血糖仪 LCD Flash 单片机

BH67F2475

版本：V1.50　日期：2024-11-21

www.holtek.com

目录

特性

CPU 特性

- 工作电压：
  - ♦ V_{DD} (单片机)：
    - – f_{SYS}=6MHz：2.2V~5.5V
    - – f_{SYS}=12MHz：2.7V~5.5V
    - – f_{SYS}=16MHz：3.3V~5.5V
  - ♦ V_{DD} (USB 模式)：
    - – f_{SYS}=6MHz/12MHz/16MHz：3.6V~5.5V
- V_{DD}=5V，系统时钟为 16MHz 时，指令周期为 0.25μs
- 暂停和唤醒功能，以降低功耗
- 振荡器类型：
  - ♦ 外部高速晶振 – HXT
  - ♦ 内部高速 12MHz RC – HIRC
  - ♦ 外部低速 32.768kHz 晶振 – LXT
  - ♦ 内部低速 32kHz RC – LIRC
- 多种工作模式：快速、低速、空闲和休眠
- 完全集成内部振荡器，无需外接元件
- 所有指令都可在 1~3 个指令周期内完成
- 查表指令
- 109 条功能强大的指令系统
- 16 层堆栈
- 位操作指令

周边特性

- Flash 程序存储器：32K×16
- 数据存储器：2048×8
- LCD 数据存储器：36×8
- True EEPROM 存储器：2048×8
- 在线应用编程功能 – IAP
- 在线系统编程功能 – ISP
- 内部片上调试功能 – OCDS
- 看门狗定时器功能
- 56 个双向 I/O 口
- 4 个与 I/O 口共用的外部中断引脚
- 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
  - ♦ 1 个 16-bit 标准型 TM – STM
  - ♦ 2 个 10-bit 周期型 TM – PTM0~PTM1
  - ♦ 1 个 10-bit 音频型 TM – ATM

- 两组通用串行接口模块 – USIM，用于 SPI、I²C 或 UART 通信
- 独立串行外设接口 – SPI
- 双时基功能，可提供固定时间的中断信号
- 血糖仪模拟前端电路
  - 6 外部通道 12-bit 分辨率的 A/D 转换器
  - 内部参考电压发生器
  - 2 个 12-bit D/A 转换器
  - 2 个内部运算放大器
- LCD 驱动器功能
  - SEG×COM：36×4，34×6 或 32×8
  - 占空比类型：1/4 Duty，1/6 Duty 或 1/8 Duty
  - 偏压电平：1/3 bias
  - 偏压类型：C 型
  - 波形类型：A 型或 B 型
- USB 接口
  - 兼容 USB 2.0 全速模式
  - 支持 4 个端点，包括端点 0
  - 除了端点 0，其余的端点都支持中断和批量传输
  - 除了端点 0，其余的端点分别配置为固定 16、32 和 64 个字节的 FIFO
  - 端点 0 支持控制传输
  - 端点 0 有 8 字节的 FIFO
  - 支持 3.3V LDO 和内置的 UDP 1.5kΩ 上拉电阻
  - 在任意 USB 模式下，内部 12MHz RC 振荡器的精准度可达到 ±0.25%
- 内建乘除法单元 – MDU
- 内建 16-bit 循环冗余校验功能 – CRC
- 低电压复位功能
- 封装类型：32-pin QFN，64/80-pin LQFP

概述

该单片机是一款具有 8 位高性能精简指令集的 Flash 单片机，内置血糖仪模拟前端处理模块、USB 接口和 LCD 驱动器，专门为带 LCD 显示的 USB 血糖仪产品而设计。

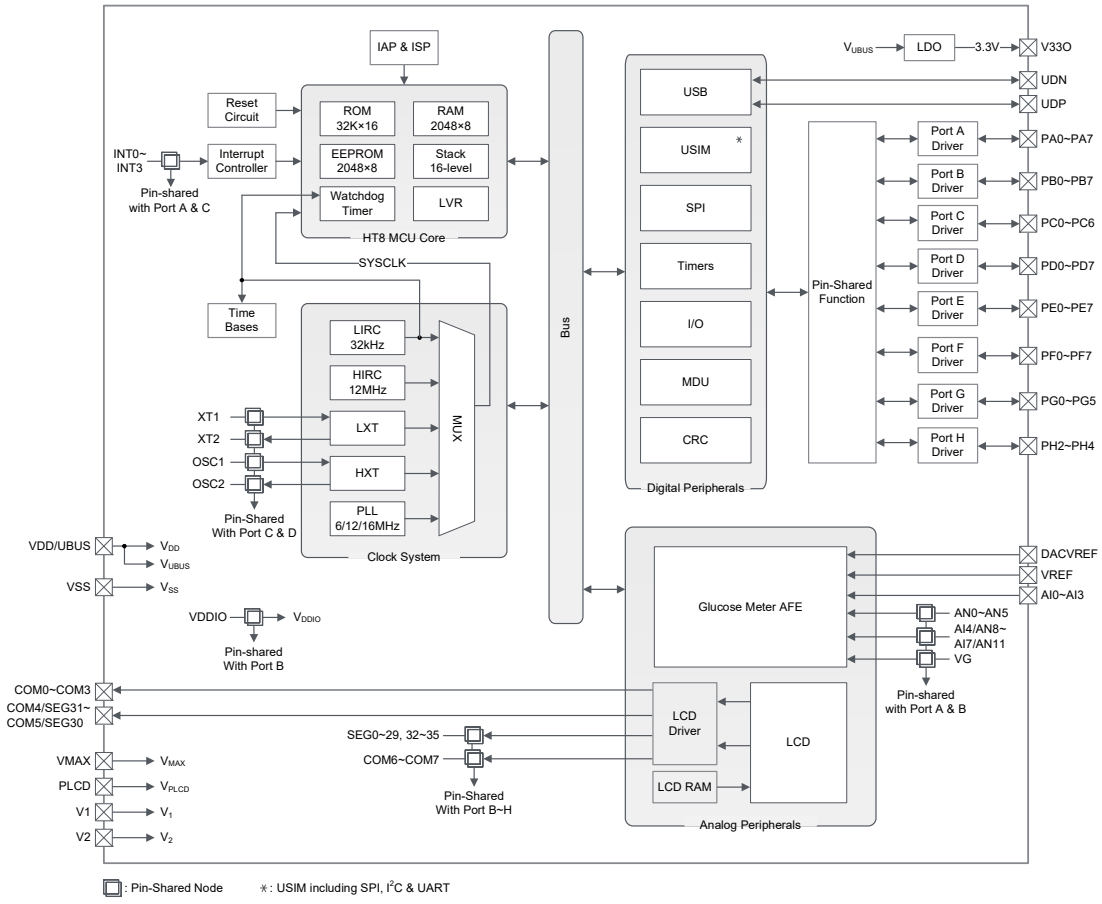
在存储器特性方面，Flash 存储器可多次编程的特性给用户提供了较大的方便。此外还包括 RAM 数据存储器和用于存储序列数据、校准数据等非易失性数据的 True EEPROM 存储器。此外通过使用 IAP 功能，便于用户直接将测量的数据存储至程序存储器中或进行应用程序更新。

在模拟特性方面，该单片机包含一个多通道 12-bit A/D 转换器、两个 12-bit D/A 转换器、两个内部运算放大器和一个完全集成的 LCD 驱动器。同时具有多个使用灵活的定时器模块，可提供定时功能、脉冲产生功能及 PWM 产生功能。内建完整的 SPI、UART、I²C 和 USB 接口功能，为设计者提供了一个易与外部硬件通信的接口。内部看门狗定时器和低电压复位等内部保护特性，外加优秀的抗干扰和 ESD 保护性能，确保单片机在恶劣的电磁干扰环境下可靠地运行。

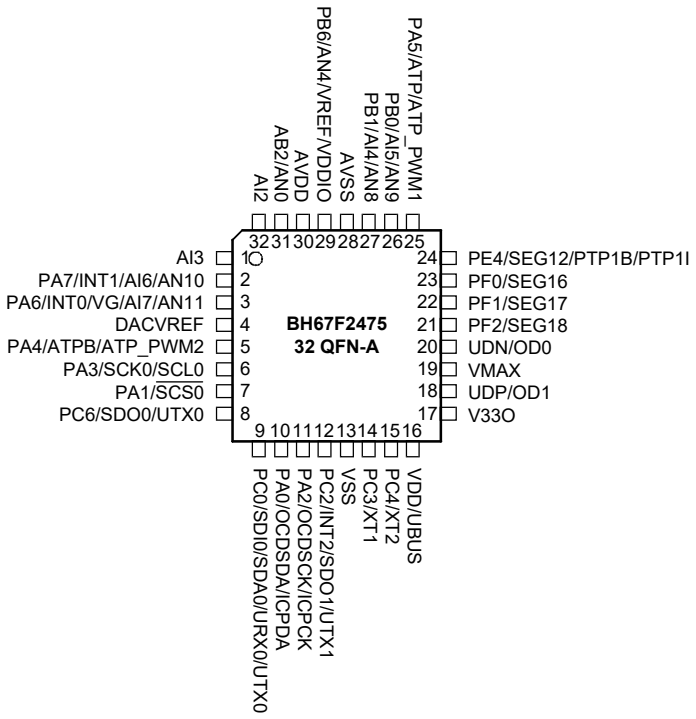
该单片机提供了丰富的内部和外部，高速和低速振荡器功能选项。内部振荡器完全内建，无需外围元器件。其在不同工作模式之间动态切换的能力，为用户提供了一个优化单片机操作和减少功耗的手段。

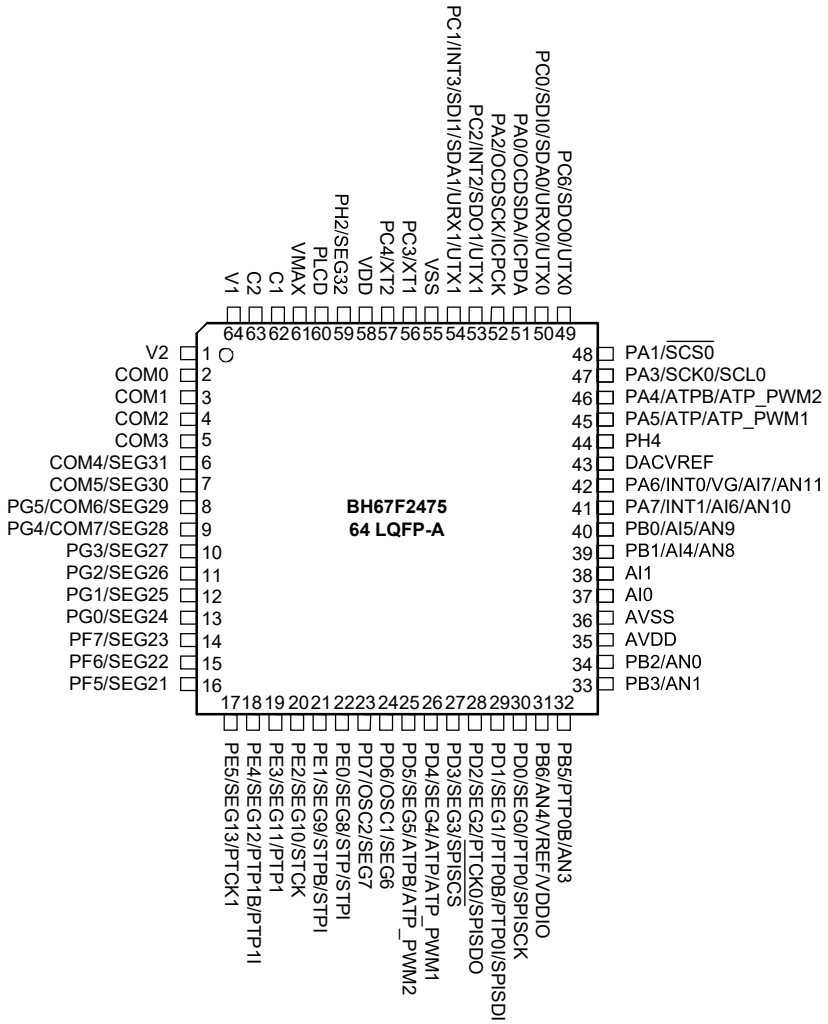
针对USB血糖仪应用，该单片机内置多个相关功能，例如内部参考电压发生器、12-bit D/A转换器、LCD驱动功能、USB接口等。外加I/O使用灵活、时基功能、乘除法单元、循环冗余校验功能等其它特性，使这款单片机可提供多个方案应用于带有 LCD 显示的 USB 血糖仪产品。

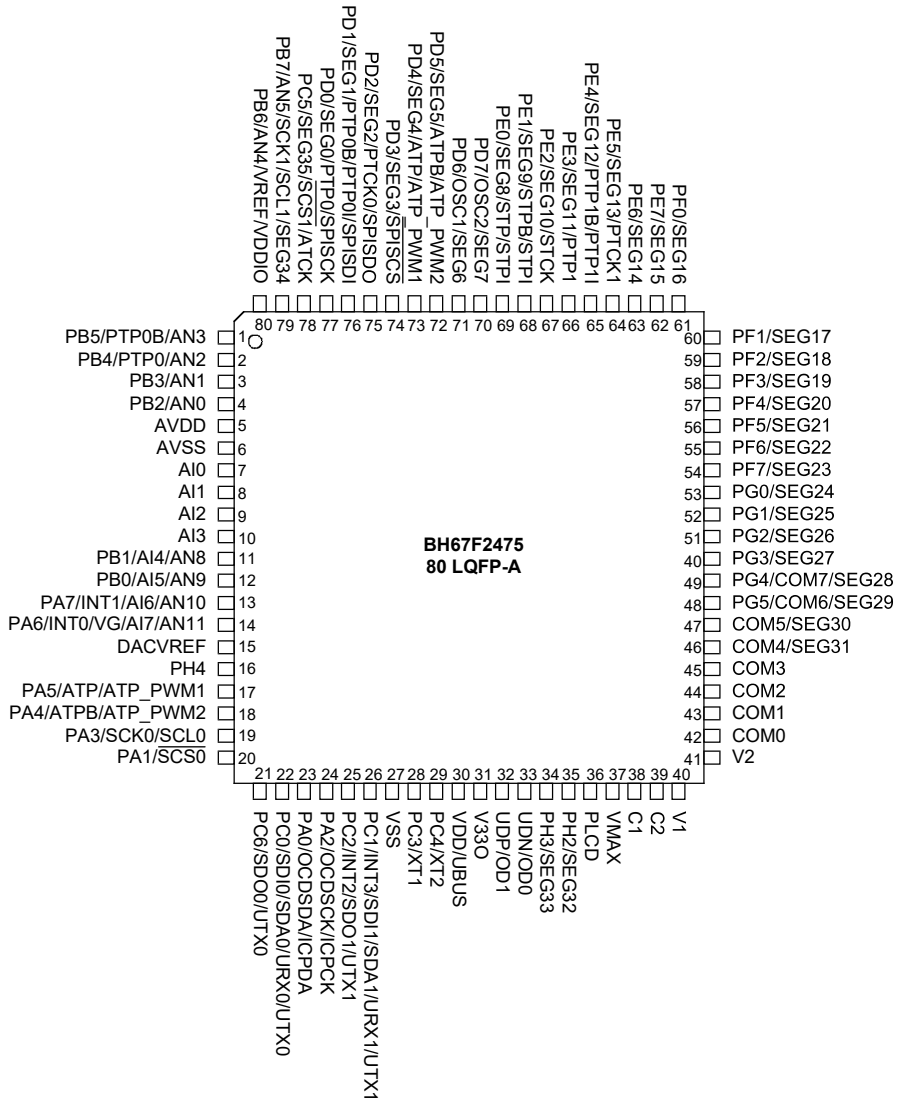
方框图



引脚图


BH67F2475
32 QFN-A
1 AI3
2 PA7/INT1/AI6/AN10
3 PA6/INT0/VG/AI7/AN11
4 DACVREF
5 PA4/ATPB/ATP_PWM2
6 PA3/SCK0/SCL0
7 PA1/SCS0
8 PC6/SDO0/UTX0
9 PC0/SDI0/SDA0/URX0/UTX0
10 PA0/OCDSDA/ICPDA
11 PA2/OCDSCK/ICPCK
12 PC2/INT2/SDO1/UTX1
13 VSS
14 PC3/XT1
15 PC4/XT2
16 VDD/UBUS
17 V33O
18 UDP/OD1
19 VMAX
20 UDN/OD0
21 PF2/SEG18
22 PF1/SEG17
23 PF0/SEG16
24 PE4/SEG12/PTP1B/PTP1I
25 PA5/ATP/ATP_PWM1
26 PB0/AI5/AN9
27 PB1/AI4/AN8
28 AVSS
29 PB6/AN4/VREF/VDDIO
30 AVDD
31 AB2/AN0
32 AI2


BH67F2475
64 LQFP-A
1 V2
2 COM0
3 COM1
4 COM2
5 COM3
6 COM4/SEG31
7 COM5/SEG30
8 PG5/COM6/SEG29
9 PG4/COM7/SEG28
10 PG3/SEG27
11 PG2/SEG26
12 PG1/SEG25
13 PG0/SEG24
14 PF7/SEG23
15 PF6/SEG22
16 PF5/SEG21
17 PE5/SEG13/PTCK1
18 PE4/SEG12/PTP1B/PTP1I
19 PE3/SEG11/PTP1
20 PE2/SEG10/STCK
21 PE1/SEG9/STPB/STPI
22 PE0/SEG8/STP/STPI
23 PD7/OSC2/SEG7
24 PD6/OSC1/SEG6
25 PD5/SEG5/ATPB/ATP_PWM2
26 PD4/SEG4/ATP/ATP_PWM1
27 PD3/SEG3/SPISCS
28 PD2/SEG2/PTCK0/SPISDO
29 PD1/SEG1/PTP0B/PTP0I/SPISDI
30 PD0/SEG0/PTP0/SPISCK
31 PB6/AN4/VREF/VDDIO
32 PB5/PTP0B/AN3
33 PB3/AN1
34 PB2/AN0
35 AVDD
36 AVSS
37 AI0
38 AI1
39 PB1/AI4/AN8
40 PB0/AI5/AN9
41 PA7/INT1/AI6/AN10
42 PA6/INT0/VG/AI7/AN11
43 DACVREF
44 PH4
45 PA5/ATP/ATP_PWM1
46 PA4/ATPB/ATP_PWM2
47 PA3/SCK0/SCL0
48 PA1/SCS0
49 PC6/SDO0/UTX0
50 PC0/SDI0/SDA0/URX0/UTX0
51 PA0/OCDSDA/ICPDA
52 PA2/OCDSCK/ICPCK
53 PC2/INT2/SDO1/UTX1
54 PC1/INT3/SDI1/SDA1/URX1/UTX1
55 VSS
56 PC3/XT1
57 PC4/XT2
58 VDD
59 PH2/SEG32
60 PLCD
61 VMAX
62 C1
63 C2
64 V1



注：1. 若共用脚有多种输出，所需引脚共用功能通过引脚共用寄存器中相应的软件控制位控制。

2. OCDSDA 和 OCDSCK 引脚为片上调试功能专用引脚。

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

引脚说明

每个引脚的功能如下表所述，而引脚配置的详细内容见规格书其它章节。下述引脚功能表格是针对最大封装提供的引脚，对于小封装会有部分引脚未出现。

引脚名称	功能	OPT	I/T	O/T	说明
PA0/OCDSDA/ICPDA	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
	OCDSDA	—	ST	CMOS	OCDS 数据 / 地址引脚
	ICPDA	—	ST	CMOS	ICP 数据 / 地址引脚
PA1/$\overline{SCS0}$	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
	$\overline{SCS0}$	PAS0	ST	CMOS	USIM0 SPI 从机选择
PA2/OCDSCK/ICPCK	PA2	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
	OCDSCK	—	ST	—	OCDS 时钟引脚
	ICPCK	—	ST	—	ICP 时钟引脚
PA3/SCK0/SCL0	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
	SCK0	PAS0	ST	CMOS	USIM0 SPI 串行时钟
	SCL0	PAS0	ST	NMOS	USIM0 I^2C 时钟线
PA4/ATPB/ATP_PWM2	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
	ATPB	PAS1	—	CMOS	ATM 反相输出
	ATP_PWM2	PAS1	—	CMOS	ATM 音频 PWM 模式输出 2
PA5/ATP/ATP_PWM1	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
	ATP	PAS1	—	CMOS	ATM 输出
	ATP_PWM1	PAS1	—	CMOS	ATM 音频 PWM 模式输出 1
PA6/INT0/VG/AI7/AN11	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	ST	—	外部中断 0 输入
	VG	PAS1	AN	—	虚拟地
	AI7/AN11	PAS1	AN	—	模拟输入，AI7 与 AN11 连动
PA7/INT1/AI6/AN10	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	ST	—	外部中断 1 输入
	AI6/AN10	PAS1	AN	—	模拟输入，AI6 与 AN10 连动
PB0/AI5/AN9	PB0	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AI5/AN9	PBS0	AN	—	模拟输入，AI5 与 AN9 连动

引脚名称	功能	OPT	I/T	O/T	说明
PB1/AI4/AN8	PB1	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AI4/AN8	PBS0	AN	—	模拟输入，AI4 与 AN8 连动
PB2/AN0	PB2	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AN0	PBS0	AN	—	A/D 转换器外部输入通道
PB3/AN1	PB3	PBPU PBS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AN1	PBS0	AN	—	A/D 转换器外部输入通道
PB4/PTP0/AN2	PB4	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTP0	PBS1	—	CMOS	PTM0 输出
	AN2	PBS1	AN	—	A/D 转换器外部输入通道
PB5/PTP0B/AN3	PB5	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	PTP0B	PBS1	—	CMOS	PTM0 反相输出
	AN3	PBS1	AN	—	A/D 转换器外部输入通道
PB6/AN4/VREF/VDDIO	PB6	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AN4	PBS1	AN	—	A/D 转换器外部输入通道
	VREF	PBS1	AN	—	A/D 转换器外部参考电压输入
	VDDIO	PBS1	PWR	—	PD0~PD3 引脚电源
PB7/AN5/SCK1/SCL1/ SEG34	PB7	PBPU PBS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	AN5	PBS1	AN	—	A/D 转换器外部输入通道
	SCK1	PBS1	ST	CMOS	USIM1 SPI 串行时钟
	SCL1	PBS1	ST	NMOS	USIM1 I^2C 时钟线
	SEG34	PBS1	—	AN	LCD SEG 输出
PC0/SDI0/SDA0/URX0/ UTX0	PC0	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SDI0	PCS0	ST	—	USIM0 SPI 串行数据输入
	SDA0	PCS0	ST	NMOS	USIM0 I^2C 数据线
	URX0/UTX0	PCS0	ST	CMOS	USIM0 UART 串行数据输入 (全双工通信)；UART 串行数据输入 / 输出 (单线通信模式)
PC1/INT3/SDI1/SDA1/ URX1/UTX1	PC1	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	INT3	PCS0 INTEG INTC2	ST	—	外部中断 3 输入
	SDI1	PCS0	ST	—	USIM1 SPI 串行数据输入
	SDA1	PCS0	ST	NMOS	USIM1 I^2C 数据线
	URX1/UTX1	PCS0	ST	CMOS	USIM1 UART 串行数据输入 (全双工通信)；UART 串行数据输入 / 输出 (单线通信模式)

引脚名称	功能	OPT	I/T	O/T	说明
PC2/INT2/SDO1/UTX1	PC2	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	INT2	PCS0 INTEG INTC1	ST	—	外部中断 2 输入
	SDO1	PCS0	—	CMOS	USIM1 SPI 串行数据输出
	UTX1	PCS0	—	CMOS	USIM1 UART 串行数据输出
PC3/XT1	PC3	PCPU PCS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	XT1	PCS0	AN	—	LXT 振荡器引脚
PC4/XT2	PC4	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	XT2	PCS1	—	AN	LXT 振荡器引脚
PC5/SEG35/$\overline{\text{SCS1}}$/ATCK	PC5	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG35	PCS1	—	AN	LCD SEG 输出
	$\overline{\text{SCS01}}$	PCS1	ST	CMOS	USIM1 SPI 从机选择
	ATCK	PCS1	ST	—	ATM 时钟输入
PC6/SDO0/UTX0	PC6	PCPU PCS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SDO0	PCS1	—	CMOS	USIM0 SPI 串行数据输出
	UTX0	PCS1	—	CMOS	USIM0 UART 串行数据输出
PD0/SEG0/PTP0/SPISCK	PD0	PDPU PDS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG0	PDS0	—	AN	LCD SEG 输出
	PTP0	PDS0	—	CMOS	PTM0 输出
	SPISCK	PDS0	ST	CMOS	SPI 串行时钟
PD1/SEG1/PTP0B/ PTP0I/SPISDI	PD1	PDPU PDS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG1	PDS0	—	AN	LCD SEG 输出
	PTP0B	PDS0	—	CMOS	PTM0 反相输出
	PTP0I	PDS0	ST	—	PTM0 捕捉输入
	SPISDI	PDS0	ST	—	SPI 串行数据输入
PD2/SEG2/PTCK0/ SPISDO	PD2	PDPU PDS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG2	PDS0	—	AN	LCD SEG 输出
	PTCK0	PDS0	ST	—	PTM0 时钟输入
	SPISDO	PDS0	—	CMOS	SPI 串行数据输出
PD3/SEG3/$\overline{\text{SPISCS}}$	PD3	PDPU PDS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG3	PDS0	—	AN	LCD SEG 输出
	$\overline{\text{SPISCS}}$	PDS0	ST	CMOS	SPI 从机选择
PD4/SEG4/ATP/ ATP_PWM1	PD4	PDPU PDS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG4	PDS1	—	AN	LCD SEG 输出
	ATP	PDS1	—	CMOS	ATM 输出
	ATP_PWM1	PDS1	—	CMOS	ATM 音频 PWM 模式输出 1

引脚名称	功能	OPT	I/T	O/T	说明
PD5/SEG5/ATPB/ ATP_PWM2	PD5	PDPU PDS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG5	PDS1	—	AN	LCD SEG 输出
	ATPB	PDS1	—	CMOS	ATM 反相输出
	ATP_PWM2	PDS1	—	CMOS	ATM 音频 PWM 模式输出 2
PD6/OSC1/SEG6	PD6	PDPU PDS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OSC1	PDS1	AN	—	HXT 振荡器引脚
	SEG6	PDS1	—	AN	LCD SEG 输出
PD7/OSC2/SEG7	PD7	PDPU PDS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	OSC2	PDS1	—	AN	HXT 振荡器引脚
	SEG7	PDS1	—	AN	LCD SEG 输出
PE0/SEG8/STP/STPI	PE0	PEPU PES0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG8	PES0	—	AN	LCD SEG 输出
	STP	PES0	—	CMOS	STM 输出
	STPI	PES0 IFS	ST	—	STM 捕捉输入
PE1/SEG9/STPB/STPI	PE1	PEPU PES0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG9	PES0	—	AN	LCD SEG 输出
	STPB	PES0	—	CMOS	STM 反相输出
	STPI	PES0 IFS	ST	—	STM 捕捉输入
PE2/SEG10/STCK	PE2	PEPU PES0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG10	PES0	—	AN	LCD SEG 输出
	STCK	PES0	ST	—	STM 时钟输入
PE3/SEG11/PTP1	PE3	PEPU PES0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG11	PES0	—	AN	LCD SEG 输出
	PTP1	PES0	—	CMOS	PTM1 输出
PE4/SEG12/PTP1B/PTP1I	PE4	PEPU PES1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG12	PES1	—	AN	LCD SEG 输出
	PTP1B	PES1	—	CMOS	PTM1 反相输出
	PTP1I	PES1	ST	—	PTM1 捕捉输入
PE5/SEG13/PTCK1	PE5	PEPU PES1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG13	PES1	—	AN	LCD SEG 输出
	PTCK1	PES1	ST	—	PTM1 时钟输入
PE6/SEG14	PE6	PEPU PES1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG14	PES1	—	AN	LCD SEG 输出

引脚名称	功能	OPT	I/T	O/T	说明
PE7/SEG15	PE7	PEPU PES1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG15	PES1	—	AN	LCD SEG 输出
PF0/SEG16~PF3/SEG19	PF0~PF3	PFPU PFS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG16~SEG19	PFS0	—	AN	LCD SEG 输出
PF4/SEG20~PF7/SEG23	PF4~PF7	PFPU PFS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG20~SEG23	PFS1	—	AN	LCD SEG 输出
PG0/SEG24~PG3/SEG27	PG0~PG3	PGPU PGS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG24~SEG27	PGS0	—	AN	LCD SEG 输出
PG4/COM7/SEG28	PG4	PGPU PGS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	COM7	PGS1	—	AN	LCD COM 输出
	SEG28	PGS1	—	AN	LCD SEG 输出
PG5/COM6/SEG29	PG5	PGPU PGS1	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	COM6	PGS1	—	AN	LCD COM 输出
	SEG29	PGS1	—	AN	LCD SEG 输出
PH2/SEG32	PH2	PHPU PHS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG32	PHS0	—	AN	LCD SEG 输出
PH3/SEG33	PH3	PHPU PHS0	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
	SEG33	PHS0	—	AN	LCD SEG 输出
PH4	PH4	PHPU	ST	CMOS	通用 I/O 口，可通过寄存器设置上拉电阻
COM0~COM3	COM0~COM3	—	—	AN	LCD COM 输出
COM4/SEG31	COM4	COMS	—	AN	LCD COM 输出
	SEG31	COMS	—	AN	LCD SEG 输出
COM5/SEG30	COM5	COMS	—	AN	LCD COM 输出
	SEG30	COMS	—	AN	LCD SEG 输出
VMAX	VMAX	—	PWR	—	LCD 最大电压，需连接到 VDD 或 V1
PLCD	PLCD	—	PWR	AN	LCD 电源
V1	V1	—	PWR	AN	LCD 电压泵
V2	V2	—	PWR	AN	LCD 电压泵
C1	C1	—	AN	AN	LCD 电压泵
C2	C2	—	AN	AN	LCD 电压泵
AI0~AI3	AI0~AI3	—	AN	—	模拟输入
DACVREF	DACVREF	—	AN	—	D/A 转换器参考电压输入
UDP/OD1	UDP	—	ST	CMOS	USB D+ 线
	OD1	—	ST	NMOS	NMOS 开漏 I/O 引脚
UDN/OD0	UDN	—	ST	CMOS	USB D- 线
	OD0	—	ST	NMOS	NMOS 开漏 I/O 引脚

引脚名称	功能	OPT	I/T	O/T	说明
V33O	V33O	—	—	PWR	USB 3.3V 稳压器输出
VDD/UBUS	VDD	—	PWR	—	数字正电源
	UBUS	—	PWR	—	USB 数字正电源
VSS	VSS	—	PWR	—	数字负电源
AVDD	AVDD	—	PWR	—	模拟正电源
AVSS	AVSS	—	PWR	—	模拟负电源

注：I/T：输入类型；　　O/T：输出类型；
PWR：电源；　　OPT：通过配置寄存器选项来配置；
CMOS：CMOS 输出；　　NMOS：NMOS 输出；
ST：施密特触发输入；　　AN：模拟信号

极限参数

电源供应电压 ..... V_{SS}-0.3V~6.0V
端口输入电压 ..... V_{SS}-0.3V~V_{DD}+0.3V
储存温度 ..... -60°C~150°C
工作温度 ..... -40°C~85°C
I_{OH} 总电流 ..... -80mA
I_{OL} 总电流 80mA
总功耗 500mW

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

直流电气特性

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

工作电压特性

Ta=-40°C~85°C

符号	参数	测试条件	最小	典型	最大	单位
V_{DD}	工作电压 – HXT	f_{SYS}=6MHz	2.2	—	5.5	V
		f_{SYS}=12MHz	2.7	—	5.5	
		f_{SYS}=16MHz	3.3	—	5.5	
	工作电压 – HIRC	f_{SYS}=12MHz	2.7	—	5.5	V
	工作电压 – LXT	f_{SYS}=32.768kHz	2.2	—	5.5	V
	工作电压 – LIRC	f_{SYS}=32kHz	2.2	—	5.5	V
	工作电压 – PLL	f_{SYS}=6MHz，PLLEN=1	2.2	—	5.5	V
		f_{SYS}=12MHz，PLLEN=1	2.7	—	5.5	
		f_{SYS}=16MHz，PLLEN=1	3.3	—	5.5	

工作电流特性

Ta=-40°C~85°C

符号	工作模式	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{DD}	低速模式 – LIRC	3V	f_{SYS}=32kHz	—	16	30	μA
		5V		—	28	50	
	低速模式 – LXT	3V	f_{SYS}=32.768kHz	—	17	30	μA
		5V		—	30	50	
	快速模式 – HIRC	3V	f_{SYS}=12MHz	—	1.5	2.8	mA
		5V		—	3.0	4.5	
	快速模式 – HXT	3V	f_{SYS}=6MHz	—	0.7	1.5	mA
		5V		—	1.6	3.0	
		3V	f_{SYS}=12MHz	—	1.3	3.0	
		5V		—	2.7	6.0	
		3.3V	f_{SYS}=16MHz	—	3	6	
		5V		—	5.0	9.5	
	快速模式 – PLL	3V	f_{SYS}=6MHz，PLLEN=1	—	2.0	3.0	mA
		5V		—	3	4	
		3V	f_{SYS}=12MHz，PLLEN=1	—	2.1	3.5	
		5V		—	3.8	7.0	
		3.3V	f_{SYS}=16MHz，PLLEN=1	—	3.2	6.5	
		5V		—	4.5	9.0	

注：当使用该表格电气特性数据时，以下几点需注意：

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

待机电流特性

Ta=25°C，除非另有说明

符号	待机模式	测试条件		最小	典型	最大	最大 @85°C	单位
		V_{DD}	条件					
I_{STB}	休眠模式	3V	WDT off	—	0.45	0.90	8.00	μA
		5V		—	0.5	2.0	10.0	
		3V	WDT on，f_{SUB}=f_{LIRC}	—	1.5	3.0	3.0	
		5V		—	2	5	5	
		3V	WDT on，f_{SUB}=f_{LXT}	—	0.7	1.5	3.0	
		5V		—	1.5	3.0	5.0	
	空闲模式 0 – LIRC	3V	f_{SUB} on	—	1.5	3.0	3.0	μA
		5V		—	2.8	5.0	5.0	
	空闲模式 0 – LXT	3V	f_{SUB} on	—	1.8	3.0	3.0	μA
		5V		—	2.8	5.0	5.0	
	空闲模式 1 – HIRC	3V	f_{SUB} on，f_{SYS}=12MHz	—	0.65	1.40	1.40	mA
		5V		—	1.3	3.0	3.0	

符号	待机模式	测试条件		最小	典型	最大	最大 @85°C	单位
		V_{DD}	条件					
I_{STB}	空闲模式 1 – HXT	3V	f_{SUB} on，f_{SYS}=12MHz	—	0.9	1.5	1.5	mA
		5V		—	1.4	3.0	3.0	
	空闲模式 1 – PLL	3V	f_{SUB} on，f_{SYS}=6MHz，PLLEN=1	—	1.0	2.5	2.5	mA
		5V		—	2.0	3.5	3.5	
		3V	f_{SUB} on，f_{SYS}=12MHz，PLLEN=1	—	1.5	3.0	3.0	
		5V		—	2.5	4.0	4.0	
		3.3V	f_{SUB} on，f_{SYS}=16MHz，PLLEN=1	—	1.7	3.5	3.5	
		5V		—	3	5	5	

注：当使用该表格电气特性数据时，以下几点需注意：
1. 任何数字输入都设置为非浮空的状态。
2. 所有测量都在无负载且所有外围功能关闭的条件下进行。
3. 无直流电流路径。
4. 所有待机电流数值都是在 HALT 指令执行后即停止执行所有指令后测得。

交流电气特性

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

内部高速振荡器 – HIRC – 频率精准度

程序烧录时，烧录器会依据用户选择的 HIRC 频率和工作电压 (5V) 对 HIRC 进行频率精准度调整。

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{HIRC}	通过烧录器调整后的 12MHz HIRC 频率	4.4V~5.25V	USB 模式，CLKADJ=1，UDP/UDN 插入主机，Ta=-40°C~85°C	-0.25%	12	+0.25%	MHz
		5V	Ta=25°C	-2%	12	+2%	
		5V	Ta=0°C~70°C	-3%	12	+3%	
		5V	Ta=-40°C~85°C	3.5%	12	+3.5%	
		2.7V~5.5V	Ta=0°C~70°C	-4%	12	+4%	
		2.7V~5.5V	Ta=-40°C~85°C	-4.5%	12	+4.5%	

注：1. 烧录器可在 5V 固定电压下对 HIRC 频率进行调整，在此提供 V_{DD}=5V 时的参数值。
2. 5V 表格列下面提供的是全压条件下的参数值。

内部低速振荡器电气特性 – LIRC

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	温度				
f_{LIRC}	LIRC 频率	3V	25°C	-2%	32	+2%	kHz
		2.2V~5.5V	-40°C~85°C	-7%	32	+7%	
t_{START}	LIRC 启动时间	—	-40°C~85°C	—	—	100	μs

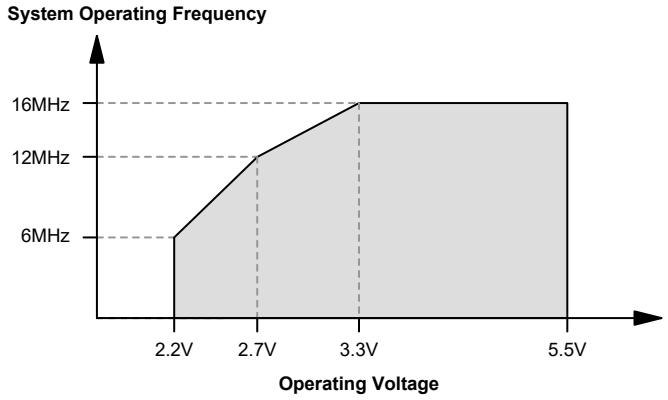
外部低速晶体振荡器 LXT 电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{LXT}	LXT 频率	2.2V~5.5V	—	—	32.768	—	kHz
t_{START}	LXT 启动时间	3V	—	—	—	1000	ms
		5V	—	—	—	1000	
Duty Cycle	占空比	—	—	40	—	60	%
R_{NEG}	负阻	2.2V	—	3×ESR	—	—	Ω

注：C1，C2 和 R_P 为外部元件。C1=C2=10pF，R_P=10MΩ。C_L=7pF，ESR=30kΩ。

工作频率电气特性曲线图



系统上电时间电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
t_{SST}	系统启动时间 (从 f_{SYS} off 的状态下唤醒)	—	$f_{SYS}=f_{LXT}$	—	1024	—	t_{SYS}
		—	$f_{SYS}=f_H\sim f_H/64$，$f_H=f_{HXT}$	—	128	—	t_{SYS}
		—	$f_{SYS}=f_H\sim f_H/64$，$f_H=f_{HIRC}$	—	16	—	t_{SYS}
		—	f_{PLL} off → on (PLLF=1)	—	2560	—	t_{PLL} (48MHz)
		—	$f_{SYS}=f_{LIRC}$	—	2	—	t_{SYS}
	系统启动时间 (从 f_{SYS} on 的状态下唤醒)	—	$f_{SYS}=f_H\sim f_H/64$， $f_H=f_{HXT}$ 或 f_{HIRC} 或 f_{PLL}	—	2	—	t_{SYS}
		—	$f_{SYS}=f_{LXT}$ 或 f_{LIRC}	—	2	—	t_{SYS}
	系统速度切换时间 (快速模式 → 低速模式或 低速模式 → 快速模式)	—	f_{HXT} off → on (HXTF=1)	—	1024	—	t_{HXT}
		—	f_{HIRC} off → on (HIRCF=1)	—	16	—	t_{HIRC}
		—	f_{PLL} off → on (PLLF=1)	—	2560	—	t_{PLL} (48MHz)
		—	f_{LXT} off → on (LXTF=1)	—	1024	—	t_{LXT}

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 (上电复位或 LVR 硬件复位)	—	RR_{POR}=5V/ms	14	16	18	ms
	系统复位延迟时间 (LVRC/WDTC/RSTC 软件复位)	—	—				
	系统复位延迟时间 (WDT 溢出复位)	—	—	14	16	18	
t_{SRESET}	软件复位最小脉宽	—	—	45	90	120	μs

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

输入 / 输出口电气特性

输入 / 输出口 (非多电源引脚) 直流电气特性

适用于 PA0~PA7、PB0~PB7、PC0~PC6、PD4~PD7、PE0~PE7、PF0~PF7、PG0~PG5 和 PH2~PH4 引脚。

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{IL}	I/O 口低电平输入电压	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	I/O 口高电平输入电压	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	I/O 口灌电流	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	I/O 口源电流	3V	$V_{OH}=0.9V_{DD}$	-4	-8	—	mA
		5V		-8	-16	—	
R_{PH}	I/O 口上拉电阻 (1)	3V	—	20	60	100	kΩ
		5V		10	30	50	
I_{LEAK}	I/O 口输入漏电流	5V	$V_{IN}=V_{DD}$ 或 $V_{IN}=V_{SS}$	—	—	±1	μA
t_{INT}	外部中断引脚最小输入脉宽	—	—	10	—	—	μs
t_{TCK}	PTMn/STM/ATM 时钟输入引脚最小输入脉宽	—	—	0.3	—	—	μs
t_{TPI}	PTMn/STM 捕捉输入引脚最小输入脉宽	—	—	0.3	—	—	μs
f_{TMCLK}	PTMn/STM 最大时钟源频率	5V	—	—	—	1	f_{SYS}
t_{CPW}	PTMn/STM 最小捕捉脉宽	—	—	t_{CPW} (2)	—	—	μs

注：1. R_{PH} 内部上拉电阻值的计算方法是：将引脚接地并设置为输入且使能上拉电阻功能，然后在特定电源电压下测量该引脚上的电流，最后电压除以测量的电流值从而得到此上拉电阻值。
2. 对于 PTMn：
若 PTnCAPTS=0，$t_{CPW}=max(2\times t_{TMCLK}, t_{TPI})$

若 PTnCAPTS=1，t_{CPW}=max(2×t_{TMCLK}, t_{TCK})
例 1：若 PTnCAPTS=0，f_{TMCLK}=16MHz，t_{TPI}=0.3μs，则 t_{CPW}=max(0.125μs, 0.3μs)=0.3μs
例 2：若 PTnCAPTS=1，f_{TMCLK}=16MHz，t_{TCK}=0.3μs，则 t_{CPW}=max(0.125μs, 0.3μs)=0.3μs
例 3：若 PTnCAPTS=0，f_{TMCLK}=8MHz，t_{TPI}=0.3μs，则 t_{CPW}=max(0.25μs, 0.3μs)=0.3μs
例 4：若 PTnCAPTS=0，f_{TMCLK}=4MHz，t_{TPI}=0.3μs，则 t_{CPW}=max(0.5μs, 0.3μs)=0.5μs

对于 STM：
t_{CPW}=max(2×t_{TMCLK}, t_{TPI})
例 1：若 f_{TMCLK}=16MHz，t_{TPI}=0.3μs，则 t_{CPW}=max(0.125μs, 0.3μs)=0.3μs
例 2：若 f_{TMCLK}=8MHz，t_{TPI}=0.3μs，则 t_{CPW}=max(0.25μs, 0.3μs)=0.3μs
例 3：若 f_{TMCLK}=4MHz，t_{TPI}=0.3μs，则 t_{CPW}=max(0.5μs, 0.3μs)=0.5μs
其中 t_{TMCLK}=1/f_{TMCLK}

输入 / 输出口（多电源引脚）直流电气特性

适用于 PD0~PD3 引脚。

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	电源 – V_{DD0}	—	—	2.2	5.0	5.5	V
V_{DDIO}	电源 – V_{DD1}	—	—	2.2	—	V_{DD}	V
V_{IL}	I/O 口低电平输入电压	5V	引脚电源 =V_{DDn}，V_{DDn}=V_{DD}，n=0~1	0	—	1.5	V
		—	引脚电源 =V_{DDn}，n=0~1	0	—	0.2V_{DDn}	
V_{IH}	I/O 口高电平输入电压	5V	引脚电源 =V_{DDn}，V_{DDn}=V_{DD}，n=0~1	3.5	—	5.0	V
		—	引脚电源 =V_{DDn}，n=0~1	0.8V_{DDn}	—	V_{DDn}	
I_{OL}	I/O 口灌电流	3V	V_{OL}=0.1V_{DDn}，V_{DDn}=V_{DD}，n=0~1	16	32	—	mA
		5V	V_{OL}=0.1V_{DDn}，V_{DDn}=V_{DD}，n=0~1	32	65	—	
			V_{OL}=0.1V_{DDn}，V_{DDn}=3V，n=0~1	20	40	—	
I_{OH}	I/O 口源电流	3V	V_{OH}=0.9V_{DDn}，V_{DDn}=V_{DD}，n=0~1	-4	-8	—	mA
		5V	V_{OH}=0.9V_{DDn}，V_{DDn}=V_{DD}，n=0~1	-8	-16	—	
			V_{OH}=0.9V_{DDn}，V_{DDn}=3V，n=0~1	-2.5	-5.0	—	
R_{PH}	I/O 口上拉电阻 (1)	3V	V_{DDn}=V_{DD}，n=0~1	20	60	100	kΩ
		5V	V_{DDn}=V_{DD}，n=0~1	10	30	50	
			V_{DDn}=3V，n=0~1	36	110	180	
I_{LEAK}	I/O 口输入漏电流	5V	V_{IN}=V_{SS} 或 V_{IN}=V_{DDn}，n=0~1	—	—	±1	μA

注：1. R_{PH} 内部上拉电阻值的计算方法是：将引脚接地并设置为输入且使能上拉电阻功能，然后在特定电源电压下测量该引脚上的电流，最后电压除以测量的电流值从而得到此上拉电阻值。
2. 条件栏中 V_{DDn} 实际适用对象 (V_{DD} 或 V_{DDIO}) 应根据 V_{DD} 栏的值和各个 V_{DDn} 电压范围决定。
3. 当 V_{DDn} 作为 I/O 电源时，需在靠近 VDDn 引脚加 0.1μF 旁路电容。

存储器电气特性

Ta=-40°C~85°C，除非另有说明

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{RW}	读 / 写工作电压	—	—	V_{DDmin}	—	V_{DDmax}	V
Flash 程序存储器							
t_{FWR}	写时间	—	FWERTS=0	—	2.2	2.7	ms
		—	FWERTS=1	—	3.0	3.6	
t_{FER}	擦除时间	—	FWERTS=0	—	3.2	3.9	ms
		—	FWERTS=1	—	3.7	4.5	
E_P	存储单元耐受性	—	—	100K	—	—	E/W
t_{RETD}	数据保存时间	—	Ta=25°C	—	40	—	Year
t_{ACTV}	ROM 激活时间 – 从暂停模式唤醒	—	—	32	—	64	μs
数据 EEPROM 存储器							
t_{EERD}	读时间	—	—	—	—	4	t_{SYS}
t_{EEWR}	写时间 (字节模式)	—	EWERTS=0	—	5.4	6.6	ms
		—	EWERTS=1	—	6.7	8.1	
	写时间 (页模式)	—	EWERTS=0	—	2.2	2.7	
		—	EWERTS=1	—	3.0	3.6	
t_{EEER}	擦除时间	—	EWERTS=0	—	3.2	3.9	ms
		—	EWERTS=1	—	3.7	4.5	
E_P	存储单元耐受性	—	—	100K	—	—	E/W
t_{RETD}	数据保存时间	—	Ta=25°C	—	40	—	Year
RAM 数据存储器							
V_{DR}	RAM 数据保存电压	—	—	1.0	—	—	V

注：1. 在计算从暂停模式唤醒的系统总启动时间时，还需加上 ROM 激活时间 t_{ACTV}。
2. “E/W”表示擦 / 写次数。

LVR 电气特性

Ta=-40°C~85°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	工作电压	—	—	2.2	—	5.5	V
V_{LVR}	低电压复位电压	—	LVR 使能，电压选择 2.1V	-5%	2.1	+5%	V
		—	LVR 使能，电压选择 2.55V		2.55		
		—	LVR 使能，电压选择 3.15V		3.15		
		—	LVR 使能，电压选择 3.8V		3.8		
t_{LVR}	产生 LVR 复位的低电压最短保持时间	—	TLVR[1:0]=00B	120	240	480	μs
			TLVR[1:0]=01B	0.5	1.0	2.0	ms
			TLVR[1:0]=10B	1	2	4	
			TLVR[1:0]=11B	2	4	8	

模拟前端电路电气特性

运算放大器电气特性

Ta=25℃，除非另有说明

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{OPA}	OPA 工作电流	3V	$V_P=V_N=(1/2)V_{DD}$	—	—	650	μA
A_{OL}	OPA 开环增益	3V	—	80	100	—	dB
Ro	输出电阻值	2.4V~3.6V	Ta=0℃~50℃， R_{LOAD}=50kΩ， 0.2V<V_{OP}<V_{DD} - 1.4V (电压跟随器配置)	—	—	260	Ω
I_{OS}	输入失调电流	2.4V~3.6V	Ta=0℃~50℃	-5	—	5	nA
TC	失调电压温度系数	3V	Ta=0℃~50℃	—	—	±20	μV /℃
GBW	增益带宽	3V	R_L=1MΩ，C_L=60pF， $V_{IN}=V_{CM}/2$	100	—	—	kHz
V_{OS}	输入失调电压	3V	未校准 (OPAnOF[4:0]=10000B)	-15	—	15	mV
V_{CM}	OPA 共模电压范围	3V	—	0.1	—	V_{DD} -1.4	V
V_{OR}	OPA 最大输出电压范围	3V	—	V_{SS} +0.1	—	V_{DD} -0.1	V

内部参考电压特性

Ta=25℃，除非另有说明

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{IREF}	内部参考电压	3V	IREFEN=1， PVREF=10000000B	-3%	2.0	+3%	V
I_{IREF}	使用内部参考电压的额外电流	—	IREFEN=1	—	650	1000	μA
TC_{IREF}	内部参考电压温度系数	3V	Ta=10℃~40℃	—	±20	±40	ppm/℃

模拟开关电气特性

Ta=25℃，除非另有说明

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
R_{OP}	导通电阻 (除 VG 外)	3V	Ta=10℃~40℃	—	—	1300	Ω
R_{VG}	导通电阻 (VG)	3V	Ta=10℃~40℃，VGSW=1	—	—	20	Ω
I_{LEAK}	漏电流 (VG)	3V	Ta=10℃~40℃	-5.0	±0.5	5.0	nA

12-bit D/A 转换器电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
DNL	非线性微分误差	3V	DACVRS[1:0]=00B	—	—	±8	LSB
INL	非线性积分误差	3V	DACVRS[1:0]=00B	—	—	±20	LSB
V_{DACO}	输出电压范围	—	—	0	—	1.00	$V_{DACVREF}$
V_{DACO_RIPPLE}	输出电压纹波	3V	DACVRS[1:0]=10B	-2	—	+2	mV

A/D 转换器电气特性

Ta=25°C，V_{DD}=AV_{DD}

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{DD}	A/D 转换器工作电压	—	—	2.7	—	5.5	V
V_{ADI}	A/D 转换器输入电压	—	—	0	—	V_{REF}	V
V_{REF}	A/D 转换器参考电压	—	—	2	—	V_{DD}	V
DNL	A/D 非线性微分误差	3V	V_{REF}=V_{DD}，t_{ADCK}=0.5μs	-3	—	+3	LSB
		5V					
		3V	V_{REF}=V_{DD}，t_{ADCK}=10μs				
		5V					
INL	A/D 非线性积分误差	3V	V_{REF}=V_{DD}，t_{ADCK}=0.5μs	-4	—	+4	LSB
		5V					
		3V	V_{REF}=V_{DD}，t_{ADCK}=10μs				
		5V					
I_{ADC}	A/D 转换器使能的额外电流	3V	无负载，t_{ADCK}=0.5μs	—	1	2	mA
		5V		—	1.5	3.0	mA
t_{ADCK}	A/D 转换器时钟周期	—	—	0.5	—	10.0	μs
t_{ON2ST}	A/D 转换器 On-to-Start 时间	—	—	4	—	—	μs
t_{ADS}	A/D 采样时间	—	—	—	4	—	t_{ADCK}
t_{ADC}	A/D 转换时间 (包括采样和保持时间)	—	—	—	16	—	t_{ADCK}

LCD 驱动器电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
V_{IN}	LCD 工作电压	—	PLCD 引脚提供 LCD 电源 (C 型)	2.0	—	3.7	V
		—	V1 引脚提供 LCD 电源 (C 型)	3.0	—	5.5	
		—	V2 引脚提供 LCD 电源 (C 型)	1.0	—	1.8	
		—	V_A 提供 LCD 电源 (C 型)	3.0	—	5.5	
		3.3V~5.5V	V_B 提供 LCD 电源 (C 型)	-10%	3.0	+10%	
		2.2V~5.5V	V_C 提供 LCD 电源，无负载 (C 型)	-10%	1.0	+10%	
I_{LCD}	LCD 使能的额外电流 – C 型	3V	无负载，$V_A=V_1=V_{DD}$，1/3 Bias，LCDPCK[2:0]=000b	—	0.6	1.2	μA
		5V		—	1	2	
		3V	无负载，$V_A=V_1=V_{DD}$，1/3 Bias，LCDPCK[2:0]=001b~111b	—	1.5	3.0	
		5V		—	2.4	4.8	
I_{LCDOL}	LCD COM 和 SEG 灌电流	3V	$V_{OL}=0.1V_{DD}$	210	420	—	μA
		5V		350	700	—	
I_{LCDOH}	LCD COM 和 SEG 源电流	3V	$V_{OH}=0.9V_{DD}$	-80	-160	—	μA
		5V		-180	-360	—	

USB 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_{DD}	工作电流 (USB)	3V	$f_{SYS}=f_{PLL}=6MHz$，无负载，USB 和 PLL on，其余外设关闭	—	4.5	9.0	mA
		5V		—	9.5	15.0	mA
		3V	$f_{SYS}=f_{PLL}=12MHz$，无负载，USB 和 PLL on，其余外设关闭	—	5	10	mA
		5V		—	10.5	16.0	mA
		5V	$f_{SYS}=f_{PLL}=16MHz$，无负载，USB 和 PLL on，其余外设关闭	—	11	18	mA

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
I_{SUS}	暂停电流 (USB) (IDLE0 模式)	5V	f_H off，$f_{SUB}=f_{LIRC}$ 或 f_{LXT}，无负载，CPU off，USB 收发器和 3.3V 稳压器 on，其余外设关闭，SUSP2=0，RCTRL=0，RUPH=1	—	360	450	μA
		5V	f_H off，$f_{SUB}=f_{LIRC}$ 或 f_{LXT}，无负载，CPU off，USB 收发器和 3.3V 稳压器 off，其余外设关闭，SUSP2=1，RCTRL=1，RUPH=1	—	240	330	μA
V_{V33O}	3.3V 稳压器输出电压	5V	I_{V33O}=70mA	3.0	3.3	3.6	V
R_{UDP}	UDP 到 V33O 的上拉电阻	5V	—	-5%	1.5	+5%	kΩ
	UDP 到 UBUS 的上拉电阻	5V	RCTRL=1	4.9	7.8	12.0	kΩ
R_{UDPN}	UDP/UDN 到 UBUS 的上拉电阻	5V	PU=1	300	650	1000	kΩ
R_{PL}	UBUS 的下拉电阻	5V	SUSP2=1，RUBUS=0	0.5	1.0	2.0	MΩ
R_{OD}	OD0/OD1 到 VDD 的上拉电阻	5V	UMS[2:0]=001B	2.0	4.7	8.0	kΩ
I_{OL_OD}	OD0/OD1 灌电流	5V	UMS[2:0]=001B，V_{OL}=0.1V_{DD}	4	6	—	mA
V_{IH}	OD0/OD1 高电平输入电压	5V	UMS[2:0]=001B，OD1O/OD0O=11B	2	—	5	V
V_{IL}	OD0/OD1 低电平输入电压	5V	UMS[2:0]=001B，OD1O/OD0O=11B	0	—	0.8	V

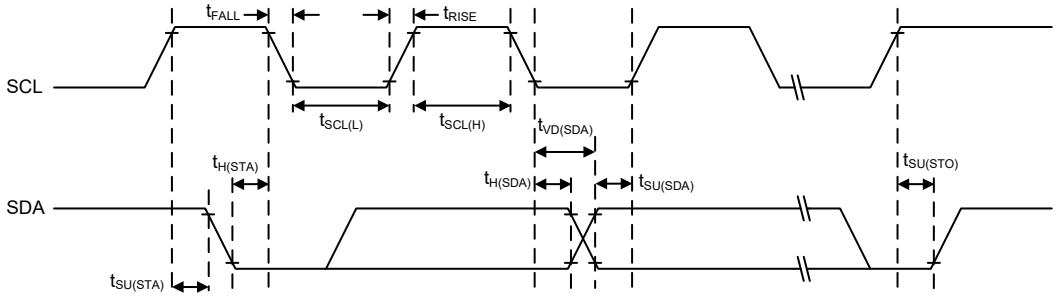
I²C 电气特性

Ta=25°C

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
f_{I2C}	I²C 标准模式 (100kHz) 时的 f_{SYS} 频率 (注)	—	无去抖时间	2	—	—	MHz
			2 个系统时钟时间去抖	4	—	—	
			4 个系统时钟时间去抖	4	—	—	
	I²C 快速模式 (400kHz) 时的 f_{SYS} 频率 (注)	—	无去抖时间	4	—	—	MHz
			2 个系统时钟时间去抖	8	—	—	
			4 个系统时钟时间去抖	8	—	—	
f_{SCL}	SCL 时钟频率	3V/5V	标准模式	—	—	100	kHz
			快速模式	—	—	400	
$t_{SCL(H)}$	SCL 时钟高电平时间	3V/5V	标准模式	3.5	—	—	μs
			快速模式	0.9	—	—	
$t_{SCL(L)}$	SCL 时钟低电平时间	3V/5V	标准模式	3.5	—	—	μs
			快速模式	0.9	—	—	
t_{FALL}	SCL 和 SDA 下降沿时间	3V/5V	标准模式	—	—	1.3	μs
			快速模式	—	—	0.34	
t_{RISE}	SCL 和 SDA 上升沿时间	3V/5V	标准模式	—	—	1.3	μs
			快速模式	—	—	0.34	

符号	参数	测试条件		最小	典型	最大	单位
		V_{DD}	条件				
$t_{SU(SDA)}$	SDA 数据建立时间	3V/5V	标准模式	0.25	—	—	μs
			快速模式	0.1	—	—	
$t_{H(SDA)}$	SDA 数据保持时间	3V/5V	—	0.1	—	—	μs
$t_{VD(SDA)}$	SDA 数据有效时间	3V/5V	—	—	—	0.6	μs
$t_{SU(STA)}$	START 条件建立时间	3V/5V	标准模式	3.5	—	—	μs
			快速模式	0.6	—	—	
$t_{H(STA)}$	START 条件保持时间	3V/5V	标准模式	4.0	—	—	μs
			快速模式	0.6	—	—	
$t_{SU(STO)}$	STOP 条件建立时间	3V/5V	标准模式	3.5	—	—	μs
			快速模式	0.6	—	—	

注：使用去抖功能可使传输更稳定，降低受干扰影响通信失败的机率。

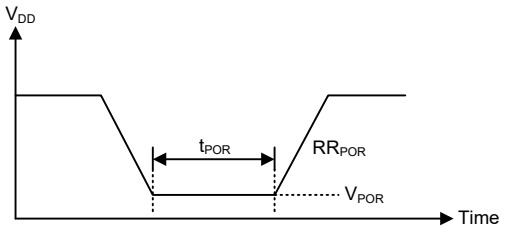


I^2C 时序图

上电复位特性

Ta=25°C

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

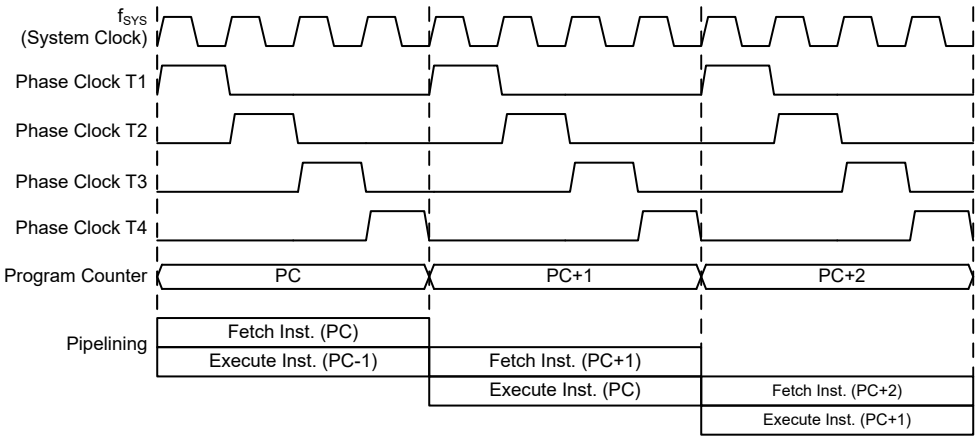


系统结构

内部系统结构是 Holtek 单片机具有良好性能的主要因素。由于采用 RISC 结构，该单片机具有高运算速度和高性能的特点。通过流水线的方式，指令的获取和执行同时进行，此举使得除了跳转和调用指令需多一个指令周期外，其它大部分标准指令或扩展指令都能分别在一个或两个指令周期内完成。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。使得该单片机适用于经济型和大量生产的控制应用。

