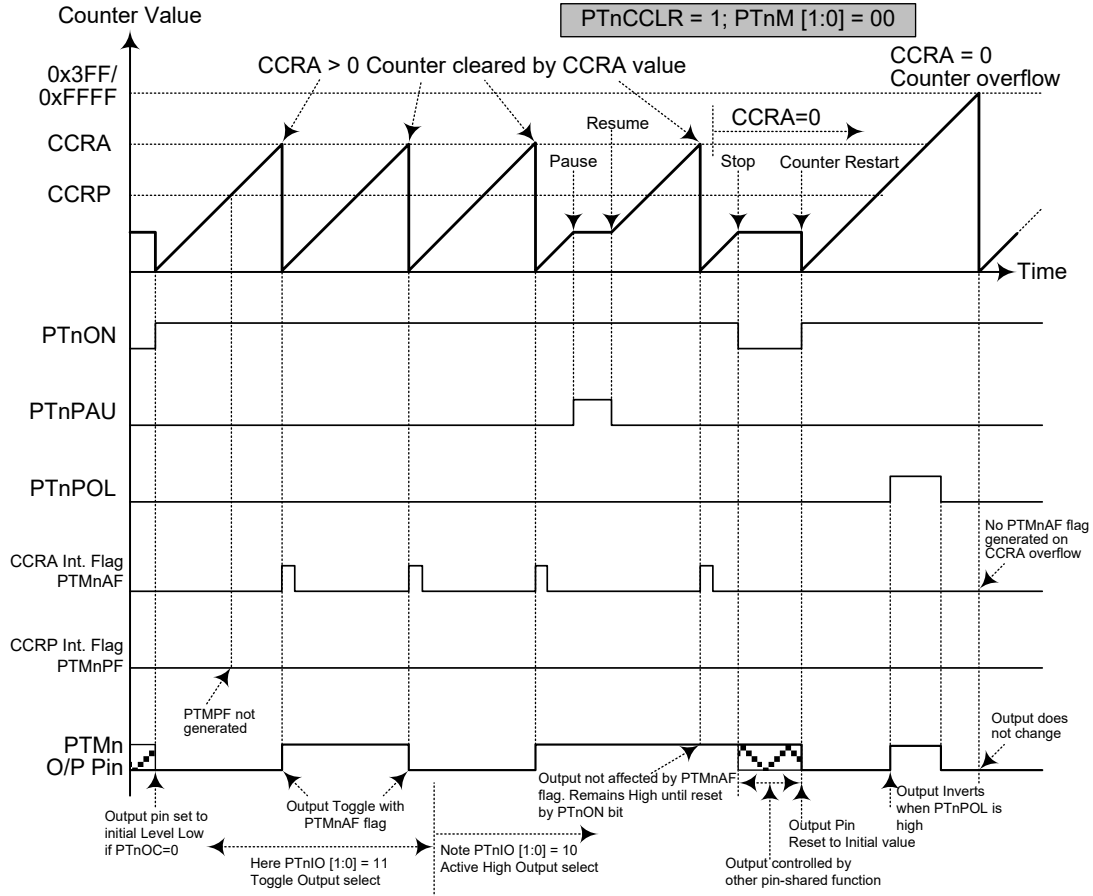
如果 PTMnC1 寄存器的 PTnCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 PTMnAF 中断请求标志产生。所以当 PTnCCLR 为高时，不会产生 PTMnPF 中断请求标志。在比较匹配输出模式中，CCRA 寄存器值不能设为“0”。

正如该模式名所言，当比较匹配发生后，PTM 输出脚状态改变。当比较器 A 比较匹配发生后 PTMnAF 中断请求标志产生时，PTM 输出脚状态改变。比较器 P 比较匹配发生时产生的 PTMnPF 标志不影响 PTM 输出脚。PTM 输出脚状态改变方式由 PTMnC1 寄存器中 PTnIO1 和 PTnIO0 位决定。当比较器 A 比较匹配发生时，PTnIO1 和 PTnIO0 位决定 PTM 输出脚输出高，低或翻转当前状态。PTM 输出脚初始值，在 PTnON 位由低到高电平的变化后通过 PTnOC 位设置。注意，若 PTnIO1 和 PTnIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 – PTnCCLR=0

- 注：1. PTnCCLR=0，比较器 P 匹配将清除计数器
 2. PTMn 输出脚仅由 PTMnAF 标志位控制
 3. 在 PTnON 上升沿 PTMn 输出脚复位至初始值
 4. 10-bit PTM 最大计数器值为 0x3FF；16-bit PTM 最大计数器值为 0xFFFF
 5. 对于 10-bit PTM，n = 0~1；对于 16-bit PTM，n = 2~3



比较匹配输出模式 – PTnCCLR=1

- 注：1. PTnCCLR=1，比较器 A 匹配将清除计数器
2. PTMn 输出脚仅由 PTMnAF 标志位控制
3. 在 PTnON 上升沿 PTMn 输出脚复位至初始值
4. 当 PTnCCLR=1 时，不会产生 PTMnPF 标志
5. 10-bit PTM 最大计数器值为 0x3FF；16-bit PTM 最大计数器值为 0xFFFF
6. 对于 10-bit PTM，n = 0~1；对于 16-bit PTM，n=2~3

定时 / 计数器模式

为使 PTMn 工作在此模式，PTMnC1 寄存器中的 PTnM1 和 PTnM0 位需要设置为“11”。定时 / 计数器模式与比较输出模式操作方式相同，并产生同样的中断请求标志。不同的是，在定时 / 计数器模式下 PTMn 输出脚未使用。因此，比较匹配输出模式中的描述和时序图可以适用于此功能。

PWM 输出模式

为使 PTMn 工作在此模式，PTMnC1 寄存器中的 PTnM1 和 PTnM0 位需要设置为“10”。PTMn 的 PWM 功能在马达控制，加热控制，照明控制等方面十分有用。给 PTMn 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就较为灵活。在 PWM 模式中，PTnCCLR 位对 PWM 周期无影响。CCRP 和 CCRA 寄存器都用于控制 PWM 方波。CCRP 寄存器通过清除内部计数从而控制 PWM 周期，CCRA 寄存器设置 PWM 的占空比。PWM 波形的周期和占空比由 CCRP 和 CCRA 寄存器的值控制。

当比较器 A 或比较器 P 比较匹配发生时，CCRA 和 CCRP 中断标志位分别产生。PTMnC1 寄存器的 PTnOC 位选择 PWM 波形的极性，PTnIO1 和 PTnIO0 位使能 PWM 输出或强制 TM 输出脚为高电平或低电平。PTnPOL 位用于 PWM 输出波形的极性反相控制。

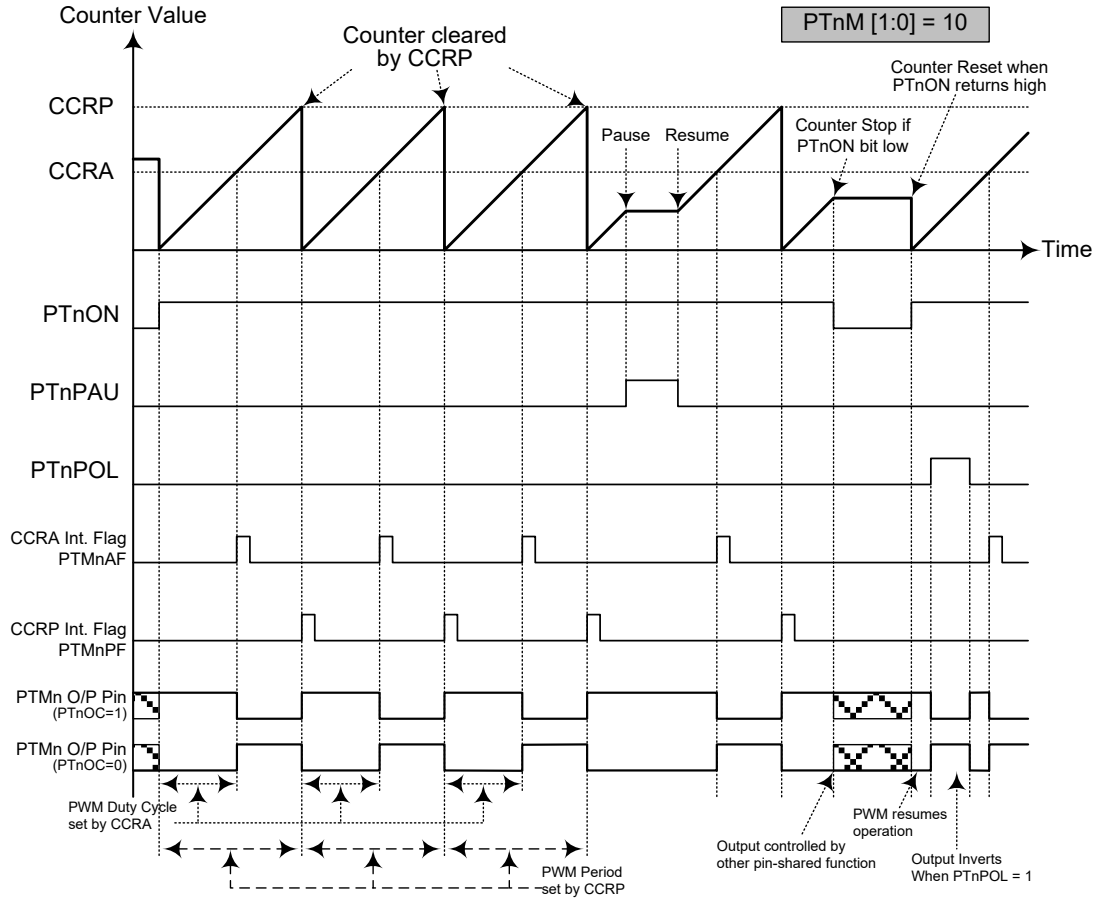
● 16-bit PTMn, PWM 输出模式

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

若 $f_{SYS}=16\text{MHz}$ ，TM 时钟源选择 $f_{SYS}/4$ ，CCRP = 512 且 CCRA=128，

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

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



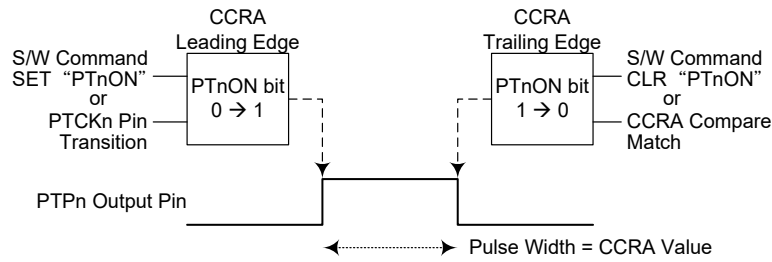
PWM 输出模式

- 注:
1. CCRP 清零计数器
 2. 计数器清除并决定 PWM 周期
 3. 当 $PTnIO[1:0]=00$ 或 01 , PWM 功能不变
 4. $PTnCCLR$ 位对 PWM 功能无影响
 5. 对于 10-bit PTM, $n = 0\sim 1$; 对于 16-bit PTM, $n = 2\sim 3$

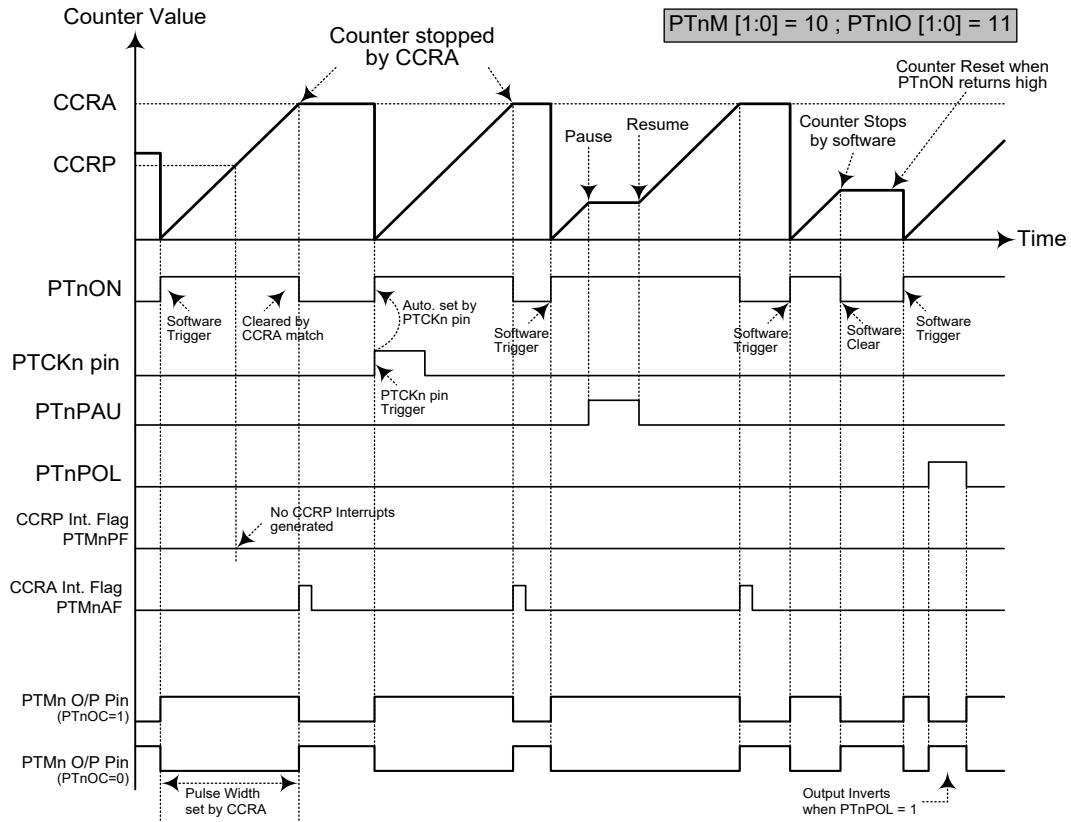
单脉冲输出模式

为使 PTMn 工作在此模式，PTMnC1 寄存器中的 PTnM1 和 PTnM0 位需要设置为“10”，并且相应的 PTnIO1 和 PTnIO0 需要设置为“11”。正如模式名所言，单脉冲输出模式，在 PTMn 输出脚将产生一个脉冲输出。

通过应用程序控制 PTnON 位由低到高的转变来触发脉冲前沿输出。而处于单脉冲模式时，PTnON 位可由 PTCKn 脚自动由低转变为高，进而依次初始化单脉冲输出。当 PTnON 位转变为高电平时，计数器将开始运行，并产生脉冲前沿。通过应用程序使 PTnON 位清零或比较器 A 比较匹配发生时，产生脉冲下降沿。而比较器 A 比较匹配发生时，会自动清除 PTnON 位并产生单脉冲输出下降沿。CCRA 的值通过这种方式控制脉冲宽度。比较器 A 比较匹配发生时，也会产生 PTMn 中断。PTnON 位在计数器重启时会发生由低到高的转变，此时计数器才复位至零。在单脉冲模式中，CCRP 寄存器和 PTnCCLR 位未使用。



单脉冲产生示意图



单脉冲输出模式

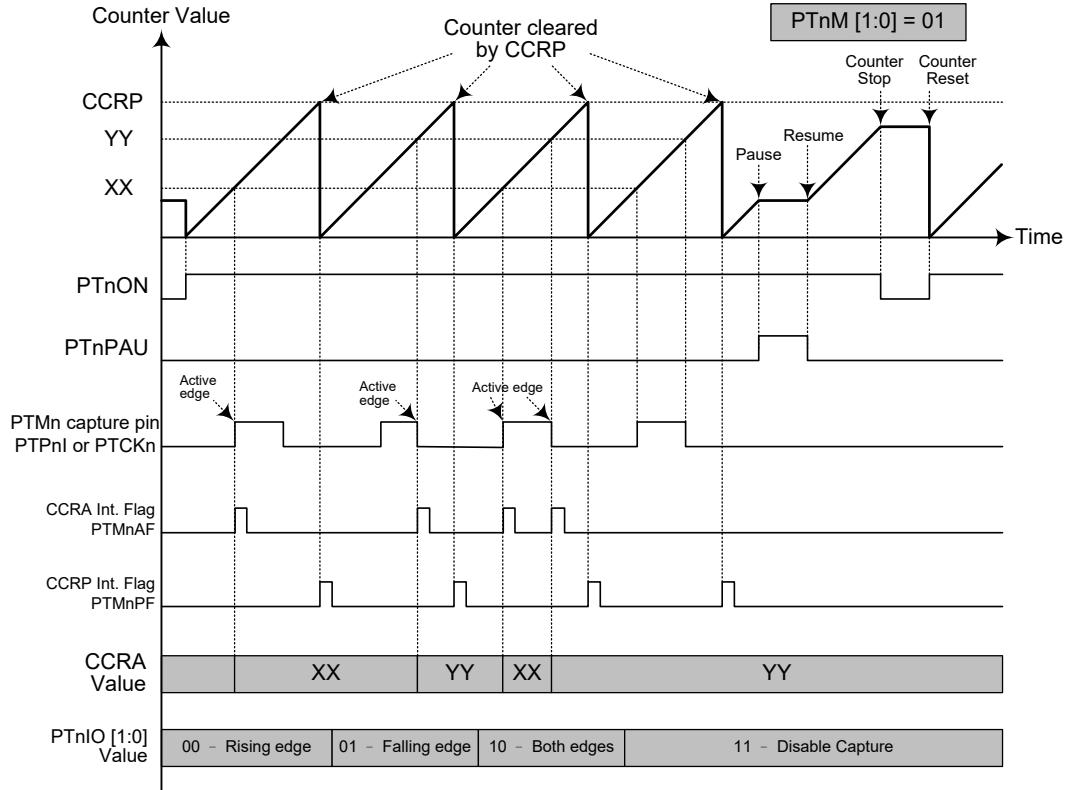
- 注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 PTCKn 脚或设置 PTnON 位为高来触发脉冲
4. PTCKn 脚有效沿会自动置位 PTnON
5. 单脉冲模式中，PTnIO [1:0] 需置位“11”，且不能更改
6. 对于 10-bit PTM，n = 0~1；对于 16-bit PTM，n=2~3

捕捉输入模式

为使 PTMn 工作在此模式，PTMnC1 寄存器中的 PTnM1 和 PTnM0 位需要设置为“01”。此模式使能外部信号捕捉并保存内部计数器当前值，因此被用于诸如脉冲宽度测量的应用中。PTPnI 或 PTCKn 引脚上的外部信号，通过设置 PTMnC1 寄存器的 PTnCAPTS 位选择。可通过设置 PTMnC1 寄存器的 PTnIO1 和 PTnIO0 位选择有效边沿类型，即上升沿，下降沿或双沿有效。通过应用程序将 PTnON 位由低到高转变时，计数器启动。

当 PTPnI 或 PTCKn 引脚出现有效边沿转换时，计数器当前值被锁存到 CCRA 寄存器，并产生 PTMn 中断。无论 PTPnI 或 PTCKn 引脚发生哪种边沿转换，计数器将继续工作直到 PTnON 位发生下降沿跳变。当 CCRP 比较匹配发生时计数器复位至零；CCRP 的值通过这种方式控制计数器的最大值。当比较器 PCCRP 比较匹配发生时，也会产生 PTMn 中断。记录 CCRP 溢出中断信号的值可以测量长脉宽。通过设置 PTnIO1 和 PTnIO0 位选择 PTPnI 或 PTCKn 引脚为上升沿，下降沿或双沿有效。如果 PTnIO1 和 PTnIO0 位都设置为高，无论 PTPnI 或 PTCKn 引脚发生哪种边沿转换都不会产生捕捉操作，但计数器仍会继续运行。

当 PTPnI 或 PTCKn 引脚与其它功能共用，PTMn 工作在输入捕捉模式时需多加注意。这是因为如果引脚被设为输出，那么该引脚上的任何电平转变都可能执行输入捕捉操作。PTnCCLR，PTnOC 和 PTnPOL 位在此模式中未使用。



捕捉输入模式

- 注：1. PTnM [1:0]=01 并通过 PTnIO [1:0] 位设置有效边沿
2. PTMn 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中
3. PTnCCLR 位未使用
4. 无输出功能 – PTnOC 和 PTnPOL 位未使用
5. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大
6. 对于 10-bit PTM，n = 0~1；对于 16-bit PTM，n=2~3

A/D 转换器

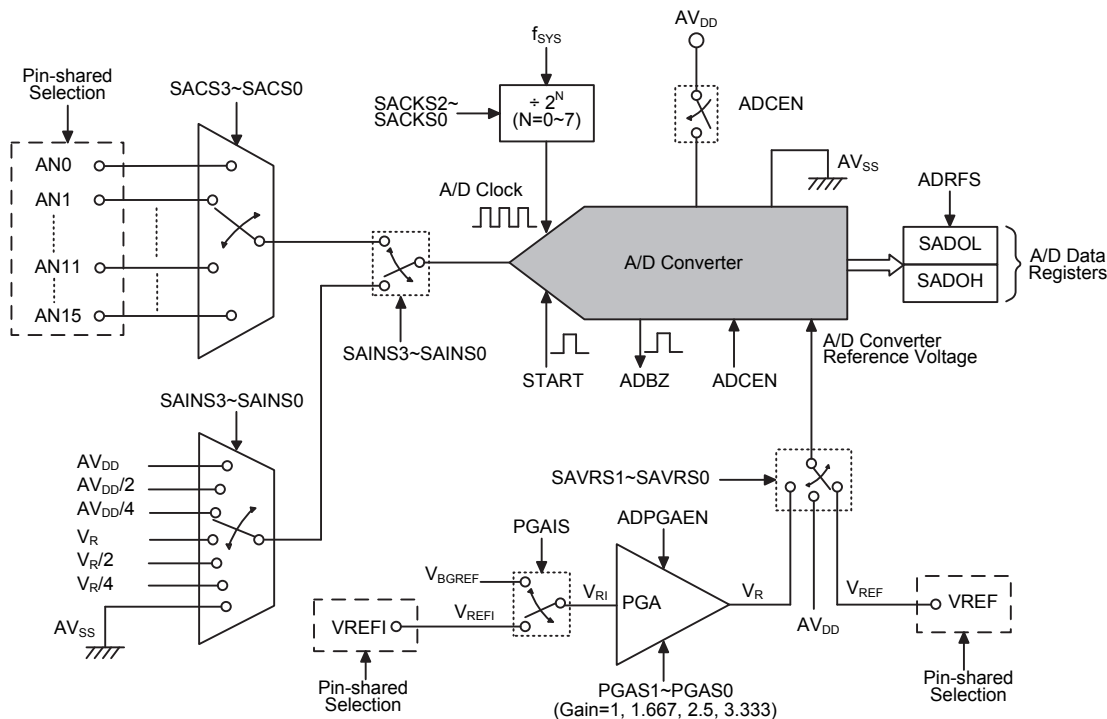
对于大多数电子系统而言，处理现实世界的模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先需要通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

A/D 转换器简介

该系列单片机包含了多通道的 A/D 转换器，它们可以直接接入外部模拟信号，如来自传感器或其它控制信号，并直接将这些信号转换成 12-bit 数字量。也可对内部模拟信号，如内部参考电压等进行 A/D 转换。选择转换外部或内部模拟信号由 SAINS 位段和 SACS 位段控制。应注意的是，若通过 SAINS 位段选择内部模拟信号，则外部通道模拟输入将被自动关闭。关于 A/D 输入信号的更多信息请参见“A/D 转换器输入信号”章节。

下图显示了带温度传感器的 A/D 转换器的内部结构，以及相关的寄存器与控制位。

单片机型号	外部输入通道	内部信号	A/D 信号选择
HT66F2350	AN0~AN11	AV _{DD} , AV _{DD} /2, AV _{DD} /4, V _R , V _R /2, V _R /4	SAINS3~SAINS0 SACS3~SACS0
HT66F2360	AN0~AN15		
HT66F2370			
HT66F2390			



注：HT66F2350 单片机的外部通道为 AN0~AN11。

A/D 转换器结构

A/D 转换器寄存器介绍

A/D 转换器的所有操作由六个寄存器控制。一对只读寄存器用来存放 12-bit ADC 数据的值。剩下三个控制寄存器，即 SADC0，SADC1 和 SADC2 寄存器，用于设置 A/D 转换器的操作和控制功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SADOL (ADRF5=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRF5=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRF5=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRF5=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRF5	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
SADC2	ADPGAEN	—	—	PGAIS	SAVRS1	SAVRS0	PGAGS1	PGAGS0
VBGRC	—	—	—	—	—	—	—	VBGREN

A/D 转换器寄存器列表

A/D 转换器数据寄存器 – SADOL, SADOH

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

ADRF5	SADOH								SADOL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D 转换器数据寄存器

A/D 转换器控制寄存器 – SADC0, SADC1, SADC2

寄存器 SADC0、SADC1 和 SADC2 用来控制 A/D 转换器的功能和操作。这些 8-bit 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，并控制和监测 A/D 转换器的开始和转换忙状态。由于该系列单片机均只包含一个实际的模数转换硬件电路，因此外部或内部模拟输入中的每一个都需要分别被发送到转换器。寄存器 SADC1 的 SAINS 位段和寄存器 SADC0 的 SACS 位段用来决定模拟信号是来自外部或内部信号。A/D 转换器还包含了一个可编程增益放大器 PGA 用于产生 A/D 转换器内部参考电压。PGA 的所有操作受 SADC2 寄存器控制。

相关引脚共用功能选择位用来定义 I/O 端口中的哪些引脚为 A/D 转换器的模拟输入，哪些引脚不作为 A/D 转换输入。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。

