内置 EEPROM A/D+LCD 型低功耗 Flash 单片机

HT67F60A/HT67F70A

版本 : V1.30　日期 : 2022-09-01

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
引脚说明 12
极限参数 20
直流电气特性 21
交流电气特性 23
A/D 转换器特性 24
LVD&LVR 电气特性 25
比较器电气特性 25
LCD 电气特性 26
上电复位特性 26
系统结构 27
时序和流水线结构 27
程序计数器 28
堆栈 28
算术逻辑单元 – ALU 29
Flash 程序存储器 29
结构 29
特殊向量 30
查表 30
查表范例 31
在线烧录 – ICP 32
片上调试 – OCDS 32
在应用编程 – IAP 33
在应用编程控制寄存器 33
Flash 存储器写功能使能步骤 37
Flash 存储器写步骤 39
数据存储器 42
结构 42
数据存储器寻址 43
通用数据存储器 43
特殊功能数据存储器 43

特殊功能寄存器 46
间接寻址寄存器 – IAR0, IAR1, IAR2 46
存储器指针 – MP0, MP1H/MP1L, MP2H/MP2L 46
存储区指针 – BP 48
累加器 – ACC 48
程序计数器低字节寄存器 – PCL 49
表格寄存器 – TBLP, TBHP, TBLH 49
状态寄存器 – STATUS 49
EEPROM 数据存储器 51
EEPROM 数据存储器结构 51
EEPROM 寄存器 51
从 EEPROM 中读取数据 52
写数据到 EEPROM 53
写保护 53
EEPROM 中断 53
编程注意事项 53
振荡器 55
振荡器概述 55
系统时钟配置 55
外部晶体 / 陶瓷振荡器 – HXT 56
内部高速 RC 振荡器 – HIRC 57
外部 32.768kHz 晶体振荡器 – LXT 57
内部 32kHz 振荡器 – LIRC 58
辅助振荡器 58
工作模式和系统时钟 59
系统时钟 59
系统工作模式 60
控制寄存器 61
快速唤醒 62
工作模式切换 63
静态电流的注意事项 67
唤醒 67
看门狗定时器 68
看门狗定时器时钟源 68
看门狗定时器控制寄存器 68
看门狗定时器操作 69
复位和初始化 70
复位功能 70
复位初始状态 74
输入 / 输出端口 79
上拉电阻 80
PA 口唤醒 80
输入 / 输出端口控制寄存器 80

引脚共用功能 80
输入 / 输出引脚结构 90
编程注意事项 91
定时器模块 – TM 91
简介 91
TM 操作 92
TM 时钟源 92
TM 中断 92
TM 外部引脚 92
TM 输入 / 输出引脚控制寄存器 93
编程注意事项 94
简易型 TM 95
简易型 TM 操作 95
简易型 TM 寄存器介绍 95
简易型 TM 工作模式 99
标准型 TM – STM 105
标准型 TM 操作 105
标准型 TM 寄存器介绍 106
标准型 TM 工作模式 110
增强型 TM – ETM 120
增强型 TM 操作 120
增强型 TM 寄存器介绍 121
增强型 TM 工作模式 126
模拟 / 数字转换器 143
A/D 简介 143
A/D 转换寄存器介绍 144
A/D 操作 146
A/D 输入引脚 147
A/D 转换率及时序图 148
A/D 转换步骤 148
编程注意事项 149
A/D 转换功能 149
A/D 转换应用范例 150
比较器 152
比较器操作 152
比较器寄存器 152
比较器中断 154
编程注意事项 154
串行接口模块 – SIM 155
SPI 接口 155
I^2C 接口 161
外围时钟输出 170
外围时钟操作 170
外围时钟寄存器 170

SPIA 串行接口模块 – SPIA 171
SPIA 接口操作 171
SPIA 寄存器 172
SPIA 通信 175
SPIA 总线使能 / 除能 177
SPIA 操作 177
错误侦测 178
LCD 驱动 179
LCD 存储器 180
LCD 时钟源 181
LCD 寄存器 181
LCD 电压源偏压 183
LCD 驱动输出 184
编程注意事项 187
中断 188
中断寄存器 188
中断操作 199
外部中断 200
比较器中断 201
A/D 转换器中断 201
时基中断 201
多功能中断 203
串行接口模块中断 203
SPIA 接口中断 203
外围设备中断 203
LVD 中断 203
EEPROM 中断 204
TM 中断 204
中断唤醒功能 204
编程注意事项 204
低电压检测 – LVD 205
LVD 寄存器 205
LVD 操作 206
配置选项 206
应用电路 207
指令集 208
简介 208
指令周期 208
数据的传送 208
算术运算 208
逻辑和移位运算 208
分支和控制转换 209
位运算 209
查表运算 209

其它运算 209
指令集概要 210
惯例 210
扩展指令集 213
指令定义 215
扩展指令定义 227
封装信息 237
48-pin LQFP (7mm × 7mm) 外形尺寸 238
64-pin LQFP (7mm × 7mm) 外形尺寸 239
80-pin LQFP (10mm × 10mm) 外形尺寸 240

注意，规格中提到的 HT67F70A 产品已终止，目前不再使用。

特性

CPU 特性

- 工作电压：
  - f_{SYS}=8MHz：2.2V~5.5V
  - f_{SYS}=12MHz：2.7V~5.5V
  - f_{SYS}=20MHz：4.5V~5.5V
- V_{DD}=5V，系统时钟为 20MHz 时，指令周期为 0.2μs
- 提供暂停和唤醒功能，以降低功耗
- 振荡模式：
  - 外部晶振 – HXT
  - 外部 32.768kHz 晶振 – LXT
  - 内部高速 RC – HIRC
  - 内部低速 32kHz RC – LIRC
- 内部集成 4/8/12MHz 振荡器，无需外接元件
- 多种工作模式：正常、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 条指令
- 多达 16 层堆栈
- 位操作指令

周边特性

- Flash 程序存储：16K×16 ~ 32K×16
- RAM 数据存储：1024×8 ~ 2048×8
- True EEPROM 存储器：128×8
- 提供在应用编程功能 – IAP
- 看门狗定时器功能
- 多达 47 个双向 I/O 口
- 多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
- 串行接口模块 – SIM，用于 SPI 或 I^2C 通信
- 一个串行 SPIA 接口
- 双比较器功能
- 双时基功能，可提供固定时间的中断信号
- 12 通道 12 位分辨精度的 A/D 转换器
- 低电压复位功能

- 低电压检测功能
- 多种封装类型
- Flash 程序存储器烧录可达 10,000 次
- Flash 程序存储器数据可保存 10 年以上
- True EEPROM 数据存储器烧录可达 100,000 次
- True EEPROM 数据存储器数据可保存 10 年以上

概述

HT67Fx0A 系列单片机是一款 LCD 型具有 8 位高性能精简指令集的 Flash 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用户提供了较大的方便。存储器方面，还包含了一个 RAM 数据存储器和一个可用于存储序号、校准数据等非易失性数据的 True EEPROM 存储器。

在模拟特性方面，该系列单片机包含一个多通道 12 位 A/D 转换器和双比较器功能。还带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 和 I^2C 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。

此系列单片机提供了丰富的 HXT、LXT、HIRC 和 LIRC 振荡器功能选项，且内建完整的系统振荡器，无需外围元器件。其在不同工作模式之间动态切换的能力，为用户提供了一个优化单片机操作和减少功耗的手段。

外加时基功能、I/O 使用灵活等其它特性，使这款单片机可以广泛应用于各种产品中，例如电子测量仪器、环境监控、手持式测量工具、家庭应用、电子控制工具、马达控制等方面。

选型表

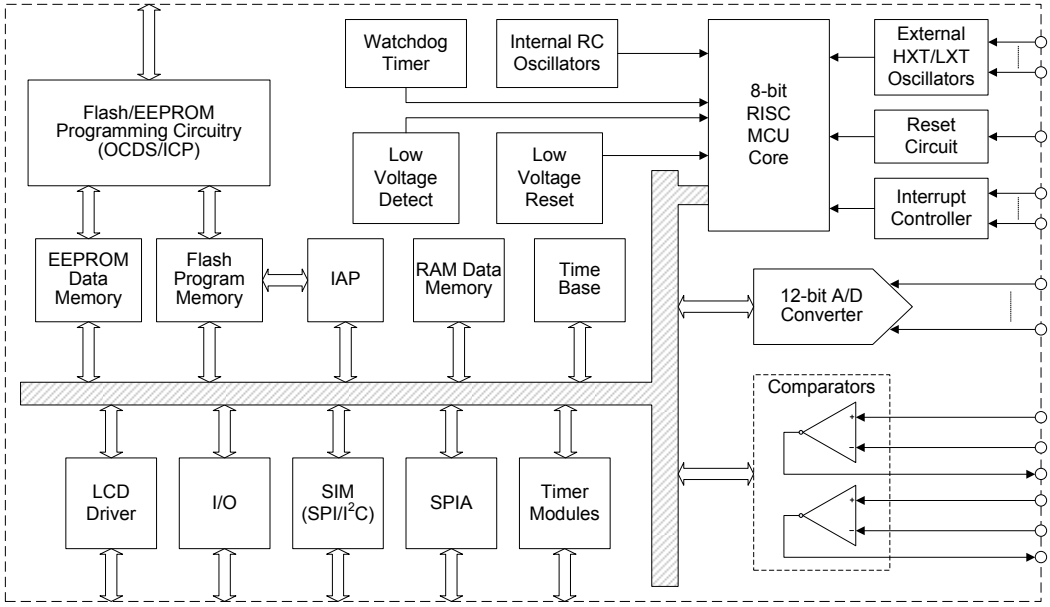
对此系列的单片机而言，大多数的特性参数都是一样的。主要差异在于程序存储器的容量和数据存储器容量。下表列出了各单片机的主要特性。

型号	ROM	RAM	EEPROM	I/O	外部中断	A/D	TM 模块
HT67F60A	16K×16	1024×8	128×8	47	4	12-bit×12	10-bit CTM×2 16-bit STM×3 10-bit ETM×1
HT67F70A	32K×16	2048×8	128×8	47	4	12-bit×12	10-bit CTM×2 16-bit STM×3 10-bit ETM×1

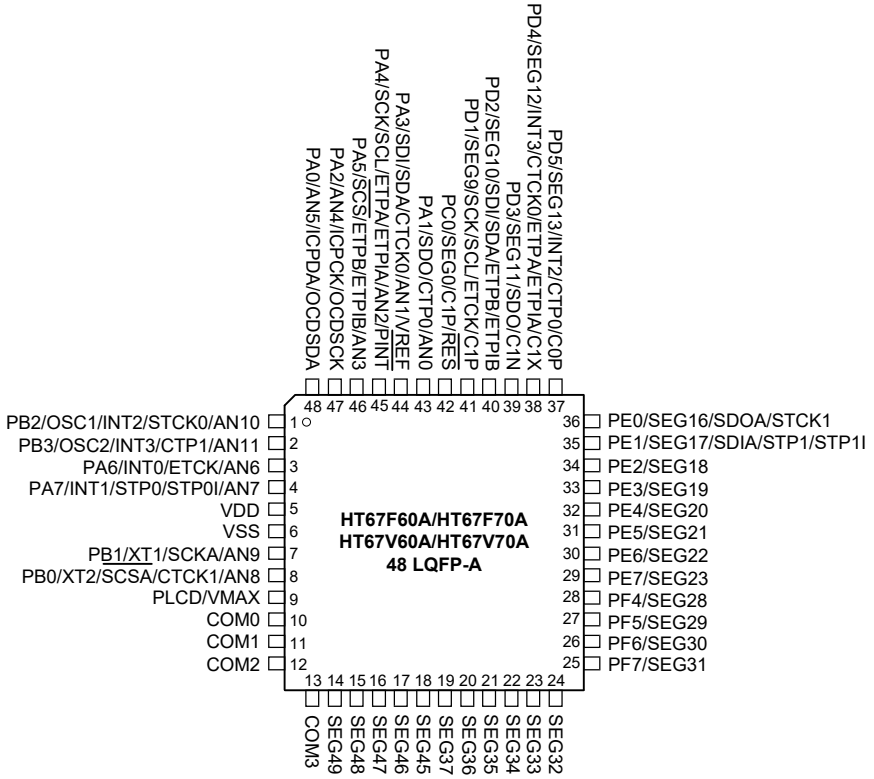
型号	SIM	SPIA	时基	比较器	LCD 驱动器	堆栈	封装类型
HT67F60A	√	√	2	2	56×4	16	48/64/80LQFP
HT67F70A	√	√	2	2	56×4	16	48/64/80LQFP

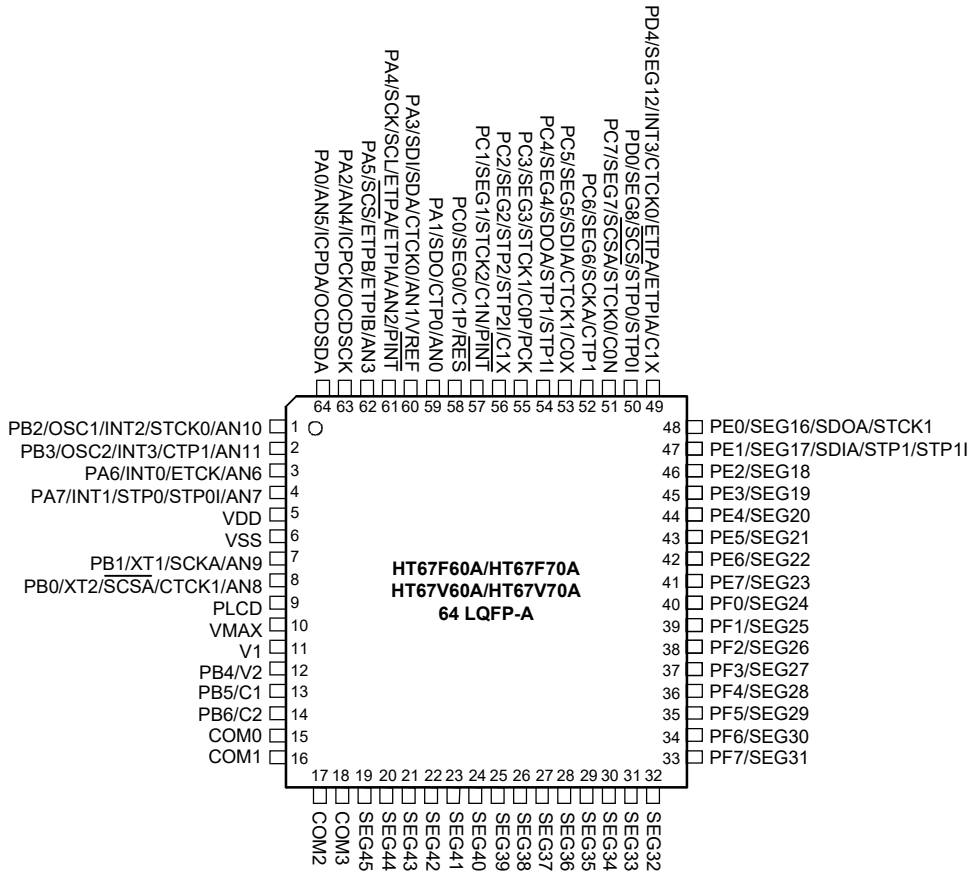
注：对于有不止一种封装形式的芯片，选型表针对较大的封装的情况。

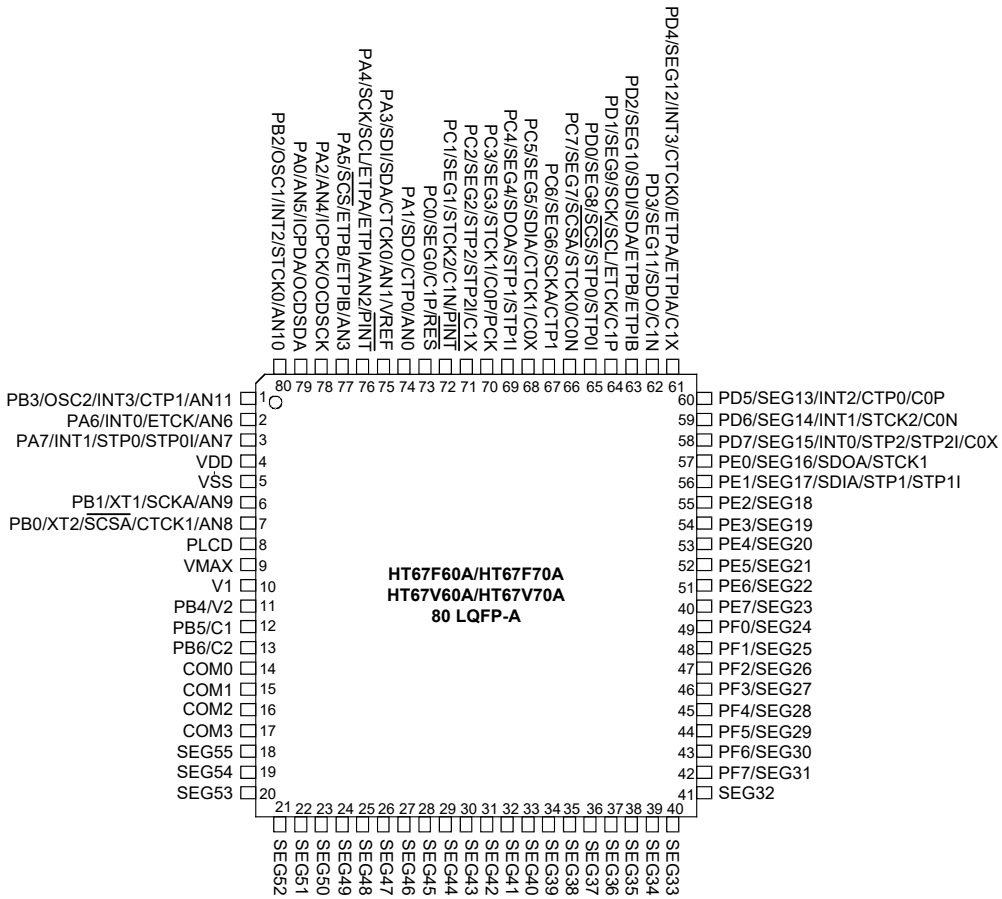
方框图


Watchdog Timer
Internal RC Oscillators
8-bit RISC MCU Core
External HXT/LXT Oscillators
Reset Circuit
Interrupt Controller
Flash/EEPROM Programming Circuitry (OCDS/ICP)
Low Voltage Detect
Low Voltage Reset
EEPROM Data Memory
Flash Program Memory
IAP
RAM Data Memory
Time Base
12-bit A/D Converter
Comparators
LCD Driver
I/O
SIM (SPI/I2C)
SPIA
Timer Modules

引脚图


PA0/AN5/ICPDA/OCDSDA
PA2/AN4/ICPCK/OCDSCK
PA5/SCS/ETPB/ETPIB/AN3
PA4/SCK/SCL/ETPA/ETPIA/AN2/PINT
PA3/SDI/SDA/CTCK0/AN1/VREF
PA1/SDO/CTP0/AN0
PC0/SEG0/C1P/RES
PD1/SEG9/SCK/SCL/ETCK/C1P
PD2/SEG10/SDI/SDA/ETPB/ETPIB
PD3/SEG11/SDO/C1N
PD4/SEG12/INT3/CTCK0/ETPA/ETPIA/C1X
PD5/SEG13/INT2/CTP0/C0P
48 47 46 45 44 43 42 41 40 39 38 37
PB2/OSC1/INT2/STCK0/AN10 1
PB3/OSC2/INT3/CTP1/AN11 2
PA6/INT0/ETCK/AN6 3
PA7/INT1/STP0/STP0I/AN7 4
VDD 5
VSS 6
PB1/XT1/SCKA/AN9 7
PB0/XT2/SCSA/CTCK1/AN8 8
PLCD/VMAX 9
COM0 10
COM1 11
COM2 12
HT67F60A/HT67F70A
HT67V60A/HT67V70A
48 LQFP-A
36 PE0/SEG16/SDOA/STCK1
35 PE1/SEG17/SDIA/STP1/STP1I
34 PE2/SEG18
33 PE3/SEG19
32 PE4/SEG20
31 PE5/SEG21
30 PE6/SEG22
29 PE7/SEG23
28 PF4/SEG28
27 PF5/SEG29
26 PF6/SEG30
25 PF7/SEG31
13 14 15 16 17 18 19 20 21 22 23 24
COM3
SEG49
SEG48
SEG47
SEG46
SEG45
SEG37
SEG36
SEG35
SEG34
SEG33
SEG32


PB2/OSC1/INT2/STCK0/AN10 1
PB3/OSC2/INT3/CTP1/AN11 2
PA6/INT0/ETCK/AN6 3
PA7/INT1/STP0/STP0I/AN7 4
VDD 5
VSS 6
PB1/XT1/SCKA/AN9 7
PB0/XT2/SCSA/CTCK1/AN8 8
PLCD 9
VMAX 10
V1 11
PB4/V2 12
PB5/C1 13
PB6/C2 14
COM0 15
COM1 16
64 PA0/AN5/ICPDA/OCDSDA
63 PA2/AN4/ICPCK/OCDSCK
62 PA5/SCS/ETPB/ETPIB/AN3
61 PA4/SCK/SCL/ETPA/ETPIA/AN2/PINT
60 PA3/SDI/SDA/CTCK0/AN1/VREF
59 PA1/SDO/CTP0/AN0
58 PC0/SEG0/C1P/RES
57 PC1/SEG1/STCK2/C1N/PINT
56 PC2/SEG2/STP2/STP2I/C1X
55 PC3/SEG3/STCK1/C0P/PCK
54 PC4/SEG4/SDOA/STP1/STP1I
53 PC5/SEG5/SDIA/CTCK1/C0X
52 PC6/SEG6/SCKA/CTP1
51 PC7/SEG7/SCSA/STCK0/C0N
50 PD0/SEG8/SCS/STP0/STP0I
49 PD4/SEG12/INT3/CTCK0/ETPA/ETPIA/C1X
48 PE0/SEG16/SDOA/STCK1
47 PE1/SEG17/SDIA/STP1/STP1I
46 PE2/SEG18
45 PE3/SEG19
44 PE4/SEG20
43 PE5/SEG21
42 PE6/SEG22
41 PE7/SEG23
40 PF0/SEG24
39 PF1/SEG25
38 PF2/SEG26
37 PF3/SEG27
36 PF4/SEG28
35 PF5/SEG29
34 PF6/SEG30
33 PF7/SEG31
HT67F60A/HT67F70A
HT67V60A/HT67V70A
64 LQFP-A
17 COM2
18 COM3
19 SEG45
20 SEG44
21 SEG43
22 SEG42
23 SEG41
24 SEG40
25 SEG39
26 SEG38
27 SEG37
28 SEG36
29 SEG35
30 SEG34
31 SEG33
32 SEG32



注：1. 针对 48-pin LQFP 封装类型，只有 R 型偏压 LCD 驱动器可用。

2. 若共用脚同时有多种输出，引脚共用功能除了由配置选项决定外，还可由相关的软件控制位决定。

3. OCDSDA 和 OCDSCK 引脚专门用于 OCDS 功能引脚，仅适用于 HT67Vx0A 系列单片机。HT67Vx0A 是 HT67Fx0A 系列单片机的 EV 芯片。支持“片上调试”功能，在开发过程中，通过 OCDSDA 和 OCDSCK 连接至 Holtek HT-IDE 开发工具进行调试。

引脚说明

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，如模数转换器，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 datasheet 其它章节。

引脚名称	功能	OPT	I/T	O/T	说明
PA0/AN5/ ICPDA/OCDSDA	PA0	PAPU PAWU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	AN5	PAS0	AN	—	A/D 转换器模拟输入
	ICPDA	—	ST	CMOS	ICP 数据 / 地址
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址，仅用于 EV 芯片
PA1/SDO/CTP0/ AN0	PA1	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SDO	PAS0	—	CMOS	SPI 数据输出
	CTP0	PAS0	—	CMOS	CTM0 输出
	AN0	PAS0	AN	—	A/D 转换器模拟输入
PA2/AN4/ICPCK/ OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	AN4	PAS0	AN	—	A/D 转换器模拟输入
	ICPCK	—	ST	CMOS	ICP 时钟引脚
	OCDSCK	—	ST	—	OCDS 时钟引脚，仅用于 EV 芯片
PA3/SDI/SDA/ CTCK0/AN1/ VREF	PA3	PAPU PAWU PAS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SDI	PAS0 IFS3	ST	—	SPI 数据输入
	SDA	PAS0 IFS3	ST	NMOS	I^2C 数据线
	CTCK0	PAS0 IFS1	ST	—	CTM0 输入
	AN1	PAS0	AN	—	A/D 转换器模拟输入
	VREF	PAS0	AN	—	A/D 转换器参考输入

引脚名称	功能	OPT	I/T	O/T	说明
PA4/SCK/SCL/ETPA/ETPIA/AN2/$\overline{\text{PINT}}$	PA4	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	SCK	PAS1 IFS3	ST	CMOS	SPI 串行时钟
	SCL	PAS1 IFS3	ST	NMOS	I^2C 时钟线
	ETPA	PAS1	—	CMOS	ETM A 输出
	ETPIA	PAS1 IFS1	ST	—	ETM A 输入
	AN2	PAS1	AN	—	A/D 转换器模拟输入
	$\overline{\text{PINT}}$	PAS1 IFS0	ST	—	外围中断
PA5/$\overline{\text{SCS}}$/ETPB/ETPIB/AN3	PA5	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	$\overline{\text{SCS}}$	PAS1 IFS3	ST	CMOS	SPI 从机选择
	ETPB	PAS1	—	CMOS	ETM B 输出
	ETPIB	PAS1 IFS1	ST	—	ETM B 输入
	AN3	PAS1	AN	—	A/D 转换器模拟输入
PA6/INT0/ETCK/AN6	PA6	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	INT0	PAS1 INTEG INTC0 IFS0	ST	—	外部中断 0
	ETCK	PAS1 IFS1	ST	—	ETM 输入
	AN6	PAS1	AN	—	A/D 转换器模拟输入
PA7/INT1/STP0/STP0I/AN7	PA7	PAPU PAWU PAS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻和唤醒功能
	INT1	PAS1 INTEG INTC0 IFS0	ST	—	外部中断 1
	STP0	PAS1	—	CMOS	STM0 输出
	STP0I	PAS1 IFS1	ST	—	STM0 输入
	AN7	PAS1	AN	—	A/D 转换器模拟输入

引脚名称	功能	OPT	I/T	O/T	说明
PB0/XT2/$\overline{\text{SCSA}}$/CTCK1/AN8	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	XT2	CO	—	LXT	LXT 振荡器引脚
	$\overline{\text{SCSA}}$	PBS0 IFS3	ST	CMOS	SPIA 从机选择
	CTCK1	PBS0 IFS2	ST	—	CTM1 输入
	AN8	PBS0	AN	—	A/D 转换器模拟输入
PB1/XT1/SCKA/AN9	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	XT1	CO	LXT	—	LXT 振荡器引脚
	SCKA	PBS0 IFS3	ST	CMOS	SPIA 串行时钟
	AN9	PBS0	AN	—	A/D 转换器模拟输入
PB2/OSC1/INT2/STCK0/AN10	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OSC1	CO	HXT	—	HXT 振荡器引脚
	INT2	PBS0 INTEG INTC3 IFS0	ST	—	外部中断 2
	STCK0	PBS0 IFS1	ST	—	STM0 输入
	AN10	PBS0	AN	—	A/D 转换器模拟输入
PB3/OSC2/INT3/CTP1/AN11	PB3	PBPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OSC2	CO	—	HXT	HXT 振荡器引脚
	INT3	PBS0 INTEG INTC3 IFS0	ST	—	外部中断 3
	CTP1	PBS0	—	CMOS	CTM1 输出
	AN11	PBS0	AN	—	A/D 转换器模拟输入
PB4/V2	PB4	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	V2	PBS1	—	—	LCD 电压泵
PB5/C1	PB5	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	C1	PBS1	—	—	LCD 电压泵
PB6/C2	PB6	PBPU PBS3	ST	CMOS	通用 I/O 口，可通过寄存器设置唤醒功能
	C2	PBS1	—	—	LCD 电压泵

引脚名称	功能	OPT	I/T	O/T	说明
PC0/SEG0/C1P/$\overline{RES}$	PC0	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG0	PCS0	—	LCD	LCD SEG 输出
	C1P	PCS0 IFS4	AN	—	比较器 1 正相输入
	$\overline{RES}$	RSTC	ST	—	复位引脚
PC1/SEG1/ $\overline{STCK2}$/C1N/ $\overline{PINT}$	PC1	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG1	PCS0	—	LCD	LCD SEG 输出
	$\overline{STCK2}$	PCS0 IFS2	ST	—	STM2 输入
	C1N	PCS0 IFS4	AN	—	比较器 1 反相输入
	$\overline{PINT}$	PCS0 IFS0	ST	—	外围中断输入
PC2/SEG2/STP2/ STP2I/C1X	PC2	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG2	PCS0	—	LCD	LCD SEG 输出
	STP2	PCS0	—	CMOS	STM2 输出
	STP2I	PCS0 IFS2	ST	—	STM2 输入
	C1X	PCS0	—	CMOS	比较器 1 输出
PC3/SEG3/ STCK1/C0P/PCK	PC3	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG3	PCS0	—	LCD	LCD SEG 输出
	STCK1	PCS0 IFS2	ST	—	STM1 输入
	C0P	PCS0 IFS4	AN	—	比较器 0 正相输入
	PCK	PCS0	—	CMOS	外围时钟输出
PC4/SEG4/ SDOA/STP1/ STP1I	PC4	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG4	PCS1	—	LCD	LCD SEG 输出
	SDOA	PCS1	ST	CMOS	SPIA 串行数据输出
	STP1	PCS1	—	CMOS	STM1 输出
	STP1I	PCS1 IFS2	ST	—	STM1 输入

引脚名称	功能	OPT	I/T	O/T	说明
PC5/SEG5/SDIA/CTCK1/C0X	PC5	PCPU PCS2	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG5	PCS1	—	LCD	LCD SEG 输出
	SDIA	PCS1 IFS3	ST	CMOS	SDIA 串行数据输入
	CTCK1	PCS1 IFS2	ST	—	CTM1 输入
	C0X	PCS1	—	CMOS	比较器 0 输出
PC6/SEG6/SCKA/CTP1	PC6	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG6	PCS1	—	LCD	LCD SEG 输出
	SCKA	PCS1 IFS3	ST	CMOS	SPIA 串行输出
	CTP1	PCS1	—	CMOS	CTM1 输出
PC7/SEG7/$\overline{\text{SCSA}}$/STCK0/C0N	PC7	PCPU PCS3	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG7	PCS1	—	LCD	LCD SEG 输出
	$\overline{\text{SCSA}}$	PCS1 IFS3	ST	CMOS	SPIA 从机选择
	STCK0	PCS1 IFS1	ST	—	STM0 输入
	C0N	PCS1 IFS4	AN	—	比较器 0 反相输入
PD0/SEG8/$\overline{\text{SCS}}$/STP0/STP0I	PD0	PDPU PDS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG8	PDS0	—	LCD	LCD SEG 输出
	$\overline{\text{SCS}}$	PDS0 IFS3	ST	CMOS	SPI 从机选择
	STP0	PDS0	—	CMOS	STM0 反相输出
	STP0I	PDS0 IFS1	ST	—	STM0 输入
PD1/SEG9/SCK/SCL/ETCK/C1P	PD1	PDPU PDS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG9	PDS0	—	LCD	LCD SEG 输出
	SCK	PDS0 IFS3	ST	CMOS	SPI 串行时钟
	SCL	PDS0 IFS3	ST	NMOS	I^2C 时钟线
	ETCK	PDS0 IFS1	ST	—	ETM 输入
	C1P	PDS0 IFS4	AN	—	比较器 1 正相输入

引脚名称	功能	OPT	I/T	O/T	说明
PD2/SEG10/SDI/SDA/ETPB/ETPIB	PD2	PDPU PDS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG10	PDS0	—	LCD	LCD SEG 输出
	SDI	PDS0 IFS3	ST	—	SPI 数据输入
	SDA	PDS0 IFS3	ST	NMOS	I^2C 数据线
	ETPB	PDS0	—	CMOS	ETM B 输出
	ETPIB	PDS0 IFS1	ST	—	ETM B 输入
PD3/SEG11/SDO/C1N	PD3	PDPU PDS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG11	PDS0	—	LCD	LCD SEG 输出
	SDO	PDS0	—	CMOS	SPI 数据输出
	C1N	PDS0 IFS4	AN	—	比较器 1 反相输入
PD4/SEG12/INT3/CTCK0/ETPA/ETPIA/C1X	PD4	PDPU PDS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG12	PDS1	—	LCD	LCD SEG 输出
	INT3	PDS1 INTEG INTC3 IFS0	ST	—	外部中断 3
	CTCK0	PDS1 IFS1	ST	—	CTM0 输入
	ETPA	PDS1	—	CMOS	ETM A 输出
	ETPIA	PDS1 IFS1	ST	—	ETM A 输入
	C1X	PDS1	—	CMOS	比较器 1 输出
PD5/SEG13/INT2/CTP0/C0P	PD5	PDPU PD1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG13	PDS1	—	LCD	LCD SEG 输出
	INT2	PDS1 INTEG INTC3 IFS0	ST	—	外部中断 2
	CTP0	PDS1	—	CMOS	CTM0 输出
	C0P	PDS1 IFS4	AN	—	比较器 0 正相输入

引脚名称	功能	OPT	I/T	O/T	说明
PD6/SEG14/INT1/STCK2/C0N	PD6	PDPU PDS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG14	PDS1	—	LCD	LCD SEG 输出
	INT2	PDS1 INTEG INTC3 IFS0	ST	—	外部中断 1
	STCK2	PDS1 IFS2	ST	—	STM2 输入
	C0N	PDS1 IFS4	AN	—	比较器 0 反相输入
PD7/SEG15/INT0/STP2/STP2I/C0X	PD7	PDPU PDS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG15	PDS1	—	LCD	LCD SEG 输出
	INT0	PDS1 INTEG INTC0 IFS0	ST	—	外部中断 0
	STP2	PDS1	—	CMOS	STM2 输出
	STP2I	PDS1 IFS2	ST	—	STM2 输入
	C0X	PDS1	—	CMOS	比较器 0 输出
PE0/SEG16/SDOA/STCK1	PE0	PEPU PES0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG16	PES0	—	LCD	LCD SEG 输出
	SDOA	PES0	ST	CMOS	SPIA 串行数据输出
	STCK1	PES0 IFS2	ST	—	STM1 输入
PE1/SEG17/SDIA/STP1/STP1I	PE1	PEPU PES0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG17	PES0	—	LCD	LCD SEG 输出
	SDIA	IFS3	ST	CMOS	SPI 串行数据输入
	STP1	PES0	—	CMOS	STM1 输出
	STP1I	PES0 IFS2	ST	—	STM1 输入
PE2/SEG18	PE2	PEPU PES0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG18	PES0	—	LCD	LCD SEG 输出
PE3/SEG19	PE3	PEPU PES0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG19	PES0	—	LCD	LCD SEG 输出
PE4/SEG20	PE4	PEPU PES1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG20	PES1	—	LCD	LCD SEG 输出

引脚名称	功能	OPT	I/T	O/T	说明
PE5/SEG21	PE5	PEPU PES1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG21	PES1	—	LCD	LCD SEG 输出
PE6/SEG22	PE6	PEPU PES1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG22	PES1	—	LCD	LCD SEG 输出
PE7/SEG23	PE7	PEPU PES1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG23	PES1	—	LCD	LCD SEG 输出
PF0/SEG24	PF0	PFPU PFS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG24	PFS0	—	LCD	LCD SEG 输出
PF1/SEG25	PF1	PFPU PFS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG25	PFS0	—	LCD	LCD SEG 输出
PF2/SEG26	PF2	PFPU PFS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG26	PFS0	—	LCD	LCD SEG 输出
PF3/SEG27	PF3	PFPU PFS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG27	PFS0	—	LCD	LCD SEG 输出
PF4/SEG28	PF4	PFPU PFS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG28	PFS1	—	LCD	LCD SEG 输出
PF5/SEG29	PF5	PFPU PFS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG29	PFS1	—	LCD	LCD SEG 输出
PF6/SEG30	PF6	PFPU PFS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG30	PFS1	—	LCD	LCD SEG 输出
PF7/SEG31	PF7	PFPU PFS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG31	PFS1	—	LCD	LCD SEG 输出
SEG32~SEG55	SEGn	—	—	LCD	LCD SEG 输出
COM0~COM3	COMn	—	—	LCD	LCD COM 输出
V1	V1	—	—	—	LCD 电压泵
VMAX	VMAX	—	PWR	—	IC 最大电压，连接至 VDD、PLCD 或 V1

引脚名称	功能	OPT	I/T	O/T	说明
PLCD	PLCD	—	PWR	—	LCD 电源
VDD	VDD	—	PWR	—	正电源
VSS	VSS	—	PWR	—	负电源、接地

注：I/T：输入类型；　　O/T：输出类型
OPT：通过配置选项 (CO) 或寄存器选项来设置
PWR：电源；　　CO：配置选项；
ST：斯密特触发输入
AN：模拟输入
CMOS：CMOS 输出；　　NMOS：NMOS 输出
LCD：LCD SEG/COM 输出
HXT：高速晶体振荡器
LXT：低速晶体振荡器

极限参数

电源供应电压 ..... V_{SS}-0.3V~V_{SS}+6.0V
端口输入电压 ..... V_{SS}-0.3V~V_{DD}+0.3V
储存温度 ..... -50˚C~125˚C
工作温度 ..... -40˚C~85˚C
I_{OL} 总电流 80mA
I_{OH} 总电流 ..... -80mA
总功耗 500mW

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

直流电气特性

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 (HXT)	—	$f_{SYS}=f_{HXT}$=8MHz	2.2	—	5.5	V
			$f_{SYS}=f_{HXT}$=12MHz	2.7	—	5.5	V
			$f_{SYS}=f_{HXT}$=20MHz	4.5	—	5.5	V
	工作电压 (HIRC)	—	$f_{SYS}=f_{HIRC}$=4MHz	2.2	—	5.5	V
			$f_{SYS}=f_{HIRC}$=8MHz	2.2	—	5.5	V
			$f_{SYS}=f_{HIRC}$=12MHz	2.7	—	5.5	V
I_{DD}	工作电流 (HXT)	3V	$f_{SYS}=f_{HXT}$=8MHz 无负载，所有外设关闭	—	1.2	2.0	mA
		5V		—	2.8	4.5	mA
		3V	$f_{SYS}=f_{HXT}$=12MHz 无负载，所有外设关闭	—	1.8	3.0	mA
		5V		—	4.0	6.0	mA
		5V	$f_{SYS}=f_{HXT}$=20MHz 无负载，所有外设关闭	—	5.5	8.5	mA
		3V	$f_{SYS}=f_{HXT}/2$，f_{HXT}=12MHz, 无负载，所有外设关闭	—	0.9	1.5	mA
		5V		—	2.1	3.3	mA
		3V	$f_{SYS}=f_{HXT}/4$，f_{HXT}=12MHz, 无负载，所有外设关闭	—	0.6	1.0	mA
		5V		—	1.6	2.5	mA
		3V	$f_{SYS}=f_{HXT}/8$，f_{HXT}=12MHz, 无负载，所有外设关闭	—	0.48	0.8	mA
		5V		—	1.2	2.0	mA
		3V	$f_{SYS}=f_{HXT}/16$，f_{HXT}=12MHz, 无负载，所有外设关闭	—	0.42	0.7	mA
		5V		—	1.1	1.7	mA
		3V	$f_{SYS}=f_{HXT}/32$，f_{HXT}=12MHz, 无负载，所有外设关闭	—	0.38	0.6	mA
		5V		—	1.0	1.5	mA
		3V	$f_{SYS}=f_{HXT}/64$，f_{HXT}=12MHz, 无负载，所有外设关闭	—	0.36	0.55	mA
		5V		—	1.0	1.5	mA
	工作电流 (HIRC)	3V	$f_{SYS}=f_{HIRC}$=4MHz, 无负载，所有外设关闭	—	0.7	1.2	mA
		5V		—	1.5	2.5	mA
		3V	$f_{SYS}=f_{HIRC}$=8MHz, 无负载，所有外设关闭	—	1.2	2.0	mA
		5V		—	2.8	4.5	mA
		3V	$f_{SYS}=f_{HIRC}$=12MHz, 无负载，所有外设关闭	—	1.5	3.0	mA
		5V		—	3.0	6.0	mA
	工作电流 (LXT)	3V	$f_{SYS}=f_{SUB}=f_{LXT}$=32.768kHz， LXTLP=0 无负载，所有外设关闭	—	10	20	μA
		5V		—	30	50	μA
		3V	$f_{SYS}=f_{SUB}=f_{LXT}$=32.768kHz， LXTLP=1 无负载，所有外设关闭	—	10	20	μA
		5V		—	30	50	μA
	工作电流 (LIRC)	3V	$f_{SYS}=f_{SUB}=f_{LIRC}$=32kHz， 无负载，所有外设关闭	—	10	20	μA
		5V		—	30	50	μA

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{STB}	IDLE0 模式静态电流	3V	f_{SYS} off, f_{SUB} on, LXTLP=1 无负载，所有外设关闭	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
	IDLE1 模式静态电流	3V	f_{SYS}=12MHz on, f_{SUB} on, 无负载，所有外设关闭	—	0.6	1.0	mA
		5V		—	1.2	2.0	mA
		3V	f_{SYS}=12MHz/64 on, f_{SUB} on, 无负载，所有外设关闭	—	0.34	0.6	mA
		5V		—	0.85	1.2	mA
		3V	f_{SYS}=f_{LXT}=32.768kHz on, f_{SUB} on, LXTLP=1, 无负载，所有外设关闭	—	1.9	4.0	μA
		5V		—	3.3	7.0	μA
	SLEEP0 模式静态电流	3V	f_{SYS} off, f_{SUB} off, WDT 除能 , 无负载， 所有外设关闭	—	0.1	1.0	μA
		5V		—	0.3	2.0	μA
	SLEEP1 模式静态电流	3V	f_{SYS} off, f_{SUB}=f_{LIRC}=32kHz on, WDT 使能 , 无负载， 所有外设关闭	—	1.3	5.0	μA
		5V		—	2.2	10.0	μA
		3V	f_{SYS} off, f_{SUB}=f_{LXT}=32.768Hz on, WDT 使能 , 无负载， 所有外设关闭，LXTLP=0	—	1.3	5.0	μA
		5V		—	2.2	10.0	μA
		3V	f_{SYS} off, f_{SUB}=f_{LXT}=32.768Hz on, WDT 使能 , 无负载， 所有外设关闭，LXTLP=1	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
V_{IL}	$\overline{RES}$ 脚以外的输入 / 输出口低电平输入电压	5	—	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	V
V_{IH}	$\overline{RES}$ 脚以外的输入 / 输出口高电平输入电压	5	—	3.5	—	5	V
		—		$0.8V_{DD}$	—	V_{DD}	V
	$\overline{RES}$ 脚高电平输入电压	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	输入 / 输出口灌电流	3V	V_{OL}=$0.1V_{DD}$	4	8	—	mA
		5V	V_{OL}=$0.1V_{DD}$	10	20	—	mA
I_{OH}	输入 / 输出口源电流	3V	V_{OH}=$0.9V_{DD}$	-2	-4	—	mA
		5V	V_{OH}=$0.9V_{DD}$	-5	-10	—	mA
R_{PH}	输入 / 输出口上拉电阻	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

交流电气特性

Ta=25˚C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{SYS}	系统时钟 (HXT)	2.2V~5.5V	—	—	8	—	MHz
		2.7V~5.5V		—	12	—	MHz
		4.5V~5.5V		—	20	—	MHz
	系统时钟 (HIRC)	2.2V~5.5V	—	—	4	—	MHz
		2.2V~5.5V		—	8	—	MHz
		2.7V~5.5V		—	12	—	MHz
	系统时钟 (LXT)	2.2V~5.5V	—	—	32.768	—	kHz
	系统时钟 (LIRC)	2.2V~5.5V	—	—	32	—	kHz
f_{HXT}	高速晶体振荡器时钟 (HXT)	2.2V~5.5V	—	0.4	—	8	MHz
		2.7V~5.5V		0.4	—	12	MHz
		4.5V ~5.5V		0.4	—	20	MHz
f_{HIRC}	高速 RC 振荡器时钟 (HIRC)	5V	Ta = 25˚C	-2%	4	+2%	MHz
				-2%	8	+2%	MHz
				-2%	12	+2%	MHz
		3.0V~5.5V	Ta = 0˚C ~85˚C	-10%	4	+10%	MHz
				-10%	4	+10%	MHz
				-10%	4	+10%	MHz
f_{LXT}	低速晶体振荡器时钟 (LXT)	—	—	—	32.768	—	kHz
f_{LIRC}	低速 RC 振荡器时钟 (LIRC)	5V	Ta = 25˚C	-3%	32	+3%	kHz
t_{TCK}	TM TCK 最小输入脉宽	—	—	0.3	—	—	μs
t_{TP}	TM TPI 最小输入脉宽	—	—	0.3	—	—	μs
t_{INT}	外部中断最小输入脉宽	—	—	10	—	—	μs
t_{RES}	外部复位最小低电平脉宽	—	—	10	—	—	μs
t_{SST}	系统启动时间 (从暂停模式中唤醒，f_{SYS} off)	—	$f_{SYS}=f_{HXT}$	1024	—	—	t_{HXT}
			$f_{SYS}=f_{HIRC}$	16	—	—	t_{HIRC}
			$f_{SYS}= f_{LXT}$	1024	—	—	t_{LXT}
			$f_{SYS}=f_{LIRC}$	2	—	—	t_{LIRC}
	系统启动时间 (从暂停模式中唤醒，f_{SYS} on)	—	—	2	—	—	t_{SYS}
	系统启动时间 (在正常模式和低速模式中切换)	—	f_{HXT} off→on	1024	—	—	t_{HXT}
		—	f_{HIRC} off→on	16	—	—	t_{HIRC}
		—	f_{LXT} off→on	1024	—	—	t_{LXT}
		—	f_{LIRC} off→on	2	—	—	t_{LIRC}

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
t_{RSTD}	系统复位延迟时间 (上电复位 , RSTC 软件复位，LVR 硬件复位 , LVRC 软件复位 , WDTC 软件复位)	—	—	25	50	100	ms
	系统复位延迟时间 ($\overline{RES}$ 复位，WDT 溢出复位)	—	—	8.3	16.7	33.3	ms
t_{EERD}	EEPROM 读周期	—	—	—	—	4	t_{SYS}
t_{EEWR}	EEPROM 写周期	—	—	—	2	4	ms

注：t_{SYS} =1/f_{SYS}

A/D 转换器特性

Ta=25°C

符号	参数	测试条件		最小值	典型值	最大值	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	2.7	—	5.5	V
V_{ADI}	输入电压	—	—	0	—	V_{DD}/V_{REF}	V
V_{REF}	参考电压	—	—	2	—	V_{DD}+0.1V	V
DNL	非线性微分误差	3V	V_{REF}=V_{DD}, t_{ADCK}=0.5μs	—	—	+3	LSB
		5V					
INL	非线性积分误差	3V	V_{REF}=V_{DD}, t_{ADCK}=0.5μs	—	—	+4	LSB
		5V					
I_{ADC}	打开 A/D 增加的功耗	3V	无负载，t_{ADCK}=0.5μs	—	1.0	2.0	mA
		5V		—	1.5	3.0	mA
t_{ADCK}	时钟周期	—	—	0.5	—	10	μs
t_{ADC}	转换时间 (包括采样和保持时间)	—	—	—	16	—	t_{ADCK}
t_{ON2ST}	A/D On-to-Start 时间	—	—	4	—	—	μs

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}	低电压复位电压	—	LVR 使能，选择 2.1V	-5%	2.1	+5%	V
			LVR 使能，选择 2.55V		2.55		
			LVR 使能，选择 3.15V		3.15		
			LVR 使能，选择 3.8V		3.8		
V_{LVD}	低电压检测电压	—	LVD 使能，选择 2.0V	-5%	2.0	+5%	V
			LVD 使能，选择 2.2V		2.2		
			LVD 使能，选择 2.4V		2.4		
			LVD 使能，选择 2.7V		2.7		
			LVD 使能，选择 3.0V		3.0		
			LVD 使能，选择 3.3V		3.3		
			LVD 使能，选择 3.6V		3.6		
			LVD 使能，选择 4.0V		4.0		
V_{BG}	Bandgap 参考电压	—	—	-5%	1.04	+5%	V
I_{OP}	LVD/LVR 工作电流	5V	LVD/LVR 使能，VBGEN=0	—	20	25	μA
		5V	LVD/LVR 使能，VBGEN=1	—	180	200	μA
t_{BGS}	V_{BG} 开启稳定时间	—	无负载	—	—	150	μs
t_{LVDS}	LVDO 稳定时间	—	LVD 使能时，VBGEN=0, LVD 除能 →LVR 使能	—	—	15	μA
		—	LVD 除能时，VBGEN=0, LVD 除能 →LVD 使能	—	—	150	μA
t_{LVR}	低电压复位最短时间	—	—	120	240	480	μs
t_{LVD}	低电压中断最短时间	—	—	60	120	240	μs

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_{CMP}	打开比较器增加的功耗	3V	—	—	50	75	μA
		5V	—	—	85	130	μA
V_{OS}	输入失调电压	5V	—	-10	—	+10	mV
V_{HYS}	迟滞宽度	5V	—	20	40	60	mV
V_{CM}	共模电压范围	—	—	0	—	V_{DD}-1.4V	V
A_{OL}	开环增益	—	—	60	80	—	dB
t_{RP}	响应时间	3V	100mV 偏置（注）	—	—	2	μs
		5V					