时序和流水线结构

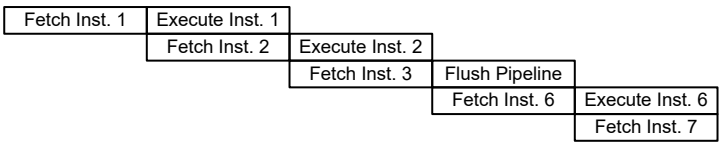
主系统时钟由 HXT、LXT、HIRC 或 LIRC 振荡器提供，它被细分为 T1~T4 四个内部产生的非重叠时序。在 T1 时间，程序计数器自动加一并抓取一条新的指令。剩下的时间 T2~T4 完成译码和执行功能，因此，一个 T1~T4 时钟周期构成一个指令周期。虽然指令的抓取和执行发生在连续的指令周期，但单片机流水线结构会保证指令在一个指令周期内被有效执行。除非程序计数器的内容被改变，如子程序的调用或跳转，在这种情况下指令将需要多一个指令周期的时间去执行。



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非连续的程序存储器地址之外，它会在每条指令执行完成以后自动加一。对于存储器容量大于 8K 的单片机，程序存储器地址可能位于某一程序存储区，可通过程序存储区指针的 PBP1~PBP0 位来选择。只有较低的 8 位，即所谓的程序计数器低字节寄存器 PCL，可以被用户直接读写。

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

程序计数器	
高字节	低字节 (PCL)
PBP1~PBP0, PC12~PC8	PCL7~PCL0

程序计数器

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

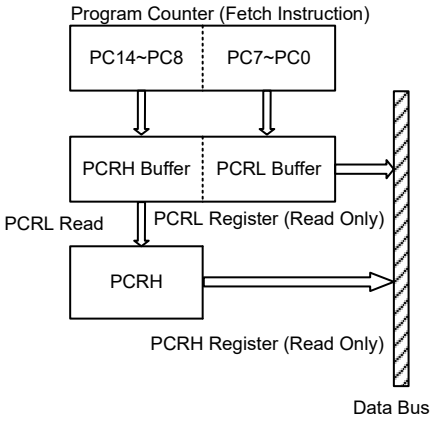
程序计数器读寄存器

程序计数器读寄存器为只读寄存器，用于读取目前程序执行地址的计数器值。先读低字节寄存器再读高字节寄存器，即读取目前程序执行地址的低字节，同时将程序计数器的高字节数据放置在 8-bit PCRH 缓存器。然后读取 PCRH 寄存器，从 8-bit PCRH 缓存器中读取数据。

下面举例说明如何读取目前程序执行地址。当目前程序执行地址为 123H 时，执行指令步骤如下：

(1) 执行 MOV A, PCRL 指令之后 → ACC 值为 23H，并且 PCRH 值为 01H；
执行 MOV A, PCRH 指令之后 → ACC 值为 01H。

(2) 执行 LMOV A, PCRL 指令之后 → ACC 值为 23H，并且 PCRH 值为 01H；
执行 LMOV A, PCRH 指令之后 → ACC 值为 01H。



● **PCRL 寄存器**

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

Bit 7~0 **D7~D0**：程序计数器读低字节寄存器 bit 7 ~ bit 0

● **PCRH 寄存器**

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

Bit 7 未定义，读为“0”

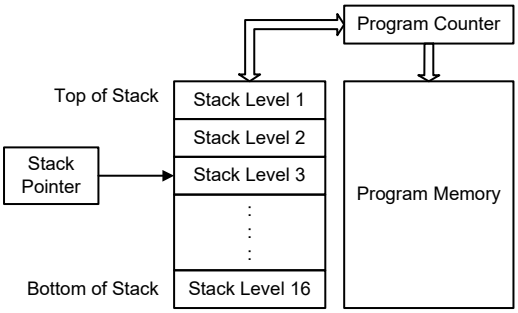
Bit 6~0 **D14~D8**：程序计数器读高字节寄存器 bit 6 ~ bit 0

堆栈

堆栈是一个特殊的存储空间，用来存储程序计数器中的内容。该单片机有 16 层堆栈，堆栈既不是数据部分也不是程序空间部分，而且它既不是可读取也不是可写入的。当前层由堆栈指针 STKPTR[3:0] 加以指示。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。



● **STKPTR 寄存器**

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

Bit 7 **OSF**：堆栈溢出标志位
0：未发生堆栈溢出
1：发生堆栈溢出
当堆栈已满，再次执行 CALL 指令时，或当堆栈为空，再次执行 RET 指令时，

OSF 位都会被置为 1。该位只能通过软件清零，硬件不会自动复位。

Bit 6~4　未定义，读为“0”

Bit 3~0　**D3~D0**：堆栈指针寄存器 bit 3 ~ bit 0

下面举例说明当发生程序分支跳转时堆栈指针及溢出标志位是如何变化的。

(1) 连续执行 17 次 CALL 指令，期间未执行 RET 指令，STKPTR[3:0] 及 OSF 位的变化如下：

CALL 执行次数	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
STKPTR[3:0] 值	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1
OSF 值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(2) 当 OSF 为 1 时，若不清除 OSF 位，则不管执行几次 RET 指令，OSF 位会一直为 1。

(3) 当堆栈为空时，连续执行 16 次 RET 指令，STKPTR[3:0] 及 OSF 位的变化如下：

RET 执行次数	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
STKPTR[3:0] 值	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSF 值	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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，LANDM，LOR，LORM，LXOR，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：

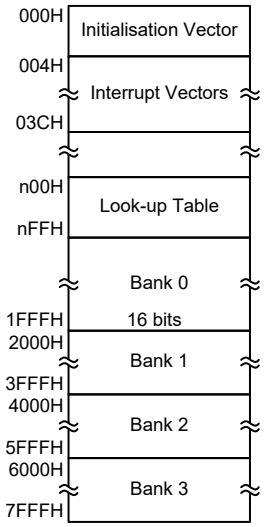
  JMP，SZ，SZA，SNZ，SIZ，SDZ，SIZA，SDZA，CALL，RET，RETI，
  LSZ，LSZA，LSNZ，LSIZ，LSIZA，LSDZ，LSDZA

Flash 程序存储器

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

结构

程序存储器的容量为 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。



程序存储器结构

特殊向量

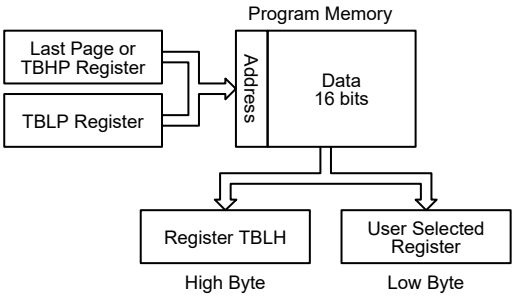
程序存储器内部某些地址保留用做诸如复位和中断入口等特殊用途。地址 000H 是芯片复位后的程序起始地址。在芯片复位之后，程序将跳到这个地址并开始执行。

查表

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

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

下图是查表中寻址 / 数据流程：



查表范例

以下范例说明表格指针和表格数据如何被定义和执行。这个例子使用的表格数据用 ORG 伪指令储存在存储器中。ORG 指令的值“1F00H”位于 ROM Bank 3，指向的地址是 32K 程序存储器中最后一页的起始地址。表格指针低字节寄存器的初始值设为 06H，这可保证从数据表格读取的第一笔数据位于程序存储器地址 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]”或“LTABRD [m]”指令被使用，则表格指针指向 TBHP 和 TBLP 指定的地址。在这个例子中，表格数据的高字节等于零，而当“TABRD [m]”或“LTABRD [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 / 可写寄存器，且能重复储存，若主程序和中断服务程序都使用表格读取指令，应该注意它的保护。使用表格读取指令，中断服务程序可能会改变 TBLH 的值，若随后在主程序中再次使用这个值，则会发生错误，因此建议避免同时使用表格读取指令。然而在某些情况下，如果同时使用表格读取指令是不可避免的，则在执行任何主程序的表格读取指令前，中断应该先除能，另外要注意的是所有与表格相关的指令，都需要两个指令周期去完成操作。

表格读取程序范例

```
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
:
:
tabrd tempreg1      ; transfers value in table referenced by table pointer
                    ; data at program
                    ; memory address "7F06H" transferred to tempreg1 and
TBLH
dec tblp            ; reduce value of table pointer by one
tabrd tempreg2      ; transfers value in table referenced by table pointer
                    ; data at program
                    ; memory address "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 last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

在线烧录 – ICP

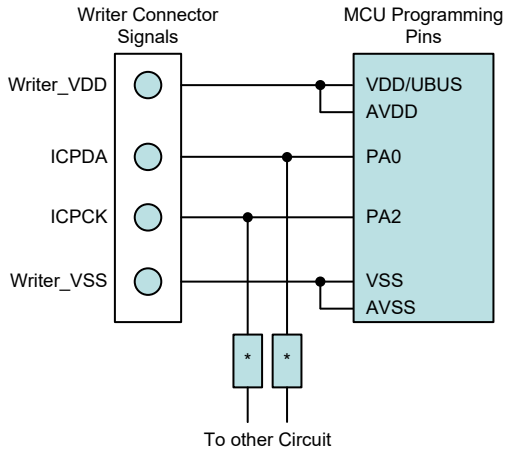
Flash 型程序存储器提供用户便利地对同一芯片进行程序的更新和修改。

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

Holtek 烧录器引脚名称	MCU 在线烧录引脚名称	引脚描述
ICPDA	PA0	串行数据 / 地址烧录
ICPCK	PA2	时钟烧录
VDD	VDD/UBUS & AVDD	电源
VSS	VSS & AVSS	地

程序存储器可以通过 4 线的接口在线进行烧录。其中一条线用于数据串行下载或上传、一条线用于串行时钟、剩下两条用于提供电源。芯片在线烧写的详细使用说明超出此文档的描述范围，将由专门的参考文献提供。

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



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

片上调试 – OCDS

该单片机还提供片上调试功能 (OCDS) 用于开发过程中的 MCU 调试。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 OCDS 功能进行调试时，单片机 OCDSDA 和 OCDSCK 引脚上的其它共用功能无效。由于这两个 OCDS 引脚与 ICP 引脚共用，因此在线烧录时仍用作 Flash 存储器烧录引脚。关于 OCDS 功能的详细描述，请参考“Holtek e-Link for 8-bit MCU OCDS 使用手册”文件。

Holtek e-Link 引脚名称	MCU OCDS 引脚名称	引脚描述
OCDSDA	OCDSDA	片上调试串行数据 / 地址输入 / 输出
OCDSCK	OCDSCK	片上调试时钟输入
VDD	VDD/UBUS & AVDD	电源
VSS	VSS & AVSS	地

在线应用编程 – IAP

Flash 型程序存储器便于用户在同一芯片上对程序进行更新和修改。单片机不仅提供 IAP 功能，而且还提供 ISP 功能，这些功能使用户可以方便地对 Flash 程序存储器进行多次编程。IAP 功能可以通过内部固件进行程序的更新，而无需外接烧录器或 PC。此外，IAP 接口通过 I/O 引脚可以设置为任何类型的通信协议，例如 UART 或 USB。关于内部固件，用户可以选择 Holtek 提供的版本或创建自己的内部固件。以下章节说明了如何实现 IAP 固件程序。

Flash 存储器读 / 写大小

Flash 存储器以页为单位进行擦 / 写操作，以字为单位进行读出操作。页的大小和写入缓冲器的大小都为 64 字。注意，在执行写入操作之前必须先执行擦除操作。

Flash 存储器擦 / 写功能成功使能时 CFWEN 位会被硬件置高，当该位被置高，便可写入数据到“写入缓冲器”。FWT 位用于启动写入程序，并指示写入操作的状态。当该位由应用程序置高时将开始一个写入程序，当写入操作结束后该位将由硬件清零。

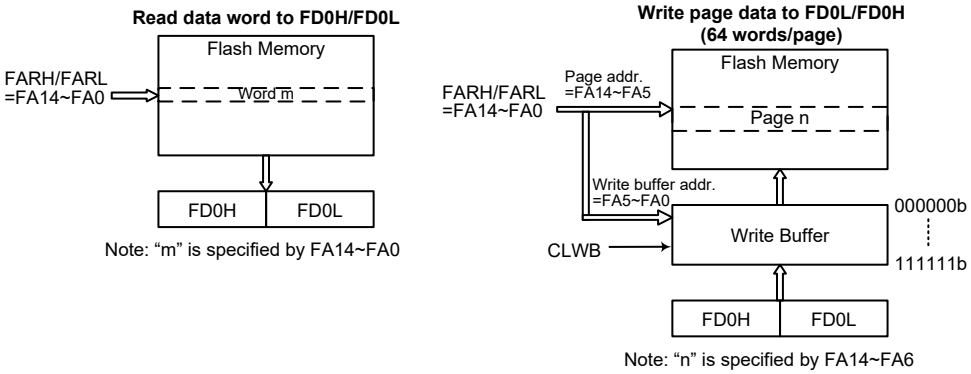
读出操作是通过一个特定的读出程序来执行的。FRDEN 位用于使能读出功能，由应用程序设置 FRD 位来启动读出程序，并指示读出操作的状态。当读出操作结束后该位将由硬件清零。

操作	格式
擦除	64 字 / 页
写入	64 字 / 次
读出	1 字 / 次
注：页大小 = 写入缓冲器大小 = 64 字。	

IAP 操作格式

页	FARH	FARL[7:6]	FARL[5:0]
0	0000 0000	00	标记地址
1	0000 0000	01	
2	0000 0000	10	
3	0000 0000	11	
4	0000 0000	00	
: :	: :	: :	
510	0111 1111	10	
511	0111 1111	11	

页序号和地址选择



Flash 存储器 IAP 读 / 写结构

写入缓冲器

执行写入操作时写入缓冲器用于临时存储写入的数据。通过执行 Flash 存储器擦 / 写使能程序成功使能 Flash 存储器擦 / 写功能后，才可将要写入的数据填入到“写入缓冲器”。通过配置 FRCR 寄存器中的 CLWB 位可以清除写入缓冲器。置高 CLWB 位可以使能清除写入缓冲器程序，完成后该位会被硬件自动清零。建议第一次使用写入缓冲器或更新写入缓冲器内的数据时，应先置高 CLWB 位将写入缓冲器清零。

写入缓冲器的大小为 64 字，与页的大小一致。写入缓冲器的地址与存储器地址位 FA14~FA6 指定的 Flash 存储器页的地址相对应。写入到 FD0L 和 FD0H 寄存器的数据会被加载到写入缓冲器。当写入数据到高字节数据寄存器 FD0H，会使存储在高 / 低字节数据寄存器内的数据都写入到“写入缓冲器”，并使 Flash 存储器地址自动加一，之后新的地址会被加载到 FARH 和 FARL 地址寄存器。当 Flash 存储器地址到达当前页的最大地址，即 64 字的页为 111111b，地址将不再增加，并停在该页的最后一个地址，此时需要再设定一个新的页地址才可进行擦 / 写操作。

写入程序结束后，硬件会自动清除写入缓冲器。注意，如果数据比对步骤发现写入到 Flash 存储器的数据不正确时，需通过应用程序手动清除写入缓冲器，在写入缓冲器被清零之后再重新对其写入数据。

IAP Flash 程序存储器寄存器

IAP Flash 程序存储器有两个地址寄存器、四对 16 位数据寄存器和两个控制寄存器，它们都位于 Sector 1。使用地址、数据和控制寄存器可以实现对 Flash 存储器的 16 位数据读写操作。这几个寄存器控制了内部 Flash 程序存储器所有操作，即地址寄存器 FARL 和 FARH，数据寄存器 FDnL 和 FDnH，控制寄存器 FCR 和 FRCR。由于这些寄存器都位于 Sector 1 中，只能通过扩展指令直接被访问，或者通过存储器指针对 MP1H/MP1L 或 MP2H/MP2L 和间接寻址寄存器 IAR1 或 IAR2 进行间接读取或写入。

寄存器名称	位							
	7	6	5	4	3	2	1	0
FCR	CFWEN	FMOD2	FMOD1	FMOD0	BWT	FWT	FRDEN	FRD
FRCR	D7	D6	—	D4	—	—	—	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	—	FA14	FA13	FA12	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0

寄存器名称	位							
	7	6	5	4	3	2	1	0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP 寄存器列表

- **FCR 寄存器**

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

Bit 7 **CFWEN**：Flash 存储器擦 / 写功能使能控制位
0：Flash 存储器擦 / 写功能除能
1：Flash 存储器擦 / 写功能已成功使能
当该位由应用程序清零时，Flash 存储器擦 / 写功能除能。注意该位不能由应用程序置高，对该位直接写“1”不会使能擦 / 写功能。此位用于指示 Flash 存储器擦 / 写功能的状态。硬件置高此位，表示 Flash 存储器擦 / 写功能成功使能，否则此位为 0，表示 Flash 存储器擦 / 写功能除能。

Bit 6~4 **FMOD2~FMOD0**：Flash 存储器模式选择位
000：写入模式
001：页擦除模式
010：保留
011：读出模式
100：保留
101：保留
110：Flash 存储器擦 / 写使能模式
111：保留
这几位用于选择 Flash 存储器的操作模式。注意在执行擦 / 写 Flash 存储器操作之前必须先成功使能“Flash 存储器擦 / 写使能模式”。

Bit 3 **BWT**：Flash 存储器擦 / 写使能程序触发控制位
0：擦 / 写使能程序未被触发或程序定时器溢出
1：擦 / 写使能程序被触发且程序定时器开始计时
该位用于启动 Flash 存储器擦 / 写使能程序和内部定时器。此位由应用程序置高，当内部定时器计时溢出后由硬件清零。需在 BWT 置高后尽快写入正确数据序列到 FD1L/FD1H、FD2L/FD2H 和 FD3L/FD3H 寄存器

Bit 2 **FWT**：Flash 存储器写入控制位
0：不启动 Flash 存储器写入程序或写入程序已完成
1：启动 Flash 存储器写入程序
此位由软件置高，当 Flash 存储器写入程序结束后由硬件清零。

Bit 1 **FRDEN**：Flash 存储器读出使能位
0：除能
1：使能
此位为 Flash 存储器读出使能位，在执行 Flash 存储器读出操作之前需将此位置高。将此位清零则禁止 Flash 存储器读出操作。

Bit 0 **FRD**：Flash 存储器读出控制位
0：不启动 Flash 存储器读出程序或读出程序已完成
1：启动 Flash 存储器读出程序
此位由软件置高，当 Flash 存储器读出程序结束后由硬件清零。

注：1. 在同一条指令中 FWT、FRDEN 和 FRD 位不可同时设置为“1”。
2. 确保 f_{SUB} 时钟在执行擦或写动作前已稳定。
3. 当读、擦或写动作成功启动后，CPU 相关操作将停止。
4. 确保读、擦或写动作成功完成后才可执行其它操作。

● FRCR 寄存器

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	—	D4	—	—	FWERTS	CLWB
R/W	R/W	R	—	R/W	—	—	R/W	R/W
POR	0	0	—	0	—	—	0	0

Bit 7 **D7**：保留位，不可用且固定为“0”
Bit 6 **D6**：保留位
Bit 5 未定义，读为“0”
Bit 4 **D4**：保留位，不可用且固定为“0”
Bit 3~2 未定义，读为“0”
Bit 1 **FWERTS**：擦除时间和写入时间选择位
0：擦除时间为 3.2ms (t_{FER}) / 写入时间为 2.2ms (t_{FWR})
1：擦除时间为 3.7ms (t_{FER}) / 写入时间为 3.0ms (t_{FWR})
Bit 0 **CLWB**：Flash 存储器写入缓冲器清除控制位
0：不启动清除写入缓冲器程序或清除程序已完成
1：启动清除写入缓冲器程序
此位由软件置高，当清除写入缓冲器程序结束后由硬件清零。

● FARL 寄存器

Bit	7	6	5	4	3	2	1	0
Name	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
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 **FA7~FA0**：Flash 程序存储器地址 bit 7 ~ bit 0

● FARH 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	FA14	FA13	FA12	FA11	FA10	FA9	FA8
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 **FA14~FA8**：Flash 程序存储器地址 bit 14 ~ bit 8

● FD0L 寄存器

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

Bit 7~0 **D7~D0**：第一个 Flash 存储器数据 bit 7 ~ bit 0
注意写入低字节数据寄存器 FD0L 的数据只能存储在 FD0L 寄存器，不能加载到

低 8 位写入缓冲器。

- **FD0H 寄存器**

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

Bit 7~0 **D15~D8**：第一个 Flash 存储器数据 bit 15 ~ bit 8

注意当写入 8 位数据到高字节数据寄存器 FD0H 时，存储在 FD0H 和 FD0L 寄存器内的 16 位数据将同时加载到 16 位写入缓冲器，此时 Flash 存储器地址寄存器 FARH 和 FARL 的内容将自动加一。

- **FD1L 寄存器**

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

Bit 7~0 **D7~D0**：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 **D15~D8**：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 **D7~D0**：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 **D15~D8**：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 **D7~D0**：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 **D15~D8**：第四个 Flash 存储器数据 bit 15 ~ bit 8

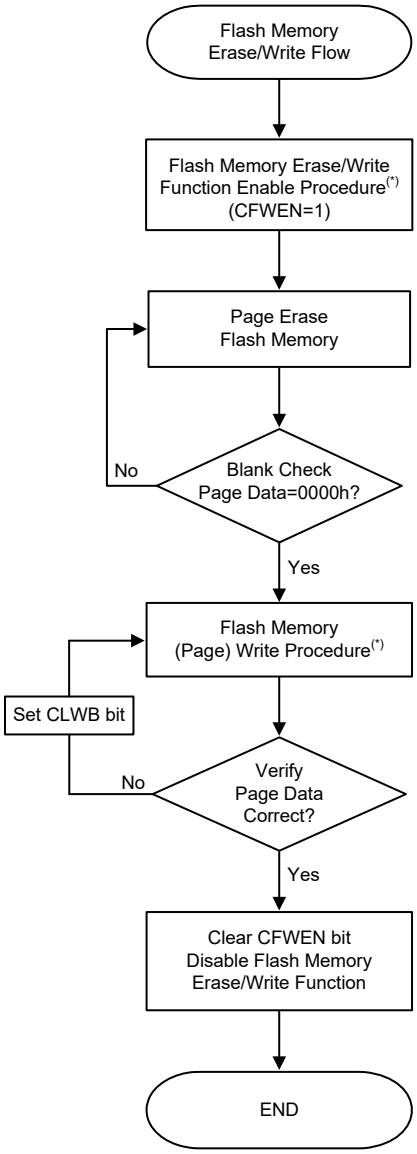
Flash 存储器擦 / 写流程

在开始更新 Flash 存储器之前，先了解 Flash 存储器擦 / 写流程操作是很重要的，用户可参考下列步骤进行程序开发，以确保 IAP 功能擦 / 写 Flash 存储器内容更新正确。

Flash 存储器擦 / 写流程说明

1. 先启动“Flash 存储器擦 / 写使能程序”。当 Flash 存储器擦 / 写功能成功使能后，FCR 寄存器中的 CFWEN 位会由硬件自动置高，此时才可执行擦 / 写 Flash 存储器操作。详细内容请参考“Flash 存储器擦 / 写使能程序”。
2. 配置 Flash 存储器地址指定要擦除的页，标记地址，然后擦除此页。

   对于页擦除操作，首先设置 FARL 和 FARH 寄存器来指定要擦除页的起始地址，然后写入任意数据到 FD0H 寄存器来标记地址。每写入一个任意数据到 FD0H 寄存器，当前地址将自动加一。当地址自动递增到当前页的最大地址，即 111111b，地址将不再增加，并停在该页的最后一个地址。注意写数据到 FD0H 是为了标记地址，这一操作必须执行以确定要擦除哪些地址。
3. 查空确认是否擦除成功，可采用 TABRD 指令进行读取并比对是否为“0000h”，如果擦除不成功返回步骤 2 再擦除一次。
4. 写入数据至该页，详细内容请参考“Flash 存储器写入程序”。
5. 采用 TABRD 指令进行读取并比对写入数据是否正确，如果写入不成功，设置 CLWB 位为“1”清除“写入缓冲器”再返回步骤 4，再写入相同数据。
6. 完成当前页擦 / 写步骤后，如果无需擦 / 写其它页，可清除 CFWEN 位来解除“Flash 存储器擦 / 写使能模式”。



Flash 存储器擦 / 写流程图

注：标上 * 的“Flash 存储器擦 / 写使能程序”及“Flash 存储器写入程序”详见后续章节。

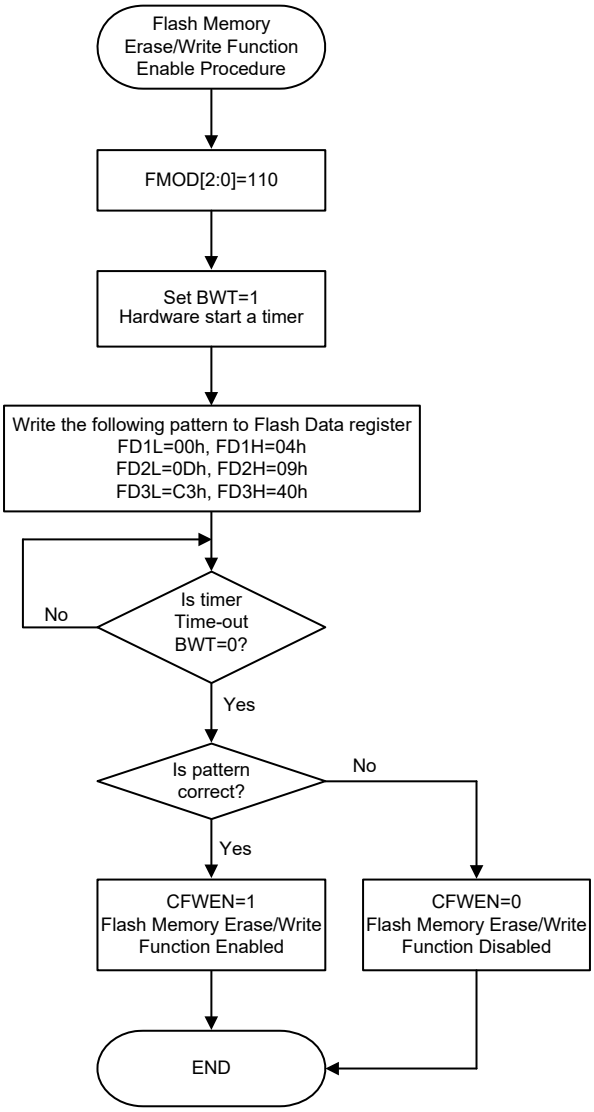
Flash 存储器擦 / 写使能程序

Flash 存储器擦 / 写使能模式是专门为保护 Flash 存储器内容不被轻易地修改而设计的。用户必须先使能 Flash 存储器擦 / 写功能，才能通过 IAP 控制寄存器来更改 Flash 存储器数据。

Flash 存储器擦 / 写使能程序说明

1. 写入数值“110”至 FCR 寄存器中的 FMOD[2:0] 位，选择 Flash 存储器擦 / 写使能模式。
2. 设置 FCR 寄存器中的 BWT 位为“1”，启动 Flash 存储器擦 / 写使能程序，此时内部硬件线路会启动内部定时器。
3. 使用者必须在 BWT 位置高后尽快填入正确数据序列至 FD1L~FD3L 和 FD1H~FD3H 寄存器中，数据序列对应为 FD1L=00h、FD1H=04h、FD2L=0Dh、FD2H=09h、FD3L=C3h、FD3H=40h。
4. 一旦定时器计时结束，无论写入的数据序列是否正确，BWT 位将由硬件自动清零。
5. 如果写入的数据序列不正确，表示 Flash 存储器擦 / 写功能没有成功使能，需重复以上步骤。如果写入的数据序列正确，表示 Flash 存储器擦 / 写功能成功使能。
6. 一旦 Flash 存储器擦 / 写功能成功使能，即可通过 IAP 控制寄存器进行页擦 / 写操作来更新 Flash 存储器内容。

将 FCR 寄存器中的 CFWEN 位清零，可除能 Flash 存储器擦 / 写功能，不必再执行以上步骤。



Flash 存储器擦 / 写使能程序

Flash 存储器写入程序

当 Flash 擦 / 写功能成功使能后，CFWEN 位会被硬件置高，此时要写入 Flash 存储器的数据才能加载到“写入缓冲器”。在开始写入程序之前，应先正确配置 IAP 控制寄存器，将所选的 Flash 存储器页的数据擦除。

写入缓冲器的大小为每页 64 字，其地址与 FA14~FA6 指定的 Flash 存储器页的地址为相对应关系。注意“写入缓冲器”的地址与对应存储器的地址必须在相同页。

Flash 存储器连续地址写入程序说明

对于写入操作每次写入的数据为 64 字。多笔连续地址的数据写入时，写入缓冲器的地址将自动加“1”。用户只需将第一笔数据的地址填入 FARL、FARH，并将第一笔数据依序填入 FD0L、FD0H (先写 FD0L 再写 FD0H，才会将 FD0L、FD0H 数据一起填入“写入缓冲器”)，写入缓冲器的地址将自动加“1”，要

填入第二笔数据时，可不用再指定地址 FARL、FARH。当连续地址到达当前页的最后一个地址时，写入缓冲器的地址将不会再自动加“1”，地址将保持为最后一个地址。

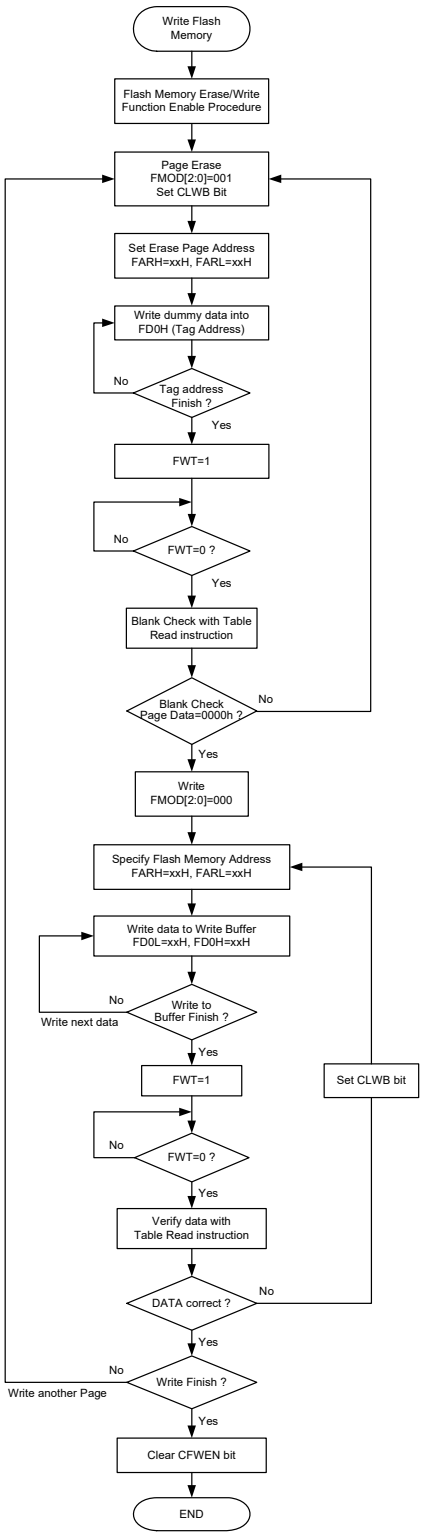
1. 启动“Flash 存储器擦 / 写使能程序”，确认 CFWEN 的值，如果 CFWEN 被硬件置高，表示可进行 IAP 擦 / 写操作。详细内容请参考“Flash 存储器擦 / 写使能程序”。
2. 设定 FMOD[2:0] 为“001”，选择擦除模式，并设置 CLWB 位为“1”清除“写入缓冲器”。设定 FWT 位为“1”，擦除由 FARH 和 FARL 指定且已标记地址的目标页，直到 FWT 变为“0”。
3. 通过查表指令读出方式进行查空，以确保擦除操作已成功完成。

   如果擦除操作不成功则返回步骤 2。

   如果擦除操作成功则接着执行步骤 4。
4. 设定 FMOD[2:0] 为“000”，选择写入模式。
5. 先将目标起始地址写入 FARL、FARH 寄存器中，将要往连续地址所在页写入的数据依序写入 FD0L、FD0H 寄存器。最多可写入 64 字。
6. 设定 FWT 位为“1”，将“写入缓冲器”的数据写入到对应的 Flash 存储器中，直到 FWT 变为“0”。
7. 通过查表指令读出方式进行数据比对，以确保写入操作已成功完成。

   如果写入操作不成功，设置 CLWB 位为“1”清除“写入缓冲器”再返回步骤 5。

   如果写入操作成功则接着执行步骤 8。
8. 将 CFWEN 位清零以除能 Flash 存储器擦 / 写功能。



Flash 存储器连续地址写入程序

注：1. 当擦或写动作成功启动后，所有 CPU 相关操作将暂停。
2. FWT 位由高变低所需时间可以通过 FRCR 寄存器中的 FWERTS 位选择。

Flash 存储器非连续地址写入程序说明

连续地址写入操作与非连续地址写入操作的主要差别在于要写入的数据是否位于连续地址。如果要写入的数据不是位于连续的地址，当一笔数据成功写入到 Flash 存储器后需重新配置另一个目标地址。

以两笔非连续的数据写入操作为例，说明如下：

1. 启动“Flash 存储器擦 / 写使能程序”，确认 CFWEN 位的值，如果 CFWEN 被硬件置高，表示可进行 IAP 擦 / 写操作。详细内容请参考“Flash 存储器擦 / 写使能程序”。
2. 设定 FMOD[2:0] 为“001”，选择擦除模式，并设置 CLWB 位为“1”清除“写入缓冲器”。设定 FWT 位为“1”，擦除由 FARH 和 FARL 指定且已标记地址的目标页，直到 FWT 变为“0”。
3. 通过查表指令读出方式进行查空，以确保擦除操作已成功完成。

   如果擦除操作不成功则返回步骤 2。

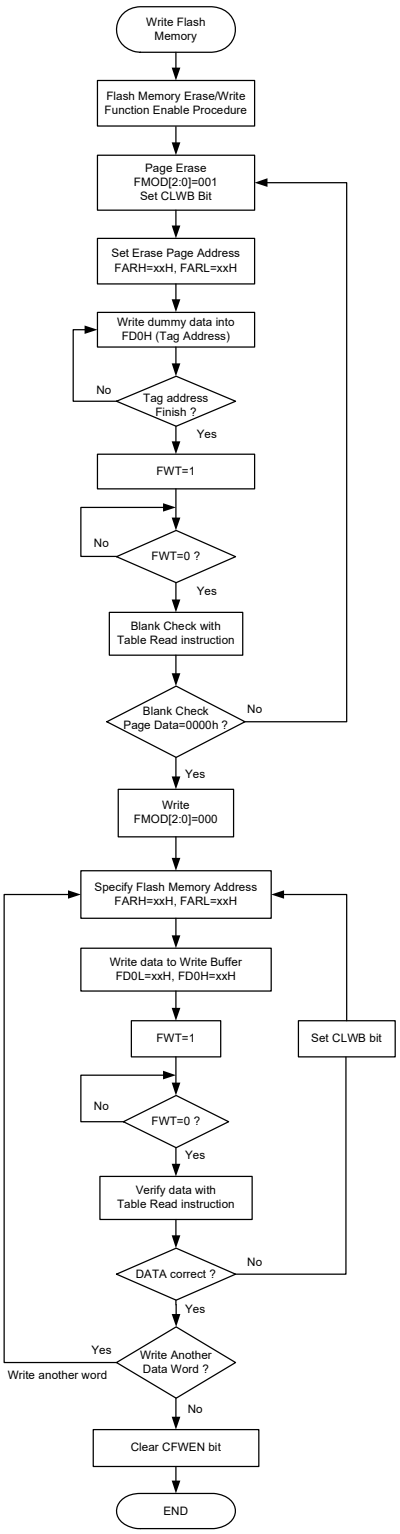
   如果擦除操作成功则接着执行步骤 4。
4. 设定 FMOD[2:0] 为“000”，选择写入模式。
5. 先将目标地址 ADDR1 写入 FARL、FARH 寄存器中，将要写入的数据 DATA1 先写入 FD0L 寄存器再写入 FD0H 寄存器。
6. 设定 FWT 位为“1”，将“写入缓冲器”的数据写入到对应的 Flash 存储器中，直到 FWT 变为“0”。
7. 通过查表指令读出方式进行数据比对，以确保写入操作已成功完成。

   如果写入操作不成功，设置 CLWB 位为“1”清除“写入缓冲器”再返回步骤 5。

   如果写入操作成功则接着执行步骤 8。
8. 再将目标地址 ADDR2 写入 FARL、FARH 寄存器中，将要写入的数据 DATA2 先写入 FD0L 寄存器再写入 FD0H 寄存器。
9. 设定 FWT 位为“1”，将“写入缓冲器”的数据写入到对应的 Flash 存储器中，直到 FWT 变为“0”。
10. 通过查表指令读出方式进行数据比对，以确保写入操作已成功完成。

    如果写入操作不成功，设置 CLWB 位为“1”清除“写入缓冲器”再返回步骤 8。

    如果写入操作成功则接着执行步骤 11。
11. 将 CFWEN 位清零以除能 Flash 存储器擦 / 写功能。



Flash 存储器非连续地址写入程序

注：1. 当擦或写动作成功启动后，所有 CPU 相关操作将暂停。

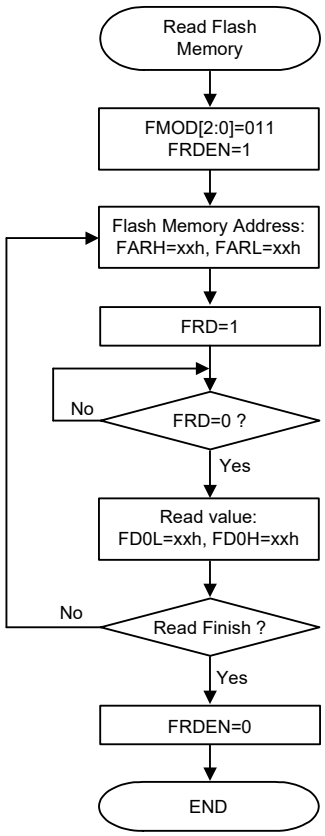
2. 在擦除或写入操作中，FWT 位由高变低所需时间可以通过 FRCR 寄存器中的 FWERTS 位选择。

Flash 存储器写入操作注意事项

1. 要开始对 Flash 存储器进行 IAP 擦 / 写操作之前，必须先完成“Flash 存储器擦 / 写使能程序”。
2. Flash 存储器擦除操作以页作为擦除单位。
3. “写入缓冲器”以页为单位对 Flash 存储器写入数据，且写入时不可跨页填写。
4. 使用 IAP 功能过程中，需确保 FRCR 寄存器的 Bit 7~2 都为“0”，以避免不可预期的错误。
5. 数据写入 Flash 存储器后，必须以查表指令“TABRD”读出方式比对所写数据是否正确，若比对写入数据不正确时，通过置高 CLWB 位将“写入缓冲器”清除后重新写入数据，且不清除 Flash 存储器直接再写入，然后再比对，直到写入正确。
6. IAP 写入与数据比对时需与最高应用频率相同。

Flash 存储器读出程序

要启动 Flash 存储器读出程序，需将 FMOD[2:0] 位设为“011”选择 Flash 存储器读出模式，将 FRDEN 位设为“1”使能读出功能。将要读出的地址填入 FARH、FARL 地址寄存器中，并将 FRD 位设为“1”，然后便可开始 Flash 存储器读出操作。当 FRD 被硬件清为“0”时，则可在 FD0H、FD0L 取得 Flash 存储器中该地址数据。进行 Flash 存储器读出操作前，不需经过 Flash 存储器擦 / 写使能程序。



Flash 存储器读出程序

注：1. 当读动作成功启动后，所有 CPU 相关操作将暂停。
2. FRD 位由高变低所需时间为 3 个指令周期 (典型值)。

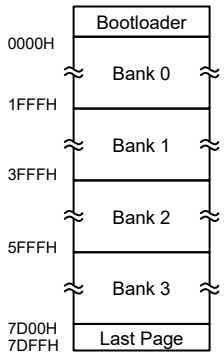
在线系统烧录 – ISP

另外，Holtek 单片机提供 2 线 USB 接口的在线系统烧录方式。用户可将进行过烧录或未经过烧录的单片机芯片连同电路板一起制成，最后阶段进行程序的更新和程序的烧录，在拔出再重新插入芯片的情况下方便地保持程序为最新版。

芯片内部程序存储器可以通过 USB 接口在系统进行烧录，即使用 UDN 和 UDP 引脚。电源由 UBUS 引脚提供。芯片在线系统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。Flash 程序存储器的读 / 写功能由一系列寄存器实现。

ISP Bootloader

单片机提供 ISP Bootloader 功能来升级 Flash 存储器的软件。用户可以选择使用 Holtek IDE 工具提供的 ISP Bootloader 软件或创建自己的 Bootloader。若选择 Holtek 的 Bootloader 软件，将占用 0.5K 字的 Flash 存储空间。下图为带 Holtek Bootloader 软件的 Flash 存储器结构图。



带 Bootloader 的 Flash 程序存储器结构

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。

数据存储器分为两部分，第一部分是特殊功能数据存储器。这些寄存器有固定的地址且与单片机的正确操作密切相关。大多特殊功能寄存器都可在程序控制下直接读取和写入，但有些被加以保护而不对用户开放。第二部分数据存储器是做一般用途使用，都可在程序控制下进行读取和写入。

该单片机还提供了专门的存储区用于存放 LCD 显示数据。本章节仅对通用功能数据存储器和特殊功能寄存器存储器进行介绍。关于 LCD 显示数据存储器的使用可参考 LCD 驱动器章节内容。

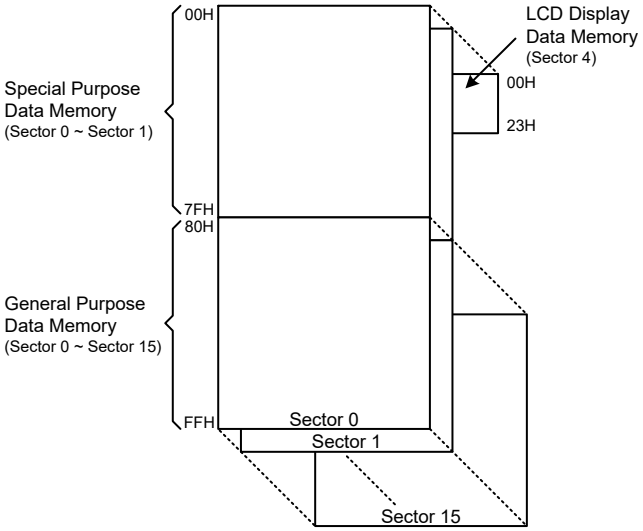
结构

数据存储器被分为若干个 Sector，都位于 8 位存储器中。每个数据存储器 Sector 分为两类，即特殊功能数据存储器和通用数据存储器。特殊功能数据寄存器地址范围为 00H~7FH，而通用数据存储器地址范围为 80H~FFH。LCD 显示数据存储器位于 Sector 4 的 00H~23H。

若用间接寻址方式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。

特殊功能数据存储器	通用数据存储器		LCD 显示数据存储器	
所在 Sector	容量	Sector：地址	容量	Sector：地址
0，1	2048×8	0：80H~FFH 1：80H~FFH : 15：80H~FFH	36×8	4：00H~23H

数据存储器概要



数据存储器结构

数据存储器寻址

此单片机支持扩展指令架构，没有专门用于数据存储器寻址的存储区指针寄存器。存储区指针 PBP 仅适用于程序存储器。对于数据存储器，使用间接寻址访问方式时所要访问的 Sector 是通过 MP1H 或 MP2H 寄存器指定，而所选 Sector 的具体数据存储器地址的选择是通过 MP1L 或 MP2L 寄存器指定。

直接寻址可用于所有 Sector，通过扩展指令可以寻址所有可用的数据存储器空间。当所访问的数据存储器位于除 Sector 0 外的任何数据存储器 Sector 时，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”有 12 个有效位，高字节表示选择的 Sector，低字节表示指定的地址。

通用数据存储器

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

特殊功能数据存储器

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

	Sector 0	Sector 1
00H	IAR0	PD
01H	MP0	PDC
02H	IAR1	PDPU
03H	MP1L	PE
04H	MP1H	PEC
05H	ACC	PEPU
06H	PCL	PF
07H	TBLP	PFC
08H	TBLH	PFPU
09H	TBHP	PG
0AH	STATUS	PGC
0BH	PBP	PGPU
0CH	IAR2	PH
0DH	MP2L	PHC
0EH	MP2H	PHPU
0FH	RSTFC	
10H	SCC	FCR
11H	HIRCC	FRCR
12H	HXTC	FARL
13H	LXTC	FARH
14H	PA	FD0L
15H	PAC	FD0H
16H	PAPU	FD1L
17H	PAWU	FD1H
18H	RSTC	FD2L
19H	LVRC	FD2H
1AH	TLVRC	FD3L
1BH	MFI0	FD3H
1CH	MFI1	
1DH	MFI2	
1EH		
1FH	WDTC	
20H	INTEG	SIM0C0
21H	INTC0	SIM0C1/U0UCR1
22H	INTC1	SIM0D/U0TXR_RXR
23H	INTC2	SIM0A/SIM0C2/U0UCR2
24H	INTC3	U0UCR3
25H	PB	SIM0TOC/U0BRG
26H	PBC	U0USR
27H	PBPU	SIM1C0
28H	PC	SIM1C1/U1UCR1
29H	PCC	SIM1D/U1TXR_RXR
2AH	PCPU	SIM1A/SIM1C2/U1UCR2
2BH	PSCR	U1UCR3
2CH	TB0C	SIM1TOC/U1BRG
2DH	TB1C	U1USR
2EH		SPIC0
2FH		SPIC1
30H	IREFC	SPID
31H	PVREF	
32H	OPA1C	
33H	OPA2C	
34H	GSC	
35H	AFEDAC	
36H	AFEDA1L	
37H	AFEDA1H	
38H	AFEDA2L	
39H	AFEDA2H	
3AH	SADC0	
3BH	SADC1	
3CH	SADC2	
3DH	SADOL	
3EH	SADOH	
3FH		

	Sector 0	Sector 1
40H	EEAL	EEC
41H	EEAH	STKPTR
42H	EED	IECC
43H	OPSW	PCRL
44H	ASWAI0	PCRH
45H	ASWAI1	CRCCR
46H	ASWAI2	CRCIN
47H	ASWAI3	CRCDL
48H	ASWAI4	CRCDH
49H	ASWAI5	
4AH	ASWAI6	
4BH	ASWAI7	
4CH		
4DH		
4EH		
4FH		
50H	ATMC0	STMC0
51H	ATMC1	STMC1
52H	ATMDL	STMDL
53H	ATMDH	STMDH
54H	ATMAL	STMAL
55H	ATMAH	STMAH
56H	ATMBL	STMRP
57H	ATMBH	PTM0C0
58H	ATMRP	PTM0C1
59H	MDUWR0	PTM0DL
5AH	MDUWR1	PTM0DH
5BH	MDUWR2	PTM0AL
5CH	MDUWR3	PTM0AH
5DH	MDUWR4	PTM0RPL
5EH	MDUWR5	PTM0RPH
5FH	MDUWCTRL	PTM1C0
60H	SYSC	PTM1C1
61H	USB_STAT	PTM1DL
62H	UINT	PTM1DH
63H	USC	PTM1AL
64H	UESR	PTM1AH
65H	UCC	PTM1RPL
66H	AWR	PTM1RPH
67H	STL	IFS
68H	SIES	PAS0
69H	MISC	PAS1
6AH	SETIO	PBS0
6BH	FIFO3	PBS1
6CH	FIFO2	PCS0
6DH	FIFO1	PCS1
6EH	FIFO0	PDS0
6FH	FRNUM0	PDS1
70H	FRNUM1	PES0
71H	PLLC	PES1
72H	LCDC0	PFS0
73H	LCDC2	PFS1
74H		PGS0
75H		PGS1
76H		PHS0
77H		
78H		COMS
79H		PMPS
7AH		
7BH		
7CH		
7DH		
7EH		
7FH	ORMC	

: Unused, read as 00H

: Reserved, cannot be changed

特殊功能数据存储器

特殊功能寄存器

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

间接寻址寄存器 – IAR0, IAR1, IAR2

间接寻址寄存器 IAR0、IAR1 和 IAR2 的地址虽位于数据存储区，但不同于普通寄存器，它们没有实际的物理地址。与定义实际存储器地址的直接存储器寻址不同，间接寻址是使用间接寻址寄存器和存储器指针来执行存储器数据操作。在间接寻址寄存器 IAR0、IAR1 和 IAR2 上的任何动作，将对存储器指针 MP0、MP1L/MP1H 或 MP2L/MP2H 所指定的存储器地址产生对应的读 / 写操作。它们总是成对出现，IAR0 和 MP0 可以访问 Sector 0，而 IAR1 和 MP1L/MP1H、IAR2 和 MP2L/MP2H 可以访问任何 Sector。因为这些间接寻址寄存器不是实际存在的，直接读取将返回“00H”的结果，而直接写入此寄存器则不做任何操作。

存储器指针 – MP0, MP1H/MP1L, MP2H/MP2L

该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存储器指针所指定的地址。MP0、IAR0 用于访问 Sector 0，而 MP1L/MP1H 和 IAR1、MP2L/MP2H 和 IAR2 可根据 MP1H 或 MP2H 寄存器访问所有的 Sector。使用扩展指令可对所有的数据 Sector 进行直接寻址。

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

间接寻址程序范例

范例 1

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

范例 2

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
code .section at 0 'code'
org 00h
start:
    mov a,04h              ; setup size of block
    mov block,a
    mov a,01h              ; setup the memory sector
    mov 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 MP1L
    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范例

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

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

程序存储区指针 – PBP

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

● **PBP 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PBP1	PBP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未定义，读为“0”

Bit 1~0　**PBP1~PBP0**：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。

Option 存储器映射寄存器 – ORMC

ORMC 寄存器用于使能 Option 存储器映射功能。Option 存储器的容量为 64 个字。当连续写入特定数据序列 55H 和 AAH 到该寄存器，Option 存储器映射功能将使能，通过使用查表指令即可读到 Option 存储器的内容，Option 存储器的 00H~3FH 地址会一一对应到程序存储器最后一页的 C0H~FFH 地址。

要成功使能 Option 存储器映射功能，该特定的数据序列 55H 和 AAH 必须在两个指令周期内连续写入。建议在写入该特定数据序列前应当先将总中断位 EMI 清零，在数据序列成功写入后，根据用户的需求在适当的时间再将其置高。当数据序列成功写入时会启动内部定时器，4×tLIRC 时间之后会自动结束映射。因此，用户需及时读出数据，否则需要重新启动 Option 存储器映射功能。每次 ORMC 寄存器被连续写入后，定时器都会重新计数。

当使用查表指令来读取 Option 存储器内容时，只可使用“TABRD [m]”指令。然而，若使用“TABRD [m]”指令来读取，必须配置 TBHP 寄存器将表格指针设定在最后一页。更多查表的描述请参考相关章节。

- **ORMC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ORMC7	ORMC6	ORMC5	ORMC4	ORMC3	ORMC2	ORMC1	ORMC0
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 **ORMC7~ORMC0**：Option 存储器映射特定数据序列
当将特定数据序列 55H 和 AAH 连续写入该寄存器，会使能 Option 存储器映射功能。需注意，单片机从空闲 / 休眠模式唤醒后，该寄存器的内容将被清除。

状态寄存器 – STATUS

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

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

SC、CZ、Z、OV、AC 和 C 标志位通常反映最近运算的状态。

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

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/W	R/W	R	R	R/W	R/W	R/W	R/W
POR	x	x	0	0	x	x	x	x

"x"：未知

Bit 7 **SC**：当 OV 与当前指令操作结果 MSB 执行"XOR"所得结果

Bit 6 **CZ**：不同指令不同标志位的操作结果
对于 SUB/SUBM/LSUB/LSUBM 指令，CZ 等于 Z 标志位。
对于 SBC/SBCM/LSBC/LSBCM 指令，CZ 等于上一个 CZ 标志位与当前零标志位执行"AND"所得结果。对于其它指令，CZ 标志位无影响。

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

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

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

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

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

Bit 0 **C**：进位标志位
0：无进位
1：如果在加法运算中结果产生了进位，或在减法运算中结果不发生借位
C 也受循环移位指令的影响。

EEPROM 数据存储器

此单片机的一个特性是内建 EEPROM 数据存储器，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。这种存储区扩展了存储空间，对设计者来说增加了许多新的应用机会。EEPROM 可以用来存储产品编号、校准值、用户特定数据、系统配置参数或其它产品信息等。EEPROM 的数据读取和写入过程也会变的更简单。

EEPROM 数据存储器结构

EEPROM 数据存储器容量为 2048×8 位。由于映射方式与程序存储器和数据存储器不同，因此不能像其它类型的存储器一样寻址。通过 EEC 控制寄存器中的 MODE 模式选择位，可以确定对 EEPROM 进行字节模式或页模式读写操作。

EEPROM 寄存器

有四个寄存器控制内部 EEPROM 数据存储器总的操作，地址寄存器 EEAL 和 EEAH、数据寄存器 EED 及控制寄存器 EEC。EEAL、EEAH 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，仅可通过 MP1L/MP1H 和 IAR1 或 MP2L/MP2H 和 IAR2 进行间接读取或写入。由于 EEC 控制寄存器位于 Sector 1 中的“40H”，在 EEC 寄存器上的任何操作被执行前，MP1L 或 MP2L 必须先设为“40H”，MP1H 或 MP2H 被设为“01H”。

寄存器名称	位							
	7	6	5	4	3	2	1	0
EEAL	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
EEAH	—	—	—	—	—	EEAH2	EEAH1	EEAH0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD

EEPROM 寄存器列表

- **EEAL 寄存器**

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

Bit 7~0 **EEAL7~EEAL0：**数据 EEPROM 地址低字节寄存器 bit 7 ~ bit 0
数据 EEPROM 地址 bit 7 ~ bit 0

- **EEAH 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	EEAH2	EEAH1	EEAH0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 未定义，读为“0”

Bit 2~0 **EEAH2~EEAH0：**数据 EEPROM 地址高字节 bit 2 ~ bit 0
数据 EEPROM 地址 bit 10 ~ bit 8

● **EED 寄存器**

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

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

● **EEC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD
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 **EWERTS**：数据 EEPROM 擦除时间和写入时间选择位
0：擦除时间为 3.2ms (t_{EEER}) / 写入时间为 2.2ms (t_{EEWR})
1：擦除时间为 3.7ms (t_{EEER}) / 写入时间为 3.0ms (t_{EEWR})

Bit 6 **EREN**：数据 EEPROM 擦使能位
0：除能
1：使能
此位用来使能数据 EEPROM 擦功能，向数据 EEPROM 擦操作之前需将此位置高。擦周期结束后，硬件自动将此位清零。将此位清零时，则禁止向数据 EEPROM 擦操作。

Bit 5 **ER**：数据 EEPROM 擦控制位
0：擦周期结束
1：开始擦周期
此位为数据 EEPROM 擦控制位，由应用程序将此位置高将激活擦周期。擦周期结束后，硬件自动将此位清零。当 EREN 未先置高时，此位置高无效。

Bit 4 **MODE**：数据 EEPROM 操作模式选择位
0：字节操作模式
1：页操作模式
此位为数据 EEPROM 操作模式选择位。当此位为高，则选择页写、擦或读操作模式。当此位为 0，则选择字节写或读操作模式。EEPROM 页缓存器大小为 16 字节。

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。无论通过 MODE 位选择了何种写模式，在写操作结束后硬件都会自动将 WREN 位清零。

Bit 2 **WR**：数据 EEPROM 写控制位
0：写周期结束
1：开始写周期
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**：数据 EEPROM 读控制位
0：读周期结束
1：开始读周期
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：1. 在同一条指令中 EREN、ER、WREN、WR、RDEN 和 RD 不能同时置为“1”。
2. 确保 f_{SUB} 时钟在执行擦 / 写动作前已稳定。
3. 确保擦 / 写动作完成后才可改写 EEPROM 相关寄存器或启动 IAP 功能。

EEPROM 读操作

此单片机有两种模式可实现从 EEPROM 中读取数据，即字节读模式和页读模式，可通过 EEC 寄存器中的 EEPROM 操作模式选择位 MODE 选择。

字节读模式

当模式选择位 MODE 为 0 时，可执行 EEPROM 字节读操作。为了实现字节读操作，EEPROM 中要读取数据的地址需先放入 EEAH 和 EEAL 寄存器中，EEC 寄存器中的读使能位 RDEN 也要置高以使能读功能，然后再置高 RD 位以开始 EEPROM 字节读操作。注意，若 RD 位已置高而 RDEN 位还未被置高则不能开始读操作。读周期结束时，RD 位将自动清零，此时可以从 EED 寄存器读取 EEPROM 数据。读到的数据在其它读或写操作执行前将一直保留在 EED 寄存器中。应用程序可轮询 RD 位以确定数据可以有效地被读取。

页读模式

当模式选择位 MODE 为 1 时，可执行 EEPROM 页读操作。页读操作中页大小可达 16 个字节。为了实现页读操作，EEPROM 中要读取页的起始地址需先放入 EEAH 和 EEAL 寄存器中，EEC 寄存器中的读使能位 RDEN 也要置高以使能读功能，然后再置高 RD 位以开始 EEPROM 页读操作。注意，若 RD 位已置高而 RDEN 位还未被置高则不能开始读操作。当前字节读周期结束时，RD 位将自动清零，此时可以从 EED 寄存器中读取 EEPROM 数据，而当前地址将由硬件自动加一。只要再置高 RD 位无需重新配置 EEPROM 地址和 RDEN 控制位，就可以连续读取下一个 EEPROM 地址的数据。应用程序可轮询 RD 位以确定数据可以有效地被读取。

EEPROM 地址高 7 位用来指定要读取页的位置，而低 4 位用来指向实际的地址。在页操作模式低 4 位地址将自动加一，而高 7 位地址不会自动增加。当 EEPROM 地址低 4 位自动递增到当前页的最大地址，即 0FH，EEPROM 地址低 4 位的值会停止在 0FH，EEPROM 地址将不会再增加。

EEPROM 页擦操作

当模式选择位 MODE 为 1 时，可执行 EEPROM 页擦操作。EEPROM 一页可擦除 16 个字节。上电复位后内部页缓存器将由硬件清零。当 EEPROM 擦使能控制位 EREN 由 1 变为 0 时，内部页缓存器也会被清零。注意当 EREN 位由 0 变为 1 时，内部页缓存器不会清零。EEPROM 地址高 7 位用来指定要擦除页的位置，而低 4 位用来指向实际的地址。在页擦操作模式每写入一字节任意数据到 EED 寄存器，低 4 位地址将自动加一，而高 7 位地址不会自动增加。当 EEPROM 地址低 4 位自动递增到当前页的最大地址，即 0FH，EEPROM 地址低 4 位的值会停止在 0FH，EEPROM 地址将不会再增加。

页擦操作需先将 EEPROM 目标页的起始地址放入 EEAH 和 EEAL 寄存器中，再将任意数据放入 EED 寄存器。一页的最大数据长度为 16 字节。注意写数据到 EED 是为了标记地址，这一操作必须执行以确定要擦除哪些地址。当一整页的任意数据都写入 EED 寄存器后，EEC 寄存器中的 EREN 位先置高以使能擦功能，然后 EEC 寄存器中的 ER 位需立即置高以开始擦操作。这两条指令必须在两个指令周期内连续执行才可成功启动一个擦除操作。进行擦除操作之前应

先将总中断使能位 EMI 清零，在一个有效的擦启动步骤完成之后再将其使能。

注：上述步骤必须依序操作，方能成功完成页擦操作，具体请参考对应的范例程序。

由于控制 EEPROM 擦除周期是一个内部时钟，与单片机的系统时钟异步，所以擦除 EEPROM 数据的时间将有所延迟。可通过轮询 EEC 寄存器中的 ER 位或判断 EEPROM 中断以侦测擦周期是否完成。若擦除周期完成，ER 位将自动清零，通知用户数据已擦除。因此，应用程序将轮询 ER 位以确定擦除周期是否结束。擦操作结束后，EREN 位将会被硬件清零。执行完一个页擦操作后，EEPROM 被擦除页的内容将全为零。

EEPROM 写操作

此单片机有两种模式可实现写数据到 EEPROM，即字节写模式和页写模式，可通过 EEC 寄存器中的 EEPROM 操作模式选择位 MODE 选择。

字节写模式

当模式选择位 MODE 为 0 时，可执行 EEPROM 字节写操作。字节写操作需先将 EEPROM 目标地址放入 EEAH 和 EEAL 寄存器中，再将要写入的数据放入 EED 寄存器。EEC 寄存器中的写使能位 WREN 先置高以使能写功能，然后 EEC 寄存器中的 WR 位需立即置高以开始写操作。这两条指令必须在两个指令周期内连续执行才可成功启动一个写操作。进行写操作之前应先将总中断使能位 EMI 清零，在一个有效的写启动步骤完成之后再将其使能。若 WR 位已置为高而 WREN 位还未被设置则不能开始写操作。

注：上述步骤必须依序操作，方能成功完成字节写操作，具体请参对应的考范例程序。

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 中断以侦测写周期是否完成。若写周期完成，WR 位将自动清零，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。写操作结束后，WREN 位将会被硬件清零。注意，字节写操作被成功启动前会自动执行字节擦除操作。

页写模式

在执行页写操作之前，务必确保已成功执行了相关的页擦除操作。当模式选择位 MODE 为 1 时，可执行 EEPROM 页写操作。EEPROM 一页可写入 16 个字节。上电复位后内部页缓存器将由硬件清零。当 EEPROM 写使能控制位 WREN 由 1 变为 0 时，内部页缓存器也会被清零。注意当 WREN 位由 0 变为 1 时，内部页缓存器不会清零。除了最多可以写入 16 字节 EEPROM 数据以外，页写操作启动的方法与字节写操作相同。EEPROM 地址高 7 位用来指定要写入页的位置，而低 4 位用来指向实际的地址。在页写操作模式每写入一字节数据到 EED 寄存器，低 4 位地址将自动加一，而高 7 位地址不会自动增加。当 EEPROM 地址低 4 位自动递增到当前页的最大地址，即 0FH，EEPROM 地址低 4 位的值会停止在 0FH，EEPROM 地址将不会再增加。此时再对 EED 寄存器写入数据也将无效。

页写操作需先将 EEPROM 目标页的起始地址放入 EEAH 和 EEAL 寄存器中，再将要写入的数据放入 EED 寄存器。一页的最大数据长度为 16 字节。注意当写入一字节数据到 EED 寄存器，EED 中的数据会加载到内部页缓存器中，然后当前地址值会自动加一。当一页数据被全部写入页缓存器，EEC 寄存器中的写使能位 WREN 先置高以使能写功能，然后 EEC 寄存器中的 WR 位需立即置

高以开始写操作。这两条指令必须在两个指令周期内连续执行才可成功启动一个写操作。进行写操作之前应先将总中断使能位 EMI 清零，在一个有效的写启动步骤完成之后再将其使能。若 WR 位已置为高而 WREN 位还未被设置则不能开始写操作。若 WR 位已置为高而 WREN 位还未被设置则不能开始写操作。

注：上述步骤必须依序操作，方能成功完成页写操作，具体请参考对应的范例程序。

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 中断以侦测写周期是否完成。若写周期完成，WR 位将自动清零，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。写操作结束后，WREN 位将会被硬件清零。

写保护

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

EEPROM 中断

EEPROM 擦 / 写周期结束后将产生 EEPROM 擦 / 写中断，需先通过设置相关中断寄存器的 DEE 位使能 EEPROM 中断。EEPROM 中断属于多功能中断。当 EEPROM 擦 / 写周期结束，DEF 请求标志位将被置位。若总中断、EEPROM 中断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，只有多功能中断可自动复位，而 EEPROM 中断标志位需通过应用程序手动复位。详情请参考中断章节。

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 MP1H 或 MP2H 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。EREN 位置位后，EEC 寄存器中的 ER 位需立即置位，以确保擦周期正确地执行。写或擦周期开始前总中断位 EMI 应先清零，在一个有效的写或擦启动步骤完成之后再将此位重新使能。注意，单片机不应在 EEPROM 读、擦或写操作完全完成之前进入空闲或休眠模式，否则 EEPROM 读、擦或写操作将失败。

程序举例

从 EEPROM 中读取一个字节数据 – 轮询法