● SADC0 寄存器 – HT66F2350

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

- Bit 7 **START**: 启动 A/D 转换位
 0→1→0: 启动 A/D 转换
 此位用于启动 A/D 转换过程。通常此位为低，但如果设为高再被清零，将启动 A/D 转换过程。
- Bit 6 **ADBZ**: A/D 转换忙标志位
 0: A/D 转换结束或未开始转换
 1: A/D 转换中
 该只读标志位用于表明 A/D 转换过程是否完成。当 START 位由低变为高再变为低时，ADBZ 位为高，表明 A/D 转换已启动。A/D 转换结束后，此位被清零。
- Bit 5 **ADCEN**: A/D 转换器功能使能控制位
 0: 除能
 1: 使能
 此位控制 A/D 内部功能。该位被置高将使能 A/D 转换器。如果该位设为低将关闭 A/D 转换器以降低功耗。当 A/D 转换器功能除能时，A/D 数据寄存器 SADOH 和 SADOL 的内容将保持不变。
- Bit 4 **ADRFS**: A/D 转换器数据格式控制位
 0: A/D 转换器数据格式 → SADOH=D[11:4]; SADOL=D[3:0]
 1: A/D 转换器数据格式 → SADOH=D[11:8]; SADOL=D[7:0]
 此位控制存放在两个 A/D 数据寄存器中的 12-bit A/D 转换结果的格式。细节方面请参考 A/D 转换器数据寄存器章节。
- Bit 3~0 **SACS3~SACS0**: A/D 转换器外部模拟输入通道选择位
 0000: AN0
 0001: AN1
 0010: AN2
 0011: AN3
 0100: AN4
 0101: AN5
 0110: AN6
 0111: AN7
 1000: AN8
 1001: AN9
 1010: AN10
 1011: AN11
 11xx: 未定义，若选择此项则输入为浮空
 此位段用于选择要转换的外部模拟输入通道。当选择外部模拟输入通道，则 SAINS 位段必须置为“0000”，“0100”或“11xx”。更多详细信息请参见“A/D 转换器输入信号选择”表格。

● SADC0 寄存器 – HT66F2360/HT66F2370/HT66F2390

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

- Bit 7 START:** 启动 A/D 转换位
0→1→0: 启动 A/D 转换
此位用于启动 A/D 转换过程。通常此位为低，但如果设为高再被清零，将启动 A/D 转换过程。
- Bit 6 ADBZ:** A/D 转换忙标志位
0: A/D 转换结束或未开始转换
1: A/D 转换中
该只读标志位用于表明 A/D 转换过程是否完成。当 START 位由低变为高再变为低时，ADBZ 位为高，表明 A/D 转换已启动。A/D 转换结束后，此位被清零。
- Bit 5 ADCEN:** A/D 转换器功能使能控制位
0: 除能
1: 使能
此位控制 A/D 内部功能。该位被置高将使能 A/D 转换器。如果该位设为低将关闭 A/D 转换器以降低功耗。当 A/D 转换器功能除能时，A/D 数据寄存器 SADOH 和 SADOL 的内容将保持不变。
- Bit 4 ADRFS:** A/D 转换器数据格式控制位
0: A/D 转换器数据格式 → SADOH=D[11:4]; SADOL=D[3:0]
1: A/D 转换器数据格式 → SADOH=D[11:8]; SADOL=D[7:0]
此位控制存放在两个 A/D 数据寄存器中的 12-bit A/D 转换结果的格式。细节方面请参考 A/D 转换器数据寄存器章节。
- Bit 3~0 SACS3~SACS0:** A/D 转换器外部模拟输入通道选择位
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6
0111: AN7
1000: AN8
1001: AN9
1010: AN10
1011: AN11
1100: AN12
1101: AN13
1110: AN14
1111: AN15
此位段用于选择要转换的外部模拟输入通道。当选择外部模拟输入通道，则 SAINS 位段必须置为“0000”，“0100”或“11xx”。更多详细信息请参见“A/D 转换器输入信号选择”表格。

● SADC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
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 7~4 **SAINS3~SAINS0**: A/D 转换器输入信号选择

0000: 外部来源 – 外部模拟通道输入, ANn

0001: 内部来源 – AV_{DD}

0010: 内部来源 – AV_{DD}/2

0011: 内部来源 – AV_{DD}/4

0100: 外部来源 – 外部模拟通道输入, ANn

0101: 内部来源 – PGA 输出 V_R

0110: 内部来源 – PGA 输出 V_R/2

0111: 内部来源 – PGA 输出 V_R/4

10xx: 内部来源 – 接地

11xx: 外部来源 – 外部模拟通道输入, ANn

当选择转换内部模拟信号, 无论 SACKS 位段的值设定为多少, 外部通道信号输入将自动关闭。此举预防了外部通道输入与内部模拟信号连接从而导致的不可预期的后果。

Bit 3 未定义, 读为 “0”

Bit 2~0 **SACKS2~SACKS0**: A/D 转换器时钟源选择位

000: f_{SYS}

001: f_{SYS}/2

010: f_{SYS}/4

011: f_{SYS}/8

100: f_{SYS}/16

101: f_{SYS}/32

110: f_{SYS}/64

111: f_{SYS}/128

此位段用于选择 A/D 转换器时钟源。建议 A/D 转换时钟应通过 SACKS2~SACKS0 位段正确配置在 500kHz~1MHz 的范围内。

● SADC2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ADPGAEN	—	—	PGAIS	SAVRS1	SAVRS0	PGAGS1	PGAGS0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7 **ADPGAEN**: PGA 使能控制

0: 除能

1: 使能

Bit 6~5 未定义, 读为 “0”

Bit 4 **PGAIS**: PGA 输入电压选择

0: 来自 VREFI 引脚

1: 来自内部参考电压 V_{BGREF}

Bit 3~2 **SAVRS1~SAVRS0**: A/D 转换器参考电压选择

00: 内部 A/D 转换器电源 AV_{DD}

01: 外部 VREF 引脚

1x: 内部 PGA 输出电压 V_R

该位段用于选择 A/D 转换器参考电压源。当选中内部参考电压源, 来自外部 VREF 引脚的参考电压将被自动关闭。

Bit 1~0 **PGAGS1~PGAGS0**: PGA 增益选择

- 00: Gain=1
- 01: Gain=1.667 ($V_R = 2V$, $V_{RI} = 1.2V$)
- 10: Gain=2.5 ($V_R = 3V$, $V_{RI} = 1.2V$)
- 11: Gain=3.333 ($V_R = 4V$, $V_{RI} = 1.2V$)

此位段用于选择PGA增益。应注意此处增益仅在PGA输入电压为1.2V时有保证。

• VBGRC 寄存器

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

Bit 7~1 未定义，读为“0”

Bit 0 **VBGREN**: Bandgap 参考电压控制

- 0: 除能
- 1: 使能

此位用于使能内部 Bandgap 参考电路。在选用 V_{BGREF} 电压之前应预先使能内部 Bandgap 参考电路。在该电路稳定并达到精确之前需要等待一定的启动时间。

A/D 转换器参考电压

A/D 转换器的参考电压可以来自正电源引脚 AVDD 或来自 VREF 引脚上的外部参考源或是内部参考电压 V_R ，通过配置 SADC2 寄存器中的 SAVRS1 和 SAVRS0 位进行选择。内部参考电压通过可编程增益放大器 PGA 进行放大，PGA 由 SADC2 寄存器中的 ADPGAEN 位控制。PGA 增益可为 1, 1.667, 2.5 或 3.333，通过 SADC2 寄存器中的 PGAGS1~PGAGS0 位段选择。PGA 输入可来自外部参考输入引脚 VREFI 或内部 Bandgap 参考电压 V_{BGREF} ，通过 SADC2 寄存器中的 PGAIS 位进行选择。由于 VREFI 和 VREFI 引脚均与其它功能共用引脚，当这两个引脚被选作参考电压引脚时，相应的引脚共用功能选择位应被预先设置以除能其它引脚共用功能。但是若选择内部参考信号作为参考源，则来自 VREFI 或 VREF 引脚的外部参考输入将由硬件自动关闭。

应注意在选用 V_{BGREF} 之前应预先使能内部 Bandgap 参考电路。在 Bandgap 参考电路稳定并达到精确之前需要等待一定的启动时间。

A/D 转换器输入信号

A/D 转换器的所有模拟输入引脚与 I/O 端口上的引脚及其它功能共用。PxS1 和 PXS0 寄存器中相应的引脚共用功能选择位决定了内部引脚是设置为 A/D 转换器模拟通道输入还是其它功能。如果相应引脚设置为 A/D 转换器模拟通道输入那么该引脚原引脚功能除能。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，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，当相关 A/D 输入功能选择位使能 A/D 输入时，端口控制寄存器的状态将被重置。

由于该系列单片机均只包含一个实际的模数转换硬件电路，因此外部或内部模拟信号中的每一个都需要分别被发送到转换器。SADC1 寄存器中的 SAINS3~SAINS0 位段用于确定转换信号是来自外部通道输入或内部模拟信号。SADC0 寄存器中的 SACS3~SACS0 位段用于确定所要转换的外部通道输入。若 SAINS3~SAINS0 位段设为“0000”，则所要转换信号为外部模拟通道信号，SACS3~SACS0 位段可决定选择哪个外部通道的信号进行转换。

若 SAINS 位段的值设为“0x01”，“0x10”或“0x11”，则选择内部模拟信号。此时无论 SACS 位段的值配置为多少，外部通道信号输入将被自动关闭，此举将预防外部通道输入将与内部模拟信号连接从而导致的不可预期的后果。

SAINS [3:0]	SACS [3:0]	输入信号	说明
0000, 0100, 11xx	0000~1011 11xx	AN0~AN11 —	外部通道模拟输入, AN _n 浮空, 无选中通道
0001	xxxx	AV _{DD}	内部信号, 来自 AV _{DD}
0010	xxxx	AV _{DD} /2	内部信号, 来自 AV _{DD} /2
0011	xxxx	AV _{DD} /4	内部信号, 来自 AV _{DD} /4
0101	xxxx	V _R	内部信号, 来自 PGA 输出 V _R
0110	xxxx	V _R /2	内部信号, 来自 PGA 输出 V _R /2
0111	xxxx	V _R /4	内部信号, 来自 PGA 输出 V _R /4
10xx	xxxx	GND	接地

A/D 转换器输入信号选择 – HT66F2350

SAINS [3:0]	SACS [3:0]	输入信号	说明
0000, 0100, 11xx	0000~1111	AN0~AN15	外部通道模拟输入, AN _n
0001	xxxx	AV _{DD}	内部信号, 来自 AV _{DD}
0010	xxxx	AV _{DD} /2	内部信号, 来自 AV _{DD} /2
0011	xxxx	AV _{DD} /4	内部信号, 来自 AV _{DD} /4
0101	xxxx	V _R	内部信号, 来自 PGA 输出 V _R
0110	xxxx	V _R /2	内部信号, 来自 PGA 输出 V _R /2
0111	xxxx	V _R /4	内部信号, 来自 PGA 输出 V _R /4
10xx	xxxx	GND	接地

A/D 转换器输入信号选择 – HT66F2360/HT66F2370/HT66F2390

A/D 转换器操作

SADC0 寄存器中的 START 位用于启动 A/D 转换。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。

SADC0 寄存器中的 ADBZ 位用于表明模数转换是否正在进行。A/D 转换成功启动后, ADBZ 位被自动置为“1”。在转换周期结束后, ADBZ 位会被清零。此外, 中断控制寄存器内相应的 A/D 中断请求标志位也会置位, 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 SADC0 寄存器中的 ADBZ 位, 检测此位是否被清除, 以作为另一种侦测 A/D 转换周期结束的方法。

A/D 转换器时钟源来自系统时钟 f_{SYS}, 可选择为 f_{SYS} 或 f_{SYS} 的分频。分频比由 SADC1 寄存器中的 SACKS 位段确定。虽然 A/D 时钟源是由系统时钟 f_{SYS} 和 SACKS2~SACKS0 位决定, 但可选择的最大 A/D 时钟源还有一些限制。由于允许的 A/D 时钟周期 t_{ADCK} 的范围为 0.5μs ~10μs, 所以选择系统时钟速度时就必须小心。如果系统时钟速度为 8MHz 时, SACKS2~SACKS0 位不能设为“000”, “001”或“111”。否则所得的 A/D 时钟周期将小于最小时钟周期或大于最大时钟周期, 导致所得 A/D 转换结果不准确。参考下表, 根据单片机的类型, 表格中加星号“*”的值需多加注意, 因为这些值可能小于所指定的 A/D 最小时钟周期。

但是, 若要转换的输入信号为温度传感器输出电压时, 建议的 A/D 时钟周期范围应为 1μs 到 2μs。

f _{sys}	A/D 时钟周期 (t _{ADCK})							
	SACKS [2:0]=000 (f _{sys})	SACKS [2:0]=001 (f _{sys} /2)	SACKS [2:0]=010 (f _{sys} /4)	SACKS [2:0]=011 (f _{sys} /8)	SACKS [2:0]=100 (f _{sys} /16)	SACKS [2:0]=101 (f _{sys} /32)	SACKS [2:0]=110 (f _{sys} /64)	SACKS [2:0]=111 (f _{sys} /128)
1 MHz	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *	128μs *
2 MHz	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *
4 MHz	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *
8 MHz	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *
12MHz	83ns *	167ns *	333ns *	667ns	1.33μs	2.67μs	5.33μs	10.67μs *
16MHz	62.5ns *	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs

A/D 时钟周期范例

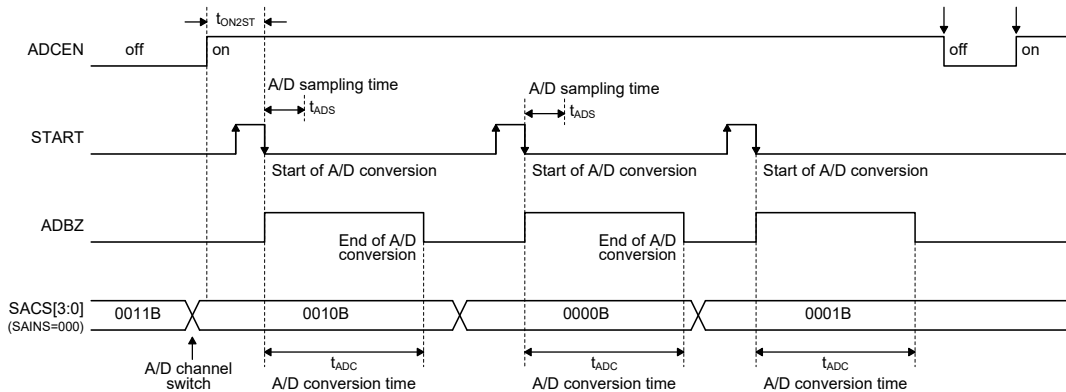
SADC0 寄存器的 ADCEN 位用于控制 A/D 转换电路电源的开启 / 关闭。该位必须置高以开启 A/D 转换器电源。当设置 ADCEN 位为高开启 A/D 转换器内部电路时，在 A/D 转换成功初始化前需一段延时，如时序图所示。即使选择无引脚作为 A/D 输入，若 ADCEN 设为“1”，仍然会产生功耗。因此当未使用 A/D 转换器功能时，在功耗敏感的应用中建议设置 ADCEN 为低以减少功耗。

转换率和时序图

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

$$\text{最大 A/D 转换率} = \text{A/D 时钟周期} \times 16$$

下列时序图表示一个外部通道输入信号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

A/D 转换步骤

下面概述实现 A/D 转换过程的各个步骤。

- 步骤 1
通过 SADC1 寄存器中的 SACKS2~SACKS0 位，选择所需的 A/D 转换器转换时钟。
- 步骤 2
将 SADC0 寄存器中的 ADCEN 位置高以使能 A/D 转换器。
- 步骤 3
通过配置 SACS 和 SAINS 位段，选择连接至内部 A/D 转换器的信号。
选择外部通道输入，前往步骤 4。
选择内部模拟信号，前往步骤 5。
- 步骤 4
若配置 SAINS 位段为“0000”，“0100”或“11xx”，将选择外部通道输入信号为 A/D 输入信号。所需的外部通道输入由 SACS 位段确定。当 A/D 输入信号来自外部通道输入，相应引脚应先通过配置相关的引脚共用功能控制位选择。完成此步骤后，前往步骤 6。
- 步骤 5
若 SAINS 位段的值为“0x01”，“0x10”或“0x11”时选中相关内部模拟信号，则外部通道模拟输入将被自动断开。完成此步骤后，前往步骤 6。
- 步骤 6
通过配置 ADRFS 位选择 A/D 转换器输出数据格式。
- 步骤 7
设置 SAVRS 位段选择 A/D 转换器参考电压源。
若 PGA 输出电压 V_R 被选作 A/D 转换器参考电压，需进一步选择 PGA 输入信号和所需的 PGA 增益。
- 步骤 8
如果要使用 A/D 转换中断，则中断控制寄存器需要正确地设置，以确保 A/D 中断功能是有用的。总中断控制位 EMI 以及 A/D 转换器中断位 ADE 需要预先置位为“1”。
- 步骤 9
现在可以通过设定 START 位从低到高再到低，开始 A/D 转换的过程。
- 步骤 10
若 A/D 转换正在进行，ADBZ 标志将置高。A/D 转换结束后，ADBZ 标志置低，并可从 SADOH 和 SADOL 寄存器读取输出数据。

注：若使用轮询 SADC0 寄存器中的 ADBZ 位的方法以检测模数转换过程是否完成，上述中断使能步骤可以忽略。

编程注意事项

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

A/D 转换功能

该系列单片机含有一个 12-bit A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于实际 A/D 转换器参考电压 V_{REF} ，因此每一位可表示 $V_{REF}/4096$ 的模拟输入值。

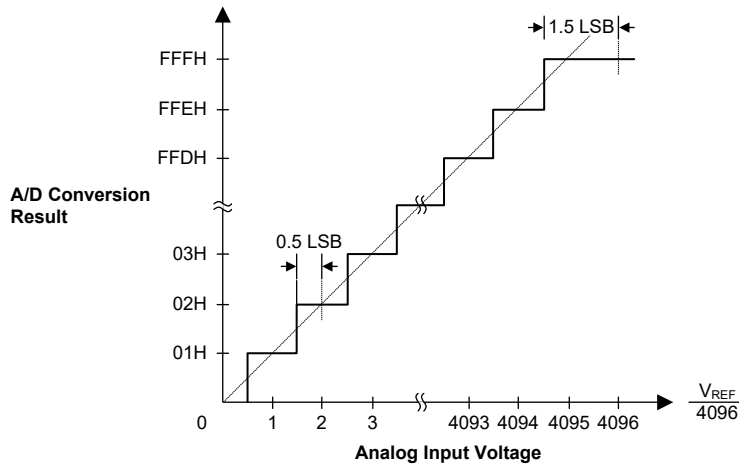
$$1 \text{ LSB} = V_{REF}/4096$$

通过下面的等式可估算 A/D 转换器输入电压值：

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

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

应注意的是此处的 V_{REF} 电压指的是 A/D 转换器参考电压源，由 SAVRS 位段确定。



理想 A/D 转换功能

A/D 转换程序范例

下面两个范例程序用来说明怎样设置和实现 A/D 转换。第一个范例是轮询 SADC0 寄存器中的 ADBZ 位来判断 A/D 转换何时完成，而第二个范例则使用中断的方式判断。

范例 1：使用轮询 ADBZ 轮询的方式来检测转换结束

```
clr ADE                ; disable ADC interrupt
mov a,03H              ; select fsys/8 as A/D clock and A/D input
mov SADC1,a            ; signal comes from external channel
mov a,00H              ; select AVDD as A/D reference voltage source
mov SADC2,a
set ADCEN
mov a,03H              ; setup PCS0 to configure pin AN0
mov PCS0, a
mov a,00H              ; select AN0 as the A/D external channel input
mov SADC0,a
:
start_conversion:
clr START              ; high pulse on start bit to initiate conversion
set START              ; reset A/D
clr START              ; start A/D
:
polling_EOC:
sz ADBZ                ; poll the SADC0 register ADBZ bit to detect end
                       ; of A/D conversion
jmp polling_EOC       ; continue polling
:
mov a,SADOL            ; read low byte conversion result value
mov SADOL_buffer,a    ; save result to user defined register
mov a,SADOH            ; read high byte conversion result value
mov SADOH_buffer,a    ; save result to user defined register
:
jmp start_conversion  ; start next A/D conversion
```

范例 2：使用中断的方式来检测转换结束

```

clr ADE                ; disable ADC interrupt
mov a,03H              ; select fsys/8 as A/D clock and A/D input
mov SADC1,a            ; signal comes from external channel
mov a,00H              ; select AVDD as the A/D reference voltage source
mov SADC2,a
set ADCEN
mov a,03H              ; setup PCS0 to configure pin AN0
mov PCS0, a
mov a,00H              ; select AN0 as the A/D external channel input
mov SADC0,a
:
Start_conversion:
clr START              ; high pulse on START bit to initiate conversion
set START              ; reset A/D
clr START              ; start A/D
clr ADF                ; clear ADC interrupt request flag
set ADE                ; enable ADC interrupt
set EMI                ; enable global interrupt
:
:
ADC_ISR:               ; ADC interrupt service routine
mov acc_stack,a        ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a    ; save STATUS to user defined memory
:
mov a,SADOL             ; read low byte conversion result value
mov SADOL_buffer,a    ; save result to user defined register
mov a,SADOH            ; read high byte conversion result value
mov SADOH_buffer,a    ; save result to user defined register
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a          ; restore STATUS from user defined memory
mov a,acc_stack       ; restore ACC from user defined memory
reti
    
```

串行接口模块 – SIM

该系列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。由于 SIM 接口引脚与其它 I/O 引脚共用，因此需通过设置 SIM 使能 / 除能位将引脚预先设置为 SIM 接口功能。因为这两种接口共用引脚和寄存器，所以要通过一个 SIMC0 寄存器中的 SIM2~SIM0 位来选择哪一种通信接口。若 SIM 功能使能，可通过上拉电阻控制寄存器选择与输入 / 输出共用的 SIM 脚的上拉电阻。

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同步串行数据接口，这个协议可以简化与外部硬件的编程要求。

SPI 通信模式为全双工模式，且能以主 / 从模式的工作方式进行通信，单片机既可以作为主机，也可以作为从机。虽然 SPI 接口理论上允许一个主机控制多个从机，但此处的 SPI 中只有一个片选信号引脚 \overline{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。

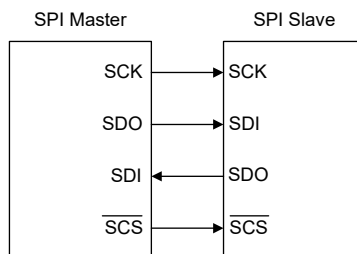
SPI 接口操作

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。SPI 可以通过 SIMC0 寄存器中的 SIMEN 位来除能或使能。连接到 SPI 接口的单片机以从主 / 从模式进行通信，且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} 引脚将处于浮空状态。

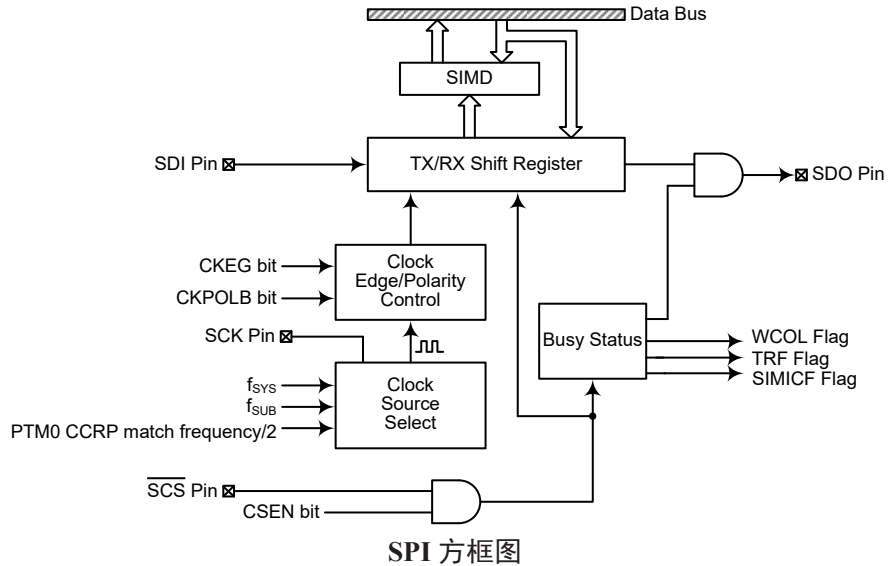
该系列单片机的 SPI 功能具有以下特点：

- 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和一些控制位如 CSEN，SIMEN 位的状态。



SPI 主机 / 从机连接



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	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
SIMD	D7	D6	D5	D4	D3	D2	D1	D0

SPI 寄存器列表

• SIMD 寄存器

寄存器 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寄存器 SIMD 实现。

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

“x”：未知

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。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	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	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 PTM0 CCRP 匹配频率 /2
- 101: SPI 从机模式
- 110: I²C 从机模式
- 111: 非 SIM 功能

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也可以选择来自 PTM0。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 未使用，读为“0”

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C 去抖时间选择位

SIMDEB1~SIMDEB0 位段仅用于 I²C 模式，细节描述请参见 I²C 接口章节。

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 脚处于浮空状态，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 HTX 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I²C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0 **SIMICF**: SIM SPI 从机模式未完成传输标志位

- 0: 未发生 SIM SPI 从机模式未完成传输情况
- 1: 发生 SIM SPI 从机模式未完成传输情况

此位仅在 SIM 设置为 SPI 从机模式时可用。当 SPI 工作在从机模式，SIMEN 和 CSEN 位同时设为“1”，但 SCS 引脚在 SPI 数据传输完成之前由外部主机设备拉高，则 SIMICF 位将由硬件与 TRF 位一同设为“1”。此时，若中断使能，则产生一个中断。如果 SIMICF 位由软件应用程序置为“1”，则 TRF 位不会置为“1”。