注：测量方式为：当一只输入脚的输入电压为 $V_{CM}=(V_{DD}-1.4)/2$ 时，另一只输入脚的输入电压从 V_{SS} 到 (V_{CM}+100mV) 或从 V_{DD} 到 (V_{CM}-100mV) 转变。

LCD 电气特性

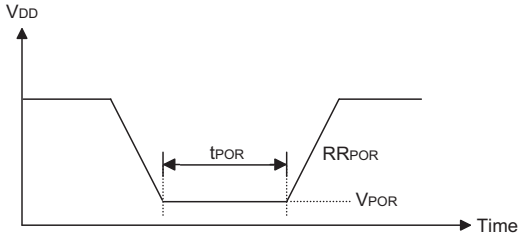
Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{OL}	LCD 引脚灌电流	3V	V_{PLCD} =3V，V_{OL} =0.1V_{PLCD}	210	420	—	mA
		5V	V_{PLCD} =5V，V_{OL} =0.1V_{PLCD}	350	700	—	mA
I_{OH}	LCD 引脚源电流	3V	V_{PLCD} =3V，V_{OL} =0.9V_{PLCD}	-80	-160	—	mA
		5V	V_{PLCD} =5V，V_{OL} =0.9V_{PLCD}	-180	-360	—	mA
I_{LCD}	LCD 工作电流 (R 型)	3V	无负载，1/3 bias，R_T=1170kΩ	—	—	5	μA
		5V		—	—	7.5	μA
		3V	无负载，1/3 bias，R_T=2250kΩ	—	—	23	μA
		5V		—	—	40	μA
		3V	无负载，1/3 bias，R_T=60kΩ	—	—	86	μA
		5V		—	—	145	μA
	LCD 工作电流 (C 型)	3V	无负载，1/3 bias	—	—	1	μA
		5V		—	—	2	μA

上电复位特性

Ta=25°C

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

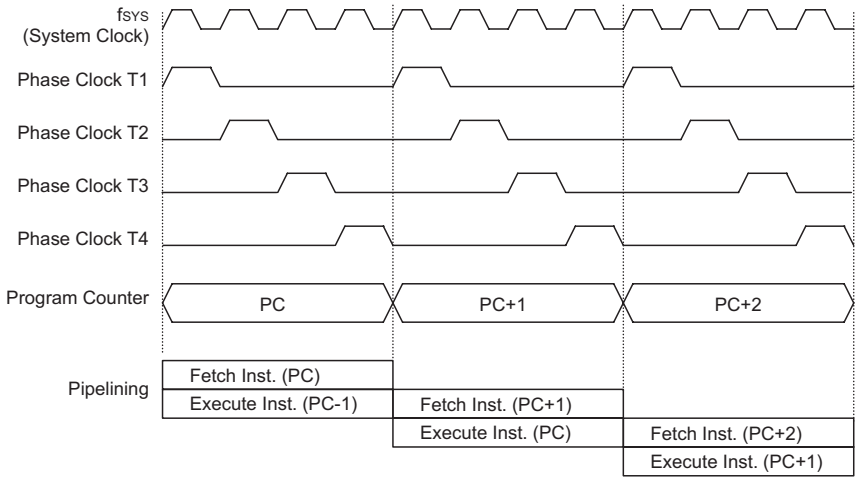


系统结构

内部系统结构是 Holtek 单片机具有良好性能的主要因素。由于采用 RISC 结构，此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外，其它指令都能在一个指令周期内完成。8 位 ALU 参与指令集中所有的运算，它可完成算术运算、逻辑运算、移位、递增、递减和分支等功能，而内部的数据路径则是以通过累加器和 ALU 的方式加以简化。有些寄存器在数据存储器中被实现，且可以直接或间接寻址。简单的寄存器寻址方式和结构特性，确保了在提供具有较大可靠度和灵活性的 I/O 和 A/D 控制系统时，仅需要少数的外部器件。使得这些单片机适用于低成本和批量生产的控制应用。

时序和流水线结构

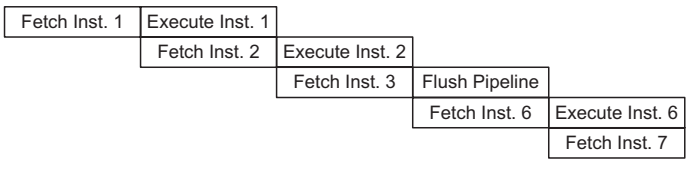
主系统时钟由 HXT、LXT、HIRC 或 LIRC 振荡器提供，它被细分为 T1~T4 四个内部产生的非重叠时序。在 T1 时间，程序计数器自动加一并抓取一条新的指令。剩下的时间 T2~T4 完成译码和执行功能，因此，一个 T1~T4 时钟周期构成一个指令周期。虽然指令的抓取和执行发生在连续的指令周期，但单片机流水线结构会保证指令在一个指令周期内被有效执行。除非程序计数器的内容被改变，如子程序的调用或跳转，在这种情况下指令将需要多一个指令周期的时间去执行。



系统时序和流水线

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

```
1            MOV A,[12H]
2            CALL DELAY
3            CPL [12H]
4            :
5            :
6  DELAY:    NOP
```



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可由程序存储器区指针位选择的，并位于特定程序存储器区中的非连续的程序存储器地址之外，它会在每条指令执行完成以后自动加一。只有较低的 8 位，即所谓的程序计数器低字节寄存器 PCL，可以被用户直接读写。

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

单片机型号	程序计数器	
	程序计数器高字节	PCL 寄存器
HT67F60A	BP0，PC12~PC8	PC7~PC0
HT67F70A	BP1~BP0，PC12~PC8	PC7~PC0

程序计数器

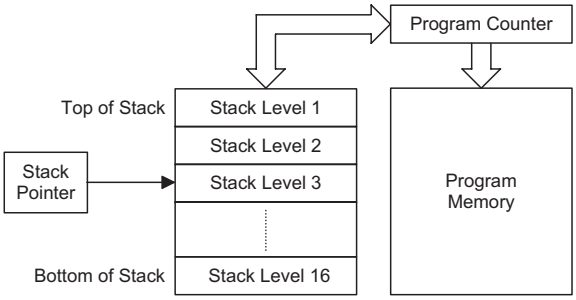
程序计数器的低字节，即程序计数器的低字节寄存器 PCL，可以通过程序控制，且它是可以读取和写入的寄存器。通过直接写入数据到这个寄存器，一个程序短跳转可直接执行，然而只有低字节的操作是有效的，跳转被限制在存储器的当前页中，即 256 个存储器地址范围内，当这样一个程序跳转要执行时，会插入一个空指令周期。PCL 的使用可能引起程序跳转，因此需要额外的指令周期。

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

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

Flash 程序存储器

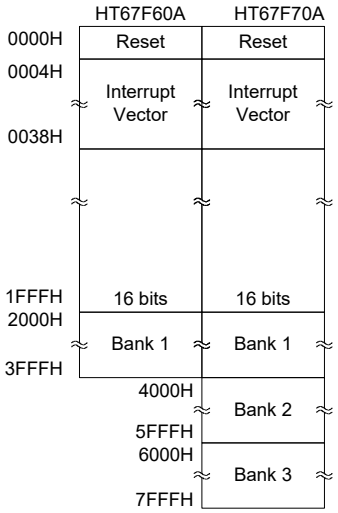
程序存储器用来存放用户代码即储存程序。程序存储器为 FLASH 类型意味着可以多次重复烧录，方便用户使用同一芯片进行程序的修改。使用适当的单片机烧录工具，此系列所有单片机提供用户灵活便利的调试方法和项目开发规划及更新。

结构

程序存储器的容量为 16K×16 位到 32K×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。

单片机型号	容量	Banks
HT66F60A	16K×16	0~1
HT66F70A	32K×16	0~3

该系列单片机程序存储器分为两个或四个 Bank，分别为 Bank 0~Bank 1 或 Bank 0~Bank 3。通过所选单片机 BP 寄存器的 Bit 0 或 Bit 0~1 位选择所需要的 Bank。



程序存储器结构

特殊向量

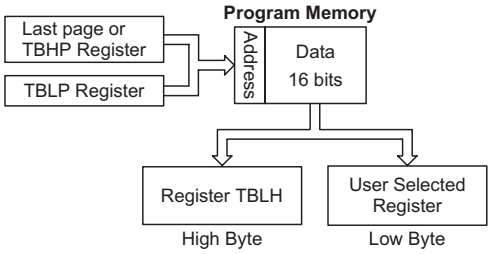
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

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。

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

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”和“rombank”指令储存在存储器的最后一页中。ORG 指令的值“1F00H”位于 ROM 的 Bank 3 中指向的地址是 32K 程序存储器中的具有容量为 8K 的 Bank 3 中最后一页的起始地址。表格指针低字节寄存器设置的初始值为“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“7F06H”，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，则表格指针指向当前页的起始地址。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，可以重新储存，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。

表格读取程序举例

```
rombank 3 code3

ds .section 'data'
Tempreg1 db ?       ; temporary register #1
tempreg2 db ?       ; temporary register #2
:
code0 .section  'code'
mov a,06h           ; initialise table pointer - note that this address
;is referenced
mov tblp,a          ; to the last page or the page that tbhp pointed
mov a,7fh           ; initialise high table pointer
mov tbhp,a          ; it is not necessary to set tbhp if executing tabrdl
:
tabrdc tempreg1
tabrdl tempreg1     ; transfers value in table referred by table pointer to
                    ; tempreg1
                    ; data at program memory address 7F06H transferred to
                    ; tempreg1 and TBLH
dec tblp            ; reduce value of table pointer by one
tabrdc tempreg2
tabrdl tempreg2     ; transfers value in table referenced by table pointer
                    ; to tempreg2
                    ; data at program memory address 7F05H transferred to
                    ; tempreg2 and TBLH
                    ; In this example the data "1AH" is transferred to
                    ; tempreg1 and data "0FH" to tempreg2
                    ; the value "00H" will be transferred to the high byte
                    ; register TBLH
:
code3 .section  'code'
org 1F00h           ; sets initial address of lastpage
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
:
```

在线烧录 – ICP

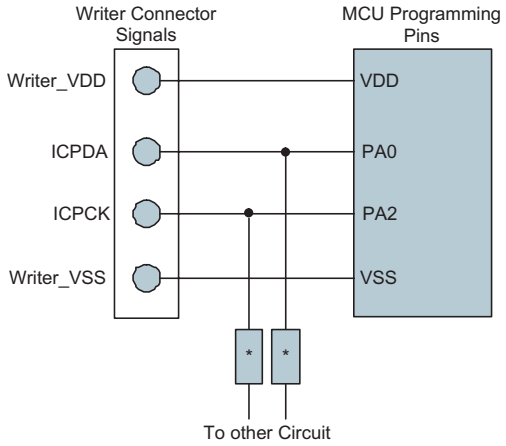
Flash 型 MCU 便于用户对同一芯片进行程序的更新和修改。

另外，Holtek 单片机提供 4 线接口的在线烧录方式。用户可将进行过编程或未经过编程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
ICPCK	PA2	串行时钟
VDD	VDD	电源
VSS	VSS	地

芯片内部程序存储器可以通过 4 线的接口在线进行烧录。其中一个单独的引脚用于数据串行下载或上传、一条用于串行时钟、两条用于提供电源。芯片在线烧录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，用户必须控制 ICPDA 和 ICPCK 脚进行数据和时钟烧录，以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 HT67Vx0A 用于 HT67Fx0A 系列单片机仿真。此 EV 芯片提供片上调试功能 (OCDS) 用于开发过程中的单片机调试。除了片上调试功能和封装类型，HT67Vx0A 和 HT67Fx0A 在功能上几乎是兼容的。用户可将 OCDSDA 和 OCDSCK 引脚连接至 Holtek HT-IDE 开发工具，从而实现单片机的仿真。OCDSDA 引脚为 OCDS 数据 / 地址输入 / 输出脚，OCDSCK 引脚为 OCDS 时钟输入脚。当用户用 EV 芯片进行调试时，OCDSDA 和 OCDSCK 引脚上的其它共用功能对 EV 芯片无效。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，参考“Holtek e-Link for 8-bit MCU OCDS User's Guide”文档。

Holtek e-Link 引脚	EV 芯片 OCDS 引脚	引脚描述
OCDSDA	OCDSDA	片上调试数据 / 地址输入 / 输出
OCDSCK	OCDSCK	片上调试时钟输入
VDD	VDD	电源
GND	VSS	地

在应用编程 – IAP

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

- 擦除页：64 个字 / 页
- 写：64 个字 / 次
- 读：1 个字 / 次

在应用编程控制寄存器

地址寄存器 FARL/FARH 和数据寄存器 FD0L/FD0H、FD1L/FD1H、FD2L/FD2H 和 FD3L/FD3H，以及位于数据存储器 Sector 1 的控制寄存器 FC0、FC1 和 FC2，都是与 IAP 相关的 Flash 存取寄存器。如果用间接寻址存取 FC0、FC1 和 FC2 寄存器，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 和 IAR2，和一对存储器指针 MP1L/MP1H 或 MP2L/MP2H 来执行。由于 FC0、FC1 和 FC2 控制寄存器位于地址 43H~45H 数据存储器 Sector 1 中，位于 43H 到 45H 地址范围的值必须首先被写入 MP1L 或 MP2L 存储器指针低字节，且“01”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 (HT67F60A)	—	—	A13	A12	A11	A10	A9	A8
FARH (HT67F70A)	—	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

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	1	1	1	0	0	0	0

Bit 7 **CFWEN**：Flash 存储器写使能控制位
0：Flash 存储器写功能除能
1：Flash 存储器写功能已被成功使能
当此位由应用程序清零后，Flash 存储器写功能除能。注意，此位写“1”则会导致无效操作。此位被用来说明 Flash 存储器写功能状态。当此位由硬件置为“1”时，就意味着 Flash 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 存储器写功能使能模式
111：保留位

Bit 3 **FWPEN**：Flash 存储器写步骤使能控制
0：除能
1：使能
此位由应用程序置位，由硬件清零。当此位置为“1”且 FMOD 字段设为“110”时，IAP 控制器将执行“Flash 存储器写功能使能”步骤。一旦 Flash 存储器写功能成功使能，无需再设置 FWPEN 位。

Bit 2 **FWT**：Flash 存储器写初始化控制位
0：不初始化 Flash 存储器写或 Flash 存储器写过程已完成
1：初始化 Flash 存储器写过程
当 Flash 存储器写过程完成，此位由软件置“1”，由硬件清零。

Bit 1 **FRDEN**：Flash 存储器读控制位
0：Flash 存储器读除能
1：Flash 存储器读使能

Bit 0 **FRD**：Flash 存储器读初始化控制位
0：不初始化 Flash 存储器读或 Flash 存储器读过程已完成
1：初始化 Flash 存储器读过程
当 Flash 存储器读过程完成，此位由软件置“1”，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	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：不初始化写缓冲区清除或写缓冲清除过程已完成
1：初始化写缓冲区清除过程
当写缓冲区清除过程完成，此位由软件置“1”，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 存储器地址 bit 7~bit 0

FARH 寄存器 – HT67F60A

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 0

FARH 寄存器 – HT67F70A

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 0

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 存储器数据 bit 7~bit 0

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 存储器数据 bit 15~bit 8

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 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 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 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 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 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 存储器数据 bit 15~bit 8

Flash 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 存储器数据，用户必须首先使能 Flash 存储器写操作，步骤如下：

1、写“110”到 FMOD2~FMOD0 位，选择 FWEN 模式。

2、FWPEN 置为“1”。步骤 1 和步骤 2 可同时执行。

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。

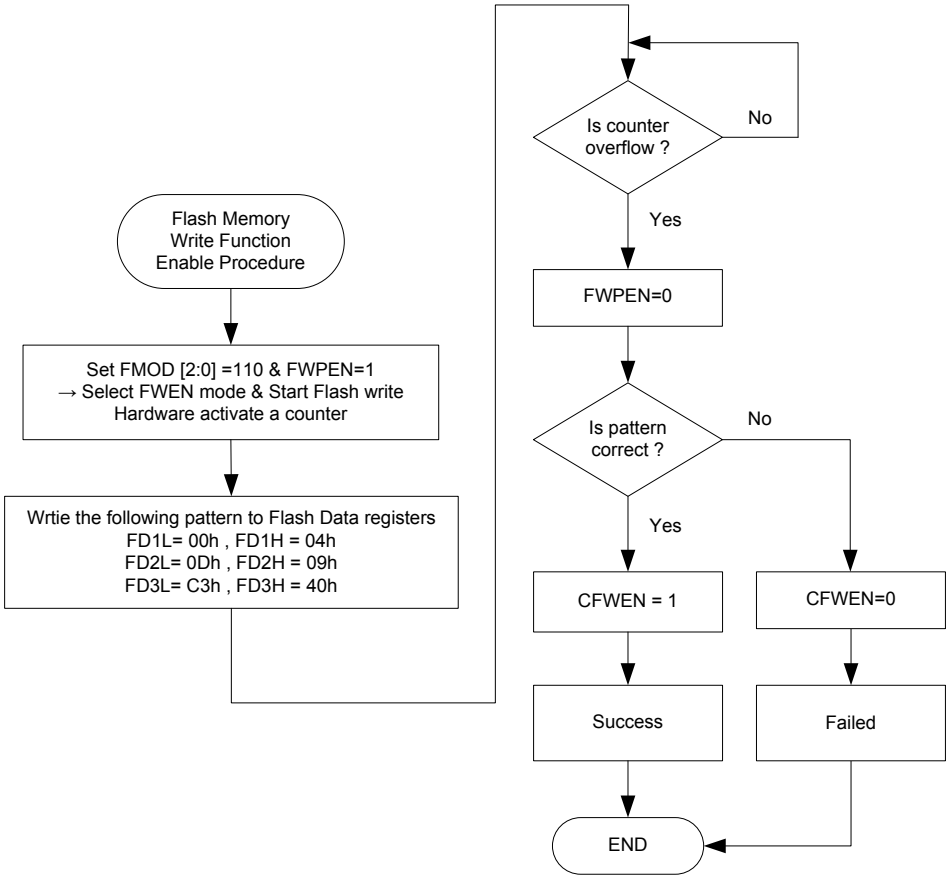
4、溢出周期为 300μs 的计数器将进行有效计时，此时允许用户将正确的数据序列写入 FD1L/FD1H~FD3L/FD3H 寄存器对。计数器时钟来自 LIRC 振荡器。

5、如果计数器溢出，模式检测后 FWPEN 位将自动由硬件清零。

6、如果数据正确，CFWEN 位由硬件置为“1”，表明 Flash 存储器写操作成功使能。

7、一旦 Flash 存储器写操作使能，用户可通过 Flash 控制寄存器更改 Flash ROM 数据。

8、用户可以清零 CFWEN 位来除能 Flash 存储器写操作。



Flash 存储器写功能使能步骤

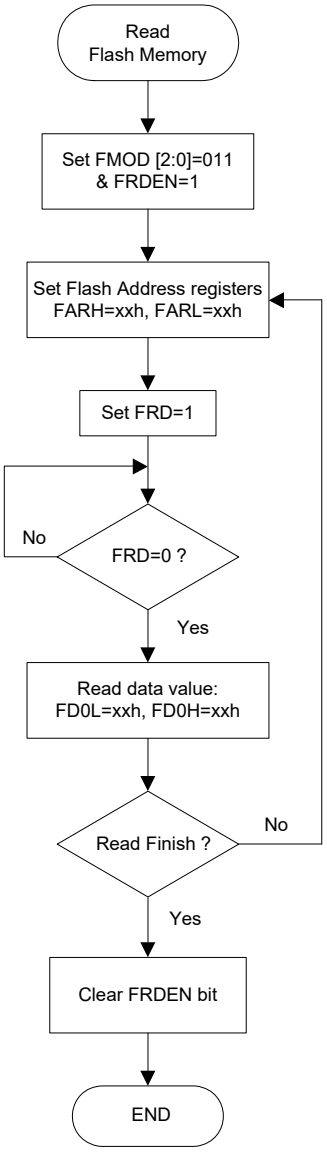
Flash 存储器写步骤

通过前面的 IAP 步骤成功使能 Flash 存储器写功能后，用户必须先擦除相应的 Flash 存储块，然后初始化 Flash 写操作。页擦除操作的数量是每页 64 个字，有效的页擦除地址由 FARH 寄存器和 FARL 寄存器的 bit 7~bit 6 位来指定。

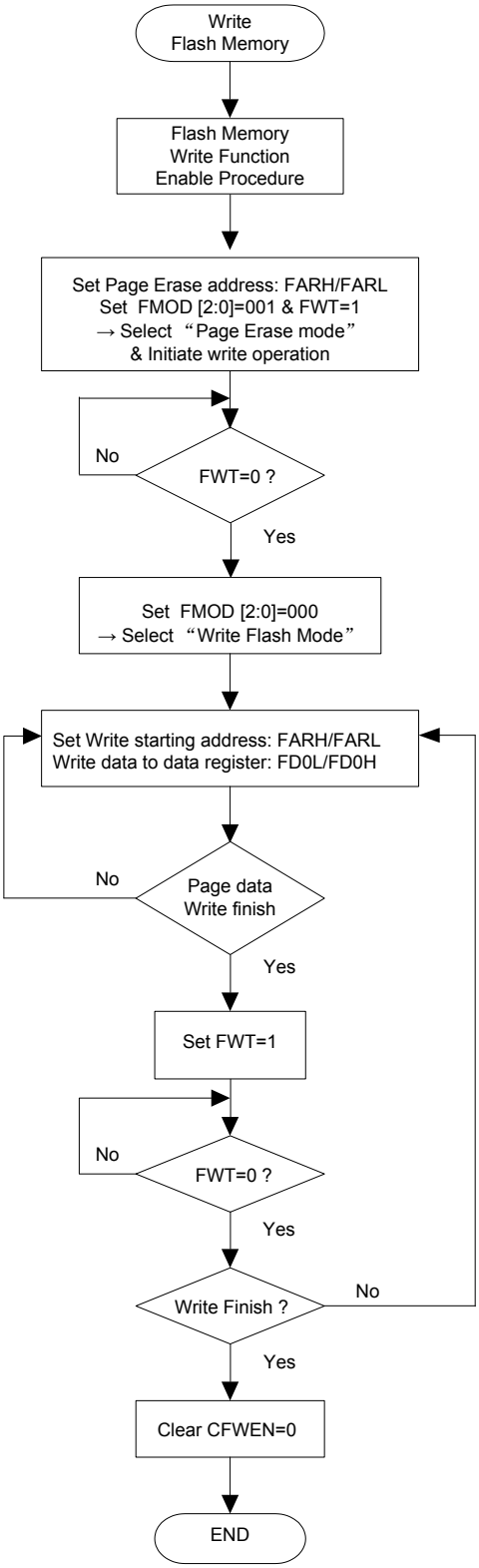
擦除页	FARH	FARL [7:6]	FARL [5:0]
0	0000 0000	00	xxxx
1	0000 0001	01	xxxx
2	0000 0010	10	xxxx
3	0000 0011	11	xxxx
4	0000 0100	00	xxxx
5	0000 0101	01	xxxx
6	0000 0110	10	xxxx
7	0000 0111	11	xxxx
8	0000 1000	00	xxxx
9	0000 1001	01	xxxx
: :	: :	: :	: :
252	0011 1111	00	xxxx
253	0011 1111	01	xxxx
254	0011 1111	10	xxxx
255	0011 1111	11	xxxx
: :	: :	: :	: :
508	0111 1111	00	xxxx
509	0111 1111	01	xxxx
510	0111 1111	10	xxxx
511	0111 1111	11	xxxx

“xxxx”：无关

注：HT67F60A 单片机中有 256 个 IAP 擦除页，HT67F70A 单片机中有 512 个 IAP 擦除页。



读 **Flash** 存储器步骤



写 Flash 存储器步骤

注：当 FWT 或 FRD 位置为“1”，单片机停止工作。

数据存储器

数据存储器是内容可更改的 8 位 RAM 内部存储器，用来储存临时数据。

数据存储器分为三部分，第一部分是特殊功能寄存器，这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二部分是通用数据存储器，所有地址都可在程序的控制下进行读取和写入。第三部分是为 LCD 存储器保留的。这个特殊数据存储器的地址直接映射到 LCD 显示器，写入这部分存储器的数据将直接影响显示的数据。

切换不同的数据存储器 Sector 可通过设置正确的存储器指针值实现。

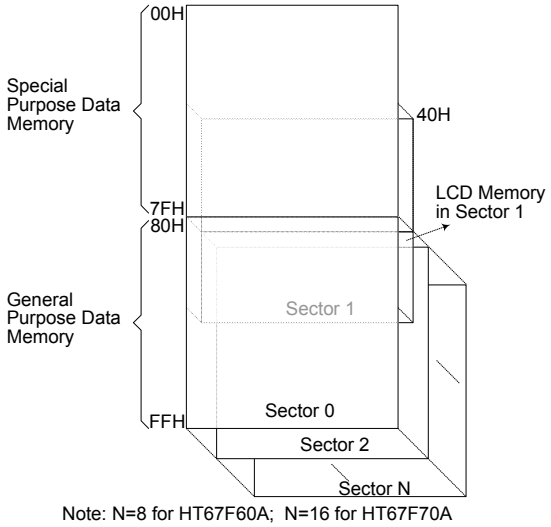
结构

数据存储器被分为几个 Sector，都位于 8 位存储器中。每个数据存储器 Sector 分为两类，特殊功能数据存储器和通用数据存储器。

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

单片机型号	特殊功能数据存储器		LCD 数据存储器		通用数据存储器	
	容量	Sectors	容量	Sectors	容量	Sectors
HT67F60A	192×8	0, 2~8: 00H~7FH 1: 40H~7FH	56×4	1: 80H~B7H	1024×8	0: 80H~FFH 2: 80H~FFH : 8: 80H~FFH
HT67F70A	192×8	0, 2~16: 00H~7FH 1: 40H~7FH	56×4	1: 80H~B7H	2048×8	0: 80H~FFH 2: 80H~FFH : 16: 80H~FFH

数据存储器列表



Note: N=8 for HT67F60A; N=16 for HT67F70A

注：HT67F60A 中，N=8；HT67F70A 中，N=16

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 BP 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”最多是 13 位，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。这个数据存储区可让使用者进行读取和写入的操作。使用位操作指令可对个别的位做置位或复位的操作，较大地方便了用户在数据存储器内进行位操作。

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 0~8
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1L
04H	MP1H
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	BP
0CH	IAR2
0DH	MP2L
0EH	MP2H
0FH	
10H	PAWU
11H	PAPU
12H	PA
13H	PAC
14H	
15H	PBPU
16H	PB
17H	PBC
18H	
19H	PCPU
1AH	PC
1BH	PCC
1CH	
1DH	PDPU
1EH	PD
1FH	PDC
20H	
21H	PEPU
22H	PE
23H	PEC
24H	
25H	PFPU
26H	PF
27H	PFC
28H	
29H	
2AH	LCDC0
2BH	LCDC1
2CH	
2DH	
2EH	
2FH	RSTC
30H	INTC0
31H	INTC1
32H	INTC2
33H	INTC3
34H	MFI0
35H	MFI1
36H	MFI2
37H	MFI3
38H	MFI4
39H	INTEG
3AH	SMOD
3BH	SMOD1
3CH	LVRC
3DH	LVDC
3EH	WDTC
3FH	SMOD2

	Sector 0, 2~8	Sector 1
40H		EEC
41H	EEA	
42H	EED	
43H		FC0
44H		FC1
45H		FC2
46H	CP0C	
47H	CP1C	
48H	ETMC0	IFS0
49H	ETMC1	IFS1
4AH	ETMC2	IFS2
4BH	ETMDL	IFS3
4CH	ETMDH	IFS4
4DH	ETMAL	
4EH	ETMAH	
4FH	ETMBL	
50H	ETMBH	STM1C0
51H	STM0C0	STM1C1
52H	STM0C1	STM1DL
53H	STM0DL	STM1DH
54H	STM0DH	STM1AL
55H	STM0AL	STM1AH
56H	STM0AH	STM1RP
57H	STM0RP	STM2C0
58H	CTM1C0	STM2C1
59H	CTM1C1	STM2DL
5AH	CTM1DL	STM2DH
5BH	CTM1DH	STM2AL
5CH	CTM1AL	STM2AH
5DH	CTM1AH	STM2RP
5EH	CTM0C0	
5FH	CTM0C1	
60H	CTM0DL	PAS0
61H	CTM0DH	PAS1
62H	CTM0AL	PBS0
63H	CTM0AH	PBS1
64H	PSC0	PCS0
65H	TB0C	PCS1
66H	TB1C	PDS0
67H	PSC1	PDS1
68H	SADC0	PES0
69H	SADC1	PES1
6AH	SADOL	PFS0
6BH	SADOH	PFS1
6CH	SIMC0	
6DH	SIMC1	
6EH	SIMD	
6FH	SIMA/SIMC2	
70H	I2CTOC	
71H	SPIAC0	
72H	SPIAC1	
73H	SPIAD	
74H	FARL	
75H	FARH	
76H	FD0L	
77H	FD0H	
78H	FD1L	
79H	FD1H	
7AH	FD2L	
7BH	FD2H	
7CH	FD3L	
7DH	FD3H	
7EH	TBC2	
7FH		

: Unused, read as 00H

HT67F60A 特殊功能数据存储器结构

	Sector 0~16
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1L
04H	MP1H
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	BP
0CH	IAR2
0DH	MP2L
0EH	MP2H
0FH	
10H	PAWU
11H	PAPU
12H	PA
13H	PAC
14H	
15H	PBPU
16H	PB
17H	PBC
18H	
19H	PCPU
1AH	PC
1BH	PCC
1CH	
1DH	PDPU
1EH	PD
1FH	PDC
20H	
21H	PEPU
22H	PE
23H	PEC
24H	
25H	PFPU
26H	PF
27H	PFC
28H	
29H	
2AH	LCDC0
2BH	LCDC1
2CH	
2DH	
2EH	
2FH	RSTC
30H	INTC0
31H	INTC1
32H	INTC2
33H	INTC3
34H	MFI0
35H	MFI1
36H	MFI2
37H	MFI3
38H	MFI4
39H	INTEG
3AH	SMOD
3BH	SMOD1
3CH	LVRC
3DH	LVDC
3EH	WDTC
3FH	SMOD2

	Sector 0, 2~16	Sector 1
40H		EEC
41H	EEA	
42H	EED	
43H		FC0
44H		FC1
45H		FC2
46H	CP0C	
47H	CP1C	
48H	ETMC0	IFS0
49H	ETMC1	IFS1
4AH	ETMC2	IFS2
4BH	ETMDL	IFS3
4CH	ETMDH	IFS4
4DH	ETMAL	
4EH	ETMAH	
4FH	ETMBL	
50H	ETMBH	STM1C0
51H	STM0C0	STM1C1
52H	STM0C1	STM1DL
53H	STM0DL	STM1DH
54H	STM0DH	STM1AL
55H	STM0AL	STM1AH
56H	STM0AH	STM1RP
57H	STM0RP	STM2C0
58H	CTM1C0	STM2C1
59H	CTM1C1	STM2DL
5AH	CTM1DL	STM2DH
5BH	CTM1DH	STM2AL
5CH	CTM1AL	STM2AH
5DH	CTM1AH	STM2RP
5EH	CTM0C0	
5FH	CTM0C1	
60H	CTM0DL	PAS0
61H	CTM0DH	PAS1
62H	CTM0AL	PBS0
63H	CTM0AH	PBS1
64H	PSC0	PCS0
65H	TB0C	PCS1
66H	TB1C	PDS0
67H	PSC1	PDS1
68H	SADC0	PES0
69H	SADC1	PES1
6AH	SADOL	PFS0
6BH	SADOH	PFS1
6CH	SIMC0	
6DH	SIMC1	
6EH	SIMD	
6FH	SIMA/SIMC2	
70H	I2CTOC	
71H	SPIAC0	
72H	SPIAC1	
73H	SPIAD	
74H	FARL	
75H	FARH	
76H	FD0L	
77H	FD0H	
78H	FD1L	
79H	FD1H	
7AH	FD2L	
7BH	FD2H	
7CH	FD3L	
7DH	FD3H	
7EH	TBC2	
7FH		

: Unused, read as 00H

HT67F70A 特殊功能数据存储器结构

特殊功能寄存器

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

间接寻址寄存器 – IAR0, IAR1, IAR2

间接寻址寄存器 IAR0、IAR1 和 IAR2 的地址虽位于 RAM 寄存器区，但其并没有实际的物理地址。间接寻址的方法准许使用间接寻址寄存器和存储器指针做数据操作，以取代定义实际存储器地址的直接存储器寻址方法。在间接寻址寄存器 IAR0、IAR1 和 IAR2 上的任何动作，将对间接寻址指针 MP0、MP1L/MP1H 或 MP2L/MP2H 所指定的存储器地址产生对应的读 / 写操作。它们总是成对出现，IAR0 和 MP0 可以访问 Sector 0，而 IAR1 和 MP1L/MP1H、IAR2 和 MP2L/MP2H 可以访问任何数据存储器 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, MP1H/MP1L, MP2H/MP2L

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器中能像普通的寄存器一般被操作，因此提供了一个寻址和数据追踪的有效方法。当对间接寻址寄存器进行任何操作时，单片机指向的实际地址是由间接寻址指针所指定的地址。MP0，IAR0 用于访问 Sector 0，而 MP1L/MP1H 和 IAR1、MP2L/MP2H 和 IAR2 根据 MP1H 或 MP2H 寄存器可以访问所有的 Sector。直接寻址通过相关的数据存储器寻址指令来访问所有的数据 Sector。

间接寻址程序举例

Example 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:
```

Example 2

```
data .section at 01F0H '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 mp1h,a
mov a,offset adres1        ; Accumulator loaded with first RAM address
mov mp1l,a                 ; setup memory pointer with first RAM address
loop:
clr IAR1                   ; clear the data at address defined by MP1
inc mp1l                   ; increment memory pointer MP1L
sdz block                  ; check if last memory location has been cleared
jmp loop
continue:
    :
```

在上面的例子中有一点值得注意，即并没有确定 RAM 地址。

使用扩展指令直接寻址程序举例

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

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

存储区指针 – BP

程序存储器被分为几个部分，具体数目由所选择的单片机型号决定。可以通过设置存储区指针 (Bank Pointer) 值来访问不同的程序存储区。存储区指针应在单片机使用“JMP”或“CALL”指令执行“分支”操作前正确地配置。在指令执行后会跳转到由程序存储区指针所选 Bank 的一个非连续的程序存储器地址。

单片机型号	位							
	7	6	5	4	3	2	1	0
HT67F60A	—	—	—	—	—	—	—	BP0
HT67F70A	—	—	—	—	—	—	BP1	BP0

BP 寄存器列表

BP 寄存器 – HT67F60A

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	BP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1　未定义，读为“0”

Bit 0　**BP0**：程序存储区选择位
0：Bank 0
1：Bank 1

BP 寄存器 – HT67F70A

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	BP1	BP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未定义，读为“0”

Bit 1~0　**BP1~BP0**：程序存储区选择位
00：Bank 0
01：Bank 1
10：Bank 2
11：Bank 3

累加器 – ACC

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

程序计数器低字节寄存器 – PCL

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

表格寄存器 – TBLP, TBHP, TBLH

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

状态寄存器 – STATUS

这 8 位的状态寄存器由零标志位 (Z)、进位标志位 (C)、辅助进位标志位 (AC)、溢出标志位 (OV)、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 PDF 和 TO 标志外，状态寄存器中的位像其它大部分寄存器一样可以被改变。任何数据写入到状态寄存器将不会改变 TO 或 PDF 标志位。另外，执行不同的指令后，与状态寄存器有关的运算可能会得到不同的结果。TO 标志位只会受系统上电、看门狗溢出或执行“CLR WDT”或“HALT”指令影响。PDF 标志位只会受执行“HALT”或“CLR WDT”指令或系统上电影响。

Z、OV、AC、C、SC 和 CZ 标志位通常反映最近运算的状态。

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

此系列所有单片机的一个特性是内建 EEPROM 数据存储器。“Electrically Erasable Programmable Read Only Memory”为电可擦可编程只读存储器，由于其非易失的存储结构，即使在电源掉电的情况下存储器内的数据仍然保存完好。这种存储区扩展了 ROM 空间，对设计者来说增加了许多新的应用机会。EEPROM 可以用来存储产品编号、校准值、用户特定数据、系统配置参数或其它产品信息等。EEPROM 的数据读取和写入过程也会变的更简单。

单片机型号	容量	地址
HT67F60A	128×8	00H~7FH
HT67F70A		

EEPROM 数据存储器结构

EEPROM 数据存储器容量为 128×8。由于映射方式与程序存储器和数据存储器不同，因此不能像其它类型的存储器一样寻址。使用 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 数据存储器总的操作。地址寄存器 EEA、数据寄存器 EED 及控制寄存器 EEC。EEA 和 EED 位于 Sector 0 中，它们能像其它特殊功能寄存器一样直接被访问。EEC 位于 Sector 1 中，可通过 MP1L/MP1H 或 MP2L/MP2H 存储器指针对和间接寻址寄存器 IAR1 或 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，存储器指针低字节寄存器 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	—	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM 寄存器列表

EEA 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
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　　**EEA6~EEA0**：数据 EEPROM 地址 Bit 6~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 地址 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：激活写周期
此位为数据 EEPROM 写控制位，由应用程序将此位置高将激活写周期。写周期结束后，硬件自动将此位清零。当 WREN 未先置高时，此位置高无效。

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

Bit 0 **RD**：EEPROM 读控制位
0：读周期结束
1：激活读周期
此位为数据 EEPROM 读控制位，由应用程序将此位置高将激活读周期。读周期结束后，硬件自动将此位清零。当 RDEN 未首先置高时，此位置高无效。

注：在同一条指令中 WREN、WR、RDEN 和 RD 不能同时置为“1”。WR 和 RD 不能同时置为“1”。

从 EEPROM 中读取数据

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

写数据到 EEPROM

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

写保护

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

EEPROM 中断

EEPROM 写周期结束后将产生 EEPROM 写中断，需先通过设置相关中断寄存器的 DEE 位使能 EEPROM 中断。由于 EEPROM 中断包含在多功能中断中，相应的多功能中断使能位需被设置。当 EEPROM 写周期结束，DEF 请求标志位及其相关多功能中断请求标志位将被置位。若 EEPROM 和多功能中断使能且堆栈未满的情况下将跳转到相应的多功能中断向量中执行。当中断被响应，只有多功能中断标志位将自动复位，而 EEPROM 中断标志将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。

程序举例

从 EEPROM 中读取数据—轮询法

```
MOV  A, EEPROM_ADRES        ; user defined address
MOV  EEA, A
MOV  A, 040H                ; setup memory pointer 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
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  AR1.2                  ; 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。振荡器选择是通过配置选项和相关的控制寄存器共同完成的。

振荡器概述

振荡器除了作为系统时钟源，还作为看门狗定时器和时基功能的时钟源。外部振荡器需要一些外围器件，而完全集成的内部振荡器不需要任何外围器件。它们提供的高速和低速系统振荡器具有较宽的频率范围。所有振荡器选择通过配置选项和寄存器选择。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。

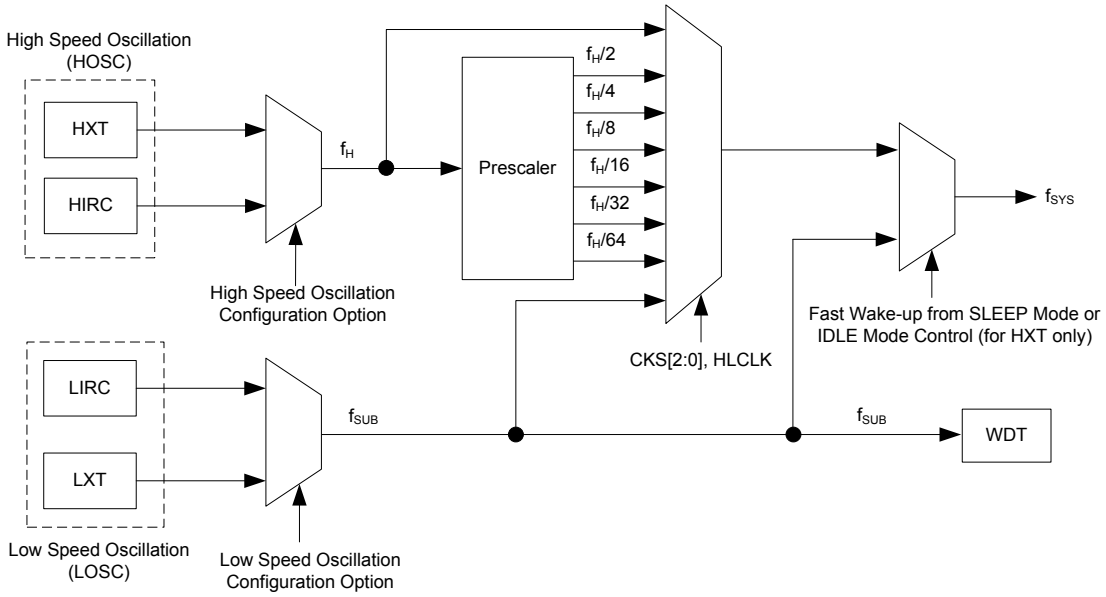
类型	名称	频率	引脚
外部高速晶振	HXT	400kHz~20MHz	OSC1/OSC2
内部高速 RC	HIRC	4/8/12 MHz	—
外部低速晶振	LXT	32.768kHz	XT1/ XT2
内部低速 RC	LIRC	32kHz	—

振荡器类型

系统时钟配置

该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 4/8/12MHz 高速振荡器 HIRC，低速振荡器有内部 32kHz 低速振荡器 LIRC 和外部 32.768kHz 晶体振荡器。使用高速或低速振荡器作为系统时钟的选择是通过设置 SMOD 寄存器中的 HLCLK 位和 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配置选项选择，低速或高速系统时钟频率由 SMOD 寄存器中的 HLCLK 位和 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。

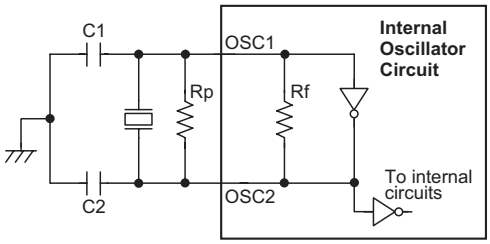


系统时钟配置

外部晶体 / 陶瓷振荡器 – HXT

外部高频晶体 / 陶瓷振荡器可通过配置选项选择。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，而不需其它外部器件。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. Rp 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		
晶体频率	C1	C2
12MHz	0pF	0pF
8 MHz	0pF	0pF
4 MHz	0pF	0pF
1 MHz	100pF	100pF
注：C1 和 C2 数值仅作参考用		

晶体振荡器电容推荐值

内部高速 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，不需其它外部器件。内部 RC 振荡器具有三种固定的频率：4MHz，8MHz，12MHz。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。在电源电压为 5V 及温度为 25°C 的条件下，4MHz，8MHz，12MHz 这三个固定频率的容差为 2%。如果选择了该内部时钟，无需额外的引脚；PB2 和 PB3 可以作为通用 I/O 口使用。

外部 32.768kHz 晶体振荡器 – LXT

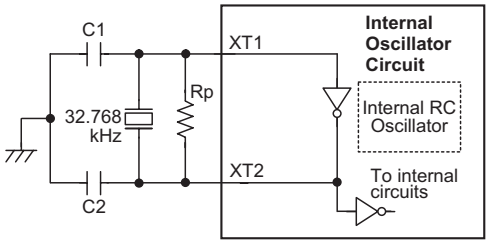
外部 32.768kHz 晶体振荡器是一个低频振荡器，经由配置选项选择。时钟频率固定为 32.768kHz，此时 XT1 和 XT2 间引脚必须连接 32.768kHz 的晶体振荡器。需要外部电阻和电容连接到 32.768kHz 晶振以帮助起振。对于那些要求精确频率的场合中，可能需要这些元件来对由制程产生的误差提供频率补偿。在系统上电期间，LXT 振荡器启动需要一定的延时。

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，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。

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，是必需的。注意，连接 32.768kHz 晶体和 XT1/XT2 之间的线应尽量短，以避免噪声干扰。引脚共用的软件控制位决定 XT1/XT2 脚是用于 LXT 还是作为普通 I/O 口或其它共用功能使用。

- 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的接近单片机。



Note: 1. Rp, C1 and C2 are required.
2. Although not shown pins have aparasitic capacitance of around 7pF.