```
MOV A, 040H                ; setup memory pointer low byte MP1L
MOV MP1L, A                ; MP1 points to EEC register
MOV A, 01H                 ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4                 ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H      ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L      ; user defined low byte address
```

```
MOV EEAL, A
SET IAR1.1                ; set RDEN bit, enable read operations
SET IAR1.0                ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0                ; check for read cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read function
CLR MP1H
MOV A, EED                ; move read data to register
MOV READ_DATA, A
```

从 EEPROM 中读取一页数据 – 轮询法

```
MOV A, 040H               ; setup memory pointer low byte MP1L
MOV MP1L, A               ; MP1 points to EEC register
MOV A, 01H                ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4                ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H     ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L     ; user defined low byte address
MOV EEAL, A
SET IAR1.1                ; set RDEN bit, enable read operations
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL READ
CALL READ
:
:
JMP PAGE_READ_FINISH
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
READ:
SET IAR1.0                ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0                ; check for read cycle end
JMP BACK
MOV A, EED                ; move read data to register
MOV READ_DATA, A
RET
:
PAGE_READ_FINISH:
CLR IAR1                  ; disable EEPROM read function
CLR MP1H
```

擦除 EEPROM 的一页数据 – 轮询法

```
MOV A, 040H               ; setup memory pointer low byte MP1L
MOV MP1L, A               ; MP1 points to EEC register
MOV A, 01H                ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4                ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H     ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L     ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
```

```
JMP Erase_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA         ; user defined data, erase mode don't care data
value
MOV EED, A
RET
:
Erase_START:
CLR EMI
SET IAR1.6                 ; set EREN bit, enable erase operations
SET IAR1.5                 ; start Erase Cycle – set ER bit – executed
                           ; immediately after setting EREN bit
SET EMI
BACK:
SZ  IAR1.5                 ; check for erase cycle end
JMP BACK
CLR MP1H
```

写入一个字节数据到 EEPROM – 轮询法

```
MOV A, 040H                ; setup memory pointer low byte MP1L
MOV MP1L, A                ; MP1 points to EEC register
MOV A, 01H                 ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4                 ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H      ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L      ; user defined low byte address
MOV EEAL, A
MOV A, EEPROM_DATA         ; user defined data
MOV EED, A
CLR EMI
SET IAR1.3                 ; set WREN bit, enable write operations
SET IAR1.2                 ; start Write Cycle – set WR bit – executed
                           ; immediately after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2                 ; check for write cycle end
JMP BACK
CLR MP1H
```

写入一页数据到 EEPROM – 轮询法

```
MOV A, 040H                ; setup memory pointer low byte MP1L
MOV MP1L, A                ; MP1 points to EEC register
MOV A, 01H                 ; setup memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4                 ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H      ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L      ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP WRITE_START
```

```
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA          ; user defined data
MOV EED, A
RET
:
WRITE_START:
CLR EMI
SET IAR1.3                  ; set WREN bit, enable write operations
SET IAR1.2                  ; start Write Cycle - set WR bit - executed
                            ; immediately after setting WREN bit
SET EMI
BACK:
SZ  IAR1.2                  ; check for write cycle end
JMP BACK
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。振荡器的选择和操作是通过应用程序控制寄存器实现的。

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中断的时钟源。外部振荡器需要一些外围器件，而集成的内部振荡器不需要任何外围器件。它们提供的高速和低速系统振荡器具有较宽的频率范围。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。

对于 USB 应用，如果选择 HXT 振荡器，HXT 引脚必须连接一个 6MHz 或 12MHz 的晶振。

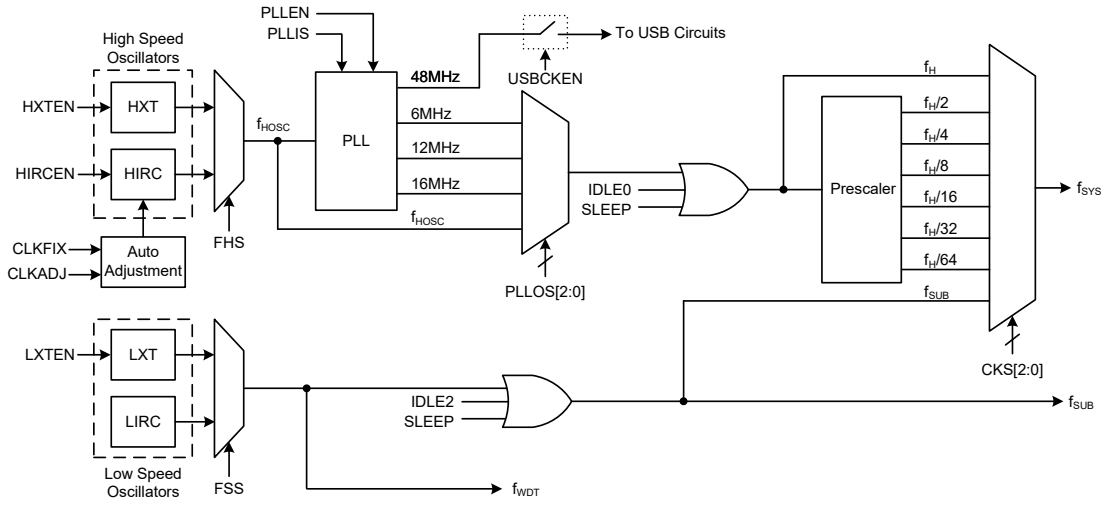
类型	名称	频率	引脚
外部高速晶振	HXT	6MHz 或 12MHz	OSC1/OSC2
内部高速 RC	HIRC	12MHz	—
外部低速晶振	LXT	32.768kHz	XT1/XT2
内部低速 RC	LIRC	32kHz	—

振荡器类型

系统时钟配置

该单片机有多个系统振荡器，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 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 晶振 LXT。使用高速或低速振荡器作为系统时钟的选择是通过设置 SCC 寄存器中的 CKS2~CKS0 位决定的，系统时钟可动态选择。

低速振荡器的实际时钟源由 SCC 寄存器的 FSS 位选择，高速振荡器的实际时钟源由 SCC 寄存器的 FHS 位选择。低速或高速系统时钟频率由 SCC 寄存器的 CKS2~CKS0 位决定的。此外，内部 PLL 频率发生器时钟源可通过软件控制位产生不同的频率提供给 USB 接口和作为系统时钟使用。



系统时钟配置

内部 PLL 频率发生器

内部 PLL 频率发生器为 USB 接口和系统时钟提供各种频率。PLL 发生器可由 PLLC 寄存器的 PLLEN 位控制使能和除能。上电复位后，PLL 控制位将被设为“1”以开启 PLL 发生器。PLL 发生器为 USB 提供了一个固定 48MHz 的工作频率，为系统时钟提供的频率有 6MHz、12MHz 或 16MHz。系统频率可由 PLLC 寄存器的 PLLEN 和 PLLOS2~PLLOS0 位选择。

下表说明了高频系统时钟 f_H 是由几个相关控制位共同选择的。

PLLEN	PLLOS2	PLLOS1	PLLOS0	f_H
0	x	x	x	f_{HOSC} – HXT 或 HIRC，取决于 SCC 寄存器中的 FHS 位。
1	0	x	x	f_{HOSC} – HXT 或 HIRC，取决于 SCC 寄存器中的 FHS 位。
1	1	0	0	f_{PLL}=6MHz
1	1	0	1	f_{PLL}=12MHz
1	1	1	0	f_{PLL}=16MHz
1	1	1	1	保留

“x”：无关

- **PLLC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	—	PLLIS	PLLOS2	PLLOS1	PLLOS0	PLLF	PLLEN
R/W	R/W	—	R/W	R/W	R/W	R/W	R	R/W
POR	0	—	0	0	0	0	0	1

Bit 7　**D7**：保留位，不可用且固定为“0”

Bit 6　未定义，读为“0”

Bit 5　**PLLIS**：PLL 输入时钟源选择

0：12MHz

1：6MHz

若 FHS=1，当使用 12MHz 晶振 / 谐振器时，HXT 频率进行 2 分频而后通过内部 PLL 电路进行 8 倍频；当使用 6MHz 晶振 / 谐振器时，HXT 直接通过 PLL 电路

进行 8 倍频。
若 FHS=0，选中 12MHz HIRC，此位将由硬件自动清零。

Bit 4~2 **PLLOS2~PLLOS0**：f_H 时钟源选择 (f_{HOSC} 或 f_{PLL})
0xx：f_{HOSC}
100：f_{PLL}=6MHz
101：f_{PLL}=12MHz
110：f_{PLL}=16MHz
111：保留

Bit 1 **PLLF**：PLL 时钟稳定标志位
0：不稳定
1：稳定
此位用于表明 PLL 时钟是否稳定。PLLEN 位置高使能 PLL 时钟后，PLLF 位会先被清零，在 PLL 时钟稳定后会被置高。

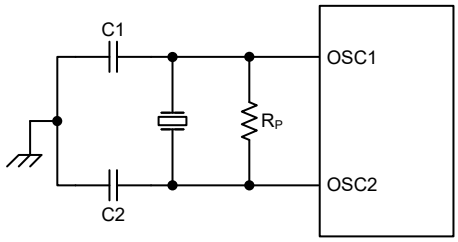
Bit 0 **PLLEN**：PLL 使能控制
0：除能
1：使能

外部晶体 / 陶瓷振荡器 – HXT

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

对于 USB 应用，如果选择 HXT 振荡器，HXT 引脚必须连接一个 6MHz 或 12MHz 的晶振。

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



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

晶体 / 陶瓷振荡器

晶体振荡器 C1 和 C2 值		
晶体频率	**C1**	**C2**
16MHz	0pF	0pF
12MHz	0pF	0pF
8MHz	0pF	0pF
6MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	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 RC 振荡器 – HIRC

内部 RC 振荡器是一个集成的系统振荡器，由 FHS 控制位选择，不需其它外部器件。内部 RC 振荡器频率固定为 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 V_{DD}、温度以及芯片制成工艺不同的影响较大程度地降低。

外部 32.768kHz 晶体振荡器 – LXT

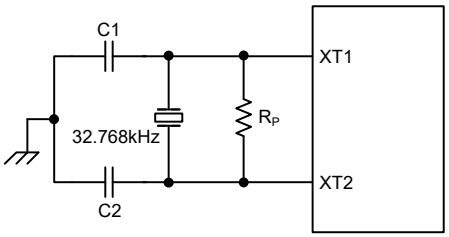
外部 32.768kHz 晶体振荡器是一个低频振荡器，由 FSS 控制位选择。时钟频率固定为 32.768kHz，此时 XT1 和 XT2 间引脚必须连接 32.768kHz 的晶体振荡器。需要外部电阻和电容连接到 32.768kHz 晶振以帮助起振。对于那些要求精准频率的场合中，可能需要这些元件来对由制程产生的误差提供频率补偿。LXTEN 位置高使能 LXT 振荡器后，LXT 振荡器启动需要一定的延时。

然而，对于一些晶体，为了保证系统频率的启动与精准度要求，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。

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

外部 LXT 振荡器

LXT 振荡器 C1 和 C2 值		
晶体频率	C1	C2
32.768kHz	10pF	10pF
注：1. C1 和 C2 数值仅作参考用 2. R_P 的建议值为 5MΩ~10MΩ		

32.768kHz 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 LXTC 寄存器中的 LXTSP 位进行模式选择。

LXTSP 位	LXT 工作模式
0	低功耗
1	快速启动

LXTSP 位置高会使能 LXT 快速启动模式。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，可以通过将 LXTSP 位清零进入低功耗模式。振荡器可以继续运行，其间耗电将少于快速启动模式。需要注意的是，在选择 LXT 振荡器时钟作为系统时钟源之前，必须适当地控制 LXT 工作模式的切换。一旦通过设置 SCC 寄存器中的 CKS2~CKS0 位和 FSS 位选择了 LXT 振荡器时钟作为系统时钟源，LXT 振荡器工作模式将不能改变。

应注意的是，无论 LXTSP 位是什么值，LXT 振荡器会一直运作，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 系统振荡器是一个低频振荡器，由 FSS 控制位选择。它是一个完全集成 RC 振荡器，典型频率值为 32kHz 且无需外部元件。芯片在制造时进行调整且内部含有频率补偿电路，使得振荡器因电源电压、温度及芯片制成工艺不同的影响较大程度地降低。

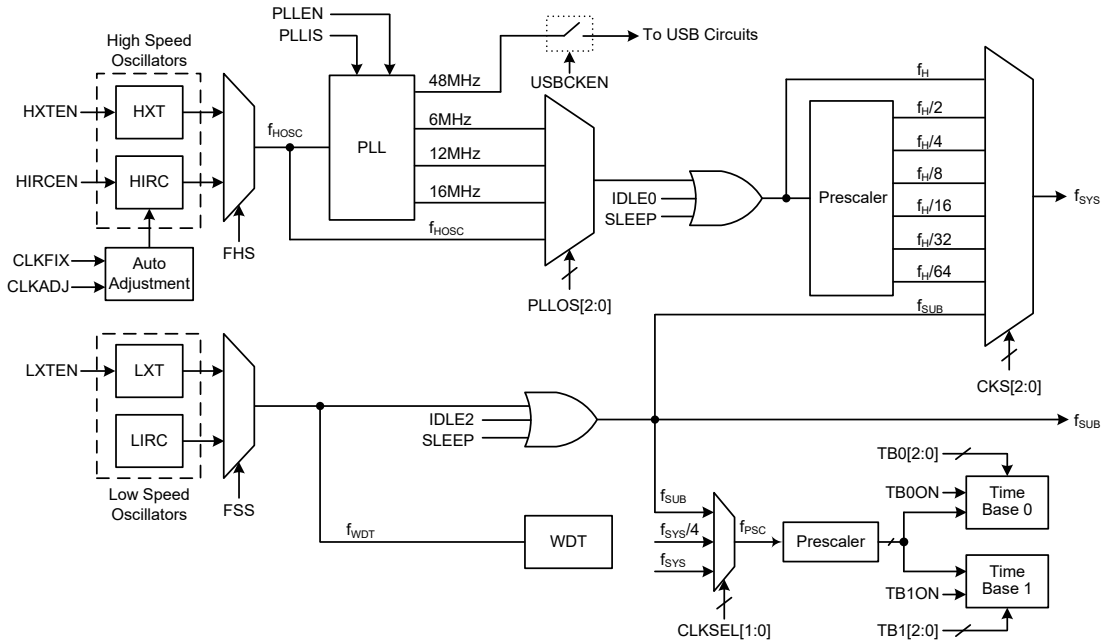
工作模式和系统时钟

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

系统时钟

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

主系统时钟可来自高频时钟源 f_H 或低频时钟源 f_{SUB}，通过 SCC 寄存器中的 CKS2~CKS0 位进行选择。高频时钟来自 HXT 振荡器、HIRC 振荡器或 PLL 输出时钟，可通过 SCC 寄存器中的 FHS 位以及 PLLC 寄存器中的 PLLEN 和 PLLOS2~PLLOS0 位选择。低频系统时钟源来自 f_{SUB}，若 f_{SUB} 被选择，低频时钟来自 LXT 或 LIRC 振荡器，可通过 SCC 寄存器中的 FSS 位选择。其它系统时钟还有高速系统振荡器的分频 $f_H/2$~$f_H/64$。



单片机时钟配置

注：当系统时钟源 f_{SYS} 由 f_H 切换为 f_{SUB} 时，可以通过设置相应的高速振荡器使能控制位，选择停止以节省耗电，或者继续振荡，为外围电路提供 f_H~f_H/64 频率的时钟源。

系统工作模式

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

工作模式	CPU	寄存器设置			f_{SYS}	f_H	f_{SUB}	f_{WDT}	f_{PLL}
		FHIDEN	FSIDEN	CKS2~CKS0					
快速模式	On	x	x	000~110	f_H~f_H/64	On	On	On	On
低速模式	On	x	x	111	f_{SUB}	On/Off(1)	On	On	On/Off(3)
空闲模式 0	Off	0	1	000~110	Off	Off	On	On	Off
				111	On				
空闲模式 1	Off	1	1	xxx	On	On	On	On	On
空闲模式 2	Off	1	0	000~110	On	On	Off	On/Off(2)	On
				111	Off				
休眠模式	Off	0	0	xxx	Off	Off	Off	On/Off(2)	Off

“x”：无关

注：1. 在低速模式中，f_H 时钟的开启或关闭由相应的振荡器使能位控制。
2. 在空闲模式 2 和休眠模式中，f_{WDT} 时钟的开启或关闭由 WDT 功能的使能或除能控制。
3. 在低速模式中，f_{PLL} 时钟的开启 / 关闭由 PLLC 寄存器中的 PLLEN 位控制。
4. 在空闲模式 0 和休眠模式中，USB 功能无效。

快速模式

快速模式的系统时钟由一个高速振荡器提供，单片机的所有功能均可在此模式中实现。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 SCC 寄存器中的 CKS2~CKS0 位选择。单片机使用高速振荡器分频作为系统时钟可减少工作电流。在 USB 模式中，PLL 电路的时钟源可来源于 HXT 或 HIRC 振荡器，该电路可产生 6MHz、12MHz 或 16MHz 作为单片机系统时钟。

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 f_{SUB} 来自 LIRC 或 LXT 振荡器，由 SCC 寄存器中的 FSS 位决定。

休眠模式

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为低时，系统进入休眠模式。在休眠模式中，CPU 停止，提供外围功能时钟源的 f_{SUB} 也会停止。然而如果 WDT 功能使能，f_{WDT} 可继续运行。

空闲模式 0

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 位为低、FSIDEN 位为高时，系统进入空闲模式 0。在空闲模式 0 中，CPU 停止，但低速振荡器会开启以驱动一些外围功能。

空闲模式 1

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 和 FSIDEN 位都为高时，系统进入空闲模式 1。在空闲模式 1 中，CPU 停止，但高速和低速振荡器都会开启以保持一些外围功能继续工作。

空闲模式 2

执行 HALT 指令后且 SCC 寄存器中的 FHIDEN 位为高、FSIDEN 位为低时，系统进入空闲模式 2。在空闲模式 2 中，CPU 停止，但高速振荡器会开启以保持一些外围功能继续工作。

控制寄存器

寄存器 SCC、HIRCC、HXTC、LXTC 和 PLLC 用于控制系统时钟和进行相应的振荡器配置。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	FHS	FSS	FHIDEN	FSIDEN
HIRCC	CLKADJ	CLKADJF	CLKFIX	—	—	—	HIRCF	HIRCEN
HXTC	—	—	—	—	—	HXTM	HXTF	HXTEN
LXTC	—	—	—	—	—	LXTSP	LXTF	LXTEN
PLLC	D7	—	PLLIS	PLLOS2	PLLOS1	PLLOS0	PLLF	PLLEN

系统工作模式控制寄存器列表

• SCC 寄存器

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

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

Bit 4 未定义，读为“0”

Bit 3 **FHS**：高频时钟 f_{HOSC} 选择位
0：HIRC
1：HXT

Bit 2 **FSS**：低频时钟选择位
0：LIRC
1：LXT

Bit 1 **FHIDEN**：CPU 关闭时高频振荡器控制位
0：除能
1：使能
此位用来控制在 CPU 执行 HALT 指令关闭后高速振荡器是激活还是停止。

Bit 0 **FSIDEN**：CPU 关闭时低频振荡器控制位
0：除能
1：使能
此位用来控制在 CPU 执行 HALT 指令关闭后低速振荡器是激活还是停止。

注：使用 CKS2~CKS0、FHS 或 FSS 位进行时钟切换设置之后，在相关时钟成功切换至目标时钟源之前需要一定的延时。因此，若接下来执行的操作需要目标时钟源立即响应，则在此之前必须规划适当的延迟时间。
时钟切换延迟时间 = $4\times t_{SYS}+[0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$，其中 $t_{Curr.}$ 指代当前的时钟周期，$t_{Tar.}$ 指代目标时钟周期，t_{SYS} 指代当前系统时钟周期。

• HIRCC 寄存器

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

Bit 7 **CLKADJ**：USB 模式中 HIRC 时钟自动调整功能控制
0：除能
1：使能
应注意若用户选择 HIRC 作为系统时钟，则 CLKADJ 位必须置“1”以自动调整 PLL 频率。

Bit 6 **CLKADJF**：HIRC 时钟自动调整稳定标志位
0：不稳定
1：稳定
此位用于表明 CLKADJ 位置高后 HIRC 频率调整操作是否完成。用户可通过应用程序控制此位持续监测 CLKADJF，确保 HIRC 频率精准度稳定地调整在

±0.25% 之内。此位可由应用程序清零。

Bit 5 **CLKFIX**：HIRC 时钟自动固定调整功能控制
0：除能
1：使能
注意当 CLKADJF=1，此位置高以将 HIRC 频率精准度固定在 ±0.25% 范围内。

Bit 4~2 未定义，读为“0”

Bit 1 **HIRCF**：HIRC 振荡器稳定标志位
0：HIRC 不稳定
1：HIRC 稳定
此位用于表明 HIRC 振荡器是否稳定。HIRCEN 位置高使能 HIRC 振荡器后，HIRCF 位会先被清零，在 HIRC 稳定后会被置高。

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

● HXTC 寄存器

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

Bit 7~3 未定义，读为“0”

Bit 2 **HXTM**：HXT 模式选择位
0：HXT 频率 ≤ 10MHz (灌电流 / 源电流较小)
1：HXT 频率 > 10MHz (灌电流 / 源电流较大)
注意，此位须根据所使用的 HXT 频率正确设置。若 HXTM=0 而 HXT 频率大于 10MHz，则低压时振荡性能可能不佳。若 HXTM=1 而 HXT 频率小于 10MHz，则振荡频率和电流可能异常。
此位必须在 HXT 使能前正确地配置。当 OSC1 和 OSC2 引脚功能已通过相关引脚共用控制位使能，且 HXTEN 位已置高使能 HXT 振荡器，此时再改变 HXTM 设定值是无效的。若 OSC1 或 OSC2 引脚功能除能，此时无论 HXTEN 位为何值，可通过软件改写 HXTM 位值。

Bit 1 **HXTF**：HXT 振荡器稳定标志位
0：HXT 不稳定
1：HXT 稳定
此位用于表明 HXT 振荡器是否稳定。HXTEN 位置高使能 HXT 振荡器后，HXTF 位会先被清零，在 HXT 稳定后会被置高。

Bit 0 **HXTEN**：HXT 振荡器使能控制位
0：除能
1：使能

● LXTC 寄存器

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

Bit 7~3 未定义，读为“0”

Bit 2 **LXTSP**：LXT 振荡器快速启动控制位
0：除能 – 低功耗模式
1：使能 – 快速启动模式
此位用来控制 LXT 振荡器工作在低功耗模式或快速启动模式。当 LXTSP 位被置高，LXT 振荡器的振荡加快，但功耗增加。如果 LXTSP 位被清零，LXT 振荡器功耗将减少，但需要较长时间才能稳定下来。需要注意的是，通过设置 SCC

寄存器中的 CKS2~CKS0 位和 FSS 位选择 LXT 振荡器作为系统时钟源后，该位不能改变。

Bit 1 **LXTF**：LXT 振荡器稳定标志位
0：LXT 不稳定
1：LXT 稳定
此位用于表明 LXT 振荡器是否稳定。LXTEN 位置高使能 LXT 振荡器后，LXTF 位会先被清零，在 LXT 稳定后会被置高。

Bit 0 **LXTEN**：LXT 振荡器使能控制位
0：除能
1：使能

● **PLLC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	D7	—	PLLIS	PLLOS2	PLLOS1	PLLOS0	PLLF	PLLEN
R/W	R/W	—	R/W	R/W	R/W	R/W	R	R/W
POR	0	—	0	0	0	0	0	1

Bit 7 **D7**：保留位，不可用且固定为“0”

Bit 6 未定义，读为“0”

Bit 5 **PLLIS**：PLL 输入时钟源选择
0：12MHz
1：6MHz
若 FHS=1，当使用 12MHz 晶振 / 谐振器时，HXT 频率进行 2 分频而后通过内部 PLL 电路进行 8 倍频；当使用 6MHz 晶振 / 谐振器时，HXT 直接通过 PLL 电路进行 8 倍频。
若 FHS=0，选中 12MHz HIRC，此位将由硬件自动清零。

Bit 4~2 **PLLOS2~PLLOS0**：f_H 时钟源选择 (f_{HOSC} 或 f_{PLL})
0xx：f_{HOSC}
100：f_{PLL}=6MHz
101：f_{PLL}=12MHz
110：f_{PLL}=16MHz
111：保留

Bit 1 **PLLF**：PLL 时钟稳定标志位
0：不稳定
1：稳定
此位用于表明 PLL 时钟是否稳定。PLLEN 位置高使能 PLL 时钟后，PLLF 位会先被清零，在 PLL 时钟稳定后会被置高。

Bit 0 **PLLEN**：PLL 使能控制
0：除能
1：使能

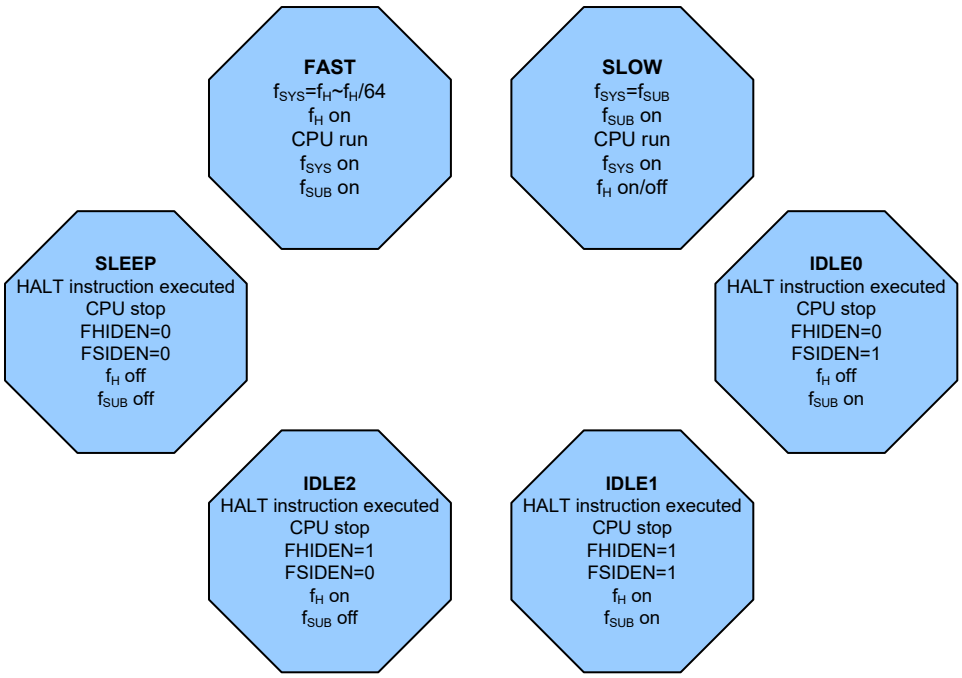
注：使用 PLLOS2~ PLLOS0 位进行时钟切换设置之后，在相关时钟成功切换至目标时钟源之前需要一定的延时。因此，若接下来执行的操作需要目标时钟源立即响应，则在此之前必须规划适当的延迟时间。
时钟切换延迟时间 = $4\times t_{SYS}+[0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$，其中 $t_{Curr.}$ 指代当前的时钟周期，$t_{Tar.}$ 指代目标时钟周期，t_{SYS} 指代当前系统时钟周期。

工作模式切换

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

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

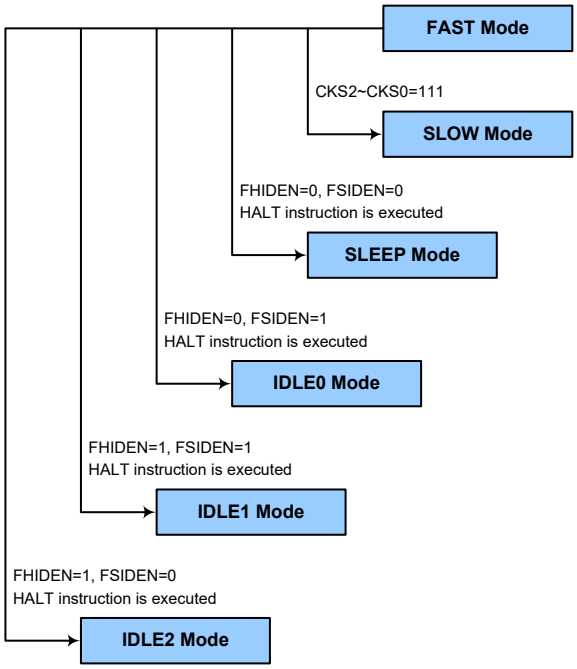
寄存器中的 FHIDEN 和 FSIDEN 位决定的。



快速模式切换到低速模式

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

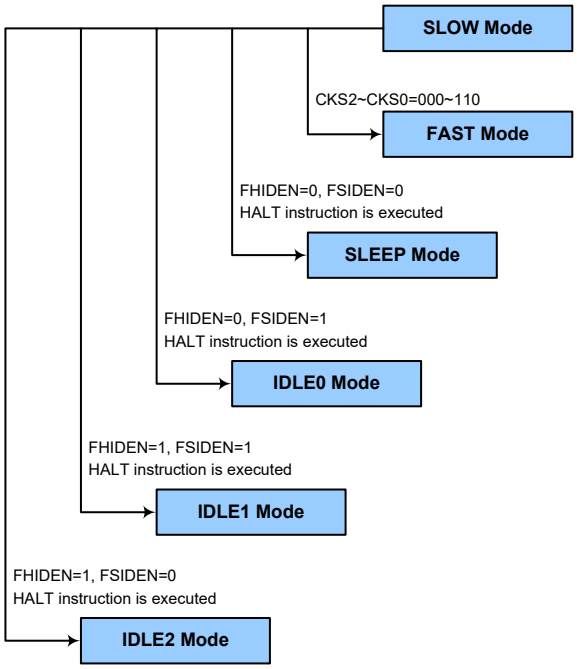
低速模式的时钟源来自 LXT 或 LIRC 振荡器，由 SCC 寄存器中的 FSS 位确定，因此要求这些振荡器在所有模式切换动作发生前稳定下来。



低速模式切换到快速模式

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

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



进入休眠模式

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

- 系统时钟停止运行，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。
- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 如果 WDT 功能使能，WDT 将被清零并重新开始计数。如果 WDT 功能除能，WDT 将被清零并停止计数。
- 如果 USB 功能使能，USB 将进入暂停模式。

进入空闲模式 0

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

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

进入空闲模式 1

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

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

进入空闲模式 2

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

- f_H 时钟开启，f_{SUB} 时钟关闭，应用程序停止在“HALT”指令处。
- 数据存储器中的内容和寄存器将保持当前值。
- 输入 / 输出口将保持当前值。

- 状态寄存器中暂停标志 PDF 将被置起，看门狗溢出标志 TO 将被清除。
- 如果 WDT 功能使能，WDT 将被清零并重新开始计数。如果 WDT 功能除能，WDT 将被清零并停止计数。

待机电流注意事项

由于单片机进入休眠或空闲模式的主要原因是将单片机的电流功耗降低到尽可能低，可能到只有几个微安的级别 (空闲模式 1 和空闲模式 2 除外)，所以如果要将电路的电流进一步降低，电路设计者还应有其它的考虑。应该特别注意的是单片机的输入 / 输出引脚。所有高阻抗输入脚都必须连接到固定的高或低电平，因为引脚浮空会造成内部振荡并导致耗电增加。这也应用于有不同封装的单片机，因为它们可能含有未引出的引脚，这些引脚也必须设为输出或带有上拉电阻的输入。

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

在空闲模式 1 和空闲模式 2 中，高速振荡器开启。若外围功能时钟源来自高速振荡器，额外的静态电流也可能会有几百微安。

唤醒

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

系统进入休眠或空闲模式之后，可以通过以下几种方式唤醒：

- USB 复位信号复位
- PA 口下降沿
- 系统中断
- WDT 溢出

若由 USB 复位唤醒，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。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时钟 f_{WDT}，而 f_{WDT} 的时钟源由 LXT 或 LIRC 时钟提供，通过配置 SCC 寄存器中的 FSS 位选择。LIRC 内部振荡器频率约为 32kHz 且这个指定的内部时钟周期随 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 寄存器用于选择溢出周期、控制 WDT 功能的使能 / 除能以及软件复位单片机。

- **WDTC 寄存器**

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

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

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

- **RSTFC 寄存器**

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

“x”：未知

Bit 7~4 未定义，读为“0”

Bit 3 **RSTF**：RSTC 寄存器软件复位标志位
详见内部复位控制章节。

Bit 2 **LVRF**：LVR 复位标志位
详见低电压复位章节。

Bit 1 **LRF**：LVR 控制寄存器软件复位标志位
详见低电压复位章节。

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

看门狗定时器操作

当 WDT 溢出时，它产生一个单片机复位的动作。这也就意味着正常工作期间，用户需在应用程序中看门狗溢出前有策略地清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，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 中的 WE4~WE0 位可提供 WDT 使能 / 除能控制以及单片机软件复位操作。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 t_{SRESET} 延迟时间后复位。上电后这些位初始化为“01010B”。

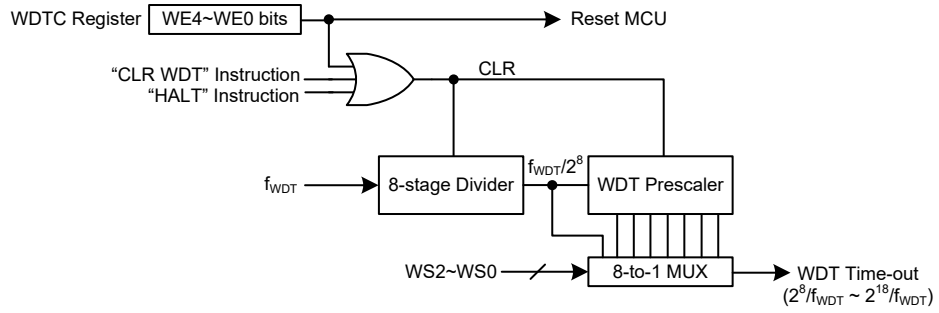
WE4~WE0 位	WDT 功能
10101B	除能
01010B	使能
其它值	单片机复位

看门狗定时器功能控制

程序正常运行时，WDT 溢出将导致单片机复位，并置位状态标志位 TO。若系统处于休眠或空闲模式，当 WDT 发生溢出时，状态寄存器中的 TO 应置位，仅 PC 和堆栈指针复位。有三种方法可以用来清除 WDT 的内容。第一种是 WDT 软件复位，即将 WE4~WE0 位设置成除了 01010B 和 10101B 外的任意值；第二种是通过软件清除指令；而第三种是通过“HALT”指令。

该单片机只使用一条清看门狗指令“CLR WDT”。因此只要执行“CLR WDT”便清除 WDT。

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



看门狗定时器

复位和初始化

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

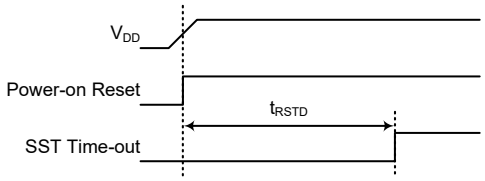
除了上电复位外，另一种复位为低电压复位即 LVR 复位，在电源供应电压低于 LVR 设定值时，系统会产生 LVR 复位。还有一种复位为看门狗溢出单片机复位，不同方式的复位操作会对寄存器产生不同的影响。

复位功能

单片机包含下面几种由内部事件触发的复位方式。

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

内部复位控制

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

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

内部复位功能控制

- **RSTC 寄存器**

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

Bit 7~0 **RSTC7~RSTC0**：复位功能控制位
01010101：无操作
10101010：无操作
其它值：MCU 复位

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

● **RSTFC 寄存器**

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

“x”：未知

Bit 7~4　未定义，读为“0”

Bit 3　**RSTF**：复位控制寄存器软件复位标志位
0：未发生
1：发生
当 RSTC 控制寄存器软件复位发生时，此位被置为“1”，且只能通过应用程序清零。

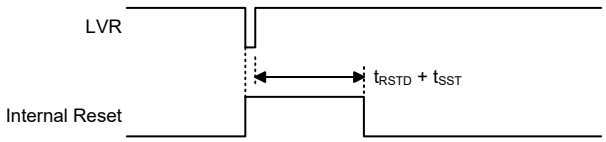
Bit 2　**LVRF**：LVR 复位标志位
详见低电压复位章节。

Bit 1　**LRF**：LVR 控制寄存器软件复位标志位
详见低电压复位章节。

Bit 0　**WRF**：WDT 控制寄存器软件复位标志位
详见看门狗定时器控制寄存器章节。

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。当电源电压低于某一预定值时，它将复位单片机。LVR 在快速和低速模式下始终使能，并会设定一个电源复位低电压 V_{LVR}。例如在更换电池的情况下，单片机供应的电压可能会在 0.9V~V_{LVR} 之间，这时 LVR 将会自动复位单片机且 RSTFC 寄存器中的 LVRF 标志位置位。LVR 包含以下的规格：有效的 LVR 信号，即在 0.9V~V_{LVR} 的低电压状态的时间，必须超过 LVR 电气特性中 t_{LVR} 参数的值。如果低电压存在时间不超过 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 t_{LVR} 值可通过 TLVRC 寄存器中的 TLVR1~TLVR0 位设置。实际的 V_{LVR} 参数值可通过 LVRC 寄存器中的 LVS7~LVS0 位设置。若由于受到干扰 LVS7~LVS0 变为其它值时，需经过一段 t_{SRESET} 延迟时间才会响应复位。此时 RSTFC 寄存器的 LRF 位被置位。注意当单片机进入空闲或休眠模式，LVR 功能将自动关闭。



低电压复位时序图

● **LVRC 寄存器**

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

Bit 7~0　**LVS7~LVS0**：LVR 电压选择
01010101：2.1V
00110011：2.55V
10011001：3.15V

10101010：3.8V
其它值：单片机复位 – 寄存器复位为 POR 值

若有以上定义的四个低电压复位值的低电压情况发生，且检测到此低电压的保持时间大于 t_{LVR}，则单片机复位发生。实际的 t_{LVR} 值可通过 TLVRC 寄存器中的 TLVR1~TLVR0 位设置。此种复位后的寄存器内容保持不变。

除了以上定义的四个低电压复位值外，其它值也能导致单片机复位。需要经过一段 t_{SRESET} 延迟时间来响应复位。但此时寄存器内容将复位为 POR 值。

- **TLVRC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	TLVR1	TLVR0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	1

Bit 7~2　未定义，读为“0”

Bit 1~0　**TLVR1~TLVR0**：产生 LVR 复位的低电压最短保持时间 (t_{LVR}) 选择
00：(7~8)×t_{LIRC}
01：(31~32)×t_{LIRC}
10：(63~64)×t_{LIRC}
11：(127~128)×t_{LIRC}

- **RSTFC 寄存器**

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

“x”：未知

Bit 7~4　未定义，读为“0”

Bit 3　**RSTF**：复位控制寄存器软件复位标志位
详见内部复位控制章节。

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

Bit 1　**LRF**：LVRC 寄存器软件复位标志位
0：未发生
1：发生
如果 LVRC 寄存器包含任何非定义的 LVR 电压设置值，此位被置为“1”，这类似于软件复位功能。此位只能通过应用程序清零。

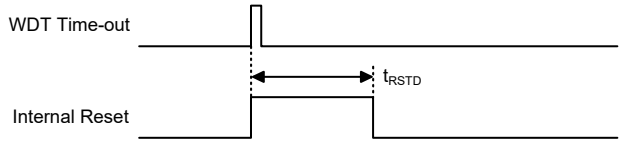
Bit 0　**WRF**：WDTC 控制寄存器软件复位标志位
详见看门狗定时器控制寄存器章节。

USB 复位

该单片机包含 USB 电路，可通过正确配置 USB 控制寄存器产生复位信号复位整个单片机。更多相关细节参考“USB 接口”章节。

正常运行时看门狗溢出复位

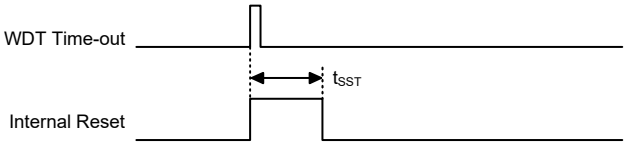
除了看门狗溢出标志位 TO 将被设为“1”之外，在快速或低速模式下正常运行时看门狗溢出复位和 LVR 硬件复位相同。



正常运行时看门狗溢出复位时序图

休眠或空闲时看门狗溢出复位

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



休眠或空闲时看门狗溢出复位时序图

复位初始状态

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

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

“u”：不改变

在单片机上电复位之后，各功能单元初始化的情形，列于下表。

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

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

寄存器	上电复位	LVR 复位 (正常操作)	WDT 溢出 (正常操作)	WDT 溢出 (空闲 / 休眠)
IAR0	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	0000 0000	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	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu
STATUS	xx00 xxxx	uuuu uuuu	uu1u uuuu	uu11 uuuu
PBP	---- --00	---- --00	---- --00	---- --uu
IAR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- 0x00	---- u1uu	---- uuuu	---- uuuu
SCC	010- 0000	010- 0000	010- 0000	uuu- uuuu
HIRCC	000- --01	uuu- --01	uuu- --01	uuu- --uu
HXTC	---- -000	---- -000	---- -000	---- -uuu
LXTC	---- -000	---- -000	---- -000	---- -uuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTC	0101 0101	0101 0101	0101 0101	uuuu uuuu
LVRC	0101 0101	uuuu uuuu	0101 0101	uuuu uuuu
TLVRC	---- --01	---- --01	---- --01	---- --uu
MFI0	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI1	-000 -000	-000 -000	-000 -000	-uuu -uuu
MFI2	-000 -000	-000 -000	-000 -000	-uuu -uuu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
INTEG	0000 0000	0000 0000	0000 0000	uuuu uuuu
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	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	-111 1111	-111 1111	-111 1111	-uuu uuuu
PCC	-111 1111	-111 1111	-111 1111	-uuu uuuu
PCPU	-000 0000	-000 0000	-000 0000	-uuu uuuu

寄存器	上电复位	LVR 复位 (正常操作)	WDT 溢出 (正常操作)	WDT 溢出 (空闲 / 休眠)
PSCR	---- --00	---- --00	---- --00	---- --uu
TB0C	0--- -000	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	0--- -000	u--- -uuu
IREFC	00-0 0-00	00-0 0-00	00-0 0-00	uu-u u-uu
PVREF	0000 0000	0000 0000	0000 0000	uuuu uuuu
OPA1C	0000 0000	0000 0000	0000 0000	uuuu uuuu
OPA2C	0000 0000	0000 0000	0000 0000	uuuu uuuu
GSC	---- ---0	---- ---0	---- ---0	---- ---u
AFEDAC	0000 0000	0000 0000	0000 0000	uuuu uuuu
AFEDA1L	0000 ----	0000 ----	0000 ----	uuuu ----
AFEDA1H	0000 0000	0000 0000	0000 0000	uuuu uuuu
AFEDA2L	0000 ----	0000 ----	0000 ----	uuuu ----
AFEDA2H	0000 0000	0000 0000	0000 0000	uuuu uuuu
SADC0	0000 0000	0000 0000	0000 0000	uuuu uuuu
SADC1	---0 0000	---0 0000	---0 0000	---u uuuu
SADC2	---- -000	---- -000	---- -000	---- -uuu
SADOL	xxxx ----	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
SADOL				uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
SADOH				---- uuuu (ADRFS=1)
EEAL	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEAH	---- -000	---- -000	---- -000	---- -uuu
EED	0000 0000	0000 0000	0000 0000	uuuu uuuu
OPSW	---- --00	---- --00	---- --00	---- --uu
ASWAI0	---- 0000	---- 0000	---- 0000	---- uuuu
ASWAI1	---- 0000	---- 0000	---- 0000	---- uuuu
ASWAI2	---- 0000	---- 0000	---- 0000	---- uuuu
ASWAI3	---- 0000	---- 0000	---- 0000	---- uuuu
ASWAI4	---- 0000	---- 0000	---- 0000	---- uuuu
ASWAI5	---- 0000	---- 0000	---- 0000	---- uuuu
ASWAI6	---- 0000	---- 0000	---- 0000	---- uuuu
ASWAI7	---- 0000	---- 0000	---- 0000	---- uuuu
ATMC0	0000 0--0	0000 0--0	0000 0--0	uuuu u--u
ATMC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
ATMDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
ATMDH	---- --00	---- --00	---- --00	---- --uu
ATMAL	0000 0000	0000 0000	0000 0000	uuuu uuuu
ATMAH	---- --00	---- --00	---- --00	---- --uu
ATMBL	0000 0000	0000 0000	0000 0000	uuuu uuuu

寄存器	上电复位	LVR 复位 (正常操作)	WDT 溢出 (正常操作)	WDT 溢出 (空闲 / 休眠)
ATMBH	---- --00	---- --00	---- --00	---- --uu
ATMRP	0000 0000	0000 0000	0000 0000	uuuu uuuu
MDUWR0	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR1	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR2	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR3	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR4	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR5	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWCTRL	00-- ----	00-- ----	00-- ----	uu-- ----
SYSC	0001 --0x	0001 --0x	0001 --0x	uuuu --ux
USB_STAT	11xx 0001	uuxx uuuu	uuxx uuuu	uuxx uuuu
UINT	---- 0000	---- uuuu	---- uuuu	---- uuuu
USC	1000 xxxx	uuuu xuux	uuuu xuux	uuuu xuux
UESR	---- xxxx	---- uuuu	---- uuuu	---- uuuu
UCC	0-0x 0-xx	u-uu u-uu	u-uu u-uu	u-uu u-uu
AWR	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
STL	---- xxxx	---- uuuu	---- uuuu	---- uuuu
SIES	xxxx xxxx	uxxx xuuu	uxxx xuuu	uxxx xuuu
MISC	xxx0 -xxx	xxuu -uxx	xxuu -uxx	xxuu -uxx
SETIO	---- 1110	---- uuuu	---- uuuu	---- uuuu
FIFO3	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
FRNUM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
FRNUM1	---- -xxx	---- -xxx	---- -xxx	---- -uuu
PLLC	0-00 0001	0-uu uu0u	0-uu uu0u	u-uu uuuu
LCDC0	0-00 ---0	0-00 ---0	0-00 ---0	u-uu ---u
LCDC2	000- -00-	000- -00-	000- -00-	uuu- -uu-
ORMC	0000 0000	0000 0000	0000 0000	0000 0000
PD	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDPU	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
PEPU	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
PFPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PG	--11 1111	--11 1111	--11 1111	--uu uuuu
PGC	--11 1111	--11 1111	--11 1111	--uu uuuu
PGPU	--00 0000	--00 0000	--00 0000	--uu uuuu

寄存器	上电复位	LVR 复位 (正常操作)	WDT 溢出 (正常操作)	WDT 溢出 (空闲 / 休眠)
PH	---1 11--	---1 11--	---1 11--	---u uu--
PHC	---1 11--	---1 11--	---1 11--	---u uu--
PHHPU	---0 00--	---0 00--	---0 00--	---u uu--
FCR	0000 0000	0000 0000	0000 0000	uuuu uuuu
FRCR	00-0 --00	00-u --00	00-u --00	uu-u --uu
FARL	0000 0000	0000 0000	0000 0000	uuuu 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
SIM0C0	1110 0000	1110 0000	1110 0000	uuuu uuuu
SIM0C1* (U0MD=0)	1000 0001	1000 0001	1000 0001	uuuu uuuu
U0UCR1* (U0MD=1)	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
SIM0D/U0TXR_RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIM0A/SIM0C2/U0UCR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
U0UCR3	---- ---0	---- ---0	---- ---0	---- ---u
SIM0TOC* (U0MD=0)	0000 0000	0000 0000	0000 0000	uuuu uuuu
U0BRG* (U0MD=1)	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
U0USR	0000 1011	0000 1011	0000 1011	uuuu uuuu
SIM1C0	1110 0000	1110 0000	1110 0000	uuuu uuuu
SIM1C1* (U1MD=0)	1000 0001	1000 0001	1000 0001	uuuu uuuu
U1UCR1* (U1MD=1)	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
SIM1D/U1TXR_RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIM1A/SIM1C2/U1UCR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
U1UCR3	---- ---0	---- ---0	---- ---0	---- ---u
SIM1TOC* (U1MD=0)	0000 0000	0000 0000	0000 0000	uuuu uuuu
U1BRG* (U1MD=1)	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
U1USR	0000 1011	0000 1011	0000 1011	uuuu uuuu
SPIC0	111- --00	111- --00	111- --00	uuu- --uu
SPIC1	--00 0000	--00 0000	--00 0000	--uu uuuu
SPID	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
EEC	0000 0000	0000 0000	0000 0000	uuuu uuuu
STKPTR	0--- 0000	0--- 0000	0--- 0000	u--- 0000
IECC	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCRL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCRH	-000 0000	-000 0000	-000 0000	-uuu uuuu
CRCCR	---- ---0	---- ---0	---- ---0	---- ---u

寄存器	上电复位	LVR 复位 (正常操作)	WDT 溢出 (正常操作)	WDT 溢出 (空闲 / 休眠)
CRCIN	0000 0000	0000 0000	0000 0000	uuuu uuuu
CRCDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
CRCDH	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMC0	0000 0---	0000 0---	0000 0---	uuuu u---
STMC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDH	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAL	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAH	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMRP	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0C0	0000 0---	0000 0---	0000 0---	uuuu u---
PTM0C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DH	---- --00	---- --00	---- --00	---- --uu
PTM0AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0AH	---- --00	---- --00	---- --00	---- --uu
PTM0RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	---- --00	---- --00	---- --00	---- --uu
PTM1C0	0000 0---	0000 0---	0000 0---	uuuu u---
PTM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DH	---- --00	---- --00	---- --00	---- --uu
PTM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1AH	---- --00	---- --00	---- --00	---- --uu
PTM1RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	---- --00	---- --00	---- --00	---- --uu
IFS	---- ---0	---- ---0	---- ---0	---- ---0
PAS0	00-- 00--	00-- 00--	00-- 00--	uu—uu--
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	--00 0000	--00 0000	--00 0000	--uu 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
PGS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PGS1	---- 0000	---- 0000	---- 0000	---- uuuu
PHS0	0000 ----	0000 ----	0000 ----	uuuu ----

寄存器	上电复位	LVR 复位 (正常操作)	WDT 溢出 (正常操作)	WDT 溢出 (空闲 / 休眠)
COMS	---- --00	---- --00	---- --00	---- --uu
PMPS	---- --00	---- --00	---- --00	---- --uu

注：“u”表示不改变
“x”表示未知
“-”表示未定义
“*”：UnUCR1 和 SIMnC1 寄存器共用同一个存储器地址，UnBRG 和 SIMnTOC 寄存器共用同一个存储器地址。复位发生后，通过应用程序手动设置 UnMD 位为“1”后可获得 UnUCR1 和 UnBRG 寄存器的默认值。

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。

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	—	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	—	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	—	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PE	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEC	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PF	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFC	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0
PG	—	—	PG5	PG4	PG3	PG2	PG1	PG0
PGC	—	—	PGC5	PGC4	PGC3	PGC2	PGC1	PGC0
PGPU	—	—	PGPU5	PGPU4	PGPU3	PGPU2	PGPU1	PGPU0

寄存器名称	位							
	7	6	5	4	3	2	1	0
PH	—	—	—	PH4	PH3	PH2	—	—
PHC	—	—	—	PHC4	PHC3	PHC2	—	—
PHPU	—	—	—	PHPU4	PHPU3	PHPU2	—	—

“—”：未定义，读为“0”

输入 / 输出逻辑功能寄存器列表

上拉电阻

许多产品应用在端口处于输入状态时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，当引脚规划为数字输入时，可由内部连接到一个上拉电阻。这些上拉电阻可通过相应的上拉控制寄存器 PAPU~PHPU 来设置，它用一个 PMOS 晶体管来实现上拉电阻功能。

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

- **PxPU 寄存器**

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

PxPUn：I/O Px 口上拉电阻控制位
0：除能
1：使能

PxPUn 位用于控制引脚上拉电阻功能。这里的 x 可以是端口 A、B、C、D、E、F、G 或 H。但是，每个 I/O 端口实际有效位可能不同。

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 口的某一个引脚发生从高电平到低电平的转换。这个功能特别适合于通过外部开关来唤醒的应用。PA 口的每个引脚可以通过设置 PAWU 寄存器来单独选择是否具有唤醒功能。

需要注意的是，只有当引脚被设置为通用功能输入类型且单片机处于空闲 / 休眠模式时，唤醒功能才会受唤醒控制寄存器控制开启，其它状态下此唤醒功能不可用。

- **PAWU 寄存器**

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

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

输入 / 输出端口控制寄存器

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

- **PxC 寄存器**

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

PxCn：I/O Px 口输入 / 输出类型选择位
0：输出
1：输入
PxCn 位用于控制引脚类型。这里的 x 可以是端口 A、B、C、D、E、F、G 或 H。但是，每个 I/O 端口实际有效位可能不同。

输入 / 输出端口电源控制

此单片机为 PD0~PD3 引脚提供了不同的端口电源选择。多电源功能仅在引脚功能被设置为具有数字输入或输出的功能 ($\overline{\text{RES}}$ 和 OCDS 功能除外) 时有效。

通过设定 PMPS 寄存器中的 PMPS1~PMPS0 位可确定端口电源是来自电源引脚 VDD 或 VDDIO。若来自 VDDIO 引脚则该引脚功能必须通过相应的引脚共用功能选择位预先设定。

必须注意的是，若 VDDIO 引脚被选作端口电源引脚，则该引脚上的输入电源电压应小于或等于单片机电源电压 V_{DD}。

- **PMPS 寄存器**

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

Bit 7~2　未定义，读为“0”

Bit 1~0　**PMPS1~PMPS0**：PD0~PD3 引脚电源选择
0x：V_{DD}
1x：V_{DDIO}
若 PB6 引脚切换到 VDDIO 功能且 PMPS1~PMPS0 位设置为“1x”，则 VDDIO 引脚输入电压可作为 PD0~PD3 引脚电源。

引脚共用功能

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

引脚共用功能选择寄存器

封装中有限的引脚个数会对某些单片机功能造成影响。然而，引脚功能共用和引脚功能选择，使得小封装单片机具有更多不同的功能。单片机包含端口“x”输出功能选择寄存器“n”，记为 PxSn，和输入功能选择寄存器，记为 IFS，这些寄存器可以用来选择多功能共用引脚上的特定功能。此外，还有一个 COMS 寄存器可在 LCD 的 SEG 和 COM 功能共用同一个引脚的时候进行功能选择。

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	—	—	PAS03	PAS02	—	—
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	—	—	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
PDS1	PDS17	PDS16	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
PES0	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
PES1	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
PFS0	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
PFS1	PFS17	PFS16	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10
PGS0	PGS07	PGS06	PGS05	PGS04	PGS03	PGS02	PGS01	PGS00
PGS1	—	—	—	—	PGS13	PGS12	PGS11	PGS10
PHS0	PHS07	PHS06	PHS05	PHS04	—	—	—	—
IFS	—	—	—	—	—	—	—	STPIPS
COMS	—	—	—	—	—	—	COMS1	COMS0

引脚共用功能选择寄存器列表

- **PAS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	—	—	PAS03	PAS02	—	—
R/W	R/W	R/W	—	—	R/W	R/W	—	—
POR	0	0	—	—	0	0	—	—

Bit 7~6 **PAS07~PAS06**：PA3 引脚共用功能选择
00：PA3
01：SCK0/SCL0
10：PA3
11：PA3

Bit 5~4 未定义，读为“0”

Bit 3~2 **PAS03~PAS02**：PA1 引脚共用功能选择
00：PA1
01：$\overline{SCS0}$
10：PA1
11：PA1

Bit 1~0 未定义，读为“0”

- **PAS1 寄存器**

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

Bit 7~6 **PAS17~PAS16**：PA7 引脚共用功能选择
00：PA7/INT1
01：AI6/AN10
10：PA7/INT1
11：PA7/INT1

Bit 5~4 **PAS15~PAS14**：PA6 引脚共用功能选择
00：PA6/INT0
01：VG
10：AI7/AN11
11：PA6/INT0

Bit 3~2 **PAS13~PAS12**：PA5 引脚共用功能选择
00：PA5
01：PA5
10：PA5
11：ATP/ATP_PWM1

Bit 1~0 **PAS11~PAS10**：PA4 引脚共用功能选择
00：PA4
01：PA4
10：PA4
11：ATPB/ATP_PWM2

- **PBS0 寄存器**

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

Bit 7~6 **PBS07~PBS06**：PB3 引脚共用功能选择
00：PB3
01：PB3
10：PB3
11：AN1

Bit 5~4 **PBS05~PBS04**：PB2 引脚共用功能选择
00：PB2
01：PB2
10：PB2
11：AN0

Bit 3~2 **PBS03~PBS02**：PB1 引脚共用功能选择
00：PB1
01：PB1
10：PB1
11：AI4/AN8

Bit 1~0 **PBS01~PBS00**：PB0 引脚共用功能选择
00：PB0
01：PB0
10：PB0
11：AI5/AN9

● **PBS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PBS17~PBS16**：PB7 引脚共用功能选择
00：PB7
01：SCK1/SCL1
10：SEG34
11：AN5

Bit 5~4 **PBS15~PBS14**：PB6 引脚共用功能选择
00：PB6
01：VDDIO
10：VREF
11：AN4

Bit 3~2 **PBS13~PBS12**：PB5 引脚共用功能选择
00：PB5
01：PTP0B
10：PB5
11：AN3

Bit 1~0 **PBS11~PBS10**：PB4 引脚共用功能选择
00：PB4
01：PTP0
10：PB4
11：AN2

● **PCS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PCS07~PCS06**：PC3 引脚共用功能选择
00：PC3
01：XT1
10：PC3
11：PC3

Bit 5~4 **PCS05~PCS04**：PC2 引脚共用功能选择
00：PC2/INT2
01：SDO1/UTX1
10：PC2/INT2
11：PC2/INT2