• SIMC2 寄存器

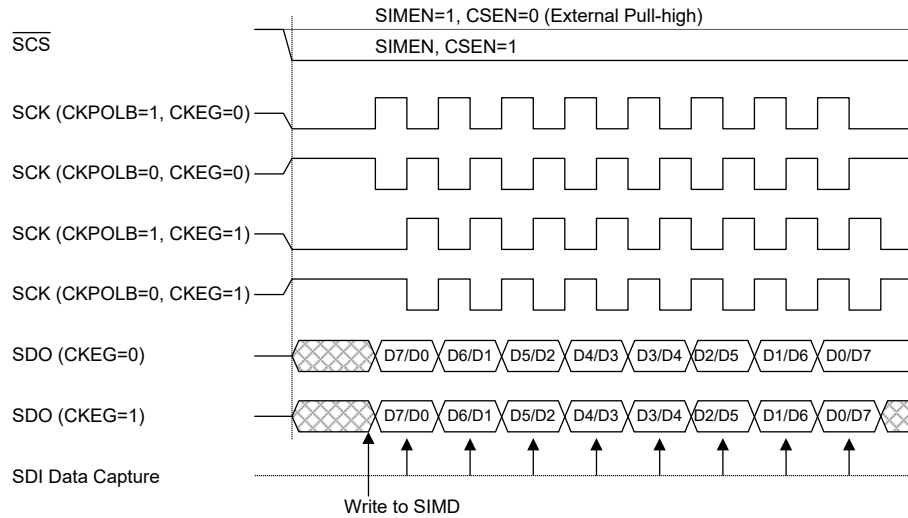
Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7~6 未定义位
用户可通过软件程序对这两位进行读写。
- Bit 5 **CKPOLB**: SPI 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 上升沿捕捉数据
1: SCK 为高电平且在 SCK 下降沿捕捉数据
CKPOLB=1
0: SCK 为低电平且在 SCK 下降沿捕捉数据
1: SCK 为低电平且在 SCK 上升沿捕捉数据
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 位决定有效时钟边沿类型, 取决于 CKPOLB 的状态。
- Bit 3 **MLS**: SPI 数据移位顺序
0: LSB 优先
1: MSB 优先
此位为数据移位选择位, 用于选择数据传输时高位优先传输还是低位优先传输。此位设置为高时高位优先传输, 为低时低位优先传输。
- Bit 2 **CSEN**: SPI \overline{SCS} 引脚控制位
0: 除能
1: 使能
CSEN 位用于 \overline{SCS} 引脚的使能 / 除能控制。此位为低时, \overline{SCS} 除能并处于浮空状态。此位为高时, \overline{SCS} 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, 数据在传输时被写入 SIMD 寄存器。若数据正在被传输时, 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。

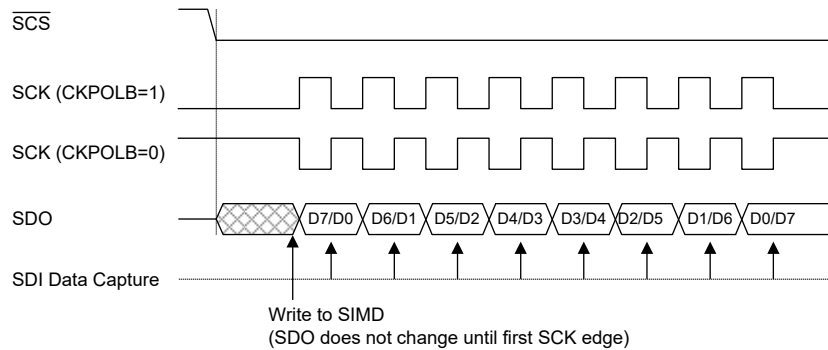
SPI 通信

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 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 SCS 信号的关系。

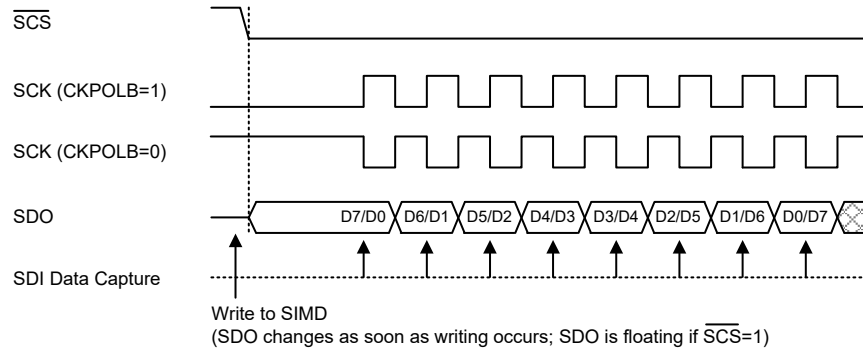
即使在单片机处于空闲模式，SPI 功能仍将继续执行。



SPI 主机模式时序图

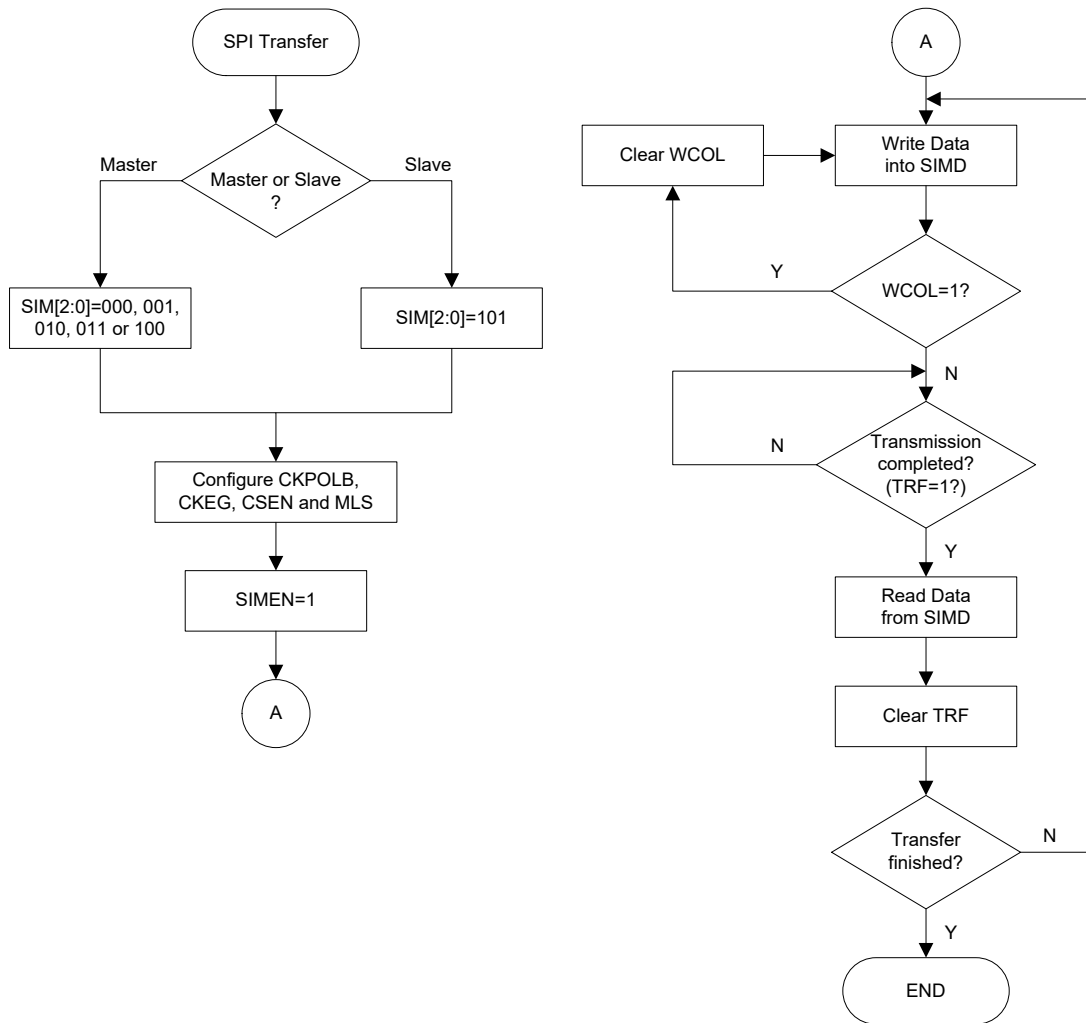


SPI 从机模式时序 - CKEG=0



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

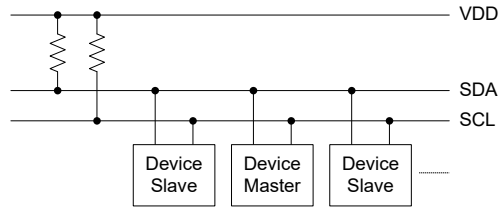
SPI 从机模式时序 - CKEG=1



SPI 传输控制流程图

I²C 接口

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

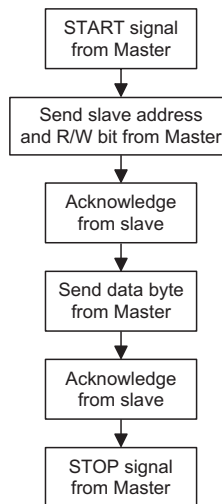


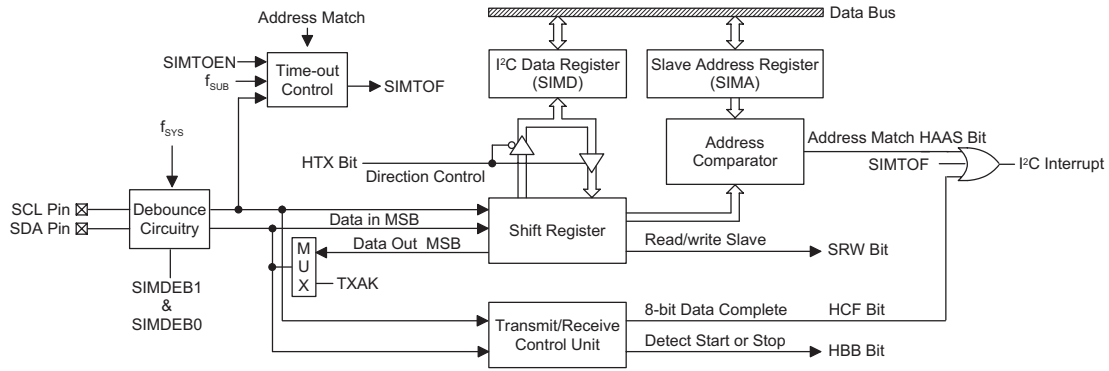
I²C 主 / 从总线连接图

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 方框图

SIMDEB1 和 SIMDEB0 位决定 I²C 接口的去抖时间。这个功能可以使用内部时钟在外部时钟上增加一个去抖间隔，会减小时钟线上毛刺发生的可能性，以避免单片机发生错误动作。如果选择了这个功能，去抖时间可以选择 2 个或 4 个系统时钟。为了达到需要的 I²C 数据传输速度，系统时钟 f_{SYS} 和 I²C 去抖时间之间存在一定的关系。I²C 标准模式或者快速模式下，用户需注意所选的系统时钟频率与标准匹配去抖时间的设置，其具体关系如下表所示。

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

I²C 最小 f_{SYS} 频率

I²C 寄存器

I²C 总线有三个控制寄存器 SIMC0、SIMC1 和 SIMTOC，一个从机地址寄存器 SIMA 及一个数据寄存器 SIMD。SIMD 寄存器用于存储正在传输和接收的数据。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。

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 SIM2~SIM0 位。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C 寄存器列表

● **SIMD 寄存器**

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。

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

“x”：未知

● **SIMA 寄存器**

SIMA 寄存器也在 SPI 接口功能中使用，但其名称改为 SIMC2。SIMA 寄存器用于存放 7 位从机地址，寄存器 SIMA 中的 bit 7 ~ bit 1 是单片机的从机地址，bit 0 未定义。

如果接至 I²C 的主机发送处的地址和寄存器 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 是同一个地址上的寄存器。

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

“x”：未知

Bit 7~1 **IICA6-IICA0**: I²C 从机地址位
 A6~A0 是从机地址 bit 6 ~ bit 0

Bit 0 未定义，读为“0”

单片机中也有两个控制 I²C 接口功能的寄存器，SIMC0 和 SIMC1。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。寄存器 SIMC1 包括多个用于表明 I²C 通信状态的相关标志位。

● **SIMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	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 PTM0 CCRP 匹配频率 / 2

101: SPI 从机模式

110: I²C 从机模式

111: 非 SIM 功能

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也可以选择来自 f_{SUB} 或 PTM0。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

- Bit 4 未使用，读为“0”
- Bit 3~2 **SIMDEB1~SIMDEB0**: I²C 去抖时间选择位
00: 无去抖时间
01: 2 个系统时钟去抖时间
1x: 4 个系统时钟去抖时间
当 SIM 通过设置 SIM2~SIM0 位段配置为“110”以选中 I²C 接口功能时，此位段用于选择 I²C 去抖动时间。
- 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 脚处于浮空状态，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 HTX 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I²C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。
- Bit 0 **SIMICF**: SIM SPI 未完成标志位
SIMICF 位仅在 SPI 模式中使用，详细定义请参见 SPI 接口章节。

● SIMC1 寄存器

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

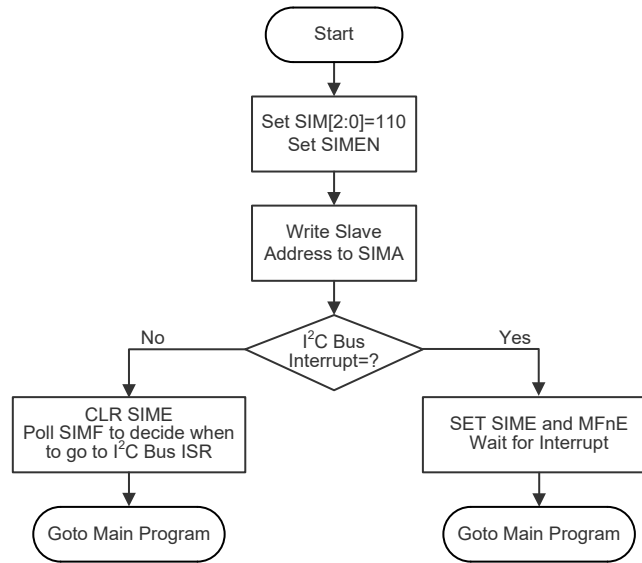
- Bit 7 **HCF**: I²C 总线数据传输结束标志位
0: 数据正在被传输
1: 8 位数据传输完成
数据正在传输时该位为低。当 8 位数据传输完成时，此位为高并产生一个中断。
- Bit 6 **HAAS**: I²C 总线地址匹配标志位
0: 地址不匹配
1: 地址匹配
此标志位用于决定从机地址是否与主机发送地址相同。若地址匹配此位为高，否则此位为低。
- Bit 5 **HBB**: I²C 总线忙标志位
0: I²C 总线闲
1: I²C 总线忙
当检测到 START 信号时 I²C 忙，此位变为高电平。当检测到 STOP 信号时 I²C 总线停止，该位变为低电平。
- Bit 4 **HTX**: I²C 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 位为高时, 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 **IAMWU**: I²C 地址匹配唤醒控制位
 0: 除能
 1: 使能 – 唤醒后或必须由应用程序清零
此位应设置为“1”使能 I²C 地址匹配以使系统从休眠或空闲模式中唤醒。若进入休眠或空闲模式前 IAMWU 已经设置以使能 I²C 地址匹配唤醒功能, 在系统唤醒后须软件清除此位以确保单片机正确地运行。
- Bit 0 **RXAK**: I²C 总线接收确认标志位
 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收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。

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 SIMTOF 位, 以判断 I²C 总线中断是来自从机地址匹配, 还是来自 8 位数据传递完毕或是来自 I²C 超时。在数据传递中, 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”和 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 SIME 位和 SIM 多功能终端使能位, 以使能 SIM 中断和多功能中断。



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 MCU 产生。总线上的所有从机都可以侦测到起始信号。如果有从机侦测到起始信号，则表明 I²C 总线处于忙碌状态，并会置位 HBB。起始信号是指在 SCL 为高电平时，SDA 线上发生从高到低的电平变化。

I²C 从机地址

总线上的所有从机都会侦测由主机发出的起始信号。发送起始信号后，紧接着主机会发送从机地址以选择要进行数据传输的从机。所有在 I²C 总线上的从机接收到 7 位地址数据后，都会将其与各自内部的地址进行比较。如果从机从主机上接收到的地址与自身内部的地址相匹配，则会产生一个 I²C 总线中断信号。地址位接下来的一位为读 / 写状态位（即第 8 位），将被保存到 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 SIMTOF 位，以判断 I²C 总线中断是来自从机地址匹配，还是来自 8 位数据传递完毕，或是来自 I²C 超时。当是从机地址匹配发生中断时，则从机或是用于发送模式并将数据写进 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。

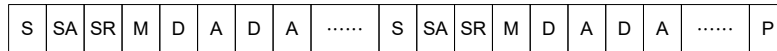
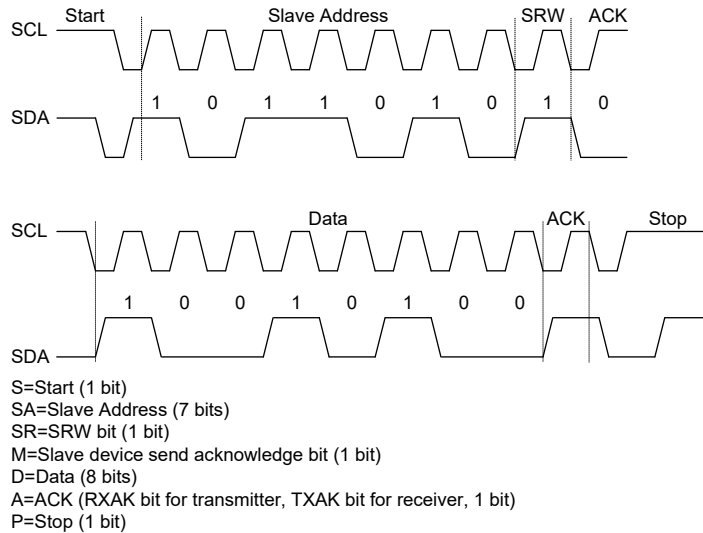
I²C 总线从机地址确认信号

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

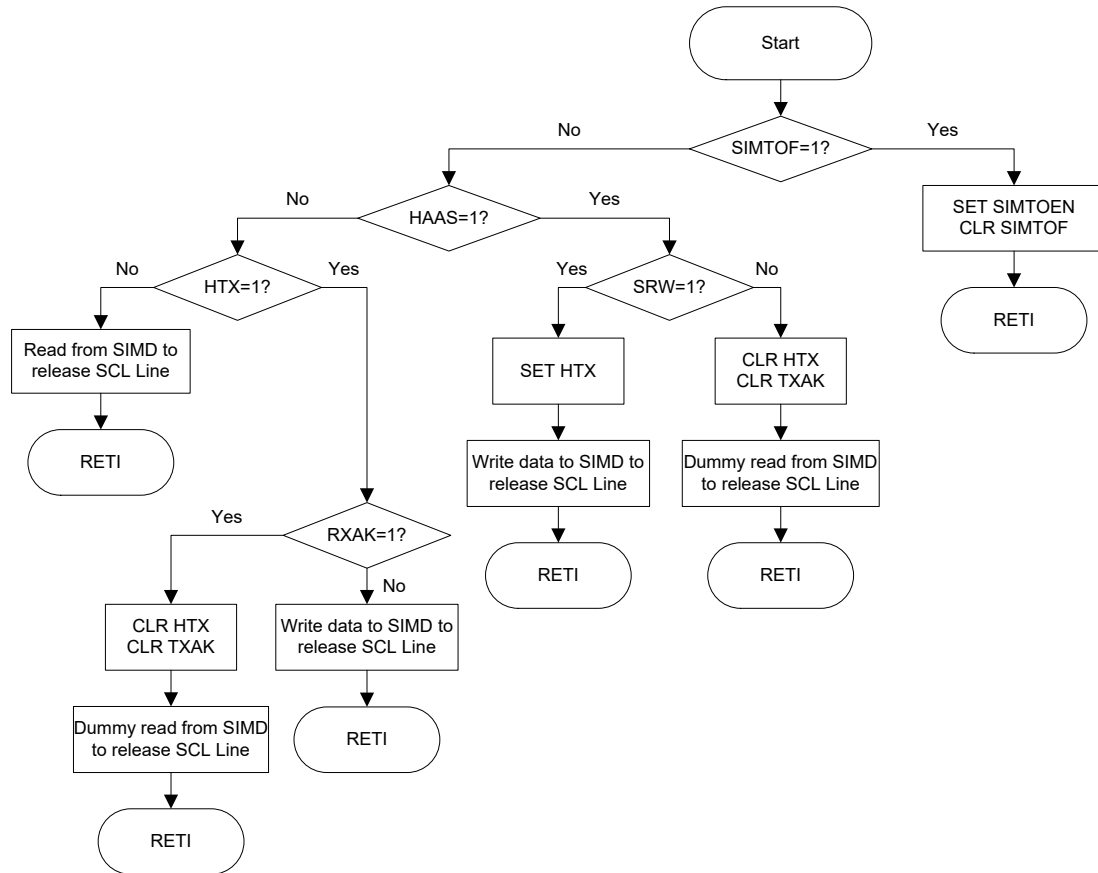
在从机确认接收到从地址后，会进行 8 位宽度的数据传输。这个数据传输顺序是的高位在前，低位在后。接收方在接收到 8 位数据后必须发出一个应答信号 (“0”) 以继续接收下一个数据。如果发送方没接收到应答信号，发送方将释放 SDA 线，同时，主机将发出 STOP 信号以释放 I²C 总线。所传送的数据存储在 SIMD 寄存器中。如果设置成发送方，从机必须先将欲传输的数据写到 SIMD 寄存器中；如果设置成接收方，从机必须从 SIMD 寄存器读取数据。

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。

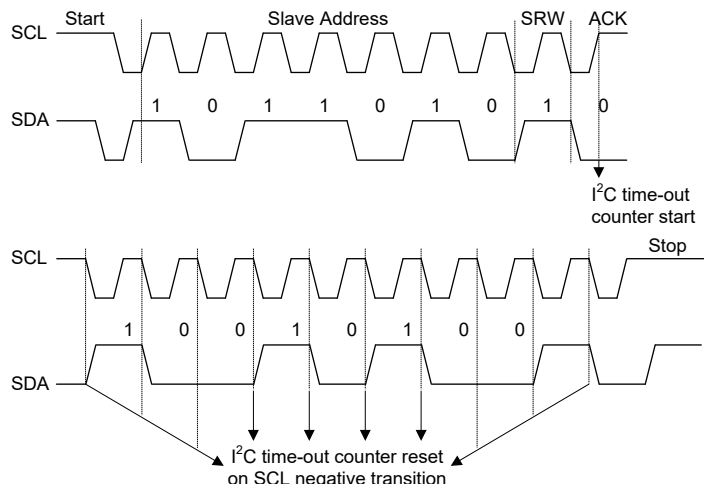
I²C 通信时序图



I²C 总线 ISR 流程图

I²C 超时功能

为了减少由于接收错误时钟源而产生的 I²C 锁定问题，系统提供了超时功能。在固定时间内如果 I²C 总线未接收到时钟源，则 I²C 电路和寄存器将会复位。超时计数器在 I²C 总线接收到“START”信号和“地址匹配”条件时，超时计数器开始计数，并在 SCL 下降沿处清零。在下一个 SCL 下降沿来临之前，如果等待时间大于 SIMTOC 寄存器设定的超时时间，则会发生超时现象。当 I²C 接收到“STOP”信号，超时计数器将停止计数。



I²C 超时

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

寄存器	I ² C 超时发生后
SIMD, SIMA, SIMC0	保持不变
SIMC1	复位至 POR

超时后的 I²C 寄存器状态

SIMTOF 标志位由应用程序清零。共有 64 个溢出周期，可通过 SIMTOC 寄存器的相关位进行选择。溢出周期可通过公式计算： $((1\sim 64)\times(32/f_{SUB}))$ 。由此可得溢出周期范围为 1ms~64ms。

● SIMTOC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0
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 **SIMTOEN**: SIM I²C 超时控制位

- 0: 除能
- 1: 使能

Bit 6 **SIMTOF**: SIM I²C 超时标志位

- 0: 未发生
- 1: 发生

该位在超时发生时置位，可由软件清零。

Bit 5~0 **SIMTOS5~SIMTOS0**: I²C 超时时间选择位

I²C 超时时钟源是 $f_{SUB}/32$

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

SPIA 串行接口模块 – SPIA

该单片机内含一个独立的 SPI 功能。重要的是，不要将 SPI 功能与 SIM 模块中的 SPI 功能混淆，其具体描述详见规格书的另一章节。将这个独立的 SPI 命名为 SPIA 以区别于 SIM 中的 SPI。

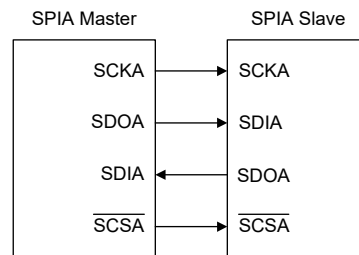
SPIA 接口常用于与外部设备如传感器、闪存或 EEPROM 内存等通信。四线 SPIA 接口最初是由摩托罗拉公司研制，是一个有相当简单的通信协议的串行数据接口，这个协议可以简化与外部硬件的编程要求。

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

SPIA 接口操作

SPIA 接口是一个全双工串行数据传输器。SPIA 接口的四线为：SDIA、SDOA、SCKA 和 \overline{SCSA} 。SDIA 和 SDOA 是数据的输入和输出线。SCKA 是串行时钟线， \overline{SCSA} 是从机的选择线。SPIA 的接口引脚与普通 I/O 口共用。通过设定 SPIAC0/SPIAC1 寄存器的对应位，来使能 SPIA 接口。SPIA 可以通过 SPIAC0 寄存器中的 SPIAEN 位来除能或使能。连接到 SPIA 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{SCSA} 引脚，所以只能拥有一个从机设备。