外部 LXT 振荡器

LXT 振荡器 C1 和 C2 值		
晶体频率	C1	C2
12MHz	0 pF	0 pF
8MHz	0 pF	0 pF
4MHz	0 pF	0 pF
1MHz	100 pF	100 pF
注：C1 和 C2 数值仅作参考用		

LXT 振荡器电容推荐值

LXT 振荡器低功耗功能

LXT 振荡器可以工作在快速启动模式或低功耗模式，可通过设置 SMOD2 寄存器中的 LXTLP 位进行模式选择。

LXTLP	LXT 工作模式
0	快速启动
1	低功耗

- **SMOD2 寄存器**

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

Bit 7~1　未定义，读为“0”

Bit 0　**LXTLP**：LXT 低功耗控制位
0：快速启动模式
1：低功耗模式

系统上电时会清零 LXTLP 位来快速启动 LXT 振荡器。在快速启动模式，LXT 振荡器将起振并快速稳定下来。LXT 振荡器完全起振后，可以通过设置 LXTLP 位为高进入低功耗模式。振荡器可以继续运行，其间耗电将少于快速启动模式。在功耗敏感的应用领域如电池应用方面，功耗必须限制为一个最小值。为了降低功耗，建议系统上电 2 秒后，在应用程序中将 LXTLP 位设为“1”。

应注意的是，无论 LXTLP 位是什么值，LXT 振荡器会正常运作，不同的只是在低功耗模式时启动时间更长。

内部 32kHz 振荡器 – LIRC

内部 32kHz 系统振荡器也是一个低频振荡器，经由配置选项选择。这种单片机有一个完全集成 RC 振荡器，它在 5V 电压下运行的典型频率值为 32kHz 且无需外部元件。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响较大程度地降低。因此，内部 32kHz 振荡器频率在 25°C 温度 5V 电压下的精度保持在 3% 以内。

辅助振荡器

低速振荡器除了提供一个系统时钟源外，也用来为看门狗定时器和时基中断提供时钟来源。

工作模式和系统时钟

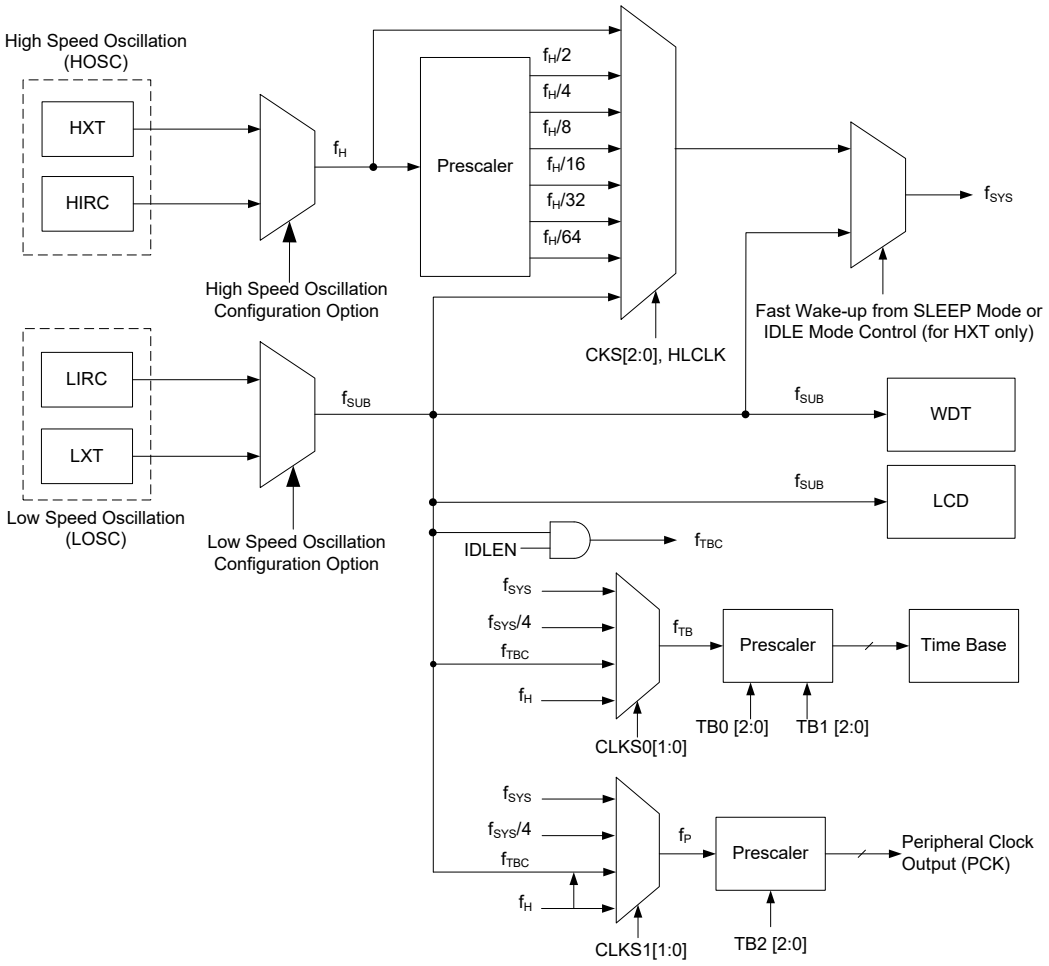
现今的应用要求单片机具有较高的性能及尽可能低的功耗，这种矛盾的要求在便携式电池供电的应用领域尤为明显。高性能所需要的高速时钟将增加功耗，反之亦然。此单片机提供高、低速两种时钟源，它们之间可以动态切换，用户可通过优化单片机操作来获得较佳性能 / 功耗比。

系统时钟

单片机为 CPU 和外围功能操作提供了多种不同的时钟源。用户使用配置选项和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 SMOD 寄存器中的 HLCLK 位及 CKS2~CKS0 位进行选择。高频时钟来自 HXT 或 HIRC 振荡器，可通过配置选项选择，低频系统时钟源来自内部时钟 f_{SUB}，若 f_{SUB} 被选择，可通过配置选项设定为 LXT 或 LIRC 振荡器。其它系统时钟还有高速系统振荡器的分频 $f_H/2$~$f_H/64$。

快速唤醒发生后，f_{SUB} 为单片机提供一个次时钟。f_{SUB} 用于时基和看门狗定时器的的时钟源。



系统时钟选项

注：当系统时钟源 f_{SYS} 由 f_H 到 f_{SUB} 转换时，高速振荡器将停止以节省耗电。因此，没有为外围电路提供 f_H~$f_H/64$ 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：正常模式和低速模式。剩余的 4 种工作模式：休眠模式 0、休眠模式 1、空闲模式 0 和空闲模式 1 用于单片机 CPU 关闭时以节省耗电。

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

正常模式

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

低速模式

此模式的系统时钟虽为较低速时钟源，但单片机仍能正常工作。该低速时钟源可来自 LXT 或 LIRC 振荡器。单片机在此模式中运行所耗工作电流较低。在低速模式下，f_H 关闭。

休眠模式 0

在 HALT 指令执行后且 SMOD 寄存器中 IDLEN 位为低时，系统进入休眠模式。在休眠模式 0 中，CPU 及 f_{SUB} 停止运行，看门狗定时器功能除能。在该模式中 LVDEN 位需置为“0”，否则将不能进入休眠模式 0 中。

休眠模式 1

在 HALT 指令执行后且 SMOD 寄存器中 IDLEN 位为低时，系统进入休眠模式。在休眠模式 1 中，CPU 停止运行。然而，若 LVDEN 位为“1”或看门狗定时器功能使能，f_{SUB} 继续运行。

空闲模式 0

执行 HALT 指令后且 SMOD 寄存器中 IDLEN 位为高，SMOD1 寄存器中 FSYSON 位为低时，系统进入空闲模式 0。在空闲模式 0 中，CPU 停止，但一些外围功能如看门狗定时器和 TM 将继续工作。在空闲模式 0 中，系统振荡器停止，f_{SUB} 时钟开启

空闲模式 1

执行 HALT 指令后且 SMOD 寄存器中 IDLEN 位为高，SMOD1 寄存器中 FSYSON 位为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，但会提供一个时钟源给一些外围功能如看门狗定时器和 TM。在空闲模式 1 中，系统振荡器继续运行，该系统振荡器可以为高速或低速系统振荡器。在该模式中，f_{SUB} 时钟开启。

控制寄存器

寄存器 SMOD 和 SOMD1 用于控制单片机内部时钟和相关的振荡器配置。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SMOD	CKS2	CKS1	CKS0	FSTEN	LTO	HTO	IDLEN	HLCLK
SMOD1	FSYSON	—	—	—	RSTF	LVRF	LRF	WRF

SMOD 寄存器

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

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

Bit 4 **FSTEN**：快速唤醒控制位 (仅用于 HXT)
0：除能
1：使能
此位为快速唤醒控制位，用于决定单片机被唤醒后 f_{SUB} 是否开始工作。当此位为高且 f_{SUB} 时钟可用，该时钟源用作临时系统时钟以提供快速唤醒时间。

Bit 3 **LTO**：低速系统振荡器就绪标志位
0：未就绪
1：就绪
此位为低速系统振荡器就绪标志位，用于表明低速系统振荡器在系统上电复位或经唤醒后何时稳定下来。当系统处于 SLEEP0 模式时，该标志为低。若系统时钟来自 LXT 振荡器，系统唤醒后该位转换为高需 1024 个时钟周期；若系统时钟来自 LIRC 振荡器，该位转换为高需 1~2 个时钟周期。

Bit 2 **HTO**：高速系统振荡器就绪标志位
0：未就绪
1：就绪
此位为高速系统振荡器就绪标志位，用于表明高速系统振荡器何时稳定下来。此标志在系统上电后经硬件清零，高速系统振荡器稳定后变为高电平。因此，此位在单片机上电后由应用程序读取的总为“1”。该标志由休眠模式或空闲模式 0 中唤醒后会处于低电平状态，若使用 HXT 振荡器，该位将在 1024 个时钟周期后变为高电平状态，若使用 HIRC 振荡器则只需 15~16 个时钟周期即可。

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

Bit 0 **HLCLK**：系统时钟选择位
0：f_H/2~ f_H/64 或 f_{SUB}
1：f_H
此位用于选择 f_H 或 f_H/2~ f_H/64 还是 f_{SUB} 作为系统时钟。该位为高时选择 f_H 作为系统时钟，为低时则选择 f_H/2~ f_H/64 或 f_{SUB} 作为系统时钟。当系统时钟由 f_H 时钟向 f_{SUB} 时钟转换时，f_H 将自动关闭以降低功耗。

SMOD1 寄存器

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

“x”：未知

Bit 7 **FSYSON**：f_{SYS} 在空闲模式下的控制位
0：除能
1：使能

Bit 6~4 未定义，读为“0”

Bit 3 **RSTF**：复位控制寄存器软件复位标志位
0：无效
1：有效
当发生 RSTC 控制寄存器软件复位时，该位被置为“1”，由应用程序清零。注意，此位只能由应用程序清零。

Bit 2 **LVRF**：LVR 复位标志位
0：无效
1：有效
当特定的低电压复位条件发生时，该位被置为“1”。该位只能由应用程序清零。

Bit 1 **LRF**：LVR 控制寄存器软件复位标志位
0：无效
1：有效
如果 LVRC 寄存器包含任何非定义的 LVR 电压值，该位被置为“1”，这类似于软件复位功能。该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
0：无效
1：有效
WDT 控制寄存器复位时，该位被置为“1”，且通过应用程序清除。注意，该位只能由应用程序清零。

快速唤醒

单片机进入休眠模式或空闲模式 0 后，系统时钟将停止以降低功耗。然而单片机再次唤醒，原来的系统时钟重新起振、稳定且恢复正常工作需要一定的时间。为确保单片机能够尽快的开始工作，系统提供了一个快速唤醒功能。需提供一个临时时钟源 f_{SUB} 先驱动系统直至原系统振荡器稳定，这个临时时钟可来自 LXT 或 LIRC 振荡器。快速启动功能的时钟源为 f_{SUB}，该功能仅在休眠模式 1 和空闲模式 0 中有效。当单片机由休眠模式 0 唤醒时，因 f_{SUB} 已停止，故快速唤醒功能无效。快速唤醒功能使能/除能由 SMOD 寄存器中 FSTEN 位控制的。

若 HXT 振荡器作为正常模式的系统时钟，且快速唤醒功能使能，系统唤醒将需 1~2 个 t_{SUB} 时钟周期。系统开始在 f_{SUB} 时钟源下运行直至 1024 个 HXT 时钟周期后 HTO 标志转换为高，系统将切换到 HXT 振荡器运行。

若系统振荡器选用 HIRC，将系统从休眠模式或空闲模式 0 中唤醒需 15~16 个时钟周期；若选用 LIRC，则需 1~2 个周期。快速唤醒位 FSTEN 在这些情况下不受影响。

系统 振荡器	FSTEN 位	唤醒时间 (休眠模式 0)	唤醒时间 (休眠模式 1)	唤醒时间 (空闲模式 0)	唤醒时间 (空闲模式 1)
HXT	0	1024 个 HXT 周期	1024 个 HXT 周期		1~2 个 HXT 周期
	1	1024 个 HXT 周期	1~2 个 f_{SUB} 周期 (系统在 f_{SUB} 下运行 1024 个 HXT 周期后切换到 HXT 振荡器运行)		1~2 个 HXT 周期
HIRC	×	15~16 个 HIRC 周期	15~16 个 HIRC 周期		1~2 个 HIRC 周期
LIRC	×	1~2 个 LIRC 周期	1~2 个 LIRC 周期		1~2 个 LIRC 周期
LXT	×	1024 个 LXT 周期	1024 个 LXT 周期		1~2 个 LXT 周期

唤醒时间

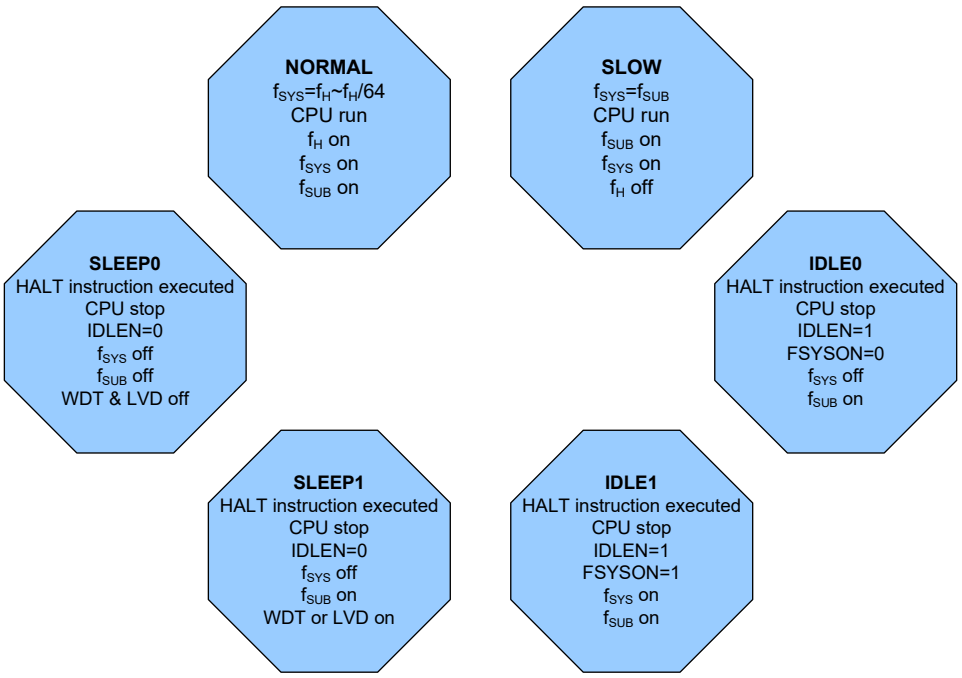
注：若看门狗定时器除能，意味着 LXT 或 LIRC 都关闭，当单片机由休眠模式 0 中唤醒时快速唤醒功能不可用。

工作模式切换

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

简单来说，正常模式和低速模式间的切换仅需设置 SMOD 中的 HLCLK 位及 CKS2~CKS0 位即可实现，而正常模式 / 低速模式与休眠模式 / 空闲模式间的切换经由 HALT 指令实现。当 HALT 指令执行后，单片机是否进入空闲模式或休眠模式由 SMOD 寄存器中的 IDLEN 位和 SMOD1 寄存器中的 FSYSON 位决定的。

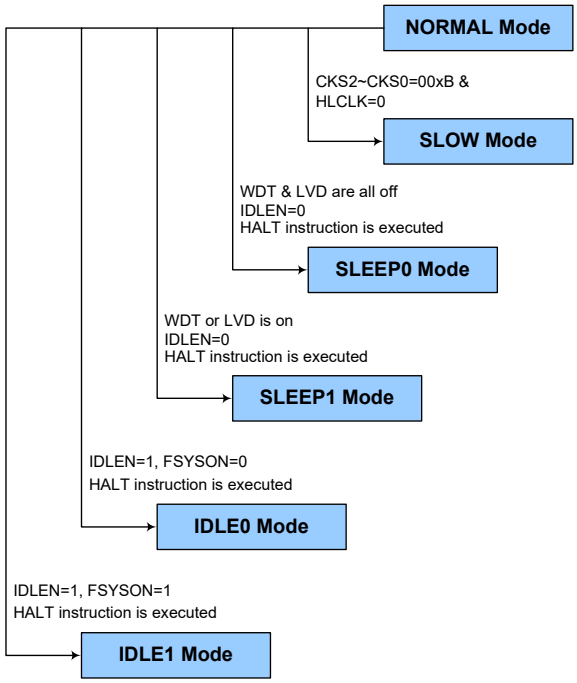
当 HLCLK 位变为低电平时，时钟源将由高速时钟源 f_H 转换成时钟源 $f_H/2$~$f_H/64$ 或 f_{SUB}。若时钟源来自 f_{SUB}，高速时钟源将停止运行以节省耗电。此时须注意，$f_H/16$ 和 $f_H/64$ 内部时钟源也将停止运行，由此会影响到如 TM 等内部功能的工作。所附流程图显示了单片机在不同工作模式间切换时的变化。



正常模式切换到低速模式

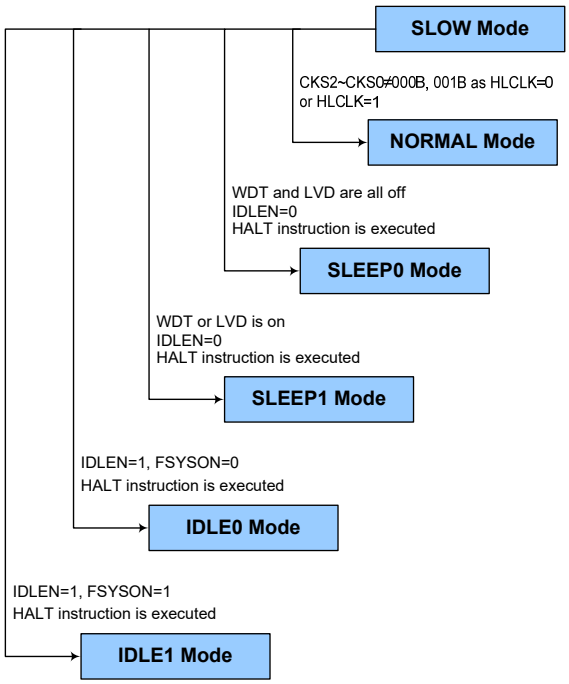
系统运行在正常模式时使用高速系统振荡器，因此较为耗电。可通过设置 SMOD 寄存器中的 HLCLK 位为“0”及 CKS2~CKS0 位为“000”或“001”使系统时钟切换至运行在低速模式下。此时将使用低速系统振荡器以节省耗电。用户可在对性能要求不高的操作中使用此方法以减少耗电。

低速模式的时钟源来自 LXT 或 LIRC 振荡器，因此要求这些振荡器在所有模式切换动作发生前稳定下来。该动作由 SMOD 寄存器中 LTO 位控制。



低速模式切换到正常模式

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



进入休眠模式 0

进入休眠模式 0 的方法仅有一种——应用程序中执行“HALT”指令前需设置寄存器 SMOD 中 IDLEN 位为“0”且 WDT 和 LVD 功能除能。在上述条件下执行该指令后，将发生的情况如下：

- 系统时钟和 f_{SUB} 时钟停止运行，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 由于 WDT 功能除能，WDT 将被清除并停止运行。

进入休眠模式 1

进入休眠模式 1 的方法仅有一种——应用程序中执行“HALT”指令前需设置寄存器 SMOD 中 IDLEN 位为“0”且 WDT 或 LVD 功能使能。在上述条件下执行该指令后，将发生的情况如下：

- 系统时钟停止运行，应用程序停止在“HALT”指令处。WDT 或 LVD 继续运行，其时钟源来自 f_{SUB}。
- 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。
- 如果 WDT 使能且其时钟源来自 f_{SUB}，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”指令前需设置寄存器 SMOD 中 IDLEN 位为“1”且 SMOD1 寄存器中的 FSYSON 位为“0”。在上述条件下执行该指令后，将发生的情况如下：

- 系统时钟停止运行，应用程序停止在“HALT”指令处，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。
- 如果 WDT 使能，由于 WDT 时钟源来自 f_{SUB}，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”指令前需设置寄存器 SMOD 中 IDLEN 位为“1”且 SMOD1 寄存器中的 FSYSON 位为“1”。在上述条件下执行该指令后，将发生的情况如下：

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

静态电流的注意事项

由于单片机进入休眠或空闲模式的主要原因是将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 除外)，所以如果要将电路的电流进一步降低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入 / 输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。

另外还需注意单片机设为输出的 I/O 引脚上的负载。应将它们设置在有最小拉电流的状态或将它们和其它的 CMOS 输入一样接到没有拉电流的外部电路上。还应注意的是，如果使能配置选项中的 LXT 或 LIRC 振荡器，会导致耗电增加。

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

唤醒

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

系统进入休眠或空闲模式之后，可以通过以下几种方式唤醒：

- 外部复位
- PA 口下降沿
- 系统中断
- WDT 溢出

若由外部 $\overline{\text{RES}}$ 引脚唤醒，系统会经过完全复位的过程；若由 WDT 溢出唤醒，则会发生看门狗定时器复位。这两种唤醒方式都会使系统复位，可以通过状态寄存器中 TO 和 PDF 位来判断它的唤醒源。系统上电或执行清除看门狗的指令，会清零 PDF；执行 HALT 指令，PDF 将被置位；看门狗计数器溢出将会置位 TO 标志并唤醒系统，这种复位会重置程序计数器和堆栈指针，其它标志保持原有状态。

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

看门狗定时器

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

看门狗定时器时钟源

WDT 定时器时钟源来自于内部时钟 f_{SUB}，而 f_{SUB} 的时钟源又由 LXT 或 LIRC 振荡器提供，可通过配置选项设置。内部振荡器 LIRC 的周期大约为 32kHz。需要注意的是，这个特殊的内部时钟周期随 V_{DD}、温度和制成的不同而变化。LXT 振荡器由一个外部 32.768kHz 晶振提供。看门狗定时器的时钟源可分频为 2^8~2^{18} 以提供更大的溢出周期，分频比由 WDTC 寄存器中的 WS2~WS0 位来决定。

看门狗定时器控制寄存器

WDTC 寄存器用于控制 WDT 功能的使能 / 除能及选择溢出周期。

WDTC 寄存器

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

Bit 7~3 **WE4~WE0**：WDT 软件控制位
10101：除能
01010：使能
其它值：复位 MCU
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 2~3 个 LIRC 时钟周期后，且 SMOD1 寄存器的 WRF 位将置为“1”。

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

SMOD1 寄存器

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

“x”：未知

Bit 7 **FSYSON**：f_{SYS} 在空闲模式下的控制位
详见其它章节

Bit 6~4 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
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：无效
1：有效
WDT 控制寄存器复位时，该位被置为“1”，且通过应用程序清除。注意，该位只能由应用程序清零。

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 WDTC 中有五位 WE4~WE0 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 2~3 个 f_{LIRC} 时钟周期后复位。上电后这些位初始化为“01010B”。

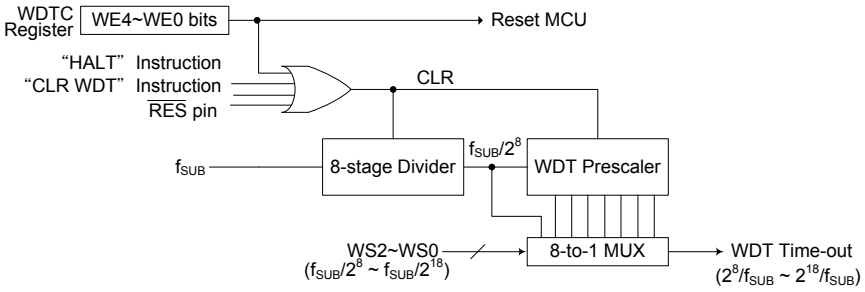
WE4~WE0 位	WDT 功能
10101B	除能
01010B	使能
其它值	复位单片机

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 将被置位，只有程序计数器 PC 和堆栈指针 SP 将被复位。有四种方法可以用来清除 WDT 的内容。第一种是外部复位，即 $\overline{RES}$ 引脚低电平。第二种是 WDT 复位，将除“01010B”和“10101B”以外的值写入 WE4~WE0 即可实现，第三种是通过看门狗定时器软件清除指令，而第四种是通过“HALT”指令。

软件指令清除看门狗寄存器只有一种方法。只要执行“CLR WDT”便清除 WDT。

当设置分频比为 2^{18} 时，溢出周期最大。例如，时钟源为 32kHz LIRC 振荡器，分频比为 2^{18} 时最大溢出周期约 8s，分频比为 2^8 时最小溢出周期约 7.8ms。



看门狗定时器

复位和初始化

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

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，$\overline{\text{RES}}$ 脚被强制拉为低电平。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。

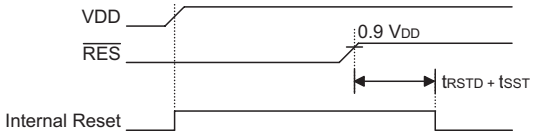
另一种复位为看门狗溢出单片机复位。不同方式的复位操作会对寄存器产生不同的影响。另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位，这种复位与 $\overline{\text{RES}}$ 脚拉低复位方式相似。

复位功能

包括内部和外部事件触发复位，单片机共有五种复位方式：

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定在预设条件。所有的输入 / 输出端口控制寄存器在上电复位时会保持高电平，以确保上电后所有引脚被设定为输入状态。



上电复位时序图

$\overline{\text{RES}}$ 引脚复位

由于复位引脚与 I/O 口共用，复位功能必须使用 RSTC 控制寄存器选择。RSTC 寄存器用来决定该引脚为外部复位引脚或 I/O 引脚。如果 RSTC 寄存器的内容被设置为除 01010101B 或 10101010B 以外的任何值，单片机会在 2~3 个 f_{LIRC} 时钟周期后发生复位。上电后寄存器的值为 01010101B。

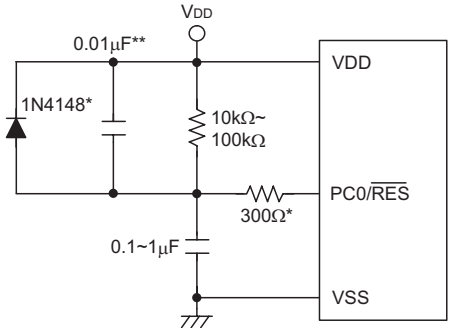
RSTC7~RSTC0 位	复位功能
01010101B	I/O 引脚
10101010B	$\overline{\text{RES}}$ 引脚
其它值	MCU 复位

内部复位功能控制

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。



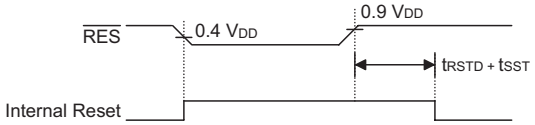
注：“*”表示建议加上此元件以加强静电保护。

“**”表示建议在电源有较强干扰场合加上此元件。

外部 $\overline{\text{RES}}$ 电路

欲知有关外部复位电路的更多信息可参考 HOLTEK 网站上的应用范例 HA0075S。

$\overline{\text{RES}}$ 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。



$\overline{\text{RES}}$ 复位时序图

● **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：外部复位引脚
其它值：复位 MCU
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 2~3 个 f_{LIRC} 时钟周期后，且 SMOD1 寄存器的 RSTF 位将置为“1”。

● **SMOD1 寄存器**

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

“x”：未知

Bit 7 **FSYSON**：f_{SYS} 在空闲模式下的控制位
详见其它章节

Bit 6~4 未定义，读为“0”

Bit 3 **RSTF**：复位控制寄存器软件复位标志位
0：无效
1：有效
此位由 RSTC 控制寄存器软件复位置位，由应用程序清零。注意，此位只能由应用程序清零。

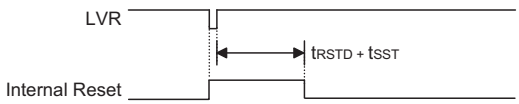
Bit 2 **LVRF**：LVR 复位标志位
详见其它章节

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。由于特定的 LVR 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落在 0.9V~V_{LVR} 的范围内，这时 LVR 将会自动复位单片机且 SMOD1 寄存器的 LVRF 位也被设置为“1”。LVR 包含以下的规格：有效的 LVR 信号，即在 0.9V~V_{LVR} 的低电压状态的时间，必须超过 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。实际的 V_{LVR} 参数值可通过 LVRC 寄存器中的 LVS 位进行选择。如果由于不利的环境干扰因素使得 LVS7~LVS0 为其它值，LVR 将在 2~3 个 f_{LIRC} 时钟周期后复位单片机。此时，SMOD1 寄存器中的 LRF 位被设置为“1”。寄存器 LVRC 上电后初始化为“01010101B”。当单片机进入省电模式时 LVR 功能将自动除能。



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：复位单片机 - 寄存器复位为 POR 值。
当低电压条件发生时，以上四个已定义的任何一个 LVR 电压值都会使单片机复位。复位动作将在 2~3 个 f_{LIRC} 时钟周期后有效。此时复位后的寄存器内容将保持不变。
任何除上述四个已定义的寄存器值，也会导致单片机复位。复位动作将在 2~3 个 f_{LIRC} 时钟周期后有效。但此时寄存器内容将复位为 POR 值。

● **SMOD1 寄存器**

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

“x”：未知

Bit 7 **FSYSON**：f_{SYS} 在空闲模式下的控制位
详见其它章节

Bit 6~4 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

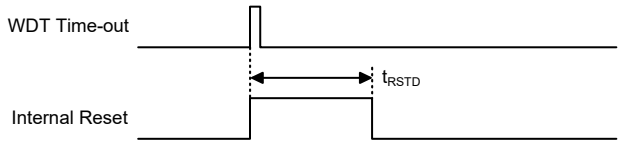
Bit 2 **LVRF**：LVR 复位标志位
0：无效
1：有效
当特定的低电压复位条件发生时，该位被置为“1”。该位只能由应用程序清零。

Bit 1 **LRF**：LVR 控制寄存器软件复位标志位
0：无效
1：有效
如果 LVRC 寄存器包含任何非定义的 LVR 电压值，该位被置为“1”，这类似于软件复位功能。该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

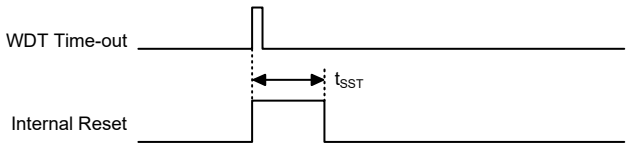
除了看门狗溢出标志位 TO 将被设为“1”之外，正常运行时看门狗溢出复位和 $\overline{RES}$ 复位相同。



正常运行时看门狗溢出时序图

休眠或空闲时看门狗溢出复位

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



休眠或空闲时看门狗溢出复位时序图

复位初始状态

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

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

注：“u”代表不改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

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

不同的复位形式对单片机内部寄存器的影响是不同的。为保证复位后程序能正常执行，了解寄存器在特定条件复位后的设置是非常重要的。下表即为不同方式复位后内部寄存器的状况。若芯片有多种封装类型，表格反应较大的封装的情况。

寄存器	HT67F60A	HT67F70A	上电复位	LVR 复位 (正常模式)	WDT 溢出 (正常模式)	WDT 溢出 (空闲 / 休眠模式)
IAR0	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR1	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	●	●	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	●		--xx xxxx	---- uuuu	---- uuuu	---- uuuu
TBHP		●	-xxx xxxx	---- uuuu	---- uuuu	---- uuuu
STATUS	●	●	xx00 xxxx	uuuu uuuu	uu1u uuuu	uu11 uuuu
BP	●		---- ---0	---- ---0	---- ---0	---- ---u
BP		●	---- --00	---- --00	---- --00	---- --uu
IAR2	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP2L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PA	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
PB	●	●	-000 1111	-000 1111	-000 1111	-uuu uuuu
PBC	●	●	-111 1111	-111 1111	-111 1111	-uuu uuuu
PCPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PE	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PF	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
LCDC0	●	●	00-- 0000	00-- 0000	00-- 0000	uu-- uuuu
LCDC1	●	●	000- 0000	000- 0000	000- 0000	uuu- uuuu
RSTC	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu

寄存器	HT67F60A	HT67F70A	上电复位	LVR 复位 (正常模式)	WDT 溢出 (正常模式)	WDT 溢出 (空闲 / 休眠模式)
INTC0	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC3	●	●	-000 -000	-000 -000	-000 -000	-uuu -uuu
MFI0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI1	●	●	-000 -000	-000 -000	-000 -000	-uuu -uuu
MFI2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI3	●	●	-000 -000	-000 -000	-000 -000	-uuu -uuu
MFI4	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTEG	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SMOD	●	●	0000 0011	0000 0011	0000 0011	uuuu uuuu
SMOD1	●	●	0--- 0x00	0--- 0x00	0--- 0x00	u--- uuuu
LVRC	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
LVDC	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
WDTC	●	●	0101 0011	0101 0011	0101 0011	uuuu uuuu
SMOD2	●	●	---- ---0	---- ---0	---- ---0	---- ---u
EEA	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
EED	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CP0C	●	●	-000 ---1	-000 ---1	-000 ---1	-uuu ---u
CP1C	●	●	-000 ---1	-000 ---1	-000 ---1	-uuu ---u
ETMC0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ETMC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ETMC2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ETMDL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ETMDH	●	●	---- --00	---- --00	---- --00	---- --uu
ETMAL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ETMAH	●	●	---- --00	---- --00	---- --00	---- --uu
ETMBL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ETMBH	●	●	---- --00	---- --00	---- --00	---- --uu
STM0C0	●	●	0000 0---	0000 0---	0000 0---	uuuu u---
STM0C1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0AL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0AH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM0RP	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1C0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1C1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu

寄存器	HT67F60A	HT67F70A	上电复位	LVR 复位 (正常模式)	WDT 溢出 (正常模式)	WDT 溢出 (空闲 / 休眠模式)
CTM1DH	●	●	---- --00	---- --00	---- --00	---- --uu
CTM1AL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1AH	●	●	---- --00	---- --00	---- --00	---- --uu
CTM0C0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0C1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DH	●	●	---- --00	---- --00	---- --00	---- --uu
CTM0AL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0AH	●	●	---- --00	---- --00	---- --00	---- --uu
PSC0	●	●	---- --00	---- --00	---- --00	---- --uu
TBC0	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
TBC1	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
PSC1	●	●	---- --00	---- --00	---- --00	---- --uu
SADC0	●	●	0110 0000	0110 0000	0110 0000	uuuu uuuu
SADC1	●	●	-000 -000	-000 -000	-000 -000	-uuu –uuu
SADOL (ADRFS=0)	●	●	xxxx ----	xxxx ----	xxxx ----	uuuu ----
SADOL (ADRFS=1)	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
SADOH (ADRFS=0)	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
SADOH (ADRFS=1)	●	●	---- xxxx	---- uuuu	---- uuuu	---- uuuu
SIMC0	●	●	111- 000-	111- 000-	111- 000-	uuu- uuu-
SIMC1	●	●	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMC1/SIMA	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
I2CTOC	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SPIAC0	●	●	111- --0-	111- --0-	111- --0-	uuu- --u-
SPIAC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SPIAD	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARH	●		--00 0000	--00 0000	--00 0000	--uu uuuu
FARH		●	-000 0000	-000 0000	-000 0000	-uuu uuuu
FD0L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TBC2	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu

寄存器	HT67F60A	HT67F70A	上电复位	LVR 复位 (正常模式)	WDT 溢出 (正常模式)	WDT 溢出 (空闲 / 休眠模式)
EEC	●	●	---- 0000	---- 0000	---- 0000	---- uuuu
FC0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC2	●	●	---- ---0	---- ---0	---- ---0	---- ---u
IFS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS3	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS4	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1C0	●	●	0000 0---	0000 0---	0000 0---	uuuu u---
STM1C1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1AL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1AH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM1RP	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2C0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2C1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2AL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2AH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
STM2RP	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PES0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PES1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PFS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PFS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu

注：“-”表示未定义
“u”表示不改变
“x”表示未知

输入 / 输出端口

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

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PBPU	—	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PB	—	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	—	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PEPU	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PE	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEC	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0
PF	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFC	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0

输入 / 输出寄存器列表

“—”：未定义，读为“0”

PAWUn：PA 唤醒功能控制
0：除能
1：使能

PAPUn/PBPUn/PCPUn/PDPUn/PEPUn/PFPUn：上拉功能控制
0：除能
1：使能

PAn/PBn/PCn/PDn/PEn/PFn：I/O 口数据位
0：数据 0
1：数据 1

PACn/PBCn/PCCn/PDCn/PECn/PFCn：I/O 口类型选择
0：输出
1：输入

上拉电阻

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

PA 口唤醒

当使用暂停指令“HALT”迫使单片机进入休眠或空闲模式，单片机的系统时钟将会停止以降低功耗，此功能对于电池及低功耗应用很重要。唤醒单片机有很多种方法，其中之一就是使 PA 口的其中一个引脚从高电平转为低电平。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。

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 **PAWU**：PA 口 bit 7~bit 0 唤醒功能控制位
0：除能
1：使能

输入 / 输出端口控制寄存器

每一个输入 / 输出口都具有各自的控制寄存器，即 PAC~PFC，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，动态的设置为带或不带上拉电阻。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。

引脚共用功能

引脚的多功能可以增加单片机应用的灵活性。有限的引脚个数将会限制设计者，而引脚的多功能将会解决很多此类问题。此外，这些引脚功能可以通过一系列寄存器进行设定。

引脚共用寄存器

封装中有限的引脚个数会对某些单片机功能造成影响。然而，引脚功能共用和引脚功能选择，使得小封装单片机具有更多不同的功能。单片机包含端口“x”输出功能选择寄存器“n”，记为 PxSn，且输入功能选择寄存器“i”，记为 IFSi，这些寄存器可以用来选择共用引脚的特定功能。

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，I^2C SDA 端口被使用，对应的输出引脚共用功能则要通过寄存器 PxSn 设置为 SDI/SDA 功能，且 SDA 信号输入也要通过 IFSi 寄存器进行合理选择。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输入 / 输出口，且也要选择中断输入信号。定时器模块输入也同理。

要注意的最重要一点是，确保所需的引脚共用功能被正确地选择和取消。要选择所需的引脚共用功能，首先应通过相应的引脚共用控制寄存器正确地选择该功能，然后再配置相应的外围功能设置以使能外围功能。要正确地取消引脚共用功能，首先应除能外围功能，然后再修改相应的引脚共用控制寄存器以选择其它的共用功能。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PAS0	PA3S1	PA3S0	PA2S1	PA2S0	PA1S1	PA1S0	PA0S1	PA0S0
PAS1	PA7S1	PA7S0	PA6S1	PA6S0	PA5S1	PA5S0	PA4S1	PA4S0
PBS0	PB3S1	PB3S0	PB2S1	PB2S0	PB1S1	PB1S0	PB0S1	PB0S0
PBS1	—	—	PB6S1	PB6S0	PB5S1	PB5S0	PB4S1	PB4S0
PCS0	PC3S1	PC3S0	PC2S1	PC2S0	PC1S1	PC1S0	PC0S1	PC0S0
PCS1	PC7S1	PC7S0	PC6S1	PC6S0	PC5S1	PC5S0	PC4S1	PC4S0
PDS0	PD3S1	PD3S0	PD2S1	PD2S0	PD1S1	PD1S0	PD0S1	PD0S0
PDS1	PD7S1	PD7S0	PD6S1	PD6S0	PD5S1	PD5S0	PD4S1	PD4S0
PES0	PE3S1	PE3S0	PE2S1	PE2S0	PE1S1	PE1S0	PE0S1	PE0S0
PES1	PE7S1	PE7S0	PE6S1	PE6S0	PE5S1	PE5S0	PE4S1	PE4S0
PFS0	PF3S1	PF3S0	PF2S1	PF2S0	PF1S1	PF1S0	PF0S1	PF0S0
PFS1	PF7S1	PF7S0	PF6S1	PF6S0	PF5S1	PF5S0	PF4S1	PF4S0
IFS0	PINTBS	—	—	—	INT3S	INT2S	INT1S	INT0S
IFS1	—	STCK0S	STP0IS	ETCKS	ETPIBS	ETPIAS	CTCK0S	—
IFS2	—	—	STCK2S	STP2IS	STCK1S	STP1IS	CTCK1S	—
IFS3	—	—	SDIAS	SDIS	SCKAS	SCKS	SCSAS	SCSS
IFS4	—	—	—	C1NS	C1PS	—	C0NS	C0PS

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	PA3S1	PA3S0	PA2S1	PA2S0	PA1S1	PA1S0	PA0S1	PA0S0
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 **PA3S1~PA3S0**：PA3 引脚功能选择
00：PA3/CTCK0
01：SDI/SDA
10：VREF
11：AN1

Bit 5~4 **PA2S1~PA2S0**：PA2 引脚功能选择
00/01/10：PA2
11：AN4

Bit 3~2 **PA1S1~PA1S0**：PA1 引脚功能选择
00：PA1
01：SDO
10：CTP0
11：AN0

Bit 1~0 **PA0S1~PA0S0**：PA0 引脚功能选择
00/01/10：PA0
11：AN5

PAS1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PA7S1	PA7S0	PA6S1	PA6S0	PA5S1	PA5S0	PA4S1	PA4S0
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 **PA7S1~PA7S0**：PA7 引脚功能选择
00/10：PA7/INT1/STP0I
01：STP0
11：AN7

Bit 5~4 **PA6S1~PA6S0**：PA6 引脚功能选择
00/01/10：PA6/INT0/ETCK
11：AN6

Bit 3~2 **PA5S1~PA5S0**：PA5 引脚功能选择
00：PA5/ETPIB
01：$\overline{SCS}$
10：ETPB
11：AN3

Bit 1~0 **PA4S1~PA4S0**：PA4 引脚功能选择
00：PA4/ETPIA/$\overline{PINT}$
01：SCK/SCL
10：ETPA
11：AN2

PBS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PB3S1	PB3S0	PB2S1	PB2S0	PB1S1	PB1S0	PB0S1	PB0S0
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 **PB3S1~PB3S0**：PB3 引脚功能选择
00：PB3
01：INT3
10：CTP1
11：AN11

Bit 5~4 **PB2S1~PB2S0**：PB2 引脚功能选择
00：PB2
01：INT2
10：STCK0
11：AN10

Bit 3~2 **PB1S1~PB1S0**：PB1 引脚功能选择
00：PB1
10：PB1
01：SCKA
11：AN9

Bit 1~0 **PB0S1~PB0S0**：PB0 引脚功能选择
00：PB0
10：$\overline{CTCK1}$
01：$\overline{SCSA}$
11：AN8

PBS1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	PB6S1	PB6S0	PB5S1	PB5S0	PB4S1	PB4S0
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　**PB6S1~PB6S0**：PB6 引脚功能选择
00/01/10：PB6
11：C2

Bit 3~2　**PB5S1~PB5S0**：PB5 引脚功能选择
00/01/10：PB5
11：C1

Bit 1~0　**PB4S1~PB4S0**：PB4 引脚功能选择
00/01/10：PB4
11：V2

PCS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PC3S1	PC3S0	PC2S1	PC2S0	PC1S1	PC1S0	PC0S1	PC0S0
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　**PC3S1~PC3S0**：PC3 引脚功能选择
00：PC3/STCK1
01：PCK
10：C0P
11：SEG3

Bit 5~4　**PC2S1~PC2S0**：PC2 引脚功能选择
00：PC2/STP2I
01：STP2
10：C1X
11：SEG2

Bit 3~2　**PC1S1~PC1S0**：PC1 引脚功能选择
00/01：PC1/STCK2/$\overline{\text{PINT}}$
10：C1N
11：SEG1

Bit 1~0　**PC0S1~PC0S0**：PC0 引脚功能选择
00/01：PC0
10：C1P
11：SEG0

PCS1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PC7S1	PC7S0	PC6S1	PC6S0	PC5S1	PC5S0	PC4S1	PC4S0
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 **PC7S1~PC7S0**：PC7 引脚功能选择
00：PC7/STCK0
01：SCSA
10：C0N
11：SEG7

Bit 5~4 **PC6S1~PC6S0**：PC6 引脚功能选择
00：PC6
01：SCKA
10：CTP1
11：SEG6

Bit 3~2 **PC5S1~PC5S0**：PC5 引脚功能选择
00：PC5/CTCK1
01：SDIA
10：C0X
11：SEG5

Bit 1~0 **PC4S1~PC4S0**：PC4 引脚功能选择
00：PC4/STP1I
01：SDOA
10：STP1
11：SEG4

PDS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PD3S1	PD3S0	PD2S1	PD2S0	PD1S1	PD1S0	PD0S1	PD0S0
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 **PD3S1~PD3S0**：PD3 引脚功能选择
00：PD3
01：SDO
10：C1N
11：SEG11

Bit 5~4 **PD2S1~PD2S0**：PD2 引脚功能选择
00：PD2/ETPIB
01：SDI/SDA
10：ETPB
11：SEG10

Bit 3~2 **PD1S1~PD1S0**：PD1 引脚功能选择
00：PD1/ETCK
01：SCK/SCL
10：C1P
11：SEG9

Bit 1~0 **PD0S1~PD0S0**：PD0 引脚功能选择
00：PD0/STP0I
01：SCS
10：STP0
11：SEG8

PDS1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PD7S1	PD7S0	PD6S1	PD6S0	PD5S1	PD5S0	PD4S1	PD4S0
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 **PD7S1~PD7S0**：PD7 引脚功能选择
00：PD7/INT0/STP2I
01：STP2
10：C0X
11：SEG15

Bit 5~4 **PD6S1~PD6S0**：PD6 引脚功能选择
00/01：PD6/INT1/STCK2
10：C0N
11：SEG14

Bit 3~2 **PD5S1~PD5S0**：PD5 引脚功能选择
00：PD5/INT2
01：CTP0
10：C0P
11：SEG13

Bit 1~0 **PD4S1~PD4S0**：PD4 引脚功能选择
00：PD4/INT3/CTCK0/ETPIA
01：ETPA
10：C1X
11：SEG12

PES0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PE3S1	PE3S0	PE2S1	PE2S0	PE1S1	PE1S0	PE0S1	PE0S0
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 **PE3S1~PE3S0**：PE3 引脚功能选择
00/01/10：PE3
11：SEG19

Bit 5~4 **PE2S1~PE2S0**：PE2 引脚功能选择
00/01/10：PE2
11：SEG18

Bit 3~2 **PES1~PE1S0**：PE1 引脚功能选择
00：PE1/STP1I
01：SDIA
10：STP1
11：SEG17

Bit 1~0 **PE0S1~PE0S0**：PE0 引脚功能选择
00/10：PE0/STCK1
01：SDOA
11：SEG16

PES1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PE7S1	PE7S0	PE6S1	PE6S0	PE5S1	PE5S0	PE4S1	PE4S0
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 **PE7S1~PE7S0**：PE7 引脚功能选择
00/01/10：PE7
11：SEG23

Bit 5~4 **PE6S1~PE6S0**：PE6 引脚功能选择
00/01/10：PE2
11：SEG22

Bit 3~2 **PE5S1~PE5S0**：PE5 引脚功能选择
00/01/10：PE5
11：SEG21

Bit 1~0 **PE4S1~PE4S0**：PE4 引脚功能选择
00/01/10：PE4
11：SEG20

PFS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PF3S1	PF3S0	PF2S1	PF2S0	PF1S1	PF1S0	PF0S1	PF0S0
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 **PF3S1~PF3S0**：PF3 引脚功能选择
00/01/10：PF3
11：SEG27

Bit 5~4 **PF2S1~PF2S0**：PF2 引脚功能选择
00/01/10：PF2
11：SEG26

Bit 3~2 **PF1S1~PF1S0**：PF1 引脚功能选择
00/01/10：PF1
11：SEG25

Bit 1~0 **PF0S1~PF0S0**：PF0 引脚功能选择
00/01/10：PF0
11：SEG24

PFS1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PF7S1	PF7S0	PF6S1	PF6S0	PF5S1	PF5S0	PF4S1	PF4S0
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 **PF7S1~PF7S0**：PF7 引脚功能选择
00/01/10：PF7
11：SEG31

Bit 5~4 **PF6S1~PF6S0**：PF6 引脚功能选择
00/01/10：PF6
11：SEG30

Bit 3~2 **PF5S1~PF5S0**：PF5 引脚功能选择
00/01/10：PF5
11：SEG29

Bit 1~0 **PF4S1~PF4S0**：PF4 引脚功能选择
00/01/10：PF4
11：SEG28

IFS0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	PINTBS	—	—	—	INT3S	INT2S	INT1S	INT0S
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **PINTBS**：$\overline{\text{PINT}}$ 输入源引脚选择
0：PC1
1：PA4

Bit 6~4 未定义，读为“0”

Bit 3 **INT3S**：INT3 输入源引脚选择
0：PB3
1：PD4

Bit 2 **INT2S**：INT2 输入源引脚选择
0：PB2
1：PD5

Bit 1 **INT1S**：INT1 输入源引脚选择
0：PA7
1：PD6

Bit 0 **INT0S**：INT0 输入源引脚选择
0：PA6
1：PD7

IFS1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	STCK0S	STP0IS	ETCKS	ETPIBS	ETPIAS	CTCK0S	—
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 **STCK0S**：STCK0 输入源引脚选择
0：PB2
1：PC7

Bit 5 **STP0IS**：STP0I 输入源引脚选择
0：PA7
1：PD0

Bit 4 **ETCKS**：ETCK 输入源引脚选择
0：PA6
1：PD1

Bit 3 **ETPIBS**：ETPIB 输入源引脚选择
0：PA5
1：PD2

Bit 2 **ETPIAS**：ETPIA 输入源引脚选择
0：PA4
1：PD4

Bit 1 **CTCK0S**：CTCK0 输入源引脚选择
0：PA3
1：PD4

Bit 0 未定义，读为“0”

IFS2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	STCK2S	STP2IS	STCK1S	STP1IS	CTCK1S	—
R/W	—	—	R/W	R/W	R/W	R/W	R/W	—
POR	—	—	0	0	0	0	0	—

Bit 7~6 未定义，读为“0”

Bit 5 **STCK2S**：STCK2 输入源引脚选择
0：PC1
1：PD6

Bit 4 **STP2IS**：STP2I 输入源引脚选择
0：PC2
1：PD7

Bit 3 **STCK1S**：STCK1 输入源引脚选择
0：PC3
1：PE0

Bit 2 **STP1IS**：STP1I 输入源引脚选择
0：PC4
1：PE1

Bit 1 **CTCK1S**：CTCK1 输入源引脚选择
0：PB0
1：PC5

Bit 0 未定义，读为“0”

IFS3 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	SDIAS	SDIS	SCKAS	SCKS	SCSAS	SCSS
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　**SDIAS**：SDIA 输入源引脚选择
0：PC5
1：PE1

Bit 4　**SDIS**：SDI 输入源引脚选择
0：PA3
1：PD2

Bit 3　**SCKAS**：SCKA 输入源引脚选择
0：PB1
1：PC6

Bit 2　**SCKS**：SCK/SCL 输入源引脚选择
0：PA4
1：PD1

Bit 1　**SCSAS**：$\overline{\text{SCSA}}$ 输入源引脚选择
0：PB0
1：PC7

Bit 0　**SCSS**：$\overline{\text{SCS}}$ 输入源引脚选择
0：PA5
1：PD0

IFS4 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	C1NS	C1PS	—	C0NS	C0PS
R/W	—	—	—	R/W	R/W	—	R/W	R/W
POR	—	—	—	0	0	—	0	0

Bit 7~5　未定义，读为“0”

Bit 4　**C1NS**：C1N 输入源引脚选择
0：PC1
1：PD3

Bit 3　**C1PS**：C1P 输入源引脚选择
0：PC0
1：PD1

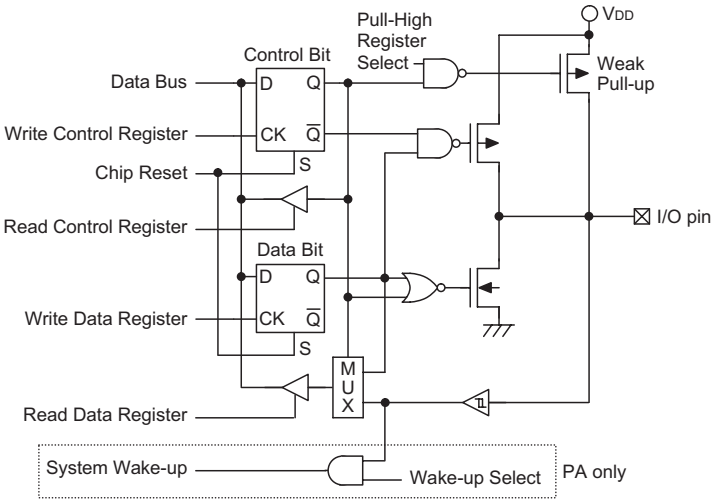
Bit 2　未定义，读为“0”

Bit 1　**C0NS**：C0N 输入源引脚选择
0：PC7
1：PD6

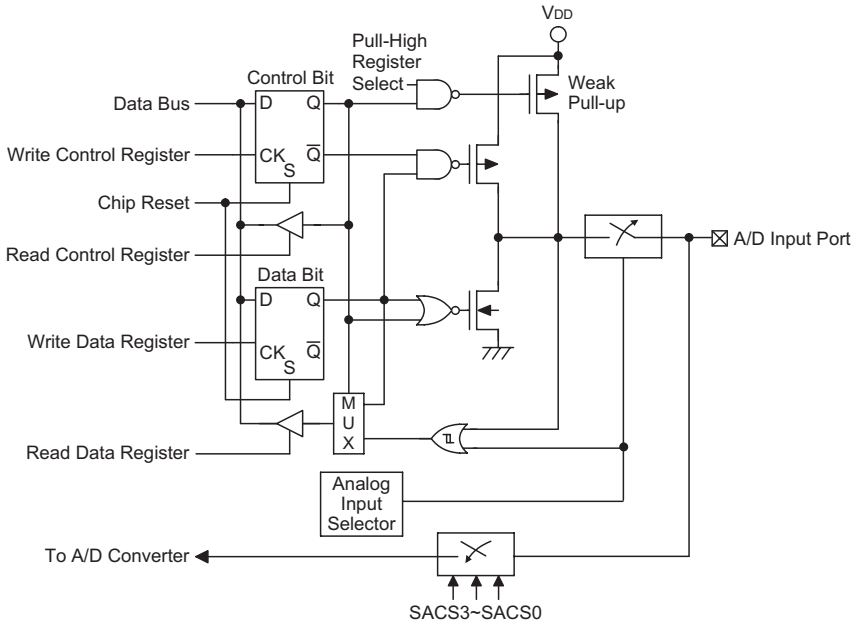
Bit 0　**C0PS**：C0P 输入源引脚选择
0：PC3
1：PD5

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。



通用输入 / 输出端口



A/D 输入 / 输出端口

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。每个定时器模块有多个独立中断。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简易型，标准型和增强型定时器章节。