Bit 3~2 **PCS03~PCS02**：PC1 引脚共用功能选择
00：PC1/INT3
01：SDI1/SDA1/URX1/UTX1
10：PC1/INT3
11：PC1/INT3

Bit 1~0 **PCS01~PCS00**：PC0 引脚共用功能选择
00：PC0
01：SDI0/SDA0/URX0/UTX0
10：PC0
11：PC0

● **PCS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
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 **PCS15~PCS14**：PC6 引脚共用功能选择
00：PC6
01：SDO0/UTX0
10：PC6
11：PC6

Bit 3~2 **PCS13~PCS12**：PC5 引脚共用功能选择
00：PC5/ATCK
01：$\overline{\text{SCS1}}$
10：PC5/ATCK
11：SEG35

Bit 1~0 **PCS11~PCS10**：PC4 引脚共用功能选择
00：PC4
01：XT2
10：PC4
11：PC4

● **PDS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PDS07~PDS06**：PD3 引脚共用功能选择
00：PD3
01：PD3
10：$\overline{\text{SPISCS}}$
11：SEG3

Bit 5~4 **PDS05~PDS04**：PD2 引脚共用功能选择
00：PD2/PTCK0
01：PD2/PTCK0
10：SPISDO
11：SEG2

Bit3~2 **PDS03~PDS02**：PD1 引脚共用功能选择
00：PD1/PTP0I
01：SPISDI
10：PTP0B
11：SEG1

Bit 1~0 **PDS01~PDS00**：PD0 引脚共用功能选择
00：PD0
01：SPISCK
10：PTP0
11：SEG0

● **PDS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PDS17	PDS16	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
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 **PDS17~PDS16**：PD7 引脚共用功能选择
00：PD7
01：OSC2
10：PD7
11：SEG7

Bit 5~4 **PDS15~PDS14**：PD6 引脚共用功能选择
00：PD6
01：OSC1
10：PD6
11：SEG6

Bit 3~2 **PDS13~PDS12**：PD5 引脚共用功能选择
00：PD5
01：PD5
10：ATPB/ATP_PWM2
11：SEG5

Bit 1~0 **PDS11~PDS10**：PD4 引脚共用功能选择
00：PD4
01：PD4
10：ATP/ATP_PWM1
11：SEG4

● **PES0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PES07~PES06**：PE3 引脚共用功能选择
00：PE3
01：PE3
10：PTP1
11：SEG11

Bit 5~4 **PES05~PES04**：PE2 引脚共用功能选择
00：PE2/STCK
01：PE2/STCK
10：PE2/STCK
11：SEG10

Bit 3~2 **PES03~PES02**：PE1 引脚共用功能选择
00：PE1/STPI
01：PE1/STPI
10：STPB
11：SEG9

Bit 1~0 **PES01~PES00**：PE0 引脚共用功能选择
00：PE0/STPI
01：PE0/STPI
10：STP
11：SEG8

● **PES1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
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 **PES17~PES16**：PE7 引脚共用功能选择
00：PE7
01：PE7
10：PE7
11：SEG15

Bit 5~4 **PES15~PES14**：PE6 引脚共用功能选择
00：PE6
01：PE6
10：PE6
11：SEG14

Bit 3~2 **PES13~PES12**：PE5 引脚共用功能选择
00：PE5/PTCK1
01：PE5/PTCK1
10：PE5/PTCK1
11：SEG13

Bit 1~0 **PES11~PES10**：PE4 引脚共用功能选择
00：PE4/PTP1I
01：PE4/PTP1I
10：PTP1B
11：SEG12

● **PFS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PFS07~PFS06**：PF3 引脚共用功能选择
00：PF3
01：PF3
10：PF3
11：SEG19

Bit 5~4 **PFS05~PFS04**：PF2 引脚共用功能选择
00：PF2
01：PF2
10：PF2
11：SEG18

Bit 3~2 **PFS03~PFS02**：PF1 引脚共用功能选择
00：PF1
01：PF1
10：PF1
11：SEG17

Bit 1~0 **PFS01~PFS00**：PF0 引脚共用功能选择
00：PF0
01：PF0
10：PF0
11：SEG16

● **PFS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PFS17	PFS16	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PFS17~PFS16**：PF7 引脚共用功能选择
00：PF7
01：PF7
10：PF7
11：SEG23

Bit 5~4 **PFS15~PFS14**：PF6 引脚共用功能选择
00：PF6
01：PF6
10：PF6
11：SEG22

Bit 3~2 **PFS13~PFS12**：PF5 引脚共用功能选择
00：PF5
01：PF5
10：PF5
11：SEG21

Bit 1~0 **PFS11~PFS10**：PF4 引脚共用功能选择
00：PF4
01：PF4
10：PF4
11：SEG20

● **PGS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PGS07	PGS06	PGS05	PGS04	PGS03	PGS02	PGS01	PGS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PGS07~PGS06**：PG3 引脚共用功能选择
00：PG3
01：PG3
10：PG3
11：SEG27

Bit 5~4 **PGS05~PGS04**：PG2 引脚共用功能选择
00：PG2
01：PG2
10：PG2
11：SEG26

Bit 3~2 **PGS03~PGS02**：PG1 引脚共用功能选择
00：PG1
01：PG1
10：PG1
11：SEG25

Bit 1~0 **PGS01~PGS00**：PG0 引脚共用功能选择
00：PG0
01：PG0
10：PG0
11：SEG24

● **PGS1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PGS13	PGS12	PGS11	PGS10
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4　未定义，读为“0”

Bit 3~2　**PGS13~PGS12**：PG5 引脚共用功能选择
00：PG5
01：PG5
10：COM6
11：SEG29

Bit 1~0　**PGS11~PGS10**：PG4 引脚共用功能选择
00：PG4
01：PG4
10：COM7
11：SEG28

● **PHS0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	PHS07	PHS06	PHS05	PHS04	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

Bit 7~6　**PHS07~PHS06**：PH3 引脚共用功能选择
00：PH3
01：PH3
10：PH3
11：SEG33

Bit 5~4　**PHS05~PHS04**：PH2 引脚共用功能选择
00：PH2
01：PH2
10：PH2
11：SEG32

Bit 3~0　未定义，读为“0”

● **IFS 寄存器**

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

Bit 7~1　未定义，读为“0”

Bit 0　**STPIPS**：STPI 输入源引脚选择
0：PE0
1：PE1

● **COMS 寄存器**

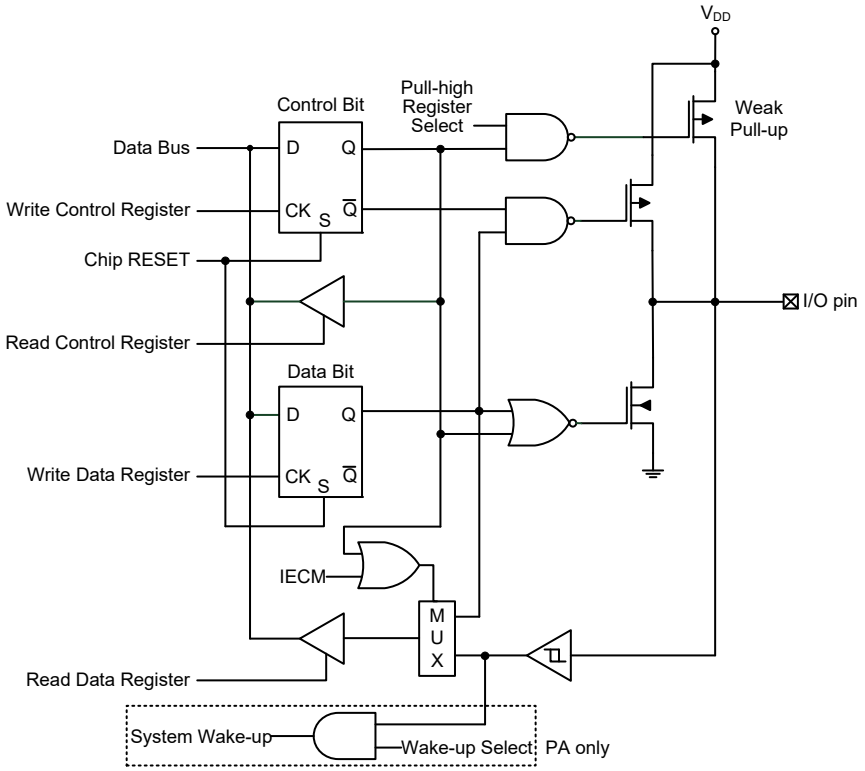
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	COMS1	COMS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未定义，读为“0”

Bit 1 **COMS1**：COM5/SEG30 引脚共用功能选择
0：SEG30
1：COM5
Bit 0 **COMS0**：COM4/SEG31 引脚共用功能选择
0：SEG31
1：COM4

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。



输入 / 输出端口逻辑功能结构

读端口功能

读端口功能用于管理数据读取路径，即读取的数据来自数据锁存还是 I/O 引脚。该功能专为 I/O 功能和 A/D 通道上的 IEC 60730 自诊断测试而设计。寄存器 IECC 用于控制读端口功能。若此功能除能，引脚将作为所选中的共用功能引脚。若向寄存器 IECC 写入一个特定的数据模式“11001010”，内部信号 IECM 将被置高以使能读端口功能。读端口功能使能后执行读端口指令“mov a, Px”，相应引脚上的值将被传至累加器 ACC，其中“x”代表相应的 I/O 端口名称。读端口功能只控制读取路径，不会影响到引脚功能配置以及当前单片机的操作。若向 IECC 寄存器写入除 11001010 以外的其它值，内部信号 IECM 将被清零，除能读端口功能且数据读取路径来自锁存器。若此功能除能，引脚将作为所选中的共用引脚功能进行正常操作。

● **IECC 寄存器**

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

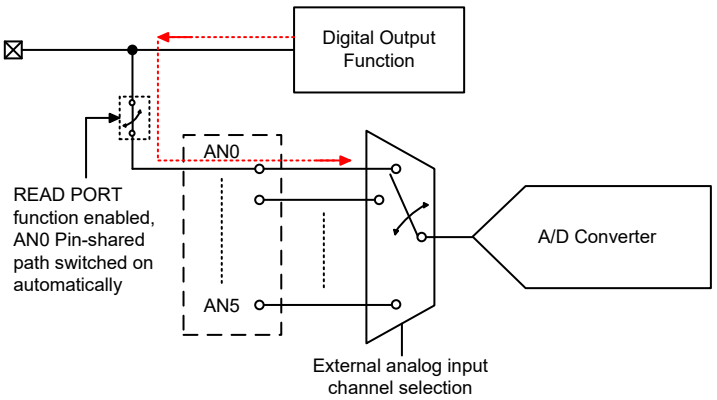
Bit 7~0 **IECS7~IECS0**：读端口功能使能控制 bit 7 ~ bit 0
11001010：IECM=1 – 读端口功能使能
其它值：IECM=0 – 读端口功能除能

读端口功能	除能		使能	
端口控制寄存器位 – PxC.n	1	0	1	0
I/O 功能	引脚值	数据锁存值	引脚值	
数字输入功能				
数字输出功能 (USIMn 除外)	0			
USIMn：SCKn/SCLn，SDIn/SDAn，URXn/UTXn	引脚值			
模拟功能	0			

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

读端口模式的另一个功能是检查 A/D 通道。当读端口功能除能，若相应的选择位没有选中 A/D 输入引脚功能，则从外部引脚到内部模拟输入的 A/D 通道将会被关闭。对于带 A/D 转换通道的单片机，如 A/D AN0~AN5，通过适当配置 A/D 控制寄存器中的外部模拟输入通道选择位并选中相应的模拟输入引脚功能，所需的 A/D 转换通道将被开启。而读端口模式的功能则是强制开启 A/D 通道。例如，无论 AN0 是否选择作为模拟输入引脚使用，只要读端口功能使能，AN0 模拟输入通道都将开启。通过这种方式，AN0 模拟输入路径可与其共用引脚上的数字输出内部连接，然后在无外接模拟输入电压的情况下对相应的数字数据进行转换，从而实现 AN0 模拟输入通道检测。

注意，当使用读端口功能检查 A/D 路径时，A/D 转换器参考电压需等于 I/O 电源电压。



A/D 通道输入路径内部连接

编程注意事项

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

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

定时器模块 – TM

控制和测量时间在任何单片机中都是一个很重要的部分。该单片机提供几个定时器模块 (简称 TM)，来实现和时间有关的功能。定时器模块是包括多种操作的定时单元，提供的操作有：定时 / 事件计数器，捕捉输入，比较匹配输出，单脉冲输出以及 PWM 输出等功能。每个定时器模块有两个或三个独立中断。每个 TM 外加的输入输出引脚，扩大了定时器的灵活性，便于用户使用。

这里只介绍各种 TM 的共性，更多详细资料请参考标准型、周期型和音频型定时器章节。

简介

该单片机包含多个 TM，每个 TM 可被划分为一个特定的类型，即标准型 TM、周期型 TM 或音频型 TM。其中标准型和音频型 TM 各有一个，分别命名为 STM 和 ATM。周期型 TM 有三个，命名为 PTM0~PTM1。虽然性质相似，但不同 TM 特性复杂度不同。本章介绍标准型、周期型和音频型 TM 的共性，更多详细资料分别见后面各章。此三种类型 TM 的特性和区别见下表。

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

TM 功能概要

TM 操作

不同类型的 TM 提供从简单的定时操作到 PWM 信号产生等多种功能。理解 TM 操作的关键是比较 TM 内独立运行的计数器的值与内部比较器的预置值。当计数器的值与比较器的预置值相同时，则比较匹配，TM 中断信号产生，清零计数器并改变 TM 输出引脚的状态。用户选择内部时钟或外部引脚输入时钟来驱动内部 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 xTnCK2~xTnCK0 位，选择所需的时钟源，其中 x 代表 S、P 或 A 型 TM，n 代表具体 TM 编号。由于该单片机只包含一个 STM 和一个 ATM，因此其相关的引脚、寄存器和控制位名称中都不带编号。TM 时钟源来自系统时钟 f_{SYS} 或内部高速时钟 fH 的分频比或 f_{SUB} 时钟源或外部 xTCKn 引脚。xTCKn 引脚时钟源可允许外部信号作为 TM 时钟源或用于事件计数。

TM 中断

每个标准型或周期型 TM 都有两个内部中断，分别是内部比较器 A 或比较器 P，当比较匹配发生时产生 TM 中断。音频型 TM 包含三个内部比较器，即内部比较器 A，比较器 B 和比较器 P，当比较匹配发生时，可产生中断，因此 ATM 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 都有一个 TM 输入引脚 xTCKn，而 STM 和 PTMn 还有另一个输入引脚 xTPnI。xTMn 输入引脚 xTCKn 作为 xTMn 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 引脚可选择上升沿有效或下降沿有效。xTCKn 引脚还可用作 STM 或 PTMn 单脉冲输出模式的外部触发引脚。

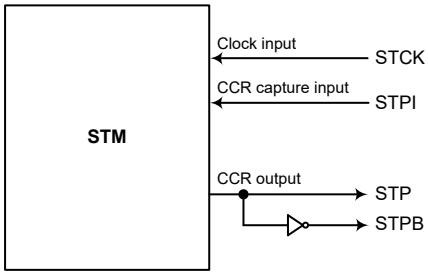
对于 STM 和 PTMn，xTPnI 引脚作为捕捉输入脚，其有效边沿有上升沿、下降沿和双沿，通过设置 xTMnC1 寄存器中的 xTnIO1~xTnIO0 位来选择有效边沿类型。除 PTPnI 引脚外，PTCKn 引脚也可用作 PTMn 捕捉输入模式的外部触发引脚。

每个 TM 都有两个输出引脚 xTPn 和 xTPnB。xTPnB 是 xTPn 输出的反相信号。当 TM 工作在比较匹配输出模式且比较匹配发生时，这些引脚会由 TM 控制切换到高电平或低电平或翻转。外部 xTPn 和 xTPnB 输出引脚也被 TM 用来产生 PWM 输出波形。

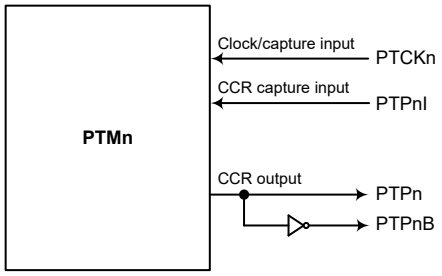
由于 TM 输入和输出引脚与其它功能共用同一引脚，在使用 TM 功能前，用户需要先通过引脚共用功能选择寄存器对相应位进行设置以选择 TM 引脚功能。更多引脚共用功能选择详见引脚共用功能章节。

STM		PTM		ATM	
输入	输出	输入	输出	输入	输出
STCK, STPI	STP, STPB	PTCK0, PTP0I PTCK1, PTP1I	PTP0, PTP0B PTP1, PTP1B	ATCK	ATP/ATP_PWM1, ATPB/ATP_PWM2

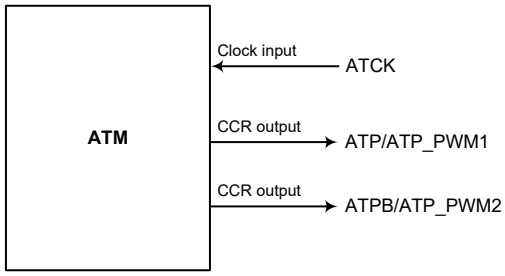
TM 外部引脚



STM 功能引脚方框图



PTMn 功能引脚框方图 (n=0~1)

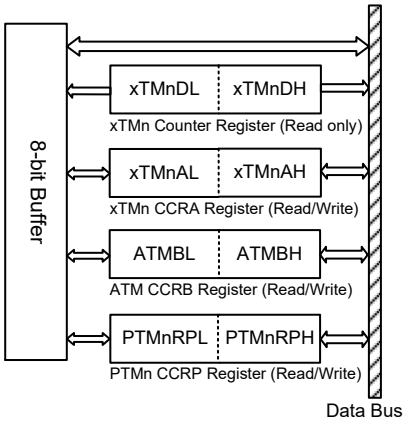


ATM 功能引脚方框图

编程注意事项

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

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



读写流程如下步骤所示：

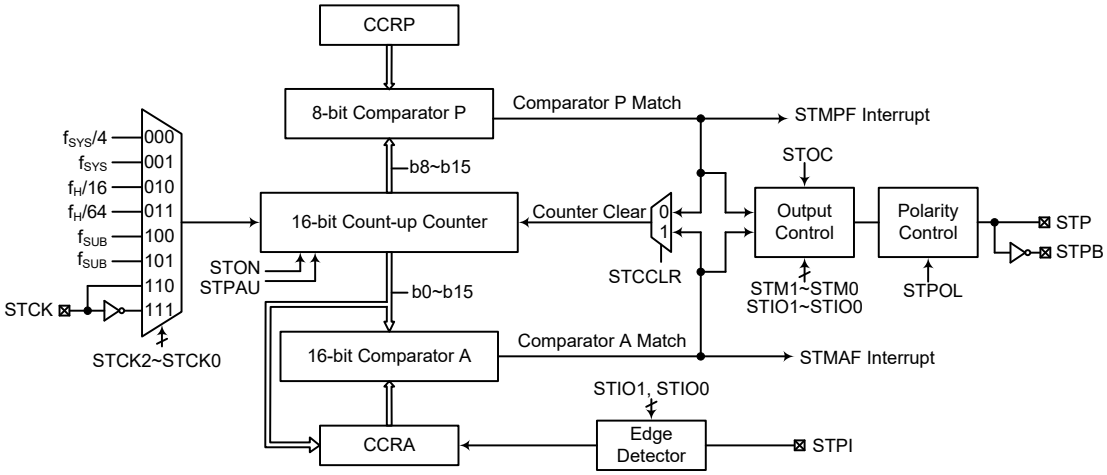
- 写数据至 CCRA、CCRB 或 CCRP
  - ♦ 步骤 1. 写数据至低字节寄存器 xTMnAL、ATMBL 或 PTMnRPL
    - – 注意，此时数据仅写入 8-bit 缓存器。

♦ 步骤 2. 写数据至高字节寄存器 xTMnAH、ATMBH 或 PTMnRPH
– 注意，此时数据直接写入高字节寄存器，同时锁存在 8-bit 缓存器中的数据写入低字节寄存器。

● 由计数器寄存器和 CCRA、CCRB 或 CCRP 中读取数据
♦ 步骤 1. 由高字节寄存器 xTMnDH、xTMnAH、ATMBH 或 PTMnRPH 读取数据
– 注意，此时高字节寄存器中的数据直接读取，同时由低字节寄存器读取的数据锁存至 8-bit 缓存器中。
♦ 步骤 2. 由低字节寄存器 xTMnDL、xTMnAL、ATMBL 或 PTMnRPL 读取数据
– 注意，此时读取 8-bit 缓存器中的数据。

标准型 TM – STM

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



注：STM 外部引脚与其它功能共用引脚，因此在使用 STM 功能前，需确保已通过相关的引脚共用功能选择寄存器选择了 STM 引脚功能。对于 STCK 和 STPI 引脚还需设置相应的端口控制寄存器，将该引脚设置为输入口。

16-bit 标准型 TM 方框图

标准型 TM 操作

标准型 TM 核心是一个由用户选择的内部或外部时钟源驱动的 16 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 STON 位发生上升沿跳变清除计数器。此外，计数器溢出或比较匹配也会自动清除计数器。上述条件发生时，通常情况会产生 STM 中断信号。标准型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，也可以控制两个输出脚。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，STMRP 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
STMC0	STPAU	STCK2	STCK1	STCK0	STON	—	—	—
STMC1	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
STMDL	D7	D6	D5	D4	D3	D2	D1	D0
STMDH	D15	D14	D13	D12	D11	D10	D9	D8
STMAL	D7	D6	D5	D4	D3	D2	D1	D0
STMAH	D15	D14	D13	D12	D11	D10	D9	D8
STMRP	D7	D6	D5	D4	D3	D2	D1	D0

16-bit 标准型 TM 寄存器列表

- **STMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	STPAU	STCK2	STCK1	STCK0	STON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **STPAU**：STM 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，STM 保持上电状态并继续耗电。当此位由低到高转换时，计数器将保留其当前计数值，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 **STCK2~STCK0**：STM 计数时钟选择位
000：$f_{SYS}/4$
001：f_{SYS}
010：$f_H/16$
011：$f_H/64$
100：f_{SUB}
101：f_{SUB}
110：STCK 上升沿时钟
111：STCK 下降沿时钟
此三位用于选择 STM 的时钟源。外部引脚时钟源能被选择在上升沿或下降沿有效。f_{SYS} 是系统时钟，f_H 和 f_{SUB} 是其它的内部时钟源，细节方面请参考工作模式和系统时钟章节。

Bit 3 **STON**：STM 计数器 On/Off 控制位
0：Off
1：On
此位控制 STM 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 STM。清零此位将停止计数器并关闭 STM 减少耗电。当此位经由高到低转换时，内部计数器将保持其当前计数值，直到此位再次改变为高电平。
若 STM 处于比较匹配输出模式、PWM 输出模式或单脉冲输出模式，当 STON 位经由低到高转换时，STM 输出脚将复位至 STOC 位指定的初始值。

Bit 2~0 未定义，读为“0”

● **STMC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
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 **STM1~STM0**：STM 工作模式选择位
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 STM 需要的工作模式。为了确保操作可靠，STM 应在 STM1 和 STM0 位有任何改变前先关掉。在定时 / 计数器模式，STM 输出脚状态为未知。

Bit 5~4 **STIO1~STIO0**：STM 外部引脚功能选择位
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：强制无效状态
01：强制有效状态
10：PWM 输出
11：单脉冲输出
捕捉输入模式
00：在 STPI 上升沿输入捕捉
01：在 STPI 下降沿输入捕捉
10：在 STPI 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 STM 外部引脚如何改变状态。这两位值的选择取决于 STM 运行在哪种模式下。
在比较匹配输出模式下，STIO1 和 STIO0 位决定当从比较器 A 比较匹配输出发生时 STM 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 STM 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。STM 输出脚的初始值通过 STMC1 寄存器的 STOC 位设置取得。注意，由 STIO1 和 STIO0 位得到的输出电平必须与通过 STOC 位设置的初始值不同，否则当比较匹配发生时，STM 输出脚将不会发生变化。在 STM 输出脚改变状态后，通过 STON 位由低到高电平的转换复位至初始值。
在 PWM 输出模式，STIO1 和 STIO0 用于决定比较匹配条件发生时怎样改变 STM 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。需注意，只可在 STM 关闭后才能更改 STIO1 和 STIO0 位的值。若在 STM 运行时改变 STIO1 和 STIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **STOC**：STM STP 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 STM 输出脚输出控制位。它取决于 STM 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 STM 处于定时 / 计数器模式，此位不起作用。在比较匹配输出模式时，其决定比较匹配发生前 STM 输出脚的逻辑电平值。在 PWM 输出模式时，其决定 PWM 信号是高有效还是低有效。在单脉

冲输出模式，其决定 STON 位由低变高时 STM 输出脚的逻辑电平。

Bit 2 **STPOL**：STM STP 输出极性控制位
0：同相
1：反相
此位控制 STP 输出脚的极性。此位为高时 STM 输出脚反相，为低时 STM 输出脚同相。若 STM 处于定时 / 计数器模式时此位不起作用。

Bit 1 **STDPX**：STM 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 寄存器哪个用于 PWM 波形的周期控制、哪个用于占空比控制。

Bit 0 **STCCLR**：选择 STM 计数器清零条件位
0：STM 比较器 P 匹配
1：STM 比较器 A 匹配
此位用于选择清除计数器的方法。标准型 STM 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。STCCLR 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。STCCLR 位在 PWM 输出、单脉冲或输入捕捉模式时未使用。

● STMDL 寄存器

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

Bit 7~0 **D7~D0**：STM 计数器低字节寄存器 bit 7 ~ bit 0
STM 16-bit 计数器 bit 7 ~ bit 0

● STMDH 寄存器

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

Bit 7~0 **D15~D8**：STM 计数器高字节寄存器 bit 7 ~ bit 0
STM 16-bit 计数器 bit 15 ~ bit 8

● STMAL 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**：STM CCRA 低字节寄存器 bit 7 ~ bit 0
STM 16-bit CCRA bit 7 ~ bit 0

● **STMAH 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 **D15~D8**：STM CCRA 高字节寄存器 bit 7 ~ bit 0
STM 16-bit CCRA bit 15 ~ bit 8

● **STMRP 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**：STM CCRP 8-bit 寄存器，与 STM 计数器 bit 15 ~ bit 8 比较
比较器 P 匹配周期 =
0：65536 个 STM 时钟周期
1~255：(1~255)×256 个 STM 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 STCCLR 位设为 0 时，此比较结果可用于清除内部计数器。STCCLR 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 STMC1 寄存器的 STM1 和 STM0 位选择任意模式。

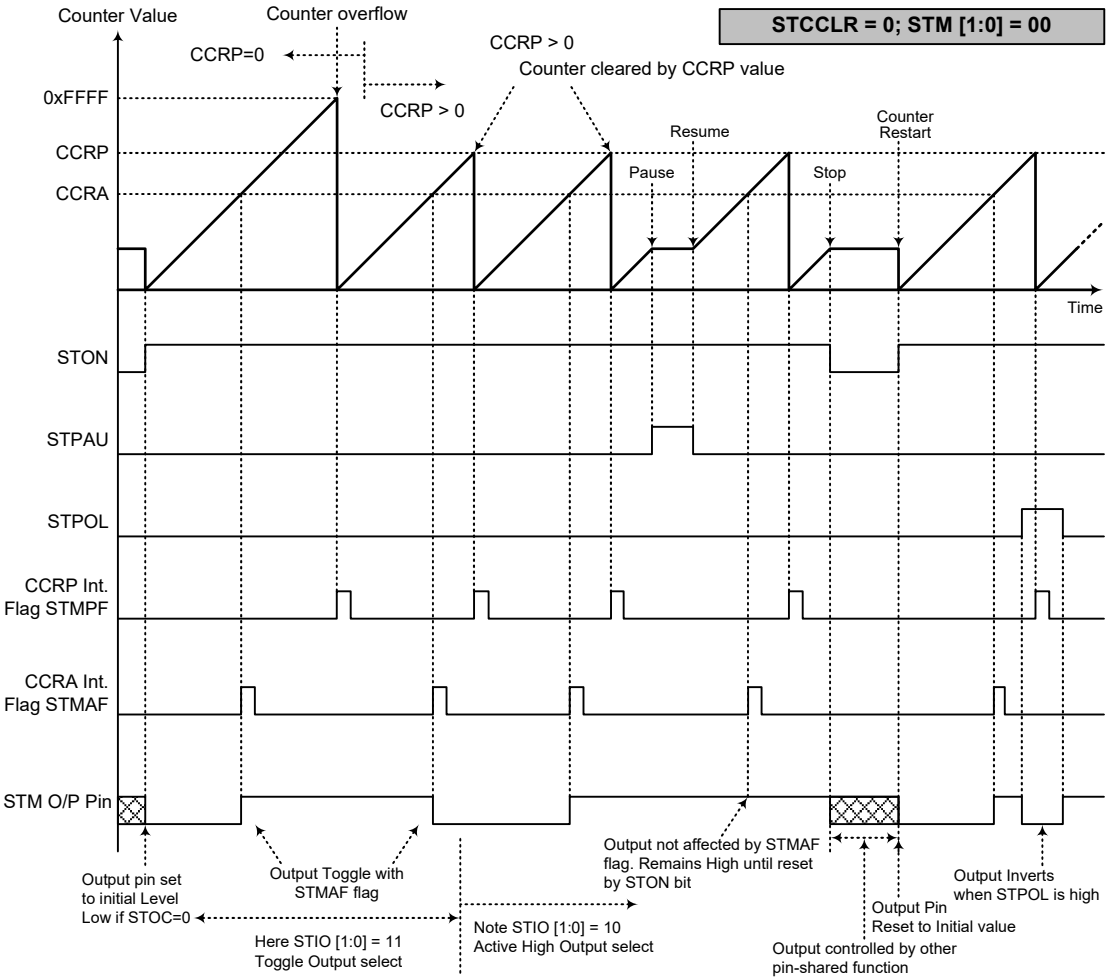
比较匹配输出模式

为使 STM 工作在此模式，STMC1 寄存器中的 STM1 和 STM0 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 STCCLR 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 STMAF 和 STMPF 将分别置位。

如果 STMC1 寄存器的 STCCLR 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 STMAF 中断请求标志。所以当 STCCLR 为高时，不会产生 STMPF 中断请求标志。在比较匹配输出模式下，CCRA 不能设为“0”。

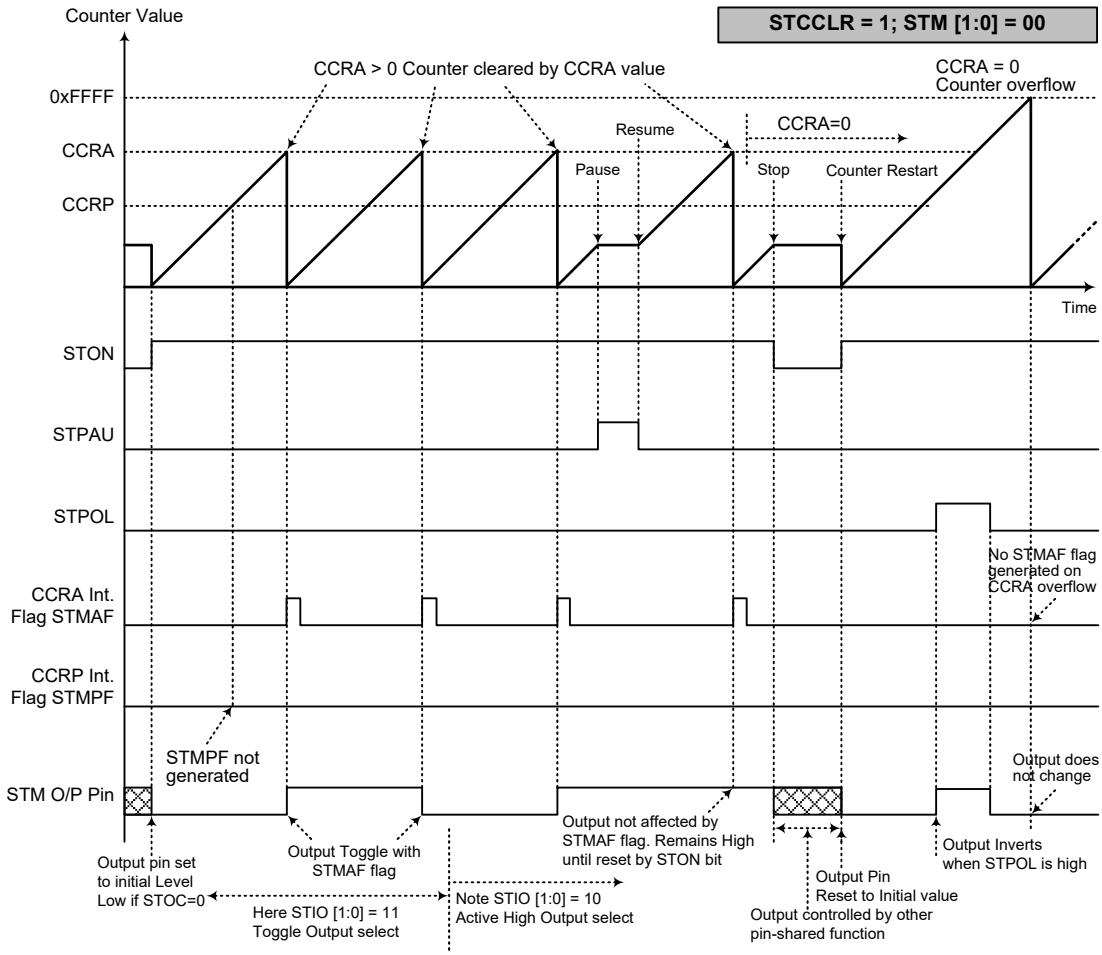
如果 CCRA 位都清除为零，当计数器的值达到 16 位最大值 FFFFH 时将溢出，但此时不会产生 STMAF 中断请求标志。

正如该模式名所言，当比较匹配发生后，STM 输出脚状态改变。当比较器 A 比较匹配发生后 STMAF 标志产生时，STM 输出脚状态改变。比较器 P 比较匹配发生时产生的 STMPF 标志不影响 STM 输出脚。STM 输出脚状态改变方式由 STMC1 寄存器中 STIO1 和 STIO0 位决定。当比较器 A 比较匹配发生时，STIO1 和 STIO0 位决定 STM 输出脚输出高，低或翻转当前状态。在 STON 位由低到高后，STM 输出脚初始状态为 STOC 位所指定的电平。注意，若 STIO1 和 STIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 – STCCLR=0

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



比较匹配输出模式 – STCCLR=1

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

定时 / 计数器模式

为使 STM 工作在此模式，STMC1 寄存器中的 STM1 和 STM0 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 STM 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 STM 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 STM 工作在此模式，STMC1 寄存器中的 STM1 和 STM0 位需要设置为“10”。STM 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 STM 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。

由于 PWM 波形的周期和占空比可调，其波形的选择就较为灵活。在 PWM 输出模式中，STCCLR 位不影响 PWM 周期。CCRP 和 CCRA 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 STMC1 寄存器的 STDPX 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。STMC1 寄存器中的 STOC 位决定 PWM 波形的极性，STIO1 和 STIO0 位使能 PWM 输出或将 STM 输出脚置为逻辑高或逻辑低。STPOL 位对 PWM 输出波形的极性取反。

- **16-bit STM，PWM 输出模式，边沿对齐模式，STDPX=0**

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

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

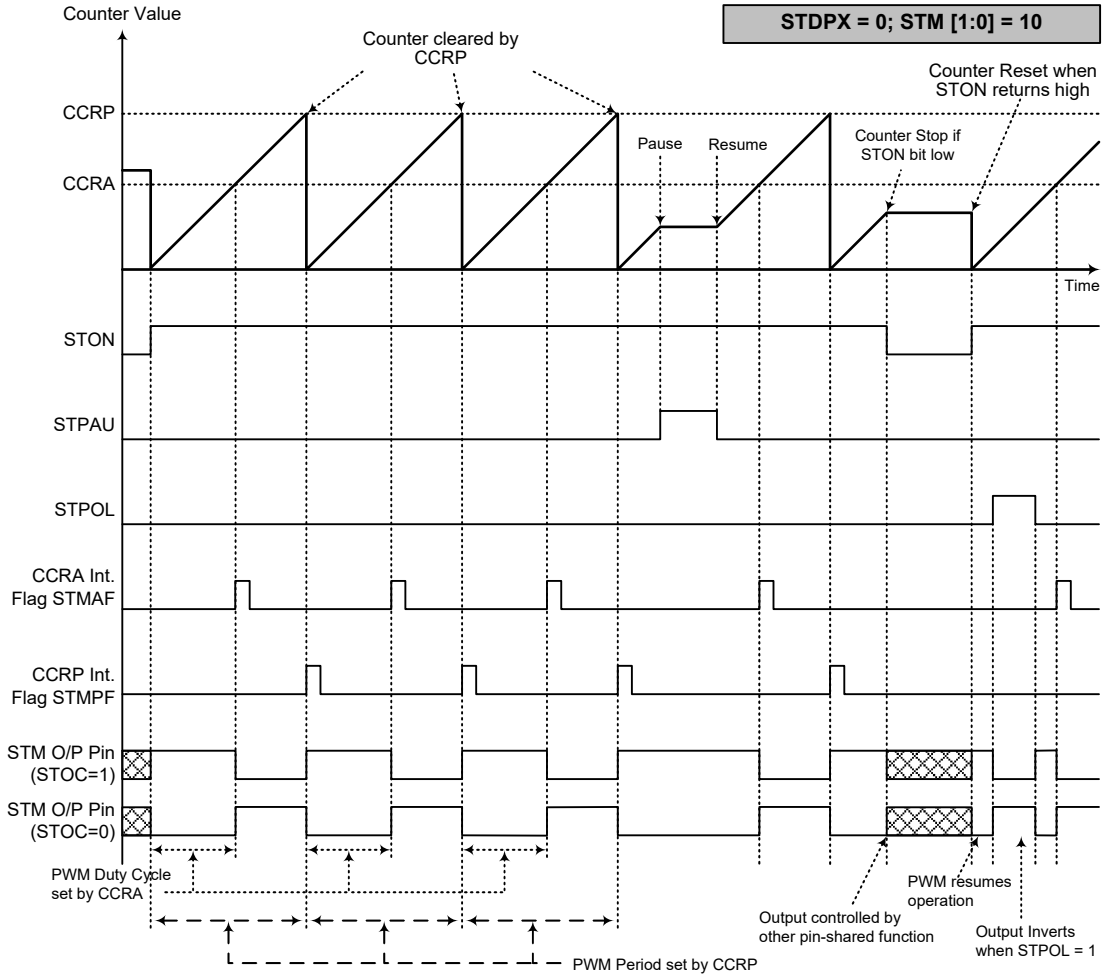
STM PWM 输出频率 = $(f_{SYS}/4)/(2\times256)=f_{SYS}/2048$=8kHz，duty=128/(2×256)=25%，

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 STM，PWM 输出模式，边沿对齐模式，STDPX=1**

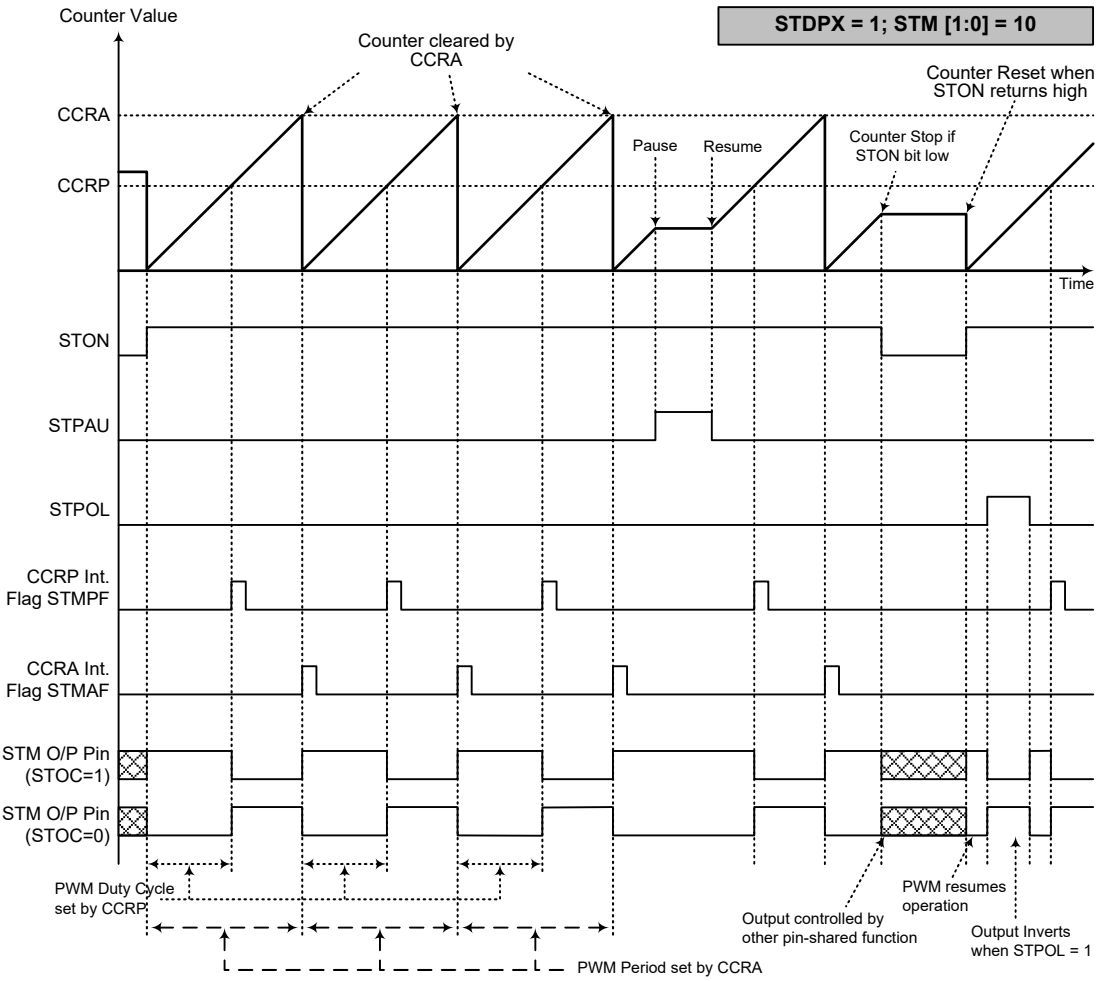
CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

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



PWM 输出模式 – STDPX=0

注：1. STDPX=0，CCRP 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 STIO[1:0]=00 或 01，PWM 功能不变
4. STCCLR 位不影响 PWM 操作



PWM 输出模式 – STDPX=1

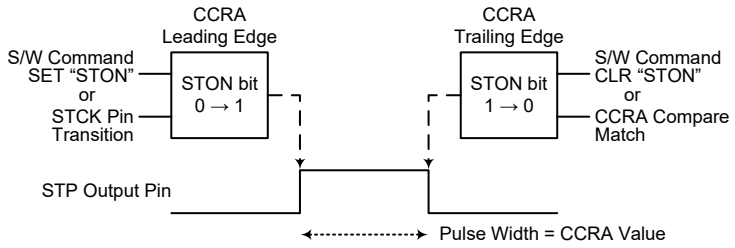
注：1. STDPX=1，CCRA 清除计数器
2. 计数器清零并设置 PWM 周期
3. 当 STIO[1:0]=00 或 01，PWM 功能不变
4. STCCLR 位不影响 PWM 操作

单脉冲输出模式

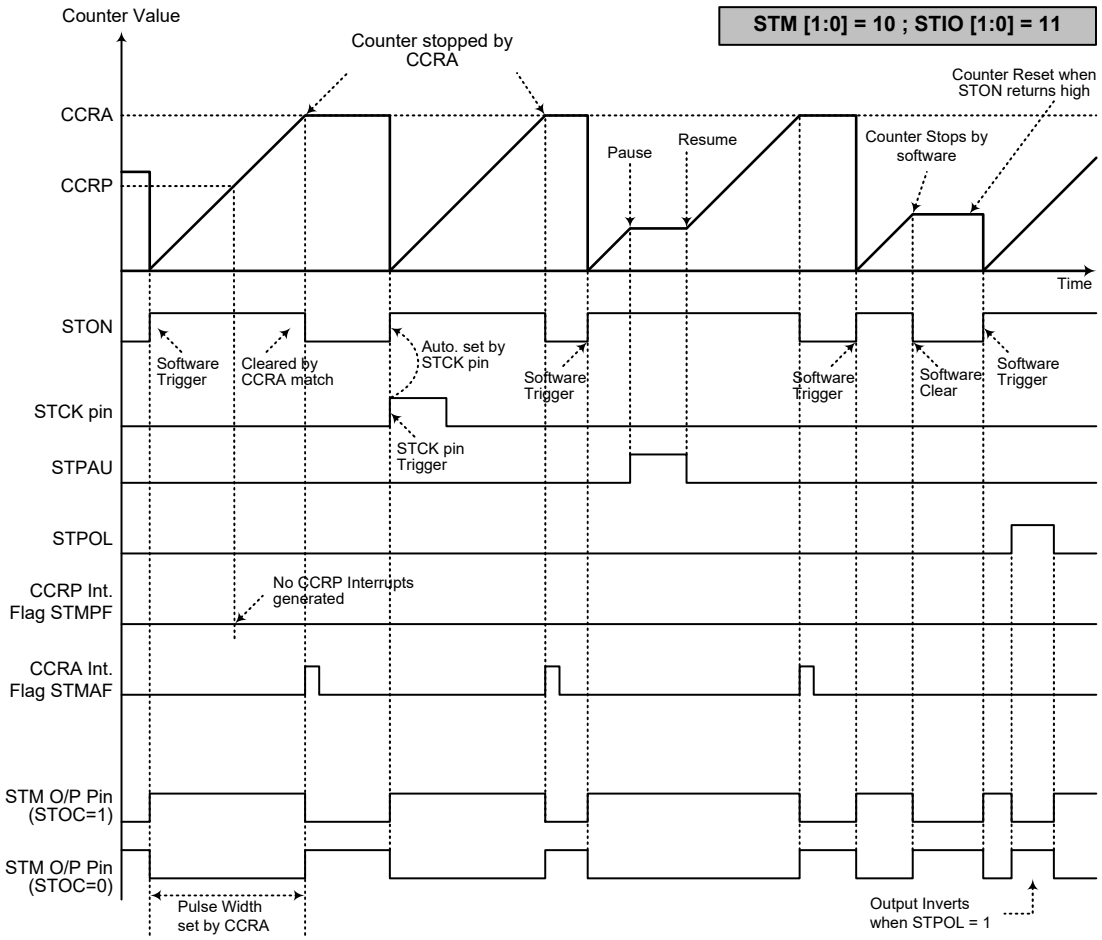
为使 STM 工作在此模式，STMC1 寄存器中的 STM1 和 STM0 位需要设置为“10”，同时 STIO1 和 STIO0 位需要设置为“11”。正如模式名所言，单脉冲输出模式，在 STM 输出脚将产生一个脉冲输出。

通过应用程序控制 STON 位由低到高的转变来触发脉冲前沿输出。而处于单脉冲输出模式时，STON 位可在 STCK 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 STON 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 STON 位保持高电平。通过应用程序使 STON 位清零或比较器 A 比较匹配发生时，产生脉冲后沿。

然而，比较器 A 比较匹配发生时，会自动清除 STON 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 STM 中断。STON 位在计数器重启时会发生由低到高的转变，此时计数器才复位至零。在单脉冲输出模式中，CCRP 寄存器、STCCLR 和 STDPX 位未使用。



单脉冲产生示意图



单脉冲输出模式

注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 STCK 脚或设置 STON 位为高来触发脉冲
4. STCK 脚有效沿会自动置位 STON
5. 单脉冲输出模式中，STIO[1:0] 需置为“11”，且不能更改

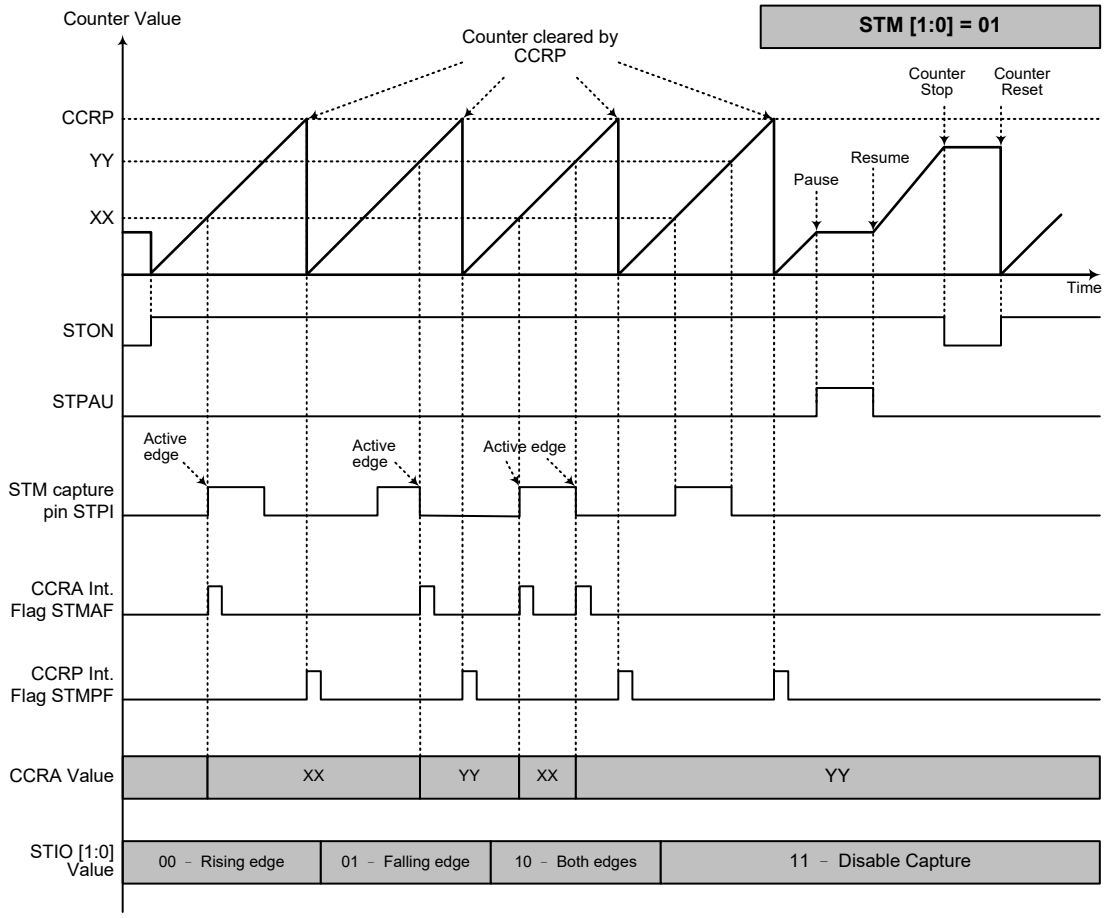
捕捉输入模式

为使 STM 工作在此模式，STMC1 寄存器中的 STM1 和 STM0 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。STPI 脚上的外部信号，通过设置 STMC1 寄存器的 STIO1 和 STIO0 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 STON 位由低到高转变时，计数器启动。

当 STPI 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 STM 中断。无论 STPI 引脚发生哪种边沿转换，计数器将继续工作直到 STON 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 STM 中断。记录 CCRP 溢出中断信号的值可以测量脉宽。通过设置 STIO1 和 STIO0 位选择 STPI 引脚为上升沿，下降沿或双沿有效。如果 STIO1 和 STIO0 位都设置为高，无论 STPI 引脚发生哪种边沿转换都不会产生捕捉操作，但计数器仍会继续运行。

有几点注意事项须留意。如果捕捉脉宽小于 2 个定时器时钟周期，则可能会被硬件忽略。当计数器的值被有效捕捉边沿锁存到 CCRA 寄存器后，再过 0.5 个定时器时钟周期，STMAF 标志位将被置高。从接收到有效捕捉边沿，到开始将计数器值锁存到 CCRA 寄存器的动作，这之间的延迟时间小于 1.5 个定时器时钟周期。

STCCLR 和 STDPX 位在此模式中未使用。

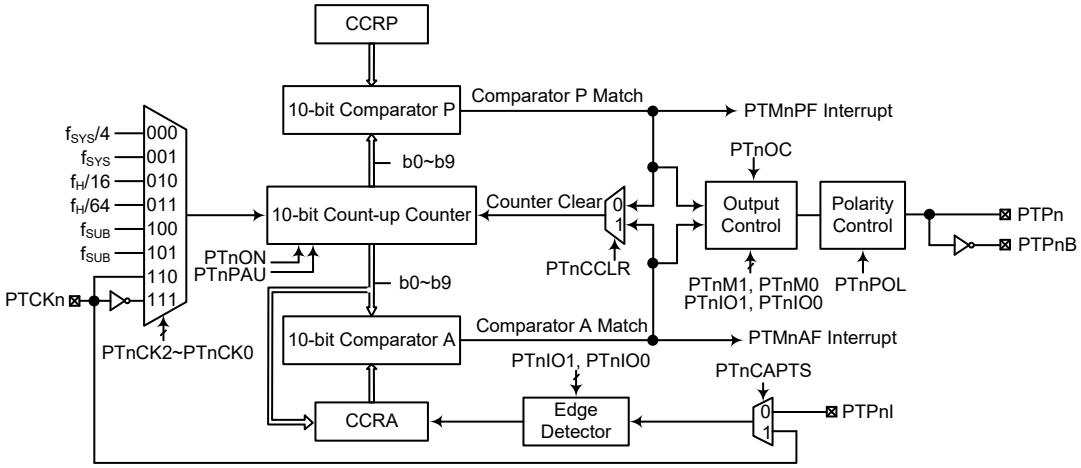


捕捉输入模式

注：1. STM[1:0]=01 并通过 STIO1 和 STIO0 位设置有效边沿
2. STM 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中
3. STCCLR 位未使用
4. 无输出功能 – STOC 和 STPOL 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. 捕捉输入模式需在有 STM 计数时钟的情况下才可使用

周期型 TM – PTM

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



注：PTMn 外部引脚与其它功能共用引脚，因此在使用 PTMn 功能前，需确保已通过相关的引脚共用功能选择寄存器选择了 PTMn 引脚功能。对于 PTCKn 和 PTPnI 引脚还需设置相应的端口控制寄存器，将该引脚设置为输入口。

10-bit 周期型 TM 方框图 (n=0~1)

周期型 TM 操作

周期型 TM 核心是一个由用户选择的内部时钟源驱动的 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 CCRA 和 CCRP 寄存器中的值进行比较。CCRP 和 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 PTnON 位发生上升沿跳变清除计数器。此外，计数器溢出或比较匹配也会自动清除计数器。上述条件发生时，通常情况会产生 PTMn 中断信号。周期型 TM 可工作在不同的模式，可由包括来自输入脚的不同时钟源驱动，也可以控制两个输出脚。所有工作模式的设定都是通过设置相关寄存器来实现的。

周期型 TM 寄存器介绍

周期型 TM 的所有操作由一系列寄存器控制。一对只读寄存器用来存放 10 位计数器的值，两对读 / 写寄存器存放 10 位 CCRA 和 CCRP 的值。剩下两个控制寄存器用来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
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8

寄存器名称	位							
	7	6	5	4	3	2	1	0
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	—	—	—	—	—	—	D9	D8

10-bit 周期型 TM 寄存器列表 (n=0~1)

- **PTMnC0 寄存器 (n=0~1)**

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

Bit 7 **PTnPAU**：PTMn 计数器暂停控制位
0：运行
1：暂停
通过设置此位为高可使计数器暂停，清零此位恢复正常计数器操作。当处于暂停状态时，PTMn 保持上电状态并继续耗电。当此位由低到高转变时，计数器将保留其当前计数值，直到此位再次改变为低电平，并从此值开始继续计数。

Bit 6~4 **PTnCK2~PTnCK0**：PTMn 计数时钟选择位
000：$f_{SYS}/4$
001：f_{SYS}
010：$f_H/16$
011：$f_H/64$
100：f_{SUB}
101：f_{SUB}
110：PTCKn 上升沿
111：PTCKn 下降沿
此三位用于选择 PTMn 的时钟源。外部引脚时钟源能被选择在上升沿或下降沿有效。f_{SYS} 是系统时钟，f_H 和 f_{SUB} 是其它的内部时钟源，细节方面请参考工作模式和系统时钟章节。

Bit 3 **PTnON**：PTMn 计数器 On/Off 控制位
0：Off
1：On
此位控制 PTMn 的总开关功能。设置此位为高则使能计数器使其运行，清零此位则除能 PTMn。清零此位将停止计数器并关闭 PTMn 减少耗电。当此位经由低到高转变时，内部计数器将复位清零；当此位经由高到低转换时，内部计数器将保持其当前计数值，直到此位再次改变为高电平。
若 PTMn 处于比较匹配输出模式、PWM 输出模式或单脉冲输出模式，当 PTnON 位经由低到高转换时，PTMn 输出脚将复位至 PTnOC 位指定的初始值。

Bit 2~0 未定义，读为“0”

- **PTMnC1 寄存器 (n=0~1)**

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
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

7~6 **PTnM1~PTnM0**：PTMn 工作模式选择位
00：比较匹配输出模式
01：捕捉输入模式
10：PWM 输出模式或单脉冲输出模式
11：定时 / 计数器模式
这两位设置 PTMn 需要的工作模式。为了确保操作可靠，PTMn 应在 PTnM1 和

PTnM0 位有任何改变前先关掉。在定时 / 计数器模式，PTMn 输出脚状态为未知。

Bit 5~4 **PTnIO1~PTnIO0**：PTMn 外部引脚功能选择位

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：强制无效状态
- 01：强制有效状态
- 10：PWM 输出
- 11：单脉冲输出

捕捉输入模式
- 00：在 PTPnI 或 PTCKn 上升沿输入捕捉
- 01：在 PTPnI 或 PTCKn 下降沿输入捕捉
- 10：在 PTPnI 或 PTCKn 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 PTMn 外部引脚如何改变状态。这两位值的选择取决于 PTMn 运行在哪种模式下。

在比较匹配输出模式下，PTnIO1 和 PTnIO0 位决定当从比较器 A 比较匹配输出发生时 PTMn 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 PTMn 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。PTMn 输出脚的初始值通过 PTMnC1 寄存器的 PTnOC 位设置取得。注意，由 PTnIO1 和 PTnIO0 位得到的输出电平必须与通过 PTnOC 位设置的初始值不同，否则当比较匹配发生时，PTMn 输出脚将不会发生变化。在 PTMn 输出脚改变状态后，通过 PTnON 位由低到高电平的转换复位至初始值。

在 PWM 输出模式，PTnIO1 和 PTnIO0 用于决定比较匹配条件发生时怎样改变 PTMn 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。需注意，只可在 PTMn 关闭后才能更改 PTnIO1 和 PTnIO0 位的值。若在 PTMn 运行时改变 PTnIO1 和 PTnIO0 的值，PWM 输出的值是无法预料的。

Bit 3 **PTnOC**：PTMn PTPn 输出控制位

比较匹配输出模式
- 0：初始低
- 1：初始高

PWM 输出模式 / 单脉冲输出模式
- 0：低有效
- 1：高有效

这是 PTMn 输出脚输出控制位。它取决于 PTMn 此时正运行于比较匹配输出模式还是 PWM 输出模式 / 单脉冲输出模式。若 PTMn 处于定时 / 计数器模式，此位不起作用。在比较匹配输出模式时，其决定比较匹配发生前 PTMn 输出脚的逻辑电平值。在 PWM 输出模式时，其决定 PWM 信号是高有效还是低有效。在单脉冲输出模式，其决定 PTnON 位由低变高时 PTMn 输出脚的逻辑电平。

Bit 2 **PTnPOL**：PTMn PTPn 输出极性控制位
- 0：同相
- 1：反相

此位控制 PTPn 输出脚的极性。此位为高时 PTMn 输出脚反相，为低时 PTMn 输出脚同相。若 PTMn 处于定时 / 计数器模式时此位不起作用。

Bit 1 **PTnCAPTS**：PTMn 捕捉触发源选择位
- 0：来自 PTPnI 引脚
- 1：来自 PTCKn 引脚

Bit 0 **PTnCCLR**：PTMn 计数器清零条件选择位
0：PTMn 比较器 P 匹配
1：PTMn 比较器 A 匹配
此位用于选择清除计数器的方法。周期型 TM 包括两个比较器 – 比较器 A 和比较器 P，两者都可以用作清除内部计数器。PTnCCLR 位设为高，计数器在比较器 A 比较匹配发生时被清除；此位设为低，计数器在比较器 P 比较匹配发生或计数器溢出时被清除。计数器溢出清除的方法仅在 CCRP 被清除为 0 时才能生效。PTnCCLR 位在 PWM 输出模式、单脉冲输出模式或输入捕捉模式时未使用。