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

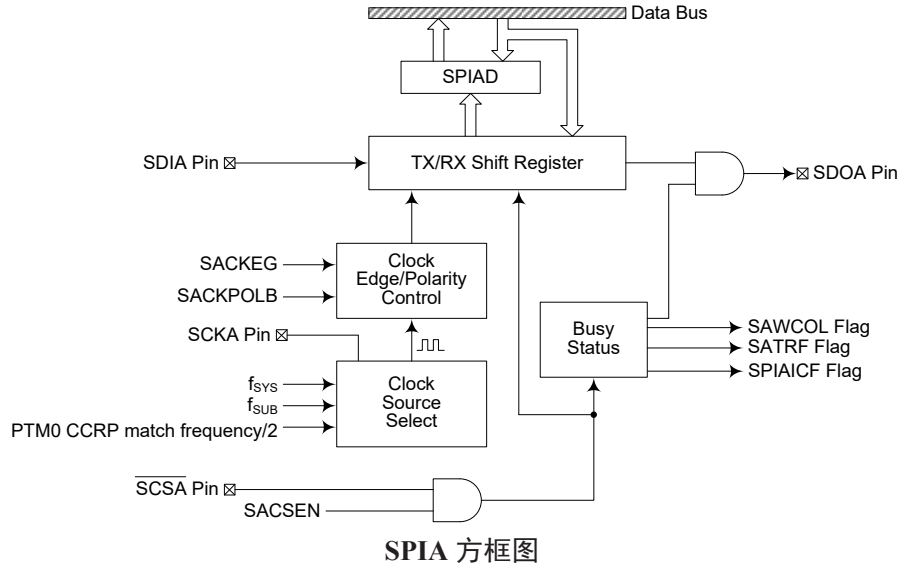


SPIA 主 / 从机连接方式

该单片机的 SPIA 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

SPIA 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和 SACSSEN、SPIAEN 位的状态。



SPIA 寄存器

有三个寄存器用于控制 SPIA 接口的所有操作，其中有一个数据寄存器 SPIAD、两个控制寄存器 SPIAC0 和 SPIAC1。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SPIAC0	SASPI2	SASPI1	SASPI0	—	—	—	SPIAEN	SPIAICF
SPIAC1	—	—	SACKPOLB	SACKEG	SAMLS	SACSEN	SAWCOL	SATRF
SPIAD	D7	D6	D5	D4	D3	D2	D1	D0

SPIA 寄存器列表

SPIA 数据寄存器

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

• SPIAD 寄存器

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

“x”：未知

SPIA 控制寄存器

单片机中也有两个控制 SPIA 接口功能的寄存器，SPIAC0 和 SPIAC1。寄存器 SPIAC0 用于控制使能 / 除能功能和设置数据传输的时钟频率。寄存器 SPIAC1 用于其它的控制功能如 LSB/MSB 选择，写冲突标志位等。

• SPIAC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SASPI2	SASPI1	SASPI0	—	—	—	SPIAEN	SPIAICF
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	1	1	1	—	—	—	0	0

Bit 7~5 **SASPI2~SASPI0**: SPIA 工作模式控制位

- 000: SPIA 主机模式; SPIA 时钟为 $f_{SYS}/4$
- 001: SPIA 主机模式; SPIA 时钟为 $f_{SYS}/16$
- 010: SPIA 主机模式; SPIA 时钟为 $f_{SYS}/64$
- 011: SPIA 主机模式; SPIA 时钟为 f_{SUB}
- 100: SPIA 主机模式; SPIA 时钟为 PTM0 CCRP 匹配频率 / 2
- 101: SPIA 从机模式
- 11x: SPIA 除能

Bit 4~2 未定义, 读为“0”

Bit 1 **SPIAEN**: SPIA 控制位

- 0: 除能
- 1: 使能

此位为 SPIA 接口的开 / 关控制位。此位为“0”时, SPIA 接口除能, SDIA、SDOA、SCKA 和 SCSA 脚作为 I/O 口或其它共用引脚, SPIA 工作电流减小到最小值。此位为“1”时, SPIA 接口使能。

Bit 0 **SPIAICF**: SPIA 未完成传输标志位

- 0: SPIA 传输完成
- 1: SPIA 未完成传输

此位仅当 SPIA 配置在 SPIA 从机模式时有效。如果 SPIA 工作在从机模式且 SPIAEN 和 SACSSEN 位都为“1”, 但在 SPIA 数据传输完全结束前 SCSA 线被外部主机拉高, SPIAICF 和 SATRF 位都会被置高。在这种情况下, 如果相应的中断功能使能将产生一个中断。然而, 如果 SPIAICF 位是由软件应用程序设为 1, 那么 SATRF 位将不会置高。

• SPIAC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	SACKPOLB	SACKEG	SAMLS	SACSSEN	SAWCOL	SATRF
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 **SACKPOLB**: 时钟线的基础状态位

- 0: 当时钟无效时, SCKA 引脚为高电平
- 1: 当时钟无效时, SCKA 引脚为低电平

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

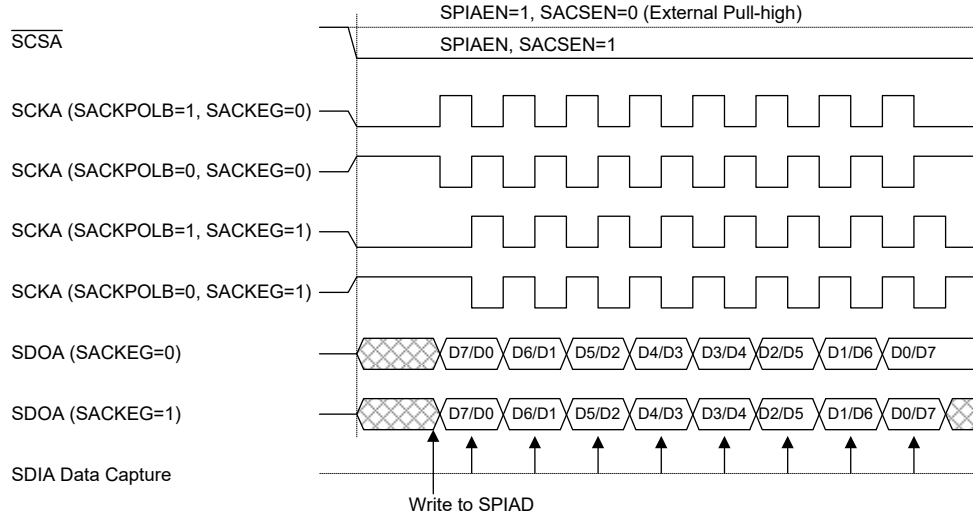
- Bit 4 **SACKEG**: SPIA 的 SCKA 有效时钟边沿类型位
SACKPOLB=0
0: SCKA 为高电平且在 SCKA 上升沿抓取数据
1: SCKA 为高电平且在 SCKA 下降沿抓取数据
SACKPOLB=1
0: SCKA 为低电平且在 SCKA 下降沿抓取数据
1: SCKA 为低电平且在 SCKA 上升沿抓取数据
SACKEG 和 SACKPOLB 位用于设置 SPIA 总线上时钟信号输入和输出方式。在执行数据传输前, 这两位必须被设置, 否则将产生错误的时钟边沿信号。SACKPOLB 位决定时钟线的基本状态, 若时钟无效且此位为高, 则 SCKA 为低电平, 若时钟无效且此位为低, 则 SCKA 为高电平。SACKEG 位决定有效时钟边沿类型, 取决于 SACKPOLB 的状态。
- Bit 3 **SAMLS**: SPIA 数据移位顺序控制位
0: LSB 优先
1: MSB 优先
数据移位选择位, 用于选择数据传输时高位优先传输还是低位优先传输。此位设置为高时高位优先传输, 为低时低位优先传输。
- Bit 2 **SACSEN**: SPIA \overline{SCSA} 引脚控制位
0: 除能
1: 使能
SACSEN 位用于 \overline{SCSA} 引脚的使能 / 除能控制。此位为低时, \overline{SCSA} 引脚功能除能并处于浮空状态。此位为高时, \overline{SCSA} 引脚功能使能并作为选择引脚。
- Bit 1 **SAWCOL**: SPIA 写冲突标志位
0: 无冲突
1: 冲突
SAWCOL 标志位用于监测数据冲突的发生。此位为高时, 数据在传输时被写入 SPIAD 寄存器。若数据正在被传输时, 此操作无效。此位可被应用程序清零。
- Bit 0 **SATRF**: SPIA 发送 / 接收结束标志位
0: 数据正在发送
1: 数据发送结束
SATRF 位为发送 / 接收结束标志位, 当 SPIA 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。

SPIA 通信

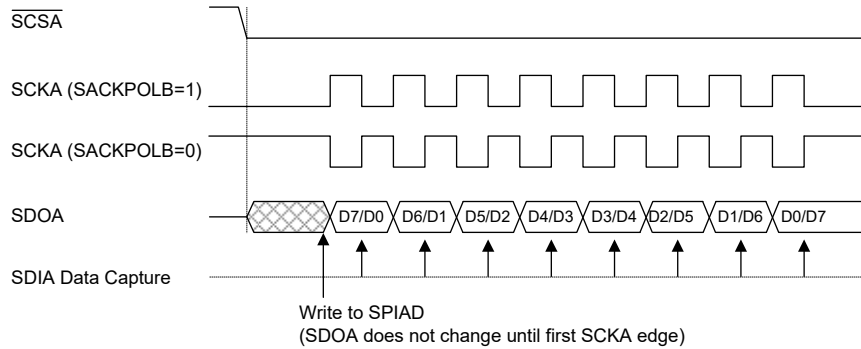
将 SPIAEN 设置为高, 使能 SPIA 功能之后, 单片机处于主机模式, 当数据写入到寄存器 SPIAD 的同时传输 / 接收开始进行。数据传输完成时, SATRF 位将自动被置位但清除只能通过应用程序完成。单片机处于从机模式时, 收到主机发来的信号之后, 会传输 SPIAD 中的数据, 而且在 SDIA 引脚上的数据也会被移位到 SPIAD 寄存器中。

主机应在输出时钟信号之前先输出一个 \overline{SCSA} 信号以使能从机, 从机的数据传输功能也应在与 \overline{SCSA} 信号相关的适当时候准备就绪, 这由 SACKPOLB 和 SACKEG 位决定。所附时序图表明了 SACKPOLB 和 SACKEG 位各种设置情况下从机数据与 \overline{SCSA} 信号的关系。

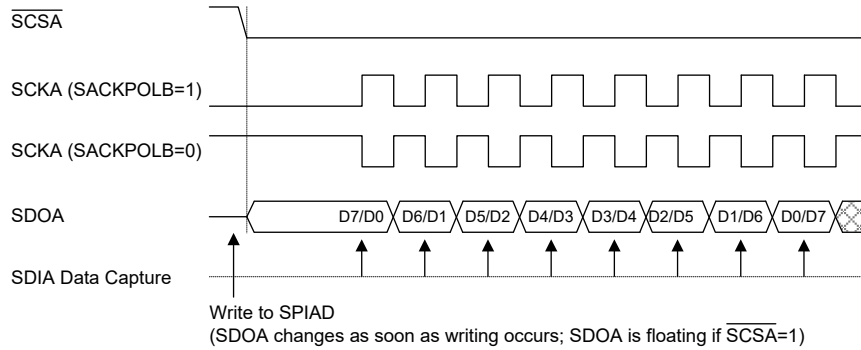
空闲模式下, SPIA 功能仍将继续执行。



SPIA 主机模式时序图

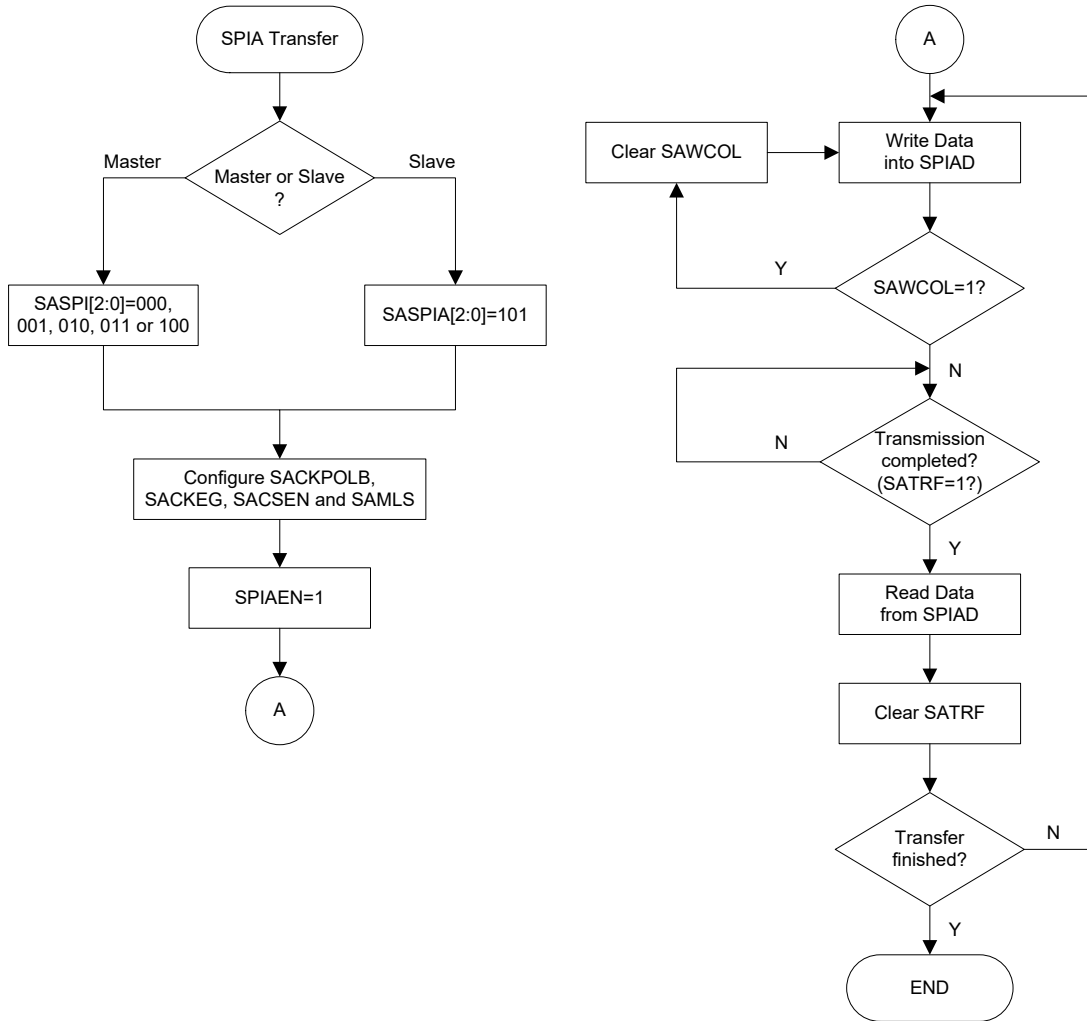


SPIA 从机模式时序图 – SACKEG=0



Note: For SPIA slave mode, if SPIAEN=1 and SACSSEN=0, SPIA is always enabled and ignores the SCSA level.

SPIA 从机模式时序图 – SACKEG=1



SPIA 传输控制流程图

SPIA 总线使能 / 除能

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

设置 SCKA、SDIA、SDOA、 \overline{SCSA} 为 I/O 口或其它共用引脚可除能 SPIA。

SPIA 操作

四线制 SPIA 接口可完成所有主 / 从模式通信工作。

在 SPIAC1 寄存器中，SACSEN 位控制 SPIA 接口的所有功能。设置此位为高， \overline{SCSA} 信号线有效将使能 SPIA 接口。设置此位为低， \overline{SCSA} 信号线作为 I/O 口或其它共用引脚将除能 SPIA 接口。SACSEN 位和 SPIAC0 寄存器中的 SPIAEN 位设置为高，使得 SDIA 信号线浮空且 SDOA 信号线为高电平。主机模式中，如果 SCKA 信号线为高还是低取决于 SPIAC1 寄存器中的时钟极性选择位 SACKPOLB。从机模式中，SCKA 信号线浮空。如果 SPIAEN 位设置为低，SPIA 接口被除能，SCSA、SDIA、SDOA 和 SCKA 可通过引脚共用功能控制位选择作为 I/O 口或其它共用引脚功能。主机模式中，主机一直用于产生时钟信号，当数据写入到 SPIAD 寄存器后，主机完成所有的时钟和数据传输初始化。从机模式中，由外部主机发出数据传送 / 接收时钟信号。下面介绍主从模式中数据传输步骤。

主机模式：

- 步骤 1
设置 SPIAC0 控制寄存器中的 SASPI2~SASPI0 位，选择 SPIA 主机模式和时钟源。
- 步骤 2
设置 SACSEN 和 SAML5 位，选择高位或低位数据优先传送，这必须与从机设备一致。
- 步骤 3
设置 SPIAC0 控制寄存器中的 SPIAEN 位，使能 SPIA 接口功能。
- 步骤 4
对于写操作：写数据到 SPIAD 寄存器，实际上，数据被存储在 TXRX 缓存器中。再使用 SCKA 和 SCSA 信号线将数据输出。之后跳至步骤 5。
对于读操作：使用 SDIA 信号线将存储在 TXRX 缓存器中的数据移出，并全部锁存至 SPIAD 寄存器。
- 步骤 5
检测 SAWCOL 位，若此位为高，则发生数据冲突并跳回至步骤 4；若为低，则继续执行下面的步骤。
- 步骤 6
检测 SATRF 位或等待 SPIA 串行总线中断发生。
- 步骤 7
从 SPIAD 寄存器中读数据。
- 步骤 8
清除 SATRF。
- 步骤 9
跳回至步骤 4。

从机模式：

- 步骤 1
设置 SPIAC0 控制寄存器中的 SASPI2~SASPI0 位，选择 SPIA 从机模式。
- 步骤 2
设置 SACSEN 和 SAML5 位，选择高位或低位数据优先传送，这必须与主机设备一致。
- 步骤 3
设置 SPIAC0 控制寄存器中的 SPIAEN 位，使能 SPIA 接口功能。
- 步骤 4
对于写操作：写数据到 SPIAD 寄存器，实际上，数据被存储在 TXRX 缓存器中。等待主机时钟 SCKA 信号和 \overline{SCSA} 信号。跳至步骤 5。
对于读操作：使用 SDIA 信号线将 TXRX 缓存器中的数据移出，并全部锁存至 SPIAD 寄存器。
- 步骤 5
检测 SAWCOL 位，若此位为高，则发生数据冲突并跳回至步骤 4；若为低，则继续执行下面的步骤。
- 步骤 6
检测 SATRF 位或等待 SPIA 串行总线中断发生。
- 步骤 7
从 SPIAD 寄存器中读数据。
- 步骤 8
清除 SATRF。
- 步骤 9
跳回至步骤 4。

错误侦测

SPIAC1 寄存器中的 SAWCOL 位用于数据传输期间监测数据冲突的发生。此位由 SPIA 串行接口设置为高，但必须由应用程序来清零。此位置高，表示有数据冲突发生。即如果在数据发送期间，写数据到 SPIAD 寄存器，数据的继续写入会被阻止，此时数据冲突发生。

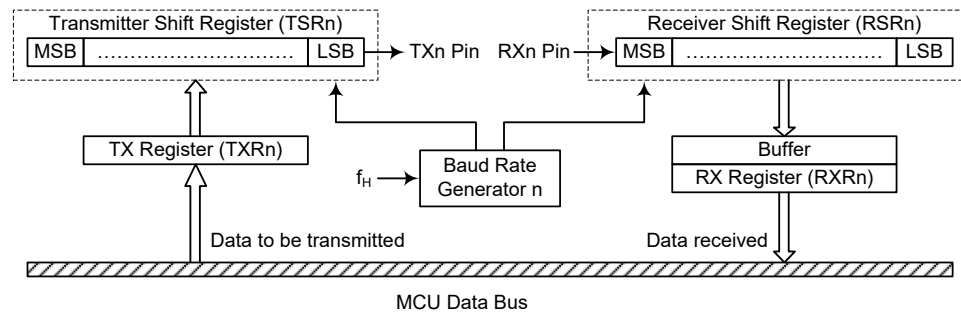
UART 接口

该系列单片机具有多达 3 个全双工的异步串行通信接口 – UART，可以很方便的与其它具有串行口的芯片通信。UART 具有许多功能特性，发送或接收串行数据时，将数据组成一个 8 位或 9 位的数据块传输，同时具有检测数据覆盖或帧错误的功能。UART 功能占用一个内部中断向量，当接收到数据或数据发送结束，触发 UART 中断。

单片机型号	UART 接口
HT66F2350/HT66F2360	2
HT66F2370/HT66F2390	3

内建的 UART 模块包含下列特性：

- 全双工通用异步接收器 / 发送器
- 8 位或 9 位传输格式
- 奇校验、偶校验或无校验
- 1 位或 2 位停止位
- 8 位预分频的波特率发生器
- 奇偶、帧、噪声和溢出检测
- 支持地址检测中断 (最后一位 =1)
- 独立的发送和接收使能
- 2-byte FIFO 接收缓冲器
- RX 引脚唤醒功能
- 发送和接收中断源：
 - ◆ 发送器为空
 - ◆ 发送器空闲
 - ◆ 接收完成
 - ◆ 接收器溢出
 - ◆ 地址模式检测



UART 数据传输方框图

UART 外部引脚

内部 UARTn 有两个外部引脚 TXn 和 RXn，可与外部串行接口进行通信。TXn 和 RXn 分别为 UARTn 发送脚和接收脚，与 I/O 口或其它功能共用引脚。在使用 UARTn 功能前，应先通过相应的引脚共用功能选择寄存器，选择 TXn 和 RXn 引脚功能。当 UARTEFn 和 TXENn/RXENn 位置高时，将自动设置这些 I/O 脚或其它共用功能脚作为 TXn 输出和 RXn 输入，并且除能 TXn 和 RXn 引脚上的上拉电阻功能。当 UARTEFn、TXENn 或 RXENn 位清零除能 TXn 或 RXn 引脚功能后，TXn 或 RXn 引脚将处于浮空状态。这时 TXn 或 RXn 引脚是否连接内部上拉电阻是由相应的 I/O 上拉电阻控制位决定的。

UART 数据传输方案

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

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

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

UART 状态和控制寄存器

与 UARTn 功能相关的有五个寄存器——控制 UARTn 模块整体功能的 UnSR、UnCR1 和 UnCR2 寄存器，控制波特率的 BRGn 寄存器，管理发送和接收数据的数据寄存器 TXR_RXRn。

- **TXR_RXRn 寄存器**

TXR_RXRn 是一个数据寄存器，用来存储 TXn 引脚将要发送或 RXn 引脚正在接收的数据。

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

“x”：未知

Bit 7~0 **D7~D0**: UARTn 发送 / 接收数据位

• UnSR 寄存器

寄存器 UnSR 是 UARTn 的状态寄存器，可以通过程序读取。所有 UnSR 位是只读的。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	PERRn	NFn	FERRn	OERRn	RIDLEn	RXIFn	TIDLEn	TXIFn
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

Bit 7 **PERRn**: 奇偶校验出错标志位

- 0: 奇偶校验正确
- 1: 奇偶校验出错

PERRn 是奇偶校验出错标志位。若 PERRn=0，奇偶校验正确；若 PERRn=1，接收到的数据奇偶校验出错。只有使能了奇偶校验此位才有效。可使用软件清除该标志位，即先读取 UnSR 寄存器再读 TXR_RXRn 寄存器来清除此位。

Bit 6 **NFn**: 噪声干扰标志位

- 0: 没有受到噪声干扰
- 1: 受到噪声干扰

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

Bit 5 **FERRn**: 帧错误标志位

- 0: 无帧错误发生
- 1: 有帧错误发生

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

Bit 4 **OERRn**: 溢出错误标志位

- 0: 无溢出错误发生
- 1: 有溢出错误发生

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

Bit 3 **RIDLEn**: 接收状态标志位

- 0: 正在接收数据
- 1: 接收器空闲

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

Bit 2 **RXIFn**: 接收寄存器状态标志位

- 0: TXR_RXRn 寄存器为空
- 1: TXR_RXRn 寄存器含有有效数据

RXIFn 是接收寄存器状态标志位。当 RXIFn=0，TXR_RXRn 寄存器为空；当 RXIFn=1，TXR_RXRn 寄存器接收到新数据。当数据从移位寄存器加载到 TXR_RXRn 寄存器中，如果 UnCR2 寄存器中的 RIEn=1，则会触发中断。当接收数据时检测到一个或多个错误时，相应的标志位 NFn、FERRn 或 PERRn 会在同一周期内置位。读取 UnSR 寄存器再读 TXR_RXRn 寄存器，如果 TXR_RXRn 寄存器中没有新的数据，那么将清除 RXIFn 标志。

- Bit 1 TIDLEn:** 数据发送完成标志位
0: 数据传输中
1: 无数据传输
TIDLEn 是数据发送完成标志位。若 TIDLEn=0, 数据传输中。当 TXIFn=1 且数据发送完毕或者暂停字被发送时, TIDLEn 置位。TIDLEn=1, TXn 引脚空闲且处于逻辑高状态。读取 UnSR 寄存器再写 TXR_RXRn 寄存器将清除 TIDLEn 位。数据字符或暂停字就绪时, 不会产生该标志位。
- Bit 0 TXIFn:** 发送数据寄存器 TXR_RXRn 状态位
0: 数据还没有从缓冲器加载到移位寄存器中
1: 数据已从缓冲器加载到移位寄存器中 (TXR_RXRn 数据寄存器为空)
TXIFn 是发送数据寄存器为空标志位。若 TXIFn=0, 数据还没有从缓冲器加载到移位寄存器中; 若 TXIFn=1, 数据已从缓冲器中加载到移位寄存器中。读取 UnSR 寄存器再写 TXR_RXRn 寄存器将清除 TXIFn。当 TXENn 被置位, 由于发送缓冲器未滿, TXIFn 也会被置位。

● UnCR1 寄存器

UnCR1 和 UnCR2 是 UARTn 的两个控制寄存器, 用来定义各种 UARTn 功能, 例如 UARTn 的使能与除能、奇偶校验控制和传输数据的长度等等。详细解释如下:

Bit	7	6	5	4	3	2	1	0
Name	UARTENn	BNO n	PRENn	PRTn	STOPSn	TXBRKn	RX8n	TX8n
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