简介

该系列单片机包含 6 个 TM，每个 TM 可被划分为一个特定的类型，即简易型 TM，标准型 TM 或增强型 TM。虽然性质相似，但不同 TM 特性复杂度不同。本章介绍简易型，标准型和增强型 TM 的共性，更多详细资料分别见后面各章。三种类型 TM 的特性和区别见下表。

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

TM 功能概要

单片机	CTM	STM	ETM
HT67F60A	10-bit CTM0 10-bit CTM1	16-bit STM0 16-bit STM1 16-bit STM2	10-bit ETM
HT67F70A			

TM 类型参考

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时钟来驱动内部 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 xTCK2~xTCK0 位，选择所需的时钟源，其中 x 代表 C、S 或 E，n 代表具体 TM 编号。该时钟源来自系统时钟 f_{SYS} 的分频比或内部高速时钟 f_H 或 f_{TBC} 时钟源或外部 xTCK 引脚。注意，这些位设置位“101”时，将选择为保留的时钟输入，实际上断开 TM 时钟源。xTCK 引脚时钟源用于允许外部信号作为 TM 时钟源或用于事件计数。

TM 中断

简易型 TM 和标准型 TM 都拥有两个内部中断，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。增强型 TM 有三个内部比较器，即比较器 A 或比较器 B 或比较器 P，相应的有三个内部中断。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，都有两个或三个 TM 输入引脚 xTCKn 和 xTPnI 或 ETPIA/ETPIB。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。TM 输入引脚可以选择上升沿或下降沿。STCKn 和 ETCK 引脚还可分别用作 STMn 和 ETM 单脉冲输出模式的外部触发输入引脚。

另一种 TM 输入引脚 STPnI 或 ETPIA/ETPIB 作为捕捉输入脚，其有效边沿有上升沿、下降沿和双沿，通过设置 STMnC1 或 ETMC1/ETMC2 寄存器中的 STnIO1~STnIO0 或 ETAIO1~ETAIO0/ETBIO1~ETBIO0 位来选择有效边沿类型。

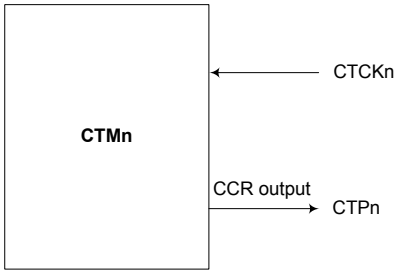
每个 TM 有一个或多个输出引脚，通过引脚共用功能章节描述的相关共用功能选择位来设置。当 TM 工作在比较匹配输出模式且比较匹配发生时，这些引脚会由 TM 控制切换到高电平或低电平或翻转。外部 xTP 或 ETPIA/ETPIB 输出引脚也被 TM 用来产生 PWM 输出波形。当 TM 输出引脚与其它功能共用时，TM 输出功能需要通过寄存器先被设置。

类型	CTM	STM	ETM	引脚控制寄存器
输入	CTCK0, CTCK1	STCK0, STCK1, STCK2 STP0I, STP1I, STP2I	ETCK ETPIA, ETPIB	IFSi
输出	CTP0, CTP1	STP0, STP1, STP2	ETPA, ETPB	PxSn

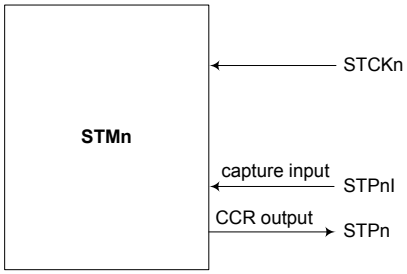
TM 外部引脚

TM 输入 / 输出引脚控制寄存器

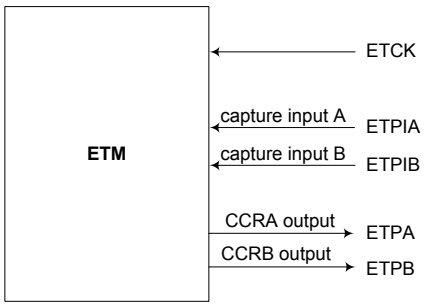
通过设置一或两个与 TM 输入 / 输出引脚相关的寄存器的一位，选择作为 TM 输入 / 输出功能或其它共用功能。正确设置选择位时，相关引脚用作 TM 输入 / 输出。更多引脚共用功能选择详见引脚共用功能章节。



CTMn 功能引脚控制框图 (n=0~1)



STMn 功能引脚控制框图 (n=0~2)

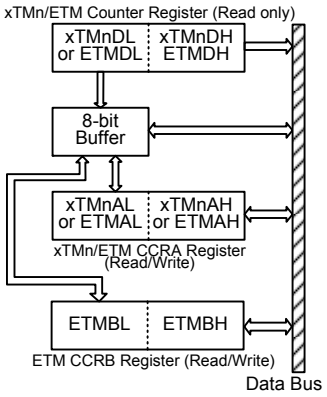


ETM 功能引脚控制框图

编程注意事项

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

正如 CCRA 和 CCRB 寄存器按照下图方式执行且具体存取这些寄存器对的方式如上所述，建议使用“MOV”指令，通过以下步骤访问 CCRA 和 CCRB 低字节，命名为 xTMAL 或 ETMAL 和 ETMBL。若不采用以下步骤访问 CCRA 和 CCRB 将导致不可预期的结果。



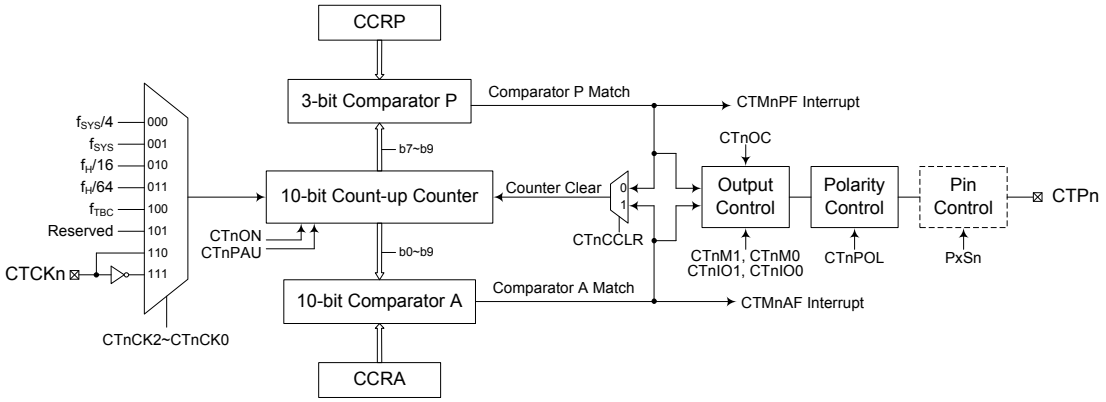
读写流程如下步骤所示：

- 写数据至 CCRB 或 CCRA
  - ♦ 步骤 1. 写数据至低字节寄存器 xTMnAL，ETMAL 或 ETMBL
    - – 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，ETMAH 或 ETMBH
    - – 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。
- 由计数器寄存器和 CCRB 或 CCRA 中读取数据
  - ♦ 步骤 1. 由高字节寄存器 xTMnDH，ETMDH，xTMnAH，ETMAH 或 ETMBH 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，ETMDL，xTMnAL，ETMAL 或 ETMBL 读取数据
    - – 注意，此时读取 8-bit 缓存器中的数据。

简易型 TM

虽然简易型 TM 是三种 TM 类型中最简单的形式，但仍然包括三种工作模式，即比较匹配输出，定时 / 事件计数器和 PWM 输出模式。简易型 TM 也由外部输入脚控制并驱动两个外部输出脚。两个外部输出脚信号可以相同也可以相反。

单片机	TM 类型	TM 输入引脚	TM 输出引脚
HT67F60A HT67F70A	10-bit CTM (CTM0, CTM1)	CTCK0, CTCK1	CTP0, CTP1



简易型 TM 方框图 (n=0 或 1)

简易型 TM 操作

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

通过应用程序改变 10 位计数器值的唯一方法是使 CTnON 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 TM 中断信号。简易型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，也可以控制输出脚。所有工作模式的设定都是通过设置相关寄存器来实现的。

简易型 TM 寄存器介绍

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
CTMnC0	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
CTMnC1	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
CTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnDH	—	—	—	—	—	—	D9	D8
CTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnAH	—	—	—	—	—	—	D9	D8

10-bit 简易型 TM 寄存器列表 (n=0 或 1)

CTMnDL 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 CTMn 计数器低字节寄存器 bit 7~bit 0
CTMn 10-bit 计数器 bit 7~bit 0

CTMnDH 寄存器

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

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

CTMnAL 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 CTMn CCRA 低字节寄存器 bit 7~bit 0
CTMn 10-bit CCRA bit 7~bit 0

CTMnAH 寄存器

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

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

CTMnC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
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 **CTnPAU**：CTMn 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，CTMn 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，从此值开始继续计数。

Bit 6~4 **CTnCK2~CTnCK0**：选择 CTMn 计数时钟位
000：$f_{SYS}/4$
001：f_{SYS}
010：$f_H/16$
011：$f_H/64$
100：f_{TBC}
101：保留位
110：CTCKn 上升沿时钟
111：CTCKn 下降沿时钟
此三位用于选择 CTMn 的时钟源。选择保留时钟输入将有效地除能内部计数器。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_{TBC} 是其它的内部时钟源，细节方面请参考振荡器章节。

Bit 3 **CTnON**：CTMn 计数器 On/Off 控制位
0：Off
1：On
此位控制 CTMn 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 CTMn。清零此位将停止计数器并关闭 CTM 减少耗电。当此位经由低到高转换时，内部计数器将复位清零，当此位由高到低转换时，内部计数器将保持其剩余值。
若 CTMn 处于比较匹配输出模式时，当 CTnON 位经由低到高转换时，CTM 输出脚将复位至 CTnOC 位指定的初始值。

Bit 2~0 **CTnRP2~CTnRP0**：CTMn CCRP 3-bit 寄存器，与 CTMn 计数器 bit 9~bit 7 比较
000：1024 个 CTMn 时钟周期
001：128 个 CTMn 时钟周期
010：256 个 CTMn 时钟周期
011：384 个 CTMn 时钟周期
100：512 个 CTMn 时钟周期
101：640 个 CTMn 时钟周期
110：768 个 CTMn 时钟周期
111：896 个 CTMn 时钟周期
此三位设定内部 CCRP 3-bit 寄存器的值，然后与内部计数器的高三位进行比较。如果 CTnCCLR 位设定为 0 时，比较结果为 0 并清除内部计数器。CTnCCLR 位设为低，内部计数器在比较器 P 比较匹配发生时被重置；由于 CCRP 只与计数器高三位比较，比较结果是 128 时钟周期的倍数。CCRP 被清零时，实际上会使得计数器在最大值溢出。

CTMnC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
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 **CTnM1~CTnM0**：选择 CTMn 工作模式位
00：比较匹配输出模式
01：未定义
10：PWM 模式
11：定时 / 计数器模式
这两位设置 CTMn 需要的工作模式。为了确保操作可靠，CTMn 应在 CTnM1 和 CTnM0 位有任何改变前先关掉。在定时 / 计数器模式，CTMn 输出脚控制必须除能。

Bit 5~4 **CTnIO1~CTnIO0**：选择 CTMn 外部引脚 CTPn 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
00：PWM 输出无效状态
01：PWM 输出有效状态
10：PWM 输出
11：未定义
定时 / 计数器模式
未使用
此两位用于决定在一定条件达到时 CTMn 输出脚如何改变状态。这两位值的选择决定 CTMn 运行在哪种模式下。
在比较匹配输出模式下，CTnIO1 和 CTnIO0 位决定当比较器 A 比较匹配输出发生时 CTMn 输出脚如何改变状态。当比较器 A 比较匹配输出发生时 CTMn 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。CTMn 输出脚的初始值通过 CTMnC1 寄存器的 CTnOC 位设置取得。注意，由 CTnIO1 和 CTnIO0 位得到的输出电平必须与通过 CTnOC 位设置的初始值不同，否则当比较匹配发生时，CTMn 输出脚将不会发生变化。在 CTMn 输出脚改变状态后，通过 CTnON 位由低到高电平的转换复位至初始值。
在 PWM 模式，CTnIO1 和 CTnIO0 用于决定比较匹配条件发生时怎样改变 CTMn 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 CTMn 关闭时改变 CTnIO1 和 CTnIO0 位的值是很有必要的。若在 CTMn 运行时改变 CTnIO1 和 CTnIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **CTnOC**：CTMn CTPn 输出控制位
比较匹配输出模式
0：初始低
1：初始高
PWM 输出模式
0：低有效
1：高有效
这是 CTMn 输出脚输出控制位。它取决于 CTMn 此时正运行于比较匹配输出模式还是 PWM 模式。若 CTMn 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 CTMn 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 **CTnPOL**：CTPn 输出极性控制位
0：同相
1：反相
此位控制 CTPn 输出脚的极性。此位为高时 CTMn 输出脚反相，为低时 CTMn 输出脚同相。若 CTMn 处于定时 / 计数器模式时其不受影响。

Bit 1 **CTnDPX**：CTMn 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 **CTnCCLR**：选择 CTMn 计数器清零条件位
0：CTMn 比较器 P 匹配
1：CTMn 比较器 A 匹配
此位用于选择清除计数器的方法。简易型 TM 包括两个比较器 - 比较器 A 和比较器 P。这两个比较器每个都可以用作清除内部计数器。CTnCCLR 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。CTnCCLR 位在 PWM 模式时未使用。

简易型 TM 工作模式

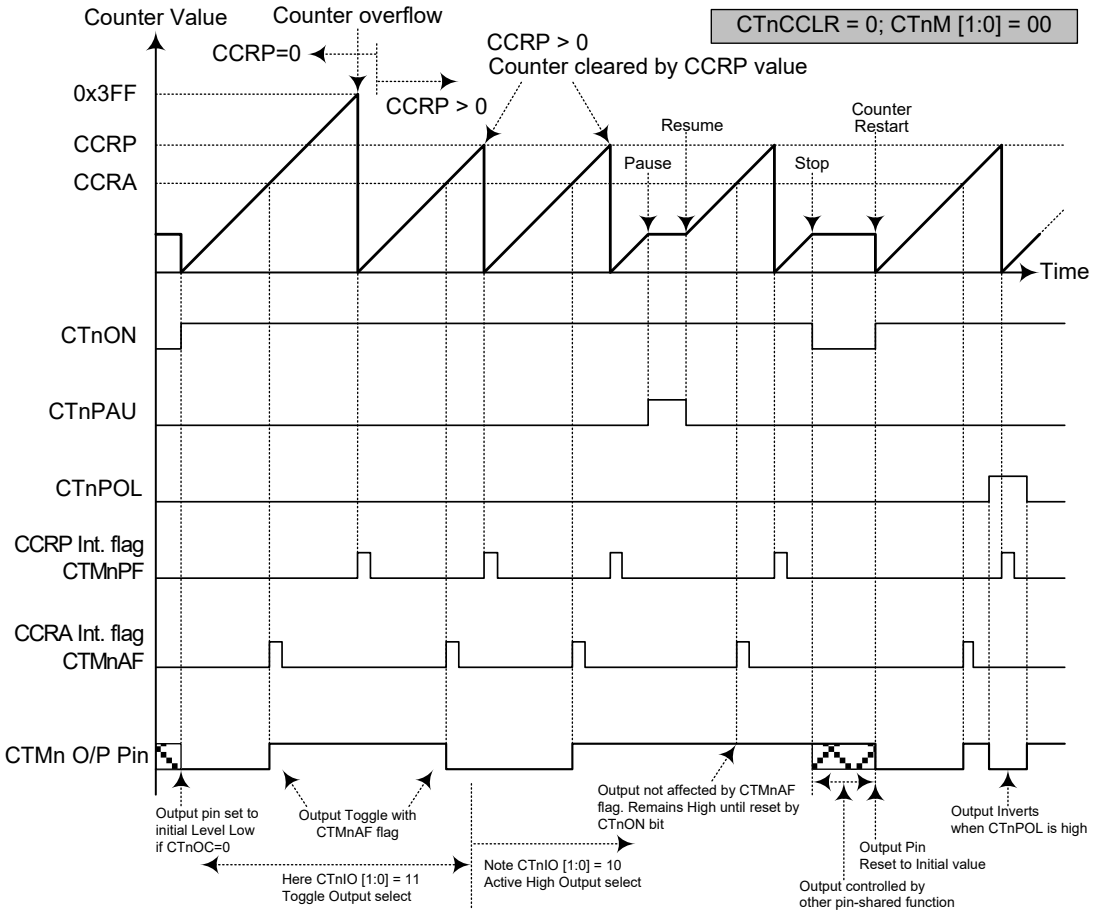
简易型 TM 有三种工作模式，即比较匹配输出模式，PWM 模式或定时 / 计数器模式。通过设置 CTMnC1 寄存器的 CTnM1 和 CTnM0 位选择任意工作模式。

比较匹配输出模式

为使 CTMn 工作在此模式，CTMnC1 寄存器中的 CTnM1 和 CTnM0 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 CTnCCLR 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 CTMnAF 和 CTMnPF 将分别置起。

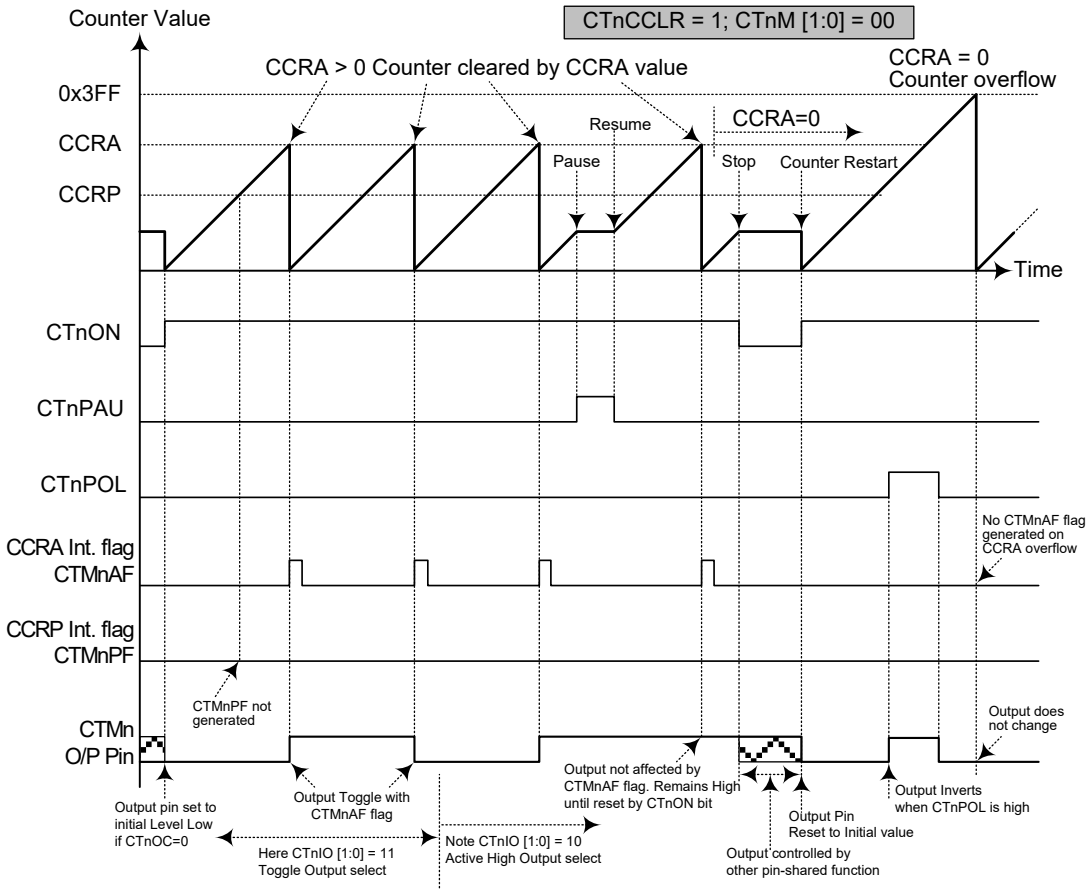
如果 CTMnC1 寄存器的 CTnCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅 CTMnAF 中断请求标志产生。所以当 CTnCCLR 为高时，不产生 CTMnPF 中断请求标志。如果 CCRA 被清零，当计数达到最大值 3FFH 时，计数器溢出，而此时不产生 CTMnAF 请求标志。

正如该模式名所言，当比较匹配发生后，CTMn 输出脚状态改变。当比较器 A 比较匹配发生后 CTMnAF 标志产生时，CTMn 输出脚状态改变。比较器 P 比较匹配发生时产生的 CTMnPF 标志不影响 CTMn 输出脚。CTMn 输出脚状态改变方式由 CTMnC1 寄存器中 CTnIO1 和 CTnIO0 位决定。当比较器 A 比较匹配发生时，CTnIO1 和 CTnIO0 位决定 TM 输出脚输出高，低或翻转当前状态。CTMn 输出脚初始值，在 CTnON 位由低到高电平的变化后通过 CTnOC 位设置。注意，若 CTnIO1 和 CTnIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 -- CTnCCLR=0

注：1. CTnCCLR=0，比较器 P 匹配将清除计数器
2. CTMn 输出脚仅由 CTMnAF 标志位控制
3. 在 CTnON 上升沿 TM 输出脚复位至初始值
4. n=0 或 1



比较匹配输出模式 -- CTnCCLR=1

注：1. CTnCCLR=1，比较器 A 匹配将清除计数器
2. CTMn 输出脚仅由 CTMnAF 标志位控制
3. 在 CTnON 上升沿 TM 输出脚复位至初始值
4. 当 CTnCCLR=1 时，CTMnPF 标志位不会产生
5. n=0 或 1

定时 / 计数器模式

为使 CTMn 工作在此模式，CTMnC1 寄存器中的 CTnM1 和 CTnM0 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 CTMn 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 CTMn 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 CTMn 工作在此模式，CTMnC1 寄存器中的 CTnM1 和 CTnM0 位需要设置为“10”。CTMn 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 CTMn 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，CTnCCLR 位不影响 PWM 操作。CCRA 和 CCRP 寄存器决定 PWM 波形，一个用来清除内部计数器并控制 PWM 波形的频率，另一个用来控制占空比。哪个寄存器控制频率或占空比取决于 CTMnC1 寄存器的 CTnDPX 位。所以 PWM 波形频率和占空比由 CCRA 和 CCRP 寄存器共同决定。

当比较器 A 或比较器 P 比较匹配发生时，将产生 CCRA 或 CCRP 中断标志。CTMnC1 寄存器中的 CTnOC 位决定 PWM 波形的极性，CTnIO1 和 CTnIO0 位使能 PWM 输出或将 CTMn 输出脚置为逻辑高或逻辑低。CTnPOL 位对 PWM 输出波形的极性取反。

- **10-bit CTMn，PWM 模式，边沿对齐模式，CTnDPX=0**

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

若 f_{SYS}=16MHz，CTMn 时钟源选择 f_{SYS}/4，CCRP=2，CCRA=128，

CTMn PWM 输出频率 =(f_{SYS}/4) / (2×256)= f_{SYS}/2048=7.8125kHz，

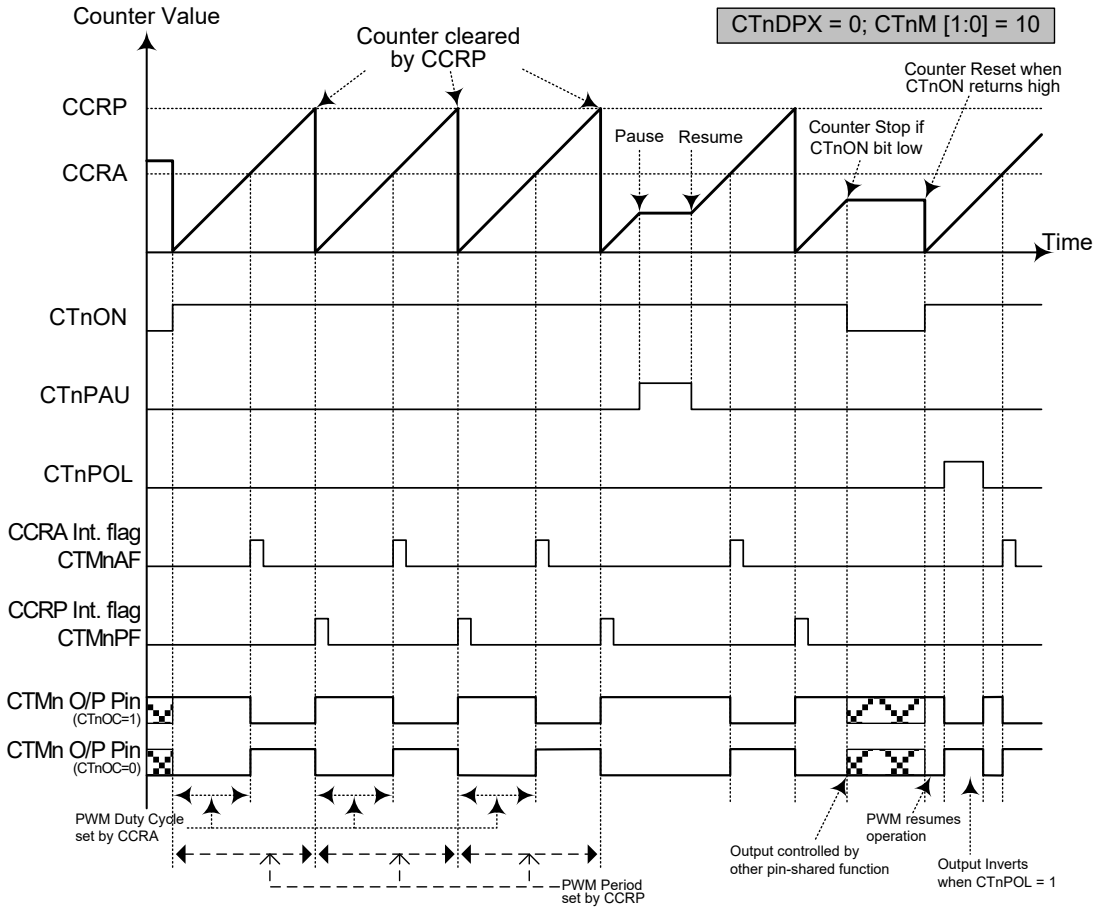
duty=128/(2×256)=25%

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%

- **10-bit CTMn，PWM 模式，边沿对齐模式，CTnDPX=1**

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

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



PWM 输出模式 -- CTnDPX=0

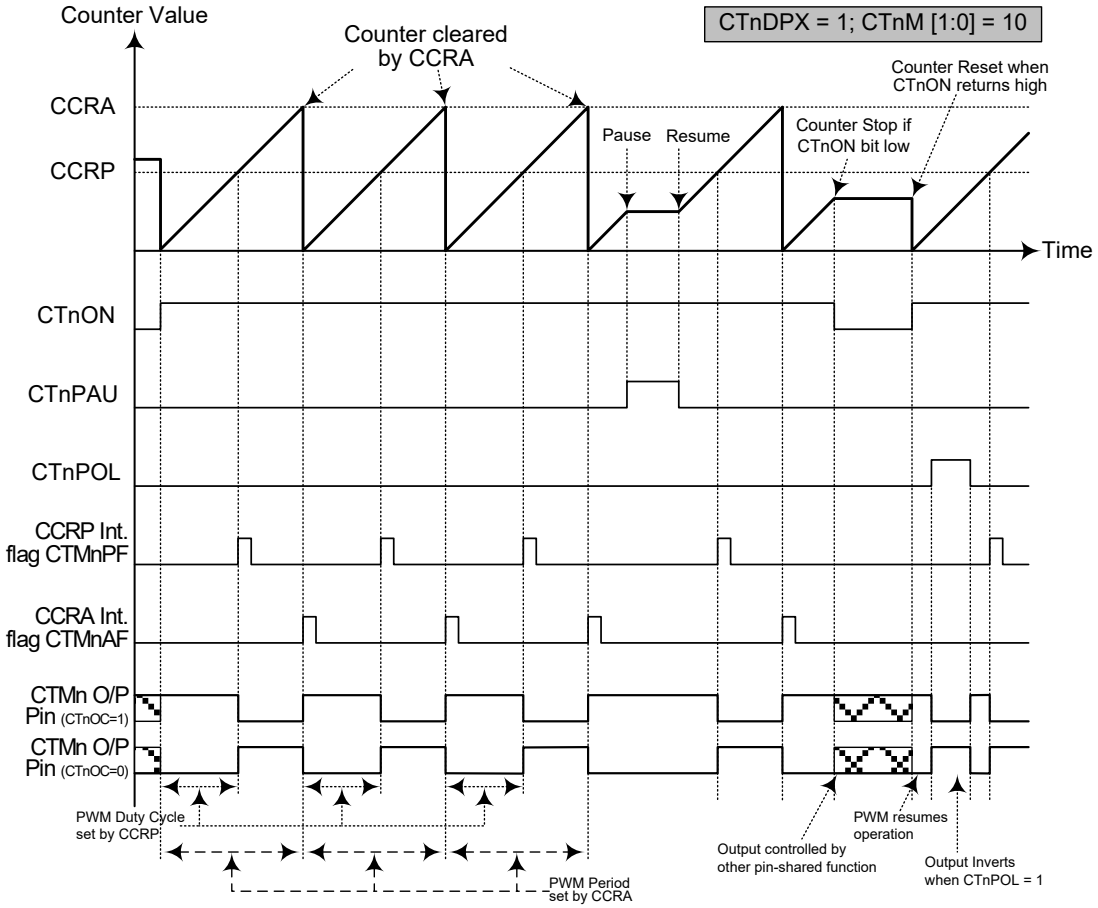
注：1. CTnDPX=0，CCRP 清除计数器

2. 计数器清零并设置 PWM 周期

3. 当 CTnIO1，CTnIO0=00 或 01，PWM 功能不变

4. CTnCCLR 位不影响 PWM 操作

5. n=0 或 1



PWM 输出模式 -- CTnDPX=1

注：1. CTnDPX=1，CCRA 清除计数器

2. 计数器清零并设置 PWM 周期

3. 当 CTnIO1，CTnIO0=00 或 01，PWM 功能不变

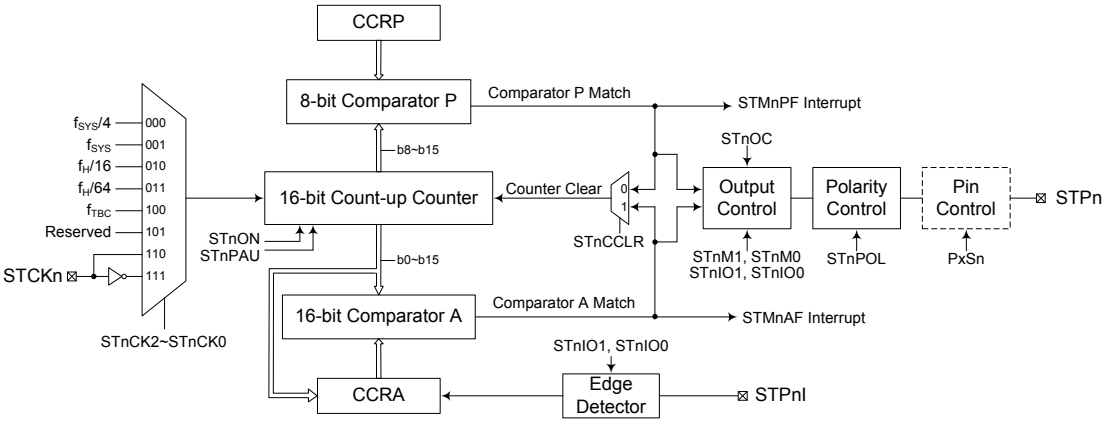
4. CTnCCLR 位不影响 PWM 操作

5. n=0 或 1

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	TM 类型	TM 输入引脚	TM 输出引脚
HT67F60A HT67F70A	16-bit STM (STM0, STM1, STM2)	STCK0, STCK1, STCK2; STP0I, STP1I, STP2I	STP0, STP1, STP2



标准型 TM 框图 (n=0，1 或 2)

标准型 TM 操作

标准型 TM 是 16 位宽度。核心是一个由用户选择的内部或外部时钟源驱动的 16 位向上计数器，它还包括两个内部比较器即比较器 A 和比较器 P。这两个比较器将计数器的值与 CCRP 和 CCRA 寄存器中的值进行比较。CCRP 是 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 位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 STMn 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 的所有操作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=0，1 或 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_{TBC}
101：保留位
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_{TBC} 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。当此位经由低到高转换时，内部计数器将复位清零，当此位由高到低转换时，内部计数器将保持其剩余值。
若 STMn 处于比较匹配输出模式时，当 STnON 位经由低到高转换时，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 或 STPnI 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 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 输出脚如何改变状态。这两位值的选择决定 STMn 运行在哪种模式下。
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 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 STMn 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。STMn 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，STMn 输出脚将不会发生变化。在 STMn 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 模式，STnIO1 和 STnIO0 用于决定比较匹配条件发生时怎样改变 STMn 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 输出的值是无法预料的。

Bit 3 **STnOC**：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 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 **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 STMn 输出脚反相，为低时 STMn 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：比较器 P 匹配
1：比较器 A 匹配
此位用于选择清除计数器的方法。标准型 STMn 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，对应于 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，比较结果为 0 并清除内部计数器。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

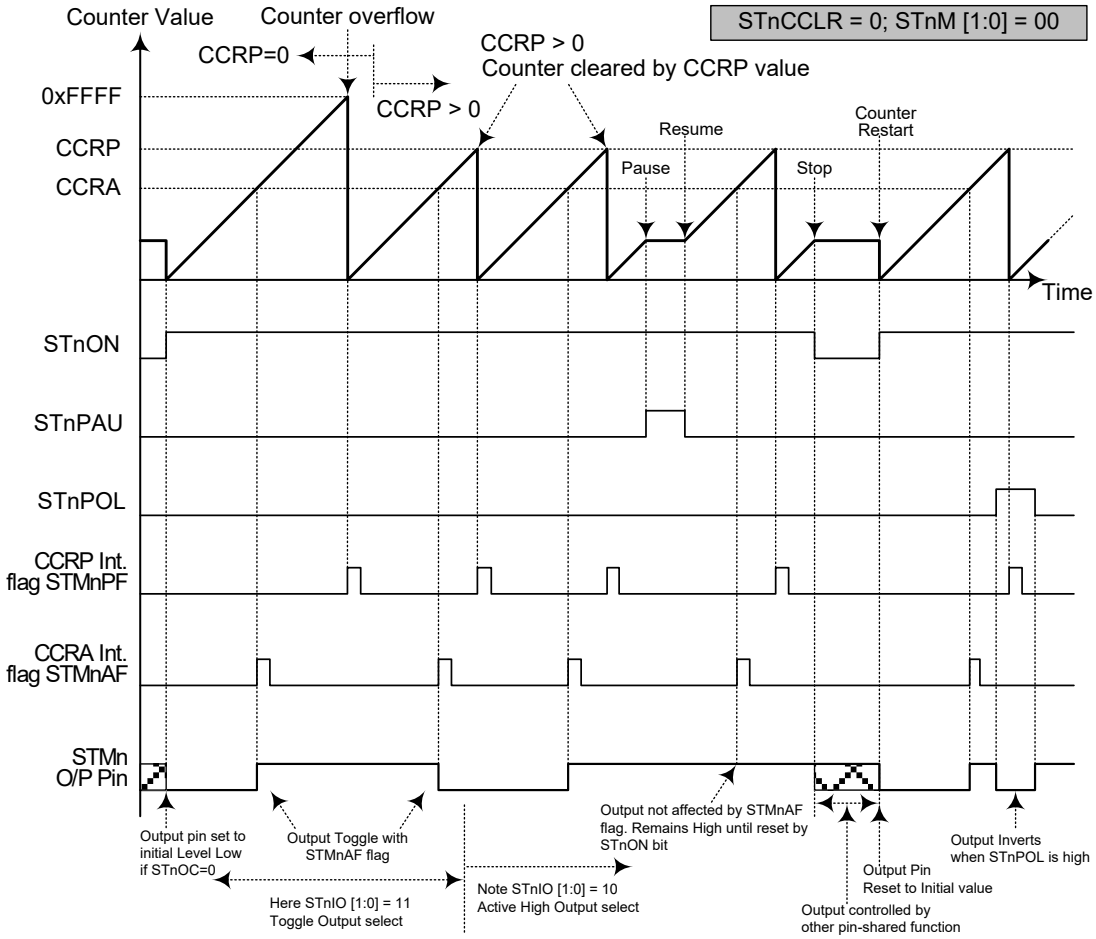
标准型 TM 有五种工作模式，即比较匹配输出模式，PWM 输出模式，单脉冲输出模式，捕捉输入模式或定时 / 计数器模式。通过设置 STMnC1 寄存器的 STnM1 和 STnM0 位选择任意模式。

比较匹配输出模式

为使 STMn 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。

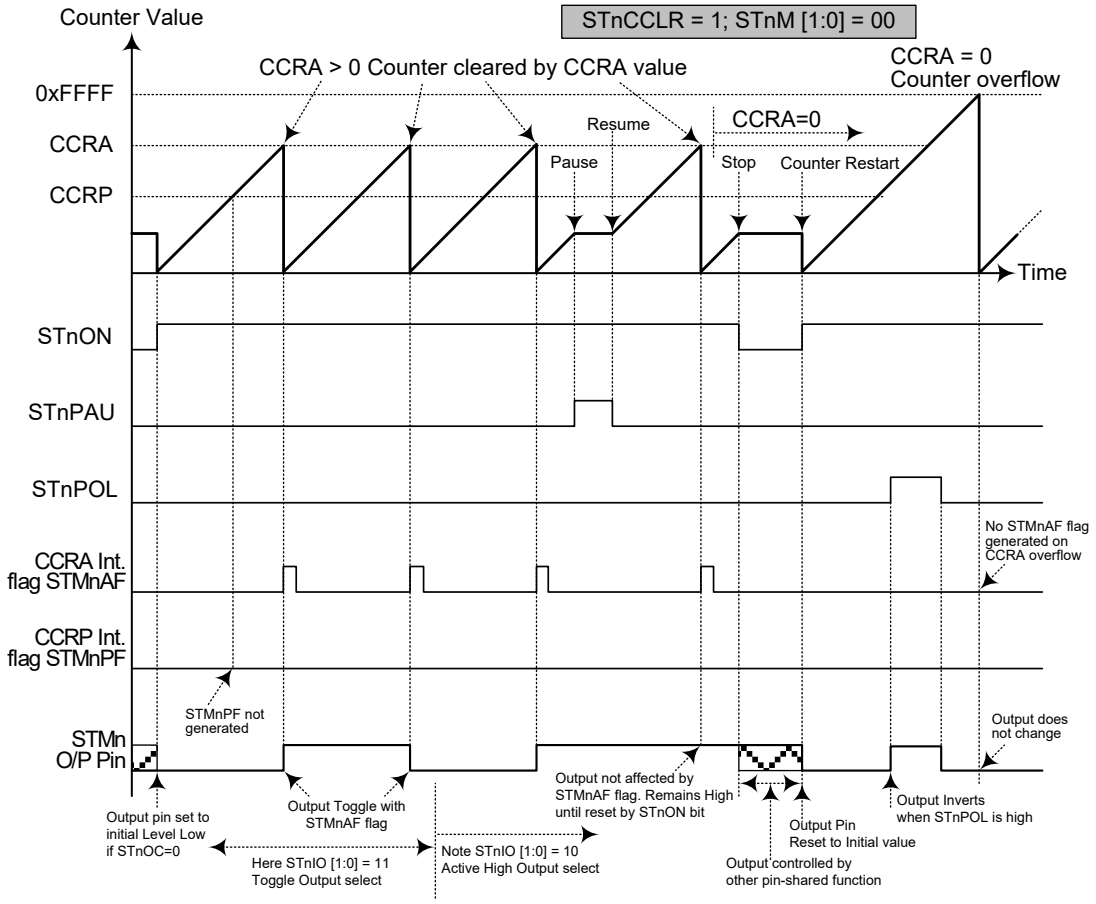
如果 STMnC1 寄存器的 STnCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅产生 STMnAF 中断请求标志。所以当 STnCCLR 为高时，不会产生 STMnPF 中断请求标志。在比较匹配输出模式下，CCRA 不能设为“0”。

正如该模式名所言，当比较匹配发生后，STMn 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，STMn 输出脚状态改变。比较器 P 比较匹配发生时产生的 STMnPF 标志不影响 STMn 输出脚。STMn 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 STMn 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, 1 或 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 STM 输出脚复位至初始值
4. 当 STnCCLR=1 时，不会产生 STMnPF 标志
5. n=0, 1 或 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 模式中，STnCCLR 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 STMn 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 模式，边沿对齐模式，STnDPX=0**

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

若 f_{SYS}=16MHz，STMn 时钟源选择 f_{SYS}/4，CCRP=2，CCRA=128，

STMn PWM 输出频率 =(f_{SYS}/4) / (2×256)= f_{SYS}/2048=7.8125kHz，

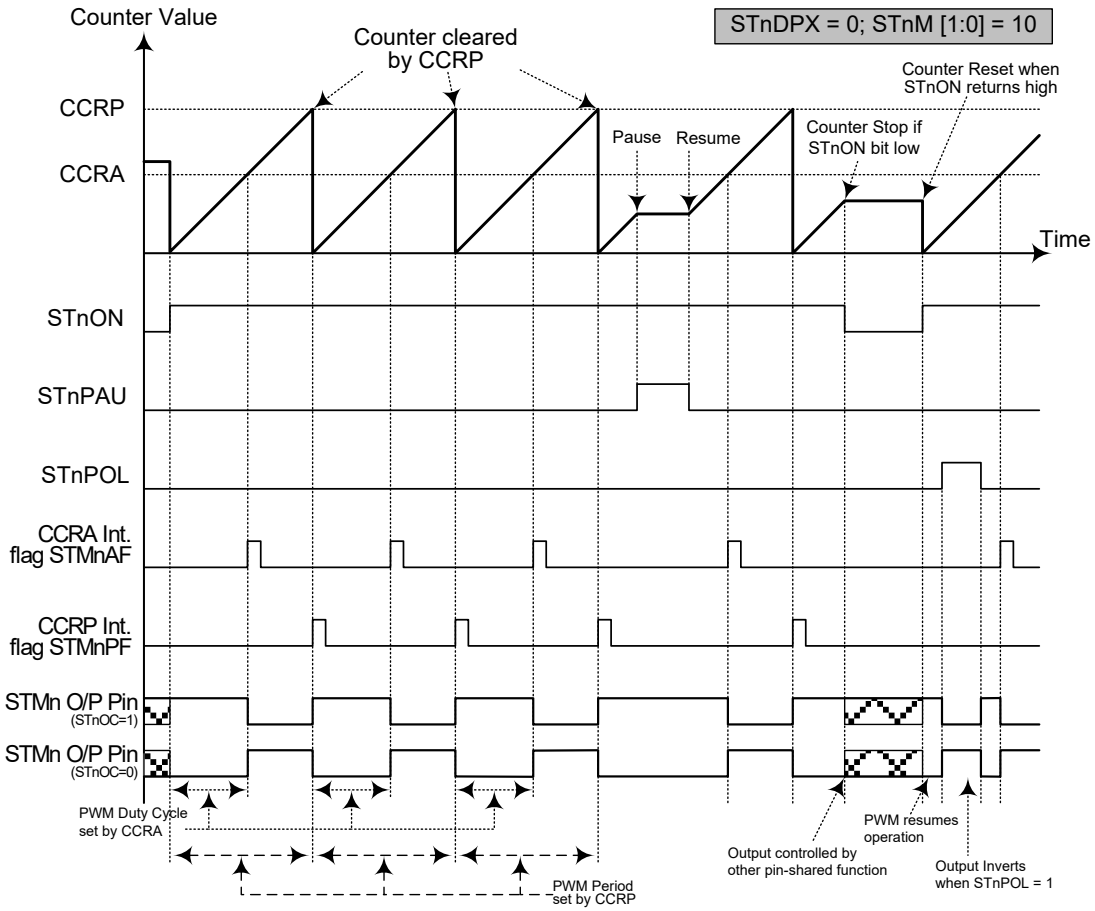
duty=128/(2×256)=25%

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 模式，边沿对齐模式，STnDPX=1**

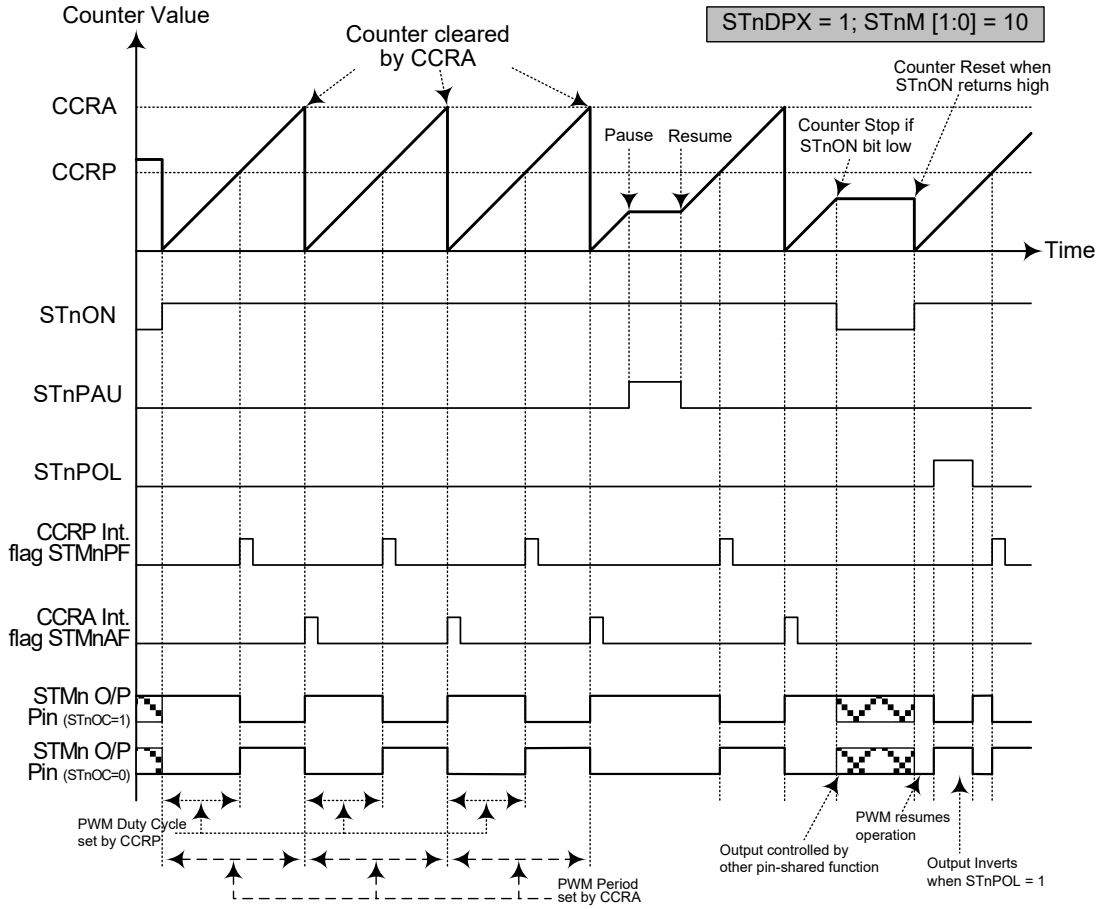
CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

PWM 的输出周期由 CCRA 寄存器的值与 STMn 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, 1 或 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