- **PTMnDL 寄存器 (n=0~1)**

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

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

- **PTMnDH 寄存器 (n=0~1)**

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

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

- **PTMnAL 寄存器 (n=0~1)**

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

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

- **PTMnAH 寄存器 (n=0~1)**

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

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

● **PTMnRPL 寄存器 (n=0~1)**

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

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

● **PTMnRPH 寄存器 (n=0~1)**

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

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

周期型 TM 工作模式

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

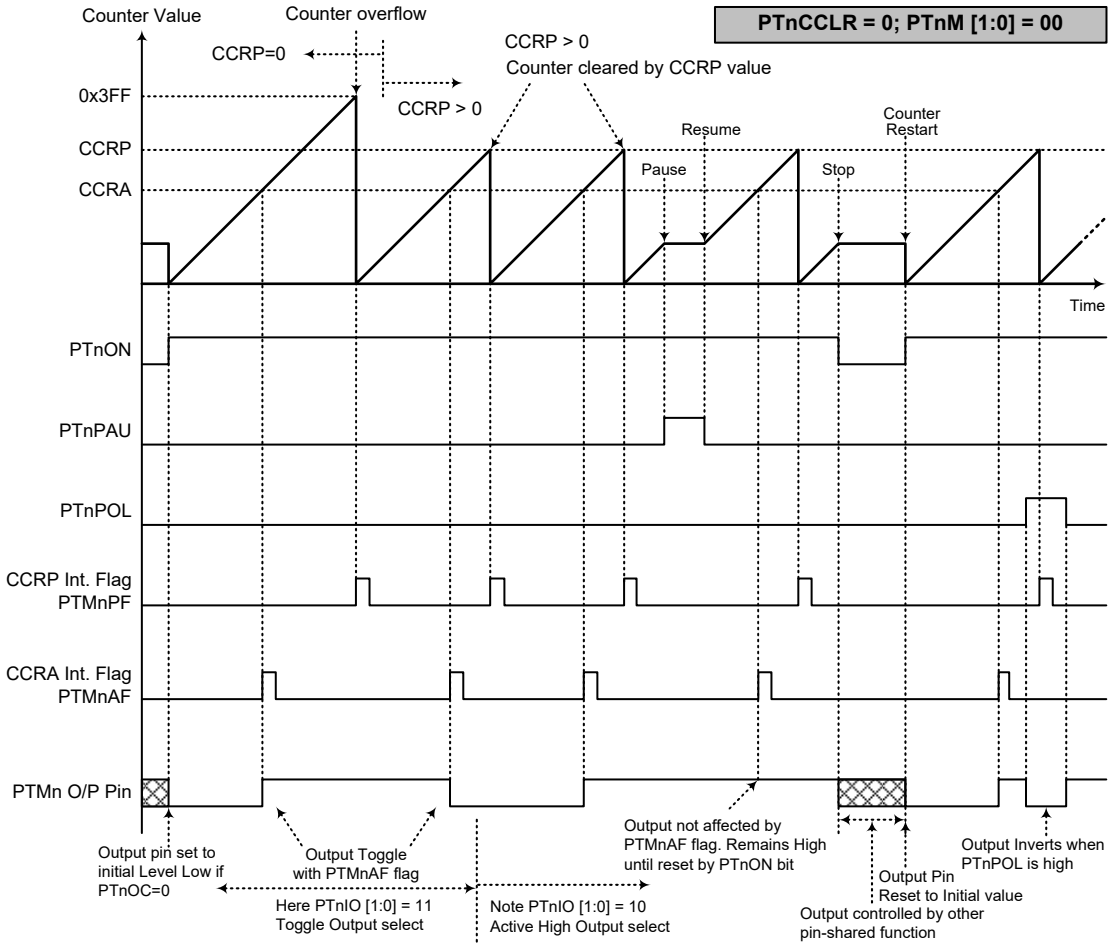
比较匹配输出模式

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

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

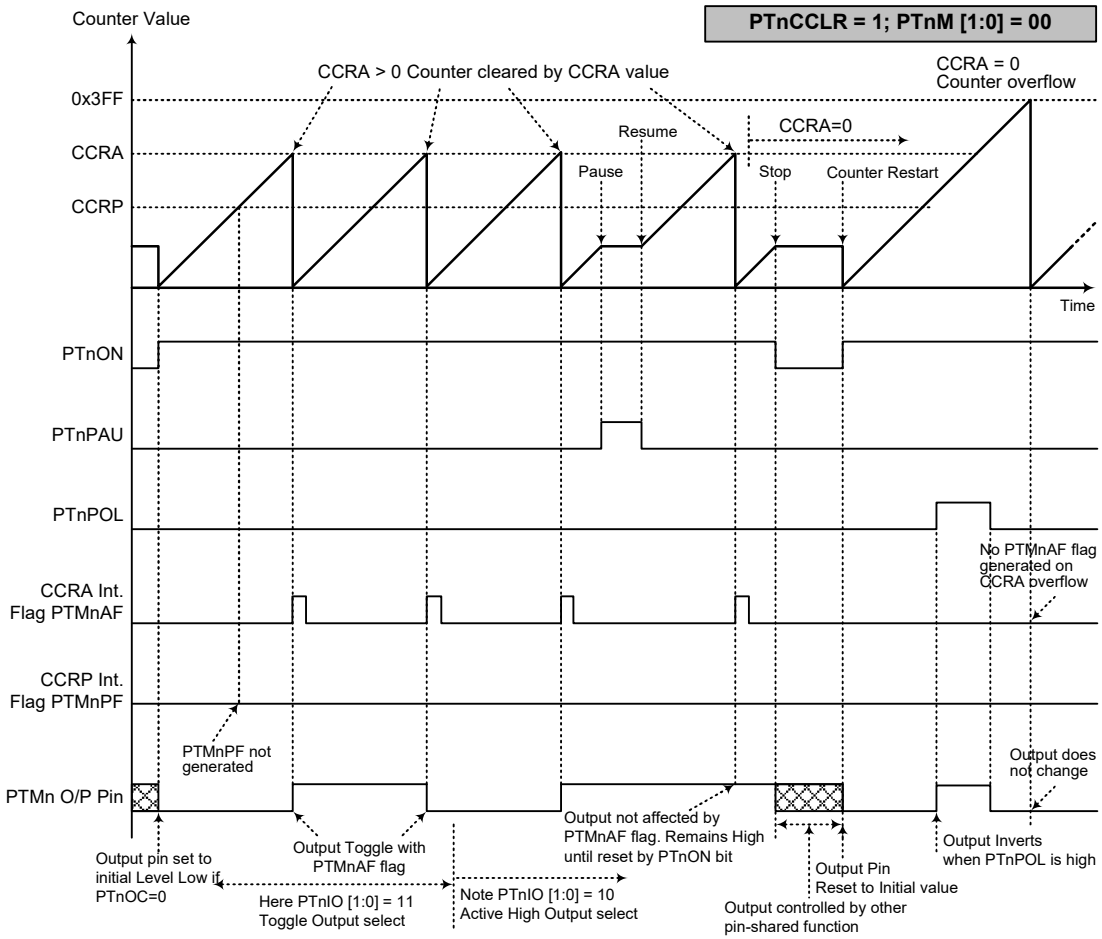
如果 CCRA 位都清除为零，当计数器的值达到 10 位最大值 3FFH 时将溢出，但此时不会产生 PTMnAF 中断请求标志。

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



比较器匹配输出模式 – PTnCCLR=0 (n=0~1)

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



比较器匹配输出模式 – PTnCCLR=1 (n=0~1)

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

定时 / 计数器模式

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

PWM 输出模式

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

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

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

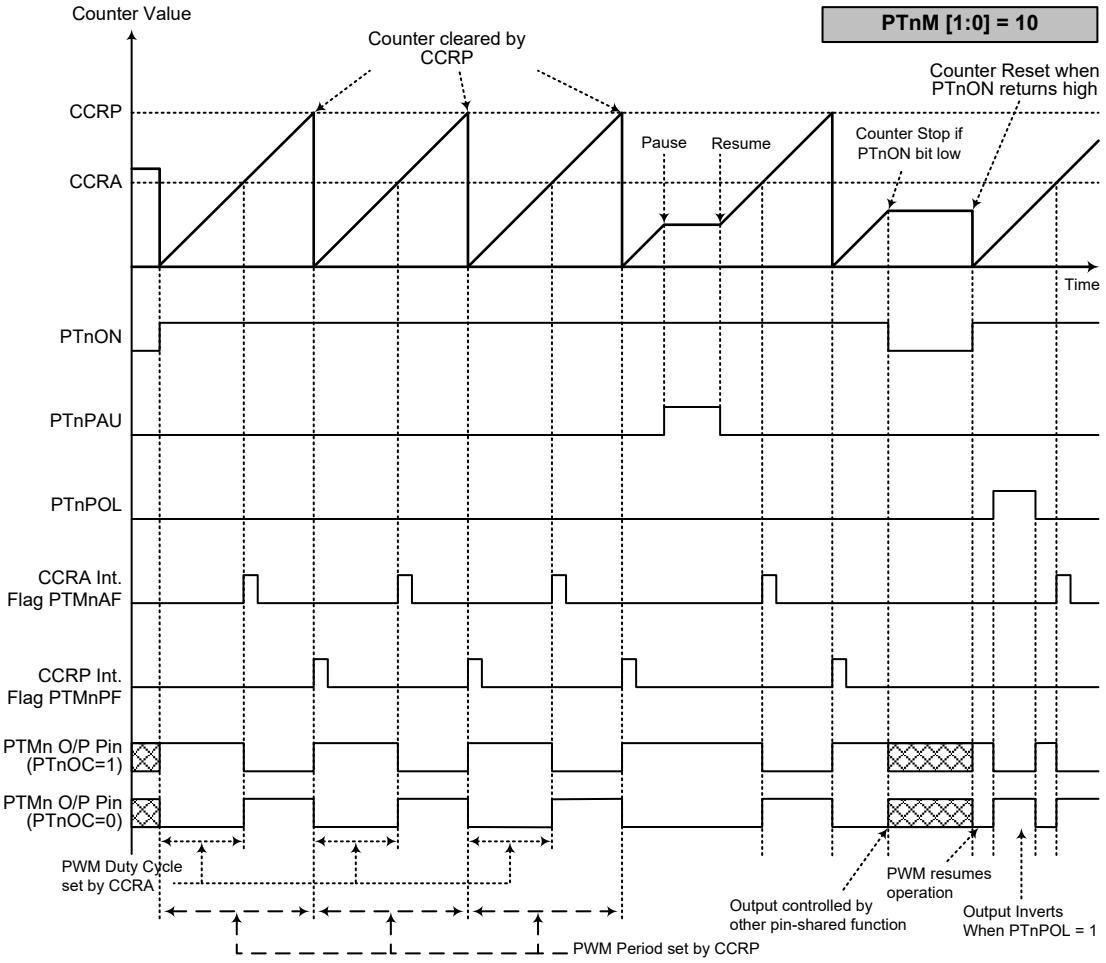
- **10-bit PTMn，PWM 输出模式，边沿对齐模式 (n=0~1)**

CCRP	1~1023	0
Period	1~1023	1024
Duty	CCRA	

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

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

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



PWM 输出模式 (n=0~1)

注：1. CCRP 清除计数器

2. 计数器清零并设置 PWM 周期

3. 当 PTnIO[1:0]=00 或 01，PWM 功能不变

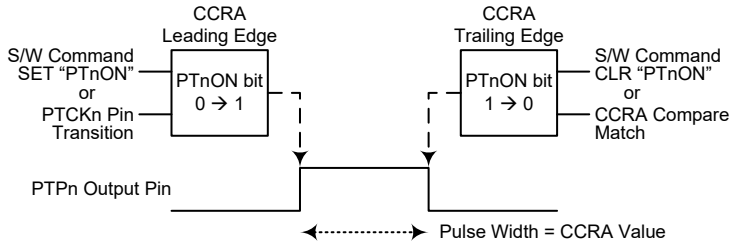
4. PTnCCLR 位对 PWM 功能无影响

单脉冲输出模式

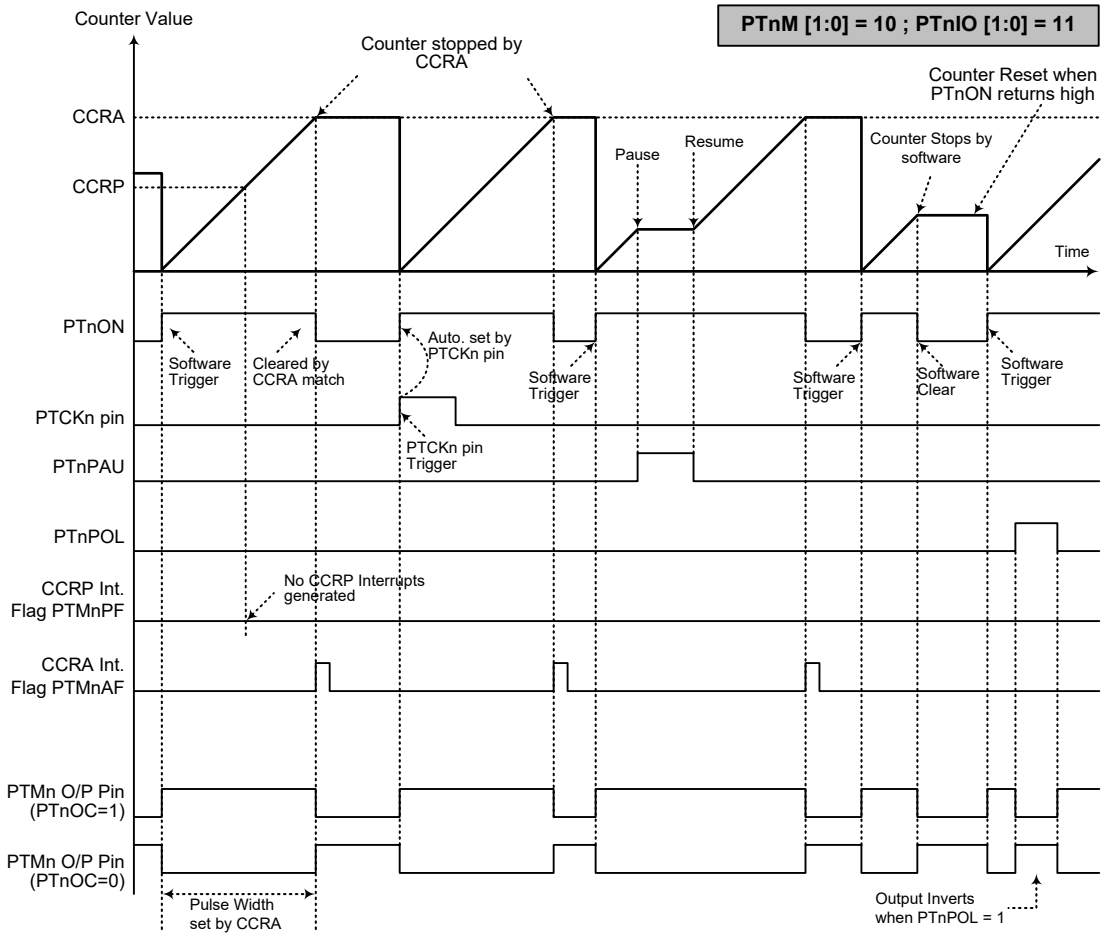
为使 PTMn 工作在此模式，PTMnC1 寄存器中的 PTnM1 和 PTnM0 位需要设置为“10”，并且相应的 PTnIO1 和 PTnIO0 需要设置为“11”。正如模式名所言，单脉冲输出模式，在 PTMn 输出脚将产生一个脉冲输出。

通过应用程序控制 PTnON 位由低到高的转变来触发脉冲前沿输出。而处于单脉冲输出模式时，PTnON 位可在 PTCKn 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 PTnON 位转变为高电平时，计数器将开始运行，并产生脉冲前沿。通过应用程序使 PTnON 位清零或比较器 A 比较匹配发生时，产生脉冲后沿。

而比较器 A 比较匹配发生时，会自动清除 PTnON 位并产生单脉冲输出边沿跳转。CCRA 的值通过这种方式控制脉冲宽度。比较器 A 比较匹配发生时，也会产生 PTMn 中断。PTnON 位在计数器重启时会发生由低到高的转变，此时计数器才复位至零。在单脉冲输出模式中，CCRP 寄存器和 PTnCCLR 位未使用。



单脉冲产生示意图 (n=0~1)



单脉冲输出模式 (n=0~1)

注：1. 通过 CCRA 匹配停止计数器
2. CCRP 未使用
3. 通过 PTCKn 脚或设置 PTnON 位为高来触发脉冲
4. PTCKn 脚有效沿会自动置位 PTnON
5. 单脉冲输出模式中，PTnIO[1:0] 需置为“11”，且不能更改

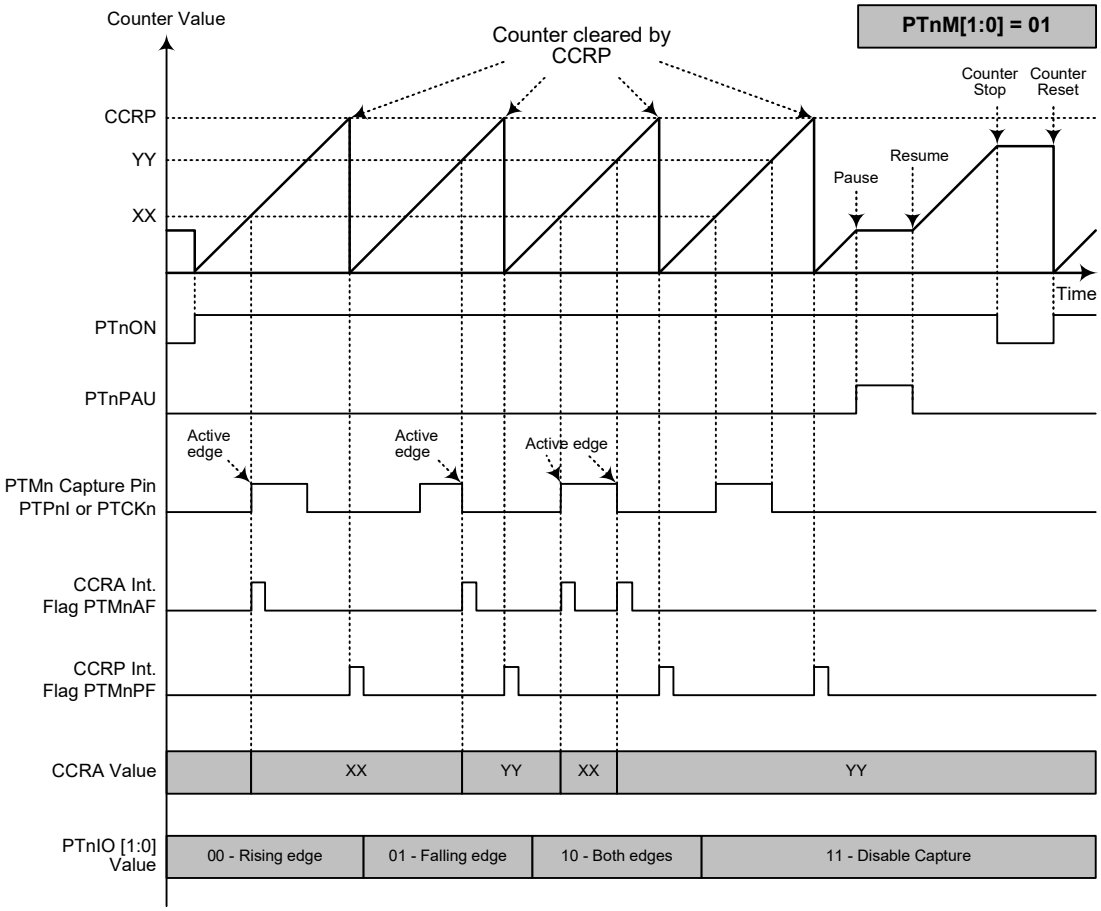
捕捉输入模式

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

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

有几点注意事项须留意。如果 PTCKn 用作捕捉输入源，则不能将其选作 PTMn 的时钟源。如果捕捉脉宽小于 2 个定时器时钟周期，则可能会被硬件忽略。当计数器的值被有效捕捉边沿锁存到 CCRA 寄存器后，再过 0.5 个定时器时钟周期，PTMnAF 标志位将被置高。从接收到有效捕捉边沿，到开始将计数器值锁存到 CCRA 寄存器的动作，这之间的延迟时间小于 1.5 个定时器时钟周期。

PTnCCLR，PTnOC 和 PTnPOL 位在此模式中未使用。

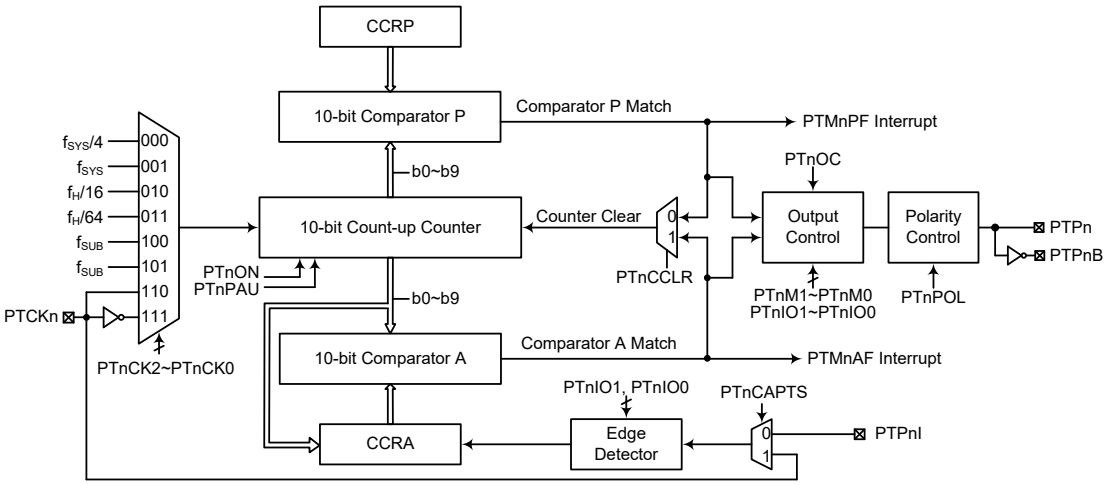


捕捉输入模式 (n=0~1)

注：1. PTnM[1:0]=01 并通过 PTnIO[1:0] 位设置有效边沿
2. PTMn 捕捉输入脚的有效边沿将计数器的值转移到 CCRA 中
3. PTnCCLR 位未使用
4. 无输出功能 – PTnOC 和 PTnPOL 位未使用
5. 计数器值由 CCRP 决定，在 CCRP 为“0”时，计数器计数值可达最大
6. 捕捉输入模式需在有 PTMn 计数时钟的情况下才可使用

音频型 TM – ATM

音频型 TM 包括 3 种工作模式，即比较匹配输出，定时 / 计数器和 PWM 输出模式。其中 PWM 输出模式又包含两种子模式，命名为正常 PWM 输出模式和音频 PWM 输出模式。音频型 TM 由一个外部输入脚控制并驱动多个外部输出脚。产生的输出可以为相同的信号，也可以为相反信号。当工作在音频 PWM 输出模式，会在 ATP_PWM1 和 ATP_PWM2 引脚产生音频信号，可经由外部 PWM 驱动器推动外部扬声器。



注：1. ATM 外部引脚与其它功能共用引脚，因此在使用 ATM 之前应该合理配置相关引脚共用功能选择寄存器以确保使能 ATM 引脚功能。对于 ATCK 引脚还需设置相应的端口控制寄存器，将该引脚设置为输入口。

2. ATP_PWM1 和 ATP_PWM2 引脚功能只有当 ATM 工作于音频 PWM 输出模式时才会使用。在其它模式下 (即比较匹配输出和正常 PWM 输出模式)，ATM 产生 ATP 和 ATPB 输出。ATPB 为 ATP 输出的反相信号。

10-bit 音频型 TM 方框图

音频型 TM 操作

音频型 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，它还包括三个内部比较器即比较器 A，比较器 B 和比较器 P。这几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 8 位宽度，与计数器的高 8 位比较；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 ATON 位发生上升沿跳变清除计数器。此外，计数器溢出或选择的比较器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 ATM 中断信号。音频型 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，也可以控制多个输出脚。所有工作模式的设定都是通过设置相关寄存器来实现的。

音频型 TM 寄存器介绍

音频型 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 位内部计数器的值，两对读 / 写寄存器存放 10 位 CCRA 和 CCRB 的值。ATMRP 寄存器用于存放 8 位 CCRP 的值。剩下两个控制寄存器设置不同的 ATM 操作和控制模式。

寄存器名称	位							
	7	6	5	4	3	2	1	0
ATMC0	ATPAU	ATCK2	ATCK1	ATCK0	ATON	—	—	ATPWM_SEL
ATMC1	ATM1	ATM0	ATIO1	ATIO0	ATOC	ATPOL	ATDPX	ATCCLR
ATMDL	D7	D6	D5	D4	D3	D2	D1	D0
ATMDH	—	—	—	—	—	—	D9	D8
ATMAL	D7	D6	D5	D4	D3	D2	D1	D0
ATMAH	—	—	—	—	—	—	D9	D8
ATMBL	D7	D6	D5	D4	D3	D2	D1	D0
ATMBH	—	—	—	—	—	—	D9	D8
ATMRP	ATRP7	ATRP6	ATRP5	ATRP4	ATRP3	ATRP2	ATRP1	ATRP0

10-bit 音频型 TM 寄存器列表

- **ATMC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ATPAU	ATCK2	ATCK1	ATCK0	ATON	—	—	ATPWM_SEL
R/W	R/W	R/W	R/W	R/W	R/W	—	—	R/W
POR	0	0	0	0	0	—	—	0

Bit 7 **ATPAU**：ATM 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，ATM 保持上电状态并继续耗电。当此位由低到高转换时，计数器将保留其当前计数值，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 **ATCK2~ATCK0**：ATM 计数器时钟选择位
000：$f_{SYS}/4$
001：f_{SYS}
010：$f_H/16$
011：$f_H/64$
100：f_{SUB}
101：f_H
110：ATCK 上升沿时钟
111：ATCK 下降沿时钟
此三位用于选择 ATM 的时钟源。外部引脚时钟源能被选择在上升沿或下降沿有效。f_{SYS} 是系统时钟，f_H 和 f_{SUB} 是其它的内部时钟源，细节方面请参考工作模式和系统时钟章节。

Bit 3 **ATON**：ATM 计数器 On/Off 控制位
0：Off
1：On
此位控制 ATM 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 ATM。清零此位将停止计数器计数并关闭 ATM 减少耗电。当此位经由高到低转换时，内部计数器将保持其当前计数值，直到此位再次改变为高电平。若 ATM 处于比较匹配输出模式时，当 ATON 位经由低到高的转换时，ATM 输出脚将复位至 ATOC 位指定的初始值。

Bit 2~1 未定义，读为“0”

Bit 0 **ATPWM_SEL**：ATM PWM 输出子模式选择
0：正常 PWM 输出模式
1：音频 PWM 输出模式
此位控制 PWM 输出子模式选择。当通过 ATMC1 寄存器中的相关位已选中 PWM 输出模式后，再将此位从低设置为高，可使 ATM 从正常 PWM 输出模式切换到音频 PWM 输出模式。

● **ATMC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ATM1	ATM0	ATIO1	ATIO0	ATOC	ATPOL	ATDPX	ATCCLR
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 **ATM1~ATM0**：ATM 工作模式选择位
00：比较匹配输出模式
01：未定义模式
10：PWM 输出模式
11：定时 / 计数器模式
这两位设置 ATM 需要的工作模式。为了确保操作可靠，ATM 应在 ATM1 和 ATM0 位有任何改变前先关掉。在定时 / 计数器模式，ATM 输出脚状态为未知。

Bit 5~4 **ATIO1~ATIO0**：ATM 外部引脚功能选择位
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
此两位用于决定在满足特定条件时 ATM 输出引脚如何改变状态。这两位值的选择取决于 ATM 运行在哪种模式下。
在比较匹配输出模式下，ATIO1 和 ATIO0 位决定当从比较器 A 比较匹配输出发生时 ATM 输出脚如何改变状态。当从比较器 A 比较匹配输出发生时 ATP 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。ATP 输出脚的初始值通过 ATMC1 寄存器的 ATOC 位设置取得。注意，由 ATIO1 和 ATIO0 位选择的输出电平必须与通过 ATOC 位设置的初始值不同，否则当比较匹配发生时，ATM 输出脚将不会发生变化。在 ATM 输出脚改变状态后，通过 ATON 位由低到高电平的转换复位至初始值。
在 PWM 输出模式，ATIO1 和 ATIO0 决定比较匹配条件发生时怎样改变 ATM 输出脚的状态。PWM 输出功能通过这两位的变化进行更新。需注意，只可在 ATM 关闭后才能更改 ATIO1 和 ATIO0 位的值。若在 ATM 运行时改变 ATIO1 和 ATIO0 的值，PWM 输出的值将无法预料。

Bit 3 **ATOC**：ATM 输出脚控制位
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 ATM 输出脚输出控制位。它取决于 ATM 此时正运行于比较匹配输出模式还是 PWM 输出模式。若 ATM 处于定时 / 计数器模式，此位不起作用。在比较匹配输出模式时，其决定比较匹配发生前 ATM 输出脚 ATP 的逻辑电平值。在 PWM 输出模式时，其决定 PWM 信号是高有效还是低有效。

Bit 2 **ATPOL**：ATM 输出极性控制位
0：同相
1：反相
此位控制 ATM 输出脚的极性。此位为高时 ATM 输出脚 ATP (正常 PWM 输出模

式) 或 ATP_PWM1 和 ATP_PWM2 (音频 PWM 输出模式) 反相，为低时输出脚同相。若 ATM 处于定时 / 计数器模式时此位不起作用。

Bit 1 **ATDPX**：ATM 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
当 ATM 工作于正常 PWM 输出模式下时，此位决定 CCRA 与 CCRP 寄存器哪个用于 PWM 波形的周期控制、哪个用于占空比控制。当通过设置 ATPWM_SEL 位为高使 ATM 工作在音频 PWM 输出模式下时，ATDPX 位会被硬件清零，即意味着 CCRP 寄存器用于 ATP_PWM1 和 ATP_PWM2 输出的 PWM 波形的周期控制。ATP_PWM1 和 ATP_PWM2 输出的 PWM 波形的占空比则分别由 CCRA 和 CCRB 寄存器值决定。

Bit 0 **ATCCLR**：选择 ATM 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。音频型 TM 包括比较器 A 和比较器 P。这两个比较器每个都可以用来清除内部计数器。ATCCLR 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。ATCCLR 位在 PWM 输出模式时未使用。

● ATMDL 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 ATM 计数器低字节寄存器 bit 7~bit 0
ATM 10-bit 计数器 bit 7~bit 0

● ATMDH 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 ATM 计数器高字节寄存器 bit 1 ~ bit 0
ATM 10-bit 计数器 bit 9 ~ bit 8

● ATMAL 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 ATM CCRA 低字节寄存器 bit 7~bit 0
ATM 10-bit CCRA bit 7~bit 0

● **ATMAH 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　ATM CCRA 高字节寄存器 bit 1 ~ bit 0
ATM 10-bit CCRA bit 9 ~ bit 8

● **ATMBL 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　ATM CCRB 低字节寄存器 bit 7~bit 0
ATM 10-bit CCRB bit 7~bit 0

● **ATMBH 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　ATM CCRB 高字节寄存器 bit 1 ~ bit 0
ATM 10-bit CCRB bit 9 ~ bit 8

● **ATMRP 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	ATRP7	ATRP6	ATRP5	ATRP4	ATRP3	ATRP2	ATRP1	ATRP0
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　**ATRP7~ATRP0**：ATM CCRP 8-bit 寄存器，与 ATM 计数器 bit 9 ~ bit 2 比较
比较器 P 匹配周期 =
0：1024 个 ATM 时钟周期
1~255：(1~255)×4 个 ATM 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 ATCCLR 位设为 0 时，此比较结果可用于清除内部计数器。由于 CCRP 只与计数器高八位比较，比较结果是 4 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。

音频型 TM 工作模式

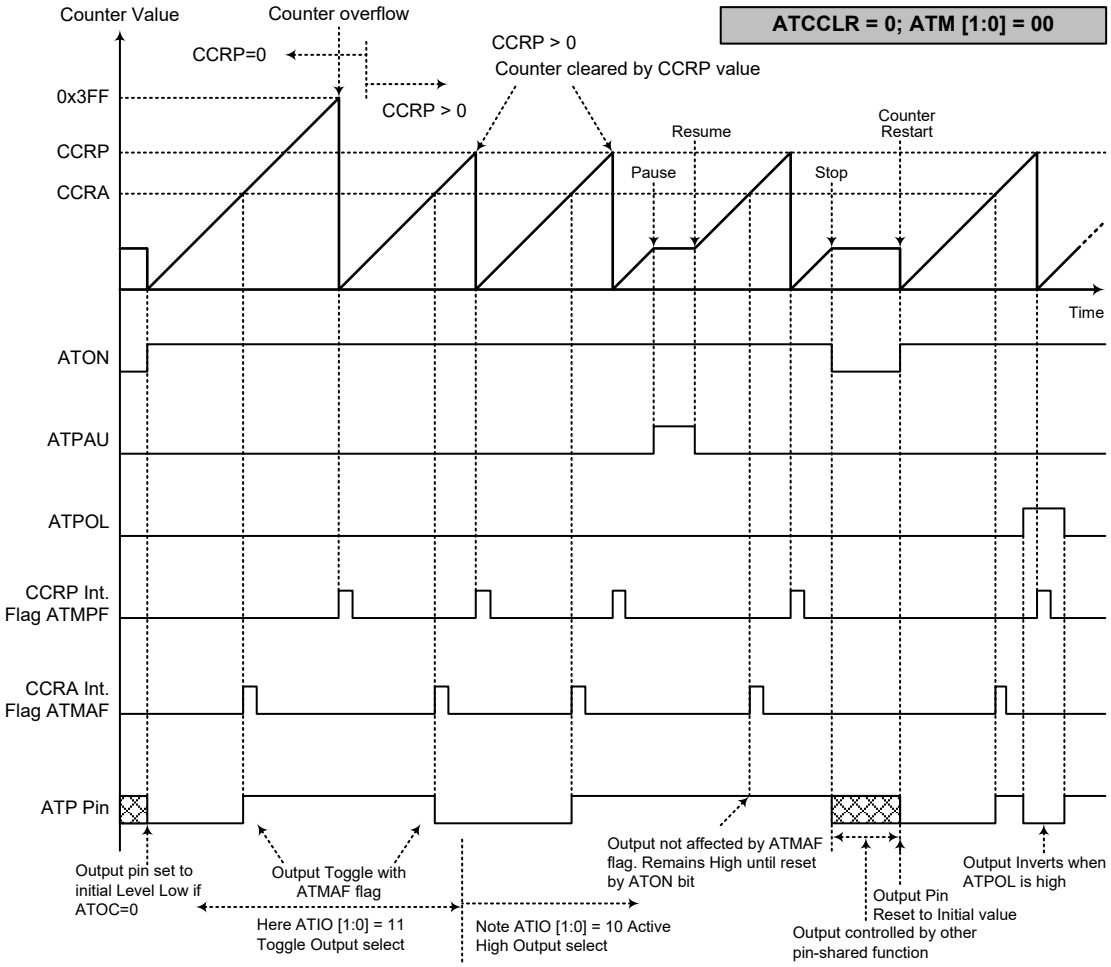
音频型 TM 有三种工作模式，即比较匹配输出模式，PWM 输出模式或定时 / 计数器模式。通过设置 ATMC1 寄存器的 ATM1 和 ATM0 位选择任意模式。

比较匹配输出模式

为使 TM 工作在此模式，ATMC1 寄存器中的 ATM1 和 ATM0 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，有三种方法可将其清零，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 ATCCLR 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 ATMAF 和 ATMPF 将分别置位。

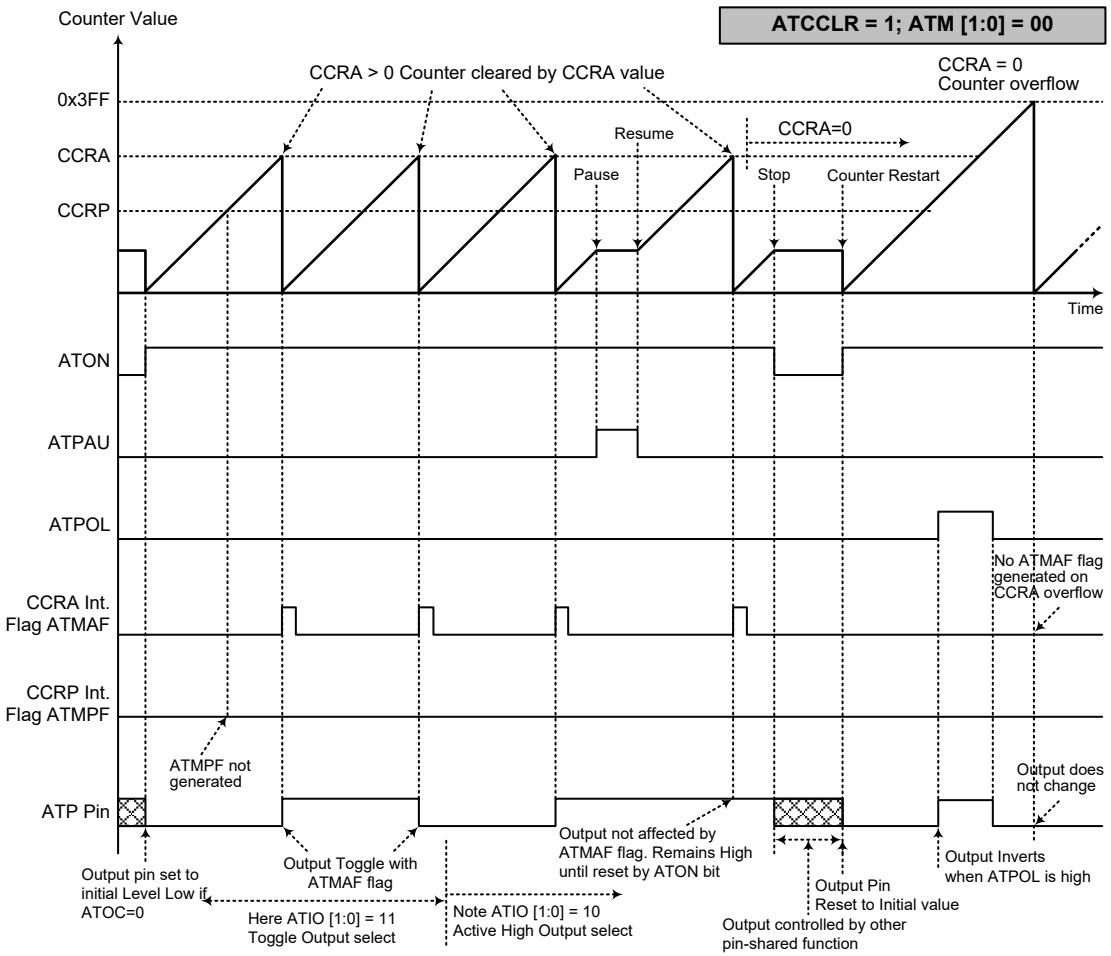
如果 ATMC1 寄存器的 ATCCLR 位设置为高，当比较器 A 比较匹配发生时计数器被清零。此时，即使 CCRP 寄存器的值小于 CCRA 寄存器的值，仅产生 ATMAF 中断请求标志。所以当 ATCCLR 为高时，不会产生 ATMPF 中断请求标志。如果 CCRA 位都清为零，当计数器的值达到 10 位最大值 3FFH 时将溢出，但此时不会产生 ATMAF 中断请求标志。

正如该模式名所言，当比较匹配发生后，ATM 输出脚状态改变。当比较器 A 比较匹配发生后 ATMAF 标志产生时，ATM 输出脚状态改变。比较器 P 比较匹配发生时产生的 ATMPF 标志不影响 ATM 输出脚。ATM 输出脚状态改变方式由 ATMC1 寄存器中 ATIO1 和 ATIO0 位决定。当比较器 A 比较匹配发生时，ATIO1 和 ATIO0 位决定 ATM 输出脚输出高，低或翻转当前状态。在 ATON 位由低到高电平的变化后，ATM 输出脚初始状态为 ATOC 位所指定的电平。注意，若 ATIO1 和 ATIO0 位同时为 0 时，引脚输出不变。



比较匹配输出模式 – **ATCCLR=0**

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



比较匹配输出模式 – **ATCCLR=1**

注：1. ATCCLR=1，比较器 A 匹配将清除计数器
2. ATM 输出脚仅由 ATMAF 标志位控制
3. 在 ATON 上升沿 TM 输出脚复位至初始值
4. 当 ATCCLR=1 时，不会产生 ATMPF 标志位

定时 / 计数器模式

为使 ATM 工作在此模式，ATMC1 寄存器中的 ATM1 和 ATM0 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 ATM 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 ATM 输出脚用作普通 I/O 脚或其它功能。

PWM 输出模式

为使 ATM 工作在此模式，ATMC1 寄存器中的 ATM1 和 ATM0 位需要设置为“10”。ATM 的 PWM 功能在扬声器驱动，马达控制，加热控制，照明控制等方面十分有用。给 ATM 输出脚提供一个频率固定但占空比可调的信号，将产生一个有效值等于 DC 均方根的 AC 方波。由于 PWM 波形的周期和占空比可调，其波形的选择就较为灵活。在 PWM 输出模式中，ATCCLR 位不影响 PWM 周期。

ATM 提供了两种 PWM 输出模式，分别为正常 PWM 输出模式和音频 PWM 输出模式，通过 ATMC0 寄存器中的 ATPWM_SEL 位选择。这两种工作模式相关控制位及差别列于下表。

项目	正常 PWM 输出模式	音频 PWM 输出模式
ATPWM_SEL 位	0	1
ATM CCRB 中断	×	√
输出引脚	ATP, ATPB	ATP_PWM1, ATP_PWM2
PWM 输出控制	ATIO1~ATIO0 位	
输出有效电平	ATOC 位	
输出极性控制	ATPOL 位	
周期 / 占空比控制	ATDPX 位	ATDPX 位固定为 0。音频 PWM 波形周期由 CCRP 控制。

正常 / 音频 PWM 输出模式比较表

正常 PWM 输出模式

在正常 PWM 输出模式，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 ATMC1 寄存器的 ATDPX 位。所以 PWM 波形由 CCRA 和 CCRP 寄存器共同决定。

若定义的 Duty 值等于或大于 Period 值，PWM 输出占空比为未知。

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。ATMC1 寄存器中的 ATOC 位决定 PWM 波形的极性，ATIO1 和 ATIO0 位使能 PWM 输出或将 ATM 输出脚置为逻辑高或逻辑低。ATPOL 位对 PWM 输出波形的极性取反。

- **10-bit ATM，正常 PWM 输出模式，边沿对齐模式，ATDPX=0**

CCRP	1~255	0
Period	CCRP×4	1024
Duty	CCRA	

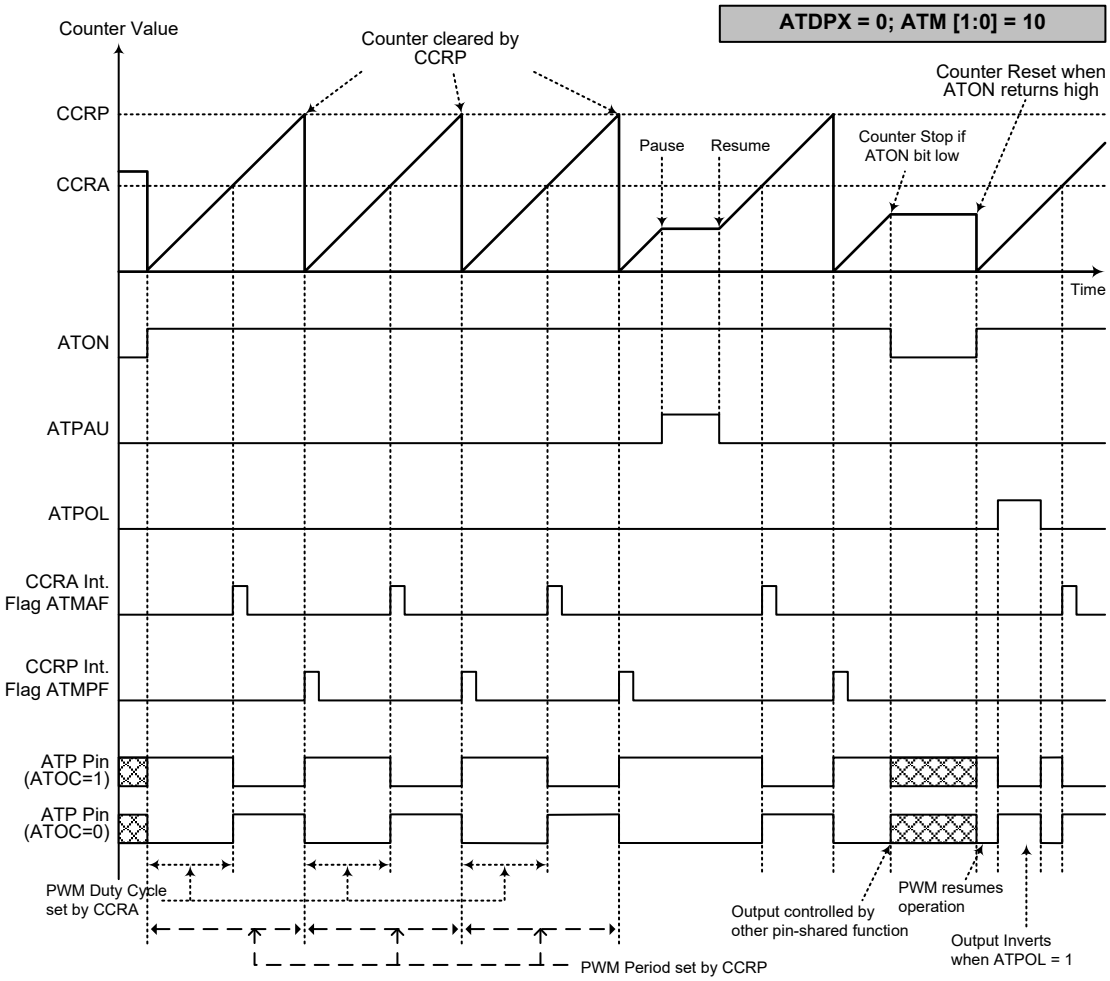
若 f_{SYS}=8MHz，ATM 时钟源为 f_{SYS}/4，CCRP=64，CCRA=64，

ATM PWM 输出频率 = (f_{SYS}/4)/(64×4)=f_{SYS}/1024=7.8125kHz，duty=64/(64×4)=25%。

● **10-bit ATM，正常 PWM 输出模式，边沿对齐模式，ATDPX=1**

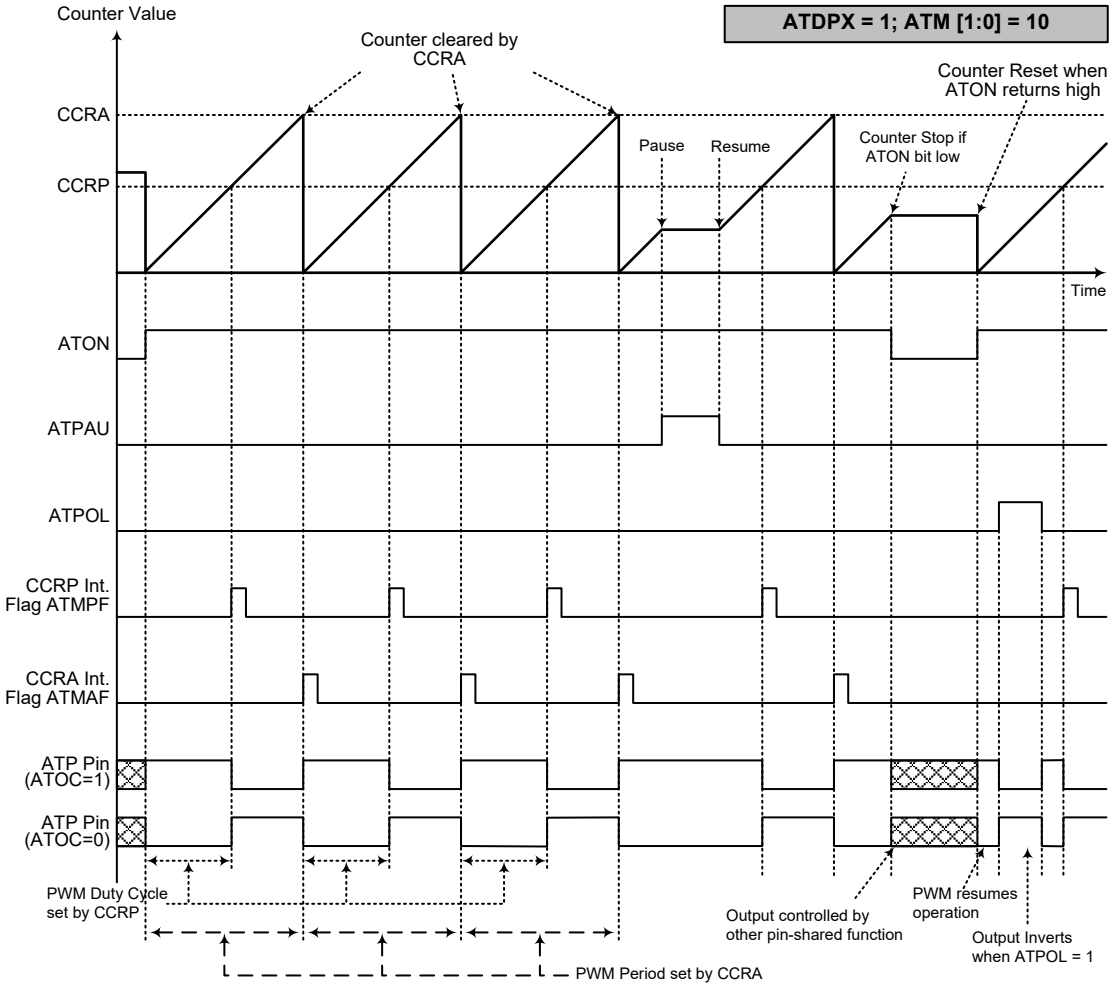
CCRP	1~255	0
Period	CCRA	
Duty	CCRP×4	1024

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



ATM 正常 PWM 输出模式 – ATDPX=0

注：1. ATM[1:0]=10，ATPWM_SEL=0，ATDPX=0
2. CCRP 清除计数器
3. 计数器清零并设置 PWM 周期
4. 当 ATIO[1:0]=00 或 01，PWM 功能不变
5. ATCCLR 位不影响 PWM 操作



ATM 正常 PWM 输出模式 – ATDPX=1

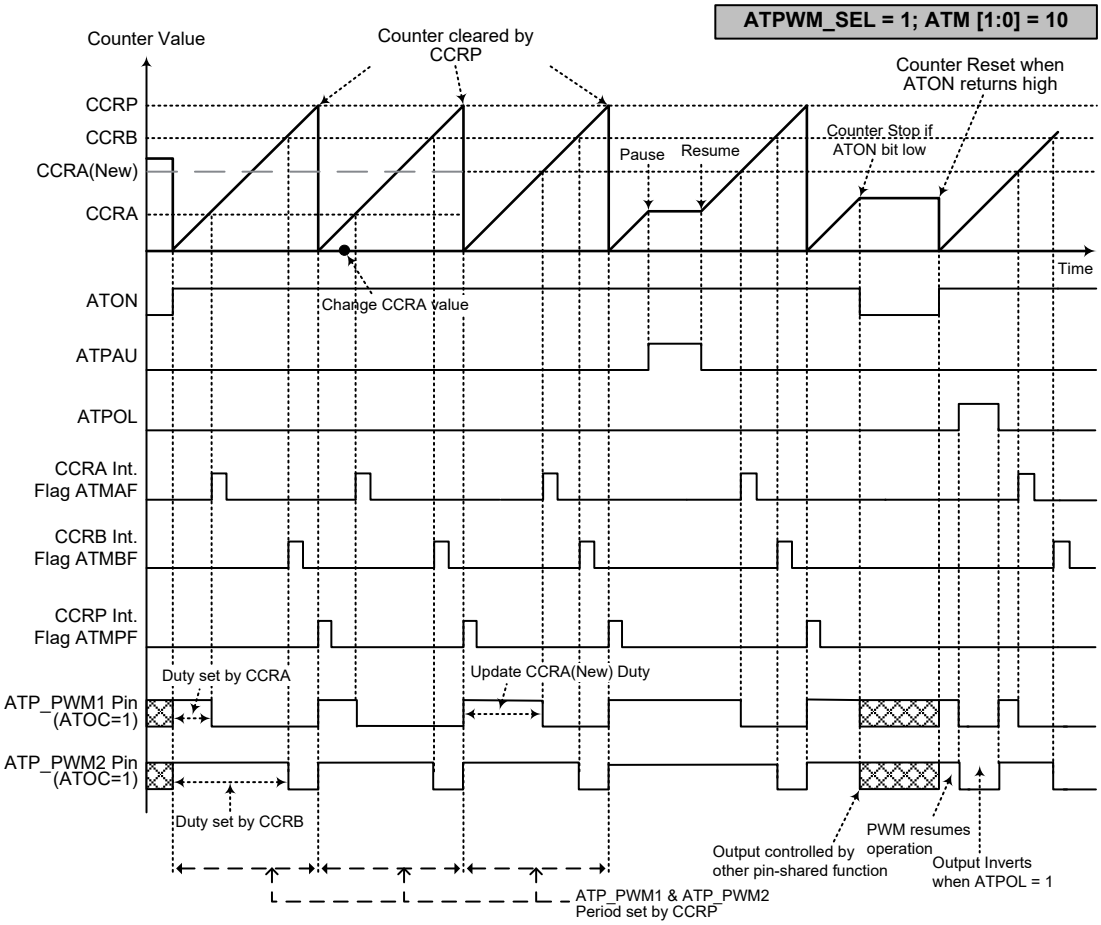
注：1. ATM[1:0]=10，ATPWM_SEL=0，ATDPX=1
2. CCRA 清除计数器
3. 计数器清零并设置 PWM 周期
4. 当 ATIO[1:0]=00 或 01，PWM 功能不变
5. ATCCLR 位不影响 PWM 操作

音频 PWM 输出模式

在音频 PWM 输出模式，CCRA、CCRB 和 CCRP 寄存器用于控制 ATP_PWM1 和 ATP_PWM2 输出的 PWM 波形。CCRP 寄存器用来清除内部计数器从而控制 ATP_PWM1 和 ATP_PWM2 这两路 PWM 波形的频率，CCRA 和 CCRB 寄存器则分别控制这两路 PWM 波形占空比。

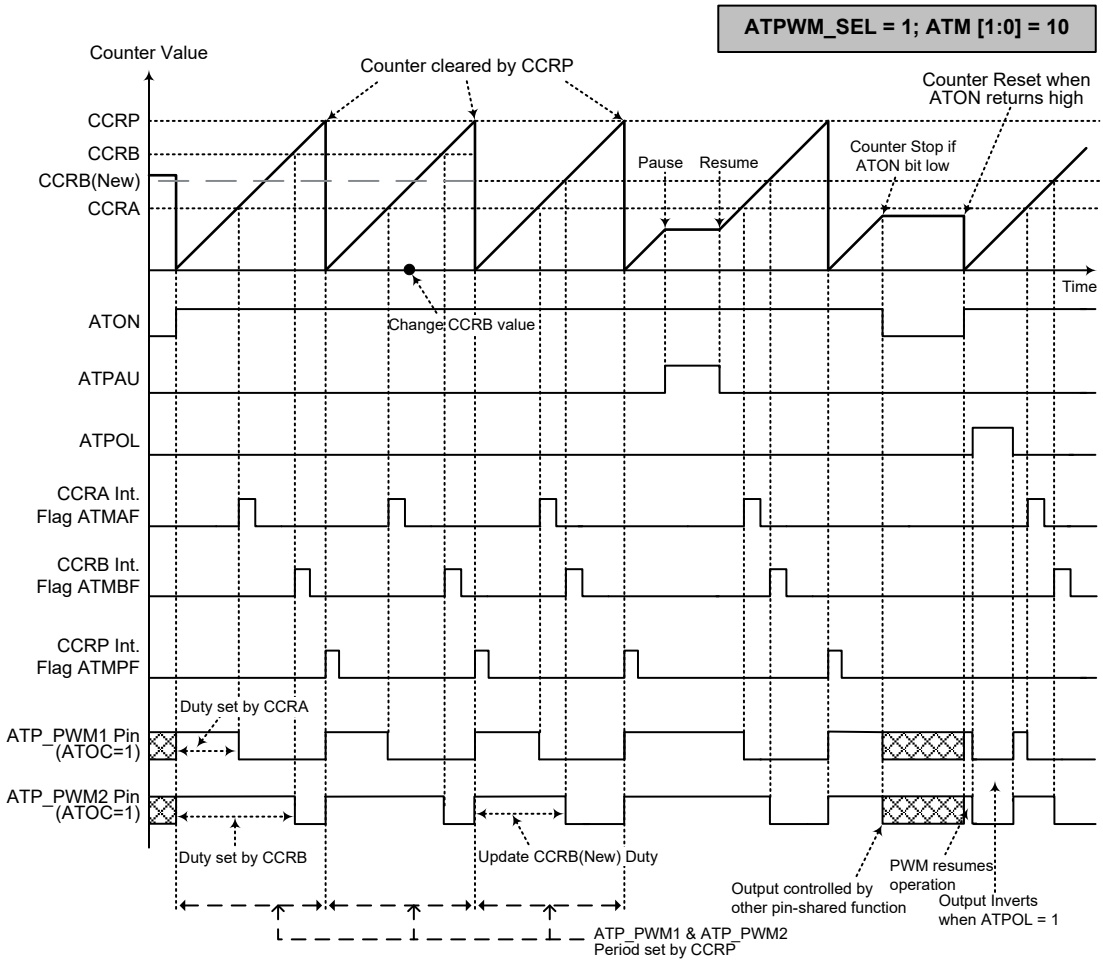
当任一比较器 A、比较器 B 或比较器 P 比较匹配发生时，将产生 CCRA、CCRB 或 CCRP 中断标志。在音频 PWM 输出模式，除了 ATDPX 位由硬件固定为 0，其它与 ATP_PWM1 和 ATP_PWM2 相关的控制位都与正常 PWM 输出模式时相同。

需注意的是，在音频 PWM 输出模式，CCRP、CCRA 和 CCRB 都有影子寄存器，在一个 PWM 周期内对这些寄存器的值进行修改，并不会立即更新此值到实际寄存器，而是在当前 PWM 周期结束即 CCRP 计数器溢出产生对应中断请求后，才会更新。



ATM 音频 PWM 输出模式 – 改变 CCRA 值

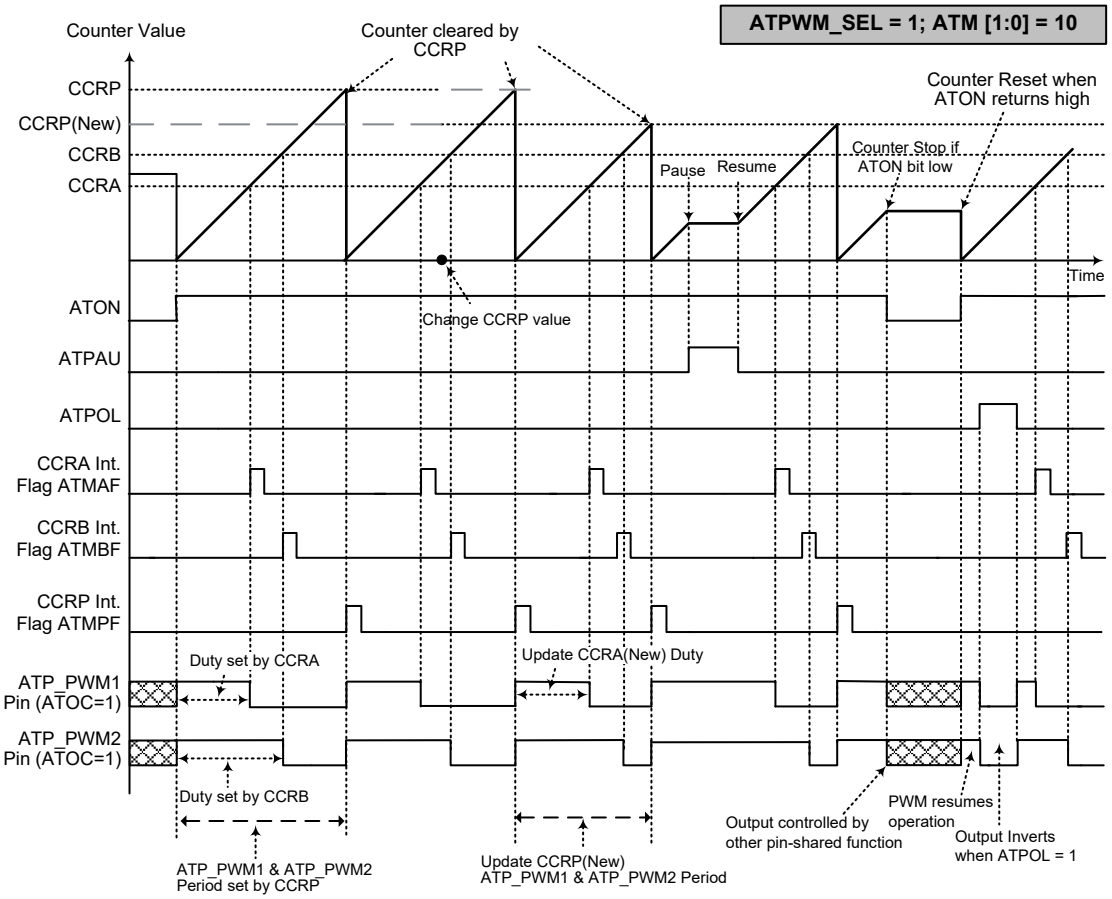
注：1. ATM[1:0]=10，ATPWM_SEL=1
2. CCRP 清除计数器
3. 计数器清零并设置 PWM 周期
4. CCRA 设置 ATP_PWM1 占空比，CCRB 设置 ATP_PWM2 占空比
5. 当 ATIO[1:0]=00 或 01，PWM 功能不变
6. ATCCLR 位不影响 PWM 操作，ATDPX 位由硬件固定为 0