“x”: 未知

- Bit 7 UARTENn:** UARTn 功能使能位
0: UARTn 除能, TXn 和 RXn 脚处于高阻态状态
1: UARTn 使能, TXn 和 RXn 脚作为 UARTn 功能引脚
此位为 UARTn 的使能位。UARTENn=0, UARTn 除能, RXn 和 TXn 处于高阻态状态; UARTENn=1, UARTn 使能, TXn 和 RXn 将分别由 TXENn 和 RXENn 控制。当 UARTn 被除能将清除缓冲器, 所有缓冲器中的数据将被忽略, 另外波特率计数器、错误和状态标志位被复位, TXENn、RXENn、TXBRKn、RXIFn、OERRn、FERRn、PERRn 和 NFn 清零, 而 TIDLEn、TXIFn 和 RIDLEn 置位, UnCR1、UnCR2 和 BRGn 寄存器中的其它位保持不变。若 UARTn 工作时 UARTENn 清零, 所有发送和接收将停止, 模块也将复位成上述状态。当 UARTn 再次使能时, 它将在上次配置下重新工作。
- Bit 6 BNO n:** 发送数据位数选择位
0: 8-bit 传输数据
1: 9-bit 传输数据
BNO n 是发送数据位数选择位。BNO n=1, 传输数据为 9 位; BNO n=0, 传输数据为 8 位。若选择了 9 位数据传输格式, RX8n 和 TX8n 将分别存储接收和发送数据的第 9 位。
- Bit 5 PRENn:** 奇偶校验使能位
0: 奇偶校验除能
1: 奇偶校验使能
此位为奇偶校验使能位。PRENn=1, 使能奇偶校验; PRENn=0, 除能奇偶校验。
- Bit 4 PRTn:** 奇偶校验选择位
0: 偶校验
1: 奇校验
奇偶校验选择位。PRTn=1, 奇校验; PRTn=0, 偶校验。

- Bit 3 **STOPS_n**: 停止位的长度选择位
0: 有一位停止位
1: 有两位停止位
此位用来设置停止位的长度。STOP_n=1, 有两位停止位; STOP_n=0, 只有一位停止位。
- Bit 2 **TXBRK_n**: 暂停字发送控制位
0: 没有暂停字要发送
1: 发送暂停字
TXBRK_n 是暂停字发送控制位。TXBRK_n=0, 没有暂停字要发送, TX_n 引脚正常操作; TXBRK_n=1, 将会发送暂停字, 发送器将发送逻辑“0”。若 TXBRK_n 为高, 缓冲器中数据发送完毕后, 发送器输出将至少保持 13 位宽的低电平直至 TXBRK_n 复位。
- Bit 1 **RX8_n**: 接收 9-bit 数据传输格式中的第 9 位 (只读)
此位只有在传输数据为 9 位的格式中有效, 用来存储接收数据的第 9 位。BNO_n 是用来控制传输位数是 8 位还是 9 位。
- Bit 0 **TX8_n**: 发送 9-bit 数据传输格式中的第 9 位 (只写)
此位只有在传输数据为 9 位的格式中有效, 用来存储发送数据的第 9 位。BNO_n 是用来控制传输位数是 8 位还是 9 位。

• UnCR2 寄存器

UnCR2 是 UART_n 的第二个控制寄存器, 它的主要功能是控制发送器、接收器以及各种 UART_n 中断源的使能或除能。它也可用来控制波特率, 使能接收唤醒和地址侦测。详细解释如下:

Bit	7	6	5	4	3	2	1	0
Name	TXEN _n	RXEN _n	BRGH _n	ADDEN _n	WAKEN _n	RIEN _n	TIEN _n	TEIEN _n
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7 **TXEN_n**: UART_n 发送使能位
0: UART_n 发送除能
1: UART_n 发送使能
此位为发送使能位。TXEN_n=0, 发送将被除能, 发送器立刻停止工作。另外发送缓冲器将被复位。若 TXEN_n=1 且 UARTE_n=1, 则发送将被使能, TX_n 引脚将由 UART_n 来控制。在数据传输时清除 TXEN_n 将中止数据发送且复位发送器。
- Bit 6 **RXEN_n**: UART_n 接收使能位
0: UART_n 接收除能
1: UART_n 接收使能
此位为接收使能位。RXEN_n=0, 接收将被除能, 接收器立刻停止工作。另外接收缓冲器将被复位。若 RXEN_n=1 且 UARTE_n=1, 则接收将被使能, RX_n 引脚将由 UART_n 来控制。在数据传输时清除 RXEN_n 将中止数据接收且复位接收器。
- Bit 5 **BRGH_n**: 波特率发生器高/低速选择位
0: 低速波特率
1: 高速波特率
此位为波特率发生器高/低速选择位, 它和 BRG_n 寄存器一起控制 UART_n 的波特率。若 BRGH_n=0, 为低速模式。
- Bit 4 **ADDEN_n**: 地址检测使能位
0: 地址检测除能
1: 地址检测使能
此位为地址检测使能和除能位。ADDEN_n=1, 地址检测使能, 此时数据的第 8 位 (BNO_n=0) 或第 9 位 (BNO_n=1) 为高, 那么接到的是地址而非数据。若相应的中断使能且接收到的值最高位为 1, 那么中断请求标志将会被置位, 若地址检测功能使能且最高位为 0, 那么将不会产生中断且收到的数据也会被忽略。

- Bit 3 **WAKEn**: RXn 脚下降沿唤醒 UARTn 功能使能位
 0: RXn 脚下降沿唤醒 UARTn 功能除能
 1: RXn 脚下降沿唤醒 UARTn 功能使能
 此位用于控制 RXn 引脚下降沿时是否唤醒 UARTn 功能。此位仅当 UARTn 时钟源 f_H 关闭时有效。若 UARTn 时钟源 f_H 还开启, 则无 RXn 引脚唤醒 UARTn 功能无效。若此位置高且 UARTn 时钟 f_H 关闭, 当 RXn 引脚发生下降沿时会产生 UARTn 唤醒请求。若相应的中断使能, 将产生 RXn 引脚唤醒 UARTn 的中断, 以告知单片机使其通过应用程序开启 UARTn 时钟源 f_H , 从而唤醒 UARTn 功能。否则, 若此位为低, 即使 RXn 引脚发生下降沿也无法恢复 UARTn 功能。
- Bit 2 **RIEn**: 接收中断使能位
 0: 接收中断除能
 1: 接收中断使能
 此位为接收中断使能或除能位。若 RIEn=1, 当 OERRn 或 RXIFn 置位时, UARTn 的中断请求标志置位; 若 RIEn=0, UARTn 中断请求标志不受 OERRn 和 RXIFn 影响。
- Bit 1 **TIEn**: 发送器空闲中断使能位
 0: 发送器空闲中断除能
 1: 发送器空闲中断使能
 此位为发送器空闲中断的使能或除能位。若 TIEn=1, 当发送器空闲触发 TIDLEn 置位时, UARTn 的中断请求标志置位; 若 TIEn=0, UARTn 中断请求标志不受 TIDLEn 的影响。
- Bit 0 **TEIEn**: 发送寄存器为空中断使能位
 0: 发送寄存器为空中断除能
 1: 发送寄存器为空中断使能
 此位为发送寄存器为空中断的使能或除能位。若 TEIEn=1, 当发送器为空中断触发 TXIFn 置位时, UARTn 的中断请求标志置位; 若 TEIEn=0, UARTn 中断请求标志不受 TXIFn 的影响。

波特率发生器

UARTn 自身具有一个波特率发生器, 通过它可以设定数据传输速率。波特率是由一个独立的内部 8 位计数器产生, 它由 BRGn 寄存器和 UnCR2 寄存器的 BRGHn 位来控制。BRGHn 是决定波特率发生器处于高速模式还是低速模式, 从而决定计算公式的选用。BRGn 寄存器的值 N 可根据下表中的公式计算, N 的范围是 0 到 255。

UnCR2 的 BRGHn 位	0	1
波特率 (BR)	$f_H / [64 (N+1)]$	$f_H / [16 (N+1)]$

为得到相应的波特率, 首先需要设置 BRGHn 来选择相应的计算公式从而算出 BRGn 的值。由于 BRGn 的值不连续, 所以实际波特率和理论值之间有一个偏差。下面举例怎样计算 BRGn 寄存器中的值 N 和误差。

• BRGn 寄存器

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

“x”: 未知

- Bit 7~0 **D7~D0**: 波特率值
 软件设置 BRGHn 位 (设置波特率发生器的速度) 和 BRGn 寄存器 (设置波特率的值), 一起控制 UARTn 的波特率。

波特率和误差的计算

若选用 4MHz 时钟频率且 BRGHn=0，若期望的波特率为 4800，计算它的 BRGn 寄存器的值 N，实际波特率和误差。

根据上表，波特率 $BR = f_H / [64 (N+1)]$

转换后的公式 $N = [f_H / (BR \times 64)] - 1$

带入参数 $N = [4000000 / (4800 \times 64)] - 1 = 12.0208$

取最接近的值，十进制 12 写入 BRGn 寄存器，实际波特率如下

$BR = 4000000 / [64 \times (12+1)] = 4808$

因此，误差 = $(4808 - 4800) / 4800 = 0.16\%$

UART 模块的设置与控制

UARTn 采用标准的不归零码传输数据，这种方法通常被称为 NRZ 法。它由 1 位起始位，8 位或 9 位数据位和 1 位或者两位停止位组成。奇偶校验是由硬件自动完成的，可设置成奇校验、偶校验和无校验三种格式。常用的数据传输格式由 8 位数据位，1 位停止位，无校验组成，用 8、N、1 表示，它是系统上电的默认格式。数据位数、停止位数和奇偶校验由 UnCR1 寄存器的 BNO_n、PRT_n、PREN_n 和 STOPS_n 设定。用于数据发送和接收的波特率由一个内部的 8 位波特率发送器产生，数据传输时低位在前高位在后。尽管 UARTn 发送器和接收器在功能上相互独立，但它们使用相同的数据传输格式和波特率，在任何情况下，停止位是必须的。

UART 的使能和除能

UARTn 是由 UnCR1 寄存器的 UARTEN_n 位来使能和除能的。若 UARTEN_n、TXEN_n 和 RXEN_n 都为高，则 TX_n 和 RX_n 分别为 UARTn 的发送端口和接收端口。若没有数据发送，TX_n 引脚默认状态为高电平。

UARTEN_n 清零将除能 TX_n 和 RX_n，通过设置相关引脚共用控制位，这两个引脚可用作普通 I/O 口或其它引脚共用功能。当 UARTn 被除能时将清空缓冲器，所有缓冲器中的数据将被忽略，另外一些使能控制、错误标志和状态标志将被复位，如 TXEN_n、RXEN_n、TXBRK_n、RXIF_n、OERR_n、FERR_n、PERR_n 和 NF_n 清零，而 TIDLE_n、TXIF_n 和 RIDLE_n 置位，UnCR1、UnCR2 和 BRGn 寄存器中的其它位保持不变。若 UARTn 工作时 UARTEN_n 清零，所有发送和接收将停止，模块也将复位成上述状态。当 UARTn 再次使能时，它将在上次配置下重新工作。

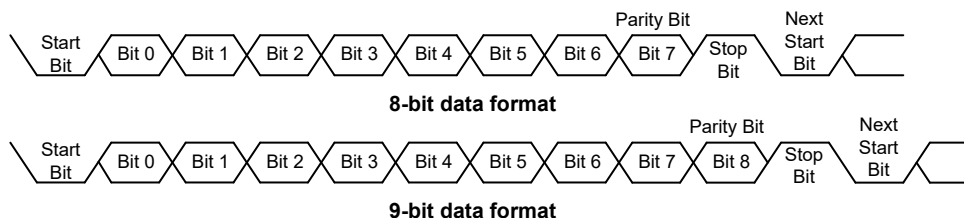
数据位、停止位位数以及奇偶校验的选择

数据传输格式由数据长度、是否校验、校验类型、地址位以及停止位长度组成。它们都是由 UnCR1 寄存器的各个位控制的。BNO_n 决定数据传输是 8 位还是 9 位；PRT_n 决定校验类型；PREN_n 决定是否选择奇偶校验；而 STOPS_n 决定选用 1 位还是 2 位停止位。下表列出了各种数据传输格式。若地址检测功能使能，地址位，即数据字节的最高位，用来确定此帧是地址还是数据。停止位的长度和数据位的长度无关，且只有发送器需设置停止位长度。

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

发送和接收数据格式

下图是传输 8 位和 9 位数据的波形。



UART 发送器

UnCR1 寄存器的 BNO_n 位是控制数据传输的长度。BNO_n=1 其长度为 9 位，第 9 位 MSB 存储在 UnCR1 寄存器的 TX8_n 中。发送器的核心是发送移位寄存器 TSR_n，它的数据由发送寄存器 TXR_RXR_n 提供，应用程序只须将发送数据写入 TXR_RXR_n 寄存器。上组数据的停止位发出前，TSR_n 寄存器禁止写入。如果还有新的数据要发送，一旦停止位发出，待发数据将会从 TXR_RXR_n 寄存器加载到 TSR_n 寄存器。TSR_n 不像其它寄存器一样映射到数据存储器，所以应用程序不能对其进行读写操作。TXEN_n=1，发送使能，但若 TXR_RXR_n 寄存器没有数据或者波特率没有设置，发送器将不会工作。先写 TXR_RXR_n 寄存器再置高 TXEN_n 也会触发发送。当发送器使能，若 TSR_n 寄存器为空，数据写入 TXR_RXR_n 寄存器将会直接加载到 TSR_n 寄存器中。发送器工作时，TXEN_n 清零，发送器将立刻停止工作并且复位，此时通过设置相关引脚共用控制位，TX_n 引脚用作普通 I/O 口或其它引脚共用功能。

发送数据

当 UART_n 发送数据时，数据从移位寄存器中移到 TX_n 引脚上，其低位在前高位在后。在发送模式中，TXR_RXR_n 寄存器在内部总线和发送移位寄存器间形成一个缓冲。如果选择 9 位数据传输格式，最高位 MSB 取自 UnCR1 寄存器的 TX8_n。

发送器初始化可由如下步骤完成：

- 正确地设置 BNO_n、PRT_n、PREN_n 和 STOPS_n 位以确定数据长度、校验类型和停止位长度。
- 设置 BRG_n 寄存器，选择期望的波特率。
- 置高 TXEN_n，使能 UART_n 发送器且使 TX_n 作为 UART_n 的发送端。
- 读取 UnSR 寄存器，然后将待发数据写入 TXR_RXR_n 寄存器。注意，此步骤会清除 TXIF_n 标志位。

如果要发送多个数据只需重复上一步骤。应注意的是当 TXIFn=0 时，数据将禁止写入 TXR_RXRn 寄存器。可以通过以下步骤来清除 TXIFn：

1. 读取 UnSR 寄存器
2. 写 TXR_RXRn 寄存器

只读标志位 TXIFn 由 UARTn 硬件置位。若 TXIFn=1，TXR_RXRn 寄存器为空，其它数据可以写入而不会覆盖之前的数据。若 TEIE=1，TXIFn 标志位会产生中断。在数据传输时，写 TXR_RXRn 指令会将待发数据暂存在 TXR_RXRn 寄存器中，当前数据发送完毕后，待发数据被加载到发送移位寄存器中。当发送器空闲时，写 TXR_RXRn 指令会将数据直接加载到 TSRn 寄存器中，数据传输立刻开始且 TXIFn 置位。当发送完停止位或暂停帧后，表示一帧数据已发送完毕，此时 TIDLEn 位将被置位。可以通过以下步骤来清除 TIDLEn：

1. 读取 UnSR 寄存器
2. 写 TXR_RXRn 寄存器

清除 TXIFn 和 TIDLEn 软件执行次序相同。

发送暂停字

若 TXBRKn=1，下一帧将会发送暂停字。它是由一个起始位、13×N (N=1, 2……) 位逻辑 0 组成。置位 TXBRKn 将会发送暂停字，而清除 TXBRKn 将产生停止位，传输暂停字不会产生中断。需要注意的是，暂停字至少 13 位宽。若 TXBRKn 持续为高，那么发送器会一直发送暂停字；当应用程序将 TXBRKn 清零后，发送器结束最后一帧暂停字的发送后接着发送一位或两位停止位。最后一帧暂停字的结尾自动为高电平，以确保下一帧数据起始位的检测。

UART 接收器

UARTn 接收器支持 8 位或者 9 位数据接收。若 BNO=1，数据长度为 9 位，而最高位 MSB 存放在 UnCR1 寄存器的 TXR_RXRn 中。接收器的核心是串行移位寄存器 RSRn。RXn 引脚上的数据送入数据恢复器中，它在 16 倍波特率的频率下工作，而串行移位器工作在正常波特率下。当在 RXn 引脚上检测到停止位，若 TXR_RXRn 寄存器为空，数据从 RSRn 寄存器中加载到 TXR_RXRn 寄存器。RXn 引脚上的每一位数据会被采样三次以判断其逻辑状态。RSRn 不像其它寄存器一样映射在数据存储区，所以应用程序不能对其进行读写操作。

接收数据

当 UARTn 接收数据时，数据低位在前高位在后，连续地从 RXn 引脚进入移位寄存器。TXR_RXRn 寄存器在内部总线和接收移位寄存器间形成一个缓冲。TXR_RXRn 寄存器是一个两层的 FIFO 缓冲器，它能保存两帧数据的同时接收第三帧数据，应用程序必须保证在接收完第三帧前读取 TXR_RXRn 寄存器，否则忽略第三帧数据并且发生溢出错误。接收器的启动可由如下步骤完成：

- 正确地设置 BNO、PRTn 和 PRENn 位以确定数据长度和校验类型。
- 设置 BRGn 寄存器，选择期望的波特率。
- 置高 RXENn，使能 UARTn 发送器且使 RXn 作为 UARTn 的接收端。

此时接收器被使能并检测起始位。

接收数据将会发生如下事件：

- 当 TXR_RXRn 寄存器中包含有效数据时，UnSR 寄存器中的 RXIFn 位将会置位，溢出错误发生之前至多还有一帧数据可读。
- 若 RIEn=1，数据从 RSRn 寄存器加载到 TXR_RXRn 寄存器中将产生中断。
- 若接收器检测到帧错误、噪声干扰错误、奇偶出错或溢出错误，那么相应的错误标志位置位。

可以通过如下步骤来清除 RXIFn：

1. 读取 UnSR 寄存器
2. 读取 TXR_RXRn 寄存器

接收暂停字

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

UARTn 接收到暂停字会产生以下事件：

- 帧错误标志位 FERRn 置位。
- TXR_RXRn 寄存器清零。
- OERRn、NF n、PERRn、RIDLEn 或 RXIFn 可能会置位。

空闲状态

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

接收中断

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

接收错误处理

UARTn 会产生几种接收错误，下面部分将描述各错误以及怎样处理。

溢出 – OERRn 标志

TXR_RXRn 寄存器是一个两层的 FIFO 缓冲器，它能保存两帧数据的同时接收第三帧数据，应用程序必须保证在接收完第三帧前读取 TXR_RXRn 寄存器，否则发生溢出错误。

产生溢出错误时将会发生以下事件：

- UnSR 寄存器中 OERRn 被置位。
- TXR_RXRn 寄存器中数据不会丢失。
- RSRn 寄存器数据将会被覆盖。
- 若 RIEn=1，将会产生中断。

先读取 UnSR 寄存器再读取 TXR_RXRn 寄存器可将 OERRn 清零。

噪声干扰 – NF_n 标志

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

- 在 RXIF_n 上升沿，UnSR 寄存器中只读标志位 NF_n 置位。
- 数据从 RSR_n 寄存器加载到 TXR_RXR_n 寄存器中。
- 不产生中断，但此位置位发生在 RXIF_n 置位产生中断的同周期内。
- 先读取 UnSR 寄存器再读取 TXR_RXR_n 寄存器可将 NF_n 清零。

帧错误 – FERR_n 标志

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

奇偶校验错误 – PERR_n 标志

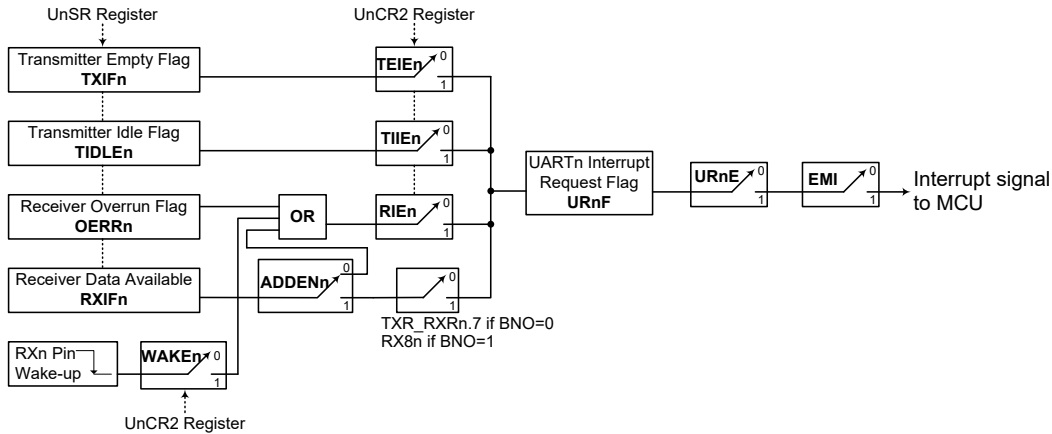
若接收到数据出现奇偶校验错误，UnSR 寄存器中只读标志 PERR_n 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。此标志位同接收的数据分别记录在 UnSR 寄存器和 TXR_RXR_n 寄存器中，此标志位可被任何复位清零。注意，在读取相应的数据之前必须先访问 UnSR 寄存器中的 FERR_n 和 PERR_n 错误标志位。

UART 模块中断结构

几个独立的 UART_n 条件可以产生一个 UART_n 中断。当条件满足时，会产生一个低脉冲信号。发送寄存器为空、发送器空闲、接收器数据有效、溢出和地址检测和 RX_n 引脚唤醒都会产生中断。若 UART_n 中断允许且堆栈未满，程序将会跳转到相应的中断向量执行中断服务程序，而后再返回主程序。其中四种情况，若其 UnCR2 寄存器中相应中断允许位被置位，则 UnSR 寄存器中对应中断标志位将产生 UART_n 中断。发送器相关的两个中断情况有各自对应的中断允许位，而接收器相关的两个中断情况共用一个中断允许位。这些允许位可用于禁止个别的 UART_n 中断源。

地址检测也是 UART_n 的中断源，它没有相应的标志位，若 UnCR2 寄存器中 ADDEN_n=1，当检测到地址将会产生 UART_n 中断。RX_n 引脚唤醒也可以产生 UART_n 中断，它没有相应的标志位，当 UART_n 时钟源 f_{rt} 关闭且 UnCR2 中的 WAKEN 和 RIEN 位被置位，RX 引脚上有下降沿时会产生 UART_n 中断。应注意 RX_n 唤醒中断发生时将会有一定时间的延迟，即系统上电时间，从而允许振荡器在系统恢复正常操作之前重启并达到稳定。

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



UARTn 中断框图

地址检测模式

置位 UnCR2 寄存器中的 ADDENn 将启动地址检测模式。若此位为“1”，可产生接收数据有效中断，其请求标志位为 RXIFn。若 ADDENn 有效，只有在接收到数据最高位为 1 才会产生中断，中断允许位 URnE 和 EMI 也要使能才会产生中断。地址的最高位为第 9 位 (BNO=1) 或第 8 位 (BNO=0)，若此位为高，则接收到的是地址而非数据。只有接收的数据的最后一位为高才会产生中断。若 ADDENn 除能，每接收到一个有效数据便会置位 RXIFn，而不用考虑数据的最后一位。地址检测和奇偶校验在功能上相互排斥，若地址检测模式使能，为了确保操作正确，必须将奇偶校验使能位清零以除能奇偶校验。

ADDENn	Bit 9 (BNO=1) Bit 8 (BNO=0)	产生 UARTn 中断
0	0	√
	1	√
1	0	×
	1	√

ADDENn 位功能

UART 模块暂停和唤醒

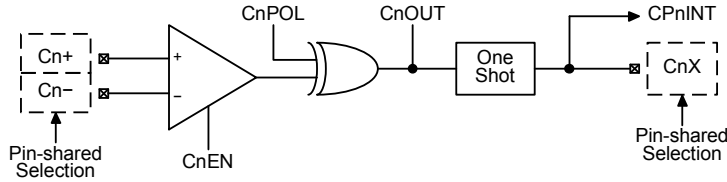
UARTn 时钟 f_H 关闭后 UARTn 模块将停止运行。当传送数据时 UARTn 时钟 f_H 关闭，发送将停止直到 UARTn 模块时钟再次使能。同样地，当接收数据时单片机进入空闲或休眠模式，数据接收也会停止。当单片机进入空闲或休眠模式，UnSR、UnCR1、UnCR2、接收/发送寄存器以及 BRGn 寄存器都不会受到影响。建议在单片机进入空闲或休眠模式前先确保数据发送或接收已完成。

UARTn 功能中包括了 RXn 引脚的唤醒功能，由 UnCR2 寄存器中 WAKEn 位控制。当 UARTn 时钟 f_H 关闭时，若 WAKEn 位与 UARTn 允许位 UARTENn、接收器允许位 RXENn 和接收器中断允许位 RIEEn 都被置位，则 RXn 引脚的下降沿可触发产生 RXn 引脚唤醒 UARTn 的中断。唤醒后系统需延时一段时间才能正常工作，在此期间，RXn 引脚上的任何数据将被忽略。

若要唤醒并产生 UARTn 中断，除了唤醒使能控制位和接收中断使能控制位需置位外，全局中断允许位 EMI 和 UARTn 中断使能控制位 URnE 也必须置位；若这两控制位没有被置位，那么，单片机将可以被唤醒但不会产生中断。同样唤醒后系统需一定的延时才能正常工作，然后才会产生 UARTn 中断。

比较器

该系列单片机包含两个独立的模拟比较器，通过寄存器控制可灵活提供暂停，极性选择，响应时间等功能。比较器引脚与正常 I/O 端口共用引脚，当比较器功能未使用时，此引脚可做普通引脚使用而不浪费 I/O 资源。



比较器 (n=0~1)