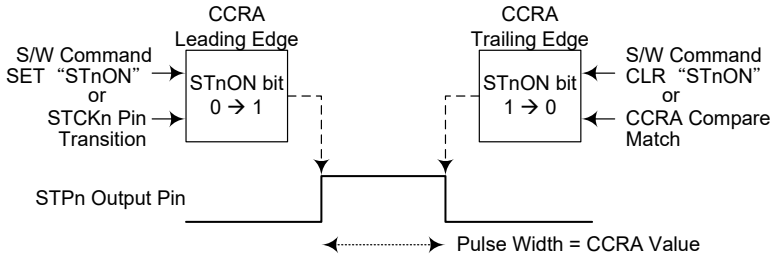
5. n=0, 1 或 2

单脉冲输出模式

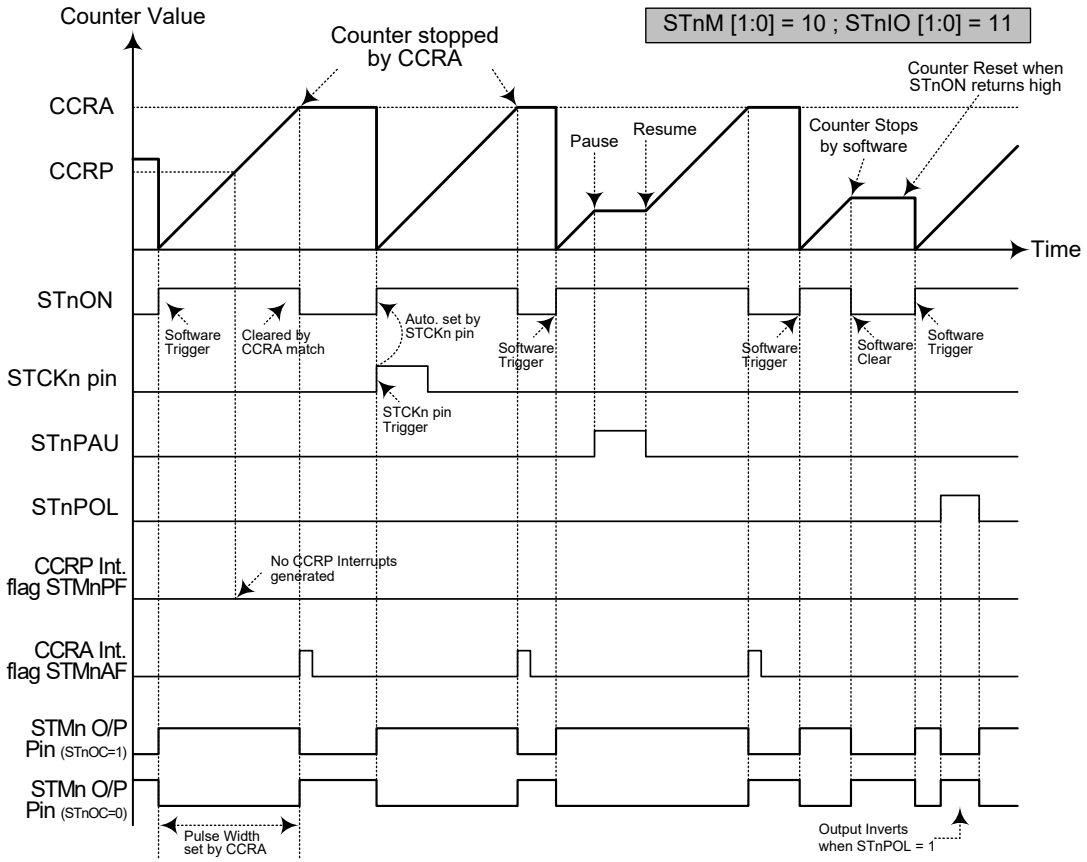
为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 位需要设置为“10”，同时 STnIO1 和 STnIO0 位需要设置为“11”。正如模式名所言，单脉冲输出模式，在 STMn 输出脚将产生一个脉冲输出。

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。而处于单脉冲模式时，STnON 位在 STCKn 脚发生有效边沿跳转时自动由低转变为高，进而开始单脉冲输出。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边沿跳变。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

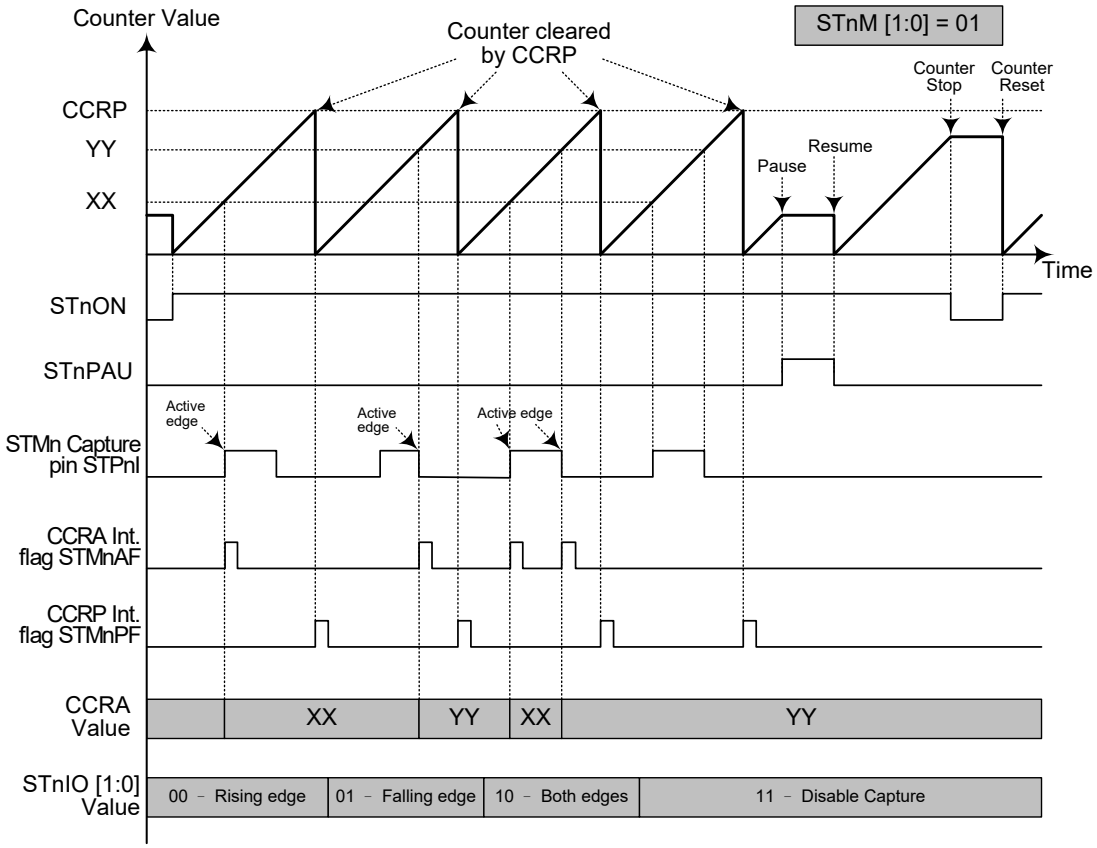
注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 STCKn 脚或设置 STnON 位为高来触发脉冲
4. STCKn 脚有效沿会自动置位 STnON
5. 单脉冲模式中，STnIO[1:0] 需置位“11”，且不能更改。
6. n=0, 1 或 2

捕捉输入模式

为使 STMn 工作在此模式，STMnC1 寄存器中的 STnM1 和 STnM0 位需要设置为“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 位由低到高转变时，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。无论 STPnI 引脚发生哪种边沿转换，计数器将继续工作直到 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 位都设置为高，无论 STPnI 引脚发生哪种边沿转换都不会产生捕捉操作，但计数器将会继续运行。

STnCCLR 和 STnDPX 位在此模式中未使用。



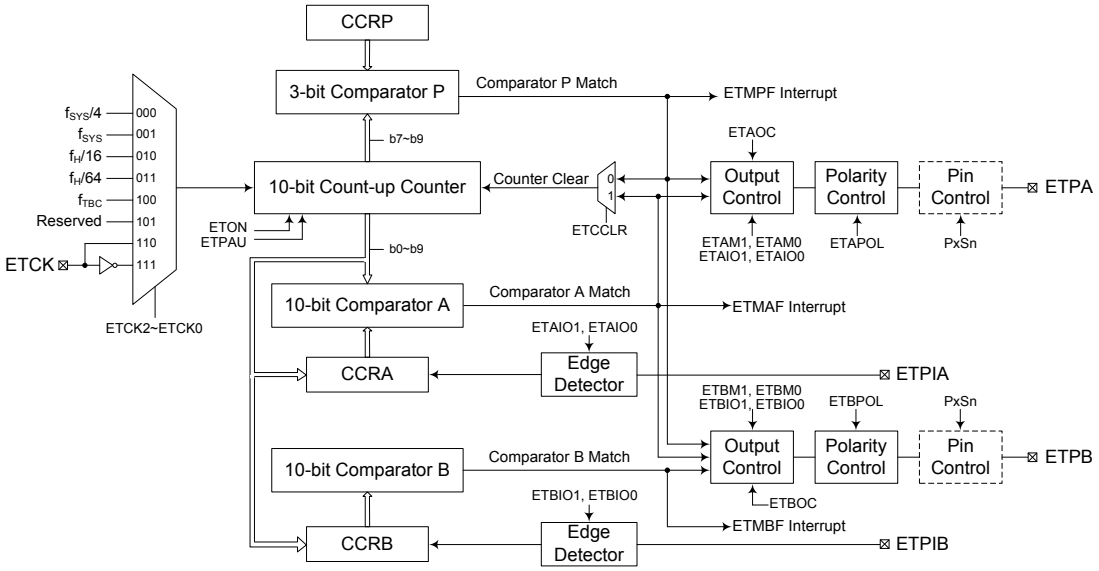
捕捉输入模式

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, 1 或 2

增强型 TM – ETM

增强型 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。增强型 TM 也由三个外部输入脚控制并驱动两个外部输出脚。

单片机	TM 类型	TM 输入引脚	TM 输出引脚
HT67F60A HT67F70A	10-bit ETM (ETM)	ETCK; ETPIA, ETPIB	ETPA, ETPB



增强型 TM 方框图

增强型 TM 操作

增强型 TM 核心是一个由用户选择的内部或外部时钟源驱动的 10 位向上 / 向下计数器，它还包括三个内部比较器即比较器 A，比较器 B 和比较器 P。这三个比较器将计数器的值与 CCRA，CCRB 和 CCRP 寄存器中的值进行比较。CCRP 是 3 位的，与计数器的高 3 位比较；而 CCRA 和 CCRB 是 10 位的，与计数器的所有位比较。

通过应用程序改变 10 位计数器值的唯一方法是使 ETON 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 ETM 中断信号。增强型 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。

增强型 TM 寄存器介绍

增强型 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 位计数器的值，两对读 / 写寄存器存放 10 位 CCRA 和 CCRB 的值。剩下三个控制寄存器用来设置不同的操作和控制模式，以及 CCRP 的 3 个位。

寄存器名称	位							
	7	6	5	4	3	2	1	0
ETMC0	ETPAU	ETCK2	ETCK1	ETCK0	ETON	ETRP2	ETRP1	ETRP0
ETMC1	ETAM1	ETAM0	ETAIO1	ETAIO0	ETAOC	ETAPOL	ETCDN	ETCCLR
ETMC2	ETBM1	ETBM0	ETBIO1	ETBIO0	ETBOC	ETBPOL	ETPWM1	ETPWM0
ETMDL	D7	D6	D5	D4	D3	D2	D1	D0
ETMDH	—	—	—	—	—	—	D9	D8
ETMAL	D7	D6	D5	D4	D3	D2	D1	D0
ETMAH	—	—	—	—	—	—	D9	D8
ETMBL	D7	D6	D5	D4	D3	D2	D1	D0
ETMBH	—	—	—	—	—	—	D9	D8

10-bit 增强型 TM 寄存器列表

ETMDL 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 ETM 计数器低字节寄存器 bit 7~bit 0
TM1 10-bit 计数器 bit 7~bit 0

ETMDH 寄存器

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

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

ETMAL 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 ETM CCRA 低字节寄存器 bit 7~bit 0
ETM 10-bit CCRA bit 7~bit 0

ETMAH 寄存器

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

Bit 7~2 未定义，读为“0”

Bit 1~0 ETM CCRA 高字节寄存器 bit 1~bit 0
ETM 10-bit CCRA bit 9~bit 8

ETMBL 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 ETM CCRB 低字节寄存器 bit 7~bit 0
ETM 10-bit CCRB bit 7~bit 0

ETMBH 寄存器

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

Bit 7~2 未定义，读为“0”

Bit 1~0 ETM CCRB 高字节寄存器 bit 1~bit 0
ETM 10-bit CCRB bit 9~bit 8

ETMC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ETPAU	ETCK2	ETCK1	ETCK0	ETON	ETRP2	ETRP1	ETRP0
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 **ETPAU**：ETM 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，ETM 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 **ETCK2~ETCK0**：选择 ETM 计数时钟位
000：$f_{SYS}/4$
001：f_{SYS}
010：$f_H/16$
011：$f_H/64$
100：f_{TBC}
101：保留位
110：ETCK 上升沿时钟
111：ETCK 下降沿时钟

此三位用于选择 ETM 的时钟源。选择保留时钟输入将有效地除能内部计数器。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_{TBC} 是其它的内部时钟源，细节方面请参考振荡器章节。

Bit 3 **ETON**：ETM 计数器 On/Off 控制位
0：Off
1：On
此位控制 ETM 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 ETM。清零此位将停止计数器并关闭 ETM 减少耗电。当此位经由低到高转换时，内部计数器将复位清零，当此位由高到低转换时，内部计数器将保持其剩余值。
若 ETM 处于比较匹配输出模式时，当 ETON 位经由低到高转换时，ETM 输出脚将复位至 ETOC 位指定的初始值。

Bit 2~0 **ETRP2~ETRP0**：ETM CCRP 3-bit 寄存器，与 ETM 计数器 bit 9~bit 7 比较
000：1024 个 ETM 时钟周期
001：128 个 ETM 时钟周期
010：256 个 ETM 时钟周期
011：384 个 ETM 时钟周期
100：512 个 ETM 时钟周期
101：640 个 ETM 时钟周期
110：768 个 ETM 时钟周期
111：896 个 ETM 时钟周期
此三位设定内部 CCRP 3-bit 寄存器的值，然后与内部计数器的高三位进行比较。如果 ETCCLR 位设定为 0 时，比较结果为 0 并清除内部计数器。ETCCLR 位设为低，内部计数器在比较器 P 比较匹配发生时被重置；由于 CCRP 只与计数器高三位比较，比较结果是 128 时钟周期的倍数。CCRP 被清零时，实际上会使得计数器在最大值溢出。

ETMC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ETAM1	ETAM0	ETAIO1	ETAIO0	ETAOC	ETAPOL	ETCDN	ETCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **ETAM1~ETAM0**：选择 ETM CCRA 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 ETM 需要的工作模式。为了确保操作可靠，ETM 应在 ETAM1 和 ETAM0 位有任何改变前先关掉。在定时 / 计数器模式，ETM 输出脚控制必须除能。

Bit 5~4 **ETAIO1~ETAIO0**：选择 ETM 外部引脚 ETPA 或 ETPIA 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 ETPIA 上升沿输入捕捉
01：在 ETPIA 下降沿输入捕捉
10：在 ETPIA 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 ETM ETPA 输出脚如何改变状态。这两位值的选择决定 ETM 运行在哪种模式下。

在比较匹配输出模式下，ETAIO1 和 ETAIO0 位决定当从比较器 A 比较匹配输出发生时 ETM 输出脚 ETPA 如何改变状态。当从比较器 A 比较匹配输出发生时 ETM 输出脚能设为切换高、切换低或翻转当前状态。若此两位同时为零时，这个输出将不会改变。ETM 输出脚的初始值通过 ETMC1 寄存器的 ETAOC 位设置取得。注意，由 ETAIO1 和 ETAIO0 位得到的输出电平必须与通过 ETAOC 位设置的初始值不同，否则当比较匹配发生时，ETM 输出脚将不会发生变化。在 ETM 输出脚改变状态后，通过 ETON 位由低到高电平的转换复位至初始值。

在 PWM 模式，ETAIO1 和 ETAIO0 用于决定比较匹配条件发生时怎样改变 ETM 输出脚 ETPA 的状态。PWM 输出功能通过这两位的变化进行更新。仅在 ETM 关闭时改变 ETAIO1 和 ETAIO0 位的值是很有必要的。若在 ETM 运行时改变 ETAIO1 和 ETAIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **ETAOC**：ETPA 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 ETM 输出脚 ETPA 输出控制位。它取决于 ETM 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 ETM 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 ETM 输出脚的逻辑电平值。在 PWM 输出模式 / 单脉冲模式时，其决定 PWM 信号是高有效还是低有效。

Bit 2 **ETAPOL**：ETPA 输出极性控制位
0：同相
1：反相

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

Bit 1 **ETCDN**：ETM 计数器向上 / 向下计数标志位
0：向上计数
1：向下计数

Bit 0 **ETCCLR**：选择 ETM 计数器清零条件位
0：ETM 比较器 P 匹配
1：ETM 比较器 A 匹配

此位用于选择清除计数器的方法。增强型 TM 包括三个比较器 -- 比较器 P、比较器 A 和比较器 B，其中比较器 P 和比较器 A 都可以用作清除内部计数器。ETCCLR 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。ETCCLR 位在 PWM 输出，单脉冲或输入捕捉模式时未使用。

ETMC2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	ETBM1	ETBM0	ETBIO1	ETBIO0	ETBOC	ETBPOL	ETPWM1	ETPWM0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **ETBM1~ETBM0**：选择 ETM CCRB 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 ETM CCRB 需要的工作模式。为了确保操作可靠，ETM 应在 ETBM1 和 ETBM0 位有任何改变前先关掉。在定时 / 计数器模式，ETM 输出脚控制必须除能。

Bit 5~4 **ETBIO1~ETBIO0**：选择 ETPB 或 ETPIB 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 ETPIB 上升沿输入捕捉
01：在 ETPIB 下降沿输入捕捉
10：在 ETPIB 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 ETM 输出脚 ETPB 如何改变状态。这两位值的选择决定 ETM 运行在哪种模式下。

在比较匹配输出模式下，ETBIO1 和 ETBIO0 位决定当比较器 B 比较匹配输出发生时 ETM 输出脚如何改变状态。当比较器 B 比较匹配输出发生时 ETM 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。ETM 输出脚的初始值通过 ETMC2 寄存器的 ETBOC 位设置取得。注意，由 ETBIO1 和 ETBIO0 位得到的输出电平必须与通过 ETBOC 位设置的初始值不同，否则当比较匹配发生时，ETM 输出脚将不会发生变化。在 ETM 输出脚改变状态后，通过 ETON 位由低到高电平的转换复位至初始值。

在 PWM 模式，ETBIO1 和 ETBIO0 用于决定比较匹配条件发生时怎样改变 ETM 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 ETM 关闭时改变 ETBIO1 和 ETBIO0 位的值是很有必要的。若在 ETM 运行时改变 ETBIO1 和 ETBIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **ETBOC**：ETPB 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 ETM 输出脚 ETPB 输出控制位。它取决于 ETM 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 ETM 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 ETM 输出脚的逻辑电平值。在 PWM 输出模式 / 单脉冲模式时，其决定 PWM 信号是高有效还是低有效。

Bit 2 **ETBPOL**：ETPB 输出极性控制位
0：同相
1：反相
此位控制 ETPB 输出脚的极性。此位为高时 ETM 输出脚 ETPB 反相，为低时 ETM 输出脚同相。若 ETM 处于定时 / 计数器模式时其不受影响。

Bit 1~0 **ETPWM1~ETPWM0**：选择 ETM PWM 模式位
00：边沿对齐
01：中心对齐，向上计数比较匹配
10：中心对齐，向下计数比较匹配
11：中心对齐，向上 / 下计数比较匹配

增强型 TM 工作模式

增强型 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 ETMC1 寄存器的 ETAM1 和 ETAM0 位和 ETMC2 寄存器的 ETBM1 和 ETBM0 位选择任意模式。

ETM 工作模式	CCRA 比较匹配输出模式	CCRA 定时 / 计数器模式	CCRA PWM 输出模式	CCRA 单脉冲输出模式	CCRA 输入捕捉模式
CCRB 比较匹配输出模式	√	—	—	—	—
CCRB 定时 / 计数器模式	—	√	—	—	—
CCRB PWM 输出模式	—	—	√	—	—
CCRB 单脉冲输出模式	—	—	—	√	—
CCRB 输入捕捉模式	—	—	—	—	√

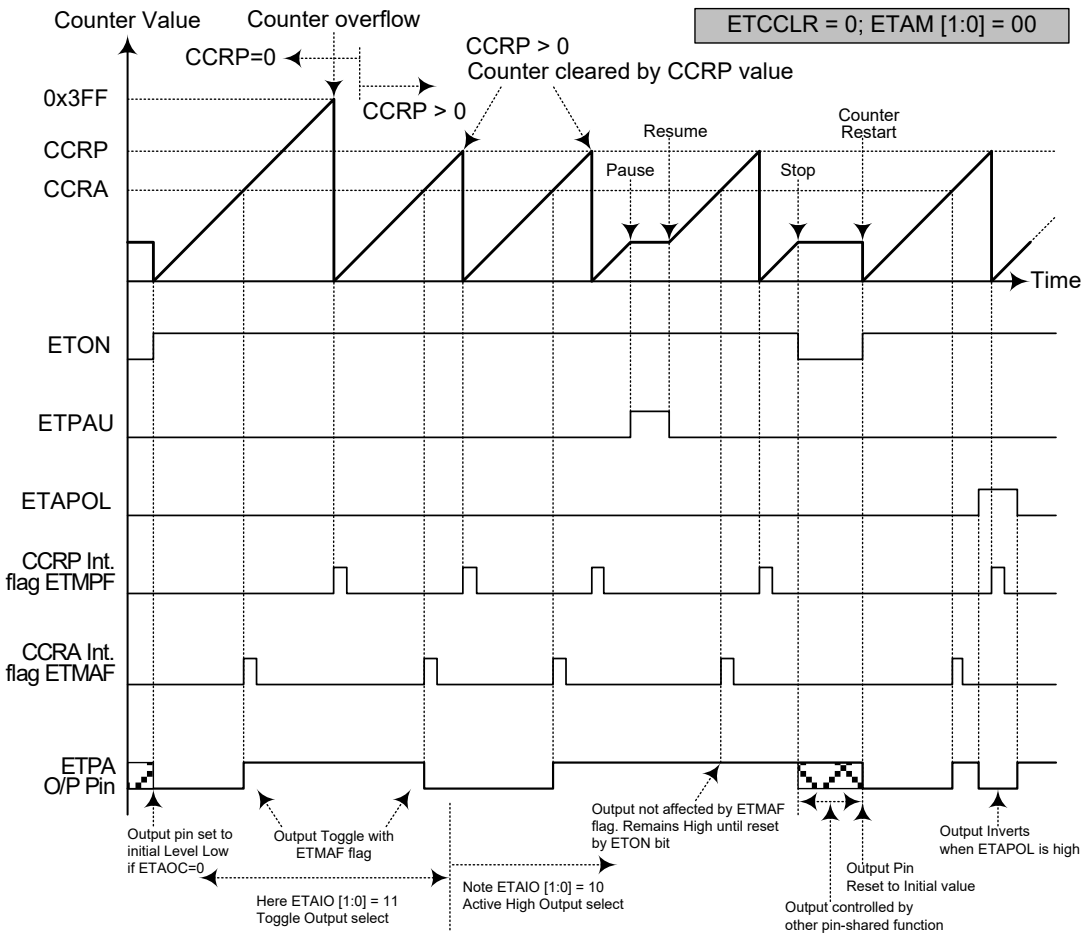
“√”：允许，“—”：不允许

比较匹配输出模式

为使 ETM 工作在此模式，ETMC1 寄存器的 ETAM1，ETAM0 位和 ETMC2 寄存器的 ETBM1，ETBM0 位需要全部清零。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 ETCCLR 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 ETMAF 和 ETMPF 将分别置起。

如果 ETMC1 寄存器的 ETCCLR 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 ETMAF 中断请求标志产生。所以当 ETCCLR 为高时，不会产生 ETMPF 中断请求标志。在比较匹配输出模式中，CCRA 不能被设置为“0”。

正如该模式名所言，当比较匹配发生后，ETM 输出脚状态改变。当比较器 A 或比较器 B 比较匹配发生后 ETMAF 或 ETMBF 中断请求标志产生时，ETM 输出脚状态改变。比较器 P 比较匹配发生时产生的 ETMPF 标志不影响 ETM 输出脚。ETM 输出脚状态改变方式由 ETM CCRA 的 ETMC1 寄存器中 ETMAIO1 和 ETMAIO0 位，ETM CCRB 的 ETMC2 寄存器中的 ETMBIO1 和 ETMBIO0 位决定。当比较器 A 或比较器 B 比较匹配发生时，ETMAIO1，ETMAIO0 位 (对于 ETPA 引脚) 和 ETMBIO1，ETMBIO0 位 (对于 ETPB 引脚) 决定 ETM 输出脚输出高，低或翻转当前状态。ETM 输出脚初始值，在 ETON 位由低到高电平的变化后由 ETAOC 或 ETBOC 位设置。注意，若 ETAIO1，ETAIO0 和 ETBIO1，ETBIO0 位同时为 0 时，引脚输出不变。

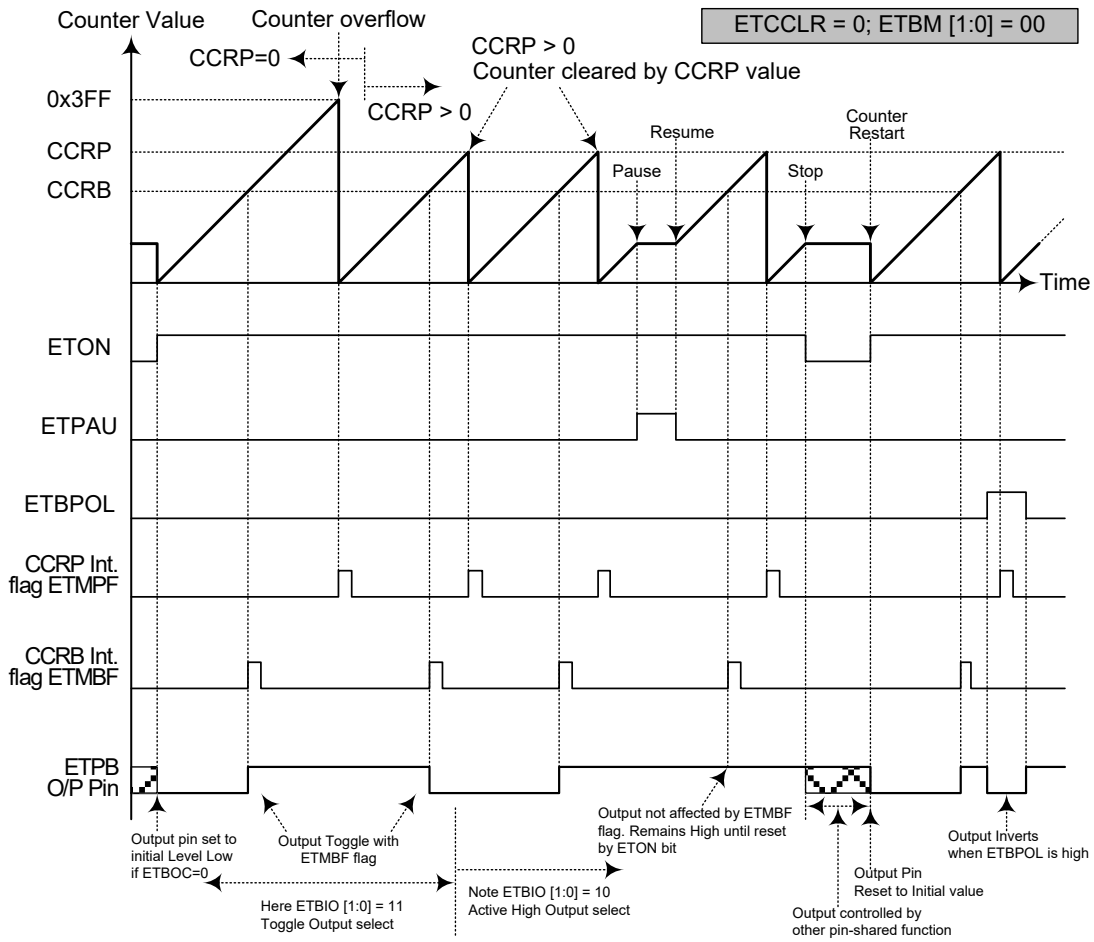


10-bit ETM CCRA 比较匹配输出模式 -- ETCCLR=0

注：1. ETCCLR=0，比较器 P 匹配将清除计数器

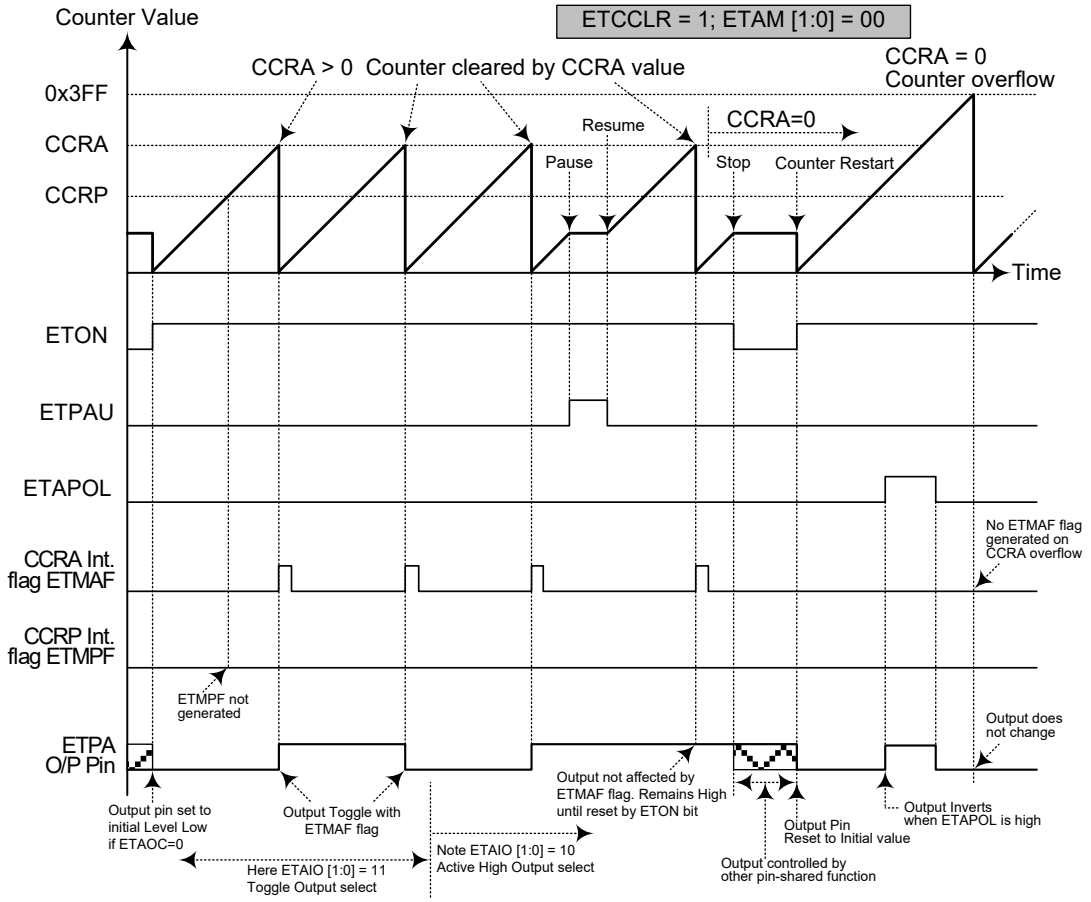
2. ETPA 输出脚仅由 ETMAF 标志位控制

3. 在 ETON 上升沿 ETM 输出脚复位至初始值



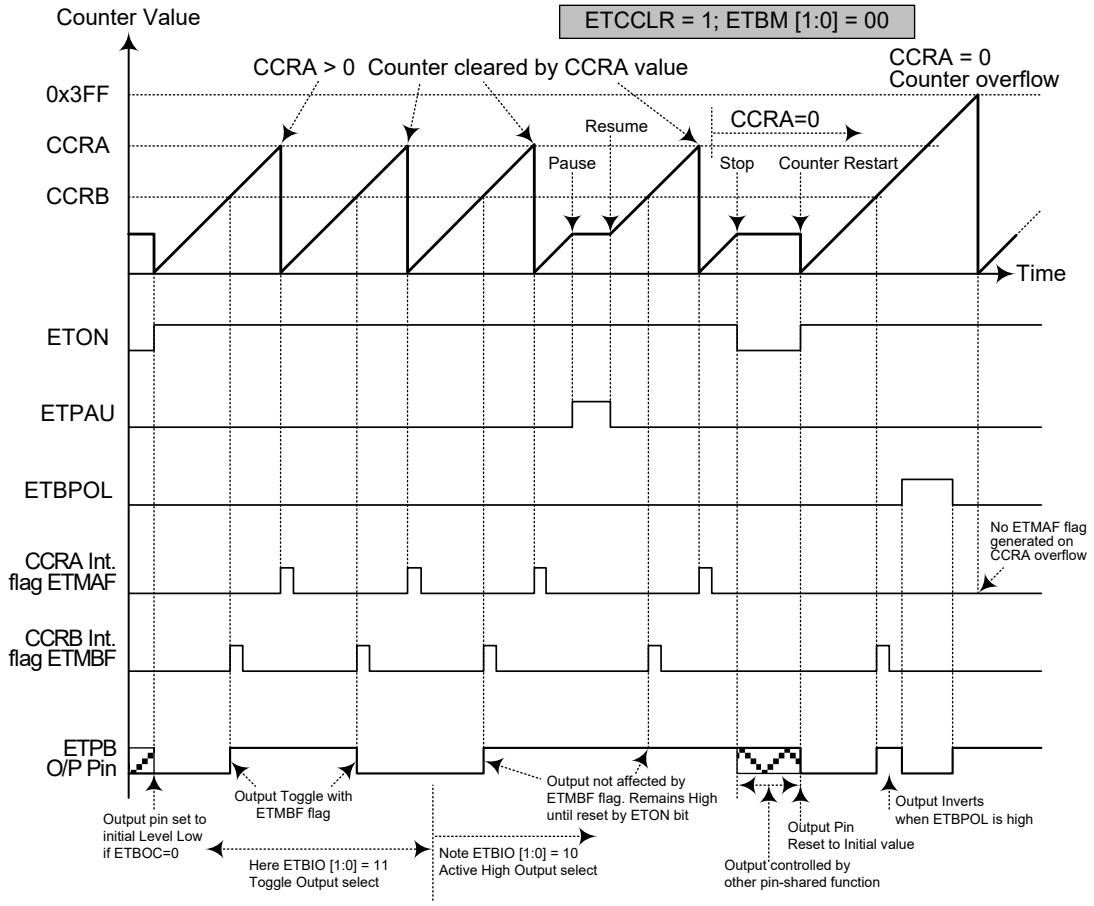
10-bit ETM CCRB 比较匹配输出模式 -- ETCCLR=0

注：1. ETCCLR=0，比较器 P 匹配将清除计数器
2. ETPB 输出脚仅由 ETMBF 标志位控制
3. 在 ETON 上升沿 ETM 输出脚复位至初始值



10-bit ETM CCRA 比较匹配输出模式 -- ETCCLR=1

注：1. ETCCLR=1，比较器 A 匹配将清除计数器
2. ETPA 输出脚仅由 ETMAF 标志位控制
3. 在 ETON 上升沿输出脚复位至初始值
4. 当 ETCCLR=1 时，不会产生 ETMPF 标志



10-bit ETM CCRB 比较匹配输出模式 -- ETCCLR=1

注：1. ETCCLR=1，比较器 A 匹配将清除计数器
2. ETPB 输出脚仅由 ETMBF 标志位控制
3. 在 ETON 上升沿 ETPB 输出脚复位至初始值
4. 当 ETCCLR=1 时，不会产生 ETMPF 标志

定时 / 计数器模式

为使 ETM 工作在此模式，ETMC1 寄存器的 ETAM1，ETAM0 位和 ETMC2 寄存器的 ETBM1，ETBM0 位需要全部设为高。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 ETM 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 ETM 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 ETM 工作在此模式，ETAM1，ETAM0 和 ETBM1，ETBM0 位需要分别设置为“10”，且 ETAIO1，ETAIO0 和 ETBIO1，ETBIO0 位也需要分别设置为“10”。ETM 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 ETM 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，ETCCLR 位决定 PWM 周期控制方式。当 ETCCLR 设为高，CCRA 寄存器控制 PWM 周期。在这种情况下，CCRB 寄存器设置 PWM 的占空比 (针对 ETPB 输出脚)。CCRP 寄存器和 ETPA 输出脚未使用。PWM 输出只在 ETPB 输出脚产生。当 ETCCLR 清零时，PWM 周期通过 CCRP 三位中八个值之一设置，并且是 128 的倍数。此时，CCRA 和 CCRB 寄存器设置不同占空比，在 ETPA 和 ETPB 引脚输出两个 PWM 波形。

ETPWM1 和 ETPWM0 位决定 PWM 的对齐方式，即边沿或中心对齐方式。在边沿对齐方式中，当计数器清零时，产生 PWM 前沿信号。与此同时电流发生跳变，这在高功耗应用中会出现问题。在中心对齐方式中，PWM 波形中心持续产生有效信号，因此可以减少电流跳变引起的功耗问题。

当比较器 A，比较器 B 或比较器 P 比较匹配发生时，CCRA，CCRB 和 CCRP 中断标志位分别产生。ETMC1 寄存器的 ETAOC 位和 ETMC2 寄存器的 ETBOC 位选择 PWM 波形的极性，ETAIO1，ETAIO0 和 ETBIO1，ETBIO0 位使能 PWM 输出或迫使 ETM 输出脚为高电平或低电平。ETAPOL 和 ETBPOL 位用来取反 PWM 输出波形的极性。

- **10-bit ETM，PWM 模式，边沿对齐模式，ETCCLR=0**

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

若 f_{SYS}=12MHz，ETM 时钟源选择 f_{SYS}/4，CCRP=100b，CCRA=128，CCRB=256，

ETPA PWM 输出频率 =(f_{SYS}/4)/512= f_{SYS}/2048=5.8594kHz，duty=128/512=25%

ETPB PWM 输出频率 =(f_{SYS}/4)/512= f_{SYS}/2048=5.8594kHz，duty=256/512=50%

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

- **10-bit ETM，PWM 模式，边沿对齐模式，ETCCLR=1**

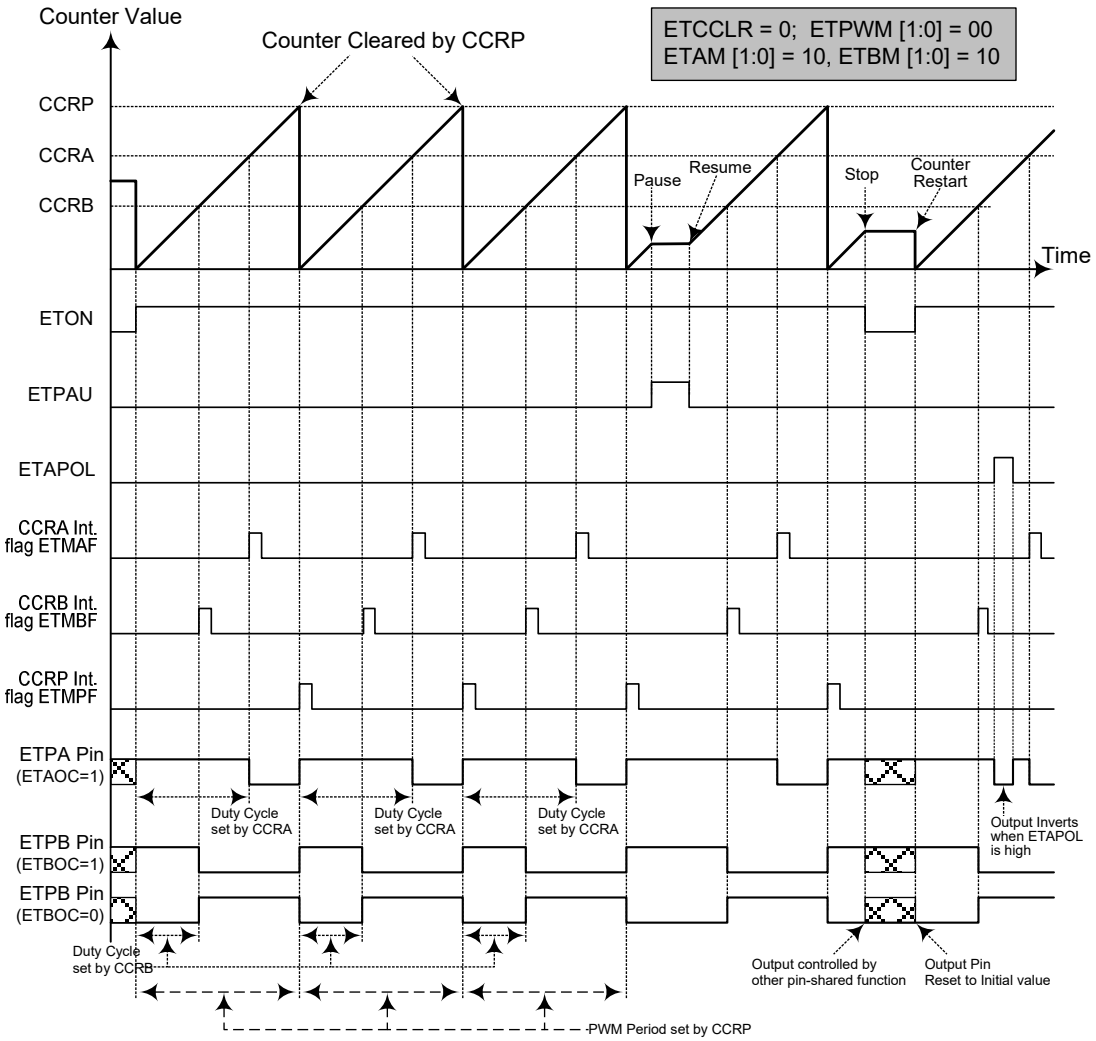
CCRA	1	2	3	.....	511	512	.....	1021	1022	1023
Period	1	2	3	.....	511	512	.....	1021	1022	1023
B Duty	CCRB									

- **10-bit ETM，PWM 模式，中心对齐模式，ETCCLR=0**

CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	256	512	768	1024	1280	1536	1792	2046
A Duty	(CCRA×2) - 1							
B Duty	(CCRB×2) - 1							

- **10-bit ETM，PWM 模式，中心对齐模式，ETCCLR=1**

CCRA	1	2	3	.....	511	512	.....	.....	1021	1022	1023
Period	2	4	6	.....	1022	1024	.....	.....	2042	2044	2046
B Duty	(CCRB×2) - 1										

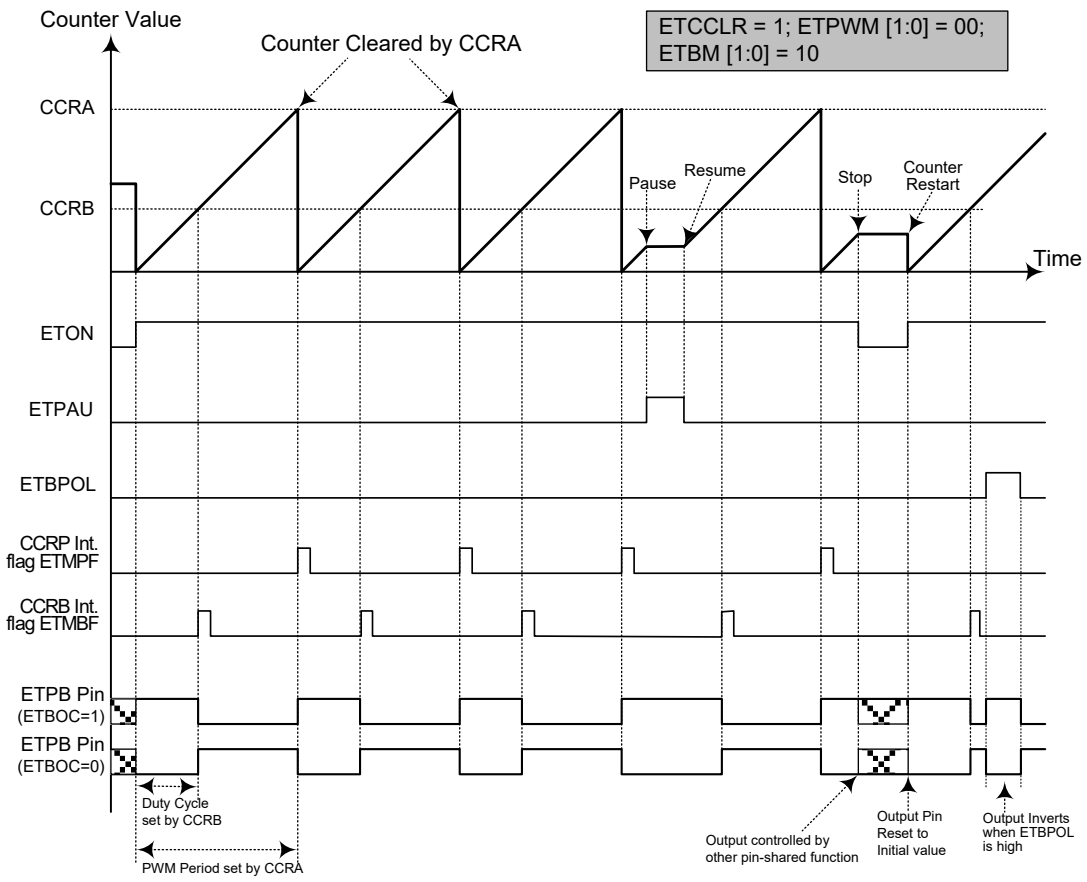


10-bit ETM PWM 模式 -- 边沿对齐

注：1. ETCCLR=0，CCRP 清除计数器并决定 PWM 周期

2. 当 ETAIO[1:0] (或 ETBIO[1:0])=00 或 01，PWM 功能不变

3. CCRA 控制 ETPA PWM 占空比，CCRB 控制 ETPB PWM 占空比



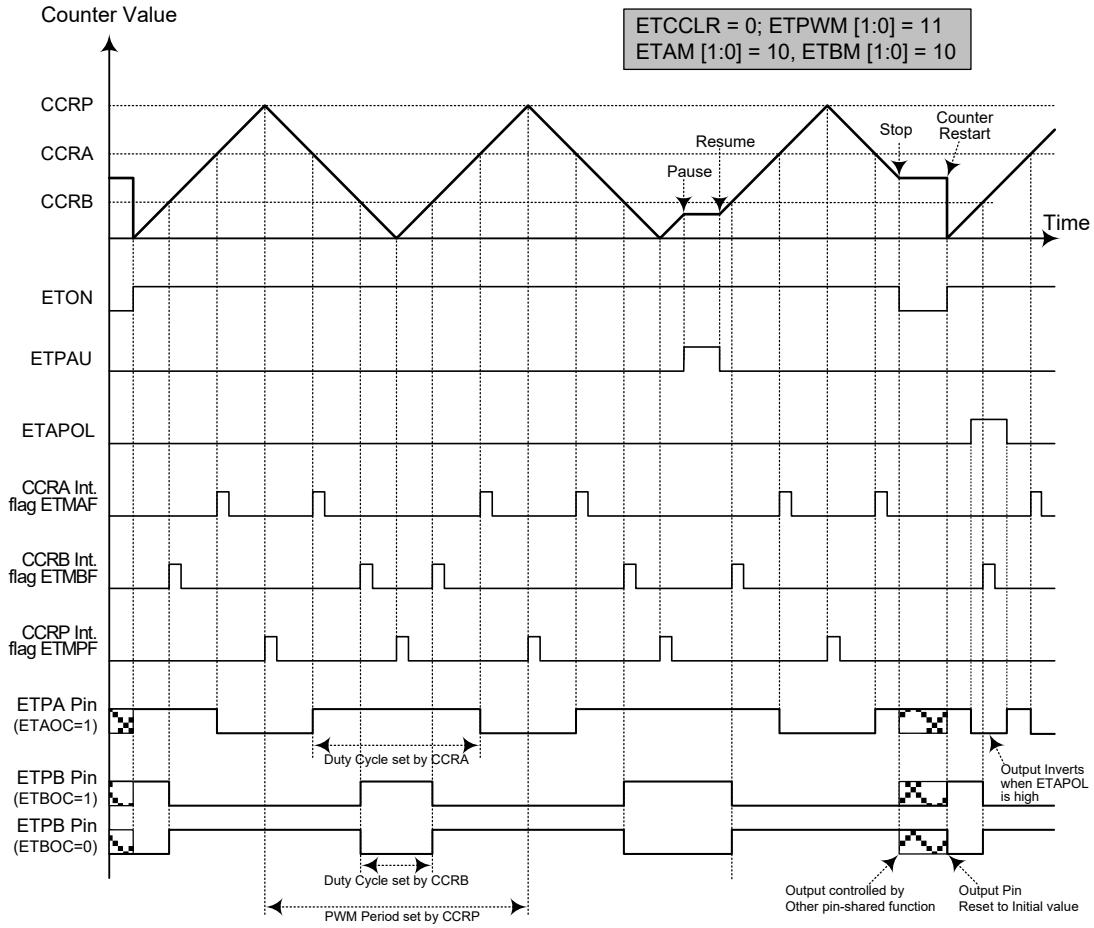
10-bit ETM PWM 模式 -- 边沿对齐

注：1. ETCCLR=1，CCRA 清除计数器并决定 PWM 周期

2. 当 ETBIO[1:0]=00 或 01，PWM 功能不变

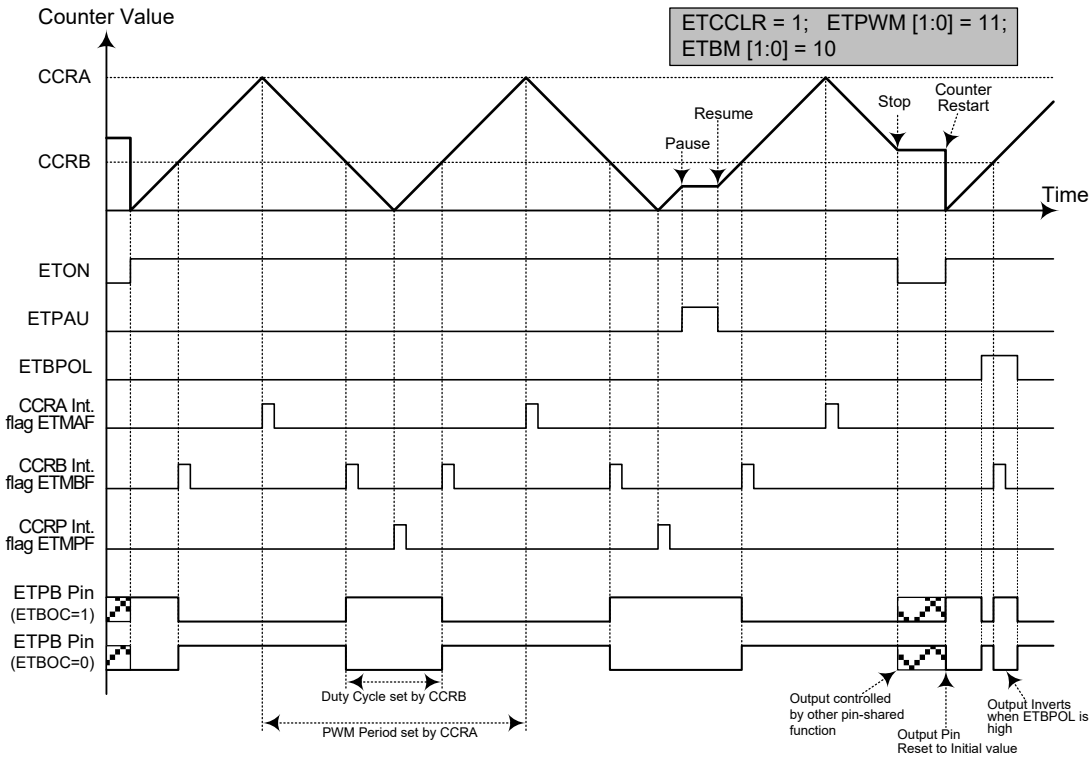
3. CCRA 控制 ETPB PWM 周期，CCRB 控制 ETPB PWM 占空比