ATM 音频 PWM 输出模式 – 改变 CCRB 值

注：1. ATM[1:0]=10，ATPWM_SEL=1

2. CCRP 清除计数器
3. 计数器清零并设置 PWM 周期
4. CCRA 设置 ATP_PWM1 占空比，CCRB 设置 ATP_PWM2 占空比
5. 当 ATIO[1:0]=00 或 01，PWM 功能不变
6. ATCCLR 位不影响 PWM 操作，ATDPX 位由硬件固定为 0

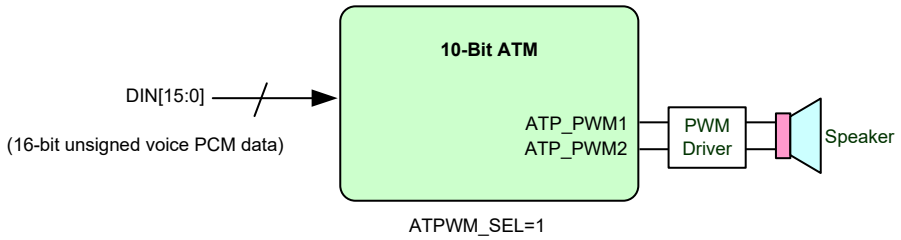


ATM 音频 PWM 输出模式 – 改变 CCRP 值

注：1. ATM[1:0]=10，ATPWM_SEL=1
2. CCRP 清除计数器
3. 计数器清零并设置 PWM 周期
4. CCRA 设置 ATP_PWM1 占空比，CCRB 设置 ATP_PWM2 占空比
5. 当 ATIO[1:0]=00 或 01，PWM 功能不变
6. ATCCLR 位不影响 PWM 操作，ATDPX 位由硬件固定为 0

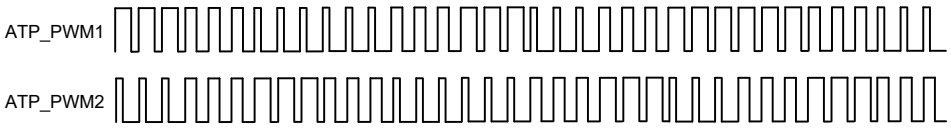
音频 PWM 输出使用介绍

ATM 设置于音频 PWM 输出模式，可以应用在把经过解压缩的 PCM 语音数据，输出为音频 PWM 波形经由外部 PWM 驱动器推动扬声器。以下的例子为如何处理 16 位无符号 PCM 语音数据，并填入正确的值到 CCRP、CCRA 和 CCRB 寄存器，以产生对应的音频 PWM 输出。

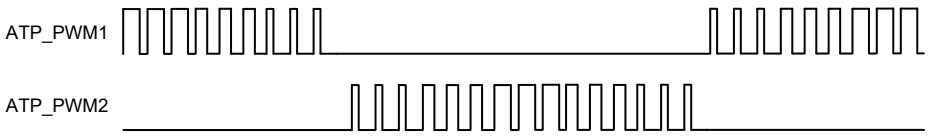


音频 PWM 输出应用示意图

音频 PWM 有两种输出模式，一种称为差分模式，一种为单端模式。波形如下图所示。



差分 PWM 模式



单端 PWM 模式

以下为一个实际的范例，说明如何搭配 ATM 产生差分 / 单端两种模式下的音频 PWM 输出。在该范例中选择的 ATM 计数器时钟为系统时钟，系统时钟来自内部 8MHz 高速时钟，要产生对应音频 PWM 输出信号，CCRA、CCRB 及 CCRP 的设置以及相关寄存器设置如下所示。

- 步骤 1 – 设置 ATMC0 寄存器

  设置 ATMC0 寄存器的 ATCK2~ATCK0 位为 001 或 101，选择 ATM 计数器时钟来自 f_{SYS} 或 f_H。设置 ATPWM_SEL 为 1 选择音频 PWM 输出模式。即在此步骤中将 ATMC0 寄存器值设为 00010--1B。

- 步骤 2 – 设置 ATMC1 寄存器

  设置 ATMC1 寄存器中的 ATM1~ATM0 位为 10，ATIO1~ATIO0 位为 10，以选择 ATM PWM 输出模式。然后设置 ATOC 位为 1，ATPOL 位为 0 以设置 PWM 输出为高有效且输出同相。即在此步骤中将 ATMC1 寄存器的值设为 10101000B。

- 步骤 3 – 设置 ATMRP 寄存器

  设置 ATMRP 寄存器值为 01000000B，选择比较器 P 匹配周期为 256 个 ATM 时钟。则音频 PWM 频率为：(f_{SYS})/256=31.25kHz。当然若设置 ATCK2~ATCK0 位为 000，则对应的音频 PWM 频率为：(f_{SYS}/4)/256=7.8125kHz。

- 步骤 4 – 引脚共用功能选择

  依需求合理设置 PAS1 寄存器中的 PAS13~PAS10 位，或设置 PDS1 寄存器中的 PDS13~PDS10 位以使能对应的 ATP_PWM1 和 ATP_PWM2 引脚功能。

- 步骤 5 – 使能 ATM 功能

  完成其它 ATM 的相关设置，比如中断等，然后设置 ATMC0 寄存器中的 ATON 位为 1，使 ATM 计数器开始计数。

- 步骤 6 – 更新 CCRA 和 CCRB 寄存器值

  依照输入数据，计算并更新 CCRA 和 CCRB 寄存器值。CCRA 和 CCRB 值计算方法可参考之后的内容。

因有两种类型的音频 PWM 输出波形，差分模式和单端输出模式。下方的范例及计算方法将基于这两种波形类型分别说明。

范例 1 – 音频 PWM 输出差分模式，DIN[15:0]=4000H

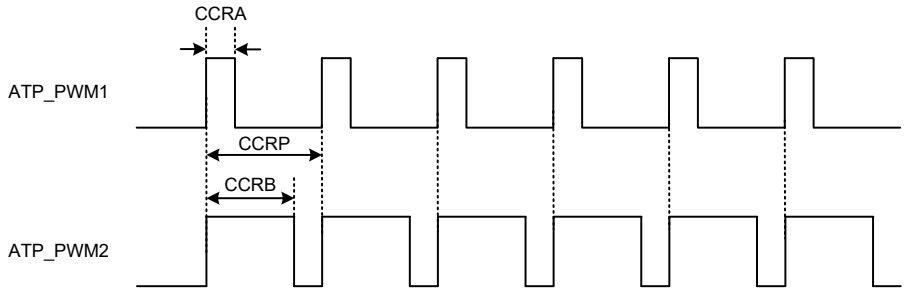
例如 DIN[15:0]=4000H (无符号语音数据)，若 ATP_PWM1 和 ATP_PWM2 产生如下所示的差分 PWM 输出，CCRA/CCRB 计算方法如下：

DIN[15:0]=4000H，DINB[15:0]=~DIN[15:0]=BFFFH

取高八位值：DIN[15:8]=40H=64，DINB[15:8]=BFH=191

ATP_PWM1：8-bit DIN[15:8]=40H=64=CCRA，Duty=64/256≈25%

ATP_PWM2：8-bit DINB[15:8]=BFH=191=CCRB，Duty=191/256≈75%



范例 2 – 音频 PWM 输出差分模式，DIN[15:0]=C000H

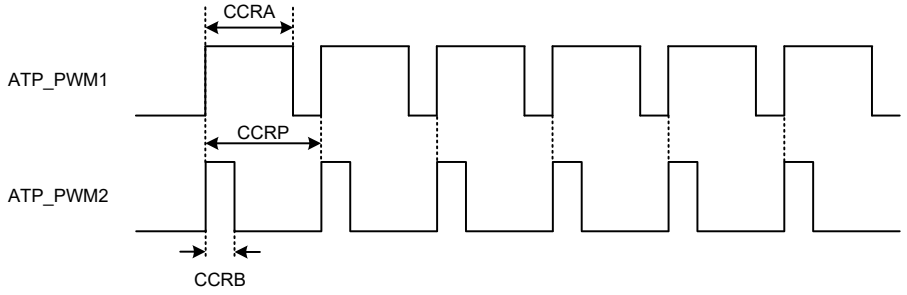
例如 DIN[15:0]=C000H (无符号语音数据)，若 ATP_PWM1 和 ATP_PWM2 产生如下所示的差分 PWM 输出，CCRA/CCRB 计算方法如下：

DIN[15:0]=C000H，DINB[15:0]=~DIN[15:0]=3FFFH

取高八位值：DIN[15:8]=C0H=192，DINB[15:8]=3FH=63

ATP_PWM1：8-bit DIN[15:8]=C0H=192=CCRA，Duty=192/256≈75%

ATP_PWM2：8-bit DINB[15:8]=3FH=63=CCRB，Duty=63/256≈25%



通过以上范例 1 ~ 范例 2，可归纳出由 DIN[15:0] 无符号语音数据产生差分 PWM 输出，CCRA/CCRB 寄存器值的计算公式：

CCRA=DIN[15:8]

CCRB=DINB[15:8]=~DIN[15:8]

下表列出了要产生差分 PWM 输出，ATP_PWM1 和 ATP_PWM2 输出信号占空比和对应 CCRA 和 CCRB 的取值。

DIN[15:0] (无符号语音数据)	ATP_PWM1 输出	ATP_PWM2 输出	CCRA/CCRB 设定值
(00H, 00H)	Pulse Duty≈0%	Pulse Duty≈100%	CCRA=DIN[15:8]=00H CCRB=~DIN[15:8]=FFH
(20H, 00H)	Pulse Duty≈12.5%	Pulse Duty≈87.5%	CCRA=DIN[15:8]=20H CCRB=~DIN[15:8]=DFH
(40H, 00H)	Pulse Duty≈25%	Pulse Duty≈75%	CCRA=DIN[15:8]=40H CCRB=~DIN[15:8]=BFH
(60H, 00H)	Pulse Duty≈37.5%	Pulse Duty≈62.5%	CCRA=DIN[15:8]=60H CCRB=~DIN[15:8]=9FH
(80H, 00H)	Pulse Duty≈50%	Pulse Duty≈50%	CCRA=DIN[15:8]=80H CCRB=~DIN[15:8]=7FH
(A0H, 00H)	Pulse Duty≈62.5%	Pulse Duty≈37.5%	CCRA=DIN[15:8]=A0H CCRB=~DIN[15:8]=5FH
(C0H, 00H)	Pulse Duty≈75%	Pulse Duty≈25%	CCRA=DIN[15:8]=C0H CCRB=~DIN[15:8]=3FH
(E0H, 00H)	Pulse Duty≈87.5%	Pulse Duty≈12.5%	CCRA=DIN[15:8]=E0H CCRB=~DIN[15:8]=1FH
(FFH, FFH)	Pulse Duty≈100%	Pulse Duty≈0%	CCRA=DIN[15:8]=FFH CCRB=~DIN[15:8]=00H

CCRA/CCRB 值 – 音频 PWM 输出差分模式

范例 3 – 音频 PWM 输出单端模式，DIN[15:0]=4000H

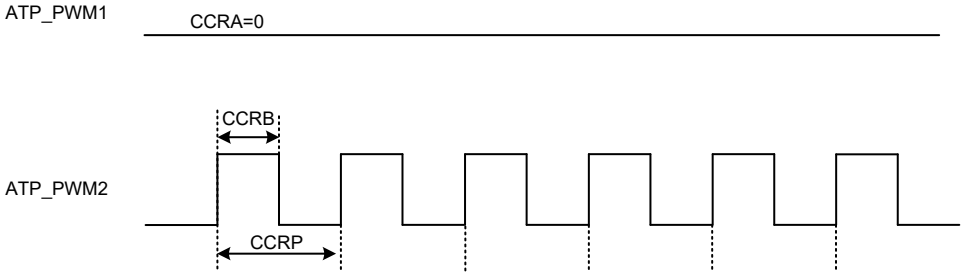
例如 DIN[15:0]=4000H (无符号语音数据)，若 ATP_PWM1 和 ATP_PWM2 产生如下所示的单端 PWM 输出，CCRA/CCRB 计算方法如下：

DIN[15:0]=4000H，DINB[15:0]=~DIN[15:0]=BFFFH

取 DINB 的 bit 14~bit 7：DINB[14:7]=7FH=127

ATP_PWM1：因 DIN[15]=0，则 CCRA=0，Duty=0/256≈0%

ATP_PWM2：DINB[14:7]=7FH=127=CCRB，Duty=127/256≈50%



范例 4 – 音频 PWM 输出单端模式，DIN[15:0]=C000H

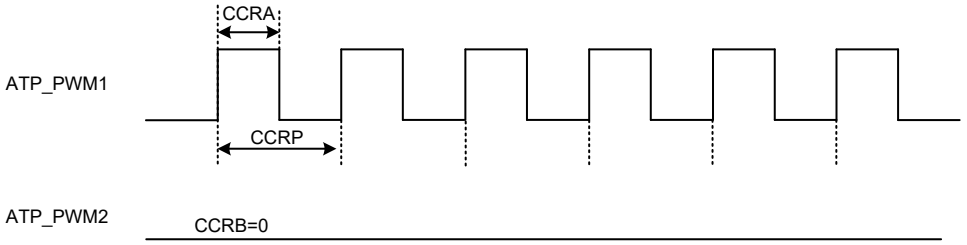
例如 DIN[15:0]=C000H (无符号语音数据)，若 ATP_PWM1 和 ATP_PWM2 产生如下所示的单端 PWM 输出，CCRA/CCRB 计算方法如下：

DIN[15:0]=C000H，DINB[15:0]=~DIN[15:0]=3FFFH

取 DIN 的 bit 14~bit 7：DIN[14:7]=80H=128

ATP_PWM1：DIN[14:7]=80H=128=CCRA，Duty=128/256≈50%

ATP_PWM2：因 DIN[15]=1，则 CCRB=0，Duty=0/256≈0%



通过以上范例 3~ 范例 4，可归纳出由 DIN[15:0] 无符号语音数据产生单端 PWM 输出，CCRA/CCRB 寄存器值的计算公式：

- 若 DIN[15]=0，CCRA=0 (ATP_PWM1 输出 0)，CCRB=~DIN[14:7]
- 若 DIN[15]=1，CCRA=DIN[14:7]，CCRB=0 (ATP_PWM2 输出 0)

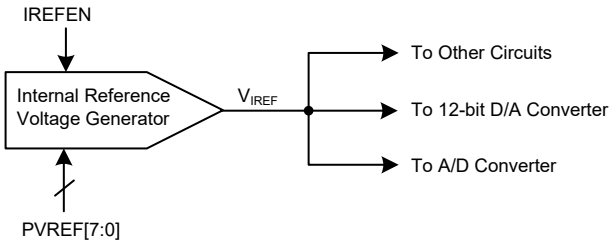
下表列出了要产生单端 PWM 输出，ATP_PWM1 和 ATP_PWM2 输出信号占空比和对应 CCRA 和 CCRB 的取值。

DIN[15:0] (无符号语音数据)	ATP_PWM1 输出	ATP_PWM2 输出	CCRA/CCRB 设定值
(00H, 00H)	0	Pulse Duty≈100%	CCRA=00H CCRB=~DIN[14:7]=FFH
(20H, 00H)	0	Pulse Duty≈75%	CCRA=00H CCRB=~DIN[14:7]=BFH
(40H, 00H)	0	Pulse Duty≈50%	CCRA=00H CCRB=~DIN[14:7]=7FH
(60H, 00H)	0	Pulse Duty≈25%	CCRA=00H CCRB=~DIN[14:7]=3FH
(80H, 00H)	0	0	CCRA=DIN[14:7]=00H CCRB=00H
(A0H, 00H)	Pulse Duty≈25%	0	CCRA=DIN[14:7]=40H CCRB=00H
(C0H, 00H)	Pulse Duty≈50%	0	CCRA=DIN[14:7]=80H CCRB=00H
(E0H, 00H)	Pulse Duty≈75%	0	CCRA=DIN[14:7]=C0H CCRB=00H
(FFH, FFH)	Pulse Duty≈100%	0	CCRA=DIN[14:7]=FFH CCRB=00H

CCRA/CCRB 值 – 音频 PWM 输出单端模式

内部参考电压发生器

该单片机包含一个内部参考电压发生器，以提供精准的参考电压 VIREF 用于 A/D 转换器或 12-bit D/A 转换器，或通过 DACVREF 引脚输出提供给其它电路使用，详细请参阅内部参考电压特性章节。



内部参考电压寄存器介绍

内部参考电压发生器由两个寄存器控制。IREFC 寄存器用于使能 / 除能控制，PVREF 寄存器用于对参考电压进行微调。

寄存器名称	位							
	7	6	5	4	3	2	1	0
IREFC	OP2DO	OP1DO	—	IREFEN	BATDEN	—	DACVRS1	DACVRS0
PVREF	D7	D6	D5	D4	D3	D2	D1	D0

内部参考电压寄存器列表

- **IREFC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OP2DO	OP1DO	—	IREFEN	BATDEN	—	DACVRS1	DACVRS0
R/W	R	R	—	R/W	R/W	—	R/W	R/W
POR	0	0	—	0	0	—	0	0

Bit 7 **OP2DO**：OPA2 数字输出；逻辑正
具体描述见运算放大器章节。

Bit 6 **OP1DO**：OPA1 数字输出；逻辑正
具体描述见运算放大器章节。

Bit 5 未定义，读为“0”

Bit 4 **IREFEN**：内部参考电压发生器使能控制
0：除能
1：使能
此位用于控制内部参考电压发生器以提供 V_{IREF} 电压给 A/D 转换器或 12-bit D/A 转换器使用。当此位置 1 时，内部参考电压 V_{IREF} 可用作参考电压。若此参考电压不提供给任何电路使用，应将此位清零以节省功耗。

Bit 3 **BATDEN**：电池电压检测电路使能控制
0：除能
1：使能
此位用于使能或除能内部电池电压检测电路。当此位置高，分压电路开启并产生 $V_{DD}/4$ 电压输出用于测量。

Bit 2 未定义，读为“0”

Bit 1~0 **DACVRS1~DACVRS0**：12-bit D/A 转换器参考电压 $V_{DACVREF}$ 选择
具体描述见 D/A 转换器章节。

● **PVREF 寄存器**

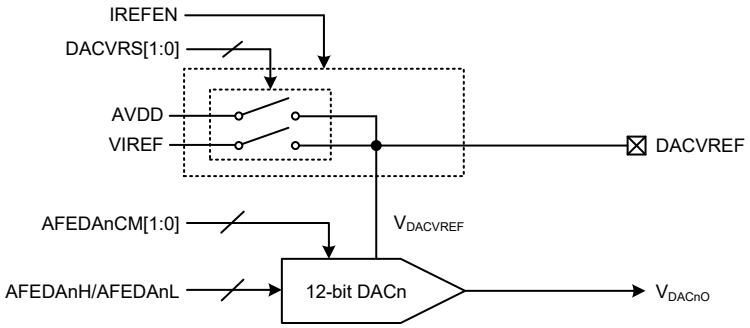
Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**：内部参考电压发生器微调控制

通过设置此寄存器可以对内部参考电压进行微调，调整幅度为 -60mV~+60mV (基于 PVREF=80H)。PVREF 寄存器的值每增加一，输出的参考电压将减少约 500μV；反之，每减少一，将增加约 500μV。

D/A 转换器 – DAC

此单片机内置两个 12-bit D/A 转换器，可提供 0~$V_{DACVREF}$ 的电压给 OPA 正输入端使用。D/A 参考电压 $V_{DACVREF}$ 还可用作 A/D 转换器的参考电压。



D/A 转换器结构 (n=1 或 2)

D/A 转换器寄存器介绍

D/A 转换器所有功能由多个寄存器控制。IREFC 寄存器中的 DACVRS1~DACVRS0 可用于选择 D/A 转换器参考电压，AFEDAC 寄存器用于 D/A 转换器 n 的使能 / 除能控制，剩下的寄存器为两对 12-bit 寄存器 AFEDAnH 和 AFEDAnL 用于 D/A 转换器 n 输出控制。

寄存器名称	位							
	7	6	5	4	3	2	1	0
IREFC	OP2DO	OP1DO	—	IREFEN	BATDEN	—	DACVRS1	DACVRS0
AFEDAC	DAC1S1	DAC1S0	DAC2S1	DAC2S0	AFEDAC2M1	AFEDAC2M0	AFEDAC1M1	AFEDAC1M0
AFEDA1L	D3	D2	D1	D0	—	—	—	—
AFEDA1H	D11	D10	D9	D8	D7	D6	D5	D4
AFEDA2L	D3	D2	D1	D0	—	—	—	—
AFEDA2H	D11	D10	D9	D8	D7	D6	D5	D4

D/A 转换器寄存器列表

● **IREFC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OP2DO	OP1DO	—	IREFEN	BATDEN	—	DACVRS1	DACVRS0
R/W	R	R	—	R/W	R/W	—	R/W	R/W
POR	0	0	—	0	0	—	0	0

Bit 7 **OP2DO**：OPA2 数字输出；逻辑正
具体描述见运算放大器章节。

Bit 6 **OP1DO**：OPA1 数字输出；逻辑正
具体描述见运算放大器章节。

Bit 5 未定义，读为“0”

Bit 4 **IREFEN**：内部参考电压发生器使能控制
具体描述见内部参考电压发生器章节。

Bit 3 **BATDEN**：电池电压检测电路使能控制
具体描述见内部参考电压发生器章节。

Bit 2 未定义，读为“0”

Bit 1~0 **DACVRS1~DACVRS0**：12-bit D/A 转换器参考电压 $V_{DACVREF}$ 选择
00/01：V_{IREF}
10：AV_{DD}
11：高阻抗 (浮空)

● **AFEDAC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	DAC1S1	DAC1S0	DAC2S1	DAC2S0	AFEDAC2M1	AFEDAC2M0	AFEDAC1M1	AFEDAC1M0
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 **DAC1S1**：DAC1S1 开关控制
0：Off
1：On

Bit 6 **DAC1S0**：DAC1S0 开关控制
0：Off
1：On

Bit 5 **DAC2S1**：DAC2S1 开关控制
0：Off
1：On

Bit 4 **DAC2S0**：DAC2S0 开关控制
0：Off
1：On

Bit 3~2 **AFEDAC2M1~AFEDAC2M0**：12-bit D/A 转换器 2 模式控制位
00：除能，输出处于高阻抗状态
01：使能
10：除能，输出为接地状态
11：使能

Bit 1~0 **AFEDAC1M1~AFEDAC1M0**：12-bit D/A 转换器 1 模式控制位
00：除能，输出处于高阻抗状态
01：使能
10：除能，输出为接地状态
11：使能

注：DAC2S1 与 DAC1S1 不能同时 On；DAC2S0 与 DAC1S0 不能同时 On。

● **AFEDAnH & AFEDAnL 寄存器 (n=1 或 2)**

寄存器	AFEDAnH								AFEDAnL							
位	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D11	D10	D9	D8	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	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	0	0	0	0	0	0	0	0	—	—	—	—

“—”：未定义，读为“0”

D11~D0：12-bit D/A 转换器 n 输出控制位

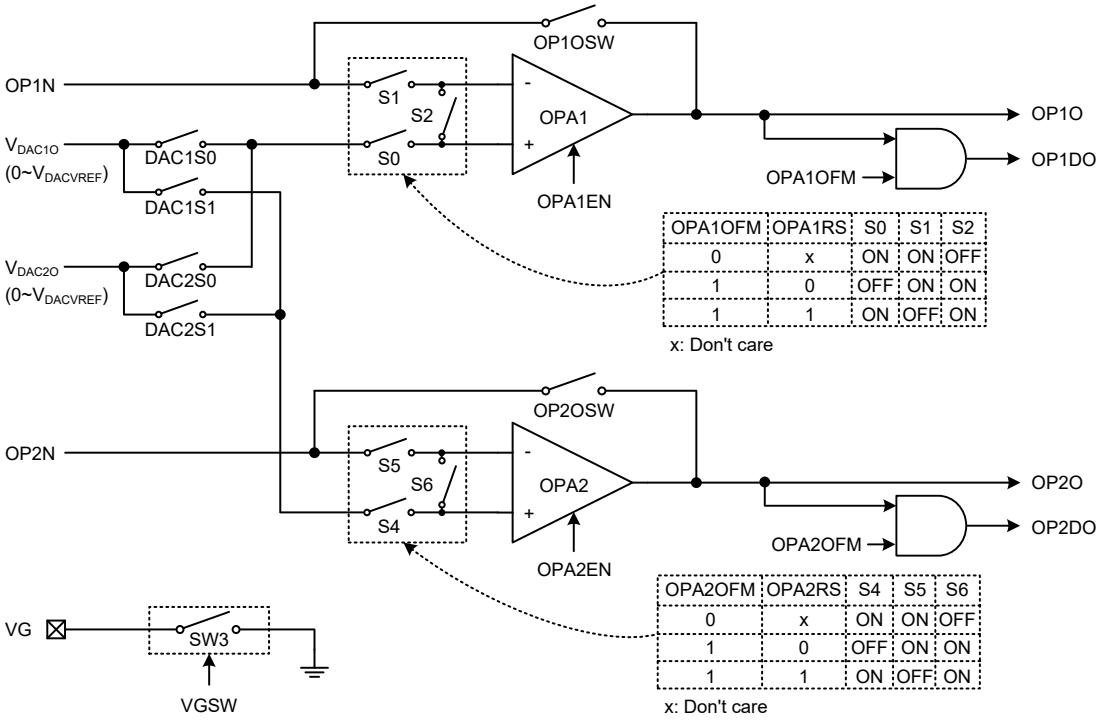
AFEDAnH 寄存器的 bit 7~bit 0 结合 AFEDAnL 寄存器的 bit 7~bit 4 形成一个 12-bit D/A 转换器 n 的值，范围为 0~4095。

DACn 输出电压 = $V_{DACVREF}$×(DACn 值 / 4096)

注：需先写入 AFEDAnL 后写入 AFEDAnH 寄存器。

运算放大器 – OPA

该单片机包含两个运算放大器 OPA1 和 OPA2，可用于测量应用产品。12-bit D/A 转换器为运算放大器正输入端提供 0~$V_{DACVREF}$ 的电压。这两个运算放大器分别通过控制开关 S0~S2 和 S4~S6 提供两种工作模式。通过合理的设置可将 OPA1 或 OPA2 输出电压内部连接到 A/D 转换器输入通道进行测量。



OPA1OFM	OPA1RS	S0	S1	S2
0	x	ON	ON	OFF
1	0	OFF	ON	ON
1	1	ON	OFF	ON

x: Don't care

OPA2OFM	OPA2RS	S4	S5	S6
0	x	ON	ON	OFF
1	0	OFF	ON	ON
1	1	ON	OFF	ON

x: Don't care

运算放大器结构

OPA 寄存器介绍

内部运算放大器的所有操作由一系列寄存器控制。OPAnC 寄存器用于 OPAn 功能的使能 / 除能控制，搭配 IREFC 寄存器中的 OPnDO 位可用于输入失调校准等。GSC 寄存器可设置 OP1OSW、OP2OSW 和 SW3 开关的 on/off，从而控制 OPA1 和 OPA2 输出以及 VG 引脚。

寄存器名称	位							
	7	6	5	4	3	2	1	0
OPA1C	OPA1OFM	OPA1RS	OPA1EN	OPA1OF4	OPA1OF3	OPA1OF2	OPA1OF1	OPA1OF0
OPA2C	OPA2OFM	OPA2RS	OPA2EN	OPA2OF4	OPA2OF3	OPA2OF2	OPA2OF1	OPA2OF0
OPSW	—	—	—	—	—	—	OP2OSW	OP1OSW
GSC	—	—	—	—	—	—	—	VGSW
IREFC	OP2DO	OP1DO	—	IREFEN	BATDEN	—	DACVRS1	DACVRS0

运算放大器寄存器列表

- **OPA1C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OPA1OFM	OPA1RS	OPA1EN	OPA1OF4	OPA1OF3	OPA1OF2	OPA1OF1	OPA1OF0
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 **OPA1OFM**：OPA1 正常模式或输入失调电压校准模式选择位
0：正常模式
1：输入失调电压校准模式
当选择输入失调电压校准模式时，应选择正输入端作为参考电压输入端。

Bit 6 **OPA1RS**：OPA1 输入失调电压校准参考选择位
0：选择 OP1N 作为参考电压输入端
1：选择 V_{DAC1O} 或 V_{DAC2O} 作为参考输入端
注：仅当 OPA1RS=1 时，才可执行 OPA1 输入失调电压校准。

Bit 5 **OPA1EN**：OPA1 使能 / 除能控制位
0：除能
1：使能

Bit 4~0 **OPA1OF4~OPA1OF0**：OPA1 输入失调电压校准控制位

- **OPA2C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OPA2OFM	OPA2RS	OPA2EN	OPA2OF4	OPA2OF3	OPA2OF2	OPA2OF1	OPA2OF0
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 **OPA2OFM**：OPA2 正常模式或输入失调电压校准模式选择位
0：正常模式
1：输入失调电压校准模式
当选择输入失调电压校准模式时，应选择正输入端作为参考电压输入端。

Bit 6 **OPA2RS**：OPA2 输入失调电压校准参考选择位
0：选择 OP2N 作为参考电压输入端
1：选择 V_{DAC1O} 或 V_{DAC2O} 作为参考输入端
注：仅当 OPA2RS=1 时，才可执行 OPA2 输入失调电压校准。

Bit 5 **OPA2EN**：OPA2 使能 / 除能控制位
0：除能
1：使能

Bit 4~0 **OPA2OF4~OPA2OF0**：OPA2 输入失调电压校准控制位

● **OPSW 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	OP2OSW	OP1OSW
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2　未定义，读为“0”

Bit 1　**OP2OSW**：开关 OP2OSW 控制位
0：Off
1：On

Bit 0　**OP1OSW**：开关 OP1OSW 控制位
0：Off
1：On

● **GSC 寄存器**

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

Bit 7~1　未定义，读为“0”

Bit 0　**VGSW**：开关 SW3 控制位
0：Off
1：On
当此位置高时，VG 引脚接地。

● **IREFC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OP2DO	OP1DO	—	IREFEN	BATDEN	—	DACVRS1	DACVRS0
R/W	R	R	—	R/W	R/W	—	R/W	R/W
POR	0	0	—	0	0	—	0	0

Bit 7　**OP2DO**：OPA2 数字输出；逻辑正
当 OPA2 功能除能时，该位被清零。
当 OPA2OFM 位置高时，OP2DO 值指示 OPA2 输出状态。具体请参考“输入失调校准”内容。

Bit 6　**OP1DO**：OPA1 数字输出；逻辑正
当 OPA1 功能除能时，该位被清零。
当 OPA1OFM 位置高时，OP1DO 值指示 OPA1 输出状态。具体请参考“输入失调校准”内容。

Bit 5　未定义，读为“0”

Bit 4　**IREFEN**：内部参考电压发生器使能控制
具体描述见内部参考电压发生器章节。

Bit 3　**BATDEN**：电池电压检测电路使能控制
具体描述见内部参考电压发生器章节。

Bit 2　未定义，读为“0”

Bit 1~0　**DACVRS1~DACVRS0**：12-bit D/A 转换器参考电压 $V_{DACVREF}$ 选择
具体描述见 D/A 转换器章节。

输入失调校准

OPAnC 寄存器的 OPAnOFM 位用于选择 OPAn 的工作模式，即正常操作或输入失调校准模式。必须要设置 OPAnRS 位为 1，才可执行输入失调电压校准。运算放大器输入失调电压校准的步骤如下所示：

- 步骤 1：设置 OPAnOFM=1 和 OPAnRS=1，使 OPAn 工作于失调校准模式，
  S0 和 S2 开关或 S4 和 S6 开关 ON，为确保 V_{OS} 在校准后足够小，校准模式的输入参考电压应该与正常模式的输入直流工作电压相同。
- 步骤 2：设置 OPAnOF[4:0]=00000，并读取 OPnDO 位的状态。
- 步骤 3：设置 OPAnOF[4:0]=OPAnOF[4:0]+1，然后读取 OPnDO 位的状态。
  如果 OPnDO 位未改变，重复步骤 3 直到 OPnDO 位的状态发生改变；
  如果 OPnDO 位发生变化，则记录此时的 OPAnOF[4:0] 值为 V_{OS1}，接着执行步骤 4。
- 步骤 4：设置 OPAnOF[4:0]=11111，然后读取 OPnDO 位的状态。
- 步骤 5：设置 OPAnOF[4:0]=OPAnOF[4:0]-1，然后读取 OPnDO 位的状态，
  如果 OPnDO 位未改变，重复步骤 5 直到 OPnDO 位的状态发生改变；
  如果 OPnDO 位发生变化，则记录此时的 OPAnOF[4:0] 值为 V_{OS2}，接着执行步骤 6。
- 步骤 6：将计算得到的 V_{OS} 值存储到 OPAnOF[4:0] 位中。校准完成。
  其中 $V_{OS}=(V_{OS1}+V_{OS2})/2$；若 $(V_{OS1}+V_{OS2})/2$ 非整数，则丢弃小数。

A/D 转换器 – ADC

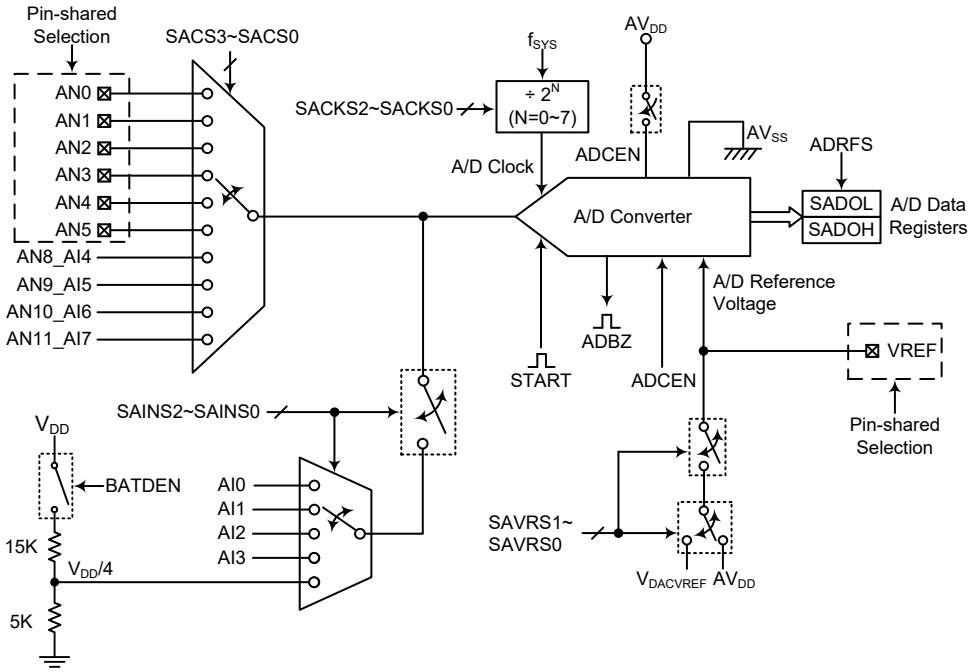
对于大多数电子系统而言，处理现实世界的模拟信号是共同的需求。为了完全由单片机来处理这些信号，首先需要通过 A/D 转换器将模拟信号转换成数字信号。将 A/D 转换器电路集成入单片机，可有效的减少外部器件，随之而来，具有降低成本和减少器件空间需求的优势。

A/D 转换器简介

此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 (如运算放大器信号 OP1N、OP2N、OP1O、OP2O 和电池电压检测电路输出 $V_{DD}/4$) 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。关于 A/D 转换器输入信号选择的具体细节可参考“A/D 转换器控制寄存器”及“A/D 转换器输入信号”章节进行了解。

外部输入通道	内部输入信号	A/D 通道选择位
6：AN0~AN5	OP1N，OP2N，OP1O，OP2O，$V_{DD}/4$	SAINS2~SAINS0，SACS3~SACS0

下图显示了 A/D 转换器内部结构和相关的寄存器。



注：1. 当选择 AIx (x=0~3) 选项时，实际模拟输入信号为通过正确配置 AISWIx 寄存器借由 AIx 引脚实现的运算放大器信号 OP1N、OP2N、OP1O 或 OP2O。

2. 当选择 ANm_AIx (m=8~11, x=4~7) 选项时，实际模拟输入信号为通过正确配置 AISWIx 寄存器借由 AIx 引脚连动 ANm 引脚实现的运算放大器信号 OP1N、OP2N、OP1O 或 OP2O。此外，AIx/ANm 引脚功能还需通过相应的引脚共用功能选择位预先设定。

3. BATDEN 位于 IREFC 寄存器中，在内部参考电压发生器章节有其相关描述。

A/D 转换器结构

A/D 转换器寄存器介绍

A/D 转换器的所有工作由几个寄存器控制。一对只读寄存器来存放 12 位 A/D 转换值，三个控制寄存器用于设置 A/D 转换器的操作和控制功能，ASWAI0~ASWAI7 寄存器通过设置相应的模拟开关的 on/off，从而控制 A/D 转换器模拟输入 AIx 连接到运算放大器信号 OP1N、OP2N、OP1O 或 OP2O。

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	—	—	—	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
SADC2	—	—	—	—	—	SAINS2	SAINS1	SAINS0
ASWAI0	—	—	—	—	AI0OP2O	AI0OP2N	AI0OP1O	AI0OP1N

寄存器名称	位							
	7	6	5	4	3	2	1	0
ASWAI1	—	—	—	—	AI1OP2O	AI1OP2N	AI1OP1O	AI1OP1N
ASWAI2	—	—	—	—	AI2OP2O	AI2OP2N	AI2OP1O	AI2OP1N
ASWAI3	—	—	—	—	AI3OP2O	AI3OP2N	AI3OP1O	AI3OP1N
ASWAI4	—	—	—	—	AI4OP2O	AI4OP2N	AI4OP1O	AI4OP1N
ASWAI5	—	—	—	—	AI5OP2O	AI5OP2N	AI5OP1O	AI5OP1N
ASWAI6	—	—	—	—	AI6OP2O	AI6OP2N	AI6OP1O	AI6OP1N
ASWAI7	—	—	—	—	AI7OP2O	AI7OP2N	AI7OP1O	AI7OP1N

A/D 转换器寄存器列表

A/D 转换器数据寄存器 – SADOL, SADOH

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

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 转换器数据寄存器

A/D 转换器控制寄存器 – SADC0, SADC1, SADC2

寄存器 SADC0、SADC1 和 SADC2 用来控制 A/D 转换器的功能和操作。这些 8 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。由于每个单片机只包含一个实际的模数转换电路，因此这些外部和内部模拟信号中的每一个都需要分别被发送到转换器。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。SADC2 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。

引脚共用功能选择寄存器的相关位用来定义 I/O 端口中的哪些引脚为 A/D 转换器的模拟输入，哪些引脚不作为 A/D 转换输入。当引脚作为 A/D 输入时，其原来的 I/O 或其它引脚共用功能消失，此外，其内部上拉电阻也将自动断开。

- **SADC0 寄存器**

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

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

Bit 6 **ADBZ**：A/D 转换忙碌标志位
0：A/D 转换结束或未开始转换
1：A/D 转换中
此位用于表明 A/D 转换过程是否完成。当 START 位由低变为高再变为低时，ADBZ 位为高，表明 A/D 转换已初始化。A/D 转换结束后，此位被清零。

Bit 5 **ADCEN**：A/D 转换器使能 / 除能控制位
0：除能
1：使能
此位控制 A/D 内部功能。该位被置高将使能 A/D 转换器。如果该位设为低将关闭 A/D 转换器以降低功耗。当 A/D 转换器功能除能时，A/D 数据寄存器 SADOH 和 SADOL 的内容将保持不变。

Bit 4 **ADRFS**：A/D 转换数据格式选择位
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。详情请参考 A/D 转换器数据寄存器章节。

Bit 3~0 **SACS3~SACS0**：A/D 转换器外部模拟通道输入选择位
0000：AN0
0001：AN1
0010：AN2
0011：AN3
0100：AN4
0101：AN5
1000：AN8_AI4 – AI4 与 AN8 连动
1001：AN9_AI5 – AI5 与 AN9 连动
1010：AN10_AI6 – AI6 与 AN10 连动
1011：AN11_AI7 – AI7 与 AN11 连动
1100~1111：未定义，输入浮空
这些位用于选择要转换的外部模拟输入通道。
需注意，当选择 ANm_AIx (m=8~11, x=4~7) 选项时，实际模拟输入信号为通过正确配置 AISWIx 寄存器借由 AIx 引脚连动 ANm 引脚实现的运算放大器信号 OP1N、OP2N、OP1O 或 OP2O。此外，AIx/ANm 引脚功能还需通过相应的引脚共用功能选择位预先设定。

● SADC1 寄存器

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

Bit 7~5 未定义，读为“0”

Bit 4~3 **SAVRS1~SAVRS0**：A/D 转换器参考电压选择位
00/11：外部 VREF 引脚
01：内部 A/D 转换器电源，AV_{DD}
10：内部 D/A 转换器参考电压，$V_{DACVREF}$
这两位用于选择 A/D 转换器参考电压。当 SAVRS1~SAVRS0 为被设为“01”或“10”选择参考电压来自内部信号 AV_{DD} 或 $V_{DACVREF}$ 时需特别注意。当选择 A/D 转换器电源或 D/A 转换器参考电压作为 A/D 转换器参考电压时，外部 VREF 引脚需通过引脚共用控制位选择作为其它共用的引脚功能。否则，外部 VREF 输入电压会和内部参考电压一起连接至内部 A/D 转换器，这将导致无法预期的结果。

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。

● **SADC2 寄存器**

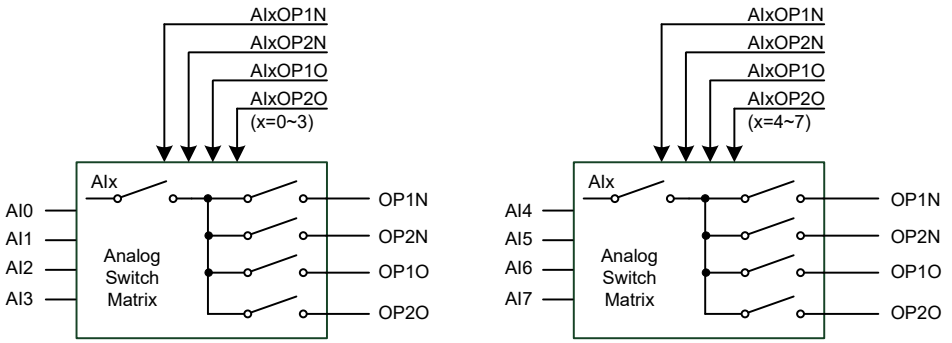
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	SAINS2	SAINS1	SAINS0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3　未定义，读为“0”

Bit 2~0　**SAINS2~SAINS0**：A/D 转换输入信号选择位
000：外部来源 – 外部模拟通道输入
001：内部来源 – 通过 AI0 引脚实现的运算放大器信号 OP1N、OP2N、OP1O 或 OP2O
010：内部来源 – 通过 AI1 引脚实现的运算放大器信号 OP1N、OP2N、OP1O 或 OP2O
011：内部来源 – 通过 AI2 引脚实现的运算放大器信号 OP1N、OP2N、OP1O 或 OP2O
100：内部来源 – 通过 AI3 引脚实现的运算放大器信号 OP1N、OP2N、OP1O 或 OP2O
101：外部来源 – 外部模拟通道输入
110：外部来源 – 外部模拟通道输入
111：内部来源 – 电池电压检测电路输出，$V_{DD}/4$
必须注意当 SAINS2~SAINS0 为“001~100”或“111”选择转换内部模拟信号时，外部输入通道一定不能作为 A/D 输入，需将 SACS3~SACS0 位设置为“1100~1111”中的任意值以确保所有的外部输入通道浮空。否则，外部输入通道会和内部模拟信号一起接至 A/D 转换器，这将导致不可预期的后果，甚至不可逆的损坏。

模拟开关控制寄存器

运算放大器信号 OP1N、OP2N、OP1O 或 OP2O 输入到 A/D 转换器必须借由 AIx 实现。ASWAI0~ASWAI7 寄存器则是用来控制 AIx 连接到 OP1N、OP2N、OP1O 或 OP2O 的模拟开关。



注：矩阵开关中的每个模拟开关都可单独控制。

模拟开关控制结构

● **ASWAIx 寄存器 (x=0~7)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	AIxOP2O	AIxOP2N	AIxOP1O	AIxOP1N
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　**AIxOP2O**：AIx 连接到 OP2O 开关控制
0：Off
1：On

Bit 2　**AIxOP2N**：AIx 连接到 OP2N 开关控制
0：Off
1：On

Bit 1　**AIxOP1O**：AIx 连接到 OP1O 开关控制
0：Off
1：On

Bit 0　**AIxOP1N**：AIx 连接到 OP1N 开关控制
0：Off
1：On

注：当任一 AIxOP1N 开关 On 时，其余的 AIxOP1N 无法 On。例如 AI0OP1N 开关 On 时，AI1OP1N~AI7OP1N 无法 On。AIxOP2N、AIxOP1O 与 AIxOP2O 亦同。

A/D 转换器操作

SADC0 寄存器中的 START 位，用于启动 A/D 转换。当单片机设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 位会自动清零。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。例如，如果系统时钟速度为 8MHz 时，SACKS2~SACKS0 位不能设为“000”、“001”或“111”。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 ($f_{SYS}/32$)	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

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。即使无引脚被选择作为 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

A/D 转换器参考电压可通过 SADC1 寄存器中 SAVRS1~SAVRS0 位选择来自电源电压 AV_{DD}、D/A 转换器参考电压 $V_{DACVREF}$ 或外部参考电压输入 VREF 引脚。若 SAVRS1~SAVRS0 设置为“01”，则参考电压来自于内部模拟电源 AV_{DD}，若 SAVRS1~SAVRS0 设置为“10”，则参考电压来自于 $V_{DACVREF}$。若 SAVRS1~SAVRS0 设置为除了“01”和“10”以外的值，则参考电压来自于 VREF 引脚输入。由于 VREF 引脚与其它功能共用引脚，需设置相关引脚共用控制位以使能 VREF 引脚功能。然而，如果内部 A/D 转换器电源电压或 D/A 转换器参考电压被选择作为参考电压源，则 VREF 外部引脚绝对不能设置为参考电压输入功能引脚，以免外部输入电压与选择的内部参考电压同时接入 A/D 转换器参考电压输入，而造成不可预期的损毁。需注意的是输入的模拟信号值一定不能超过选择的参考电压值。

SAVRS[1:0]	参考电压源	描述
00, 11	VREF 引脚	A/D 转换器外部参考电压输入引脚
01	AV_{DD}	A/D 转换器电源电压
10	$V_{DACVREF}$	D/A 转换器参考电压

A/D 转换器参考电压选择

A/D 转换器输入信号

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 转换器模拟输入脚功能或其它功能。如果对应的引脚作为 A/D 转换器模拟输入，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，则通过编程寄存器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。

另外还有五个内部模拟信号可连接到 A/D 转换器的模拟输入进行转换，包括运算放大器信号 OP1N、OP2N、OP1O 和 OP2O 以及电池电压检测电路输出，通过设置 SAINS2~SAINS0 位来选择。若选择外部通道输入，SAINS2~SAINS0 设置为“000”或“101~110”，SACS3~SACS0 决定哪个外部通道被选择。如果选择内部信号输入，则 SACS3~SACS0 位必须设置为“1100~1111”以关断外部模拟通道输入。否则会发生内部模拟信号与外部通道输入信号同时接入 A/D 转换器的情况，从而导致不可预期的错误。

SAINS[2:0]	SACS[3:0]	输入信号	描述
000, 101~110	0000~0101	AN0~AN5	外部模拟通道输入
	1000~1011	AN8_AI4~AN11_AI7	运算放大器信号 OP1N、OP2N、OP1O 或 OP2O，通过 AIx 连动 ANm 引脚实现
	1100~1111	—	输入浮空，未选择外部通道
001	1100~1111	AI0	运算放大器信号 OP1N、OP2N、OP1O 或 OP2O，通过 AI0 引脚实现

SAINS[2:0]	SACS[3:0]	输入信号	描述
010	1100~1111	AI1	运算放大器信号 OP1N、OP2N、OP1O 或 OP2O，通过 AI1 引脚实现
011	1100~1111	AI2	运算放大器信号 OP1N、OP2N、OP1O 或 OP2O，通过 AI2 引脚实现
100	1100~1111	AI3	运算放大器信号 OP1N、OP2N、OP1O 或 OP2O，通过 AI3 引脚实现
111	1100~1111	$V_{DD}/4$	电池电压检测电路输出

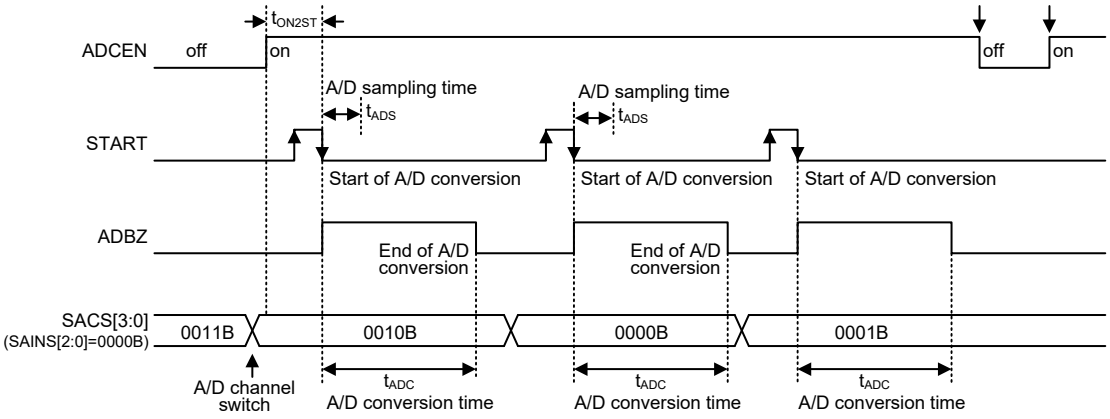
A/D 转换器输入信号选择

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 4 个 A/D 时钟周期，而数据转换需 12 个 A/D 时钟周期。所以一个完整的 A/D 转换时间，t_{ADC}，一共需要 16 个 A/D 时钟周期。

最大 A/D 转换率 = 1/(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 – 外部通道输入

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 位置高以使能 A/D 转换器功能。

- 步骤 3

  通过 SADC2 寄存器中的 SAINS[2:0] 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，接着应设置

SACS3~SACS0 位选择哪个外部通道接至 A/D 转换器。通过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。接着执行步骤 6。

- 步骤 5

  选择内部模拟信号前，应将 SACS3~SACS0 设置为 1100~1111 中的任意值以断开外部通道。接着设置 SAINS2~SAINS0 位选择所需的内部模拟信号。接着执行步骤 6。

- 步骤 6

  通过设置 SADC1 寄存器中的 SAVRS1~SAVRS0 位选择参考电压。若选择内部参考电压，必须通过合理设置相关引脚共用控制位将外部参考电压输入引脚功能除能。

- 步骤 7

  设置 SADC0 寄存器中的 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

  现在可以通过设定 SADC0 寄存器中的 START 位从“0”到“1”再回到“0”，开始 A/D 转换的过程。

- 步骤 10

  可以轮询 SADC0 寄存器中的 ADBZ 位，检查模数转换过程是否完成。当此位为逻辑高时，表示转换正在进行中。当此位为逻辑低时，表示转换过程已经完成。转换完成后，可读取 A/D 数据寄存器 SADOL 和 SADOH 获得转换后的值。

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

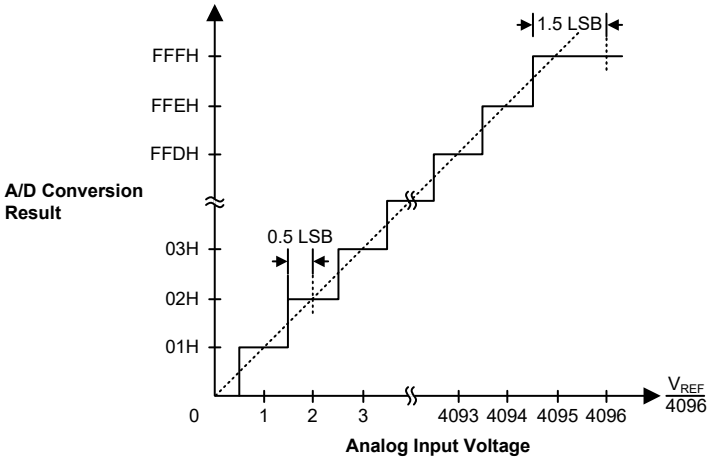
该单片机含有一个 12 位的 A/D 转换器，它们转换的最大值可达 FFFH。由于模拟输入最大值等于实际输入 A/D 转换器的参考电压 V_{REF} 的值，因此每一位可表示 $V_{REF}/4096$ 的模拟输入值。

$$1\ LSB=V_{REF}\div 4096$$

通过下面的等式可估算 A/D 转换器输入电压值：

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

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_{REF} 之前的 1.5 LSB 处改变。注意此处的 V_{REF} 为由 SAVRS[1:0] 选择的实际输入 A/D 转换器的参考电压。



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：使用轮询 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,30h                  ; setup PBS0 to configure pin AN0
mov  PBS0,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  SADOL_buffer,a         ; save result to user defined register
mov  a,SADOH                ; read high byte conversion result value
mov  SADOH_buffer,a         ; save result to user defined register
:
:
jmp  start_conversion       ; start next A/D conversion
```

范例：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,30h                  ; setup PBS0 to configure pin AN0
```

```
mov PBS0,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 interrupt service routine
ADC_ISR:
mov acc_stack,a            ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a         ; save STATUS to user defined memory
:
:
mov a,SADOL                ; read low byte conversion result value
mov SADOL_buffer,a         ; save result to user defined register
mov a,SADOH                ; read high byte conversion result value
mov SADOH_buffer,a         ; save result to user defined register
:
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a               ; restore STATUS from user defined memory
mov a,acc_stack            ; restore ACC from user defined memory
reti
```

通用串行接口模块 – USIM

此单片机内有两个通用串行接口模块即 USIM0 和 USIM1，每个 USIM 模块包括三种易与外部设备通信的串行接口：四线 SPI、两线 I^2C 或两线 / 单线 UART 接口。这三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。因为每个 USIM 接口引脚是与其它 I/O 引脚共用，因此在使用 USIM 功能前，要先通过相应的引脚共用功能选择寄存器选定 USIM 引脚功能。因为这三种接口共用引脚和寄存器，所以要先通过 SIMnC0 寄存器中的 USIMn UART 模式选择位 UnMD 和 SPI/I^2C 工作模式控制位 SIMn2~SIMn0 选择哪一种通信接口。若 USIMn 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 USIMn 脚的上拉电阻。

SPI 接口

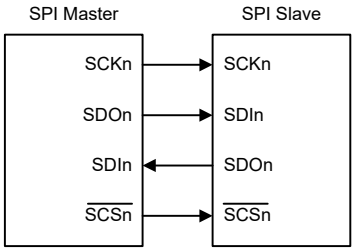
此 SPI 接口属于通用串行接口模块中的一种，不要与另一章节介绍的独立的 SPI 接口功能混淆。

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 USIM0 或 USIM1 中的 SPI 中只有一个片选信号引脚 $\overline{SCSn}$。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

USIMn SPI 接口是一个全双工串行数据传输器。SPI 接口的四线为：SDIn、SDOn、SCKn 和 $\overline{\text{SCSn}}$。SDIn 和 SDOn 是数据的输入和输出线。SCKn 是串行时钟线，$\overline{\text{SCSn}}$ 是从机的选择线。USIMn SPI 的接口引脚与普通 I/O 口和 USIMn I^2C/UART 的功能脚共用。通过设定相关引脚共用选择位和 SIMnC0/SIMnC2 寄存器的对应位，来使能 USIMn SPI 接口。连接到 SPI 接口的单片机以从主 / 从模式进行通信，且所有的数据传输由主机发起，时钟信号也由主机控制。由于单片机只有一个 $\overline{\text{SCSn}}$ 引脚，所以只能拥有一个从机设备。可通过软件控制 $\overline{\text{SCSn}}$ 引脚使能与除能，设置 CSENn 位为“1”使能 $\overline{\text{SCSn}}$ 功能，设置 CSENn 位为“0”，$\overline{\text{SCSn}}$ 引脚将处于浮空状态。

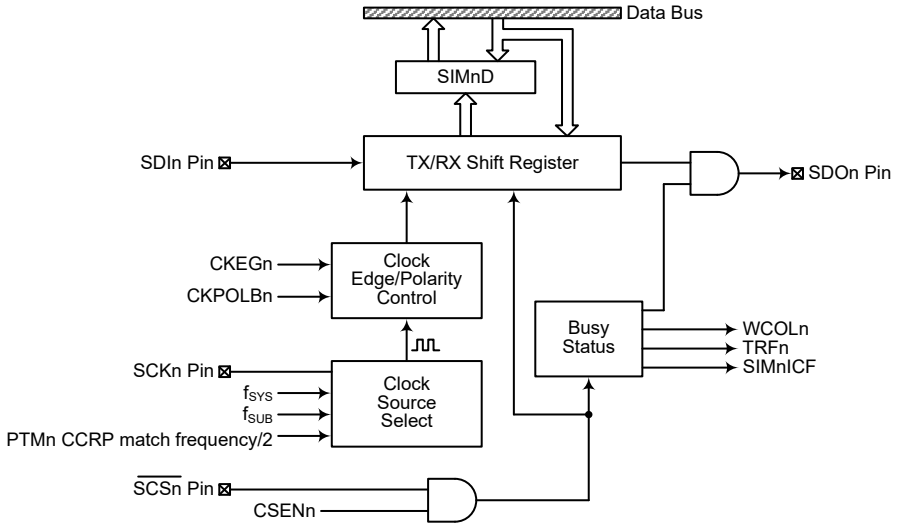


USIMn SPI 主 / 从机连接方式

该单片机的 SPI 功能具有以下特点：

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

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



USIMn SPI 方框图 (n=0~1)

SPI 寄存器

有三个内部寄存器用于控制 USIMn SPI 接口的所有操作，其中有一个数据寄存器 SIMnD、两个控制寄存器 SIMnC0 和 SIMnC2。注意，只有在合理设置 SIMnC0 寄存器中的 UnMD 位和 SIMn2~SIMn0 位选择 USIMn SPI 模式后，SIMnC2 和 SIMnD 寄存器以及它们的上电复位值才有效。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMnC0	SIMn2	SIMn1	SIMn0	UnMD	SIMnDEB1	SIMnDEB0	SIMnEN	SIMnICF
SIMnC2	D7	D6	CKPOLBn	CKEGn	MLSn	CSENn	WCOLn	TRFn
SIMnD	D7	D6	D5	D4	D3	D2	D1	D0

USIMn SPI 寄存器列表 (n=0~1)

SPI 数据寄存器

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

- **SIMnD 寄存器**

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

“x”：未知

Bit 7~0 **D7~D0**：USIMn SPI/I²C 数据寄存器位 bit 7 ~ bit 0

SPI 控制寄存器

单片机中也有两个控制 USIMn SPI 接口功能的寄存器，SIMnC0 和 SIMnC2。寄存器 SIMnC0 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 SIMnC2 用于其它的控制功能如 LSB/MSB 选择，写冲突标志位等。

- **SIMnC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIMn2	SIMn1	SIMn0	UnMD	SIMnDEB1	SIMnDEB0	SIMnEN	SIMnICF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SIMn2~SIMn0**：USIMn SPI/I²C 工作模式控制位
000：SPI 主机模式；SPI 时钟为 $f_{SYS}/4$
001：SPI 主机模式；SPI 时钟为 $f_{SYS}/16$
010：SPI 主机模式；SPI 时钟为 $f_{SYS}/64$
011：SPI 主机模式；SPI 时钟为 f_{SUB}
100：SPI 主机模式；SPI 时钟为 PTMn CCRP 匹配频率 / 2
101：SPI 从机模式
110：I²C 从机模式
111：未使用模式
当 UnMD 位清零时，这几位用于设置 USIMn SPI/I²C 功能的工作模式，除了选择 USIMn 模块的 I²C 或 SPI 功能，还可选择 USIMn SPI 的主从模式和 SPI 的主机时钟频率。USIMn SPI 时钟源可来自于系统时钟和 f_{SUB} 也可以选择来自

PTMn。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 **UnMD**：USIMn UART 模式选择位
0：USIMn SPI/I²C 模式
1：USIMn UART 模式
此位为 USIMn UART 模式选择位。当此位清零时，选择 SPI/I²C 模式，而对 SPI 或 I²C 模式的选择是通过 SIMn2~SIMn0 位实现。