比较器操作

该系列单片机包含两个比较器功能，用于比较两个模拟电压，它们的差值上提供一个输出。控制寄存器 CP0C 和 CP1C 分别用于控制两个内部比较器。比较器输出可由控制寄存器中的一个位记录，也可传输到共用的 I/O 引脚上。比较器功能还包括输出极性选择，响应时间和暂停控制。

当比较器使能，与比较器输入共用的引脚上连接的任何上拉电阻将自动断开。当比较器输入接近其切换电压时，由于输入信号的上升 / 下降速度慢，比较器输出端可能会产生伪输出信号。通过选择迟滞功能，向比较器提供少量正反馈给比较器以减少上述情况的发生。当比较器工作在输入失调校准模式时，迟滞功能将被除能。

理想情况下，比较器将在正 / 负输入信号在同一个电压点时发生切换动作，但无法避免的输入失调会造成不确定性。执行失调校准功能将减小切换失调值。比较器还提供了输出响应时间选择功能，通过 CPnC 寄存器中的 CNVTn1~CNVTn0 位段实现。

比较器寄存器

四个控制寄存器控制着每个内部比较器的操作，分别为 CPnC 和 CPnVOS 寄存器。这些寄存器中的相应位有着同样的功能，详情列于下表。

寄存器名称	位							
	7	6	5	4	3	2	1	0
CPnC	—	CnEN	CnPOL	CnOUT	CNVTn1	CNVTn0	—	—
CPnVOS	—	CnOFM	CnRSP	CnOF4	CnOF3	CnOF2	CnOF1	CnOF0

比较器寄存器列表 (n=0~1)

● CPnC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	CnEN	CnPOL	CnOUT	CNVtn1	CNVtn0	—	—
R/W	—	R/W	R/W	R	R/W	R/W	—	—
POR	—	0	0	0	0	0	—	—

- Bit 7 未定义，读为“0”
- Bit 6 **CnEN**: 比较器 n 使能控制
0: 除能
1: 使能
此位用于使能比较器功能。若此位为 0 则即使在其输入引脚上输入模拟电压，比较器将关闭以节省功耗。当比较器功能除能，比较器输出低电平。
- Bit 5 **CnPOL**: 比较器输出极性控制
0: 输出同相
1: 输出反相
若此位为零则 CnOUT 位将反映比较器的同相输出情况。若此位置高，则输出反相。
- Bit 4 **CnOUT**: 比较器输出位
CMPnPOL=0
0: Cn+ < Cn-
1: Cn+ > Cn-
CMPnPOL=1
0: Cn+ > Cn-
1: Cn+ < Cn-
此位为比较器输出位，其极性取决于比较器输入上的电压以及 CnPOL 的情况。
- Bit 3~2 **CNVtn1~CNVtn0**: 比较器响应时间选择
00: 响应时间 0 (最大)
01: 响应时间 1
10: 响应时间 2
11: 响应时间 3 (最小)
此位段用于选择比较器响应时间，详细的响应时间特性列于比较器特性表中。
- Bit 1~0 未定义，读为“0”

● CPnVOS 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	CnOFM	CnRSP	CnOF4	CnOF3	CnOF2	CnOF1	CnOF0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	1	0	0	0	0

- Bit 7 未定义，读为“0”
- Bit 6 **CnOFM**: 比较器工作模式选择位
0: 正常操作模式
1: 输入失调校准模式
此位用于使能比较器输入失调校准功能。关于输入失调校准步骤的详细信息请参考“输入失调校准”章节。
- Bit 5 **CnRSP**: 比较器输入失调校准模式参考输入选择
0: 选择 Cn- 作为参考输入
1: 选择 Cn+ 作为参考输入
- Bit 4~0 **CnOF4~CnOF0**: 比较器输入失调校准值
该位段用于实现比较器输入失调校准操作，且用于存储输入失调校准后的值。更多详细信息请参考“输入失调校准”章节。

输入失调校准

要工作在输入失调校准模式，所用的比较器输入引脚应通过正确配置相应的引脚共用功能选择位预先选中，接着将 CnOFM 位置高。详细步骤如下所述：

- 步骤 1. 设置 CnOFM=1，使能比较器输入失调校准模式。
- 步骤 2. 设置 CnOF[4:0]=00000，并读取 CnOUT 位。
- 步骤 3. 将 CnOF[4:0] 位段加一，并读取 CnOUT 位。
若 CnOUT 位状态未改变，重复步骤 3 直到 CnOUT 位状态改变。
若 CnOUT 位状态改变，记录 CnOF[4:0] 位段值为 VCnOS1 并跳到步骤 4。
- 步骤 4. 设置 CnOF[4:0]=11111，并读取 CnOUT 位。
- 步骤 5. 将 CnOF[4:0] 位段减一，并读取 CnOUT 位。
若 CnOUT 位状态未改变，重复步骤 5 直到 CnOUT 位状态改变。
若 CnOUT 位状态改变，记录 CnOF[4:0] 位段值为 VCnOS2 并跳到步骤 6。
- 步骤 6. 重新存储比较器输入失调校准值 VCnOS 到 CnOF[4:0] 位段中。至此，失调校准步骤完成。

$$VCnOS = \frac{VCnOS1 + VCnOS2}{2}$$

比较器中断

每个比较器都有其独立的中断功能。当任一输出位状态改变，其相应的中断标志位将置位，若相应的中断使能位置位，则程序将跳转至其相应的中断向量。应注意，产生中断的是 CnOUT 位状态的变化而非输出引脚。当单片机处于休眠或空闲模式且比较器使能，若此时外部输入导致比较器输出改变状态，所产生的中断标志位也将产生唤醒。若要防止唤醒的发生，则需在单片机进入休眠或空闲模式前应将中断标志位置高。

编程注意事项

若比较器使能，则当单片机进入休眠或空闲模式时比较器将保持开启，但这会产生一定的功耗。因此用户需要考虑在进入休眠或空闲模式之前将比较器除能。由于比较器引脚与 I/O 引脚共用，若比较器功能使能，这些引脚的 I/O 寄存器将读为“0”（端口控制寄存器为“1”）或读为端口数据寄存器的值（端口控制寄存器为“0”）。

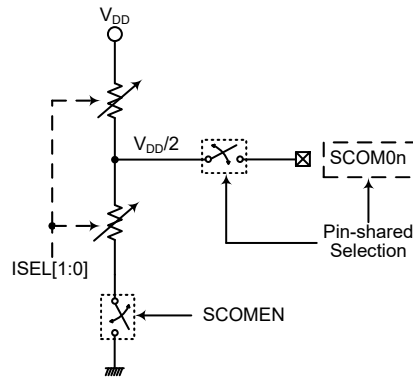
软件控制 LCD 驱动

该系列单片机具有驱动外部 LCD 面板的能力。LCD 驱动的 COM 脚 SCOM0~SCOM3 与 I/O 口共用。LCD 的 COM 控制信号通过应用程序实现。

LCD 操作

该系列单片机通过设置相关 I/O 口为 COM 引脚以驱动外部的晶体面板。LCD 驱动功能是由几个 LCD 控制寄存器一起控制的，另外，这些寄存器还可设置 LCD 的开启和关闭以及 SCOMn 引脚 R 型偏压电流，使得 LCD COM 驱动产生必需的电压 V_{SS} 、 $V_{DD}/2$ 和 V_{DD} ，从而实现 1/2 偏压操作。

SCOMC 寄存器中的 SCOMEN 位是 LCD 驱动的主控制位。通过相应的引脚共用功能选择位选择 SCOMn 引脚用于 LCD 驱动。需注意的，端口控制寄存器不需要预先设置为输出以使能 LCD 驱动操作。



软件控制 LCD 驱动结构

LCD 偏压电流控制

LCD COM 驱动器可以提供多种驱动电流选择以适应不同 LCD 面板的需求。通过设置 SCOMC 寄存器 ISEL0 位和 ISEL1 位可以选择不同的偏压电流。所有的 COM 引脚与 I/O 口共用，通过相应的引脚共用功能选择位将这些引脚选作 SCOM 引脚。

• SCOMC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	ISEL1	ISEL0	SCOMEN	—	—	—	—
R/W	—	R/W	R/W	R/W	—	—	—	—
POR	—	0	0	0	—	—	—	—

Bit 7 未定义，读为“0”

Bit 6~5 **ISEL1~ISEL0**: SCOM 典型偏压电流选择位 ($V_{DD}=5V$)

00: 25 μ A

01: 50 μ A

10: 100 μ A

11: 200 μ A

Bit 4 **SCOMEN**: 软件控制 LCD 驱动使能控制位

0: 除能

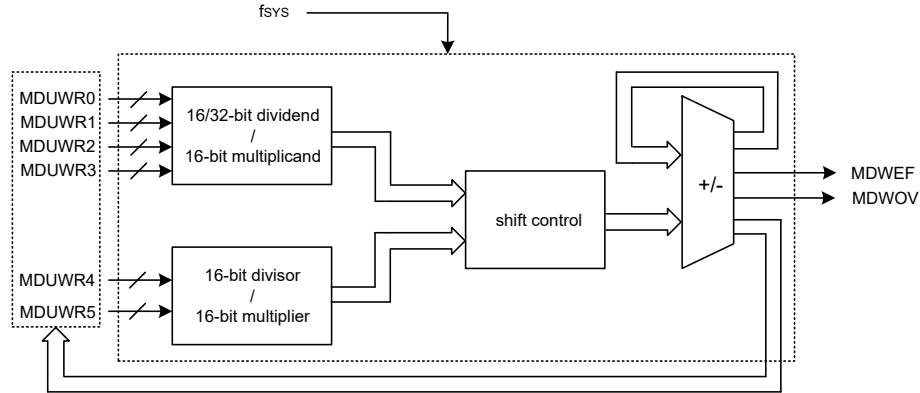
1: 使能

若 SCOMEN 为 1，SCOMn 线可通过配置相应的引脚共用选择位使能。若 SCOMEN 为 0，则 SCOMn 的输出固定在 V_{DD} 电平。应注意相应的引脚共用选择位应在 SCOMn 功能使能之前预先恰当设置。

Bit 3~0 未定义，读为“0”

16 位乘除法单元 – MDU

该系列单片机内置了 16 位乘除法单元，即 MDU，是 16 位 unsigned 乘法与 32 位 /16 位 unsigned 除法器。此乘除法器可取代软件乘除法的运算，节省大量运算时间、程序存储器和数据存储器空间，降低单片机负载，以达到提升整体系统性能。



16-bit MDU 方框图

MDU 寄存器

乘法和除法操作通过特定的步骤实现，即按照特定的顺序对一系列寄存器写值。状态寄存器 MDUWCTRL 用于指示运算操作的状态。六个数据寄存器每个寄存器用于存储不同 MDU 操作中的操作数。

寄存器名称	位							
	7	6	5	4	3	2	1	0
MDUWR0	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR1	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR2	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR3	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR4	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR5	D7	D6	D5	D4	D3	D2	D1	D0
MDUWCTRL	MDWEF	MDWOV	—	—	—	—	—	—

MDU 寄存器列表

• MDUWRn 寄存器 (n=0~5)

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

“x”：未知

Bit 7~0 **D7~D0**: 16-bit MDU 数据寄存器 n

● MDUWCTRL 寄存器

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

- Bit 7 **MDWEF**: 16-bit MDU 错误标志位
 0: 正常
 1: 异常
 如果在运算过程中 MDUWRn 寄存器被改写或被读取, MDWEF 位由硬件自动置位。当运算完成且 MDWEF 位为 1 时, 可通过读取 MDUWCTRL 寄存器将此位清零。
- Bit 6 **MDWOV**: 16-bit MDU 溢出标志位
 0: 溢出未发生
 1: 乘法结果大于 FFFFH 或除数为 0
 每完成一次运算, 硬件自动更新此位以指示当前运算的情况。
- Bit 5~0 未定义, 读为“0”

乘除法单元操作

乘除法单元用于乘法运算还是除法运算取决于寄存器 MDUWR0~MDUWR5 的写入顺序。无论是被除数、被乘数、除数或乘数的低字节数据, 必须在其高字节数据之前先写入对应的 MDU 数据寄存器中。所有的 MDU 操作都在已按照正确顺序写入 MDUWRn 寄存器且 MDUWR5 寄存器被写入后开始执行。不一定要连续往 MDU 数据寄存器中写数据, 但必须要按照正确的顺序写入。因此, 只要不影响写操作, 在正确的写入顺序中间允许插入非写入 MDUWRn 的指令或中断。写入顺序与 MDU 运算的对应关系如下:

- 32-bit/16-bit 除法运算: 依序从 MDUWR0 写到 MDUWR5
- 16-bit/16-bit 除法运算: 依序写入 MDUWR0、MDUWR1、MDUWR4 和 MDUWR5, 跳过 MDUWR2 和 MDUWR3 不写
- 16-bit×16-bit 乘法运算: 依序写入 MDUWR0、MDUWR4、MDUWR1 和 MDUWR5, 跳过 MDUWR2 和 MDUWR3 不写

写顺序确定后, MDU 开始执行对应的运算。不同的 MDU 运算所需的计算时间不同。在执行运算操作过程中, 禁止对六个 MDU 数据寄存器执行读或写动作。当每次运算操作完成后, 应通过 MDUWCTRL 寄存器确认该运算是否正确。若运算正确, 可按照特定顺序从对应的 MDU 数据寄存器中读取运算结果。MDU 运算及其所需的计算时间如下所示:

- 32-bit/16-bit 除法运算: $17 \times t_{\text{SYS}}$
- 16-bit/16-bit 除法运算: $9 \times t_{\text{SYS}}$
- 16-bit×16-bit 乘法运算: $11 \times t_{\text{SYS}}$

运算操作完成后，结果存储在对应的 MDU 数据寄存器中，且需按照特定的顺序读取。不一定要连续从 MDU 数据寄存器中读取结果，但必须按照正确的顺序读取。因此，只要不影响读操作，在正确的读取顺序中间允许插入非读取 MDUWRn 的指令或中断。运算结果读取顺序与 MDU 运算的对应关系如下：

- 32-bit/16-bit 除法运算：依序从 MDUWR0 到 MDUWR3 读取商数，依序从 MDUWR4 和 MDUWR5 读取余数
- 16-bit/16-bit 除法运算：依序从 MDUWR0 和 MDUWR1 读取商数，依序从 MDUWR4 和 MDUWR5 读取余数
- 16-bit×16-bit 乘法运算：依序从 MDUWR0 到 MDUWR3 读取乘积

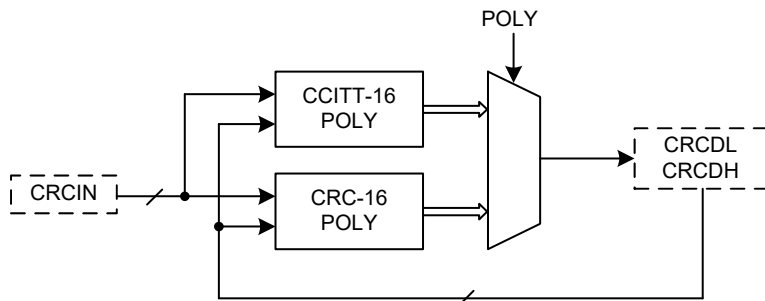
MDU 读 / 写顺序以及计算时间的注意事项总结如下表所示。

运算 事项	32-bit / 16-bit 除法	16-bit / 16-bit 除法	16-bit × 16-bit 乘法
写顺序 最先写 ↓ ↓ ↓ ↓ ↓ 最后写	被除数字节 0 写入 MDUWR0 被除数字节 1 写入 MDUWR1 被除数字节 2 写入 MDUWR2 被除数字节 3 写入 MDUWR3 除数字节 0 写入 MDUWR4 除数字节 1 写入 MDUWR5	被除数字节 0 写入 MDUWR0 被除数字节 1 写入 MDUWR1 除数字节 0 写入 MDUWR4 除数字节 1 写入 MDUWR5	被乘数字节 0 写入 MDUWR0 乘数字节 0 写入 MDUWR4 被乘数字节 1 写入 MDUWR1 乘数字节 1 写入 MDUWR5
计算时间	17 × t _{sys}	9 × t _{sys}	11 × t _{sys}
读顺序 最先读 ↓ ↓ ↓ ↓ 最后读	从 MDUWR0 读取商数字节 0 从 MDUWR1 读取商数字节 1 从 MDUWR2 读取商数字节 2 从 MDUWR3 读取商数字节 3 从 MDUWR4 读取余数字节 0 从 MDUWR5 读取余数字节 1	从 MDUWR0 读取商数字节 0 从 MDUWR1 读取商数字节 1 从 MDUWR4 读取余数字节 0 从 MDUWR5 读取余数字节 1	从 MDUWR0 读取乘积字节 0 从 MDUWR1 读取乘积字节 1 从 MDUWR2 读取乘积字节 2 从 MDUWR3 读取乘积字节 3

MDU 操作总结

循环冗余校验 – CRC

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



CRC 方框图

CRC 寄存器

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
CRCIN	D7	D6	D5	D4	D3	D2	D1	D0
CRCDL	D7	D6	D5	D4	D3	D2	D1	D0
CRCDH	D7	D6	D5	D4	D3	D2	D1	D0
CRCCR	—	—	—	—	—	—	—	POLY

CRC 寄存器列表

• CRCIN 寄存器

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

Bit 7~0 **D7~D0**: CRC 输入数据寄存器

• CRCDL 寄存器

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

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

● **CRC DH 寄存器**

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

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

● **CRCCR 寄存器**

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

Bit 7~1 未定义，读为“0”

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

CRC 操作

CRC 发生器提供了基于 CRC16 和 CCITT CRC16 多项式的 16-bit CRC 计算结果。在该 CRC 发生器中，仅有两个多项式可用于数值计算，不支持其它生成多项式的 16-bit CRC 计算结果。

下方两个表达式可用于 CRC 生成多项式，通过 CRCCR 控制寄存器中的 POLY 位选择。CRC 计算结果称为 CRC 校验和 CRCSUM，并存储于 CRC 校验和寄存器对 CRC DH 和 CRC DL 中。

- CRC-CCITT: $X^{16} + X^{12} + X^5 + 1$
- CRC-16: $X^{16} + X^{15} + X^2 + 1$

CRC 计算

每次对 CRC IN 寄存器进行写操作，都会将存储在 CRC DH 和 CRC DL 寄存器对中的前个 CRC 值和新的输入数据结合起来。CRC 单元计算 CRC 数据寄存器值是按字节进行的。CRC 校验和的计算需要一个 MCU 指令周期。

CRC 计算步骤

步骤 1. 清除校验和寄存器对 CRC DH 和 CRC DL。

步骤 2. 对 8-bit 输入数据字节和 16-bit CRCSUM 高字节进行异或操作，其结果称为临时 CRCSUM。

步骤 3. 将临时 CRCSUM 值左移一位，并向最低有效位 LSB 填入“0”。

步骤 4. 检查在步骤 3 中完成移位后的临时 CRCSUM 值。

若 MSB 为“0”，则该移位后的临时 CRCSUM 将作为新的临时 CRCSUM。

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

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

步骤 5. 重复步骤 3 到步骤 4，直到输入数据字节的所有位都经过计算。

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

CRC 计算范例

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

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

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

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

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

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

程序存储器 CRC 校验和计算范例：

- 清除校验和寄存器对 CRCDH 和 CRCDL。
- 通过 CRCCR 寄存器中的 POLY 位选择 CRC-CCITT 或 CRC-16 多项式作为生成多项式。
- 执行表格读取指令，读取程序存储器数据值。
- 将表格数据低字节写入 CRCIN 寄存器，并结合当前 CRCSUM 值进行 CRC 计算。计算后将得到一个新的 CRCSUM 值并存储在 CRC 校验和寄存器对 CRCDH 和 CRCDL 中。
- 将表格数据高字节写入 CRCIN 寄存器，并结合当前 CRCSUM 值进行 CRC 计算。计算后将得到一个新的 CRCSUM 值并存储在 CRC 校验和寄存器对 CRCDH 和 CRCDL 中。
- 重复步骤 3 到步骤 5 以读取下一个程序存储器数据值并执行 CRC 计算，直到读取了所有的程序存储器数据，接着进行连续的 CRC 计算。计算后 CRC 校验和寄存器中的值为最终的 CRC 计算结果。

低电压检测 – LVD

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

LVD 寄存器

低电压检测功能由 LVDC 寄存器控制。VLVD2~VLVD0 位用于选择 8 个固定电压中的一个参考点。LVDO 位被置位时低电压情况发生，若 LVDO 位为低表明 V_{DD} 电压工作在当前所设置低电压水平值之上。LVDEN 位用于控制低电压检测功能的开启/关闭，设置此位为高使能此功能，反之，关闭内部低电压检测电路。低电压检测会有一些的功耗，在不使用时可考虑关闭此功能，此举在功耗要求严格的电池供电应用中值得考虑。

• LVDC 寄存器

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

Bit 7~6 未定义，读为“0”

Bit 5 **LVDO**: LVD 输出标志位

- 0: 未检测到低电压
- 1: 检测到低电压

Bit 4 **LVDEN**: 低电压检测控制位

- 0: 除能
- 1: 使能

Bit 3 **VBGEN**: Bandgap 电压输出控制位

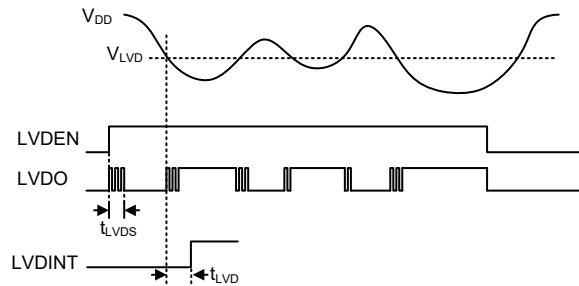
- 0: 除能
- 1: 使能

Bit 2~0 **VLVD2~VLVD0**: 选择 LVD 电压位

- 000: 2.0V
- 001: 2.2V
- 010: 2.4V
- 011: 2.7V
- 100: 3.0V
- 101: 3.3V
- 110: 3.6V
- 111: 4.0V

LVD 操作

通过比较电源电压 V_{DD} 与存储在 LVDC 寄存器中的预置电压值的结果，低电压检测功能工作。其设置的范围为 2.0V~4.0V。当电源电压 V_{DD} 低于预置电压值时，LVDO 位被置为高，表明低电压产生。低电压检测功能由一个自动使能的参考电压提供。当单片机进入休眠模式时，即使 LVDEN 位为高，低电压检测器将自动被除能。低电压检测器使能后，读取 LVDO 位前，电路稳定需要一定的延时 t_{LVDS} 。注意， V_{DD} 电压可能上升或下降比较缓慢，在 V_{LVD} 电压值附近时，LVDO 位可能有多种变化。



LVD 操作

低电压检测器中断包含在多功能中断中。它是除了轮询 LVDO 位之外的另一种检测低电压的方法。中断条件产生置位 LVDO 并延时 t_{LVD} 后，中断产生。若 LVDEN 位为高，当单片机掉电时低电压检测器保持有效状态。此种情况下，若 V_{DD} 降至小于 LVD 预置电压值时，中断请求标志位 LVF 将被置位，中断产生，单片机将从空闲模式中被唤醒。若不要求低电压检测的唤醒功能使能，在单片机进入空闲模式前应将 LVF 标志置为高。

中断

中断是单片机一个重要功能。当外部事件或内部功能如发生触摸动作或定时 / 计数器溢出等，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。此系列单片机提供多个外部中断和内部中断功能，外部中断由 INT0~INT3 引脚，而内部中断由各种内部功能，如定时器模块、时基、LVD、EEPROM、SIM、UART 和 A/D 转换器等产生。

中断寄存器

中断控制基本上是在一定单片机条件发生时设置请求标志位，应用程序中中断使能位的设置是通过位于特殊功能数据存储器中的一系列寄存器控制的。寄存器总的分为三类。第一类是 INTC0~INTC3 寄存器，用于设置基本的中断；第二类是 MF10~MF15 寄存器，用于设置多功能中断；第三类是 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，紧接着是中断号 (可选)，最后的字母 “E” 代表使能 / 除能位，“F” 代表请求标志位。

功能	使能位	请求标志位	注释
总中断	EMI	—	—
INTn 脚	INTnE	INTnF	n = 0 ~ 3
比较器	CPnE	CPnF	n = 0 ~ 1
A/D 转换器	ADE	ADF	—
时基	TBnE	TBnF	n = 0 ~ 1
多功能	MFnE	MFnF	n = 0 ~ 5
UART	URnE	URnF	n = 0 ~ 1 (HT66F2350/60) n = 0 ~ 2 (HT66F2370/90)
LVD	LVE	LVF	—
EEPROM 写操作	DEE	DEF	—
SIM	SIME	SIMF	—
SPIA	SPIAE	SPIAF	—
PTM	PTMnPE	PTMnPF	n = 0 ~ 3
	PTMnAE	PTMnAF	
STM	STMnPE	STMnPF	n = 0 ~ 2
	STMnAE	STMnAF	

中断寄存器位命名模式

寄存器名称	位							
	7	6	5	4	3	2	1	0
INTEG	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	CP0F	INT1F	INT0F	CP0E	INT1E	INT0E	EMI
INTC1	ADF	MF1F	MF0F	CP1F	ADE	MF1E	MF0E	CP1E
INTC2	MF3F	TB1F	TB0F	MF2F	MF3E	TB1E	TB0E	MF2E
INTC3	MF5F	MF4F	INT3F	INT2F	MF5E	MF4E	INT3E	INT2E
MF10	STM0AF	STM0PF	PTM0AF	PTM0PF	STM0AE	STM0PE	PTM0AE	PTM0PE
MF11	STM1AF	STM1PF	PTM1AF	PTM1PF	STM1AE	STM1PE	PTM1AE	PTM1PE
MF12	—	—	PTM2AF	PTM2PF	—	—	PTM2AE	PTM2PE
MF13	SIMF	SPIAF	DEF	LVF	SIME	SPIAE	DEE	LVE
MF14	STM2AF	STM2PF	PTM3AF	PTM3PF	STM2AE	STM2PE	PTM3AE	PTM3PE
MF15 (HT66F2350/60)	—	—	UR1F	UR0F	—	—	UR1E	UR0E
MF15 (HT66F2370/90)	—	UR2F	UR1F	UR0F	—	UR2E	UR1E	UR0E