4. 此时，ETM 引脚控制寄存器不能使能 ETPA 作为 ETM 输出引脚



10-bit ETM PWM 模式 -- 中心对齐

注：1. ETCCLR=0，CCRP 清除计数器并决定 PWM 周期
2. ETPWM[1:0]=11，PWM 为中心对齐
3. 当 ETAIO[1:0](或 ETBIO[1:0])=00 或 01，PWM 功能不变
4. CCRA 控制 ETPA PWM 占空比，CCRB 控制 ETPB PWM 占空比
5. 计数器值递减至“0”时 CCRP 将产生中断请求



10-bit ETM PWM 模式 -- 中心对齐

注：1. ETCCLR=1，CCRA 清除计数器并决定 PWM 周期

2. ETPWM[1:0]=11，PWM 为中心对齐

3. 当 ETBIO[1:0]=00 或 01 时，PWM 功能不变

4. CCRA 控制 ETPB PWM 周期，CCRB 控制 ETPB PWM 占空比

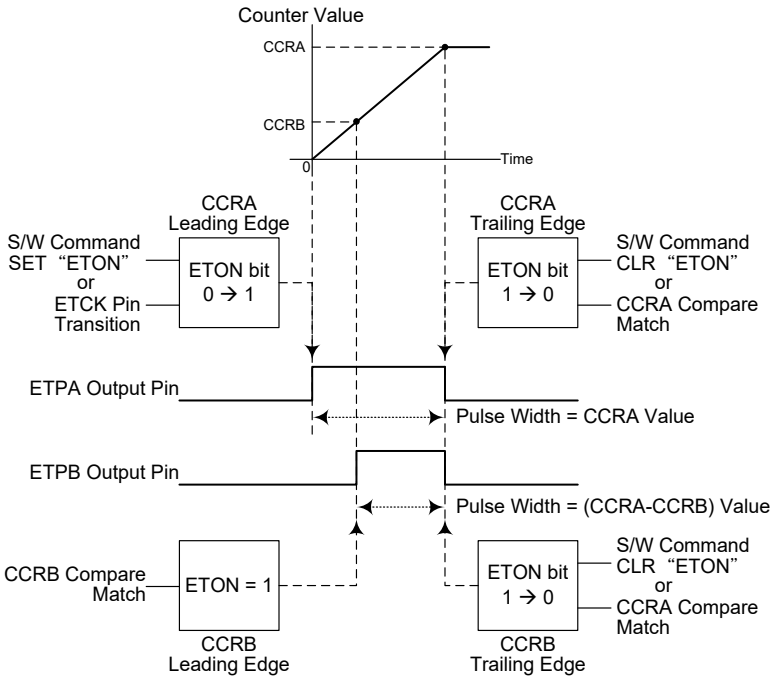
5. 计数器值递减至“0”时 CCRP 将产生中断请求

单脉冲输出模式

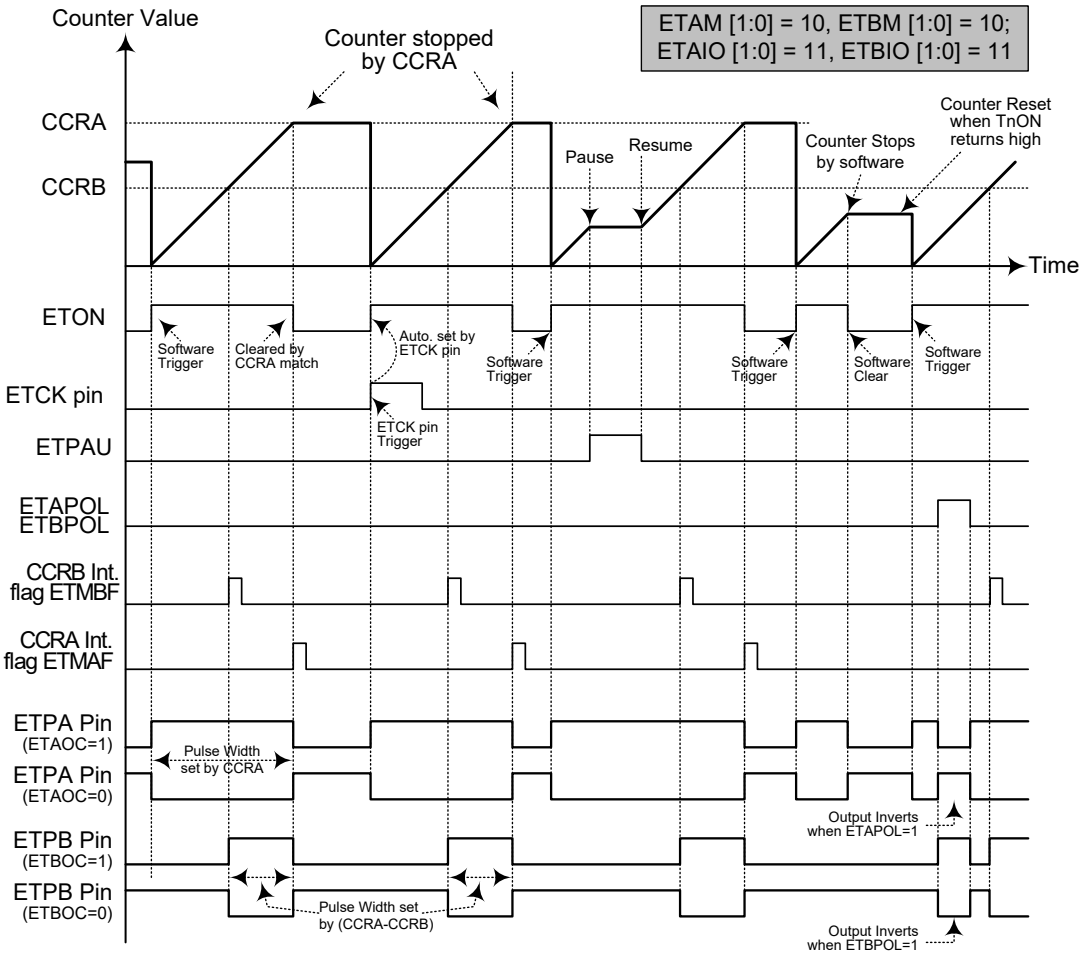
为使 TM 工作在此模式，ETAM1，ETAM0 和 ETBM1，ETBM0 位需要分别设置为“10”，并且相应的 ETAIO1，ETAIO0 和 ETBIO1，ETBIO0 需要分别设置为“11”。正如模式名所言，单脉冲输出模式，在 ETM 输出脚将产生一个脉冲输出。

通过应用程序控制 ETON 位由低到高的转变来触发 ETPA 脉冲前沿输出。通过应用程序产生比较器 B 的比较匹配来触发 ETPB 脉冲前沿输出。处于单脉冲模式时，ETON 位在 ETCK 脚发生有效边沿跳转时自动由低转变为高，进而开始 ETPA 单脉冲输出。当 ETON 位转变为高电平时，计数器将开始运行，并产生 ETPA 脉冲前沿。当脉冲有效时 ETON 位保持高电平。通过应用程序使 ETON 位清零或比较器 A 比较匹配发生时，产生 ETPA 和 ETPB 的脉冲下降沿。

而比较器 A 比较匹配发生时，会自动清除 ETON 位并产生 ETPA 和 ETPB 单脉冲输出边沿跳变。CCRA 的值通过这种方式控制 ETPA 的脉冲宽度，CCRA-CCRB 的值控制 ETPB 的脉冲宽度。比较器 A 和比较器 B 比较匹配发生时，也会产生 ETM 中断。ETON 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 ETCCLR 位未使用。



单脉冲产生示意图



10-bit ETM -- 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 ETCK 脚或设置 ETON 位为高来触发脉冲

4. ETCK 脚有效沿会自动置位 ETON

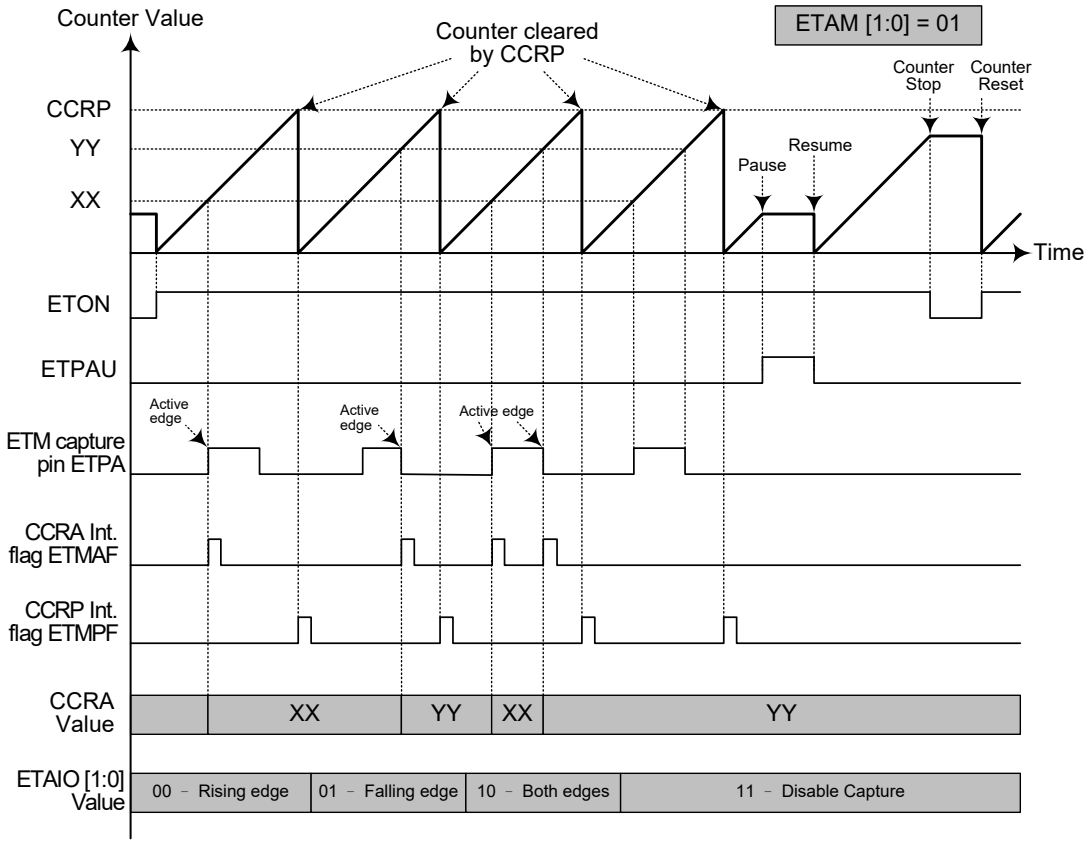
5. 单脉冲模式中，ETAIO[1:0] 和 ETBIO[1:0] 需置位“11”，且不能更改。

捕捉输入模式

为使 ETM 工作在此模式，ETMC1 寄存器的 ETAM1，ETAM0 位和 ETMC2 寄存器的 ETBM1，ETBM0 位需要分别设置为“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。ETPIA 和 ETPIB 引脚上的外部信号，通过设置 ETMC1 寄存器的 ETAIO1，ETAIO0 位和 ETMC2 寄存器的 ETBIO1，ETBIO0 位选择有效边沿类型，即上升沿，下降沿或双沿有效。计数器在 ETON 位由低到高转变时启动并通过应用程序初始化。

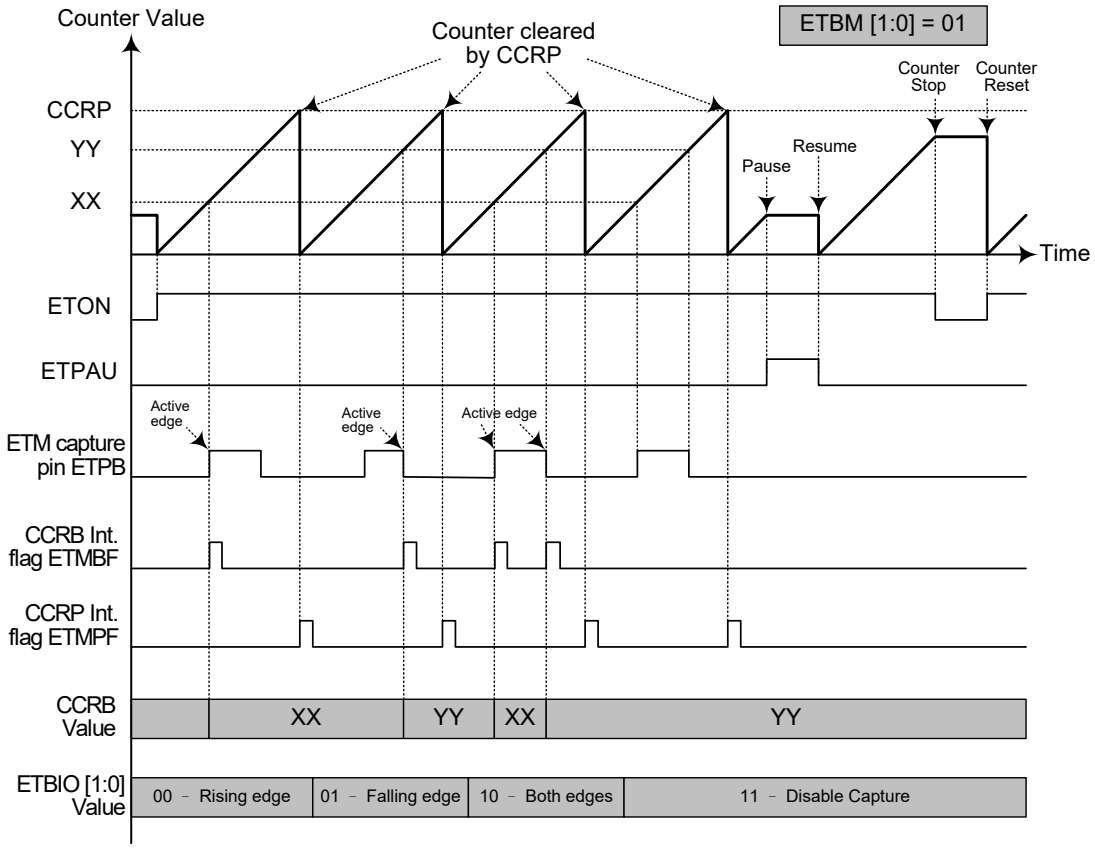
当 ETPIA 和 ETPIB 引脚出现有效边沿转换时，计数器当前值被锁存到 CCRA 和 CCRB 寄存器，并产生 ETM 中断。无论 ETPIA 和 ETPIB 引脚发生什么变化，计数器继续工作直到 ETON 位发生下降沿跳变。当 CCRP 比较匹配发生时计数器复位至零；CCRP 的值通过这种方式控制计数器的最大值。当比较器 P CCRP 比较匹配发生时，也会产生 TM 中断。记录 CCRP 溢出中断信号的值可以测量长脉宽。通过设置 ETAIO1，ETAIO0 位和 ETBIO1，ETBIO0 位选择 ETPIA 和 ETPIB 引脚为上升沿，下降沿或双沿有效。如果 ETAIO1，ETAIO0 位和 ETBIO1，ETBIO0 位都设为高，无论 ETPIA 和 ETPIB 引脚发生什么变化，不会产生捕捉操作，但计数器继续运行。

当 ETPIA 和 ETPIB 引脚与其它功能共用，ETM 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。ETCCLR，ETAOC，ETBOC，ETAPOL 和 ETBPOL 位在此模式中未使用。



10-bit ETM CCRA 捕捉输入模式

注：1. ETAM[1:0]=01 并通过 ETAIO[1:0] 位设置有效边沿
2. ETM 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中
3. ETCCLR 位未使用
4. 无输出功能 – ETAOC 和 ETAPOL 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



10-bit ETM CCRB 捕捉输入模式

注：1. ETBM[1:0]=01 并通过 ETBIO[1:0] 位设置有效边沿
2. ETM 捕捉输入脚的有效边沿将计数器的值转移到 CCRB 中
3. ETCCLR 位未使用
4. 无输出功能 – ETBOC 和 ETBPOL 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

模拟 / 数字转换器

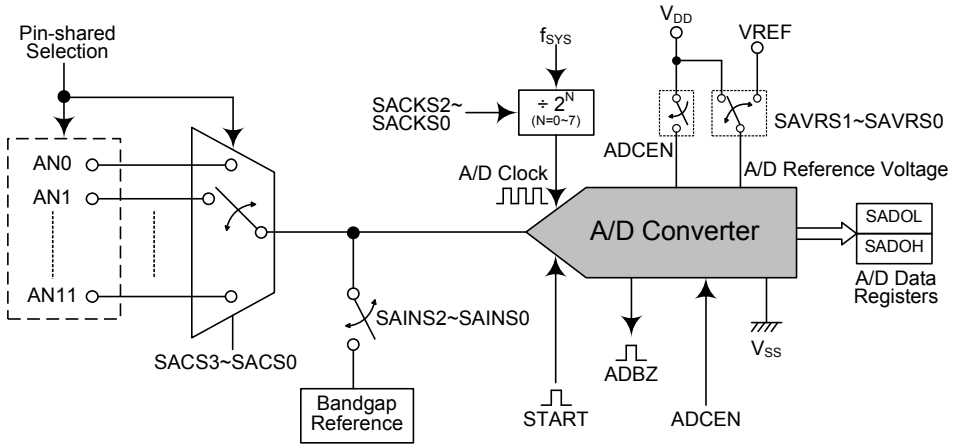
对于大多数电子系统而言，处理现实世界的模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先需要通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

A/D 简介

此系列单片机包含一个多通道的 A/D 转换器，它们可以直接接入外部模拟信号 (来自传感器或其它控制信号) 或内部模拟信号 (例如 Bandgap 参考电压) 并直接将这些信号转换成 12 位的数字量。选择转换外部或内部模拟信号由 SAINS2~SAINS0 位和 SACS3~SACS0 位共同控制。注意，若要转换内部模拟信号时，除了 SAINS 和 SACS 字段外，共用引脚控制位应正确设置。关于 A/D 输入信号的详细描述请参考“A/D 转换寄存器介绍”和“A/D 输入引脚”两节内容。

单片机	输入通道数	A/D 通道选择位	输入引脚
HT67F60A HT67F70A	12	SACS3~SACS0	AN0~AN11

下图显示了 A/D 转换器内部结构和相关的寄存器。



A/D 转换器结构

A/D 转换寄存器介绍

A/D 转换器的所有工作由四个寄存器控制。一对只读寄存器来存放 12 位 ADC 数据的值。剩下两个控制寄存器设置 A/D 转换器的操作和控制功能。

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

A/D 转换寄存器列表

A/D 转换器数据寄存器 – SADOL，SADOH

对于具有 12 位 A/D 转换器的芯片，需要两个数据寄存器存放转换结果，一个高字节寄存器 SADOH 和一个低字节寄存器 SADOL。在 A/D 转换完毕后，单片机可以直接读取这些寄存器以获得转换结果。由于寄存器只使用了 16 位中的 12 位，其数据存储格式由 SADC0 寄存器的 ADRFS 位控制，如下表所示。D0~D11 是 A/D 换转数据结果位。未使用的位读为“0”。当 A/D 转换器除能时，数据寄存器的值将清零。

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

A/D 数据寄存器

A/D 转换控制寄存器 – SADC0, SADC1

寄存器 SADC0 和 SADC1 用来控制 A/D 转换器的功能和操作。这些 8 位的寄存器定义包括选择连接至内部 A/D 转换器的模拟通道，数字化数据格式，A/D 时钟源，并控制和监视 A/D 转换器的忙碌状态。SADC0 寄存器中的 SACS3~SACS0 位用于选择哪个外部模拟输入通道被连接到内部 A/D 转换器。SADC1 寄存器中的 SAINS2~SAINS0 位用于选择外部模拟输入通道或内部模拟信号被连接到内部 A/D 转换器。若 SAINS2~SAINS0 位为“000”，则选择转换外部模拟输入信号，具体通道编号由 SACS3~SACS0 位决定。若 SAINS2~SAINS0 位为“001”，则选择内部 Bandgap 参考电压。当转换内部模拟信号时，必须小心。若内部模拟信号被选择，那么 SACS3~SACS0 必须设置为“11xx”否则，外部输入通道将与内部模拟信号相连接，这将导致不可预期的后果，甚至不可逆的损坏。

SAINS [2:0]	SACS [3:0]	输入信号	描述
000, 101, 110, 111	0000~1011	AN0~AN11	外部模拟通道输入
	11xx	—	浮空
001	11xx	V_{BG}	内部 Bandgap 参考电压
010~100	xxxx	—	保留

A/D 转换器输入信号选择

● **SADC0 寄存器**

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

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

Bit 6 **ADBZ**：A/D 转换忙碌标志位
0：无 A/D 转换正在进行
1：A/D 转换中
此只读位用于表明 A/D 转换是否正在进行。当 START 位由低变为高再变为低时，ADBZ 位为高，表明 A/D 转换已初始化。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 转换除能时，A/D 数据寄存器 SADOH 和 SADOL 的内容将清零。

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

Bit 3~0 **SACS3~SACS0**：A/D 外部模拟通道输入选择位
0000：AN0
0001：AN1
0010：AN2
0011：AN3
0100：AN4
0101：AN5
0110：AN6
0111：AN7
1000：AN8
1001：AN9
1010：AN10
1011：AN11
11xx：浮空

● **SADC1 寄存器**

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

Bit 7~5 **SAINS2~SAINS0**：A/D 输入信号选择位
000：外部信号 – 外部模拟通道输入
001：内部信号 – 内部 Bandgap 参考电压
010~100：保留
101~111：外部信号 – 外部模拟通道输入
当 SAINS2~SAINS0 被设置为“001”选择转换内部模拟信号时，必须小心。若内部模拟信号被选择，那么 SACS3~SACS0 必须设置为“11xx”。否则，外部输入通道将与内部模拟信号相连接，这将导致不可预期的后果，甚至不可逆的损坏。

Bit 4~3 **SAVRS1~SAVRS0**：A/D 参考电压选择位
00：来自于 VREF 引脚
01：来自于 VDD 引脚
1x：来自于 VREF 引脚

Bit 2~0 **SACKS2~SACKS0**：A/D 时钟源选择位
000：f_{SYS}
001：$f_{SYS}/2$
010：$f_{SYS}/4$
011：$f_{SYS}/8$
100：$f_{SYS}/16$
101：$f_{SYS}/32$
110：$f_{SYS}/64$
111：$f_{SYS}/128$
这三位用于选择 A/D 转换器的时钟源。

A/D 操作

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

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

A/D 转换器的时钟源为系统时钟 f_{SYS} 或其分频，而分频系数由 SADC1 寄存器中的 SACKS2~SACKS0 位决定。虽然 A/D 时钟源是由系统时钟 f_{SYS} 和 SACKS2~SACKS0 位决定，但可选择的最大 A/D 时钟源则有一些限制。由于允许的 A/D 时钟周期 t_{ADCK} 的范围为 0.5μs~10μs，所以选择系统时钟速度时就必须小心。如果系统时钟速度为 4MHz 时，SACKS2~SACKS0 位不能设为“000”或“11x”。必须保证设置的 A/D 转换时钟周期不小于时钟周期的最小值或大于时钟周期的最大值，否则将会产生不准确的 A/D 转换值。

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 ($f_{SYS}/62$)	SACKS[2:0] = 110 ($f_{SYS}/64$)	SACKS[2:0] = 111 ($f_{SYS}/128$)
1MHz	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *	128μs *
2MHz	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *	64μs *
4MHz	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *	32μs *
8MHz	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs	16μs *
12MHz	83ns *	167ns *	333ns *	667ns	1.33μs	2.67μs	5.33μs	10.67μs *
16MHz	62.5ns *	125ns *	250ns *	500ns	1μs	2μs	4μs	8μs
20MHz	50ns *	100ns *	200ns *	400ns *	800ns	1.6μs	3.2μs	6.4μs

A/D 时钟周期范例

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

A/D 转换器参考电压来自正电源电压 VDD 或外部参考源引脚 VREF，可通过 SAVRS1 和 SAVRS0 位来选择。当 SAVRS 字段被设为“01”，A/D 转换器参考电压将来自于 VDD 引脚。如果 SAVRS 字段被设置为除了“01”以外的值时，A/D 转换器参考电压将来自于 VREF 引脚。由于 VREF 引脚与其它功能共用，当 VREF 引脚作为参考电压输入脚时，VREF 引脚共用功能控制位必须被适当的设置以除能其它引脚功能。

SAVRS [1:0]	参考电压	描述
00	V_{REF}	A/D 转换器参考电压来自于 VREF 引脚
01	V_{DD}	A/D 转换器参考电压来自于 VDD 引脚
1x	V_{REF}	A/D 转换器参考电压来自于 VREF 引脚

A/D 转换器参考电压选择

A/D 输入引脚

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

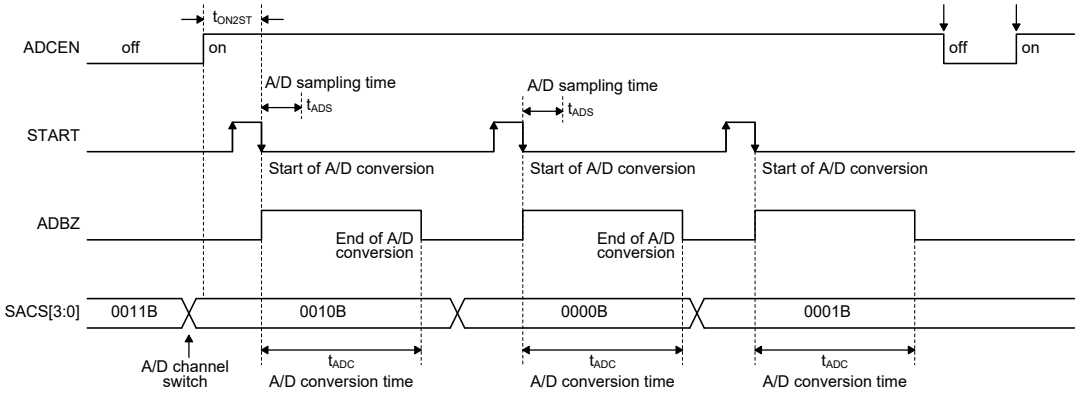
A/D 转换器有自己的参考电压引脚 VREF，而通过设置 SADC1 寄存器中的 SAVRS1~SAVRS0 位，参考电压也可以选择来自电源电压引脚。模拟输入值一定不能超过 V_{REF} 值。

A/D 转换率及时序图

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

最大 A/D 转换率 = A/D 时钟周期 /16

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



A/D 转换时序图

A/D 转换步骤

下面概述实现 A/D 转换过程的各个步骤。

- 步骤 1
  通过 SADC1 寄存器中的 SACKS2~SACKS0 位，选择所需的 A/D 转换时钟。
- 步骤 2
  将 SADC0 寄存器中的 ADCEN 位置高使能 A/D。
- 步骤 3
  通过 SADC1 寄存器中的 SAINS2~SAINS0 位，选择连接至内部 A/D 转换器的信号。
  若选择外部通道输入，接着执行步骤 4。
  若选择内部模拟信号，接着执行步骤 5。
- 步骤 4
  若已通过 SAINS2~SAINS0 位选择 A/D 输入信号来自外部通道输入，接着应设置相关的引脚共用控制位将该引脚规划为 A/D 输入引脚。通过设置 SACS3~SACS0 位选择哪个外部通道接至 A/D 转换器。接着执行步骤 6。
- 步骤 5
  若已通过 SAINS2~SAINS0 位选择 A/D 输入信号来自内部模拟信号，SACS3~SACS0 必须首先设置为“11xx”断开外部通道输入。所需的内部模拟信号可通过 SAINS2~SAINS0 选择。接着执行步骤 6。
- 步骤 6
  通过 SAVRS1~SAVRS0 位选择参考电压。
- 步骤 7
  设置 ADRFS 位选择 A/D 转换器输出数据格式。

- 步骤 8
  如果要使用中断，则中断控制寄存器需要正确地设置，以确保 A/D 中断功能是激活的。总中断控制位 EMI 需要置位为“1”，以及 A/D 转换器中断位 ADE 也需要置位为“1”。
- 步骤 9
  现在可以通过设置 START 位从“0”到“1”再回到“0”，开始模数转换的过程。
- 步骤 10
  如果 A/D 转换正在进行中，ADBZ 位会被置为逻辑高。A/D 转换完成后，ADBZ 位会被置为逻辑低，并可从 SADOH 和 SADOL 寄存器中读取输出数据。
  注：若使用轮询 SADC0 寄存器中 ADBZ 位的状态的方法来检查转换过程是否结束时，则中断使能的步骤可以省略。

编程注意事项

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

A/D 转换功能

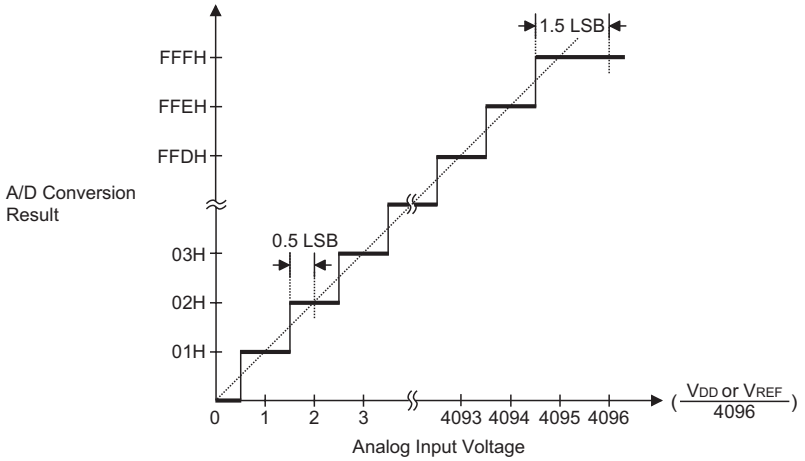
单片机含有一组 12 位的 A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于 V_{DD} 或 V_{REF} 的电压值，因此每一位可表示 V_{DD} 或 $V_{REF}/4096$ 的模拟输入值。

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

通过下面的等式可估算 A/D 转换器输入电压值：

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

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



理想的 A/D 转换功能

A/D 转换应用范例

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

范例 1：使用查询 ADBZ 的方式来检测转换结束

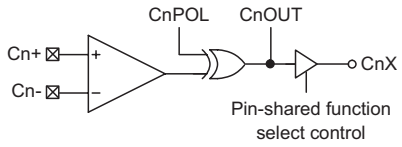
```
clr ADE                    ; disable ADC interrupt
mov a,03H
mov SADC1,a                ; select fSYS/8 as A/D clock
set ADCEN
mov a,0CH                  ; setup PAS0 to configure pin AN0
mov PAS0,a
mov a,20H
mov SADC0,a                ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr START                  ; high pulse on start bit to initiate conversion
set START                  ; reset A/D
clr START                  ; start A/D
:
polling_EOC:
sz  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 ADRL_buffer,a          ; save result to user defined register
mov a,SADOH                ; read high byte conversion result value
mov ADRH_buffer,a          ; save result to user defined register
:
jmp start_conversion       ; start next A/D conversion
```

范例 2：使用中断的方式来检测转换结束

```
clr ADE                    ; disable ADC interrupt
mov a,03H
mov SADC1,a                ; select fSYS/8 as A/D clock
set ADCEN
mov a,0Ch                  ; setup PAS0 to configure pin AN0
mov PAS0,a
mov a,20h
mov SADC0,a                ; enable and connect AN0 channel to A/D converter
:
Start_conversion:
clr START                  ; high pulse on START bit to initiate conversion
set START                  ; reset A/D
clr START                  ; start A/D
clr ADF                    ; clear 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 adrl_buffer,a          ; save result to user defined register
mov a, SADOH               ; read high byte conversion result value
mov adrh_buffer,a          ; save result to user defined register
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a               ; restore STATUS from user defined memory
mov a,acc_stack            ; restore ACC from user defined memory
reti
```

比较器

该系列芯片中含有两个独立的模拟比较器。它们具有暂停、极性选择、迟滞等功能，可通过寄存器进行灵活配置。比较器的引脚与普通 I/O 引脚共享，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

比较器操作

此系列单片机包含两个比较器功能，用于比较两个模拟电压，基于它们的差值上提供一个输出。控制寄存器 CP0C 和 CP1C 可分别控制相应的内部比较器。比较器的输出可由各自寄存器的一位记录，并且在共用的 I/O 口上输出。此外，比较器功能有输出极性，迟滞功能和暂停控制。

当比较器使能时，连接到与比较器共用的输入引脚的上拉电阻将自动失效。当比较器输入接近其切换电压时，由于输入信号上升或下降速度较慢，比较器输出端可能会产生一些伪输出信号。通过选择迟滞功能提供少量正反馈给比较器可使此种情况的发生率较大程度地降低。理想情况下正负输入信号在同一个电压点时比较器将发生开关动作，但是不可避免的输入失调电压会导致情况不确定。若迟滞功能使能，也可增加切换偏差值。

比较器寄存器

比较器工作相关的寄存器共有两个，分别对应两个比较器。两个比较器中对应的位有特殊的功能，两个寄存器列表如下。

寄存器名称	位							
	7	6	5	4	3	2	1	0
CP0C	—	C0EN	C0POL	C0OUT	—	—	—	C0HYEN
CP1C	—	C1EN	C1POL	C1OUT	—	—	—	C1HYEN

比较器寄存器列表

CP0C 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	C0EN	C0POL	C0OUT	—	—	—	C0HYEN
R/W	—	R/W	R/W	R	—	—	—	R/W
POR	—	0	0	0	—	—	—	1

Bit 7　未定义，读为“0”

Bit 6　**C0EN**：比较器开 / 关控制位
0：关闭
1：开启
此位为比较器开 / 关控制位。为“0”时，比较器关闭，即使其引脚上加有模拟电压也不会产生功耗。对功耗要求严格的应用中，当比较器未使用或单片机进入休眠或空闲模式之前，此位应清零。

Bit 5　**C0POL**：比较器输出极性位
0：输出同相
1：输出反相
此位决定比较器极性。为“0”时，C0OUT 位与比较器输出条件同相；为“1”时，C0OUT 位与比较器输出条件反相。

Bit 4　**C0OUT**：比较器输出位
C0POL=0
0: C0+ < C0-
1: C0+ > C0-
C0POL=1
0: C0+ > C0-
1: C0+ < C0-
此位为比较器输出位。此位的极性由比较器输入电压和 C0POL 位的状态决定。

Bit 3~1　未定义，读为“0”

Bit 0　**C0HYEN**：迟滞控制位
0：关闭
1：开启
此位为迟滞控制位。为“1”时，比较器有一定量迟滞，具体见比较器电气特性表。滞后产生的正反馈将减少比较器门槛附近的伪开关效应的影响。

CP1C 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	C1EN	C1POL	C1OUT	—	—	—	C1HYEN
R/W	—	R/W	R/W	R	—	—	—	R/W
POR	—	0	0	0	—	—	—	1

Bit 7 未定义，读为“0”

Bit 6 **C1EN**：比较器开 / 关控制位
0：关闭
1：开启
此位为比较器开 / 关控制位。为“0”时，比较器关闭，即使其引脚上加有模拟电压也不会产生功耗。对功耗要求严格的应用中，当比较器未使用或单片机进入休眠或空闲模式之前，此位应清零。

Bit 5 **C1POL**：比较器输出极性位
0：输出同相
1：输出反相
此位决定比较器极性。为“0”时，C1OUT 位与比较器输出条件同相；为“1”时，C1OUT 位与比较器输出条件反相。

Bit 4 **C1OUT**：比较器输出位
C1POL=0
0: C1+ < C1-
1: C1+ > C1-
C1POL=1
0: C1+ > C1-
1: C1+ < C1-
此位为比较器输出位。此位的极性由比较器输入电压和 C1POL 位的状态决定。

Bit 3~1 未定义，读为“0”

Bit 0 **C1HYEN**：迟滞控制位
0：关闭
1：开启
此位为迟滞控制位。为“1”时，比较器有一定量迟滞，具体见比较器电气特性表。滞后产生的正反馈将减少比较器门槛附近的伪开关效应的影响。

比较器中断

每个比较器都有自己的中断功能。当任何一个状态改变时，其对应的中断标志将会置位，若应答中断使能位被置位，系统将跳转至相应的中断向量中执行。注意，触发比较器产生中断的条件是 C0OUT 或 C1OUT 位状态的改变而非比较器输出引脚状态的改变。单片机处于休眠或空闲模式且比较器使能时，若为外部输入线导致比较器输出状态发生的改变，则由此产生的中断标志也可产生一个唤醒动作。若除能唤醒功能，进入休眠或空闲模式前中断标志应先置为高。

编程注意事项

若比较器使能，当单片机进入休眠或空闲模式时其仍保持有效并会有一定的耗电，用户可考虑在进入休眠或空闲模式前先关闭比较器。

由于比较器引脚与普通输入 / 输出脚共用，若比较器功能使能时，这些引脚的输入 / 输出寄存器将读为“0”(端口控制寄存器读为“1”) 或读为端口数据寄存器的值 (端口控制寄存器读为“0”)。

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 I^2C 接口。这两种接口具有相当简单的通信协议，单片机可以通过这些接口与传感器、闪存或 EEPROM 内存等硬件设备通信。SIM 接口的引脚与其它的 I/O 引脚共用，所以要先通过相关的引脚共用功能选择位选择使用 SIM 功能引脚。因为这两种接口共用引脚和寄存器，所以要通过一个 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 / 输出口共用的 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串行数据接口，这个协议可以简化与外部硬件的编程要求。

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

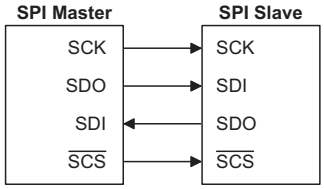
SPI 接口操作

SPI 接口是一个全双工串行数据传输器。SPI 接口的四线为：SDI、SDO、SCK 和 $\overline{SCS}$。SDI 和 SDO 是数据的输入和输出线。SCK 是串行时钟线，$\overline{SCS}$ 是从机的选择线。SPI 的接口引脚与普通 I/O 口和 I^2C 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。

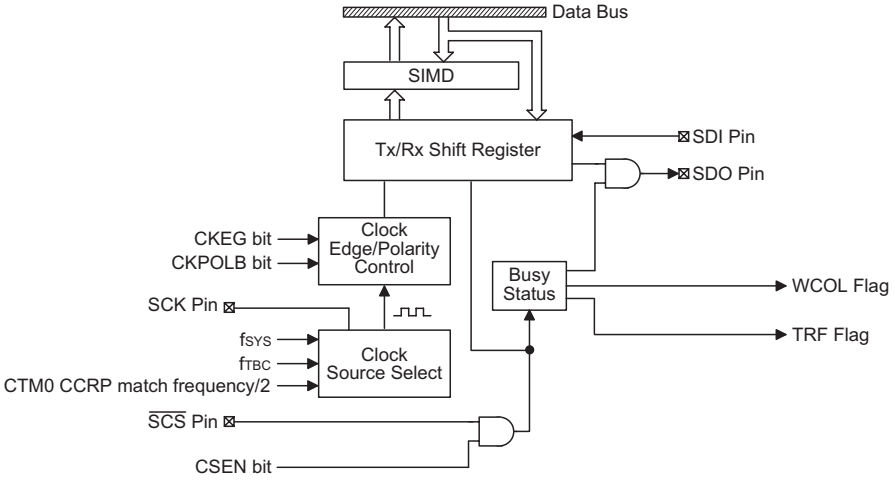
该系列单片机的 SPI 功能具有以下特点：

- 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 CSEN，SIMEN 位的状态。



SPI 主 / 从机连接方式



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	—
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF

SIM 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 SPI 总线中时，要传输的数据应先存在 SIMD 中。SPI 总线接收到数据之后，单片机就可以从 SIMD 数据寄存器中读取。所有通过 SPI 传输或接收的数据都必须通过 SIMD 实现。

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

● **SIMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	—
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	—
POR	1	1	1	—	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_{TBC}
100：SPI 主机模式；SPI 时钟为 CTM0 CCRP 匹配频率 /2
101：SPI 从机模式
110：I^2C 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 I^2C 或 SPI 功能。SPI 时钟源可来自于系统时钟也可以选择来自 CTM0。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 未定义，读为“0”

Bit 3~2 **SIMDEB1~SIMDEB0**：I^2C 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

Bit 1 **SIMEN**：SIM 控制位
0：除能
1：使能
此位为 SIM 接口的开 / 关控制位。此位为“0”时，SIM 接口除能，SDI、SDO、SCK 和 SCS 或 SDA 和 SCL 脚将失去 SPI 或 I^2C 功能，SIM 工作电流减小到最小值。此位为“1”时，SIM 接口使能。若 SIM 经由 SIM2~SIM0 位设置为工作在 SPI 接口，当 SIMEN 位由低到高转变时，SPI 控制寄存器中的设置不会发生变化，其首先应在应用程序中初始化。若 SIM 经由 SIM2~SIM0 位设置为工作在 I^2C 接口，当 SIMEN 位由低到高转变时，I^2C 控制寄存器中的设置，如 HXT 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I^2C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0 未定义，读为“0”

● **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**：时钟线的基础状态位
0：当时钟无效时，SCK 口为高电平
1：当时钟无效时，SCK 口为低电平
此位决定了时钟线的基础状态，当时钟无效时，若此位为高，SCK 为低电平，若此位为低，SCK 为高电平。

Bit 4　**CKEG**：SPI 的 SCK 有效时钟边沿类型位
CKPOLB=0
0：SCK 为高电平且在 SCK 上升沿抓取数据
1：SCK 为高电平且在 SCK 下降沿抓取数据
CKPOLB=1
0：SCK 为低电平且在 SCK 下降沿抓取数据
1：SCK 为低电平且在 SCK 上升沿抓取数据
CKEG 和 CKPOLB 位用于设置 SPI 总线上时钟信号输入和输出方式。在执行数据传输前，这两位必须被设置，否则将产生错误的时钟边沿信号。CKPOLB 位决定时钟线的基本状态，若时钟无效且此位为高，则 SCK 为低电平，若时钟无效且此位为低，则 SCK 为高电平。CKEG 位决定有效时钟边沿类型，取决于 CKPOLB 的状态。

Bit 3　**MLS**：SPI 数据移位命令位
0：LSB
1：MSB
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 $\overline{\text{SCS}}$ 引脚控制位
0：除能
1：使能
CSEN 位用于 $\overline{\text{SCS}}$ 引脚的使能 / 除能控制。此位为低时，$\overline{\text{SCS}}$ 除能并处于浮空状态。此位为高时，$\overline{\text{SCS}}$ 使能并作为选择脚。

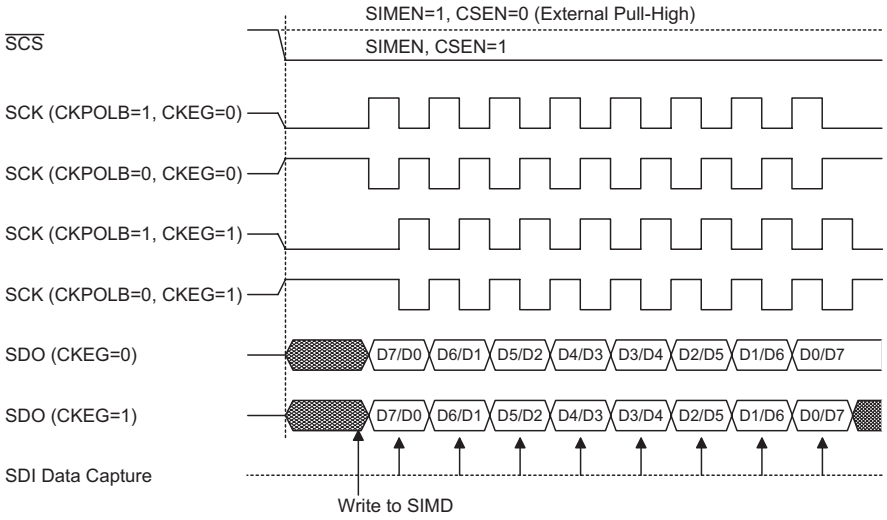
Bit 1　**WCOL**：SPI 写冲突标志位
0：无冲突
1：冲突
WCOL 标志位用于监测数据冲突的发生。此位为高时，数据在传输时被写入 SIMD 寄存器。若数据正在被传输时，此操作无效。此位可被应用程序清零。

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。

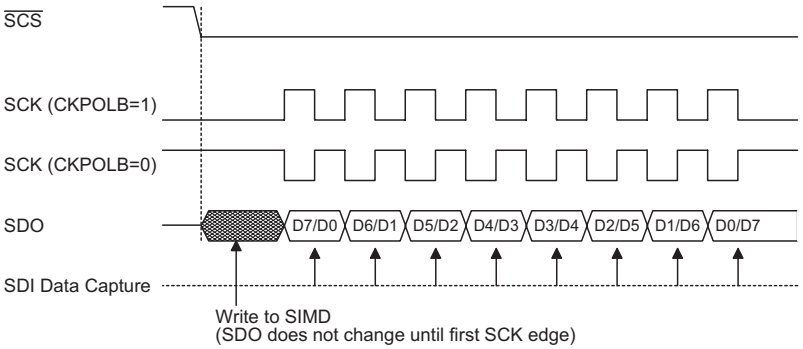
SPI 通信

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 $\overline{SCS}$ 信号以使能从机，从机的数据传输功能也应在与 $\overline{SCS}$ 信号相关的适当时候准备就绪，这由 CKPOLB 和 CKEG 位决定。所附时序图表明了在 CKPOLB 和 CKEG 位各种设置情况下从机数据与 $\overline{SCS}$ 信号的关系。