Bit 3~2 **SIMnDEB1~SIMnDEB0**：USIMn I²C 去抖时间选择位
这些位只有在 USIMn 设置成 I²C 接口模式时才有效。请参考 I²C 寄存器部分。

Bit 1 **SIMnEN**：USIMn SPI/I²C 控制位
0：除能
1：使能
此位为 USIMn SPI/I²C 接口的开 / 关控制位。此位为“0”时，USIMn SPI/I²C 接口除能，SDIn、SDOn、SCKn 和 $\overline{\text{SCSn}}$ 或 SDAn 和 SCLn 脚将失去 SPI 或 I²C 功能，USIMn 工作电流减小到最小值。此位为“1”时，USIMn SPI/I²C 接口使能。若 USIMn 经由 UnMD 位和 SIMn2~SIMn0 位设置为工作在 SPI 接口，当 SIMnEN 位由低到高转变时，USIMn 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 USIMn 经由 UnMD 位和 SIMn2~SIMn0 位设置为工作在 I²C 接口，当 SIMnEN 位由低到高转变时，USIMn I²C 控制寄存器中的设置，如 HTXn 和 TXAKn，将不会发生变化，其首先应在应用程序中初始化，此时相关 I²C 标志，如 HCFn、HAASn、HBBn、SRWn 和 RXAKn，将被设置为其默认状态。

Bit 0 **SIMnICF**：USIMn SPI 未完成标志位
0：未发生
1：发生
此位仅当 USIMn 配置在 SPI 从机模式时有效。如果 SPI 工作在从机模式且 SIMnEN 和 CSENn 位都为“1”，但在 SPI 数据传输完全结束前 $\overline{\text{SCSn}}$ 线被外部主机拉高，SIMnICF 和 TRFn 位都会被置高。在这种情况下，如果相应的中断功能使能将产生一个中断。然而，如果 SIMnICF 位是由软件应用程序设为 1，那么 TRFn 位将不会置高。

● SIMnC2 寄存器

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLBn	CKEGn	MLSn	CSENn	WCOLn	TRFn
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 **D7~D6**：未定义位
用户可通过软件程序对这两位进行读写。

Bit 5 **CKPOLBn**：USIMn SPI 时钟线的基础状态位
0：当时钟无效时，SCKn 引脚为高电平
1：当时钟无效时，SCKn 引脚为低电平
此位决定了时钟线的基础状态，若此位为高，当时钟无效时 SCKn 为低电平，若此位为低，当时钟无效时 SCKn 为高电平。

Bit 4 **CKEGn**：USIMn SPI 的有效时钟边沿类型位
CKPOLBn=0
0：SCKn 为高电平且在 SCKn 上升沿抓取数据
1：SCKn 为高电平且在 SCKn 下降沿抓取数据
CKPOLBn=1
0：SCKn 为低电平且在 SCKn 下降沿抓取数据
1：SCKn 为低电平且在 SCKn 上升沿抓取数据
CKEGn 和 CKPOLBn 位用于设置 SPI 总线上时钟信号输入和输出数据方式。这两位必须在执行数据传输前先被设置好，否则将产生错误的时钟边沿信号。CKPOLBn 位决定时钟线的基础状态，若此位为高，则时钟无效时 SCKn 为低电平，若此位为低，则时钟无效时 SCKn 为高电平。CKEGn 位决定有效时钟边沿类型，取决于 CKPOLBn 的状态。

Bit 3 **MLSn**：USIMn SPI 数据移位顺序控制位
0：LSB 优先
1：MSB 优先
数据移位选择位，用于选择数据传输时高位优先传输还是低位优先传输。此位设置为高时高位优先传输，为低时低位优先传输。

Bit 2 **CSENn**：USIMn SPI $\overline{\text{SCSn}}$ 引脚控制位
0：除能
1：使能
CSENn 位用于 $\overline{\text{SCSn}}$ 引脚的使能 / 除能控制。此位为低时，$\overline{\text{SCSn}}$ 除能并处于浮空状态。此位为高时，$\overline{\text{SCSn}}$ 作为选择脚。

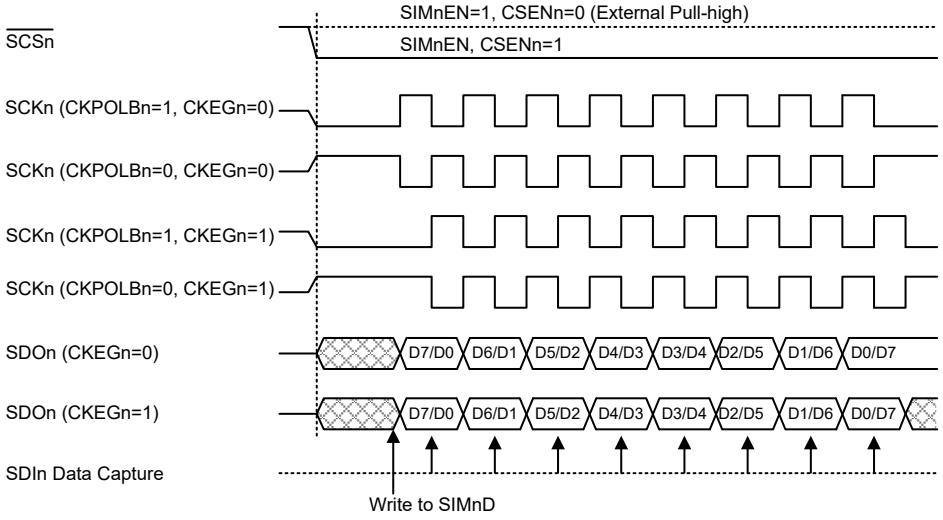
Bit 1 **WCOLn**：USIMn SPI 写冲突标志位
0：无冲突
1：冲突
WCOLn 标志位用于监测数据冲突的发生。此位为高时，表示在传输过程中有数据被写入 SIMnD 寄存器。若数据正在被传输时，写操作无效。此位可通过应用程序清零。

Bit 0 **TRFn**：USIMn SPI 发送 / 接收结束标志位
0：数据正在发送
1：数据发送结束
TRFn 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，但须通过应用程序清为“0”。此位也可用于产生中断。

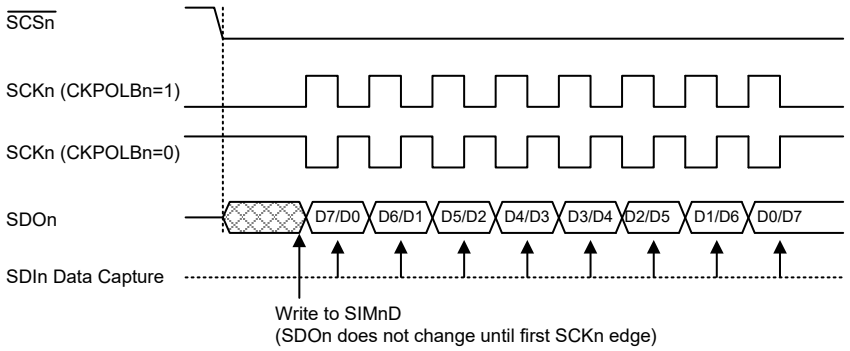
SPI 通信

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

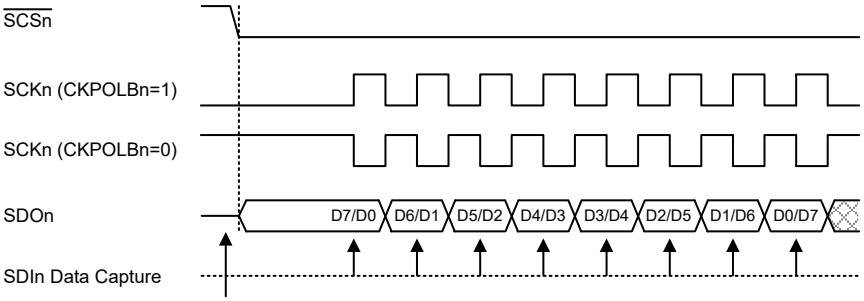
SPI 主机模式在 SPI 时钟运行情况下可以进行正常传输。



USIMn SPI 主机模式时序

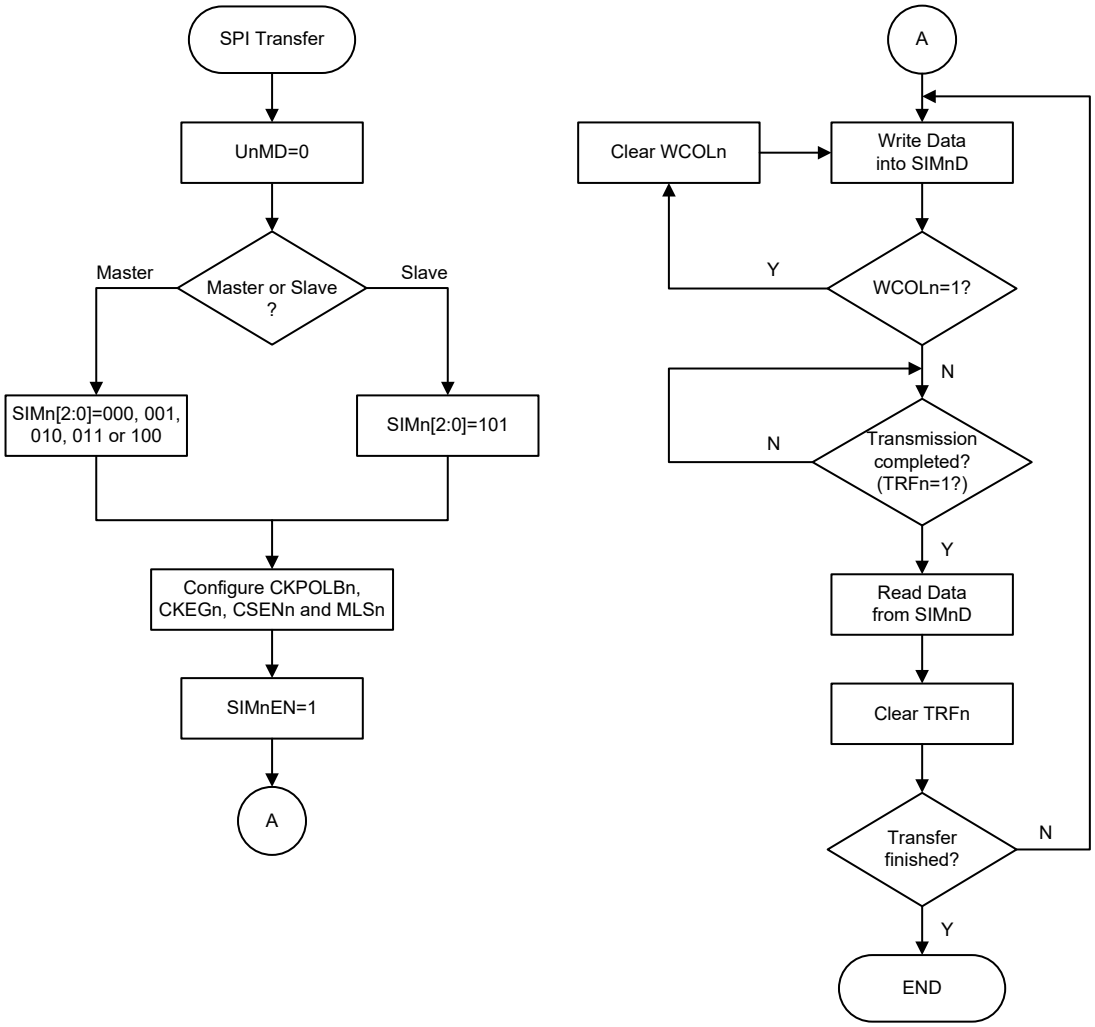


USIMn SPI 从机模式时序 – CKEGn=0



Note: For USIMn SPI slave mode, if SIMnEN=1 and CSENn=0, SPI is always enabled and ignores the SCSn level.

USIMn SPI 从机模式时序 – CKEGn=1



USIMn SPI 传输控制流程图

SPI 使能 / 除能

设置 CSENn=1、$\overline{SCSn}$=0 将使能 USIMn SPI 总线，然后等待写数据到 SIMnD 寄存器 (TXRX 缓存器)。单片机处于主机模式，数据写入 SIMnD 寄存器后，自动开始数据传输或接收操作。数据传输完成时，TRFn 位将自动被置位。单片机处于从机模式，SCKn 引脚上收到脉冲信号之后，会传输 TXRX 中的数据，或将 SDIn 引脚上的数据移入。

当 USIMn SPI 总线除能时，通过设置相应引脚共用控制位，SCKn、SDIn、SDOn、$\overline{SCSn}$ 可作为 I/O 口或其它功能引脚使用。

SPI 操作步骤

四线制 SPI 接口可完成所有主 / 从模式通信工作。

在 SIMnC2 寄存器中，CSENn 位控制 USIMn SPI 接口的所有功能。设置此位为高，$\overline{SCSn}$ 信号线有效将使能 USIMn SPI 接口。设置此位为低，USIMn SPI 接口将除能，$\overline{SCSn}$ 信号线处于浮空状态因此不能控制该 SPI 接口。CSENn 位和 SIMnC0 寄存器中的 SIMnEN 位设置为高，使得 SDIn 信号线处于浮空状态

且 SDOn 信号线为高电平。主机模式中，如果 SCKn 信号线为高还是低取决于 SIMnC2 寄存器中的时钟极性选择位 CKPOLBn。从机模式中，SCKn 信号线处于浮空状态。如果 SIMnEN 位设置为低，USIMn SPI 接口被除能，通过设置相应引脚共用控制位，$\overline{\text{SCSn}}$、SDIn、SDOn 和 SCKn 可作为 I/O 口或其它功能引脚使用。主机模式中，当数据被写入 SIMnD 寄存器后，主机发起数据传输，并控制时钟信号。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 SIMnC0 控制寄存器中的 UnMD 和 SIMn2~SIMn0 位，选择 USIMn SPI 主机模式和时钟源。

- 步骤 2

  设置 CSENn 和 MLSn 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 SIMnC0 控制寄存器中的 SIMnEN 位，使能 USIMn SPI 接口功能。

- 步骤 4

  对于写操作：写数据到 SIMnD 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。再使用 SCKn 和 SDOn 信号线将数据输出。接着执行步骤 5。

  对于读操作：从 SDIn 信号线移入的数据将被存储在 TXRX 缓存器中，直到所有数据接收完毕，此时数据全部锁存至 SIMnD 寄存器。

- 步骤 5

  检测 WCOLn 位，若此位为高，表示发生数据冲突则跳回至步骤 4；若为低，则继续执行下面的步骤。

- 步骤 6

  检测 TRFn 位或等待 USIMn SPI 串行总线中断发生。

- 步骤 7

  从 SIMnD 寄存器中读数据。

- 步骤 8

  清除 TRFn。

- 步骤 9

  跳回至步骤 4。

从机模式

- 步骤 1

  设置 SIMnC0 控制寄存器中的 UnMD 和 SIMn2~SIMn0 位，选择 USIMn SPI 从机模式。

- 步骤 2

  设置 CSENn 和 MLSn 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 SIMnC0 控制寄存器中的 SIMnEN 位，使能 USIMn SPI 接口功能。

- 步骤 4

  对于写操作：写数据到 SIMnD 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。等待主机时钟 SCKn 信号和 $\overline{\text{SCSn}}$ 信号。接着执行步骤 5。

  对于读操作：从 SDIn 信号线移入的数据将被存储在 TXRX 缓存器中，直到所有数据接收完毕，此时数据全部锁存至 SIMnD 寄存器。

- 步骤 5

  检测 WCOLn 位，若此位为高，表示发生数据冲突则跳回至步骤 4；若为低，则继续执行下面的步骤。

- 步骤 6

  检测 TRFn 位或等待 USIMn SPI 串行总线中断发生。

- 步骤 7

  从 SIMnD 寄存器中读数据。

- 步骤 8

  清除 TRFn。

- 步骤 9

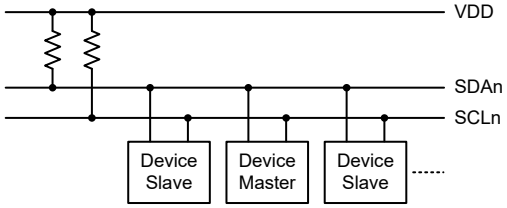
  跳回至步骤 4。

错误侦测

SIMnC2 寄存器中的 WCOLn 位用于数据传输期间监测数据冲突的发生。此位由 USIMn SPI 串行接口设置为高，但须通过应用程序清除为零。在数据传输期间如果写数据到 SIMnD，此位被置高提示数据冲突发生，并阻止数据继续被写入。

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。



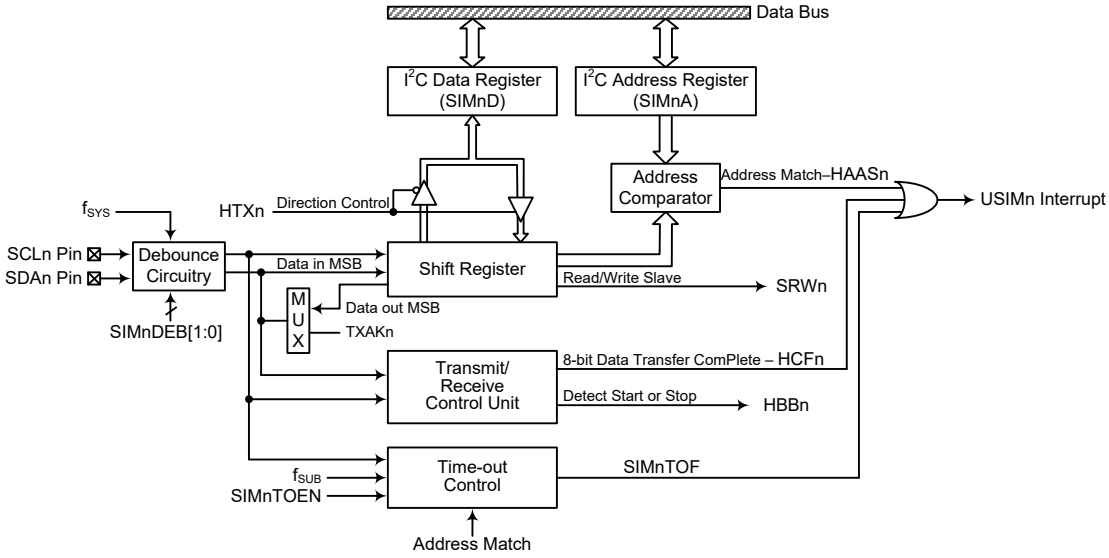
USIMn I²C 主从总线连接图

I²C 接口操作

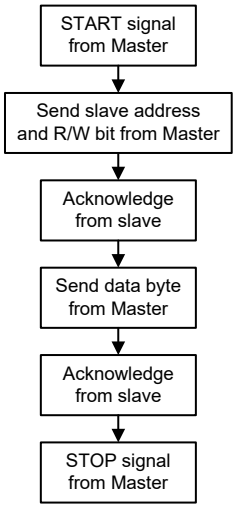
I²C 串行接口是一个双线的接口，有一条串行数据线 SDAn 和一条串行时钟线 SCLn。由于可能有多个设备在同一条总线上相互连接，所以这些设备的输出都是开漏型输出。因此应在这些输出口上都应加上拉电阻。应注意的是，I²C 总线上的每个设备都没有选择线，但分别与唯一的地址一一对应，用于 I²C 通信。

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。即使 I²C 设备被激活，与 SCLn/SDAn 引脚共用

的 I/O 口上拉电阻控制功能仍有效，其上拉电阻功能由相应的上拉电阻控制寄存器控制。



USIMn I²C 方框图 (n=0~1)



USIMn I²C 接口操作

SIMnDEB1 和 SIMnDEB0 位决定 USIMn I²C 接口的去抖时间。这个功能可以使用内部时钟在外部时钟上增加一个去抖间隔，减小时钟线上毛刺发生的可能性，以避免单片机发生误动作。如果选择了这个功能，去抖时间可以选择 2 个或 4 个系统时钟。为了达到需要的 I²C 数据传输速度，系统时钟 f_{SYS} 和 I²C 去抖时间之间存在一定的关系。I²C 标准模式或者快速模式下，用户需注意所选的系统时钟频率与标准匹配去抖时间的设置，其具体关系如下表所示。

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}>4MHz
2 个系统时钟去抖时间	f_{SYS}>4MHz	f_{SYS}>8MHz
4 个系统时钟去抖时间	f_{SYS}>4MHz	f_{SYS}>8MHz

I²C 最小 f_{SYS} 频率要求

I²C 寄存器

USIMn I²C 总线有三个控制寄存器 SIMnC0、SIMnC1 和 SIMnTOC，一个地址寄存器 SIMnA 以及一个数据寄存器 SIMnD。注意，只有在合理设置 SIMnC0 寄存器中的 UnMD 位和 SIMn2~SIMn0 位选择 USIMn I²C 模式后，SIMnC1、SIMnD、SIMnA 和 SIMnTOC 寄存器以及它们的上电复位值才有效。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMnC0	SIMn2	SIMn1	SIMn0	UnMD	SIMnDEB1	SIMnDEB0	SIMnEN	SIMnICF
SIMnC1	HCFn	HAASn	HBBn	HTXn	TXAKn	SRWn	IAMWUn	RXAKn
SIMnA	SIMnA6	SIMnA5	SIMnA4	SIMnA3	SIMnA2	SIMnA1	SIMnA0	D0
SIMnD	D7	D6	D5	D4	D3	D2	D1	D0
SIMnTOC	SIMnTOEN	SIMnTOF	SIMnTOS5	SIMnTOS4	SIMnTOS3	SIMnTOS2	SIMnTOS1	SIMnTOS0

USIMn I²C 寄存器列表 (n=0~1)

I²C 数据寄存器

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

- **SIMnD 寄存器**

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

“x”：未知

Bit 7~0 **D7~D0**：USIMn SPI/I²C 数据寄存器位 bit 7 ~ bit 0

I²C 地址寄存器

SIMnA 寄存器也在 USIMn SPI 接口功能中使用，但其名称改为 SIMnC2。SIMnA 寄存器用于存放 7 位从机地址，寄存器 SIMnA 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 的主机发送出的地址和寄存器 SIMnA 中存储的地址相符，那么就选中了这个从机。

- **SIMnA 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIMnA6	SIMnA5	SIMnA4	SIMnA3	SIMnA2	SIMnA1	SIMnA0	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 7~1 **SIMnA6~SIMnA0**：USIMn I²C 从机地址位
SIMnA6~SIMnA0 是从机地址 bit 6 ~ bit 0。

Bit 0 **D0**：保留位，此位可通过软件程序进行读写。

I²C 控制寄存器

单片机中有三个控制 USIMn I²C 接口功能的寄存器，SIMnC0、SIMnC1 和 SIMnTOC。寄存器 SIMnC0 用于控制使能 / 除能功能和选择 I²C 从机模式以及去抖时间。寄存器 SIMnC1 包括多个用于指示 I²C 传输状态的相关标志位。SIMnTOC 寄存器用于控制 I²C 超时功能，此寄存器在 I²C 超时一节介绍。

- **SIMnC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIMn2	SIMn1	SIMn0	UnMD	SIMnDEB1	SIMnDEB0	SIMnEN	SIMnICF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SIMn2~SIMn0**：USIMn SPI/I²C 模式控制位
000：SPI 主机模式；SPI 时钟为 f_{SYS}/4
001：SPI 主机模式；SPI 时钟为 f_{SYS}/16
010：SPI 主机模式；SPI 时钟为 f_{SYS}/64
011：SPI 主机模式；SPI 时钟为 f_{SUB}
100：SPI 主机模式；SPI 时钟为 PTMn CCRP 匹配频率 / 2
101：SPI 从机模式
110：I²C 从机模式
111：未使用模式
当 UnMD 位清零时，这几位用于设置 USIMn SPI/I²C 功能的工作模式，除了选择 USIMn 模块的 I²C 或 SPI 功能，还可选择 USIMn SPI 的主从模式和 SPI 的主机时钟频率。USIMn SPI 时钟源可来自于系统时钟和 f_{SUB} 也可以选择来自 PTMn。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4 **UnMD**：USIMn UART 模式选择位
0：USIMn SPI/I²C 模式
1：USIMn UART 模式
此位为 USIMn UART 模式选择位。当此位清零时，选择 SPI/I²C 模式，而对 SPI 或 I²C 模式的选择是通过 SIMn2~SIMn0 位实现。

Bit 3~2 **SIMnDEB1~SIMnDEB0**：USIMn I²C 去抖时间选择位
00：无去抖时间
01：2 个系统时钟去抖时间
1x：4 个系统时钟去抖时间
当设置 UnMD 位为“0”、SIMn2~SIMn0 位为“110”将 USIMn 设置为 I²C 接口功能时，这两个位用于选择 I²C 去抖时间。

Bit 1 **SIMnEN**：USIMn SPI/I²C 控制位
0：除能
1：使能
此位为 USIMn SPI/I²C 接口的开 / 关控制位。此位为“0”时，USIMn SPI/I²C 接口除能，SDIn、SDOn、SCKn 和 $\overline{SCSn}$ 或 SDAn 和 SCLn 脚将失去 SPI 或 I²C 功能，USIMn 工作电流减小到最小值。此位为“1”时，USIMn SPI/I²C 接口使能。若 USIMn 经由 UnMD 位和 SIMn2~SIMn0 位设置为工作在 SPI 接口，当 SIMnEN 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 USIMn 经由 UnMD 位和 SIMn2~SIMn0 位设置为工作在 I²C 接口，当 SIMnEN 位由低到高转变时，I²C 控制寄存器中的设置，如 HTXn 和 TXAKn，将不会发生变化，其首先应在应用程序中初始化，此时相关 I²C 标志，如 HCFn、HAASn、HBBn、SRWn 和 RXAKn，将被设置为其默认状态。

Bit 0 **SIMnICF**：USIMn SPI 未完成标志位
此位仅当 USIMn 配置在 SPI 从机模式时有效。请参考 SPI 寄存器部分。

● **SIMnC1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	HCFn	HAASn	HBBn	HTXn	TXAKn	SRWn	IAMWUn	RXAKn
R/W	R	R	R	R/W	R/W	R	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7 **HCFn**：USIMn I²C 总线数据传输结束标志位
0：数据正在被传输
1：8 位数据传输完成
数据正在传输时该位为低。当 8 位数据传输完成时，此位为高并产生一个中断。

Bit 6 **HAASn**：USIMn I²C 地址匹配标志位
0：地址不匹配
1：地址匹配
此标志位用于决定从机地址是否与主机发送的地址相同。若地址匹配此位为高，否则此位为低。

Bit 5 **HBBn**：USIMn I²C 总线忙标志位
0：I²C 总线闲
1：I²C 总线忙
当检测到 START 信号时 I²C 忙，此位变为高电平。当检测到 STOP 信号时 I²C 总线空闲，该位变为低电平。

Bit 4 **HTXn**：USIMn I²C 从机处于发送或接收模式标志位
0：从机处于接收模式
1：从机处于发送模式

Bit 3 **TXAKn**：USIMn I²C 总线发送应答标志位
0：从机发送应答标志
1：从机没有发送应答标志
从机接收完 8 位数据之后，该位将在第九个从机时钟时被传到总线上。如果从机想要接收更多的数据，则应在接收数据之前将此位设置为“0”。

Bit 2 **SRWn**：USIMn I²C 从机读 / 写位
0：从机应处于接收模式
1：从机应处于发送模式
SRWn 位是从机读写位。决定主机是否希望传输数据或接收来自 I²C 总线的数据。当传输地址和从机的地址相同时，HAASn 位会被设置为高，从机将检测 SRWn 位来决定进入发送模式还是接收模式。如果 SRWn 位为高时，主机会请求从总线上读数据，此时从机处于传输模式。当 SRWn 位为“0”时，主机往总线上写数据，从机处于接收模式以读取数据。

Bit 1 **IAMWUn**：USIMn I²C 地址匹配唤醒控制位
0：除能
1：使能
此位设置为“1”则使能 I²C 地址匹配使系统从休眠或空闲模式中唤醒的功能。若进入休眠或空闲模式前 IAMWUn 已经置高以使能 I²C 地址匹配唤醒功能，在系统唤醒后须软件清除此位以确保单片机正确地运行。

Bit 0 **RXAKn**：USIMn I²C 总线接收应答标志位
0：从机接收到应答标志
1：从机没有接收到应答标志
RXAKn 位是接收应答标志位。如果 RXAKn 位为“0”，即表示 8 位数据传输之后，从机在第九个时钟有接受到一个应答信号。如果从机处于发送状态，从机作为发送方会检查 RXAKn 位来判断主机接收方是否愿意继续接收下一个字节。因此发送方会一直发送数据，直到 RXAKn 为“1”时才停止发送数据。这时，发送方将释放 SDAn 线，主机方可发出停止信号从而释放 I²C 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，SIMnC1 寄存器的 HAASn 位会被置位，同时产生 USIMn 中断。进入中断服务程序后，系统要检测 HAASn 位和 SIMnTOF 位，以判断中断源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 USIMn I²C 超时。在数据传递中，要注意的是，在 7 位从机地址被发送后，接下来的一位，即第 8 位，是读 / 写控制位，该位的值会反映到 SRWn 位中。从机通过检测 SRWn 位以确定自己是要进入发送模式还是接收模式。在 I²C 总线开始传送数据前，需要先初始化 I²C 总线，初始化 I²C 总线步骤如下：

- 步骤 1

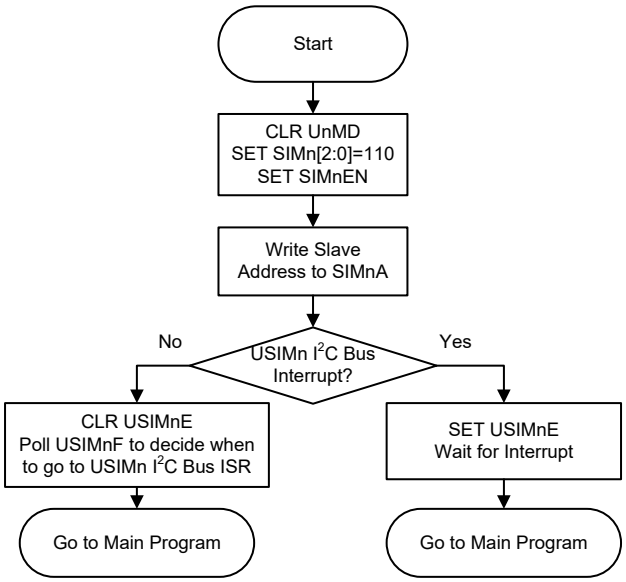
  设置 SIMnC0 寄存器中 UnMD 位为“0”、SIMn2~SIMn0 位为“110”和 SIMnEN 位为“1”，以使能 USIMn I²C 总线。

- 步骤 2

  向 USIMn I²C 总线地址寄存器 SIMnA 写入从机地址。

- 步骤 3

  设置中断控制寄存器中的 USIMnE 位以使能 USIMn 中断。



USIMn I²C 总线初始化流程图

I²C 总线起始信号

起始信号只能由连接 I²C 总线的主机产生，而不是由从机产生。总线上的所有从机都可以侦测到起始信号。如果有从机侦测到起始信号，则表明 I²C 总线处于忙碌状态，并会置位 HBBn。起始信号是指在 SCLn 为高电平时，SDAn 线上发生从高到低的电平变化。

I²C 从机地址

总线上的所有从机都会侦测由主机发出的起始信号。发送起始信号后，紧接着

主机会发送从机地址以选择要进行数据传输的从机。所有在 I²C 总线上的从机接收到 7 位地址数据后，都会将其与各自内部的地址进行比较。如果从机从主机上接收到的地址与自身内部的地址相匹配，则会产生一个 USIMn 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 SIMnC1 寄存器的 SRWn 位，从机随后发出一个低电平应答信号 (即第 9 位)。当从机地址匹配时，从机会将状态标志位 HAASn 置位。

USIMn I²C 总线中断有三个中断源，当程序运行至中断服务子程序时，通过检测 HAASn 位和 SIMnTOF 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 USIMn 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 SIMnD 寄存器，或是用于接收模式并从 SIMnD 寄存器中读取空值以释放 SCLn 线。

I²C 总线读 / 写信号

SIMnC1 寄存器的 SRWn 位用来表示主机是要从 I²C 总线上读取数据还是要将数据写到 I²C 总线上。从机通过检测该位以确定自己是作为发送方还是接收方。当 SRWn 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 SRWn 清“0”，表示主机要写数据到 I²C 总线上，从机则做为接收方，从 I²C 总线上读取数据。

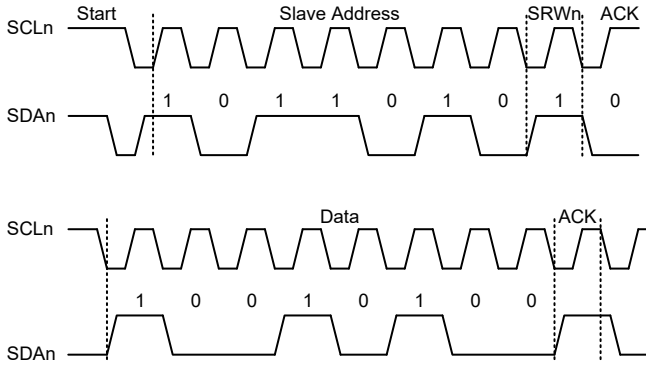
I²C 总线从机地址应答信号

主机发送呼叫地址后，当 I²C 总线上的任何从机内部地址与其匹配时，会发送一个应答信号。此应答信号会通知主机有从机已经接收到了呼叫地址。如果主机没有收到应答信号，则主机必须发送停止 (STOP) 信号以结束通信。当 HAASn 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 SRWn 位，以确定自己是作为发送方还是作为接收方。如果 SRWn 位为高，从机须设置成发送方，这样会置位 SIMnC1 寄存器的 HTXn 位。如果 SRWn 位为低，从机须设置成接收方，这样会清零 SIMnC1 寄存器的 HTXn 位。

I²C 总线数据和应答信号

在从机确认接收到从地址后，会进行 8 位宽度的数据传输。这个数据传输顺序是的高位在前，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。如果从机发送方没接收到来自主机接收方的应答信号，发送方将释放 SDAn 线，此时主机方可发出 STOP 信号以释放 I²C 总线。所传送的数据存储在 SIMnD 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 SIMnD 寄存器中；如果设置成接收方，从机必须从 SIMnD 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 (TXAKn)。被设为发送方的从机将检测寄存器 SIMnC1 中的 RXAKn 位以判断是否传输下一个字节的数据，如果从机不传输下一个字节，那么它将释放 SDAn 线并等待接收主机的停止信号。

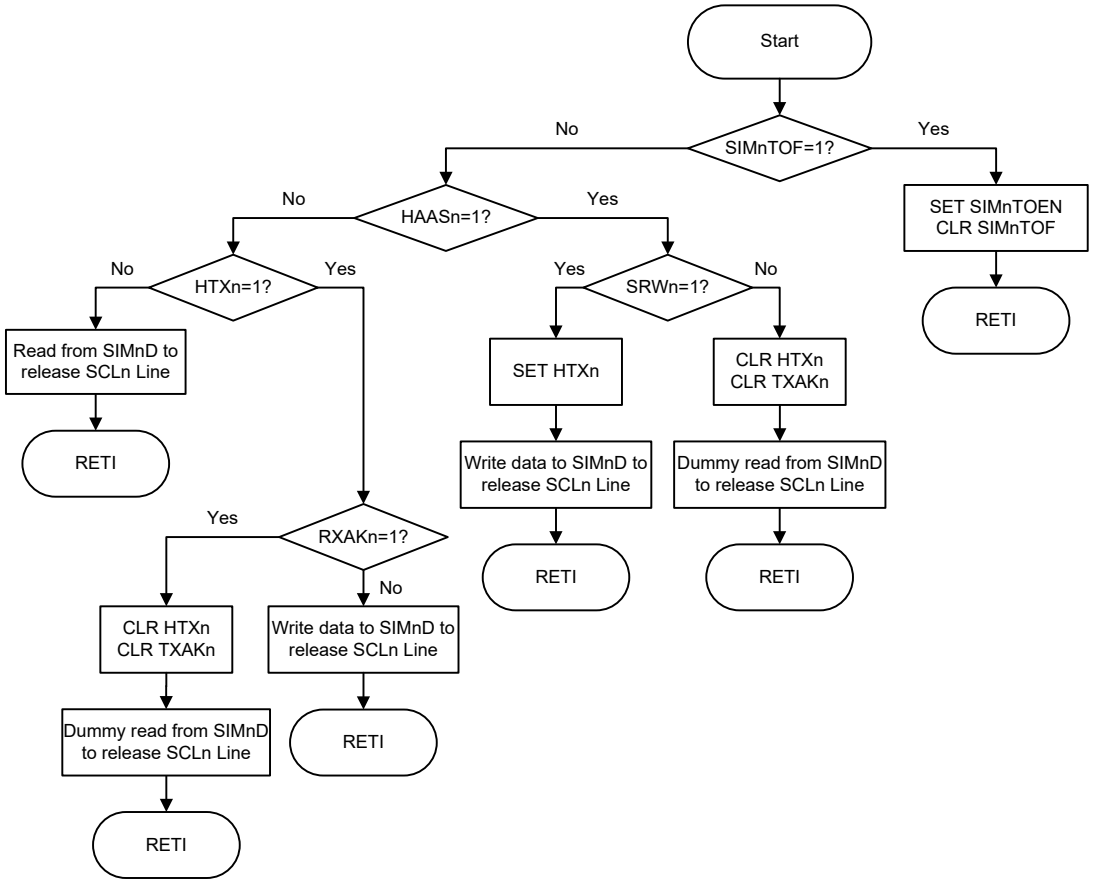


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRWn bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAKn bit for transmitter, TXAKn bit for receiver, 1 bit)
P=Stop (1 bit)

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 SIMnD 寄存器；若设置为接收模式，需立即从 SIMnD 寄存器中虚读数据以释放 SCLn 线。

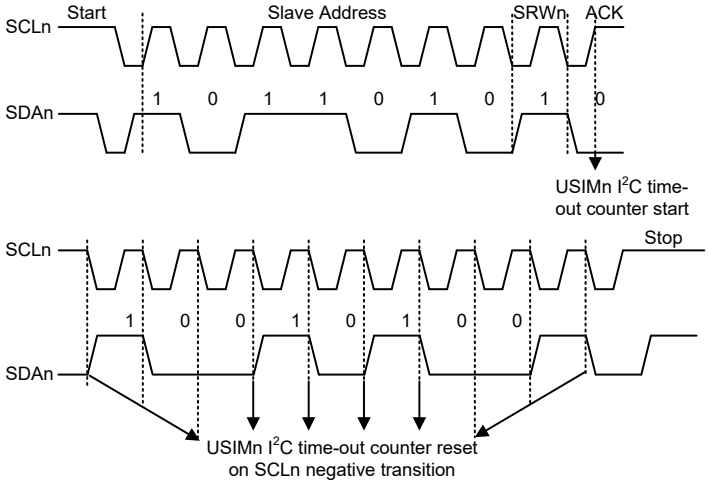
USIMn I²C 通信时序图



USIMn 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 SCLn 下降沿清零。在下一个 SCLn 下降沿到来之前，如果超时时间大于 SIMnTOC 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。



USIMn I²C 超时时序图

当 I²C 超时计数器溢出时，计数器将停止计数，SIMnTOEN 位被清零，且 SIMnTOF 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 USIMn 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
SIMnD，SIMnA，SIMnC0	保持不变
SIMnC1	复位至 POR

超时发生后的 USIMn I²C 寄存器

SIMnTOF 标志位由应用程序清零。共有 64 个超时周期，可通过 SIMnTOC 寄存器的 SIMnTOS[5:0] 位进行选择。超时周期可通过公式计算：$(1\sim64)\times(32/f_{SUB})$。由此可得超时周期范围为 1ms~64ms。

- **SIMnTOC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIMnTOEN	SIMnTOF	SIMnTOS5	SIMnTOS4	SIMnTOS3	SIMnTOS2	SIMnTOS1	SIMnTOS0
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 **SIMnTOEN**：USIMn I²C 超时控制位
0：除能
1：使能

Bit 6 **SIMnTOF**：USIMn I²C 超时标志位
0：超时未发生
1：超时发生
当发生超时，此位由硬件自动置位；此位必须通过应用程序清零。

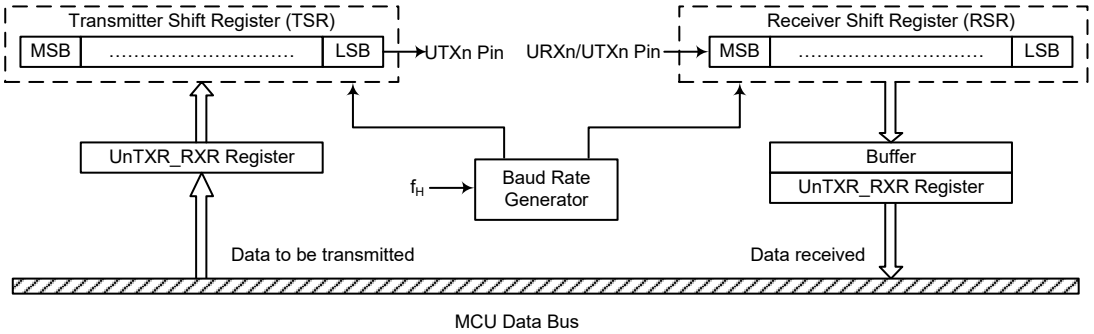
Bit 5~0 **SIMnTOS5~SIMnTOS0**：USIMn I^2C 超时时间选择位
I^2C 超时时钟源是 $f_{SUB}/32$。
I^2C 超时时间计算方法：(SIMnTOS[5:0]+1)×(32/f_{SUB})。

UART 接口

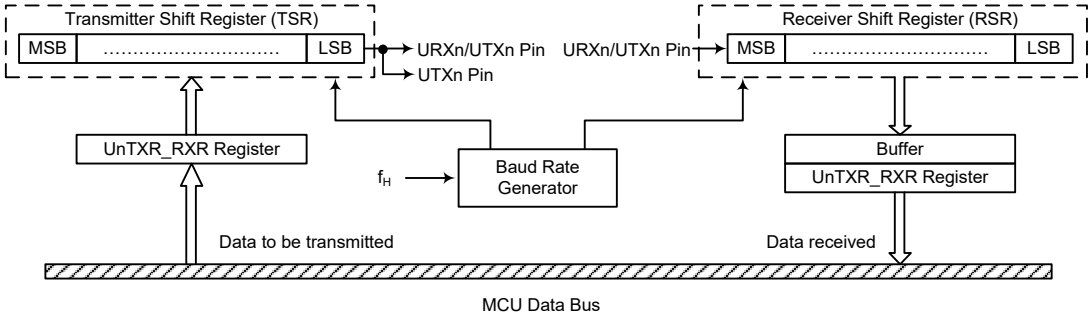
每个 USIM 模块都包含一个全双工或半双工的异步串行通信接口 – UART，可以很方便的与其它具有串行口的芯片通信。UART 具有许多功能特性，发送或接收串行数据时，将数据组成一个 8 位或 9 位的数据块，连同数据特征位一并传输。具有检测数据覆盖或帧错误等功能。USIMn UART 功能与 SPI 和 I^2C 接口共用一个内部中断向量，当接收到数据或数据发送结束，触发中断。

内置的 UART 功能包含以下特性：

- 全双工或半双工 (单线通信模式) 通用异步接收器 / 发送器
- 8 位或 9 位传输格式
- 奇校验、偶校验或无校验
- 1 位或 2 位停止位
- 8 位预分频的波特率发生器
- 奇偶、帧、噪声和溢出检测
- 支持地址检测中断 (最后一位 = 1)
- 独立的发送和接收使能
- 2-byte FIFO 接收缓冲器
- URXn/UTXn 引脚唤醒功能
- 发送和接收中断
- 中断可由下列条件触发：
  - ♦ 发送器为空
  - ♦ 发送器空闲
  - ♦ 接收完成
  - ♦ 接收器溢出
  - ♦ 地址检测



USIMn UART 数据传输方框图 – UnSWM=0 (n=0~1)



USIMn UART 数据传输方框图 – UnSWM=1 (n=0~1)

UART 外部引脚

内部 UART 有两个外部引脚 UTXn 和 URXn/UTXn，可与外部串行接口进行通信。UTXn 和 URXn/UTXn 与 I/O 口或其它功能共用引脚。在使用 USIMn UART 功能前，应先通过相应的引脚共用功能选择寄存器，选择 UTXn 和 URXn/UTXn 引脚功能。当 UnMD、URnEN、UnTXEN 和 UnRXEN 位置高时，将自动设置这些 I/O 脚或其它共用功能脚作为发送输出和接收输入。此时，用作发送输出的引脚其内部上拉电阻会被除能，而用作接收输入的引脚其内部上拉电阻由相应的上拉电阻控制位控制。当 UnMD、URnEN、UnTXEN 或 UnRXEN 位清零除能 UTXn 或 URXn/UTXn 引脚功能后，UTXn 或 URXn/UTXn 引脚将处于浮空状态。

UART 单线模式

UART 功能支持单线模式通信，通过 UnUCR3 寄存器中的 UnSWM 位选择。当设置该位为高，UART 将工作在单线模式。在单线模式下，单个 URXn/UTXn 引脚通过相关控制位的不同设置即可完成数据的发送与接收。设置 UnRXEN 位为高，URXn/UTXn 引脚用作接收引脚。将 UnRXEN 位清零，同时设置 UnTXEN 位为高，URXn/UTXn 引脚用作发送引脚。

在单线模式下建议不要将 UnRXEN 位和 UnTXEN 位同时设置为高。若 UnRXEN 位和 UnTXEN 位同时为高，UnRXEN 位具有更高的优先级，此时 UART 为接收器状态。

需特别注意的是，UART 章节所有内容是基于 UART 全双工通信来对 UART 功能进行描述，相关的说明除引脚的使用外，对半双工通信 (单线模式) 同样适用。在理解单线模式通信时，全双工通信中使用的 UTXn 引脚需替换为 URXn/UTXn 引脚。

在单线模式下，通过合理的软件配置，数据也可以在 UTXn 引脚发送。因此数据可通过 URXn/UTXn 和 UTXn 引脚输出。

UART 数据传输方案

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

数据在波特率发生器的控制下，低位在前高位在后，从外部引脚 URXn/UTXn 进入接收移位寄存器 RSR。当数据接收完成，数据从接收移位寄存器移入可被用户程序操作的 UnTXR_RXR 寄存器中。UnTXR_RXR 寄存器被映射到单片机

数据存储器中，而接收移位寄存器没有实际地址，所以接收移位寄存器不可直接操作。

需要注意的是，发送和接收都是共用同一个数据寄存器，即 UnTXR_RXR 寄存器。

UART 状态和控制寄存器

与 USIMn UART 功能相关的有七个寄存器，SIMnC0 寄存器中的 UnMD 位用于选择 USIMn UART 接口功能。UnUCR3 寄存器中的 UnSWM 位用于使能 / 除能 UART 单线模式。其它包括控制 USIMn UART 整体功能的 UnUSR、UnUCR1 和 UnUCR2 寄存器，控制波特率的 UnBRG 寄存器，管理发送和接收数据的数据寄存器 UnTXR_RXR。注意，只有在 SIMnC0 寄存器中的 UnMD 位设置为“1”后，USIMn UART 相关的寄存器以及它们的上电复位值才有效。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SIMnC0	SIMn2	SIMn2	SIMn0	UnMD	SIMnDEB1	SIMnDEB0	SIMnEN	SIMnICF
UnUSR	UnPERR	UnNF	UnFERR	UnOERR	UnRIDLE	UnRXIF	UnTIDLE	UnTXIF
UnUCR1	URnEN	UnBNO	UnPREN	UnPRT	UnSTOPS	UnTXBRK	UnRX8	UnTX8
UnUCR2	UnTXEN	UnRXEN	UnBRGH	UnADDEN	UnWAKE	UnRIE	UnTIIE	UnTEIE
UnUCR3	—	—	—	—	—	—	—	UnSWM
UnTXR_RXR	UnTXRX7	UnTXRX6	UnTXRX5	UnTXRX4	UnTXRX3	UnTXRX2	UnTXRX1	UnTXRX0
UnBRG	UnBRG7	UnBRG6	UnBRG5	UnBRG4	UnBRG3	UnBRG2	UnBRG1	UnBRG0

USIMn UART 寄存器列表 (n=0~1)

- **SIMnC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SIMn2	SIMn1	SIMn0	UnMD	SIMnDEB1	SIMnDEB0	SIMnEN	SIMnICF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	0	0	0

Bit 7~5 **SIMn2~SIMn0**：USIMn SPI/I²C 模式控制位
当 UnMD 位清零时，这几位用于设置 USIMn SPI/I²C 功能的工作模式。更多细节详见 SPI 或 I²C 寄存器章节。

Bit 4 **UnMD**：USIMn UART 模式选择位
0：USIMn SPI/I²C 模式
1：USIMn UART 模式
此位为 USIMn UART 模式选择位。当当此位清零时，选择 SPI/I²C 模式，而对 SPI 或 I²C 模式的选择是通过 SIMn2~SIMn0 位实现。

Bit 3~2 **SIMnDEB1~SIMnDEB0**：USIMn I²C 去抖时间选择位
详见 I²C 寄存器章节。

Bit 1 **SIMnEN**：USIMn SPI/I²C 控制位
0：除能
1：使能
此位仅当 UnMD 位设置为“0”选择 SPI 或 I²C 模式时才有效。详见 SPI 或 I²C 寄存器章节。

Bit 0 **SIMnICF**：USIMn SPI 未完成标志位
详见 SPI 寄存器章节。

● **UnUSR 寄存器**

寄存器 UnUSR 是 USIMn UART 的状态寄存器，可以通过程序读取以得知当前 UART 状态。所有 UnUSR 位是只读的。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	UnPERR	UnNF	UnFERR	UnOERR	UnRIDLE	UnRXIF	UnTIDLE	UnTXIF
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

Bit 7 **UnPERR**：奇偶校验出错标志位
0：奇偶校验正确
1：奇偶校验出错

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

Bit 6 **UnNF**：噪声干扰标志位
0：没有受到噪声干扰
1：受到噪声干扰

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

Bit 5 **UnFERR**：帧错误标志位
0：无帧错误发生
1：有帧错误发生

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

Bit 4 **UnOERR**：溢出错误标志位
0：无溢出错误发生
1：有溢出错误发生

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

Bit 3 **UnRIDLE**：接收状态标志位
0：正在接收数据
1：接收器空闲

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

Bit 2 **UnRXIF**：接收寄存器状态标志位
0：UnTXR_RXR 寄存器为空
1：UnTXR_RXR 寄存器含有有效数据

UnRXIF 是接收寄存器状态标志位。当 UnRXIF=0，UnTXR_RXR 寄存器为空；当 UnRXIF=1，UnTXR_RXR 寄存器接收到新数据。当数据从移位寄存器加载到 UnTXR_RXR 寄存器中，如果 UnUCR2 寄存器中的 UnRIE=1，则会触发中断。当接收数据时检测到一个或多个错误时，相应的标志位 UnNF、UnFERR 或 UnPERR 会在同一周期内置位。读取 UnUSR 寄存器再读 UnTXR_RXR 寄存器，如果 UnTXR_RXR 寄存器中没有新的数据，那么将清除 UnRXIF 标志。

Bit 1 **UnTIDLE**：数据发送完成标志位
0：数据传输中
1：无数据传输

UnTIDLE 是数据发送完成标志位。若 UnTIDLE=0，数据传输中。当 UnTXIF=1

且数据发送完毕或者暂停字被发送时，UnTIDLE 置位。UnTIDLE=1，UTXn 引脚空闲且处于逻辑高状态。读取 UnUSR 寄存器再写 UnTXR_RXR 寄存器将清除 UnTIDLE 位。数据字符或暂停字就绪时，不会产生该标志位。

Bit 0 **UnTXIF**：发送数据寄存器 UnTXR_RXR 状态位
0：数据还没有从缓冲器加载到移位寄存器中
1：数据已从缓冲器加载到移位寄存器中 (UnTXR_RXR 数据寄存器为空)
UnTXIF 是发送数据寄存器为空标志位。若 UnTXIF=0，数据还没有从缓冲器加载到移位寄存器中；若 UnTXIF=1，数据已从缓冲器中加载到移位寄存器中。读取 UnUSR 存器再写 UnTXR_RXR 寄存器将清除 UnTXIF。当 UnTXEN 被置位，由于发送缓冲器未满，UnTXIF 也会被置位。

● UnUCR1 寄存器

UnUCR1、UnUCR2 和 UnUCR3 是 USIMn UART 的三个控制寄存器，用来定义各种 UART 功能，例如 UART 的使能与除能、奇偶校验控制、传输数据的长度以及单线模式通信等等。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	URnEN	UnBNO	UnPREN	UnPRT	UnSTOPS	UnTXBRK	UnRX8	UnTX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

“x”：未知

Bit 7 **URnEN**：USIMn UART 功能使能位
0：UART 除能，UTXn 和 URXn/UTXn 引脚处于浮空状态
1：UART 使能，UTXn 和 URXn/UTXn 引脚作为 UART 功能引脚
此位为 USIMn UART 的使能位。URnEN=0，UART 除能，URXn/UTXn 和 UTXn 处于浮空状态；URnEN=1，若 UnMD 位置高，UART 使能，UTXn 和 URXn/UTXn 将分别由 UnSWM 模式选择位、UnTXEN 和 UnRXEN 控制。当 UART 被除能将清除缓冲器，所有缓冲器中的数据将被忽略，另外波特率计数器、错误和状态标志位被复位，UnTXEN、UnRXEN、UnTXBRK、UnRXIF、UnOERR、UnFERR、UnPERR 和 UnNF 清零，而 UnTIDLE、UnTXIF 和 UnRIDLE 置位，UnUCR1、UnUCR2、UnUCR3 和 UnBRG 寄存器中的其它位保持不变。若 UART 工作时 URnEN 清零，所有发送和接收将停止，接口也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

Bit 6 **UnBNO**：发送数据位数选择位
0：8-bit 传输数据
1：9-bit 传输数据
UnBNO 是发送数据位数选择位。UnBNO=1，传输数据为 9 位；UnBNO=0，传输数据为 8 位。若选择了 9 位数据传输格式，UnRX8 和 UnTX8 将分别存储接收和发送数据的第 9 位。

Bit 5 **UnPREN**：奇偶校验使能位
0：奇偶校验除能
1：奇偶校验使能
此位为奇偶校验使能位。UnPREN=1，使能奇偶校验；UnPREN=0，除能奇偶校验。

Bit 4 **UnPRT**：奇偶校验选择位
0：偶校验
1：奇校验
奇偶校验选择位。UnPRT=1，奇校验；UnPRT=0，偶校验。

Bit 3 **UnSTOPS**：发送器停止位的长度选择位
0：有一位停止位
1：有两位停止位
此位用来设置发送器停止位的长度。UnSTOPS=1，有两位停止位；UnSTOPS=0，只有一位停止位。

Bit 2 **UnTXBRK**：暂停字发送控制位
0：没有暂停字要发送
1：发送暂停字
UnTXBRK 是暂停字发送控制位。UnTXBRK=0，没有暂停字要发送，UTXn 引脚正常操作；UnTXBRK=1，将会发送暂停字，发送器将发送逻辑“0”。若 UnTXBRK 为高，缓冲器中数据发送完毕后，发送器输出将至少保持 13 位宽的低电平直至 UnTXBRK 复位。

Bit 1 **UnRX8**：接收 9-bit 数据传输格式中的第 9 位 (只读)
此位只有在传输数据为 9 位的格式中有效，用来存储接收数据的第 9 位。UnBNO 是用来控制传输位数是 8 位还是 9 位。

Bit 0 **UnTX8**：发送 9-bit 数据传输格式中的第 9 位 (只写)
此位只有在传输数据为 9 位的格式中有效，用来存储发送数据的第 9 位。UnBNO 是用来控制传输位数是 8 位还是 9 位。

- **UnUCR2 寄存器**

UnUCR2 是 USIMn UART 的第二个控制寄存器，它的主要功能是控制发送器、接收器以及各种 USIMn UART 模式中断源的使能或除能。它也可用来控制波特率，使能接收唤醒和地址侦测。详细解释如下：

Bit	7	6	5	4	3	2	1	0
Name	UnTXEN	UnRXEN	UnBRGH	UnADDEN	UnWAKE	UnRIE	UnTIIE	UnTEIE
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 **UnTXEN**：USIMn UART 发送使能位
0：UART 发送除能
1：UART 发送使能
此位为发送使能位。UnTXEN=0，发送将被除能，发送器立刻停止工作。另外发送缓冲器将被复位，此时 UTXn 引脚将处于浮空状态。若 UnTXEN=1 且 UnMD=1 及 URnEN=1，则发送将被使能，UTXn 引脚将由 UART 来控制。在数据传输时清除 UnTXEN 将中止数据发送且复位发送器，此时 UTXn 引脚将处于浮空状态。

Bit 6 **UnRXEN**：USIMn UART 接收使能位
0：UART 接收除能
1：UART 接收使能
此位为接收使能位。UnRXEN=0，接收将被除能，接收器立刻停止工作。另外接收缓冲器将被复位，此时 URXn/UTXn 引脚将处于浮空状态。若 UnRXEN=1 且 UnMD=1 及 URnEN=1，则接收将被使能，URXn/UTXn 引脚将由 UART 来控制。在数据传输时清除 UnRXEN 将中止数据接收且复位接收器，此时 URXn/UTXn 引脚将处于浮空状态。

Bit 5 **UnBRGH**：波特率发生器高低速选择位
0：低速波特率
1：高速波特率
此位为波特率发生器高低速选择位，它和 UnBRG 寄存器一起控制 USIMn UART 的波特率。UnBRGH=1，为高速模式；UnBRGH=0，为低速模式。

Bit 4 **UnADDEN**：地址检测使能位
0：地址检测除能
1：地址检测使能
此位为地址检测使能和除能位。UnADDEN=1，地址检测使能，此时数据的第 8 位 (UnBNO=0) 或第 9 位 (UnBNO=1) 为高，那么接到的是地址而非数据。若相应的中断使能且接收到的值最高位为 1，那么中断请求标志将会被置位，若地址检测功能使能且最高位为 0，那么将不会产生中断且收到的数据也会被忽略。

Bit 3 **UnWAKE**：URXn/UTXn 引脚下降沿唤醒 USIMn UART 功能使能位
0：URXn/UTXn 引脚下降沿唤醒 UART 功能除能
1：URXn/UTXn 引脚下降沿唤醒 UART 功能使能

此位用于控制 URXn/UTXn 引脚下降沿时是否唤醒 USIMn UART 功能。此位仅当 UART 时钟源 f_H 关闭时有效。若 UART 时钟源 f_H 还开启，则无 URXn/UTXn 引脚唤醒 UART 功能无效。若此位置高且 UART 时钟 f_H 关闭，当 URXn/UTXn 引脚发生下降沿时会产生 UART 唤醒请求。若相应的中断使能，将产生 URXn/UTXn 引脚唤醒 UART 的中断，以告知单片机使其通过应用程序开启 UART 时钟源 f_H，从而唤醒 UART 功能。否则，若此位为低，即使 URXn/UTXn 引脚发生下降沿也无法恢复 UART 功能。

Bit 2 **UnRIE**：接收中断使能位
0：接收中断除能
1：接收中断使能
此位为接收中断使能或除能位。若 UnRIE=1，当 UnOERR 或 UnRXIF 置位时，USIMn 的中断请求标志 USIMnF 置位；若 UnRIE=0，USIMn 中断请求标志 USIMnF 不受 UnOERR 和 UnRXIF 影响。

Bit 1 **UnTIIE**：发送器空闲中断使能位
0：发送器空闲中断除能
1：发送器空闲中断使能
此位为发送器空闲中断的使能或除能位。若 UnTIIE=1，当发送器空闲触发 UnTIDLE 置位时，USIMn 的中断请求标志 USIMnF 置位；若 UnTIIE=0，USIMn 中断请求标志 USIMnF 不受 UnTIDLE 的影响。

Bit 0 **UnTEIE**：发送寄存器为空中断使能位
0：发送寄存器为空中断除能
1：发送寄存器为空中断使能
此位为发送寄存器为空中断的使能或除能位。若 UnTEIE=1，当发送器为空触发 UnTXIF 置位时，USIMn 的中断请求标志 USIMnF 置位；若 UnTEIE=0，USIMn 中断请求标志 USIMnF 不受 UnTXIF 的影响。

● UnUCR3 寄存器

UnUCR3 寄存器用于使能 USIMn UART 单线模式通信。顾名思义，在单线模式下 UART 只需要使用一条线，URXn/UTXn，在 UnUCR2 寄存器中的 UnRXEN 和 UnTXEN 位控制下即可完成通信。

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

Bit 7~1 未定义，读为“0”

Bit 0 **UnSWM**：单线模式使能控制
0：除能，URXn/UTXn 引脚仅用作 UART 接收功能
1：使能，URXn/UTXn 引脚在 UnRXEN 和 UnTXEN 位控制下可用作接收或发送功能
需注意的是，单线模式使能时，若将 UnRXEN 和 UnTXEN 位同时设置为高，URXn/UTXn 引脚用作接收功能。