中断寄存器列表

● INTEG 寄存器

Bit	7	6	5	4	3	2	1	0
Name	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **INT3S1~INT3S0**: INT3 脚中断边沿控制位

- 00: 除能中断
- 01: 上升沿中断
- 10: 下降沿中断
- 11: 双沿中断

Bit 5~4 **INT2S1~INT2S0**: INT2 脚中断边沿控制位

- 00: 除能中断
- 01: 上升沿中断
- 10: 下降沿中断
- 11: 双沿中断

Bit 3~2 **INT1S1~INT1S0**: INT1 脚中断边沿控制位

- 00: 除能中断
- 01: 上升沿中断
- 10: 下降沿中断
- 11: 双沿中断

Bit 1~0 **INT0S1~INT0S0**: INT0 脚中断边沿控制位

- 00: 除能中断
- 01: 上升沿中断
- 10: 下降沿中断
- 11: 双沿中断

• INTC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	CP0F	INT1F	INT0F	CP0E	INT1E	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

- Bit 7 未定义，读为“0”
- Bit 6 **CP0F**: 比较器 0 中断请求标志位
0: 无请求
1: 中断请求
- Bit 5 **INT1F**: INT1 中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **INT0F**: INT0 中断请求标志位
0: 无请求
1: 中断请求
- Bit 3 **CP0E**: 比较器 0 中断控制位
0: 除能
1: 使能
- Bit 2 **INT1E**: INT1 中断控制位
0: 除能
1: 使能
- Bit 1 **INT0E**: INT0 中断控制位
0: 除能
1: 使能
- Bit 0 **EMI**: 总中断控制位
0: 除能
1: 使能

• INTC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ADF	MF1F	MF0F	CP1F	ADE	MF1E	MF0E	CP1E
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 **MF1F**: 多功能中断 1 请求标志位
0: 无请求
1: 中断请求
- Bit 5 **MF0F**: 多功能中断 0 请求标志位
0: 无请求
1: 中断请求
- Bit 4 **CP1F**: 比较器 1 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 **MF1E**: 多功能中断 1 控制位
 0: 除能
 1: 使能
- Bit 1 **MF0E**: 多功能中断 0 控制位
 0: 除能
 1: 使能
- Bit 0 **CP1E**: 比较器 1 中断控制位
 0: 除能
 1: 使能

● **INTC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	MF3F	TB1F	TB0F	MF2F	MF3E	TB1E	TB0E	MF2E
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 **MF3F**: 多功能中断 3 请求标志位
 0: 无请求
 1: 中断请求
- Bit 6 **TB1F**: 时基 1 中断请求标志位
 0: 无请求
 1: 中断请求
- Bit 5 **TB0F**: 时基 0 中断请求标志位
 0: 无请求
 1: 中断请求
- Bit 4 **MF2F**: 多功能中断 2 请求标志位
 0: 无请求
 1: 中断请求
- Bit 3 **MF3E**: 多功能中断 3 控制位
 0: 除能
 1: 使能
- Bit 2 **TB1E**: 时基 1 中断控制位
 0: 除能
 1: 使能
- Bit 1 **TB0E**: 时基 0 中断控制位
 0: 除能
 1: 使能
- Bit 0 **MF2E**: 多功能中断 2 控制位
 0: 除能
 1: 使能

• INTC3 寄存器

Bit	7	6	5	4	3	2	1	0
Name	MF5F	MF4F	INT3F	INT2F	MF5E	MF4E	INT3E	INT2E
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 **MF5F**: 多功能中断 5 请求标志位
0: 无请求
1: 中断请求
- Bit 6 **MF4F**: 多功能中断 4 请求标志位
0: 无请求
1: 中断请求
- Bit 5 **INT3F**: INT3 中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **INT2F**: INT2 中断请求标志位
0: 无请求
1: 中断请求
- Bit 3 **MF5E**: 多功能中断 5 控制位
0: 除能
1: 使能
- Bit 2 **MF4E**: 多功能中断 4 控制位
0: 除能
1: 使能
- Bit 1 **INT3E**: INT3 中断控制位
0: 除能
1: 使能
- Bit 0 **INT2E**: INT2 中断控制位
0: 除能
1: 使能

• MF10 寄存器

Bit	7	6	5	4	3	2	1	0
Name	STM0AF	STM0PF	PTM0AF	PTM0PF	STM0AE	STM0PE	PTM0AE	PTM0PE
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 **STM0AF**: STM0 比较器 A 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 6 **STM0PF**: STM0 比较器 P 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 5 **PTM0AF**: PTM0 比较器 A 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **PTM0PF**: PTM0 比较器 P 匹配中断请求标志位
0: 无请求
1: 中断请求

- Bit 3 **STM0AE**: STM0 比较器 A 匹配中断控制位
 0: 除能
 1: 使能
- Bit 2 **STMOPE**: STM0 比较器 P 匹配中断控制位
 0: 除能
 1: 使能
- Bit 1 **PTM0AE**: PTM0 比较器 A 匹配中断控制位
 0: 除能
 1: 使能
- Bit 0 **PTM0PE**: PTM0 比较器 P 匹配中断控制位
 0: 除能
 1: 使能

● **MF11 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	STM1AF	STM1PF	PTM1AF	PTM1PF	STM1AE	STM1PE	PTM1AE	PTM1PE
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 **STM1AF**: STM1 比较器 A 匹配中断请求标志位
 0: 无请求
 1: 中断请求
- Bit 6 **STM1PF**: STM1 比较器 P 匹配中断请求标志位
 0: 无请求
 1: 中断请求
- Bit 5 **PTM1AF**: PTM1 比较器 A 匹配中断请求标志位
 0: 无请求
 1: 中断请求
- Bit 4 **PTM1PF**: PTM1 比较器 P 匹配中断请求标志位
 0: 无请求
 1: 中断请求
- Bit 3 **STM1AE**: STM1 比较器 A 匹配中断控制位
 0: 除能
 1: 使能
- Bit 2 **STM1PE**: STM1 比较器 P 匹配中断控制位
 0: 除能
 1: 使能
- Bit 1 **PTM1AE**: PTM1 比较器 A 匹配中断控制位
 0: 除能
 1: 使能
- Bit 0 **PTM1PE**: PTM1 比较器 P 匹配中断控制位
 0: 除能
 1: 使能

● MF12 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM2AF	PTM2PF	—	—	PTM2AE	PTM2PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6 未定义，读为“0”
- Bit 5 **PTM2AF**: PTM2 比较器 A 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **PTM2PF**: PTM2 比较器 P 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 3~2 未定义，读为“0”
- Bit 1 **PTM2AE**: PTM2 比较器 A 匹配中断控制位
0: 除能
1: 使能
- Bit 0 **PTM2PE**: PTM2 比较器 P 匹配中断控制位
0: 除能
1: 使能

● MF13 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SIMF	SPIAF	DEF	LVF	SIME	SPIAE	DEE	LVE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7 **SIMF**: SIM 中断请求标志位
0: 无请求
1: 中断请求
- Bit 6 **SPIAF**: SPIA 中断请求标志位
0: 无请求
1: 中断请求
- Bit 5 **DEF**: 数据 EEPROM 中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **LVF**: LVD 中断请求标志位
0: 无请求
1: 中断请求
- Bit 3 **SIME**: SIM 中断控制位
0: 除能
1: 使能
- Bit 2 **SPIAE**: SPIA 中断控制位
0: 除能
1: 使能
- Bit 1 **DEE**: 数据 EEPROM 中断控制位
0: 除能
1: 使能
- Bit 0 **LVE**: LVD 中断控制位
0: 除能
1: 使能

● MF14 寄存器

Bit	7	6	5	4	3	2	1	0
Name	STM2AF	STM2PF	PTM3AF	PTM3PF	STM2AE	STM2PE	PTM3AE	PTM3PE
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 **STM2AF**: STM2 比较器 A 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 6 **STM2PF**: STM2 比较器 P 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 5 **PTM3AF**: PTM3 比较器 A 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **PTM3PF**: PTM3 比较器 P 匹配中断请求标志位
0: 无请求
1: 中断请求
- Bit 3 **STM2AE**: STM2 比较器 A 匹配中断控制位
0: 除能
1: 使能
- Bit 2 **STM2PE**: STM2 比较器 P 匹配中断控制位
0: 除能
1: 使能
- Bit 1 **PTM3AE**: PTM3 比较器 A 匹配中断控制位
0: 除能
1: 使能
- Bit 0 **PTM3PE**: PTM3 比较器 P 匹配中断控制位
0: 除能
1: 使能

● MF15 寄存器 – HT66F2350/HT66F2360

Bit	7	6	5	4	3	2	1	0
Name	—	—	UR1F	UR0F	—	—	UR1E	UR0E
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6 未定义，读为“0”
- Bit 5 **UR1F**: UART1 传输中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **UR0F**: UART0 传输中断请求标志位
0: 无请求
1: 中断请求
- Bit 3~2 未定义，读为“0”
- Bit 1 **UR1E**: UART1 传输中断控制位
0: 除能
1: 使能
- Bit 0 **UR0E**: UART0 传输中断控制位
0: 除能
1: 使能

• MF15 寄存器 – HT66F2370/HT66F2390

Bit	7	6	5	4	3	2	1	0
Name	—	UR2F	UR1F	UR0F	—	UR2E	UR1E	UR0E
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

- Bit 7 未定义，读为“0”
- Bit 6 **UR2F**: UART2 传输中断请求标志位
0: 无请求
1: 中断请求
- Bit 5 **UR1F**: UART1 传输中断请求标志位
0: 无请求
1: 中断请求
- Bit 4 **UR0F**: UART0 传输中断请求标志位
0: 无请求
1: 中断请求
- Bit 3 未定义，读为“0”
- Bit 2 **UR2E**: UART2 传输中断控制位
0: 除能
1: 使能
- Bit 1 **UR1E**: UART1 传输中断控制位
0: 除能
1: 使能
- Bit 0 **UR0E**: UART0 传输中断控制位
0: 除能
1: 使能

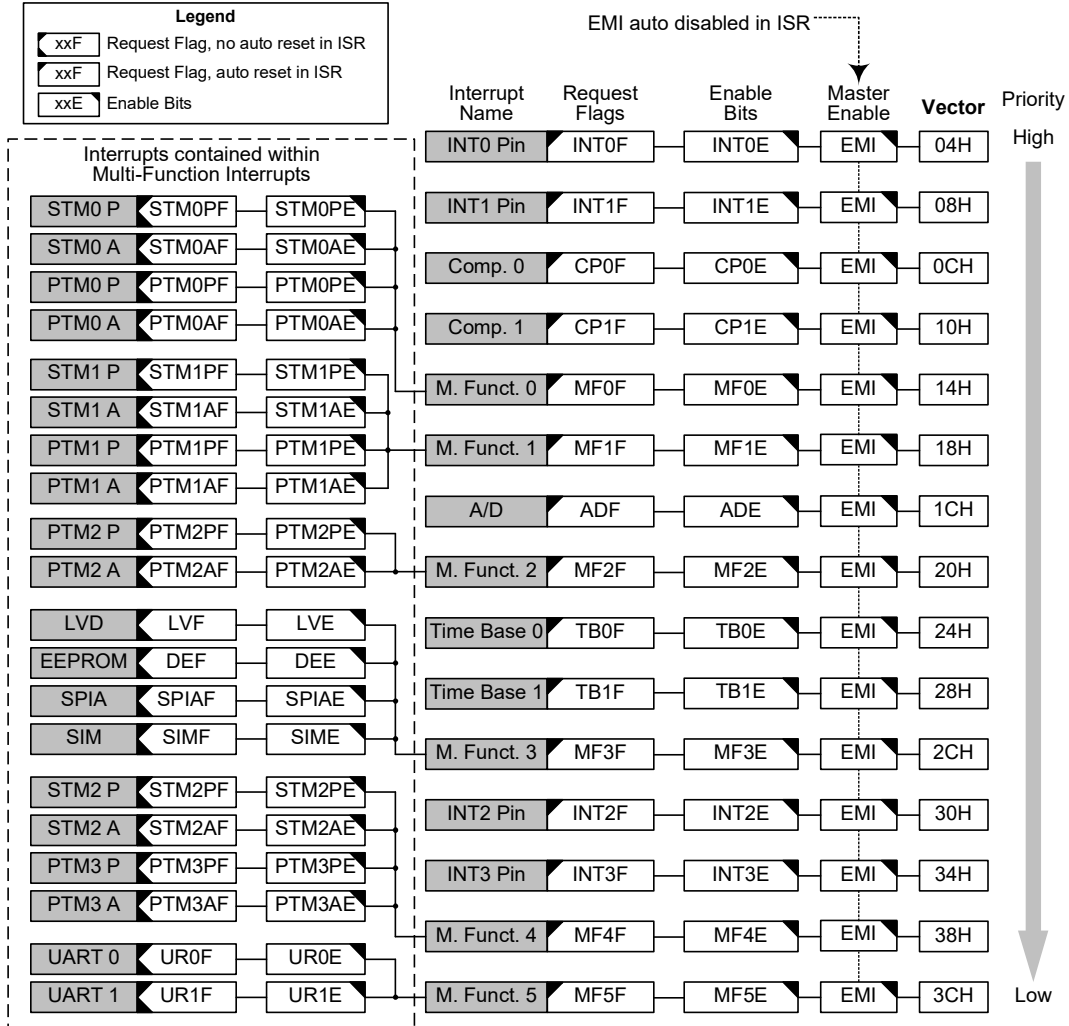
中断操作

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

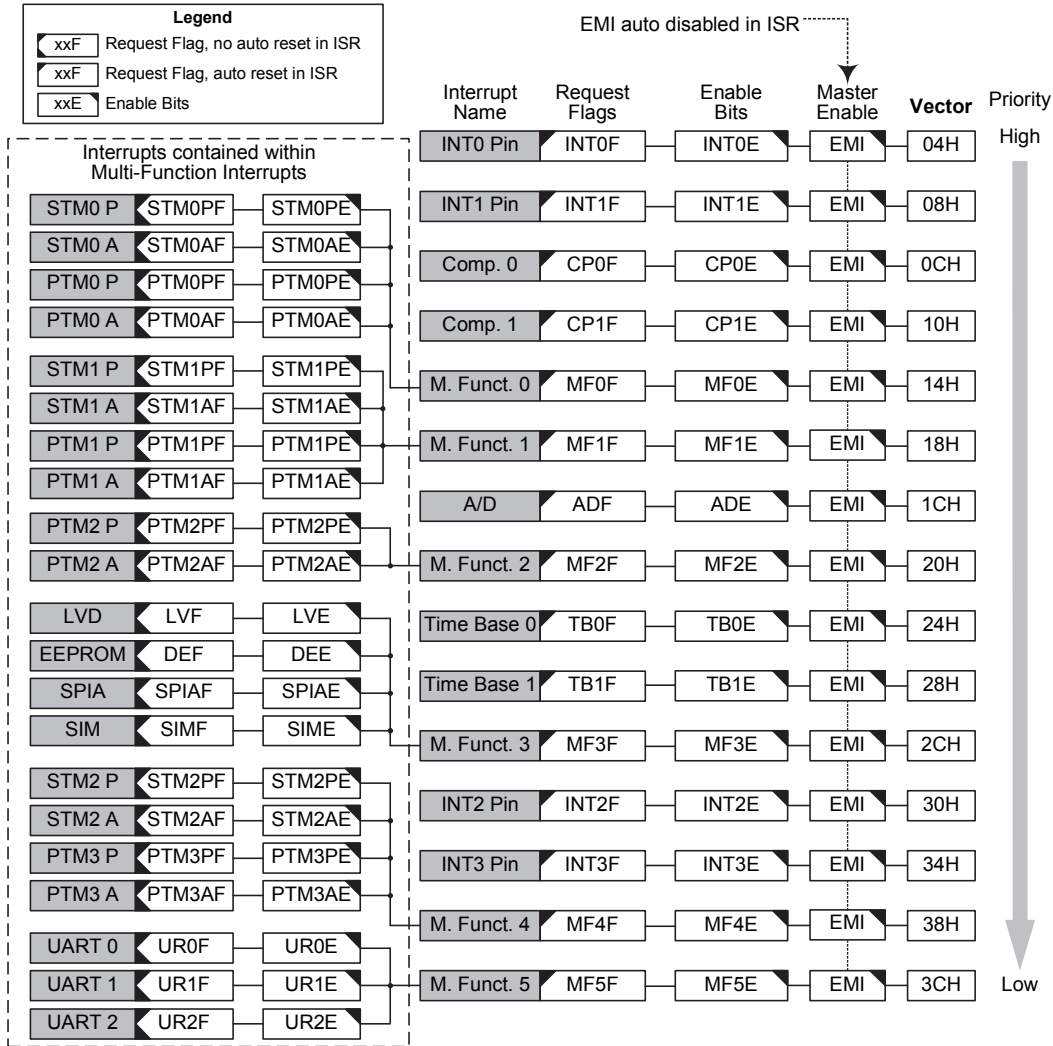
当中断发生时，下条指令的地址将被压入堆栈。相应的中断向量地址加载至 PC 中。系统将从此向量取下条指令。中断向量处通常为跳转指令，以跳转到相应的中断服务程序。中断服务程序必须以“RETI”指令返回至主程序，以继续执行原来的程序。

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

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



中断结构 - HT66F2350/HT66F2360



中断结构 – HT66F2370/HT66F2390

外部中断

通过 INT0~INT3 引脚上的信号变化可控制外部中断。当触发沿选择位设置好触发类型，INT0~INT3 引脚的状态发生变化，外部中断请求标志 INT0F~INT3F 被置位时外部中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和相应中断使能位 INT0E~INT3E 需先被置位。此外，必须使用 INTEG 寄存器使能外部中断功能并选择触发沿类型。外部中断引脚和普通 I/O 口共用，如果相应寄存器中的中断使能位被置位，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 INT0F~INT3F 会自动复位且 EMI 位会被清零以除能其它中断。注意，即使此引脚被用作外部中断输入，其上拉电阻选择仍保持有效。

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。

比较器中断

比较器中断由两个内部比较器控制。当比较器输出位改变状态，比较器中断请求标志位 CP0F 或 CP1F 将被置位，比较器中断产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和比较器中断使能位 CP0E 或 CP1E 需先被设置。当中断使能，堆栈未满且比较器输入导致比较器输出改变状态时，将调用比较器中断向量子程序。当响应比较器中断服务子程序时，EMI 将自动清除以除能其它中断，比较器中断请求标志位也将自动清除。

多功能中断

该系列单片机具有 5 种多功能中断。与其它中断不同，这些没有独立源，但由其它现有的中断源构成，即 TM 中断，LVD 中断，EEPROM 写操作中断，SIM 接口中断，SPIA 接口中断和 UART 接口中断。

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

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

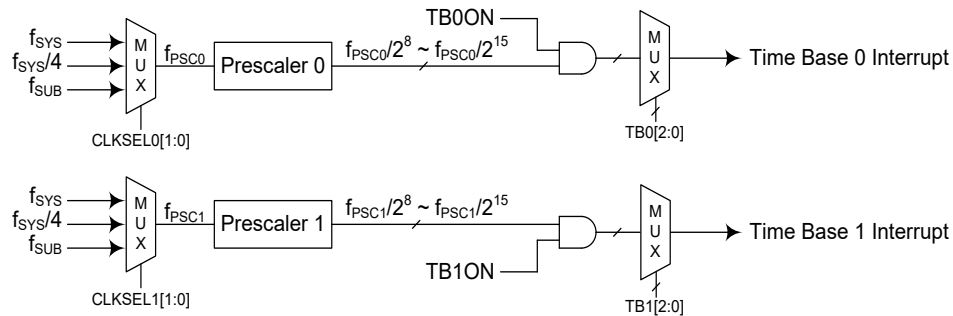
A/D 转换器中断

A/D 转换动作的结束控制 A/D 转换器中断。当 A/D 转换器中断请求标志 ADF 被置位，即 A/D 转换过程完成时，中断请求发生。若要跳转到相应中断向量地址，总中断控制位 EMI、A/D 转换器中断使能位 ADE 需先被置位。当中断使能，堆栈未满且 A/D 转换动作结束时，将调用 A/D 转换器中断向量子程序。当 A/D 转换器中断响应时，ADF 标志将自动清除，EMI 将被自动清零以除能其它中断。

时基中断

时基中断提供一个固定周期的中断信号，由各自的定时器功能产生溢出信号控制。当出现这种情况时其各自的中断请求标志 $TBnF$ 被置位，中断请求发生。若要跳转到其相应的中断向量地址，总中断使能位 EMI 和时基使能位 $TBnE$ 需先被置位。当中断使能，堆栈未滿且时基溢出时，将调用它们各自的中断向量子程序。当响应中断服务子程序时，相应的中断请求标志位 $TBnF$ 会自动复位且 EMI 位会被清零以除能其它中断。

时基中断的目的是提供一个固定周期的中断信号。其时钟源 f_{PSC0} 或 f_{PSC1} 来源于内部时钟源 f_{SYS} ， $f_{SYS}/4$ 或 f_{SUB} ，经过一个分频器，其分频比可通过配置 $TB0C$ 和 $TB1C$ 寄存器中的位选择以获得更长的中断周期。时钟源控制时基中断周期，分别通过 $PSC0R$ 和 $PSC1R$ 寄存器中的 $CLKSEL0[1:0]$ 和 $CLKSEL1[1:0]$ 位进行选择。



时基中断

• PSC0R 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL01	CLKSEL00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义, 读为“0”

Bit 1~0 **CLKSEL01~ CLKSEL00**: 预分频器 0 时钟源 f_{PSC0} 选择
 00: f_{SYS}
 01: $f_{SYS}/4$
 1x: f_{SUB}

• PSC1R 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL11	CLKSEL10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义, 读为“0”

Bit 1~0 **CLKSEL11~ CLKSEL10**: 预分频器 1 时钟源 f_{PSC1} 选择
 00: f_{SYS}
 01: $f_{SYS}/4$
 1x: f_{SUB}

● **TB0C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	—	—	—	—	TB02	TB01	TB00
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB0ON**: 时基 0 使能控制位

0: 除能

1: 使能

Bit 6~3 未定义, 读为“0”

Bit 2~0 **TB02~TB00**: 时基 0 溢出周期选择位

000: $2^8/f_{psc0}$

001: $2^9/f_{psc0}$

010: $2^{10}/f_{psc0}$

011: $2^{11}/f_{psc0}$

100: $2^{12}/f_{psc0}$

101: $2^{13}/f_{psc0}$

110: $2^{14}/f_{psc0}$

111: $2^{15}/f_{psc0}$

● **TB1C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TB1ON	—	—	—	—	TB12	TB11	TB10
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB1ON**: 时基 1 使能控制位

0: 除能

1: 使能

Bit 6~3 未定义, 读为“0”

Bit 2~0 **TB12~TB10**: 时基 1 溢出周期选择位

000: $2^8/f_{psc1}$

001: $2^9/f_{psc1}$

010: $2^{10}/f_{psc1}$

011: $2^{11}/f_{psc1}$

100: $2^{12}/f_{psc1}$

101: $2^{13}/f_{psc1}$

110: $2^{14}/f_{psc1}$

111: $2^{15}/f_{psc1}$

TM 中断

标准型和周期型 TM 各有两个内部中断，即内部比较器 A 或内部比较器 P，当发生比较匹配时将产生 TM 中断。所有的 TM 中断都包含在多功能中断中。不同的 TM 都有两个中断请求标志位和两个使能位。当 TM 比较器 P、A 匹配情况发生时，相应 TM 中断请求标志被置位，TM 中断请求产生。

若要程序跳转到相应中断向量地址，总中断控制位 EMI、相应 TM 中断使能位和相关多功能中断使能位 MFnE 需先被置位。当中断使能，堆栈未满且 TM 比较器匹配情况发生时，可跳转至相关多功能中断向量程序执行。当 TM 中断响应，EMI 将被自动清零以除能其它中断。但 TM 中断标志位必须通过应用程序手动清除。

LVD 中断

低电压检测中断属于多功能中断。当低电压检测功能检测到一个低电压时，LVD 中断请求标志 LVF 被置位，LVD 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI、低电压中断使能位 LVE 和相关的多功能中断使能位需先被置位。当中断使能，堆栈未满且低电压条件发生时，可跳转至 LVD 中断向量程序执行。当低电压中断响应，EMI 将被自动清零以除能其它中断，但 LVF 中断标志位必须通过应用程序手动清除。

EEPROM 中断

EEPROM 写中断属于多功能中断。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 和相关的多功能中断使能位需先被置位。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程序执行。当 EEPROM 中断响应，EMI 将被自动清零以除能其它中断，但 DEF 中断标志位必须通过应用程序手动清除。

串行接口模块中断

串行接口模块中断，即 SIM 中断属于多功能中断。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 和相关的多功能中断使能位需先被置位。当中断使能，堆栈未满且以上任何情况发生时，可跳转至相应中断向量程序执行。当串行接口中断响应，EMI 将被自动清零以除能其它中断，但 SIMF 中断请求标志位必须通过应用程序手动清除。

SPIA 接口中断

SPIA 接口中断属于多功能中断。当一个字节数据已由 SPIA 接口接收或发送完，中断请求标志 SPIAF 被置位，SPIAF 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 SPIAE 和相关的多功能中断使能位需先被置位。当中断使能，堆栈未满且一个字节数据已由 SPIA 接口接收或发送完时，可跳转至相应中断向量程序执行。当串行接口中断响应，EMI 将被自动清零以除能其它中断，但 SPIAF 中断请求标志位必须通过应用程序手动清除。

UART 接口中断

UART 传输中断属于多功能中断。UARTn 模块中，发送器为空、发送器空闲、接收器数据有效、接收器溢出、地址检测和 RX 引脚唤醒都会触发产生 UARTn 中断。当总中断使能位 EMI，UARTn 中断使能位 URnE 和多功能中断使能位被置位，允许程序跳转到相应的中断向量地址。当中断使能，堆栈未滿且以上任何一种情况发生时，将调用 UARTn 中断向量子程序。当响应中断服务子程序时，EMI 位会被清零以除能其它中断，多功能中断请求标志位也会被自动清零，但 UARTn 中断标志位 URnF 必须通过应用程序手动清零。