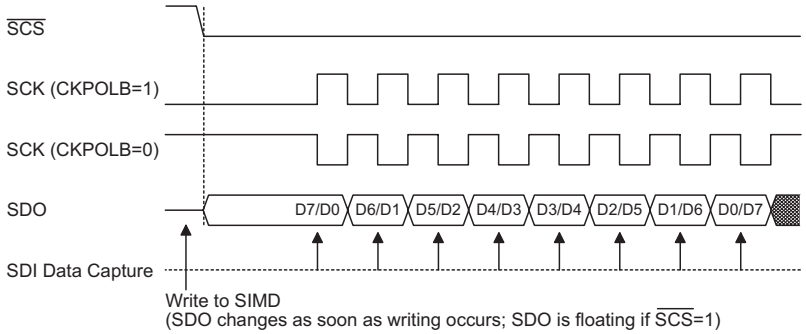
即使在单片机处于空闲模式，SPI 功能仍将继续执行。



SPI 主机模式时序

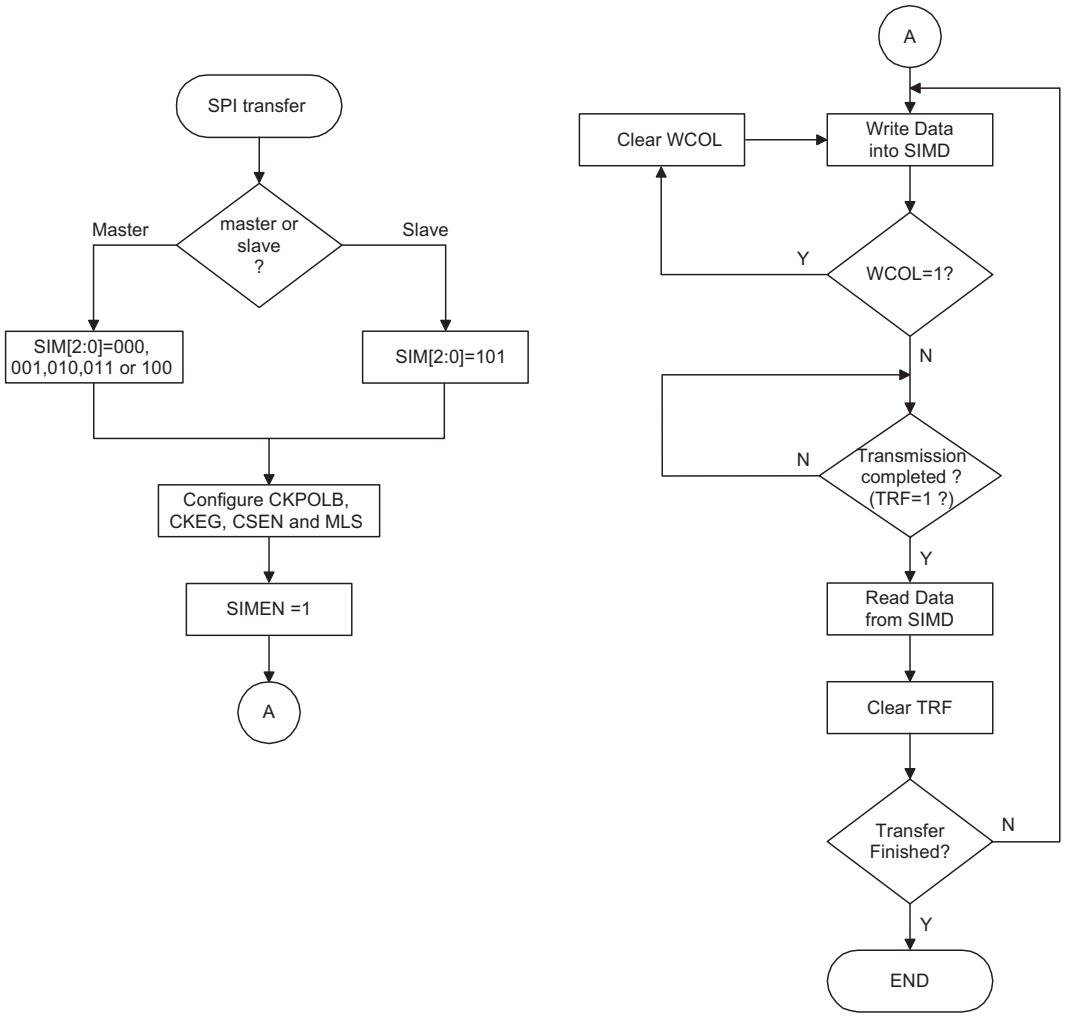


SPI 从机模式时序 -- CKEG=0



注：对于 SPI 从机模式，如果 SIMEN=1 且 CSEN=0，SPI 是一直使能的，忽略 SCS 电平值。

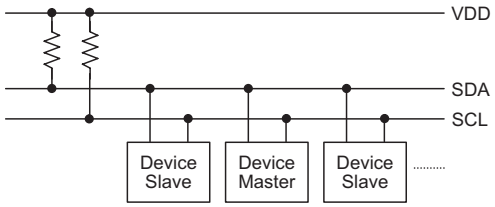
SPI 从机模式时序 -- CKEG=1



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，使之在很多的应用场合中大受欢迎。

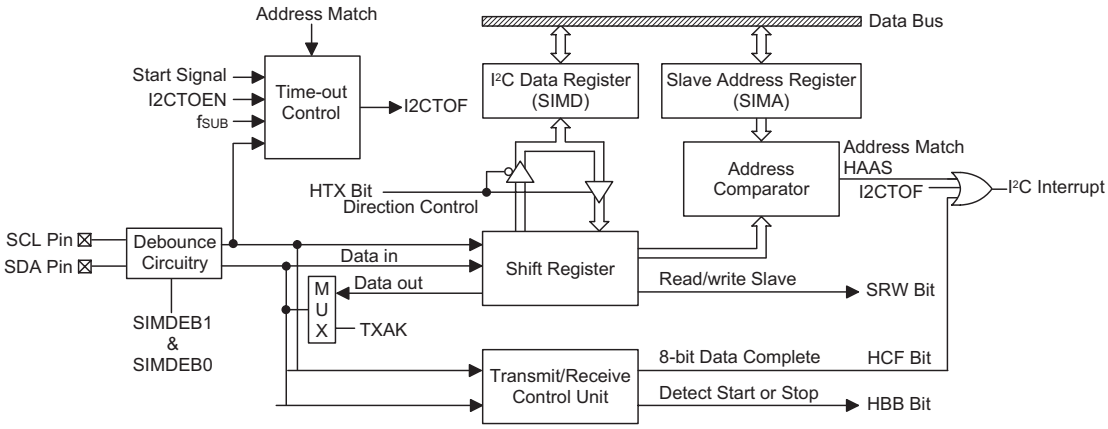


I²C 主从总线连接图

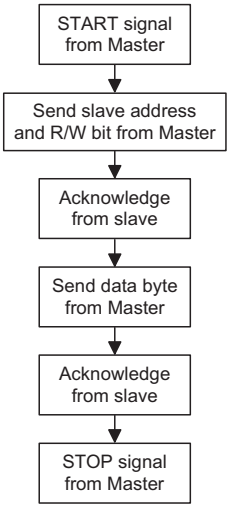
I²C 接口操作

I²C 串行接口是一个双线的接口，有一条串行数据线 SDA 和一条串行时钟线 SCL。由于可能有多个设备在同一条总线上相互连接，所以这些设备的输出都是开漏型输出。因此应在这些输出口上都应加上拉电阻。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，以避免单片机发生误动作。如果选择了这个功能，去抖时间可以选择 2 个或 4 个系统时钟。为了达到需要的 I²C 数据传输速度，系统时钟 f_{SYS} 和 I²C 去抖时间之间存在一定的关系。I²C 标准模式或者快速模式下，用户需注意所选的系统时钟频率与标准匹配去抖时间的设置，其具体关系如下表所示。

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

I²C 最小 f_{SYS} 频率

I²C 寄存器

I²C 总线有三个控制寄存器 SIMC0、SIMC1 和 SIMA，及一个数据寄存器 SIMD。SIMD 寄存器，SPI 章节中已有介绍，用于存储正在传输和接收的数据，当单片机将数据写入 I²C 总线之前，实际将被传输的数据存放在寄存器 SIMD 中。从 I²C 总线接收到数据之后，单片机就可以从寄存器 SIMD 中得到这个数据。I²C 总线上的所有传输或接收到的数据都必须通过 SIMD。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	—
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	D0

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 是同一个寄存器。

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	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 7~1 **IICA6~IICA0**：I²C 从机地址位
IICA6~IICA0 是从机地址 bit 6 ~ bit 0。

Bit 0 未定义
此位可通过软件程序进行读写。

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 传输状态的相关标志位。

● **SIMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	—
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	—
POR	1	1	1	—	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_{TBC}
100：SPI 主机模式；SPI 时钟为 CTM0 CCRP 匹配频率 /2
101：SPI 从机模式
110：I^2C 从机模式
111：无 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 I^2C 或 SPI 功能。SPI 时钟源可来自于系统时钟也可以选择来自 CTM0。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 未定义，读为“0”

Bit 3~2 **SIMDEB1~SIMDEB0**：I^2C 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

Bit 1 **SIMEN**：SIM 控制位
0：除能
1：使能
此位为 SIM 接口的开 / 关控制位。此位为“0”时，SIM 接口除能，SDI、SDO、SCK 和 SCS 或 SDA 和 SCL 脚将失去 SPI 或 I^2C 功能，SIM 工作电流减小到最小值。此位为“1”时，SIM 接口使能。若 SIM 经由 SIM2~SIM0 位设置为工作在 SPI 接口，当 SIMEN 位由低到高转变时，SPI 控制寄存器中的设置不会发生变化，其首先应在应用程序中初始化。若 SIM 经由 SIM2~SIM0 位设置为工作在 I^2C 接口，当 SIMEN 位由低到高转变时，I^2C 控制寄存器中的设置，如 HXT 和 TXAK，将不会发生变化，其首先应在应用程序中初始化，此时相关 I^2C 标志，如 HCF、HAAS、HBB、SRW 和 RXAK，将被设置为其默认状态。

Bit 0 未定义，读为“0”

● **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**：I^2C 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
HCF 是数据传输标志位。数据正在传输时该位为低。当 8 位数据传输完成，此位为高并产生一个中断。

Bit 6 **HAAS**：I^2C 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**：I^2C 总线忙标志位
0：I^2C 总线闲
1：I^2C 总线忙
当检测到 START 信号时 I^2C 忙，此位变为高电平。当检测到 STOP 信号时 I^2C 总线停止，该位变为低电平。

Bit 4 **HTX**：I^2C 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**：I^2C 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**：I^2C 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 I^2C 从机读写位。决定主机是否希望传输或接收来自 I^2C 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**：I^2C 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 I^2C 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 I^2C 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**：I^2C 总线接收确认标志位
0：从机接收到确认标志
1：从机没有接收到确认标志
RXAK 位是接收确认标志位。如果 RXAK 位被重设为“0”即 8 位数据传输之后，设备在第九个时钟有接受到一个正确的确认位。如果单片机处于发送状态，发送方会检查 RXAK 位来判断接收方是否愿意继续接收下一个字节。因此直到 RXAK 为“1”时，传输方停止发送数据。这时，传输方将释放 SDA 线，主机发出停止信号释放 I^2C 总线。

I²C 总线通信

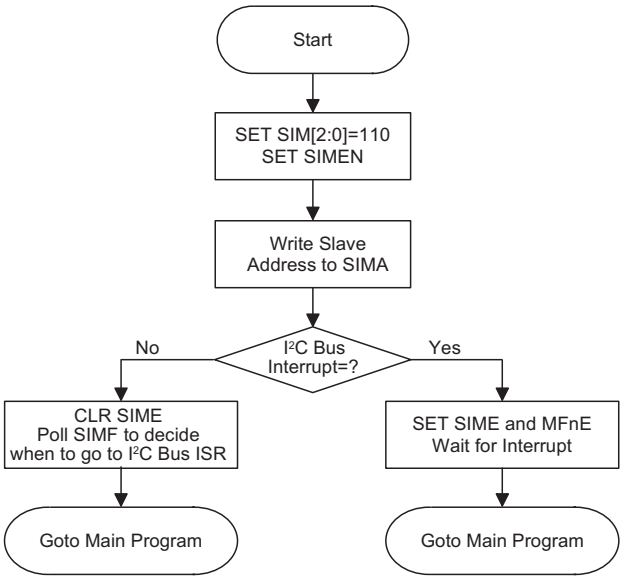
I²C 总线上的通信需要四步完成，一个起始信号，一个从机地址发送，一个数据传输，还有一个停止信号。当起始信号被写入 I²C 总线时，总线上的所有从机都会接收到这个起始信号并且被通知总线上会即将有数据到达。数据的前 7 位是从机地址，高位在前，低位在后。如果发出的地址和从机地址匹配，SIMC1 寄存器的 HAAS 位会被置位，同时产生 I²C 中断。进入中断服务程序后，系统要检测 HAAS 位和 I2CTOF 位，以判断 I²C 总线中断是来自从机地址匹配，还是来自 8 位数据传递完毕或是来自 I²C 超时。在数据传递中，注意的是，在 7 位从机地址被发送后，接下来的一位，即第 8 位，是读 / 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 位为“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 SIME 位和中断控制寄存器中的 SIM 多功能中断使能位，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。

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 I2CTOF 位，以判断 I²C 总线中断是来自从机地址匹配，还是来自 8 位数据传递完毕，或是来自 I²C 超时。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。

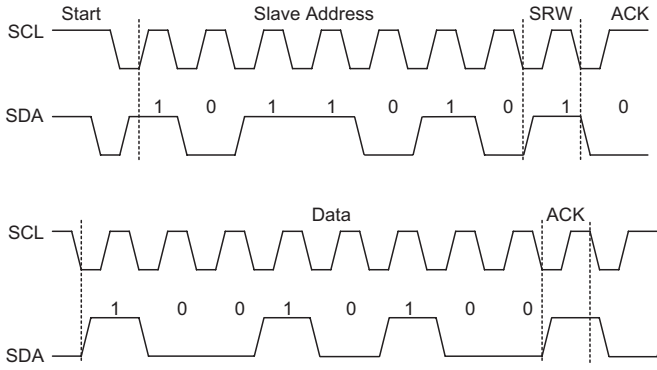
I²C 总线从机地址确认信号

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。

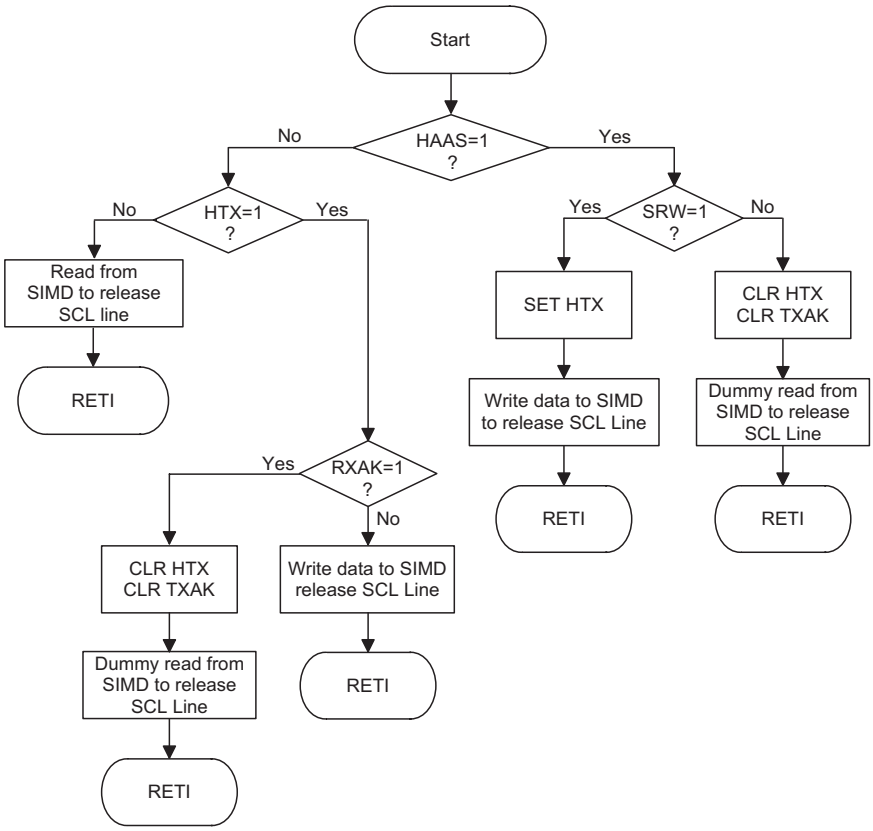


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	--------	S	SA	SR	M	D	A	D	A	--------	P

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

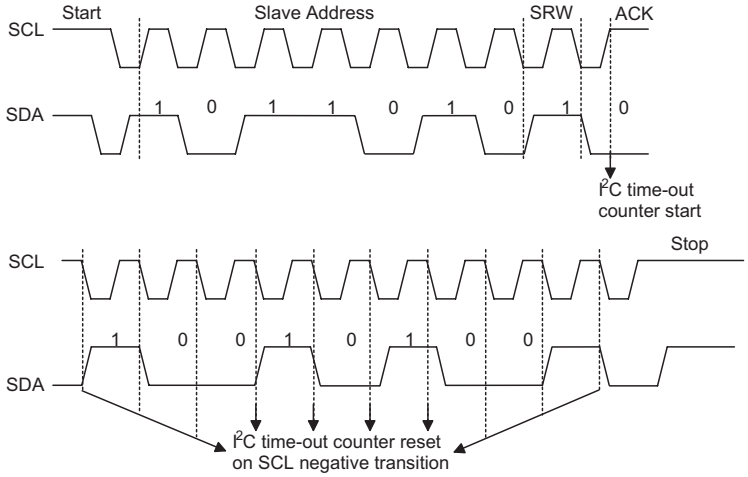
I²C 通信时序图



I²C 总线 ISR 流程图

I²C 超时控制

超时功能可减少 I²C 接收错误的时钟源而引起的锁死问题。如果连接到 I²C 总线的时钟源经过一段时间还未接收到，则在一定的超时周期后，I²C 电路和寄存器将复位。超时计数器在 I²C 总线“START”和“地址匹配”条件下开始计数，且在 SCL 下降沿清零。在下一个 SCL 下降沿到来之前，如果超时时间大于 I2CTOC 寄存器指定的超时周期，则超时发生。I²C“STOP”条件发生时超时功能终止。



I²C 超时时序图

当 I²C 超时计数器溢出时，计数器将停止计数，I2CTOEN 位被清零，且 I2CTOF 位被置高以表明超时计数器中断发生。超时计数器中断使用的也是 I²C 中断向量。当 I²C 超时发生时，I²C 内部电路会被复位，寄存器也将发生如下复位情况。

寄存器	I²C 超时发生后
SIMD, SIMA, SIMC0	无变化
SIMC1	复位到 POR 条件

I2CTOF 标志位由应用程序清零。共有 64 个溢出周期，可通过 I2CTOC 寄存器的 I2CTOS5~I2CTOS0 位进行选择。溢出周期可通过公式计算：((1~64)×(32/f_{SUB}))。由此可得溢出周期范围为 1ms~64ms。

- **I2CTOC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	I2CTOEN	I2CTOF	I2CTOS5	I2CTOS4	I2CTOS3	I2CTOS2	I2CTOS1	I2CTOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **I2CTOEN**：I²C 超时控制位
0：除能
1：使能

Bit 6 **I2CTOF**：I²C 超时标志位
0：超时未发生
1：超时发生

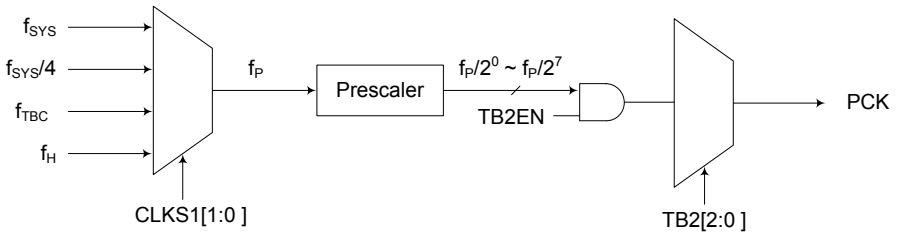
Bit 5~0 **I2CTOS5~I2CTOS0**：I²C 超时时间选择位
I²C 超时时钟源是 f_{SUB}/32
I²C 超时时间计算公式：(I2CTOS[5:0]+1)×(32/f_{SUB})

外围时钟输出

外围时钟输出功能使单片机能够为外部硬件提供和单片机时钟同步的时钟信号。

外围时钟操作

外围时钟输出引脚 PCK 与输入 / 输出脚共用，可以通过相关的共用引脚功能选择位来选择。外围时钟功能由 TBC2 寄存器的 TB2EN 位控制。外围时钟输出的时钟源来自系统时钟 f_{SYS}，系统时钟分频，高速振荡器 f_H 或 f_{TBC}，可通过 PSC1 寄存器的 CLKS11 和 CLKS10 位选择。TBC2 寄存器的 TB2EN 位是总的开 / 关控制位，当该位为高时使能外围时钟，为低时除能外围时钟。系统时钟所需要的分频比由 TBC2 寄存器中的 TB22，TB21 和 TB20 位来选择。如果当系统进入休眠模式外围时钟源关闭，将除能外围时钟输出功能。



外围时钟输出

外围时钟寄存器

有两个内部寄存器用于控制外围时钟输出的所有操作，分别是 PSC1 和 TBC2。

寄存器名称	位							
	7	6	5	4	3	2	1	0
PSC1	—	—	—	—	—	—	CLKS11	CLKS10
TBC2	TB2EN	—	—	—	—	TB22	TB21	TB20

PCK 寄存器列表

PSC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKS11	CLKS10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未定义，读为“0”

Bit 1~0　**CLKS11，CLKS10**：外围时钟源 f_P 选择位
00：f_{SYS}
01：$f_{SYS}/4$
10：f_{TBC}
11：f_H

TBC2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TB2EN	—	—	—	—	TB22	TB21	TB20
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB2EN**：外围时钟功能使能控制位
0：除能
1：使能

Bit 6~3 未定义，读为“0”

Bit 1~0 **TB22, TB21, TB20**：外围时钟输出分频选择位
000：f_P
001：$f_P/2$
010：$f_P/4$
011：$f_P/8$
100：$f_P/16$
101：$f_P/32$
110：$f_P/64$
111：$f_P/128$

SPIA 串行接口模块 – SPIA

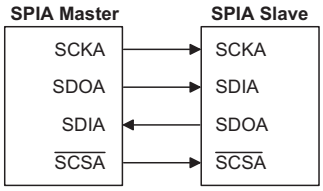
单片机内含一个独立的 SPI 功能。重要的是，不要将这个独立的 SPI 功能与 SIM 模块中的 SPI 功能混淆，其具体描述详见规格书的另一章节。这个独立的 SPI 功能命名为 SPIA 来区别 SIM 模块的 SPI 功能。

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，单片机既可以做为主机，也可以做为从机。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

SPI 接口是一个全双工串行同步数据传输器。SPI 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 的接口引脚与其他功能共用引脚。通过设定引脚共用功能选择寄存器的对应位，来选择 SPIA 接口引脚。SPIA 接口可以通过 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 SACSEN 位为“1”使能 $\overline{SCSA}$ 功能，设置 SACSEN 位为“0”，$\overline{SCSA}$ 引脚将浮空。

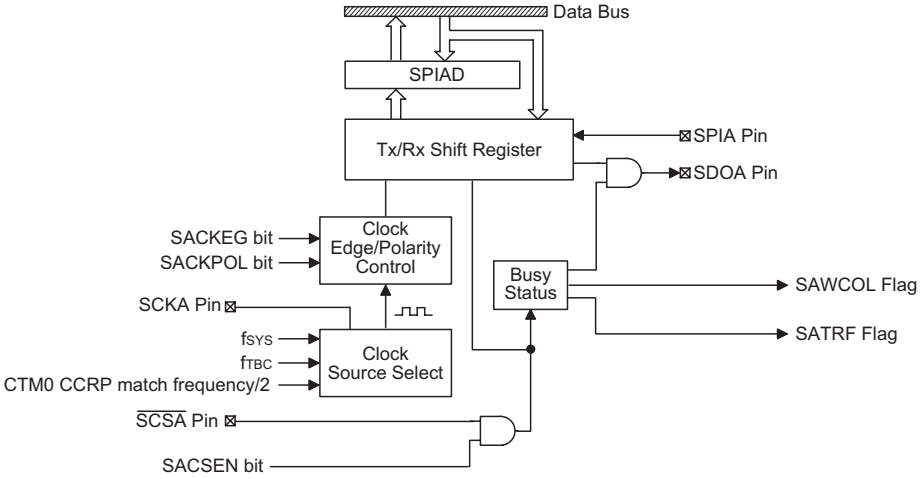


SPIA 主 / 从机连接方式

该系列单片机的 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

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 SACSEN 和 SPIAEN 位的状态。



SPIA 方框图

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 和两个控制寄存器 SPIAC0 和 SPIAC1。

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

SPIA 寄存器列表

SPIAD 寄存器

SPIAD 寄存器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 寄存器实现。

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 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	—
R/W	R/W	R/W	R/W	—	—	—	R/W	—
POR	1	1	1	—	—	—	0	—

Bit 7~5 **SASPI2~SASPI0**：SPIA 主机 / 从机时钟选择位
000：SPIA 主机模式，时钟为 $f_{SYS}/4$
001：SPIA 主机模式，时钟为 $f_{SYS}/16$
010：SPIA 主机模式，时钟为 $f_{SYS}/64$
011：SPIA 主机模式，时钟为 f_{TBC}
100：SPIA 主机模式，CTM0 CCRP 匹配频率 /2
101：SPIA 从机模式
11x：保留位

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 $\overline{SCSA}$ 脚将失去 SPIA 功能，SPIA 工作电流减小到最小值。此位为“1”时，SPIA 接口使能。

Bit 0 未定义，读为“0”

SPIAC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	SACKPOLB	SACKEG	SAMLS	SACSEN	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**：SPIA 时钟线的基础状态位
0：当时钟无效时，SCKA 口为高电平
1：当时钟无效时，SCKA 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，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 数据移位命令位
0：LSB
1：MSB
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 $\overline{\text{SCSA}}$ 引脚控制位
0：除能
1：使能
SACSEN 位用于 $\overline{\text{SCSA}}$ 引脚的使能 / 除能控制。此位为低时，$\overline{\text{SCSA}}$ 处于浮空状态。此位为高时，$\overline{\text{SCSA}}$ 使能并作为选择脚。

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。

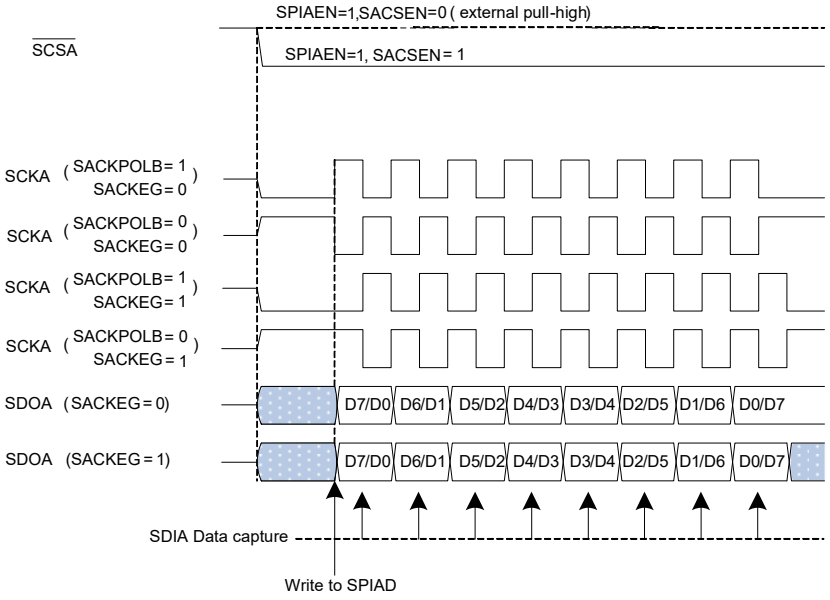
Bit 0　**SATRF**：SPIA 发送 / 接收结束标志位
0：数据正在发送
1：数据发送结束
SATRF 位为发送 / 接收结束标志位，当 SPIA 数据传输结束时，此位自动置为高，但须通过应用程序设置为“0”。此位也可用于产生中断。

SPIA 通信

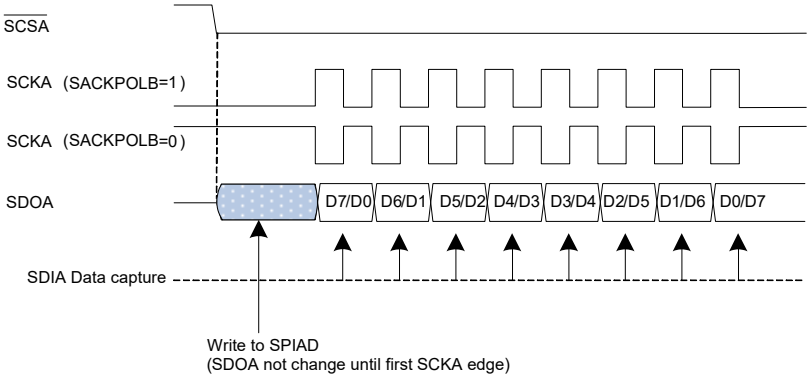
将 SPIAEN 设置为高，使能 SPIA 功能之后，单片机处于主机模式，当数据写入到寄存器 SPIAD 的同时传输 / 接收开始进行。数据传输完成时，SATRF 位将自动被置位但清除只能通过应用程序完成。单片机处于从机模式时，收到主机发来的信号之后，会传输 SPIAD 中的数据，而且在 SDIA 引脚上的数据也会被移位到 SPIAD 寄存器中。

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

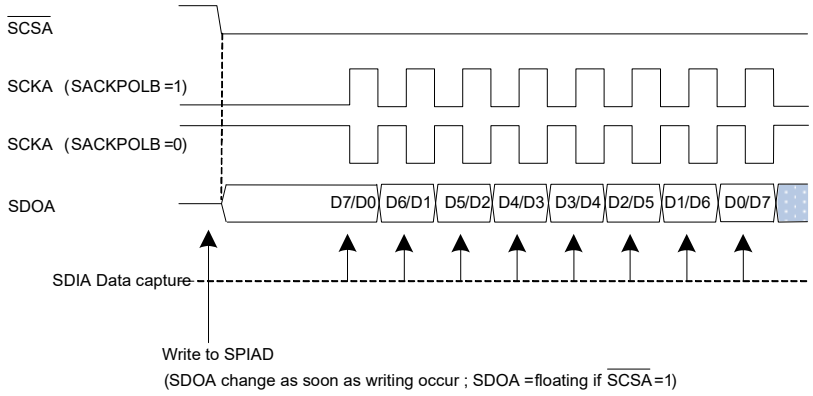
如果 SPIA 时钟源是开启的，SPIA 功能仍将继续执行。



SPIA 主机模式



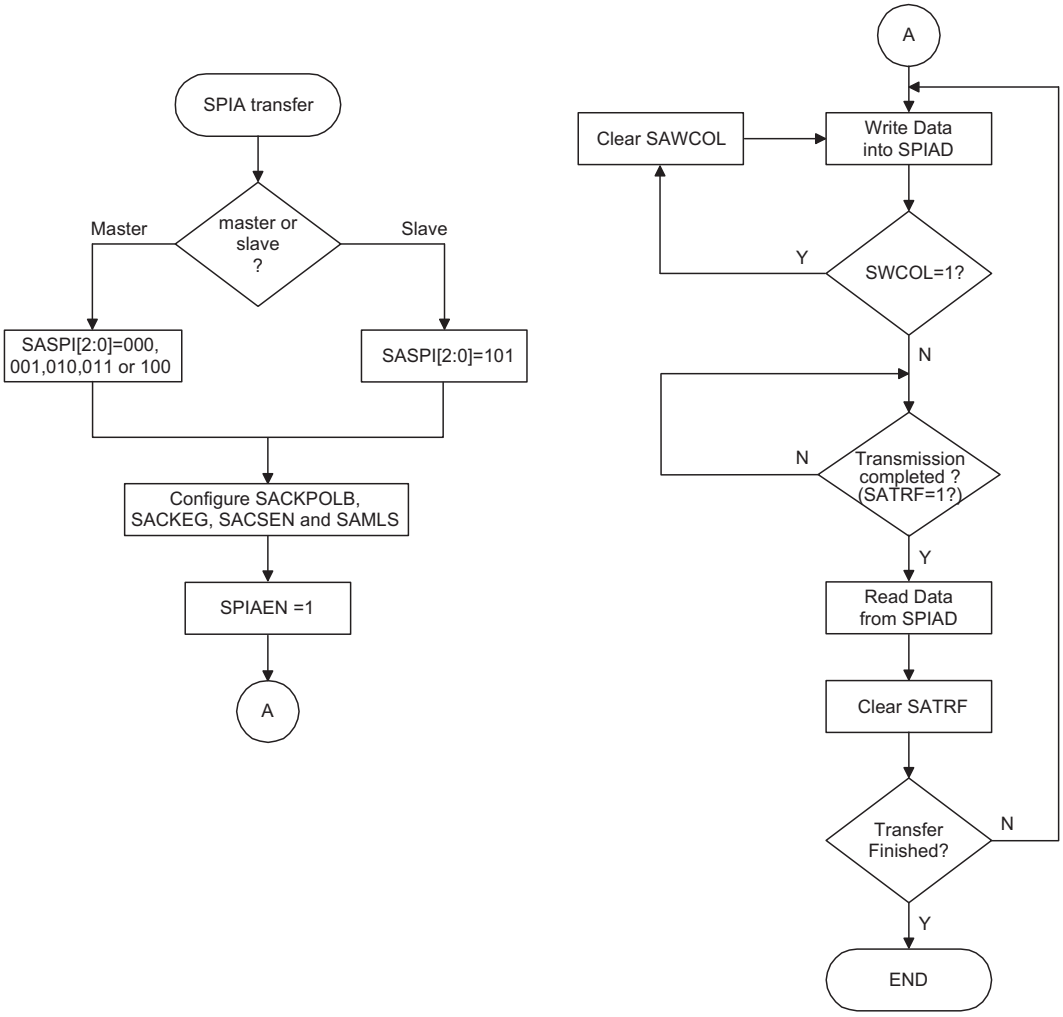
SPIA 从机模式 (SACKEG=0)



SPIA 从机模式 (SACKEG=1)

注：对于 SPIA 从机模式，如果 SPIAEN=1 且 SACSEN=0，SPIA 一直使能且忽略 SCSA 电平。

SPIA 主机 / 从机模式时序图



SPIA 传输控制流程图

SPIA 总线使能 / 除能

设置 SACSEN=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 (TXRX 缓存器) 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。

当 SPIA 总线除能，可通过共用引脚功能控制位设置 SCKA、SDIA、SDOA、$\overline{SCSA}$ 为普通 I/O 口或其它引脚共用功能。

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，SPIA 接口除能，$\overline{SCSA}$ 引脚浮空。如果 SPIAC0 寄存器中的 SACSEN 和 SPIAEN 位设置为高，使得 SDIA 信号线处于浮空状态且 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 寄存器的时钟极性选择位 SACKPOLB。从机模式中，SCKA 信号线处于浮空状态。如果 SPIAEN 位设置为低，SPIA 接口被除能，且 $\overline{SCSA}$、SDIA、SDOA 和 SCKA 将作为普通 I/O 口或其它引脚共用功能。主机模式中，主机将一直产生时钟信号。当数据被写入 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，选择时钟源和主机模式。
- 步骤 2
  设置 SACSEN 和 SAMLS 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 $\overline{SCSA}$ 信号线将数据输出。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。

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 SAMLS 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。等待主机时钟信号 SCKA 和 $\overline{\text{SCSA}}$ 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，而由应用程序来清除为零。在数据传输期间，如果写数据到 SPIAD 寄存器，此时数据冲突发生且不允许数据继续被写入。

LCD 驱动

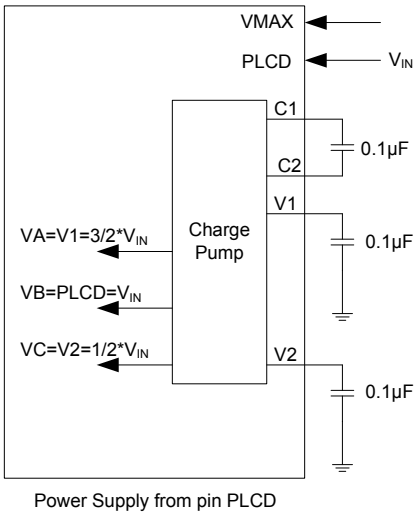
对于设计中带有 LCD 功能的大批量应用，选择定制而非较昂贵的基于字符的显示方式可以有效地降低成本。然而，驱动此类定制的显示器需要振幅及时间可变的 COM 和 SEG 信号，且需要很多特殊的考虑以正确地操作 LCD。此系列单片机有内部 LCD 信号产生电路及多种配置选项，可以自动地产生时间与振幅可变的信号直接驱动 LCD，与用户 LCD 的接口连接也相当容易。

此系列单片机都含有驱动使 LCD 各种类型显示的选项。下表显示此系列单片机带有的功能选项。

单片机型号	占空比	驱动输出	偏压	偏压类型	波形类型
HT67F60A	1/4	56×4	1/3	C 或 R	A 或 B
HT67F70A					

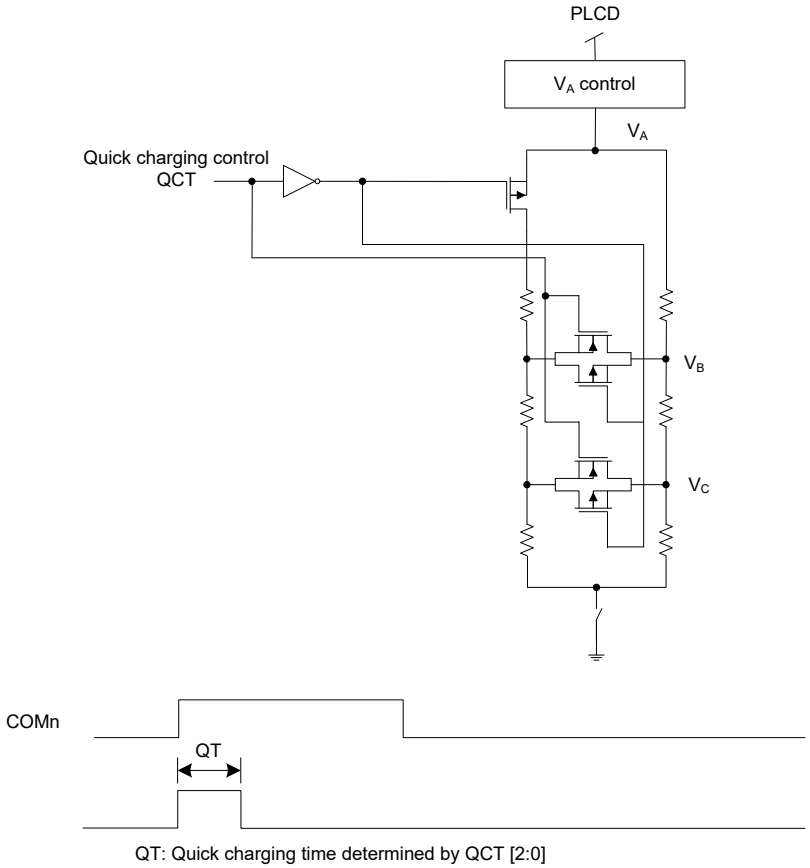
注：对于 48LQFP 封装类型，只有 R 型可用。

LCD 选项



注：VMAX 引脚必须连接最大电压值以使引脚防漏。

C 型偏置电压配置 – 1/3 Bias



注：当 R 型 LCD 除能时，DC 路径将被切换。

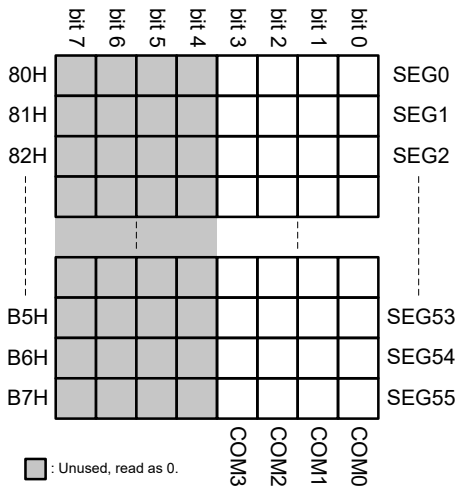
R 型偏压配置 – 1/3 Bias

LCD 存储器

数据存储器中有一部分区域是专门为 LCD 的显示数据而保留，即显示存储区。单片机内部显示驱动电路会自动读取任何写入此处的数据并据此产生 LCD 驱动信号。因此任何写入 LCD 存储器的数据，会立即映射到连接单片机的 LCD 显示器上。

由于 LCD 存储器地址与通用数据存储器地址重叠，因此 LCD 数据存储在独立的数据存储区 Sector 1 中。数据存储器 Sector 的选择是通过数据存储器中的一个特殊功能寄存器来完成的，即高字节寄存器 MP1H 或 MP2H。当要存取 LCD 存储器时，首先要将 MP1H 或 MP2H 的值设为“01H”来选择对 Sector 1 操作。此后，用户可以通过存储器指针低字节 MP1L 或 MP2L 使用间接寻址方式来对存储区进行操作。选择了 Sector 1 之后，使用 MP1L 或 MP2L 可以对以“80H”作为起始地址的存储区操作，就可以直接对显示存储区进行读或者写的操作了。直接寻址显示存储区可由相关扩展指令实现。

所附显示存储器映射图显示了此系列单片机内部显示存储器与 COM、SEG 端的显示映射。



LCD 存储器映射图

LCD 时钟源

LCD 时钟是由内部时钟源 f_{SUB} 通过内部分频电路进行 8 分频获得，其中 f_{SUB} 的时钟源可通过配置选项选择来自于 LIRC 或 LXT 振荡器。该方法用于产生理想的频率为 4kHz 的 LCD 时钟，以获得更好的 LCD 显示效果。

LCD 寄存器

LCD 控制寄存器 LCDC0 和 LCDC1 位于数据存储区，用于设置 LCD 驱动器的各种特性。LCD 寄存器的各个位可用来设置如 LCD 波形类型、偏压类型、偏压电阻选择和 LCD 的使能和除能。

寄存器 LCDC0 中的 LCDEN 位只有当单片机工作于正常模式、低速模式或空闲模式时才可以控制 LCD 的使能与除能。如果单片机处于休眠模式，则显示将一直处于除能状态。寄存器 LCDC0 中的 RSEL2~ RSEL0 位用于选择内部电阻来提供 LCD 适当的 R 型偏压电流。在应用中，选择匹配的 LCD 面板也可以降低偏压电流。此外，LCDC0 寄存器中的 TYPE 位是用于选择 A 型或 B 型的 LCD 控制信号，RCT 位是用来选择 R 型或 C 型偏压类型。

LCDC1 寄存器中的 PLCD3~PLCD0 位用来选择 R 型偏压电路中的 V_A 电压值，QCT2~QCT0 位用来选择快速充电时间周期。

寄存器名称	位							
	7	6	5	4	3	2	1	0
LCDC0	TYPE	RCT	—	—	RSEL2	RSEL1	RSEL0	LCDEN
LCDC1	QCT2	QCT1	QCT0	—	PLCD3	PLCD2	PLCD1	PLCD0

LCDC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	TYPE	RCT	—	—	RSEL2	RSEL1	RSEL0	LCDEN
R/W	R/W	R/W	—	—	R/W	R/W	R/W	R/W
POR	0	0	—	—	0	0	0	0

Bit 7 **TYPE**：LCD 波形类型选择位
0：A 型
1：B 型

Bit 6 **RCT**：LCD 偏压类型选择位
0：R 型
1：C 型

Bit 5~4 未定义，读为“0”

Bit 3~1 **RSEL2~RSEL0**：R 型偏压电阻选择位
000：1170 kΩ
001：225 kΩ
010：60 kΩ
011：快速充电模式 – 在 60 kΩ 和 1170 kΩ 之间切换
1xx：快速充电模式 – 在 60 kΩ 和 225 kΩ 之间切换

Bit 0 **LCDEN**：LCD 功能使能控制位
0：除能
1：使能
在正常模式、低速模式或空闲模式下，LCD 使能 / 除能由该位控制。在休眠模式下，LCD 一直关闭。

LCDC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	QCT2	QCT1	QCT0	—	PLCD3	PLCD2	PLCD1	PLCD0
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 **QCT2~QCT0**：R 型偏压快速充电时间周期选择位
000：1 t_{SUB}
001：2 t_{SUB}
010：3 t_{SUB}
011：4 t_{SUB}
100：5 t_{SUB}
101：6 t_{SUB}
110：7 t_{SUB}
111：8 t_{SUB}
注意，t_{SUB} 是 LCD 时钟源 f_{SUB} 的周期。

Bit 4 未定义，读为“0”

Bit 3~0 **PLCD3~PLCD0**：R 型偏压 V_A 电压选择位
0000：8/16 × V_{PLCD}
0001：9/16 × V_{PLCD}
0010：10/16 × V_{PLCD}
0011：11/16 × V_{PLCD}
0100：12/16 × V_{PLCD}
0101：13/16 × V_{PLCD}
0110：14/16 × V_{PLCD}
0111：15/16 × V_{PLCD}
1xxx：V_{PLCD}

LCD 电压源偏压

LCD 驱动器需要几种电压值以产生时间振幅可变的信号。单片机有 R 型和 C 型偏压，可通过软件控制位 RCT 选择。选择 C 型的偏压将使能内部充电泵。

R 型偏压

对于 R 型偏压，必须要在 PLCD 引脚上提供外部 LCD 电压源，以产生内部偏压电压。这个外部电压源可以是单片机的电压源也可以是其它电压源。对于 R 型 1/3 偏压的结构，要用到 V_{SS}、V_A、V_B 和 V_C 四种电压值。V_A 由 PLCD3~PLCD0 位选择等于不同的 V_{PLCD} 电压比例，范围值是 $8/16V_{PLCD}$~V_{PLCD}，V_B 等于 $V_{PLCD}\times2/3$，V_C 等于 $V_{PLCD}\times1/3$。

不同的内部偏压电阻阻值可由 LCDC0 寄存器中 RSEL2~RSEL0 位来选择。在 PLCD 引脚上的电压值决定偏流值。VMAX 引脚的连接方式取决于加在 PLCD 引脚上的电压，如果 V_{DD} 大于或等于 PLCD 引脚上的电压，则 VMAX 引脚连接到 VDD。需注意的是，PLCD R 型偏压值不应大于 VDD 引脚的电压值。如果使用 R 型偏压，则不需要连接有外部电容或电阻。

条件	VMAX 连接
$V_{DD} \geq V_{PLCD}$	VMAX 连接至 VDD
$V_{DD} < V_{PLCD}$	禁用条件

R 型偏压 VMAX 引脚连接

C 型偏压

对于 C 型偏压，在 PLCD 引脚上提供外部 LCD 电压源以产生内部偏置电压。C 型偏压使用内部充电泵电路，产生高于 PLCD 或 V2 引脚上的电压。这项特性在单片机提供电压小于 LCD 所需电压时非常有用。为了产生所需的电压值，必须要在引脚 C1 与 C2 之间连接充电泵电容。

对于 C 型 1/3 偏压选项，要用到 V_{SS}、V_A、V_B 和 V_C 四种电压值。V_A 由内部产生，其值等于 $V_{PLCD}\times3/2$，V_B 等于 V_{PCLD}，V_C 等于 $V_{PCLD}\times1/2$。VMAX 引脚的连接方式取决于加在 PLCD 引脚上的电压，必须注意的是这些充电泵所产生的内部电压不要超过 V_{DD} 的最大值 5.5V。

条件	VMAX 连接
$V_{DD} > V_{PLCD} \times 1.5$	VMAX 连接至 VDD
否则	VMAX 连接至 V1

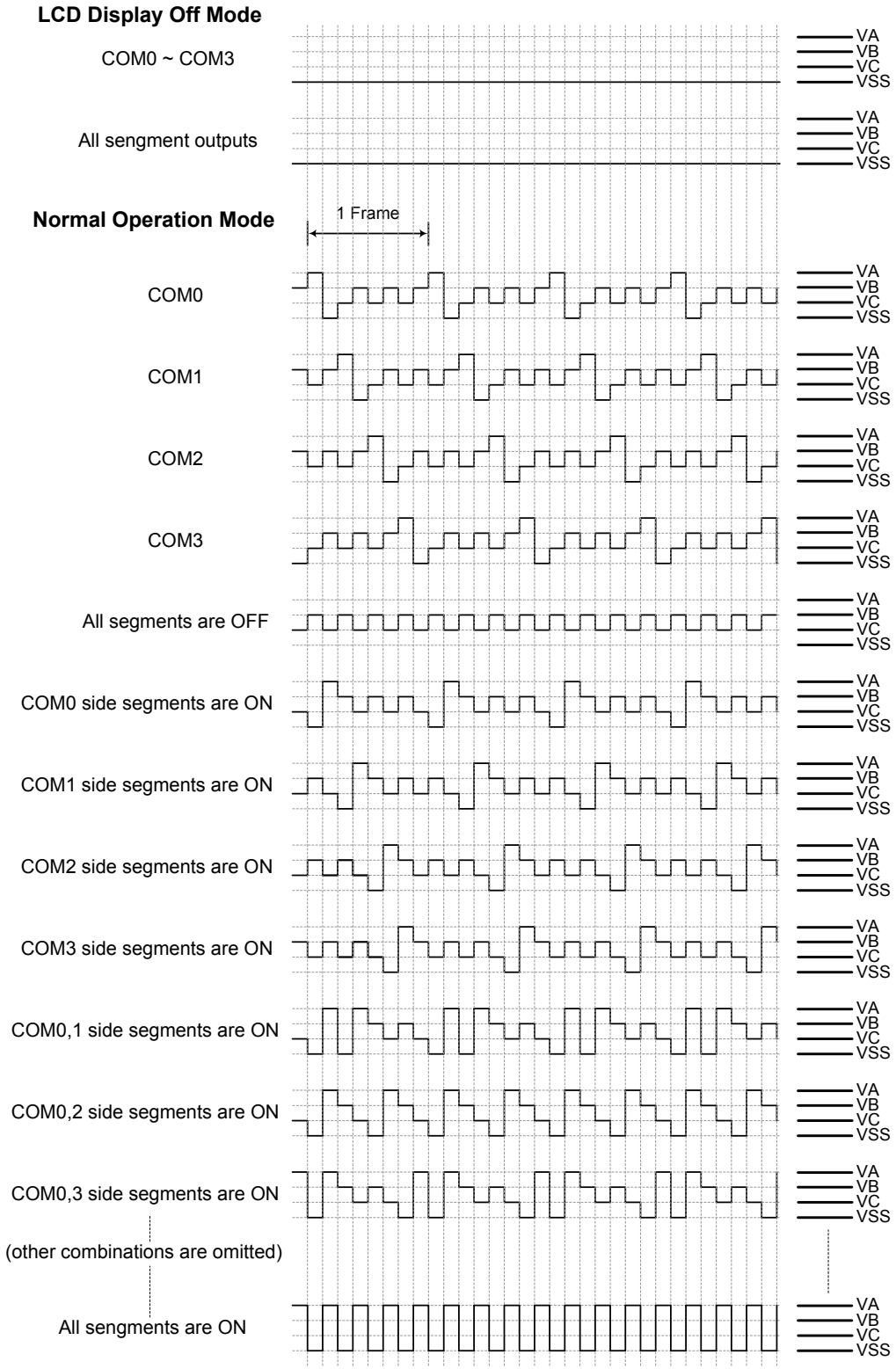
C 型偏压 VMAX 引脚连接

LCD 驱动输出

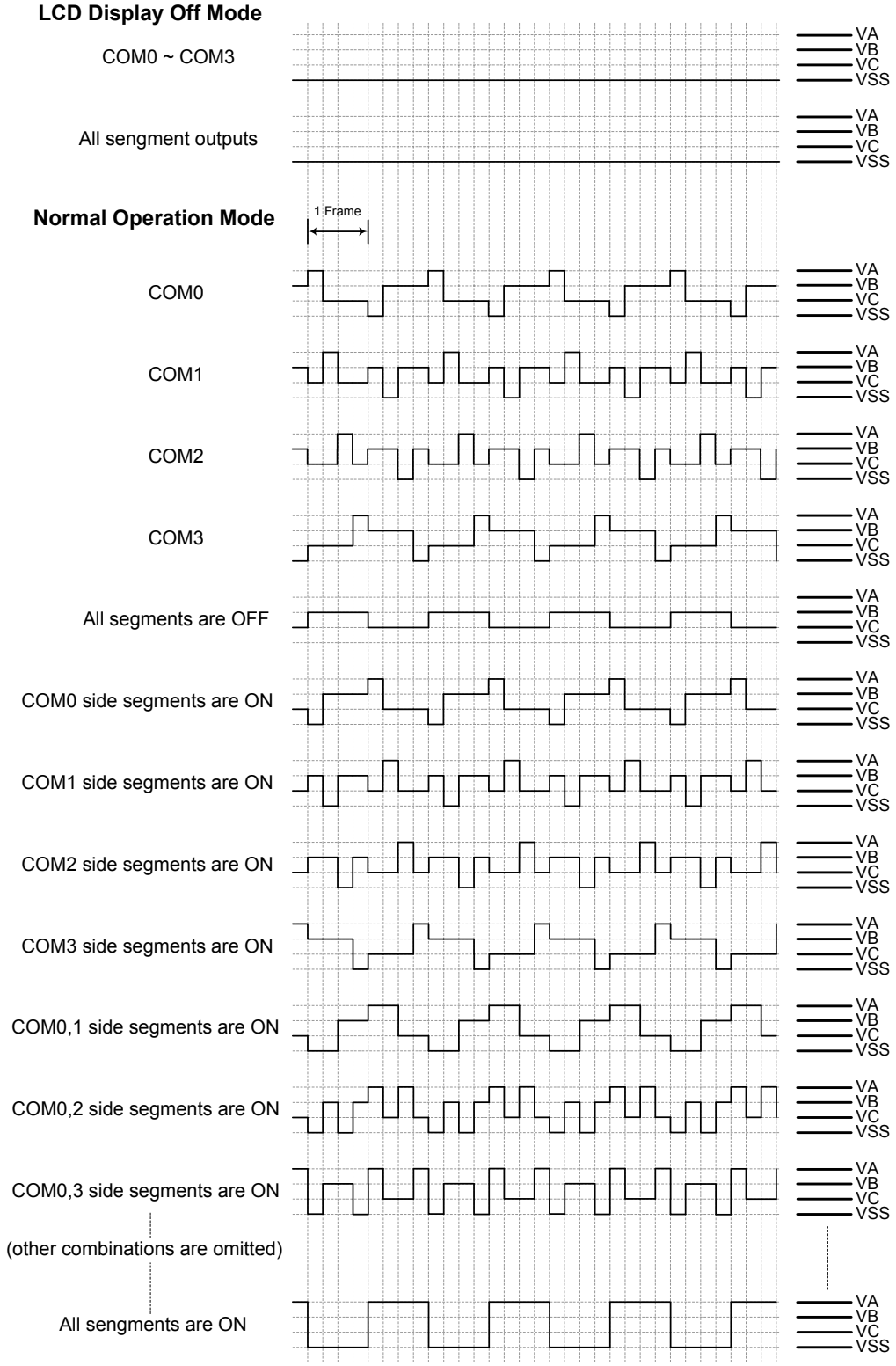
LCD 驱动器提供的 COM 和 SEG 输出数目，以及偏压和波形类型选项，取决于 LCD 控制位的设置。偏压类型可以通过软件控制位来选择 C 型偏置或者是 R 型偏置。