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，低电压或比较器输入改变都可能导致其相应的中断标志被置位，由此产生中断，因此必须注意避免伪唤醒情况的发生。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。

有的中断为多功能中断，当响应中断服务时，只有多功能中断请求标志 MFnF 会自动清零，其独立中断请求标志需通过应用程序手动清零。

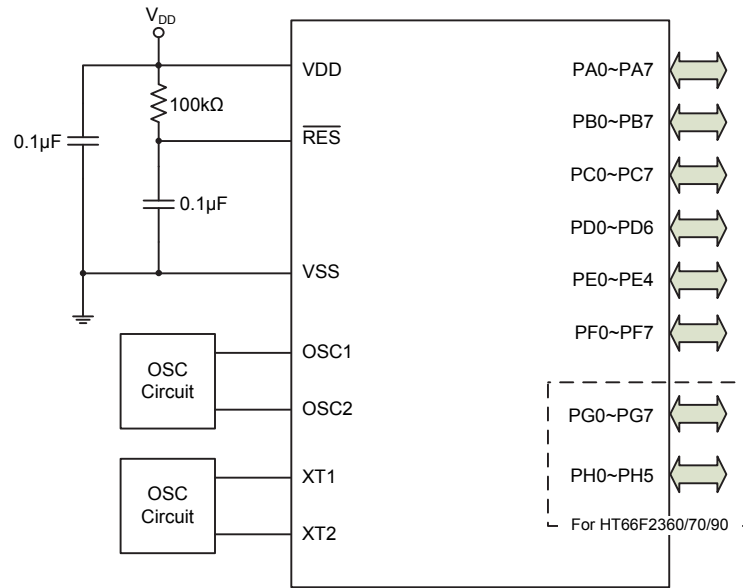
建议在中断服务子程序中不要使用“CALL 子程序”指令。中断通常发生在不可预料的情况或是需要立刻执行的某些应用。假如只剩下一层堆栈且没有控制好中断，当“CALL 子程序”在中断服务子程序中执行时，将破坏原来的控制序列。

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，使用几种 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

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

惯例

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

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

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

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

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

扩展指令集

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

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

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

注: 1. 对扩展跳转指令而言, 如果比较的结果牵涉到跳转即需 3 个周期, 如果没有发生跳转, 则只需两个周期。
 2. 任何扩展指令若要改变 PCL 的内容将需要 3 个周期来执行。

指令定义

ADC A, [m]	Add Data Memory to ACC with Carry
指令说明	将指定的数据存储器、累加器内容以及进位标志相加，结果存放到累加器。
功能表示	$ACC \leftarrow ACC + [m] + C$
影响标志位	OV、Z、AC、C、SC
ADCM A, [m]	Add ACC to Data Memory with Carry
指令说明	将指定的数据存储器、累加器内容和进位标志位相加，结果存放到指定的数据存储器。
功能表示	$[m] \leftarrow ACC + [m] + C$
影响标志位	OV、Z、AC、C、SC
ADD A, [m]	Add Data Memory to ACC
指令说明	将指定的数据存储器和累加器内容相加，结果存放到累加器。
功能表示	$ACC \leftarrow ACC + [m]$
影响标志位	OV、Z、AC、C、SC
ADD A, x	Add immediate data to ACC
指令说明	将累加器和立即数相加，结果存放到累加器。
功能表示	$ACC \leftarrow ACC + x$
影响标志位	OV、Z、AC、C、SC
ADDM A, [m]	Add ACC to Data Memory
指令说明	将指定的数据存储器和累加器内容相加，结果存放到指定的数据存储器。
功能表示	$[m] \leftarrow ACC + [m]$
影响标志位	OV、Z、AC、C、SC
AND A, [m]	Logical AND Data Memory to ACC
指令说明	将累加器中的数据和指定数据存储器内容做逻辑与，结果存放到累加器。
功能表示	$ACC \leftarrow ACC \text{ "AND" } [m]$
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 ← Stack EMI ← 1
影响标志位	无
RL [m]	Rotate Data Memory left
指令说明	将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。
功能表示	$[m].(i+1) \leftarrow [m].i$ (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=0~6) $[m].7 \leftarrow [m].0$
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=0~6) $ACC.7 \leftarrow [m].0$
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=0~6) $[m].7 \leftarrow C$ $C \leftarrow [m].0$
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=0~6) $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
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，结果存放累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$ACC \leftarrow ACC - [m] - \bar{C}$
影响标志位	OV、Z、AC、C、SC、CZ

SBC A, x 指令说明	Subtract immediate data from ACC with Carry 将累加器减去立即数以及进位标志的反，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$ACC \leftarrow ACC - [m] - \bar{C}$
影响标志位	OV、Z、AC、C、SC、CZ
SBCM A, [m] 指令说明	Subtract Data Memory from ACC with Carry and result in Data Memory 将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$[m] \leftarrow ACC - [m] - \bar{C}$
影响标志位	OV、Z、AC、C、SC、CZ
SDZ [m] 指令说明	Skip if Decrement Data Memory is 0 将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	$[m] \leftarrow [m] - 1$ ，如果 $[m]=0$ 跳过下一条指令执行
影响标志位	无
SDZA [m] 指令说明	Skip if decrement Data Memory is zero with result in ACC 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] 指令说明	Set Data Memory 将指定数据存储器的每一位设置为 1。
功能表示	$[m] \leftarrow FFH$
影响标志位	无

<p>SET [m].i 指令说明 功能表示 影响标志位</p>	<p>Set bit of Data Memory 将指定数据存储器的第 i 位置位为 1。 $[m].i \leftarrow 1$ 无</p>
<p>SIZ [m] 指令说明 功能表示 影响标志位</p>	<p>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。 $[m] \leftarrow [m] + 1$，如果 $[m]=0$ 跳过下一条指令执行 无</p>
<p>SIZA [m] 指令说明 功能表示 影响标志位</p>	<p>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。 $ACC \leftarrow [m] + 1$，如果 $ACC=0$ 跳过下一条指令执行 无</p>
<p>SNZ [m].i 指令说明 功能表示 影响标志位</p>	<p>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。 如果 $[m].i \neq 0$，跳过下一条指令执行 无</p>
<p>SNZ [m] 指令说明 功能表示 影响标志位</p>	<p>Skip if Data Memory is not 0 指定数据存储器的内容会先被读出，后又被重新写入指定存储器内。判断指定存储器，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果为 0，则程序继续执行下一条指令。 如果 $[m] \neq 0$，跳过下一条指令执行 无</p>

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

<p>SZ [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Skip if Data Memory is 0</p> <p>指定数据存储器的内容会先被读出，后又被重新写入指定存储器内。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。</p> <p>如果 [m]=0，跳过下一条指令执行</p> <p>无</p>
<p>SZA [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Skip if Data Memory is 0 with data movement to ACC</p> <p>将指定存储器内容复制到累加器，并判断指定存储器内容是否为 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。</p> <p>ACC ←[m]，如果 [m]=0，跳过下一条指令执行</p> <p>无</p>
<p>SZ [m].i 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Skip if bit i of Data Memory is 0</p> <p>判断指定存储器的第 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。</p> <p>如果 [m].i=0，跳过下一条指令执行</p> <p>无</p>
<p>TABRD [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Read table (specific page) to TBLH and Data Memory</p> <p>将表格指针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定存储器且将高字节移至 TBLH。</p> <p>[m] ← 程序代码 (低字节)</p> <p>TBLH ← 程序代码 (高字节)</p> <p>无</p>
<p>TABRDL [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Read table (last page) to TBLH and Data Memory</p> <p>将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定存储器且将高字节移至 TBLH。</p> <p>[m] ← 程序代码 (低字节)</p> <p>TBLH ← 程序代码 (高字节)</p> <p>无</p>

ITABRD [m]	Increment table pointer low byte first and read table (specific page) to TBLH and data memory
指令说明	自加表格指针低字节 TBLP，将表格指针 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定的数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
ITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and data memory
指令说明	自加表格指针低字节 TBLP，将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定的数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
XOR A, [m]	Logical XOR Data Memory to ACC
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。
功能表示	ACC ← ACC “XOR” [m]
影响标志位	Z
XORM A, [m]	Logical XOR ACC to Data Memory
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果放到数据存储器。
功能表示	[m] ← ACC “XOR” [m]
影响标志位	Z
XOR A, x	Logical XOR immediate data to ACC
指令说明	将累加器的数据与立即数逻辑异或，结果存放到累加器。
功能表示	ACC ← ACC “XOR” x
影响标志位	Z

扩展指令定义

扩展指令被用来直接存取存储在任何数据存储器 Sector 中的数据。

LADC A, [m]	Add Data Memory to ACC with Carry
指令说明	将指定的数据存储器、累加器内容以及进位标志相加，结果存放到累加器。
功能表示	$ACC \leftarrow ACC + [m] + C$
影响标志位	OV、Z、AC、C、SC
LADCM A, [m]	Add ACC to Data Memory with Carry
指令说明	将指定的数据存储器、累加器内容和进位标志位相加，结果存放到指定的数据存储器。
功能表示	$[m] \leftarrow ACC + [m] + C$
影响标志位	OV、Z、AC、C、SC
LADD A, [m]	Add Data Memory to ACC
指令说明	将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、SC
LADDM A, [m]	Add ACC to Data Memory
指令说明	将指定的数据存储器和累加器内容相加，结果存放到指定的数据存储器。
功能表示	$[m] \leftarrow ACC + [m]$
影响标志位	OV、Z、AC、C、SC
LAND A, [m]	Logical AND Data Memory to ACC
指令说明	将累加器中的数据和指定数据存储器内容做逻辑与，结果存放到累加器。
功能表示	$ACC \leftarrow ACC \text{ "AND" } [m]$
影响标志位	Z
LANDM A, [m]	Logical AND ACC to Data Memory
指令说明	将指定数据存储器内容和累加器中的数据做逻辑与，结果存放到数据存储器。
功能表示	$[m] \leftarrow ACC \text{ "AND" } [m]$
影响标志位	Z

LCLR [m]	Clear Data Memory
指令说明	将指定数据存储器的内容清零。
功能表示	$[m] \leftarrow 00H$
影响标志位	无
LCLR [m].i	Clear bit of Data Memory
指令说明	将指定数据存储器的第 i 位内容清零。
功能表示	$[m].i \leftarrow 0$
影响标志位	无
LCPL [m]	Complement Data Memory
指令说明	将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1。
功能表示	$[m] \leftarrow \overline{[m]}$
影响标志位	Z
LCPLA [m]	Complement Data Memory with result in ACC
指令说明	将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1，结果被存放回累加器且数据寄存器的内容保持不变。
功能表示	$ACC \leftarrow \overline{[m]}$
影响标志位	Z
LDAA [m]	Decimal-Adjust ACC for addition with result in Data Memory
指令说明	将累加器中的内容转换为 BCD (二进制转成十进制) 码。如果低四位的值大于“9”或 AC=1，那么 BCD 调整就执行对低四位加“6”，否则低四位保持不变；如果高四位的值大于“9”或 C=1，那么 BCD 调整就执行对高四位加“6”。BCD 转换实质上是根据累加器和标志位执行 00H，06H，60H 或 66H 的加法运算，结果存放于数据存储器。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

LDEC [m]	Decrement Data Memory
指令说明	将指定数据存储器的内容减 1。
功能表示	$[m] \leftarrow [m] - 1$
影响标志位	Z
LDECA [m]	Decrement Data Memory with result in ACC
指令说明	将指定数据存储器的内容减 1，把结果存放回累加器并保持指定数据存储器的内容不变。
功能表示	$ACC \leftarrow [m] - 1$
影响标志位	Z
LINC [m]	Increment Data Memory
指令说明	将指定数据存储器的内容加 1。
功能表示	$[m] \leftarrow [m] + 1$
影响标志位	Z
LINCA [m]	Increment Data Memory with result in ACC
指令说明	将指定数据存储器的内容加 1，结果存放回累加器并保持指定的数据存储器内容不变。
功能表示	$ACC \leftarrow [m] + 1$
影响标志位	Z
LMOV A, [m]	Move Data Memory to ACC
指令说明	将指定数据存储器的内容复制到累加器中。
功能表示	$ACC \leftarrow [m]$
影响标志位	无
LMOV [m], A	Move ACC to Data Memory
指令说明	将累加器的内容复制到指定数据存储器。
功能表示	$[m] \leftarrow ACC$
影响标志位	无
LOR A, [m]	Logical OR Data Memory to ACC
指令说明	将累加器中的数据和指定的数据存储器内容逻辑或，结果存放到累加器。
功能表示	$ACC \leftarrow ACC \text{ "OR" } [m]$
影响标志位	Z

LORM A, [m]	Logical OR ACC to Data Memory
指令说明	将存在指定数据存储器中的数据 and 累加器逻辑或，结果放到数据存储器。
功能表示	$[m] \leftarrow \text{ACC} \text{ "OR" } [m]$
影响标志位	Z
LRL [m]	Rotate Data Memory left
指令说明	将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。
功能表示	$[m].(i+1) \leftarrow [m].i \ (i=0\sim6)$ $[m].0 \leftarrow [m].7$
影响标志位	无
LRLA [m]	Rotate Data Memory left with result in ACC
指令说明	将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位，结果送到累加器，而指定数据存储器的内容保持不变。
功能表示	$\text{ACC}.(i+1) \leftarrow [m].i \ (i=0\sim6)$ $\text{ACC}.0 \leftarrow [m].7$
影响标志位	无
LRLC [m]	Rotate Data Memory Left through Carry
指令说明	将指定数据存储器的内容连同进位标志左移 1 位，第 7 位取代进位标志且原本的进位标志移到第 0 位。
功能表示	$[m].(i+1) \leftarrow [m].i \ (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位	C
LRLCA [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	$\text{ACC}.(i+1) \leftarrow [m].i \ (i=0\sim6)$ $\text{ACC}.0 \leftarrow C$ $C \leftarrow [m].7$
影响标志位	C

LRR [m]	Rotate Data Memory right
指令说明	将指定数据存储器的内容循环右移 1 位且第 0 位移到第 7 位。
功能表示	$[m].i \leftarrow [m].(i+1)$ (i=0~6) $[m].7 \leftarrow [m].0$
影响标志位	无
LRRA [m]	Rotate Data Memory right with result in ACC
指令说明	将指定数据存储器的内容循环右移 1 位，第 0 位移到第 7 位，移位结果存放累加器，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=0~6) $ACC.7 \leftarrow [m].0$
影响标志位	无
LRRC [m]	Rotate Data Memory right through Carry
指令说明	将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位。
功能表示	$[m].i \leftarrow [m].(i+1)$ (i=0~6) $[m].7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位	C
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC
指令说明	将指定数据存储器的内容连同进位标志右移 1 位，第 0 位取代进位标志且原本的进位标志移到第 7 位，移位结果送回累加器，但是指定数据寄存器的内容保持不变。
功能表示	$ACC.i \leftarrow [m].(i+1)$ (i=0~6) $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
影响标志位	C
LSBC A, [m]	Subtract Data Memory from ACC with Carry
指令说明	将累加器减去指定数据存储器的内容以及进位标志的反，结果存放累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$ACC \leftarrow ACC - [m] - \bar{C}$
影响标志位	OV、Z、AC、C、SC、CZ

LSBCMA, [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 to 数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$[m] \leftarrow ACC - [m] - \bar{C}$
影响标志位	OV、Z、AC、C、SC、CZ
LSDZ [m]	Skip if Decrement Data Memory is 0
指令说明	将指定的数据存储器的内容减 1，判断是否为 0，若为 0 则跳过下一条指令，由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	$[m] \leftarrow [m] - 1$ ，如果 $[m]=0$ 跳过下一条指令执行
影响标志位	无
LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC
指令说明	将指定数据存储器内容减 1，判断是否为 0，如果为 0 则跳过下一条指令，此结果将存放 to 累加器，但指定数据存储器内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	$ACC \leftarrow [m] - 1$ ，如果 $ACC=0$ 跳过下一条指令执行
影响标志位	无
LSET [m]	Set Data Memory
指令说明	将指定数据存储器的每一个位置位为 1。
功能表示	$[m] \leftarrow FFH$
影响标志位	无
LSET [m].i	Set bit of Data Memory
指令说明	将指定数据存储器的第 i 位置位为 1。
功能表示	$[m].i \leftarrow 1$
影响标志位	无

<p>LSIZ [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Skip if increment Data Memory is 0</p> <p>将指定的数据存储器的内容加 1，判断是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。</p> <p>$[m] \leftarrow [m] + 1$，如果 $[m]=0$ 跳过下一条指令执行</p> <p>无</p>
<p>LSIZA [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Skip if increment Data Memory is zero with result in ACC</p> <p>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，此结果会被存放累加器，但是指定数据存储器的内容不变。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。</p> <p>$ACC \leftarrow [m] + 1$，如果 $ACC=0$ 跳过下一条指令执行</p> <p>无</p>
<p>LSNZ [m].i 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Skip if bit i of Data Memory is not 0</p> <p>判断指定数据存储器的第 i 位，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果为 0，则程序继续执行下一条指令。</p> <p>如果 $[m].i \neq 0$，跳过下一条指令执行</p> <p>无</p>
<p>LSNZ [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Skip if Data Memory is not 0</p> <p>指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定数据存储器，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果为 0，则程序继续执行下一条指令。</p> <p>如果 $[m] \neq 0$，跳过下一条指令执行</p> <p>无</p>
<p>LSUB A, [m] 指令说明</p> <p>功能表示</p> <p>影响标志位</p>	<p>Subtract Data Memory from ACC</p> <p>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。</p> <p>$ACC \leftarrow ACC - [m]$</p> <p>OV、Z、AC、C、SC、CZ</p>

LSUBM 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中的数据，结果存放到指定的数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示	$[m] \leftarrow ACC - [m]$
影响标志位	OV、Z、AC、C、SC、CZ
LSWAP [m] 指令说明	Swap nibbles of Data Memory 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
LSWAPA [m] 指令说明	Swap nibbles of Data Memory with result in ACC 将指定数据存储器的低 4 位和高 4 位互相交换，再将结果存放到累加器且指定数据寄存器的数据保持不变。
功能表示	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
影响标志位	无
LSZ [m] 指令说明	Skip if Data Memory is 0 指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定数据存储器的内容是否为 0，若为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	如果 $[m]=0$ ，跳过下一条指令执行
影响标志位	无
LSZA [m] 指令说明	Skip if Data Memory is 0 with data movement to ACC 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内容是否为 0，若为 0 则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	$ACC \leftarrow [m]$ ，如果 $[m]=0$ ，跳过下一条指令执行
影响标志位	无

LSZ [m].i	Skip if bit i of Data Memory is 0
指令说明	判断指定数据存储器的第 i 位是否为 0，若为 0，则跳过下一条指令。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为 3 个周期的指令。如果结果不为 0，则程序继续执行下一条指令。
功能表示	如果 [m].i=0，跳过下一条指令执行
影响标志位	无
LTABRD [m]	Move the ROM code (specific page) to TBLH and data memory
指令说明	将表格针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
LTABRDL [m]	Read table (last page) to TBLH and Data Memory
指令说明	将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
LITABRD [m]	Increment table pointer low byte first and read table (specific page) to TBLH and data memory
指令说明	自加表格指针低字节 TBLP，将表格针对 TBHP 和 TBLP 所指的程序代码低字节 (指定页) 移至指定的数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无
LITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and data memory
指令说明	自加表格指针低字节 TBLP，将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定的数据存储器且将高字节移至 TBLH。
功能表示	[m] ← 程序代码 (低字节) TBLH ← 程序代码 (高字节)
影响标志位	无

LXOR A, [m]	Logical XOR Data Memory to ACC
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。
功能表示	$ACC \leftarrow ACC \text{ "XOR" } [m]$
影响标志位	Z
LXORM A, [m]	Logical XOR ACC to Data Memory
指令说明	将累加器的数据和指定的数据存储器内容逻辑异或，结果放到数据存储器。
功能表示	$[m] \leftarrow ACC \text{ "XOR" } [m]$
影响标志位	Z

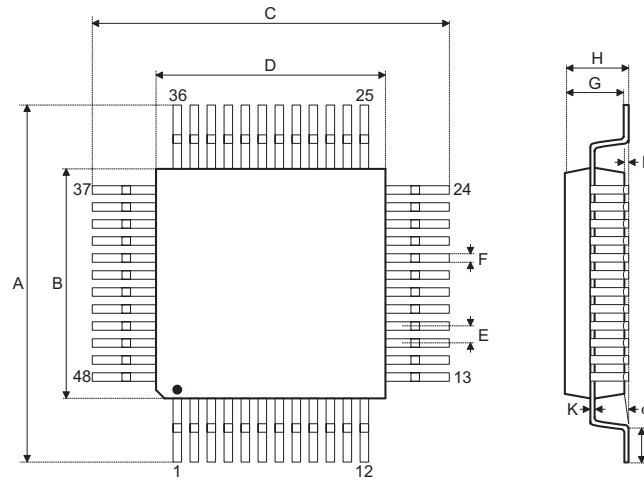
封装信息

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

封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

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

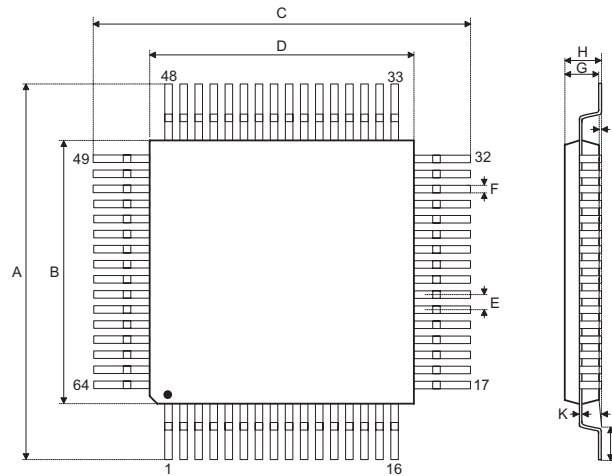
48-pin LQFP (7mm × 7mm) 外形尺寸



符号	尺寸 (单位: inch)		
	最小值	典型值	最大值
A	—	0.354 BSC	—
B	—	0.276 BSC	—
C	—	0.354 BSC	—
D	—	0.276 BSC	—
E	—	0.020 BSC	—
F	0.007	0.009	0.011
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

符号	尺寸 (单位: mm)		
	最小值	典型值	最大值
A	—	9.000 BSC	—
B	—	7.000 BSC	—
C	—	9.000 BSC	—
D	—	7.000 BSC	—
E	—	0.500 BSC	—
F	0.170	0.220	0.270
G	1.350	1.400	1.450
H	—	—	1.600
I	0.050	—	0.150
J	0.450	0.600	0.750
K	0.090	—	0.200
α	0°	—	7°

64-pin LQFP (7mm × 7mm) 外形尺寸



符号	尺寸 (单位: inch)		
	最小值	典型值	最大值
A	—	0.354 BSC	—
B	—	0.276 BSC	—
C	—	0.354 BSC	—
D	—	0.276 BSC	—
E	—	0.016 BSC	—
F	0.005	0.007	0.009
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

符号	尺寸 (单位: mm)		
	最小值	典型值	最大值
A	—	9.00 BSC	—
B	—	7.00 BSC	—
C	—	9.00 BSC	—
D	—	7.00 BSC	—
E	—	0.40 BSC	—
F	0.13	0.18	0.23
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°

Copyright© 2023 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

本文件出版时 HOLTEK 已针对所载信息为合理注意，但不保证信息准确无误。文中提到的信息仅是提供作为参考，且可能被更新取代。HOLTEK 不担保任何明示、默示或法定的，包括但不限于适合商品化、令人满意的质量、规格、特性、功能与特定用途、不侵害第三方权利等保证责任。HOLTEK 就文中提到的信息及该信息之应用，不承担任何法律责任。此外，HOLTEK 并不推荐将 HOLTEK 的产品使用在会由于故障或其他原因而可能会对人身安全造成危害的地方。HOLTEK 特此声明，不授权将产品使用于救生、维生或安全关键零部件。在救生 / 维生或安全应用中使用 HOLTEK 产品的风险完全由买方承担，如因该等使用导致 HOLTEK 遭受损害、索赔、诉讼或产生费用，买方同意出面进行辩护、赔偿并使 HOLTEK 免受损害。HOLTEK (及其授权方，如适用) 拥有本文件所提供信息 (包括但不限于内容、数据、示例、材料、图形、商标) 的知识产权，且该信息受著作权法和其他知识产权法的保护。HOLTEK 在此并未明示或暗示授予任何知识产权。HOLTEK 拥有不事先通知而修改本文件所载信息的权利。如欲取得最新的信息，请与我们联系。

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for [32-bit Microcontrollers - MCU category](#):

Click to view products by [Holtek manufacturer](#):

Other Similar products are found below :

[MCF51AC256AVFUE](#) [MCF51AC256BCFUE](#) [MCF51AC256BVFUE](#) [MB91F464AAPMC-GSE2](#) [R5S726B0D216FP#V0](#) [MB91F248PFV-GE1](#) [MB91243PFV-GS-136E1](#) [SAK-TC1782F-320F180HR BA](#) [TC364DP64F300WAAKXUMA1](#) [R5F566NNDDFP#30](#)
[R5F566NNDDFC#30](#) [R5F566NNDDBD#20](#) [MC96F8216ADBN](#) [A96G181HDN](#) [A96G140KNN](#) [A96G174FDN](#) [A31G213CL2N](#)
[A96G148KNN](#) [A96G174AEN](#) [AC33M3064TLBN-01](#) [V3s](#) [T3](#) [A40i-H](#) [V526](#) [A83T](#) [R11](#) [V851s](#) [A133](#) [V833](#) [F1C100S](#) [T3L](#) [T507](#) [A33](#)
[A63](#) [T113-i](#) [H616](#) [V853](#) [V533](#) [R16-J](#) [V536-H](#) [A64-H](#) [V831](#) [V3LP](#) [T113-S3](#) [F1C200S](#) [F133-A](#) [R128-S2](#) [D1-H](#) [ADUCM360BCPZ128-TR](#)
[APT32S003F8PT](#)