由于 LCD 基本性质的缘故，它们的像素点只能加上 AC 电压，如果加上 DC 电压，将会引起永久性的损害。因此 LCD 显示器的对比度由提供到每个像素的实际 RMS 电压控制，这个值相当于 COM 引脚上的 RMS 电压值减去 SEG 引脚上的电压值。RMS 电压必须大于 LCD 的饱和电压，以便能打开像素点，但同时也要小于阈值电压，以便能关闭像素点。

因为要将 DC 电压限制为 0 且以尽量少的连接数来控制尽可能多的像素点，因此需要产生时间振幅可变的信号供给 LCD 使用。这些时间与振幅都可变的信号由单片机内的 LCD 驱动电路自动产生。占空比决定使用 common 口的个数，也称为底板或 COMs。1/4 占空比的 COM 口个数为 4，因此该值定义了每个 LCD 信号帧内的时间片数。单片机提供两种类型的信号即 A 型和 B 型，通过寄存器 LCDC0 中的 TYPE 位加以选择。B 型提供较低频率的信号，然而，较低的频率可能引起闪烁，从而影响显示的清晰度。



LCD 驱动输出 – A 型 – 1/4 Duty, 1/3 Bias



LCD 驱动输出 – B 型 – 1/4 Duty, 1/3 Bias

注：对于 1/3 R 型偏压，$V_A=V_{PLCD}$，$V_B=V_{PLCD} \times 2/3$，$V_C=V_{PLCD} \times 1/3$
对于 1/3 C 型偏压，$V_A=V_{PLCD} \times 3/2$，$V_B=V_{PLCD}$，$V_C=V_{PLCD} \times 1/2$

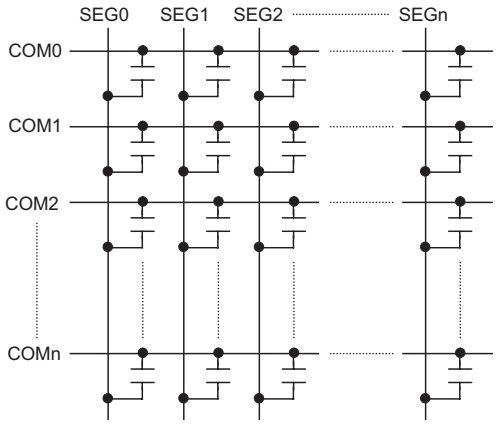
编程注意事项

LCD 编程时要注意几点，其中之一就是在单片机上电后，要保证 LCD 存储器正确地初始化。与通用数据存储器一样，在上电后，LCD 存储器的内容是未知的。由于 LCD 存储器的内容会映射到实际的 LCD，所以在上电后，为获得正确的显示图形，初始化此存储器内容是非常重要的。

在实际应用中，必须要考虑 LCD 的实际容性负载。对于单片机来说，LCD 的像素点一般可以看作电容性的负载，要确保所连接的像素点不能过多。这点对可以连接多个 LCD 像素点的 COM 口来说尤为重要。接下来的流程图描述 LCD 的等效电路。

另外还有一个要注意的就是当单片机进入空闲模式或低速模式后所发生的变化。LCDC0 控制寄存器的中的 LCD 使能控制位 LCDEN 会清零以降低功耗。当此位被清零，就会停止产生显示的驱动信号，并处于一种低功耗的空白显示的状态。

要注意当上电复位后，LCDEN 位会被清零，显示功能关闭。



LCD 面板等效电路

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定时器模块或 A/D 转换器有效，并且产生中断时，系统会暂时中止当前的程序而转到执行相对应的中断服务程序。此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 和 $\overline{\text{PINT}}$ 引脚动作产生，而内部中断由各种内部功能，如定时器模块、比较器、时基、LVD、EEPROM、SIM 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专用数据存储器中的一系列寄存器控制的。寄存器的数量由所选单片机的型号决定，但总的分为三类。第一类是 INTC0~INTC3 寄存器，用于设置基本的中断；第二类是 MFI0~MFI4 寄存器，用于设置多功能中断；最后一种有 INTEG 寄存器，用于设置外部中断边沿触发类型。

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

功能	使能位	请求标志	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=0~4
LVD	LVE	LVF	—
EEPROM 写操作	DEE	DEF	—
SIM	SIME	SIMF	—
$\overline{\text{PINT}}$ 脚	XPE	XPF	—
SPIA	SPIAE	SPIAF	—
CTM	CTMnPE	CTMnPF	n=0~1
	CTMnAE	CTMnAF	
STM	STMnPE	STMnPF	n=0~2
	STMnAE	STMnAF	
ETM	ETMPE	ETMPF	—
	ETMAE	ETMAF	
	ETMBE	ETMBF	

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	—	MF4F	INT3F	INT2F	—	MF4E	INT3E	INT2E
MFI0	STM0AF	STM0PF	CTM0AF	CTM0PF	STM0AE	STM0PE	CTM0AE	CTM0PE
MFI1	—	ETMBF	ETMAF	ETMPF	—	ETMBE	ETMAE	ETMPE
MFI2	SIMF	XPF	CTM1AF	CTM1PF	SIME	XPE	CTM1AE	CTM1PE
MFI3	—	SPIAF	DEF	LVF	—	SPIAE	DEE	LVE
MFI4	STM2AF	STM2PF	STM1AF	STM1PF	STM2AE	STM2PE	STM1AE	STM1PE

中断寄存器内容

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
01：上升沿
10：下降沿
11：双沿

Bit5~4 **INT2S1~INT2S0**：INT2 脚中断边沿控制位
00：除能
01：上升沿
10：下降沿
11：双沿

Bit 3~2 **INT1S1~INT1S0**：INT1 脚中断边沿控制位
00：除能
01：上升沿
10：下降沿
11：双沿

Bit 1~0 **INT0S1，INT0S0**：INT0 脚中断边沿控制位
00：除能
01：上升沿
10：下降沿
11：双沿

INTC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	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	—	MF4F	INT3F	INT2F	—	MF4E	INT3E	INT2E
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　**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　未定义，读为“0”

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

MFI0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	STM0AF	STM0PF	CTM0AF	CTM0PF	STM0AE	STM0PE	CTM0AE	CTM0PE
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 **CTM0AF**：CTM0 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4 **CTM0PF**：CTM0 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 **STM0PE**：STM0 比较器 P 匹配中断控制位
0：除能
1：使能

Bit 1 **CTM0AE**：CTM0 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0 **CTM0PE**：CTM0 比较器 P 匹配中断控制位
0：除能
1：使能

MFI1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	ETMBF	ETMAF	ETMPF	—	ETMBE	ETMAE	ETMPE
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　**ETMBF**：ETM 比较器 B 匹配中断请求标志位
0：无请求
1：中断请求

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

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

Bit 3　未定义，读为“0”

Bit 2　**ETMBE**：ETM 比较器 B 匹配中断控制位
0：除能
1：使能

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

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

MFI2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	SIMF	XPF	CTM1AF	CTM1PF	SIME	XPE	CTM1AE	CTM1PE
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 **XPF**：外围中断请求标志位
0：无请求
1：中断请求

Bit 5 **CTM1AF**：CTM1 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求

Bit 4 **CTM1PF**：CTM1 比较器 P 匹配中断请求标志位
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 **XPE**：外围中断控制位
0：除能
1：使能

Bit 1 **CTM1AE**：CTM1 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0 **CTM1PE**：CTM1 比较器 P 匹配中断控制位
0：除能
1：使能

MFI3 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	SPIAF	DEF	LVF	—	SPIAE	DEE	LVE
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　**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　未定义，读为“0”

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

MFI4 寄存器

Bit	7	6	5	4	3	2	1	0
Name	STM2AF	STM2PF	STM1AF	STM1PF	STM2AE	STM2PE	STM1AE	STM1PE
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 **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 4 **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 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 **STM1AE**：STM1 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0 **STM1PE**：STM1 比较器 P 匹配中断控制位
0：除能
1：使能

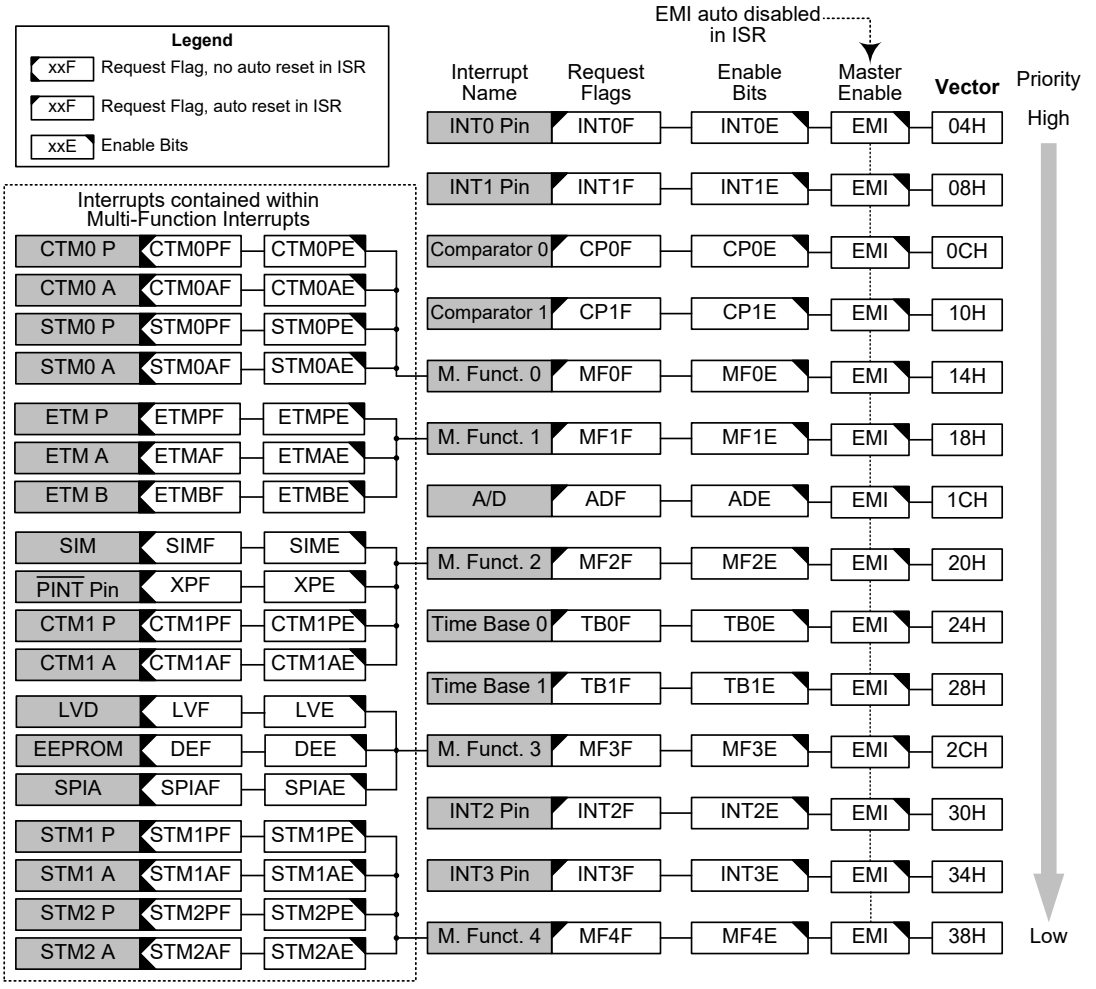
中断操作

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

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

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

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



中断结构

外部中断

通过 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和相关共用引脚选择位，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，上拉电阻仍保持有效。

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

比较器中断

比较器中断由内部比较器控制。当比较器 C0OUT 或 C1OUT 位状态改变，比较器中断请求标志 CP0F 或 CP1F 被置位，比较器中断请求产生。若要跳转到相应中断向量地址，总中断控制位 EMI 和比较器中断使能位 CP0E 和 CP1E 需先被置位。当中断使能，堆栈未满并且比较器输入产生一个比较器输出变化时，将调用比较器中断向量子程序。当响应中断服务子程序时，外部中断请求标志位会自动复位且 EMI 位会被清零以除能其它中断。

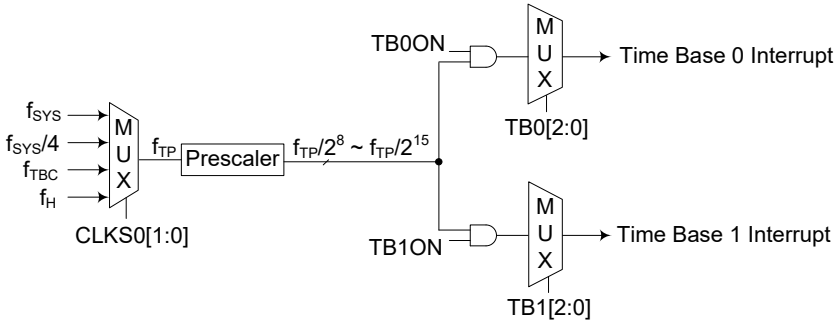
A/D 转换器中断

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

时基中断

时基中断提供一个固定周期的中断信号，由各自的定时器功能产生溢出信号控制。当各自的中断请求标志 TBnF 被置位时，中断请求发生。当总中断使能位 EMI 和时基使能位 TBnE 被置位，允许程序跳转到各自的中断向量地址。当中断使能，堆栈未满且时基溢出时，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_{TB} 来自内部时钟源 f_{TBC}，$f_{SYS}/4$，f_{SYS} 或 f_H。f_{TP} 输入时钟首先经过分频器，分频率由程序设置 TB0C 和 TB1C 寄存器相关位获取合适的分频值以提供更长的时基中断周期。相应的控制时基中断周期的时钟源可通过 PSC0 寄存器的 CLKS01 和 CLKS00 位选择。



PSC0 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKS01	CLKS00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 未定义，读为“0”

Bit 1~0 **CLKS01~ CLKS00**：时基分频器时钟源选择
00：f_{SYS}
01：$f_{SYS}/4$
10：f_{TBC}
11：f_H

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**：选择时基 0 溢出周期位
000：$f_{TP}/2^8$
001：$f_{TP}/2^9$
010：$f_{TP}/2^{10}$
011：$f_{TP}/2^{11}$
100：$f_{TP}/2^{12}$
101：$f_{TP}/2^{13}$
110：$f_{TP}/2^{14}$
111：$f_{TP}/2^{15}$

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**：选择时基 1 溢出周期位
000：$f_{TP}/2^8$
001：$f_{TP}/2^9$
010：$f_{TP}/2^{10}$
011：$f_{TP}/2^{11}$
100：$f_{TP}/2^{12}$
101：$f_{TP}/2^{13}$
110：$f_{TP}/2^{14}$
111：$f_{TP}/2^{15}$

多功能中断

此系列单片机中有多达五种多功能中断，与其它中断不同，它没有独立源，但由其它现有的中断源构成，即 TM 中断，LVD 中断，SIM 中断，EEPROM 写操作中断，外围中断和 SPIA 接口中断。

当多功能中断中任何一种中断请求标志 MFnF 被置位，多功能中断请求产生。当中断使能，堆栈未满，包括在多功能中断中的任意一个中断发生时，将调用多功能中断向量中的一个子程序。当响应中断服务子程序时，相关的多功能请求标志位会自动复位且 EMI 位会自动清零以除能其它中断。

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

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 I^2C 从机地址匹配，或 I^2C 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，堆栈未满且以上任意一种情况发生时，可跳转至相关多功能中断向量子程序中执行。当串行接口中断响应，EMI 将被自动清零以除能其它中断，多功能中断请求标志也可自动清除，但 SIMF 标志必须由应用程序清零。

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，SPIA 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、SPIA 接口中断使能位 SPIAE 和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被传送或接收完毕时，可跳转至相关多功能中断向量子程序中执行。当 SPIA 接口中断响应，EMI 将被自动清零以除能其它中断，多功能中断请求标志也可自动清除，但 SPIAF 标志必须由应用程序清零。

外围设备中断

外围设备中断和外部中断工作方式类似，属于多功能中断。当 $\overline{\text{PINT}}$ 引脚出现一个下降沿跳变，此时外围中断请求标志位 XPF 被置位，外围设备发生中断。要使程序跳转至相应中断向量地址，总中断控制位 EMI、外围设备中断使能位 XPE 和相应多功能中断使能位必须先被置位。当中断使能，堆栈未满且外围设备中断脚出现一个下降沿跳变，单片机将调用位于多功能中断向量相应子程序。当响应外围设备中断服务子程序时，EMI 位会被清零以除能其它中断，多功能中断请求标志位也将自动清除。

XPF 标志位不会自动复位，由应用程序清零。外围设备中断引脚与其它引脚共用，需正确地设置以使能外围设备中断脚。

LVD 中断

LVD 中断也属于多功能中断。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 和相应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，可跳转至相关多功能中断向量子程序中执行。当低电压中断响应，EMI 将被自动清零以除能其它中断，多功能中断请求标志也可自动清除，但 LVF 标志需在应用程序中手动清除。

EEPROM 中断

EEPROM 中断也属于多功能中断。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、EEPROM 中断使能位 DEE 和相应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，可跳转至相关多功能中断向量子程序中执行。当 EEPROM 中断响应，EMI 将被自动清零以除能其它中断，多功能中断请求标志也可自动清除，但 DEF 标志需在应用程序中手动清除。

TM 中断

简易型、标准型和增强型 TM 有两个或三个中断。分别来自比较器 A、B 或 P 匹配，属于多功能中断。所有的 TM 中断都包含有两个中断请求标志位及两个使能位。当 TM 比较器 P、A、B 匹配情况发生时，任意 TM 中断请求标志被置位，TM 中断请求产生。

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

中断唤醒功能

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

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

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

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

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

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

低电压检测 – LVD

此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 个固定的电压参考点。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。低电压检测会有一定的功耗，在不使用时可考虑关闭此功能，此举在功耗要求严格的电池供电应用中值得考虑。

LVDC 寄存器

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

Bit 7~6　未定义，读为“0”

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

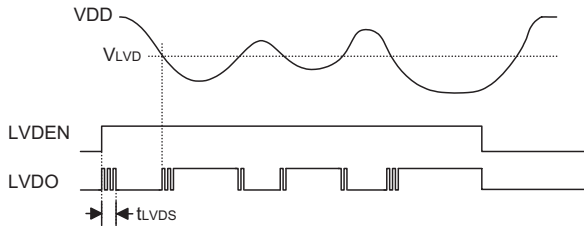
Bit 4　**LVDEN**：低电压检测控制位
0：除能
1：使能

Bit 3　**VBGEN**：Bandgap 缓冲控制位
0：除能
1：使能
注意，当 LVD 或 LVR 功能使能或 VBGEN 位置 1 时，Bandgap 使能。

Bit 2~0　**VLVD2~VLVD0**：LVD 电压选择位
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。若 LVDEN 位为高，当单片机掉电时低电压检测器保持有效状态。低电压检测器使能后，读取 LVDO 位前，电路稳定需要一定的延时 t_{LVDS}。注意，V_{DD} 电压可能上升或下降比较缓慢，在 V_{LVD} 电压值附近时，LVDO 位可能有多种变化。



LVD 操作

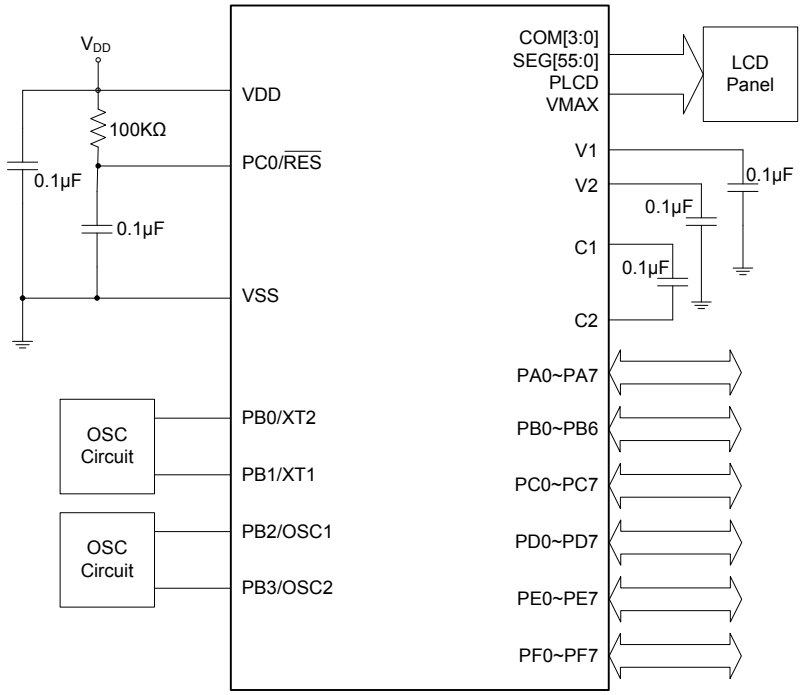
低电压检测器也有自己的中断功能，也是属于多功能中断的一种，它是除了轮询 LVDO 位之外的另一种检测低电压的方法。中断条件产生置位 LVDO 并延时 t_{LVD} 后，中断产生。若 LVDEN 位为高，当单片机掉电时低电压检测器保持有效状态。此种情况下，若 V_{DD} 降至小于 LVD 预置电压值时，中断请求标志位 LVF 将被置位，中断产生，单片机将被从休眠或空闲模式中唤醒。若不要求低电压检测的唤醒功能使能，在单片机进入休眠或空闲模式前应将 LVF 标志置为高。

配置选项

配置选项在烧写程序时写入芯片。通过 HT-IDE 的软件开发环境，使用者在开发过程中可以选择配置选项。当配置选项烧入单片机后，无法再通过应用程序修改。所有位必须按系统的需要定义，具体内容可参考下表：

序号	选项
1	高速振荡器类型选择—f_H HXT 或 HIRC
2	低速振荡器类型选择— f_{SUB} LXT 或 LIRC
3	高速内部 RC 振荡器频率选择—f_{HIRC} 4MHz、8MHz 或 12MHz

应用电路


VDD
100KΩ
0.1μF
0.1μF
PC0/RES
VSS
OSC Circuit
PB0/XT2
PB1/XT1
OSC Circuit
PB2/OSC1
PB3/OSC2
COM[3:0]
SEG[55:0]
PLCD
VMAX
LCD Panel
V1
V2
C1
C2
0.1μF
0.1μF
0.1μF
PA0~PA7
PB0~PB6
PC0~PC7
PD0~PD7
PE0~PE7
PF0~PF7

指令集

简介

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

为了更加容易理解各种各样的指令码，接下来按功能分组介绍它们。

指令周期

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

数据的传送

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

算术运算

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

逻辑和移位运算

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

分支和控制转换

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

位运算

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

查表运算

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

其它运算

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

指令集概要

当要操作的数据存储器位于数据存储器 Sector 0 时，下表说明了与数据存储器存取有关的指令。

惯例

x：立即数
m：数据存储器地址
A：累加器
i：第 0~7 位
addr：程序存储器地址

助记符	说明	指令周期	影响标志位
算术运算			
ADD A,[m]	ACC 与数据存储器相加，结果放入 ACC	1	Z, C, AC, OV, SC
ADDM A,[m]	ACC 与数据存储器相加，结果放入数据存储器	1[注]	Z, C, AC, OV, SC
ADD A, x	ACC 与立即数相加，结果放入 ACC	1	Z, C, AC, OV, SC
ADC A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	1	Z, C, AC, OV, SC
ADCM A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	1[注]	Z, C, AC, OV, SC
SUB A, x	ACC 与立即数相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SUB A,[m]	ACC 与数据存储器相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SUBM A,[m]	ACC 与数据存储器相减，结果放入数据存储器	1[注]	Z, C, AC, OV, SC, CZ
SBC A, x	ACC 与立即数、进位标志相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SBC A,[m]	ACC 与数据存储器、进位标志相减，结果放入 ACC	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	1[注]	Z, C, AC, OV, SC, CZ
DAA [m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	1[注]	C
逻辑运算			
AND A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	1	Z
OR A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	1	Z
XOR A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	1	Z
ANDM A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	1[注]	Z
ORM A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	1[注]	Z
XORM A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	1[注]	Z
AND A, x	ACC 与立即数做“与”运算，结果放入 ACC	1	Z
OR A, x	ACC 与立即数做“或”运算，结果放入 ACC	1	Z
XOR A, x	ACC 与立即数做“异或”运算，结果放入 ACC	1	Z
CPL [m]	对数据存储器取反，结果放入数据存储器	1[注]	Z
CPLA [m]	对数据存储器取反，结果放入 ACC	1	Z
递增和递减			
INCA [m]	递增数据存储器，结果放入 ACC	1	Z
INC [m]	递增数据存储器，结果放入数据存储器	1[注]	Z
DECA [m]	递减数据存储器，结果放入 ACC	1	Z
DEC [m]	递减数据存储器，结果放入数据存储器	1[注]	Z

助记符	说明	指令周期	影响标志位
移位			
RRA [m]	数据存储器右移一位，结果放入 ACC	1	无
RR [m]	数据存储器右移一位，结果放入数据存储器	1[注]	无
RRCA [m]	带进位将数据存储器右移一位，结果放入 ACC	1	C
RRC [m]	带进位将数据存储器右移一位，结果放入数据存储器	1[注]	C
RLA [m]	数据存储器左移一位，结果放入 ACC	1	无
RL [m]	数据存储器左移一位，结果放入数据存储器	1[注]	无
RLCA [m]	带进位将数据存储器左移一位，结果放入 ACC	1	C
RLC [m]	带进位将数据存储器左移一位，结果放入数据存储器	1[注]	C
数据传送			
MOV A,[m]	将数据存储器送至 ACC	1	无
MOV [m],A	将 ACC 送至数据存储器	1[注]	无
MOV A, x	将立即数送至 ACC	1	无
位运算			
CLR [m].i	清除数据存储器的位	1[注]	无
SET [m].i	置位数据存储器的位	1[注]	无
转移			
JMP addr	无条件跳转	2	无
SZ [m]	如果数据存储器为零，则跳过下一条指令	1[注]	无
SZA [m]	数据存储器送至 ACC，如果内容为零，则跳过下一条指令	1[注]	无
SNZ [m]	如果数据存储器不为零，则跳过下一条指令	1[注]	无
SZ [m].i	如果数据存储器的第 i 位为零，则跳过下一条指令	1[注]	无
SNZ [m].i	如果数据存储器的第 i 位不为零，则跳过下一条指令	1[注]	无
SIZ [m]	递增数据存储器，如果结果为零，则跳过下一条指令	1[注]	无
SDZ [m]	递减数据存储器，如果结果为零，则跳过下一条指令	1[注]	无
SIZA [m]	递增数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	1[注]	无
SDZA [m]	递减数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	1[注]	无
CALL addr	子程序调用	2	无
RET	从子程序返回	2	无
RET A, x	从子程序返回，并将立即数放入 ACC	2	无
RETI	从中断返回	2	无
查表			
TABRD [m]	读取特定页的 ROM 内容，并送至数据存储器和 TBLH	2[注]	无
TABRDL [m]	读取最后页的 ROM 内容，并送至数据存储器和 TBLH	2[注]	无
ITABRD [m]	读表指针 TBLP 自加，读取特定页的 ROM 内容，并送至数据存储器和 TBLH	2[注]	无
ITABRDL [m]	读表指针 TBLP 自加，读取最后页的 ROM 内容，并送至数据存储器和 TBLH	2[注]	无
其它指令			
NOP	空指令	1	无
CLR [m]	清除数据存储器	1[注]	无
SET [m]	置位数据存储器	1[注]	无

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

注：1. 对跳转指令而言，如果比较的结果牵涉到跳转即需 2 个周期，如果没有发生跳转，则只需一个周期。
2. 任何指令若要改变 PCL 的内容将需要 2 个周期来执行。

扩展指令集

扩展指令用来提供更大范围的数据存储器寻址。当被存取的数据存储器位于 Sector 0 之外的任何数据存储器 Sector，扩展指令可直接存取数据存储器而无需使用间接寻址，此举不仅可节省 Flash 存储器空间的使用，同时可提高 CPU 执行效率。

助记符	说明	指令周期	影响标志位
算术运算			
LADD A,[m]	ACC 与数据存储器相加，结果放入 ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	ACC 与数据存储器相加，结果放入数据存储器	2注	Z, C, AC, OV, SC
LADC A,[m]	ACC 与数据存储器、进位标志相加，结果放入 ACC	2	Z, C, AC, OV, SC
LADCM A,[m]	ACC 与数据存储器、进位标志相加，结果放入数据存储器	2注	Z, C, AC, OV, SC
LSUB A,[m]	ACC 与数据存储器相减，结果放入 ACC	2	Z, C, AC, OV, SC, CZ
LSUBM A,[m]	ACC 与数据存储器相减，结果放入数据存储器	2注	Z, C, AC, OV, SC, CZ
LSBC A,[m]	ACC 与数据存储器、进位标志相减，结果放入 ACC	2	Z, C, AC, OV, SC, CZ
LSBCM A,[m]	ACC 与数据存储器、进位标志相减，结果放入数据存储器	2注	Z, C, AC, OV, SC, CZ
LDAA [m]	将加法运算中放入 ACC 的值调整为十进制数，并将结果放入数据存储器	2注	C
逻辑运算			
LAND A,[m]	ACC 与数据存储器做“与”运算，结果放入 ACC	2	Z
LOR A,[m]	ACC 与数据存储器做“或”运算，结果放入 ACC	2	Z
LXOR A,[m]	ACC 与数据存储器做“异或”运算，结果放入 ACC	2	Z
LANDM A,[m]	ACC 与数据存储器做“与”运算，结果放入数据存储器	2注	Z
LORM A,[m]	ACC 与数据存储器做“或”运算，结果放入数据存储器	2注	Z
LXORM A,[m]	ACC 与数据存储器做“异或”运算，结果放入数据存储器	2注	Z
LCPL [m]	对数据存储器取反，结果放入数据存储器	2注	Z
LCPLA [m]	对数据存储器取反，结果放入 ACC	2	Z
递增和递减			
LINCA [m]	递增数据存储器，结果放入 ACC	2	Z
LINC [m]	递增数据存储器，结果放入数据存储器	2注	Z
LDECA [m]	递减数据存储器，结果放入 ACC	2	Z
LDEC [m]	递减数据存储器，结果放入数据存储器	2注	Z
移位			
LRRA [m]	数据存储器右移一位，结果放入 ACC	2	无
LRR [m]	数据存储器右移一位，结果放入数据存储器	2注	无
LRRCA [m]	带进位将数据存储器右移一位，结果放入 ACC	2	C
LRRC [m]	带进位将数据存储器右移一位，结果放入数据存储器	2注	C
LRLA [m]	数据存储器左移一位，结果放入 ACC	2	无
LRL [m]	数据存储器左移一位，结果放入数据存储器	2注	无
LRLCA [m]	带进位将数据存储器左移一位，结果放入 ACC	2	C
LRLC [m]	带进位将数据存储器左移一位，结果放入数据存储器	2注	C
数据传送			
LMOV A,[m]	将数据存储器送至 ACC	2	无
LMOV [m],A	将 ACC 送至数据存储器	2注	无

助记符		说明	指令周期	影响标志位
位运算				
LCLR	[m].i	清除数据存储器的位	2 注	无
LSET	[m].i	置位数据存储器的位	2 注	无
转移				
LSZ	[m]	如果数据存储器为零，则跳过下一条指令	2 注	无
LSZA	[m]	数据存储器送至 ACC，如果内容为零，则跳过下一条指令	2 注	无
LSNZ	[m]	如果数据存储器不为零，则跳过下一条指令	2 注	无
LSZ	[m].i	如果数据存储器的第 i 位为零，则跳过下一条指令	2 注	无
LSNZ	[m].i	如果数据存储器的第 i 位不为零，则跳过下一条指令	2 注	无
LSIZ	[m]	递增数据存储器，如果结果为零，则跳过下一条指令	2 注	无
LSDZ	[m]	递减数据存储器，如果结果为零，则跳过下一条指令	2 注	无
LSIZA	[m]	递增数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	2 注	无
LSDZA	[m]	递减数据存储器，将结果放入 ACC，如果结果为零，则跳过下一条指令	2 注	无
查表				
LTABRD	[m]	读取特定页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
LTABRDL	[m]	读取最后页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
LITABRD	[m]	读表指针 TBLP 自加，读取特定页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
LITABRDL	[m]	读表指针 TBLP 自加，读取最后页的 ROM 内容，并送至数据存储器和 TBLH	3 注	无
其它指令				
LCLR	[m]	清除数据存储器	2 注	无
LSET	[m]	置位数据存储器	2 注	无
LSWAP	[m]	交换数据存储器的高低字节，结果放入数据存储器	2 注	无
LSWAPA	[m]	交换数据存储器的高低字节，结果放入 ACC	2	无

注：1. 对扩展跳转指令而言，如果比较的结果牵涉到跳转即需 3 个周期，如果没有发生跳转，则只需两个周期。
2. 任何扩展指令若要改变 PCL 的内容将需要 3 个周期来执行。

指令定义

ADC A, [m] Add Data Memory to ACC with Carry
指令说明 将指定的数据存储器、累加器内容以及进位标志相加，结果存放到累加器。
功能表示 ACC ← ACC + [m] + C
影响标志位 OV、Z、AC、C、SC

ADCM A, [m] Add ACC to Data Memory with Carry
指令说明 将指定的数据存储器、累加器内容和进位标志位相加，结果存放到指定的数据存储器。
功能表示 [m] ←ACC + [m] + C
影响标志位 OV、Z、AC、C、SC

ADD A, [m] Add Data Memory to ACC
指令说明 将指定的数据存储器和累加器内容相加，结果存放到累加器。
功能表示 ACC ← ACC + [m]
影响标志位 OV、Z、AC、C、SC

ADD A, x Add immediate data to ACC
指令说明 将累加器和立即数相加，结果存放到累加器。
功能表示 ACC ← ACC + x
影响标志位 OV、Z、AC、C、SC

ADDM A, [m] Add ACC to Data Memory
指令说明 将指定的数据存储器和累加器内容相加，结果存放到指定的数据存储器。
功能表示 [m] ←ACC + [m]
影响标志位 OV、Z、AC、C、SC

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

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

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

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

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

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

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

CPL [m] Complement Data Memory
指令说明 将指定数据存储器中的每一位取逻辑反，
相当于从 1 变 0 或 0 变 1。
功能表示 [m] ← $[\overline{m}]$
影响标志位 Z

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

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

DEC [m] Decrement Data Memory
指令说明 将指定数据存储器内容减 1。
功能表示 [m] ← [m] – 1
影响标志位 Z

DECA [m] Decrement Data Memory with result in ACC
指令说明 将指定数据存储器的内容减 1，把结果存放回累加器并保持指定数据存储器的内容不变。
功能表示 ACC ← [m] – 1
影响标志位 Z

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

INC [m] Increment Data Memory
指令说明 将指定数据存储器的内容加 1。
功能表示 [m] ← [m] + 1
影响标志位 Z

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

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

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

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

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

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

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

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

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

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

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

RETI Return from interrupt

指令说明 将堆栈寄存器中的程序计数器值恢复且中断功能通过设置 EMI 位重新使能。EMI 是控制中断使能的主控制位。如果在执行 RETI 指令之前还有中断未被响应，则这个中断将在返回主程序之前被响应。

功能表示 Program Counter ←Stack
EMI ← 1

影响标志位 无

RL [m] Rotate Data Memory left

指令说明 将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位。

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

影响标志位 无

RLA [m] Rotate Data Memory left with result in ACC

指令说明 将指定数据存储器的内容左移 1 位，且第 7 位移到第 0 位，结果送到累加器，而指定数据存储器的内容保持不变。

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

影响标志位 无

RLC [m] Rotate Data Memory Left through Carry

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

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

影响标志位 C

RLC A [m] Rotate Data Memory left through Carry with result in ACC

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

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

影响标志位 C

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

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

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

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

SBC A, [m] Subtract Data Memory from ACC with Carry
指令说明 将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – [m] – $\overline{C}$
影响标志位 OV、Z、AC、C、SC、CZ

SBC A, x Subtract immediate data from ACC with Carry
指令说明 将累加器减去立即数以及进位标志的反，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – [m] – $\overline{C}$
影响标志位 OV、Z、AC、C、SC、CZ

SBCM A, [m] Subtract Data Memory from ACC with Carry and result in Data Memory
指令说明 将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 [m] ← ACC – [m] – $\overline{C}$
影响标志位 OV、Z、AC、C、SC、CZ

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

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

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

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

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

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

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

SNZ [m] Skip if Data Memory is not 0
指令说明 指定数据存储器的内容会先被读出，后又被重新写入指定数据存储器内。判断指定存储器，若不为 0，则程序跳过下一条指令执行。由于取得下一个指令时会要求插入一个空指令周期，所以此指令为2个周期的指令。如果结果为0，则程序继续执行下一条指令。
功能表示 如果 [m]≠0，跳过下一条指令执行
影响标志位 无

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

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

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

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

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

SZ [m] Skip if Data Memory is 0

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

功能表示 如果 [m]=0，跳过下一条指令执行

影响标志位 无

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

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

功能表示 ACC ←[m]，如果 [m]=0，跳过下一条指令执行

影响标志位 无

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

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

功能表示 如果 [m].i=0，跳过下一条指令执行

影响标志位 无

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

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

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

影响标志位 无

TABRDL [m] Read table (last page) to TBLH and Data Memory

指令说明 将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定数据存储器且将高字节移至 TBLH。

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

影响标志位 无

ITABRD [m] Increment table pointer low byte first and read table (specific page) to TBLH and data memory

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

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

影响标志位 无

ITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and data memory

指令说明 自加表格指针低字节 TBLP，将表格指针 TBLP 所指的程序代码低字节 (最后一页) 移至指定的数据存储器且将高字节移至 TBLH。

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

影响标志位 无

XOR A, [m] Logical XOR Data Memory to ACC

指令说明 将累加器的数据和指定的数据存储器内容逻辑异或，结果存放到累加器。

功能表示 ACC ← ACC“XOR”[m]

影响标志位 Z

XORM A, [m] Logical XOR ACC to Data Memory

指令说明 将累加器的数据和指定的数据存储器内容逻辑异或，结果放到数据存储器。

功能表示 [m] ← ACC“XOR”[m]

影响标志位 Z

XOR A, x Logical XOR immediate data to ACC

指令说明 将累加器的数据与立即数逻辑异或，结果存放到累加器。

功能表示 ACC ← ACC“XOR”x

影响标志位 Z

扩展指令定义

扩展指令被用来直接存取存储在任何数据存储器 Sector 中的数据。

LADC A, [m] Add Data Memory to ACC with Carry
指令说明 将指定的数据存储器、累加器内容以及进位标志相加，结果存放到累加器。
功能表示 ACC ← ACC + [m] + C
影响标志位 OV、Z、AC、C、SC

LADCM A, [m] Add ACC to Data Memory with Carry
指令说明 将指定的数据存储器、累加器内容和进位标志位相加，结果存放到指定的数据存储器。
功能表示 [m] ←ACC + [m] + C
影响标志位 OV、Z、AC、C、SC

LADD A, [m] Add Data Memory to ACC
指令说明 将将指定的数据存储器和累加器内容相加，结果存放到累加器。
功能表示 ACC ← ACC + [m]
影响标志位 OV、Z、AC、C、SC

LADDM A, [m] Add ACC to Data Memory
指令说明 将指定的数据存储器和累加器内容相加，结果存放到指定的数据存储器。
功能表示 [m] ←ACC + [m]
影响标志位 OV、Z、AC、C、SC

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

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

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

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

LCPL [m] Complement Data Memory
指令说明 将指定数据存储器中的每一位取逻辑反，
相当于从 1 变 0 或 0 变 1。
功能表示 [m] ← $[\overline{m}]$
影响标志位 Z

LCPLA [m] Complement Data Memory with result in ACC
指令说明 将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1，结果被存放回累加器且数据寄存器的内容保持不变。
功能表示 ACC←$[\overline{m}]$
影响标志位 Z

LDAA [m] Decimal-Adjust ACC for addition with result in Data Memory
指令说明 将累加器中的内容转换为 BCD (二进制转成十进制) 码。如果低四位的值大于“9”或 AC=1，那么 BCD 调整就执行对低四位加“6”，否则低四位保持不变；如果高四位的值大于“9”或C=1，那么BCD调整就执行对高四位加“6”。BCD 转换实质上是根据累加器和标志位执行 00H，06H，60H 或 66H 的加法运算，结果存放到数据存储器。只有进位标志位 C 受影响，用来指示原始 BCD 的和是否大于 100，并可以进行双精度十进制数的加法运算。
功能表示 [m] ← ACC + 00H 或
[m] ← ACC + 06H 或
[m] ← ACC + 60H 或
[m] ← ACC + 66H
影响标志位 C

LDEC [m] Decrement Data Memory
指令说明 将指定数据存储器的内容减 1。
功能表示 [m] ← [m] – 1
影响标志位 Z

LDECA [m] Decrement Data Memory with result in ACC
指令说明 将指定数据存储器的内容减 1，把结果存放回累加器并保持指定数据存储器的内容不变。
功能表示 ACC ← [m] – 1
影响标志位 Z

LINC [m] Increment Data Memory
指令说明 将指定数据存储器的内容加 1。
功能表示 [m] ← [m] + 1
影响标志位 Z

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

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

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

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

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

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

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

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

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

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

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

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

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

LSBC A, [m] Subtract Data Memory from ACC with Carry
指令说明 将累加器减去指定数据存储器的内容以及进位标志的反，结果存放到累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – [m] – $\overline{C}$
影响标志位 OV、Z、AC、C、SC、CZ

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

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

功能表示 [m] ← ACC – [m] – $\overline{C}$

影响标志位 OV、Z、AC、C、SC、CZ

LSDZ [m] Skip if Decrement Data Memory is 0

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

功能表示 [m] ← [m] – 1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

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

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

功能表示 ACC ← [m] – 1，如果 ACC=0 跳过下一条指令执行

影响标志位 无

LSET [m] Set Data Memory

指令说明 将指定数据存储器的每一个位置位为 1。

功能表示 [m] ← FFH

影响标志位 无

LSET [m].i Set bit of Data Memory

指令说明 将指定数据存储器的第 i 位置位为 1。

功能表示 [m].i ← 1

影响标志位 无

LSIZ [m] Skip if increment Data Memory is 0

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

功能表示 [m] ←[m] + 1，如果 [m]=0 跳过下一条指令执行

影响标志位 无

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

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

功能表示 ACC ←[m] + 1，如果 ACC=0 跳过下一条指令执行

影响标志位 无

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

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

功能表示 如果 [m].i≠0，跳过下一条指令执行

影响标志位 无

LSNZ [m] Skip if Data Memory is not 0

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

功能表示 如果 [m]≠0，跳过下一条指令执行

影响标志位 无

LSUB A, [m] Subtract Data Memory from ACC

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

功能表示 ACC ← ACC – [m]

影响标志位 OV、Z、AC、C、SC、CZ

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

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

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

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

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

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

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

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

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

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

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

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

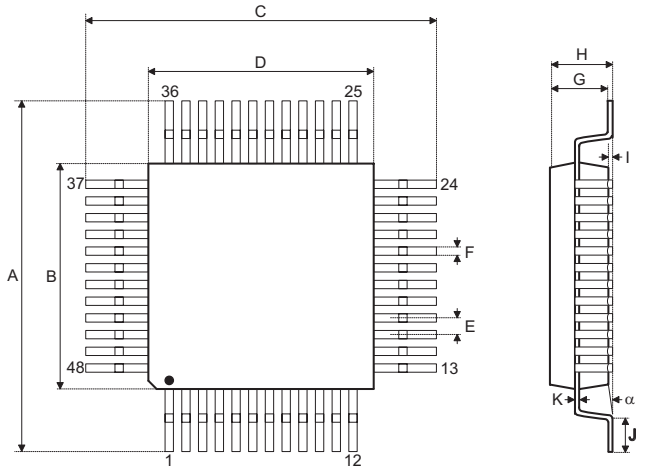
封装信息

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

封装信息的相关内容如下所示，点击可链接至 Holtek 网站相关信息页面。

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

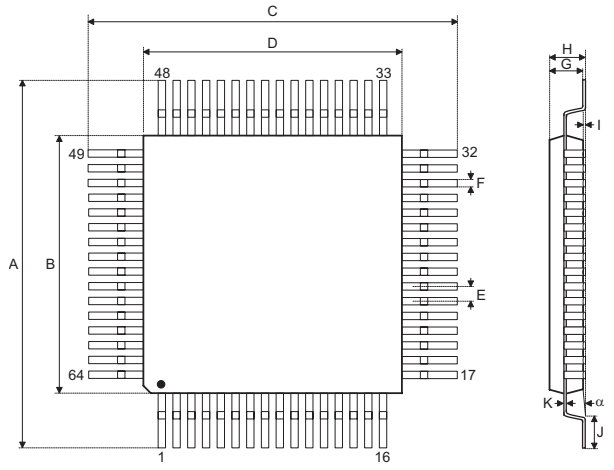
48-pin LQFP (7mm × 7mm) 外形尺寸



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	—	9.00 BSC	—
B	—	7.00 BSC	—
C	—	9.00 BSC	—
D	—	7.00 BSC	—
E	—	0.50 BSC	—
F	0.17	0.22	0.27
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°

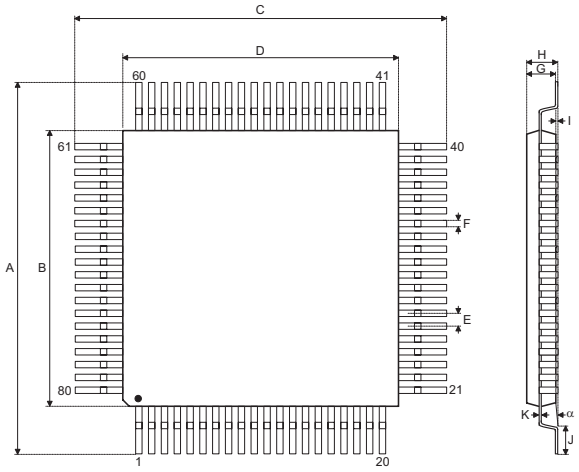
64-pin LQFP (7mm × 7mm) 外形尺寸



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°

80-pin LQFP (10mm × 10mm) 外形尺寸



符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	—	0.472 BSC	—
B	—	0.394 BSC	—
C	—	0.472 BSC	—
D	—	0.394 BSC	—
E	—	0.015 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	—	12.00 BSC	—
B	—	10.00 BSC	—
C	—	12.00 BSC	—
D	—	10.00 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°