● UnTXR_RXR 寄存器

UnTXR_RXR 是一个数据寄存器，用来存储 UTXn 引脚将要发送或 URXn/UTXn 引脚正在接收的数据。

Bit	7	6	5	4	3	2	1	0
Name	UnTXRX7	UnTXRX6	UnTXRX5	UnTXRX4	UnTXRX3	UnTXRX2	UnTXRX1	UnTXRX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”：未知

Bit 7~0 **UnTXRX7~UnTXRX0**：UART 发送 / 接收数据位 Bit 7~Bit 0

● **UnBRG 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	UnBRG7	UnBRG6	UnBRG5	UnBRG4	UnBRG3	UnBRG2	UnBRG1	UnBRG0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”：未知

Bit 7~0 **UnBRG7~UnBRG0**：波特率值

软件设置 UnUCR2 寄存器中的 UnBRGH 位 (设置波特率发生器的速度) 和 UnBRG 寄存器 (设置波特率的值)，一起控制 UART 的波特率。

注：若 UnBRGH=0，波特率 = f_H/[64×(N+1)]；

若 UnBRGH=1，波特率 = f_H/[16×(N+1)]。

波特率发生器

UART 自身具有一个波特率发生器，通过它可以设定数据传输速率。波特率是由一个独立的内部 8 位计数器产生，它由 UnBRG 寄存器和 UnUCR2 寄存器的 UnBRGH 位来控制。UnBRGH 是决定波特率发生器处于高速模式还是低速模式，从而决定计算公式的选用。UnBRG 寄存器的值 N 可根据下表中的公式计算，N 的范围是 0 到 255。

UnBRGH 位	0	1
波特率 (BR)	f_H/[64×(N+1)]	f_H/[16×(N+1)]

为得到相应的波特率，首先需要设置 UnBRGH 来选择相应的计算公式从而算出 UnBRG 的值。由于 UnBRG 的值不连续，所以实际波特率和理论值之间有一个偏差。

下面举例怎样计算 UnBRG 寄存器中的值 N 和误差。

波特率和误差的计算

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

根据上表，波特率 BR=f_H/[64 (N+1)]

转换后的公式 N=[f_H/(BR×64)] - 1

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

取最接近的值，十进制 12 写入 UnBRG 寄存器，实际波特率如下

BR=4000000/[64×(12+1)]=4808

因此，误差 = (4808 - 4800)/4800=0.16%

UART 的设置与控制

UART 采用标准的不归零码传输数据，这种方法通常被称为 NRZ 法。它由 1 位起始位，8 位或 9 位数据位和 1 位或者两位停止位组成。奇偶校验是由硬件自动完成的，可设置成奇校验、偶校验和无校验三种格式。常用的数据传输格式由 8 位数据位，1 位停止位，无校验组成，用 8、N、1 表示，它是系统上电的默认格式。数据位数、停止位数和奇偶校验由 UnUCR1 寄存器的 UnBNO、UnPRT、UnPREN 和 UnSTOPS 设定。用于数据发送和接收的波特率由一个内部的 8 位波特率发送器产生，数据传输时低位在前高位在后。尽管 UART 发送器和接收器在功能上相互独立，但它们使用相同的数据传输格式和波特率，在任何情况下，停止位是必须的。

UART 的使能和除能

UART 是由 UnUCR1 寄存器的 URnEN 位来使能和除能的。当 SIMnC0 寄存器中的 UnMD 位已设置为“1”选择 USIMn UART 模式，若 URnEN、UnTXEN 和 UnRXEN 都为高，则 UTXn 和 URXn/UTXn 分别为 UART 的发送端口和接收端口。若没有数据发送，UTXn 引脚默认状态为高电平。

URnEN 清零将除能 UTXn 和 URXn/UTXn，通过设置相关引脚共用控制位，这两个引脚可用作普通 I/O 口或其它引脚共用功能。当 UART 被除能时将清空缓冲器，所有缓冲器中的数据将被忽略，另外一些使能控制、错误标志和状态标志将被复位，如 UnTXEN、UnRXEN、UnTXBRK、UnRXIF、UnOERR、UnFERR、UnPERR 和 UnNF 清零，而 UnTIDLE、UnTXIF 和 UnRIDLE 置位，UnUCR1、UnUCR2、UnUCR3 和 UnBRG 寄存器中的其它位保持不变。若 UART 工作时 URnEN 清零，所有发送和接收将停止，接口也将复位成上述状态。当 UART 再次使能时，它将在上次配置下重新工作。

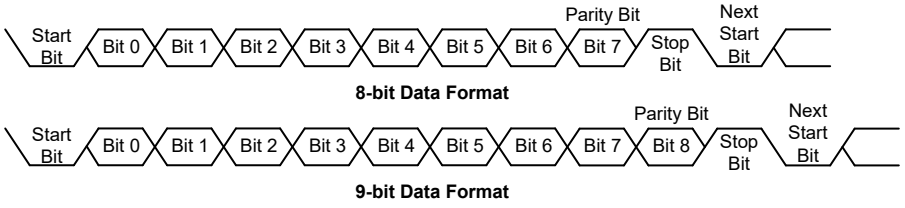
数据位、停止位位数以及奇偶校验的选择

数据传输格式由数据长度、是否校验、校验类型、地址位以及停止位长度组成。它们都是由 UnUCR1 寄存器的各个位控制的。UnBNO 决定数据传输是 8 位还是 9 位；UnPRT 决定校验类型；UnPREN 决定是否选择奇偶校验；而 UnSTOPS 决定选用 1 位还是 2 位停止位。下表列出了各种数据传输格式。若地址检测功能使能，地址位，即数据字节的最高位，用来确定此帧是地址还是数据。停止位的长度和数据位的长度无关，且只有发送器需设置停止位长度。接收器只接收一个停止位。

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

发送和接收数据格式

下图是传输 8 位和 9 位数据的波形。



UART 发送器

UnUCR1 寄存器的 UnBNO 位是控制数据传输的长度。UnBNO=1 其长度为 9 位，第 9 位 MSB 存储在 UnUCR1 寄存器的 UnTX8 中。发送器的核心是发送移位寄存器 TSR，它的数据由发送寄存器 UnTXR_RXR 提供，应用程序只须将发送数据写入 UnTXR_RXR 寄存器。上组数据的停止位发出前，TSR 寄存器禁止写入。如果还有新的数据要发送，一旦停止位发出，待发数据将会从 UnTXR_RXR 寄

存器加载到 TSR 寄存器。TSR 不像其它寄存器一样映射到数据存储器，所以应用程序不能对其进行读写操作。UnTXEN=1，发送使能，但若 UnTXR_RXR 寄存器没有数据或者波特率没有设置，发送器将不会工作。先写 UnTXR_RXR 寄存器再置高 UnTXEN 也会触发发送。当发送器使能，若 TSR 寄存器为空，数据写入 UnTXR_RXR 寄存器将会直接加载到 TSR 寄存器中。发送器工作时，UnTXEN 清零，发送器将立刻停止工作并且复位，此时通过设置相关引脚共用控制位，UTXn 引脚用作普通 I/O 口或其它引脚共用功能。

发送数据

当 UART 发送数据时，数据从移位寄存器中移到 UTXn 引脚上，其低位在前高位在后。在发送模式中，UnTXR_RXR 寄存器在内部总线和发送移位寄存器间形成一个缓冲。如果选择 9 位数据传输格式，最高位 MSB 取自 UnUCR1 寄存器的 UnTX8。

发送器的启动可由如下步骤完成：

- 正确地设置 UnBNO、UnPRT、UnPREN 和 UnSTOPS 位以确定数据长度、校验类型和停止位长度。
- 设置 UnBRG 寄存器，选择期望的波特率。
- 置高 UnTXEN，使能 UART 发送器且使 UTXn 作为 UART 的发送端。
- 读取 UnUSR 寄存器，然后将待发数据写入 UnTXR_RXR 寄存器。注意，此步骤会清除 UnTXIF 标志位。

如果要发送多个数据只需重复上一步骤。

当 UnTXIF=0 时，数据将禁止写入 UnTXR_RXR 寄存器。可以通过以下步骤来清除 UnTXIF：

1. 读取 UnUSR 寄存器
2. 写 UnTXR_RXR 寄存器

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

可以通过以下步骤来清除 UnTIDLE：

1. 读取 UnUSR 寄存器
2. 写 UnTXR_RXR 寄存器

清除 UnTXIF 和 UnTIDLE 软件执行次序相同。

发送暂停字

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

UART 接收器

UART 接收器支持 8 位或者 9 位数据接收。若 UnBNO=1，数据长度为 9 位，而最高位 MSB 存放在 UnUCR1 寄存器的 UnRX8 中。接收器的核心是串行移位寄存器 RSR。URXn/UTXn 引脚上的数据送入数据恢复器中，它在 16 倍波特率的频率下工作，而串行移位器工作在正常波特率下。当在 URXn/UTXn 引脚上检测到停止位，若 UnTXR_RXR 寄存器为空，数据从 RSR 寄存器中加载到 UnTXR_RXR 寄存器。URXn/UTXn 引脚上的每一位数据会被采样三次以判断其逻辑状态。RSR 不像其它寄存器一样映射在数据存储器，所以应用程序不能对其进行读写操作。

接收数据

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

接收器的启动可由如下步骤完成：

- 正确地设置 UnBNO、UnPRT 和 UnPREN 位以确定数据长度和校验类型。
- 设置 UnBRG 寄存器，选择期望的波特率。

置高 UnRXEN，使能 UART 接收器且使 URXn/UTXn 作为 UART 的接收端。

此时接收器被使能并检测起始位。

接收数据将会发生如下事件：

当 UnTXR_RXR 寄存器中包含有效数据时，UnUSR 寄存器中的 UnRXIF 位将会置位，溢出错误发生之前至多还有一帧数据可读。

若 UnRIE=1，数据从 RSR 寄存器加载到 UnTXR_RXR 寄存器中将产生中断。

若接收器检测到帧错误、噪声干扰错误、奇偶出错或溢出错误，那么相应的错误标志位置位。

可以通过如下步骤来清除 UnRXIF：

1. 读取 UnUSR 寄存器
2. 读取 UnTXR_RXR 寄存器

接收暂停字

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

UART 接收到暂停字会产生以下事件：

- 帧错误标志位 UnFERR 置位。
- UnTXR_RXR 寄存器清零。

● UnOERR、UnNF、UnPERR、UnRIDLE 或 UnRXIF 可能会置位。

空闲状态

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

接收中断

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

接收错误处理

UART 会产生几种接收错误，下面部分将描述各错误以及怎样处理。

溢出 – UnOERR 标志

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

产生溢出错误时将会发生以下事件：

● UnUSR 寄存器中 UnOERR 被置位。

● UnTXR_RXR 寄存器中数据不会丢失。

● RSR 寄存器数据将会被覆盖。

● 若 UnRIE=1，将会产生中断。

先读取 UnUSR 寄存器再读取 UnTXR_RXR 寄存器可将 UnOERR 清零。

噪声干扰 – UnNF 标志

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

● 在 UnRXIF 上升沿，UnUSR 寄存器中只读标志位 UnNF 置位。

● 数据从 RSR 寄存器加载到 UnTXR_RXR 寄存器中。

● 不产生中断，但此位置位发生在 UnRXIF 置位产生中断的同周期内。

先读取 UnUSR 寄存器再读取 UnTXR_RXR 寄存器可将 UnNF 清零。

帧错误 – UnFERR 标志

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

奇偶校验错误 – UnPERR 标志

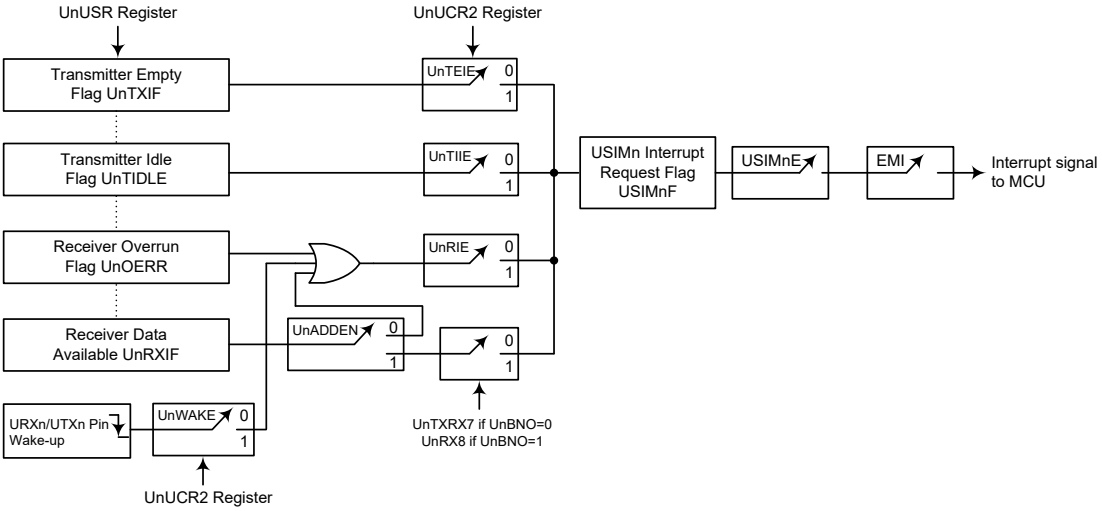
若接收到数据出现奇偶校验错误，UnUSR 寄存器中只读标志 UnPERR 置位。只有使能了奇偶校验，选择了校验类型，此标志位才有效。此标志位同接收的数据分别记录在 UnUSR 寄存器和 UnTXR_RXR 寄存器中，此标志位可被任何复位清零。注意，在读取相应的数据之前必须先访问 UnUSR 寄存器中的 UnFERR 和 UnPERR 错误标志位。

UART 中断结构

几个独立的 UART 条件可以产生一个 USIMn 中断。当条件满足时，会产生一个低脉冲信号。发送寄存器为空、发送器空闲、接收器数据有效、溢出和地址检测和 URXn/UTXn 引脚唤醒都会产生中断。若总中断使能、USIMn 中断允许且堆栈未满，程序将会跳转到相应的中断向量执行中断服务程序，而后再返回主程序。其中四种情况，若其 UnUCR2 寄存器中相应中断允许位被置位，则 UnUSR 寄存器中对应中断标志位将产生 USIMn 中断。发送器相关的两个中断情况有各自对应的中断允许位，而接收器相关的两个中断情况共用一个中断允许位。这些允许位可用于禁止个别的 USIMn UART 模式中断源。

地址检测也是 USIMn UART 模式的中断源，它没有相应的标志位，若 UnUCR2 寄存器中 UnADDEN=1，当检测到地址将会产生 USIMn 中断。URXn/UTXn 引脚唤醒也可以产生 USIMn 中断，它没有相应的标志位，当 UART 时钟源 f_H 关闭且 UnUCR2 中的 UnWAKE 和 UnRIE 位被置位，URXn/UTXn 引脚上有下降沿时会产生 USIMn 中断。

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



USIMn UART 中断框图

地址检测模式

置位 UnUCR2 寄存器中的 UnADDEN 将启动地址检测模式。若此位为“1”，可产生接收数据有效中断，其请求标志位为 UnRXIF。若 UnADDEN 有效，只有在接收到数据最高位为 1 才会产生中断，中断允许位 USIMnE 和 EMI 也要使能才会产生中断。地址的最高位为第 9 位 (UnBNO=1) 或第 8 位 (UnBNO=0)，若此位为高，则接收到的是地址而非数据。只有接收的数据的最后一位为高才会产生中断。若 UnADDEN 除能，每接收到一个有效数据便会置位 UnRXIF，而不用考虑数据的最后一位。地址检测和奇偶校验在功能上相互排斥，若地址检测模式使能，为了确保操作正确，必须将奇偶校验使能位清零以除能奇偶校验。

UnADDEN	9th Bit (UnBNO=1) 8th Bit (UnBNO=0)	产生 USIMn 中断
0	0	√
	1	√
1	0	×
	1	√

UnADDEN 位功能

UART 暂停和唤醒

UART 时钟 f_H 关闭后 UART 接口将停止运行。当传送数据时 UART 时钟 f_H 关闭，发送将停止直到 UART 接口时钟再次使能。同样地，当接收数据时单片机进入空闲或休眠模式，数据接收也会停止。当单片机进入空闲或休眠模式，UnUSR、UnUCR1、UnUCR2、UnUCR3、UnTXR_RXR 以及 UnBRG 寄存器都不会受到影响。建议在单片机进入空闲或休眠模式前先确保数据发送或接收已完成。

UART 功能中包括了 URXn/UTXn 引脚的唤醒功能，由 UnUCR2 寄存器中 UnWAKE 位控制。当 UART 时钟 f_H 关闭时，若 UnWAKE 位与 UART 模式选择位 UnMD、UART 允许位 URnEN、接收器使能位 UnRXEN 和接收器中断使能位 UnRIE 都被置位，则 URXn/UTXn 引脚的下降沿可触发产生 URXn/UTXn 引脚唤醒 UART 的中断。唤醒后系统需延时一段时间才能正常工作，在此期间，URXn/UTXn 引脚上的任何数据将被忽略。

若要产生唤醒 USIMn UART 的中断，除了唤醒使能控制位和接收中断使能控制位需置位外，全局中断允许位 EMI 和 USIMn 中断使能控制位 USIMnE 也必须置位；若这两控制位没有被置位，那么，可发生唤醒事件但不会产生中断。唤醒后系统需一定的延时才能正常工作，然后才会产生 USIMn 中断。

SPI 串行接口

该单片机内含一个独立的 SPI 功能。重要的是，不要将此 SPI 功能与 USIM 模块中的 SPI 功能混淆，其具体描述详见规格书的另一章节。

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。

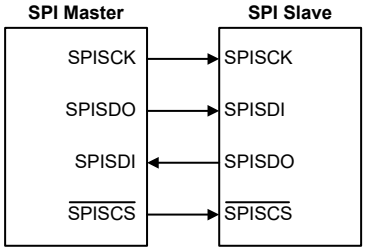
SPI 通信模式为全双工模式，且能以主 / 从模式的工作方式进行通信，单片机既可以作为主机，也可以作为从机。虽然 SPI 接口理论上允许一个主机控制多个从机，但此处的 SPI 中只有一个片选信号引脚 $\overline{\text{SPISCS}}$。若主机需要控制多个从机，可使用输入 / 输出引脚选择从机。

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：SPISDI、SPISDO、SPISCK 和 $\overline{\text{SPISCS}}$。SPISDI 和 SPISDO 是数据的输入和输出线。SPISCK 是串行时钟线，$\overline{\text{SPISCS}}$ 是从机的选择线。SPI 的接口引脚与其它功能共用引脚。通过设定引脚共用功能选择寄存器的对应位，来选择 SPI 接口引脚。SPI 接口可以通过 SPIC0 寄存器中的 SPIEN 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，且所有的数据传输由主机发起，时钟信号也由主机控制。由于单片机只有一个 $\overline{\text{SPISCS}}$ 引脚，所以只能拥有一个从机设备。若 SPI 功能使能且引脚用作 SPI 输入脚，可通过对应上拉电阻控制寄存器选择此脚的

上拉电阻。

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

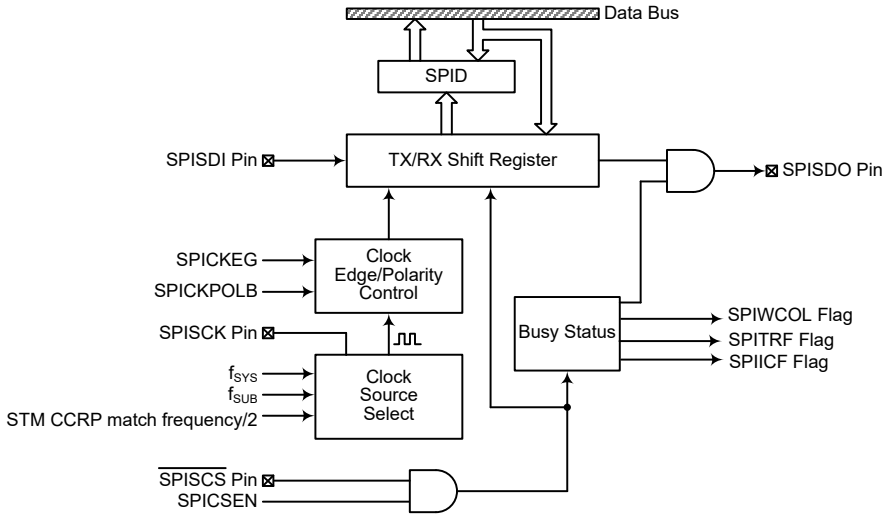


SPI 主 / 从机连接方式

该单片机的 SPI 功能具有以下特点：

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

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



SPI 方框图

SPI 寄存器

有三个寄存器用于控制 SPI 接口的所有操作，其中有一个数据寄存器 SPID、两个控制寄存器 SPIC0 和 SPIC1。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SPIC0	SPIM2	SPIM1	SPIM0	—	—	—	SPIEN	SPIICF
SPIC1	—	—	SPICKPOLB	SPICKEG	SPIMLS	SPICSEN	SPIWCOL	SPITRF
SPID	D7	D6	D5	D4	D3	D2	D1	D0

SPI 寄存器列表

SPI 数据寄存器

SPID 用于存储发送和接收的数据。在单片机将数据写入到 SPI 总线之前，要传输的数据应先存在 SPID 中。SPI 总线接收到数据之后，单片机就可以从 SPID 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 SPID 实现。

- **SPID 寄存器**

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

“x”：未知

Bit 7~0 **D7~D0**：SPI 数据寄存器位 bit 7 ~ bit 0

SPI 控制寄存器

单片机中也有两个控制 SPI 接口功能的寄存器，SPIC0 和 SPIC1。寄存器 SPIC0 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 SPIC1 用于其它的控制功能如 LSB/MSB 选择，写冲突标志位等。

- **SPIC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SPIM2	SPIM1	SPIM0	—	—	—	SPIEN	SPIICF
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	1	1	1	—	—	—	0	0

Bit 7~5 **SPIM2~SPIM0**：SPI 工作模式控制位
000：SPI 主机模式；SPI 时钟为 $f_{SYS}/4$
001：SPI 主机模式；SPI 时钟为 $f_{SYS}/16$
010：SPI 主机模式；SPI 时钟为 $f_{SYS}/64$
011：SPI 主机模式；SPI 时钟为 f_{SUB}
100：SPI 主机模式；SPI 时钟为 STM CCRP 匹配频率 / 2
101：SPI 从机模式
110：SPI 除能
111：SPI 除能
这几位用于设置 SPI 的主从模式和 SPI 的主机时钟频率。SPI 时钟源可来自于系统时钟和 f_{SUB} 也可以选择来自 STM。若选择的是作为 SPI 从机，则其时钟源从外部主机而得。

Bit 4~2 未定义，读为“0”

Bit 1 **SPIEN**：SPI 控制位
0：除能
1：使能
此位为 SPI 接口的开 / 关控制位。此位为“0”时，SPI 接口除能，SPISDI、SPISDO、SPISCK 和 $\overline{\text{SPISCS}}$ 脚将失去 SPI 功能，SPI 工作电流减小到最小值。此位为“1”时，SPI 接口使能。

Bit 0 **SPIICF**：SPI 未完成标志位
0：未发生
1：发生
此位仅当 SPI 配置在 SPI 从机模式时有效。如果 SPI 工作在从机模式且 SPIEN 和 SPICSEN 位都为“1”，但在 SPI 数据传输完全结束前 $\overline{\text{SPISCS}}$ 线被外部主机拉高，SPIICF 和 SPITRF 位都会被置高。在这种情况下，如果相应的中断功能使能将产生一个中断。然而，如果 SPIICF 位是由软件应用程序设为 1，那么 SPITRF 位将不会置高。

● SPIC1 寄存器

Bit	7	6	5	4	3	2	1	0
Name	—	—	SPICKPOLB	SPICKEG	SPIMLS	SPICSEN	SPIWCOL	SPITRF
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 **SPICKPOLB**：SPI 时钟线的基础状态位
0：当时钟无效时，SPISCK 引脚为高电平
1：当时钟无效时，SPISCK 引脚为低电平
此位决定了时钟线的基础状态，若此位为高，当时钟无效时 SPISCK 为低电平，若此位为低，当时钟无效时 SPISCK 为高电平。

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

Bit 3 **SPIMLS**：SPI 数据移位命令位
0：LSB 优先
1：MSB 优先
数据移位选择位，用于选择数据传输时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 **SPICSEN**：SPI $\overline{\text{SPISCS}}$ 引脚控制位
0：除能
1：使能
SPICSEN 位用于 $\overline{\text{SPISCS}}$ 引脚的使能 / 除能控制。此位为低时，$\overline{\text{SPISCS}}$ 除能并处于浮空状态。此位为高时，$\overline{\text{SPISCS}}$ 作为选择脚。

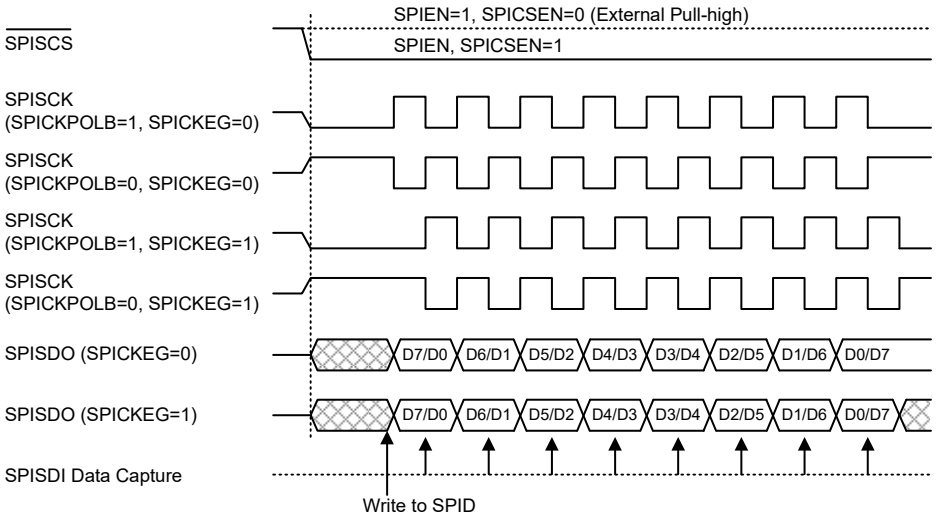
Bit 1 **SPIWCOL**：SPI 写冲突标志位
0：无冲突
1：冲突
SPIWCOL 标志位用于监测数据冲突的发生。此位为高时，表示在传输过程中有数据被写入 SPID 寄存器。若数据正在被传输时，此写操作无效。此位可被应用程序清零。

Bit 0 **SPITRF**：SPI 发送 / 接收结束标志位
0：数据正在发送
1：数据发送结束
SPITRF 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。

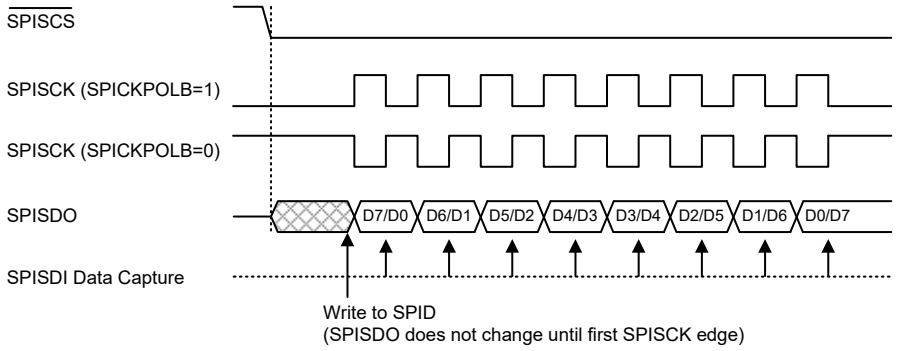
SPI 通信

将 SPIEN 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 SPID 的同时传输 / 接收开始进行。数据传输完成时，SPITRF 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 SPID 中的数据，而且在 SPISDI 引脚上的数据也会被移位到 SPID 寄存器中。主机应在输出时钟信号之前先输出一个 SPISCS 信号以使能从机，从机的数据传输功能也应在与 SPISCK 信号相关的适当时候准备就绪，这由 SPICKPOLB 和 SPICKEG 位决定。所附时序图表明了在 SPICKPOLB 和 SPICKEG 位各种设置情况下从机数据与 SPISCK 信号的关系。

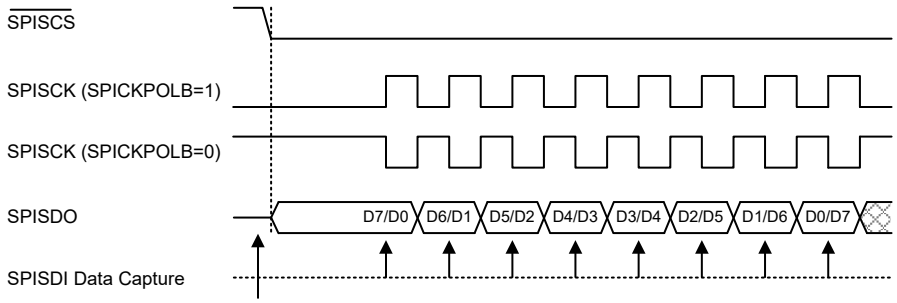
SPI 主机模式在 SPI 时钟运行情况下可以进行正常传输。



SPI 主机模式时序

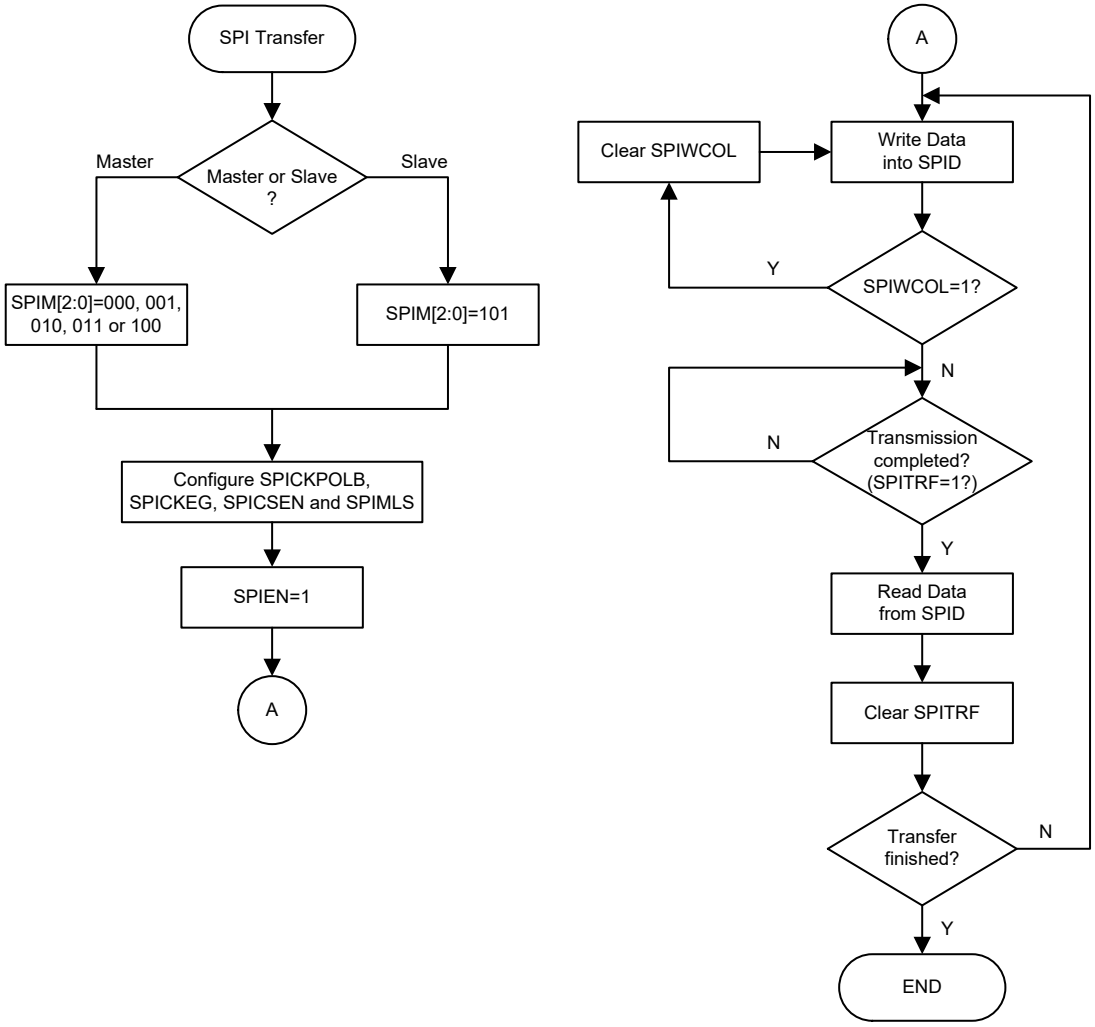


SPI 从机模式时序 – SPICKEG=0



Note: For SPI slave mode, if SPIEN=1 and SPICSEN=0, SPI is always enabled and ignores the SPISCS level.

SPI 从机模式时序 – SPICKEG=1



SPI 传输控制流程图

SPI 总线使能 / 除能

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

当 SPI 总线除能时，通过设置相应引脚共用控制位，SPISCK、SPISDI、SPISDO、$\overline{\text{SPISCS}}$ 可作为 I/O 口或其它共用引脚使用。

SPI 操作步骤

四线制 SPI 接口可完成所有主 / 从模式通信工作。由时序图可看出基本的总线工作流程。

在 SPIC1 寄存器中，SPICSEN 位控制 SPI 接口的所有功能。设置此位为高，$\overline{\text{SPISCS}}$ 信号线有效将使能 SPI 接口。设置此位为低，SPI 接口除能，$\overline{\text{SPISCS}}$ 信号线处于浮空状态因此不能控制 SPI 接口。SPICSEN 位和 SPIC0 寄存器中的 SPIEN 位设置为高，使得 SPISDI 信号线处于浮空状态，SPISDO 信号线为高电平。主机模式中，如果 SPISCK 信号线为高还是低取决于 SPIC1 寄存器中的时钟极性选择位 SPICKPOLB。从机模式中，SPISCK 信号线处于浮空状态。如果 SPIEN 位设置为低，SPI 接口被除能，通过设置相应引脚共用控制位，SPISCK、SPISDI、SPISDO、$\overline{\text{SPISCS}}$ 可作为 I/O 口或其它共用引脚使用。主机模式中，当数据被写入 SPID 寄存器后，主机发起数据传输，并控制时钟信号。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 SPIC0 控制寄存器中的 SPIM2~SPIM0 位，选择 SPI 主机模式和时钟源。

- 步骤 2

  设置 SPICSEN 和 SPIMLS 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 SPIC0 控制寄存器中的 SPIEN 位，使能 SPI 接口功能。

- 步骤 4

  对于写操作：写数据到 SPID 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。再使用 SPISCK 和 SPISDO 信号线将数据输出。跳至步骤 5。

  对于读操作：从 SPISDI 信号线移入的数据将被存储在 TXRX 缓存器中，直到所有数据接收完毕，此时数据全部锁存至 SPID 寄存器。

- 步骤 5

  检测 SPIWCOL 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 SPITRF 位或等待 SPI 串行总线中断发生。

- 步骤 7

  从 SPID 寄存器中读数据。

- 步骤 8

  清除 SPITRF。

- 步骤 9

  跳回至步骤 4。

从机模式

- 步骤 1

  设置 SPIC0 控制寄存器中的 SPIM2~SPIM0 位，选择 SPI 从机模式。

- 步骤 2

  设置 SPICSEN 和 SPIMLS 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 SPIC0 控制寄存器中的 SPIEN 位，使能 SPI 接口功能。

- 步骤 4

  对于写操作：写数据到 SPID 寄存器，实际上此时数据会被存储在 TXRX 缓存器中。等待主机时钟 SPISCK 信号和 $\overline{\text{SPISCS}}$ 信号。跳至步骤 5。

  对于读操作：从 SPISDI 信号线移入的数据将被存储在 TXRX 缓存器中，直到所有数据接收完毕，此时数据全部锁存至 SPID 寄存器。

- 步骤 5

  检测 SPIWCOL 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 SPITRF 位或等待 SPI 串行总线中断发生。

- 步骤 7

  从 SPID 寄存器中读数据。

- 步骤 8

  清除 SPITRF。

- 步骤 9

  跳回至步骤 4。

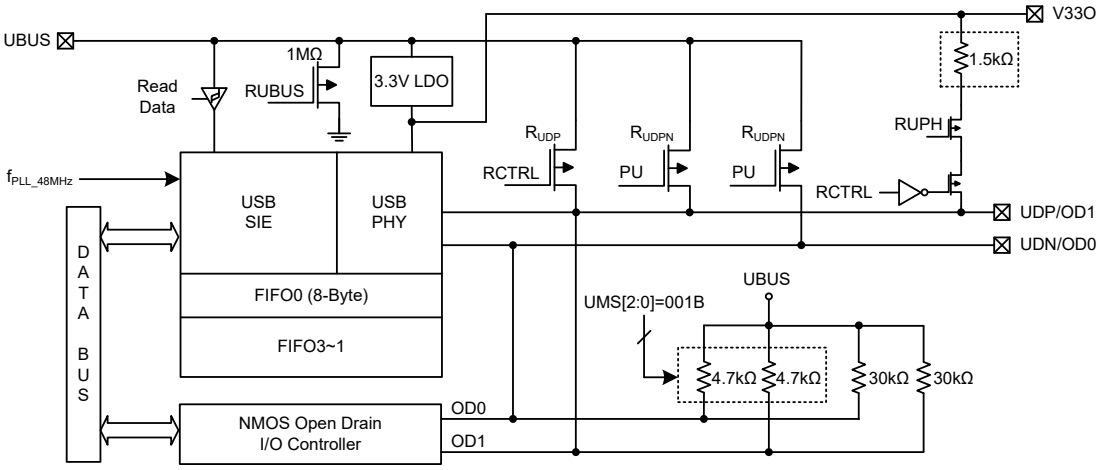
错误侦测

SPIC1 寄存器中的 SPIWCOL 位用于数据传输期间监测数据冲突的发生。此位由串行接口设置为高，而由应用程序来清除为零。在数据传输期间如果写数据到 SPID，此位被置高提示数据冲突发生，并阻止数据继续被写入。

USB 接口

USB 接口是一个 4 线串行总线，允许一个主机与多达 127 个外部设备在同一总线上进行通信。主机使用基于令牌协议的方法进行通信控制。USB 总线的其它优点包括热插拔、动态设备配置。由于 USB 数据传输协议的复杂性，不容许在规格书中提供全面的 USB 操作信息，因此，读者应该查阅其它外部信息以详细了解 USB。

该单片机具有一个 USB 接口，使 USB 外围产品的设计更加方便。



USB 接口方框图

USB 接口操作

为实现与外部 USB 主机的通信，内部 USB 模块有三个外部引脚，即 UDP、UDN 以及 3.3V 稳压器输出 V33O。串行接口引擎 (SIE) 解码输入的 USB 数据流并传输至正确的端点缓存存储器 FIFO。该 USB 模块具有 4 个端点，即 EP0~EP3。端点 0 的 FIFO 大小为 8 字节，端点 1 为 16 字节，端点 2 为 32 字节，端点 3 为 64 字节。端点 0 支持控制传输，端点 1~3 支持中断或批量传输。

USB 接口寄存器

USB 功能由一系列的寄存器控制。一系列的状态寄存器向用户提供 USB 数据传输情况以及出错情况。USB 具有独立中断，可用于显示主机正访问 USB FIFO 或 USB 操作情况的变化，如 USB 暂停模式、恢复事件或 USB 复位的发生。

寄存器名称	位							
	7	6	5	4	3	2	1	0
SYSC	BITWE	USBDIS	RUBUS	UBUSF	—	—	D1	ESDF
USB_STAT	OD1O	OD0O	OD1I	OD0I	SE1	SE0	PU	RUPH
UINT	—	—	—	—	EP3EN	EP2EN	EP1EN	EP0EN
USC	URD	UMS2	UMS1	UMS0	RESUME	URST	RMWK	SUSP
UESR	—	—	—	—	EP3F	EP2F	EP1F	EP0F
UCC	RCTRL	—	JSUSP	SUSP2	USBCKEN	—	EPS1	EPS0
AWR	AD6	AD5	AD4	AD3	AD2	AD1	AD0	WKEN
STL	—	—	—	—	STL3	STL2	STL1	STL0
SIES	NMI	UERR2	UERR1	UERR0	IN	OUT	UFERR	ASET
MISC	LEN0	READY	SETCMD	V33OS	—	CLEAR	TX	REQUEST

寄存器名称	位							
	7	6	5	4	3	2	1	0
SETIO	—	—	—	—	SETIO3	SETIO2	SETIO1	DATATG
FRNUM0	D7	D6	D5	D4	D3	D2	D1	D0
FRNUM1	—	—	—	—	—	D10	D9	D8
FIFO0	D7	D6	D5	D4	D3	D2	D1	D0
FIFO1	D7	D6	D5	D4	D3	D2	D1	D0
FIFO2	D7	D6	D5	D4	D3	D2	D1	D0
FIFO3	D7	D6	D5	D4	D3	D2	D1	D0

USB 接口寄存器列表

- **SYSC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	BITWE	USBDIS	RUBUS	UBUSF	—	—	D1	ESDF
R/W	R/W	R/W	R/W	R	—	—	R/W	R/W
POR	0	0	0	1	—	—	0	x

“x”：未知

Bit 7 **BITWE**：软件写权限使能控制位
0：使能 – 软件可写 URST、UFERR、LEN0 和 SETCMD 位
1：除能 – 软件不可写 URST、UFERR、LEN0 和 SETCMD 位

Bit 6 **USBDIS**：USB SIE 功能控制位
0：使能
1：除能
此位用于控制 USB SIE 功能。当此位置“1”，USB SIE 功能将除能。

Bit 5 **RUBUS**：UBUS 引脚下拉功能控制位
0：使能
1：除能

Bit 4 **UBUSF**：UBUS 引脚输入状态
0：低电平
1：高电平

Bit 3~2 未定义，读为“0”

Bit 1 **D1**：保留位，不可用且固定为“0”

Bit 0 **ESDF**：ESD 问题标志位
当出现 ESD 问题时此位置“1”。此位由 SIE 置位，由软件清零。

- **USB_STAT 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	OD1O	OD0O	OD1I	OD0I	SE1	SE0	PU	RUPH
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
POR	1	1	x	x	0	0	0	1

“x”：未知

Bit 7 **OD1O**：OD1 引脚输出数据，开漏 NMOS 输出

Bit 6 **OD0O**：OD0 引脚输出数据，开漏 NMOS 输出

Bit 5 **OD1I**：OD1 引脚输入状态

Bit 4 **OD0I**：OD0 引脚输入状态

Bit 3 **SE1**：USB 总线 SE1 噪声指示位
该位用于表示 SIE 已检测到 USB 总线的 SE1 噪声。该位由 SIE 置位，由软件清零。

Bit 2 **SE0**：USB 总线 SE0 噪声指示位
该位用于表示 SIE 已检测到 USB 总线的 SE0 噪声。该位由 SIE 置位，由软件清零。

Bit 1 **PU**：UDP/UDN 引脚上拉功能控制位
0：除能 – 无内部上拉电阻
1：使能 – UDP 和 UDN 之间有 600kΩ 上拉电阻

Bit 0 **RUPH**：1.5kΩ 电阻连接控制
0：UDP 和 V33O 之间未连接 1.5kΩ 电阻
1：UDP 和 V33O 之间连接 1.5kΩ 电阻

- **UINT 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	EP3EN	EP2EN	EP1EN	EP0EN
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 **EP3EN**：USB 端点 3 中断控制位
0：除能
1：使能

Bit 2 **EP2EN**：USB 端点 2 中断控制位
0：除能
1：使能

Bit 1 **EP1EN**：USB 端点 1 中断控制位
0：除能
1：使能

Bit 0 **EP0EN**：USB 端点 0 中断控制位
0：除能
1：使能

- **USC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	URD	UMS2	UMS1	UMS0	RESUME	URST	RMWK	SUSP
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R
POR	1	0	0	0	x	x	x	x

“x”：未知

Bit 7 **URD**：USB 复位信号控制位
0：USB 复位信号不能复位单片机
1：USB 复位信号将复位单片机

Bit 6~4 **UMS2~UMS0**：USB 与 OD 模式选择
000：无可用模式 – V33O 输出将浮空。相关外部引脚将处于输入浮空的状态。
001：开漏输出模式 – V33O 输出将上拉到 VDD。相关外部引脚将变成 OD0/OD1 引脚并分别连接上拉电阻到 VDD。
01x：USB 模式 – V33O 功能使能。相关外部引脚将用作 UDP/UDN 引脚。
100~111：未定义，OD0/OD1 将处于浮空状态。

Bit 3 **RESUME**：USB 恢复指示标志位
0：恢复信号无效或 USB 已经离开暂停模式
1：恢复信号有效且 USB 即将离开暂停模式
此位为只读位。当 USB 恢复事件发生，此位会被 SIE 置位，也会产生一个中断来唤醒单片机。为了检测到暂停状态，单片机必须置高 USBCKEN 位和清零 SUSP2 位。当 USB 离开暂停模式，SUSP 位清零同时 RESUME 位也会被清零。当单片机检测暂停模式时必须考虑能唤醒单片机的恢复信号。

Bit 2 **URST**：USB 复位指示位
0：没有 USB 复位事件发生
1：USB 复位事件发生
当 BITWE 位被清零时，此位只能由 USB SIE 置位和只能由 MCU 清零。当

URST 位被置位，表明 USB 复位事件发生，将产生 USB 中断。

Bit 1 **RMWK**：USB 远程唤醒命令控制位
0：无 USB 远程唤醒命令
1：启动 USB 远程唤醒命令
此位可由 MCU 置位，以强制 USB 主机离开暂停模式。设置 RMWK 位为 1 以启动远程唤醒命令。该位应保持高电平至少 1 个 USB 时钟，以确保远程唤醒命令被 SIE 接收。

Bit 0 **SUSP**：USB 暂停指示位
0：USB 离开暂停模式
1：USB 进入暂停模式
该只读位由 SIE 设置为 1，以表明 USB 总线已经进入暂停模式。当 SUSP 位由低变高时将产生相应的中断。

● UESR 寄存器

UESR 寄存器为 USB 端点中断状态寄存器，用于表明哪个端点被访问以及选择 USB 总线。端点请求标志位 EPnF 用于表明哪个端点被访问。若端点被访问，相关的端点请求标志位将置“1”，若 USB 中断使能且堆栈未满，将产生中断。当有效的端点请求标志位被响应，端点请求标志位需由软件清零。

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	EP3F	EP2F	EP1F	EP0F
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	x	x	x	x

“x”：未知

Bit 7~4 未定义，读为“0”

Bit 3 **EP3F**：端点 3 访问中断请求标志位
0：端点没有被访问
1：端点被访问

Bit 2 **EP2F**：端点 2 访问中断请求标志位
0：端点没有被访问
1：端点被访问

Bit 1 **EP1F**：端点 1 访问中断请求标志位
0：端点没有被访问
1：端点被访问

Bit 0 **EP0F**：端点 0 访问中断请求标志位
0：端点没有被访问
1：端点被访问

● UCC 寄存器

Bit	7	6	5	4	3	2	1	0
Name	RCTRL	—	JSUSP	SUSP2	USBCKEN	—	EPS1	EPS0
R/W	R/W	—	R	R/W	R/W	—	R/W	R/W
POR	0	—	0	x	0	—	x	x

“x”：未知

Bit 7 **RCTRL**：UDP 和 UBUS 之间 RUDP 电阻连接控制位
0：除能 – UDP 和 UBUS 之间未连接 R_{UDP} 电阻
1：使能 – UDP 和 UBUS 之间连接 R_{UDP} 电阻

Bit 6 未定义，读为“0”

Bit 5 **JSUSP**：USB J-STATE 暂停模式指示位
0：USB 接口未处于 J-STATE 暂停模式
1：USB 接口处于 J-STATE 暂停模式
此位用于表示 USB 接口是否处于 J-STATE 暂停模式。

Bit 4 **SUSP2**：USB 暂停模式下降低功耗控制位
0：除能
1：使能
当进入暂停模式时，若此位置高，则可减少电流以符合 USB 标准规格。

Bit 3 **USBCKEN**：USB 时钟使能控制位
0：除能
1：使能

Bit 2 未定义，读为“0”

Bit 1~0 **EPS1~EPS0**：端点 FIFO 访问选择位
00：端点 0 FIFO
01：端点 1 FIFO
10：端点 2 FIFO
11：端点 3 FIFO

● AWR 寄存器

AWR 寄存器包含了当前地址以及远程唤醒功能控制位。该寄存器初始值为“00H”。SIES 寄存器中的 ASET 位决定了是否立即将 USB 主机命令中得到的地址值载入此寄存器。

Bit	7	6	5	4	3	2	1	0
Name	AD6	AD5	AD4	AD3	AD2	AD1	AD0	WKEN
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 **AD6~AD0**：USB 设备地址 bit 6~bit 0

Bit 0 **WKEN**：USB 设备远程唤醒功能使能控制位
0：除能
1：使能

● STL 寄存器

STL 寄存器用于表明相应端点是否正确工作。一旦端点出现不正确的操作，就必须由应用程序将 STL 寄存器中的相关位置高。此寄存器内容由 USB 复位信号和 SETUP 令牌事件清零。

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	STL3	STL2	STL1	STL0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	x	x	x	x

“x”：未知

Bit 7~4 未定义，读为“0”

Bit 3 **STL3**：USB 端点 3 FIFO 操作停顿指示位
0：端点 3 FIFO 操作没有停顿
1：端点 3 FIFO 操作停顿

Bit 2 **STL2**：USB 端点 2 FIFO 操作停顿指示位
0：端点 2 FIFO 操作没有停顿
1：端点 2 FIFO 操作停顿

Bit 1 **STL1**：USB 端点 1 FIFO 操作停顿指示位
0：端点 1 FIFO 操作没有停顿
1：端点 1 FIFO 操作停顿

Bit 0 **STL0**：USB 端点 0 FIFO 操作停顿指示位
0：端点 0 FIFO 操作没有停顿
1：端点 0 FIFO 操作停顿

● **SIES 寄存器**

SIES 寄存器用于表明当前 SIE 接收到的信号状态以及控制 SIE 是否自动修改设备地址。

Bit	7	6	5	4	3	2	1	0
Name	NMI	UERR2	UERR1	UERR0	IN	OUT	UFERR	ASET
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”：未知

Bit 7 **NMI**：NAK 令牌中断屏蔽控制位
0：NAK 令牌中断没有被屏蔽
1：NAK 令牌中断被屏蔽
如果该位被置位，当单片机发送 NAK 令牌到 USB 主机，不会产生中断。否则当该位被清零，单片机从端点 n 发送 NAK 令牌到 USB 主机时，如果相应的端点中断控制位使能，会产生端点 n NAK 令牌中断。

Bit 6~4 **UERR2~UERR0**：USB SIE 错误状态
0xx：没有发生错误
100：USB PID 错误
101：位填充错误
110：CRC 错误
111：主机未响应 SIE
这几位用来表示 USB SIE 检测到何种错误。这几位由 USB SIE 置位，由应用程序清零。

Bit 3 **IN**：IN 令牌指示位
0：接收到的令牌包不是 IN 令牌
1：接收到的令牌包是 IN 令牌
该位用来表示当前从 USB 主机接收到的令牌包是否为 IN 令牌。

Bit 2 **OUT**：OUT 令牌指示位
0：接收到的令牌包不是 OUT 令牌
1：接收到的令牌包是 OUT 令牌
该位用来表示当前从 USB 主机接收到的令牌包是否为 OUT 令牌 (零长度的 OUT 令牌除外)。OUT 数据被读取后，该位应由应用程序清零。注意，当接收到下一个有效的 SETUP 令牌后，该位也会被清零。

Bit 1 **UFERR**：FIFO 访问错误指示位
0：没有发生错误
1：发生错误
UFERR 位用来表示在 FIFO 被访问期间是否发生如 CRC 错误，PID 错误或位填充错误等 USB 总线错误。当 BITWE 位被清零时，该位只能由 SIE 置位，只能由应用程序清零。

Bit 0 **ASET**：设备地址更新方式控制位
0：向 AWR 寄存器写入地址时会立即更新设备地址
1：USB 主机读取 IN 令牌数据后更新设备地址
ASET 位用来设置 SIE，以根据存储在 AWR 寄存器的地址值自动更新设备地址。当此位由软件设置为“1”，主机通过 IN 操作成功从单片机读取数据后，SIE 根据存储在 AWR 寄存器的地址值更新设备地址。否则，当该位被清除为“0”，地址写入 AWR 寄存器后，SIE 将立即更新设备地址。因此为了正常工作，软件必须在接收到下一个有效的 SETUP 令牌时将该位清除为“0”。

● **MISC 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	LEN0	READY	SETCMD	V33OS	—	CLEAR	TX	REQUEST
R/W	R/W	R	R/W	R/W	—	R/W	R/W	R/W
POR	x	x	x	0	—	x	x	x

“x”：未知

Bit 7 **LEN0**：零长度数据包指示位
0：未接收到零长度数据包
1：接收到零长度数据包
LEN0 位用来表示 USB 主机是否发送一个零长度的数据包。当 BITWE 被清零时，此位只能由硬件置位和由软件清零。单片机接收到有效的 USB SETUP 令牌后此位也可由硬件清零。

Bit 6 **READY**：所需端点 FIFO 就绪指示位
0：未就绪
1：就绪

Bit 5 **SETCMD**：SETUP 命令指示位
0：不是 SETUP 令牌
1：是 SETUP 令牌
当 BITWE 位被清零时，此位只能由硬件置位和由应用程序清零。

Bit 4 **V33OS**：电压选择
0：内部 V33O 电压
1：外部 3.3V LDO

Bit 3 未定义，读为“0”

Bit 2 **CLEAR**：FIFO 清除功能使能控制位
0：除能 – 无操作
1：使能 – 清除所请求的端点 FIFO
即使相应的 FIFO 还未就绪的情况下，置位此位仍可清除所请求的 FIFO。此位应置“1”以产生一个一定长度的正脉冲来清除所请求的 FIFO，然后清零此位。清除 FIFO 后，USB 接口输出管道端点可接收由主机发送的新数据，输入管道端点可传送新数据给主机。

Bit 1 **TX**：数据传输方向指示位
0：MCU 从 USB FIFO 中读取数据
1：MCU 写数据至 USB FIFO
该位用来定义 MCU 与 USB 端点 FIFO 之间的数据传输方向。当此位设置为高，意味着单片机想要写数据到 USB 端点 FIFO。写操作完成后，在终止 FIFO 请求之前需将其清零，以表示数据传输结束。对于单片机的读操作，需将该位清零，以表示单片机想要从 USB 端点 FIFO 读取数据。读操作完成后，在终止 FIFO 请求之前需将其置高。

Bit 0 **REQUEST**：所需 FIFO 请求控制位
0：无请求或请求完成
1：请求所需的 FIFO
此位用来请求所需访问的端点 FIFO。选择所需端点并将此位置高，便可访问相应的 FIFO。操作完成后，应将此位清零。

● **SETIO 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SETIO3	SETIO2	SETIO1	DATATG
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	0

Bit 7~4　未定义，读为“0”

Bit 3　**SETIO3**：端点 3 FIFO 控制位
0：输出管道
1：输入管道

Bit 2　**SETIO2**：端点 2 FIFO 控制位
0：输出管道
1：输入管道

Bit 1　**SETIO1**：端点 1 FIFO 控制位
0：输出管道
1：输入管道

Bit 0　**DATATG**：DATA 令牌触发位
0：DATA0 优先发送
1：DATA1 优先发送
该位用来选择 DATA 令牌触发位。当该位为 0，DATA0 将先被发送，紧接着是请求的端点 FIFO 的 IN/OUT 数据。否则，DATA1 将先被发送，接着是连续的 IN/OUT 数据传输。

注：1. 如果相关的 SETIOn 位设置为 1，相应的端点 FIFO 设置为输入管道用于 IN 令牌操作。否则，相应的端点 FIFO 设置为输出管道用于 OUT 令牌操作。此功能的目的是为了避免 USB 主机异常的发送 IN 令牌或 OUT 令牌和除能相关的端点。

2. 由于 USB 规范的定义，当 USB 主机发送一个“Set Configuration”SETUP 令牌，数据管道应该首先发送 DATA0(数据触发)。因此，当 USB 设备接收到一个“Set Configuration”SETUP 令牌，用户需要将 DATATG 位清零，确保下一个数据将首先发送 DATA0。
数据触发位设置流程：
(1) 通过设置 UCC 寄存器来选择需要的端点。
(2) 通过设置 DATATG 位来配置数据令牌触发位。
(3) 通过设置 MISC 寄存器中的 TX 位来配置 IN 或 OUT 管道方向。
(4) 置高 REQUEST 位。
(5) 切换 TX 位。
(6) 清除 REQUEST 位。

3. 该寄存器只能通过上电复位来重置。

● **FRNUM0 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	x	x	x	x	x	x	x	x

“x”：未知

Bit 7~0　**D7~D0**：每个 USB SOF 包的帧数 D7~D0

- **FRNUM1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	D10	D9	D8
R/W	—	—	—	—	—	R	R	R
POR	—	—	—	—	—	x	x	x

“x”：未知

Bit 7~3　未定义，读为“0”
Bit 2~0　**D10~D8**：每个 USB SOF 包的帧数 D10~D8

- **FIFOn 寄存器 (n=0~3)**

FIFOn 寄存器用于 USB 设备与主机之间的数据处理存储。MCU 通过相应的控制和选择位的特定组合对 FIFOn 进行读数据或写入数据的操作。

名称	类型	POR	说明
FIFO0	R/W	xxxx xxxx	端点 0 数据管道
FIFO1	R/W	xxxx xxxx	端点 1 数据管道
FIFO2	R/W	xxxx xxxx	端点 2 数据管道
FIFO3	R/W	xxxx xxxx	端点 3 数据管道

“x”：未知

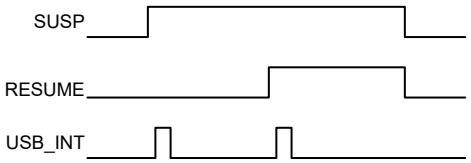
USB 暂停模式和唤醒

USB 暂停模式

如果 USB 总线上超过 3ms 没有信号，单片机将进入暂停模式。USC 寄存器中的暂停指示位 SUSP 将设置为“1”并产生 USB 中断，表示单片机应跳转到暂停状态，以满足 USB 暂停电流规范。为了满足暂停电流，应通过软件清除 UCC 寄存器中的 USB 时钟使能控制位 USBCKEN 位为“0”以除能 USB 时钟。暂停电流通过设置 UCC 寄存器中 SUSP2 位能够进一步降低。

USB 主机唤醒

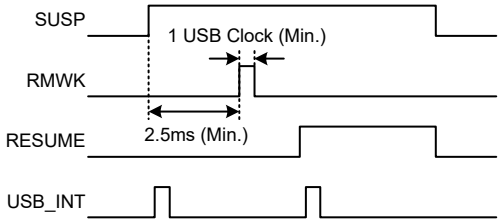
当主机发出恢复信号时，单片机将会因 USB 中断而唤醒，USC 寄存器内的 RESUME 位将会置位。为了保证单片机正常运行，应用程序必须设置 UCC 寄存器内的 USBCKEN 位为高使得 USB 主机开始与 USB 设备进行通信。在 USB 设备真正离开暂停模式后，SUSP2 和 RESUME 位都会清零。所以当单片机检测暂停位 SUSP2 时也应该考虑到监测恢复位 RESUME 的状态。



暂停与主机唤醒

USB 远程唤醒

由于单片机具有远程唤醒功能，可通过设置 USC 寄存器中的 RMWK 位发送一个唤醒脉冲来唤醒 USB 主机。一旦 USB 主机接收到来自 USB 设备的唤醒信号，将发送一个恢复信号给单片机。



暂停和远程唤醒

USB 中断

有几种 USB 情况可以产生 USB 中断。当其中一种情况发生，将会产生一个中断脉冲来引起单片机注意。这些情况包括 USB 暂停，USB 恢复，USB 复位和 USB 端点 FIFO 访问事件。当上述任一情况发生触发 USB 中断时，如果相应的中断控制位使能且堆栈未满，程序将在返回到主程序之前跳转到相应的中断向量。

USB 端点 FIFO 访问事件具有相应的指示位以显示哪个端点 FIFO 被访问。当端点 FIFO 访问标志被置位，如果相应的端点 FIFO 管道和中断控制位都使能，将产生一个 USB 中断，端点 FIFO 访问标志应由应用程序清零。当 USB 暂停或 USB 恢复情况发生，相应的标志位 SUSP 和 RESUME 将被置位，并直接产生一个 USB 中断而无需使能相应的中断控制位。SUSP 和 RESUME 位是只读位，需由 USB SIE 置位或清零。要产生 USB 中断，除了 USB 模块中相应的中断使能控制位要被置位，主机 MCU 中的总中断使能位和相关中断使能控制位也必须被置位。如果这些位没被置位，那么将不会有中断响应。

此外，若总中断使能控制位和相关中断使能控制位 SOFE 都被置位，则每个 USB SOF 包将产生一个中断，详见中断章节。

LCD 驱动器

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，且需很多特殊的考虑以正确地操作 LCD。此单片机有内部 LCD 信号产生电路，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。

此单片机含有驱动使能 LCD 各种类型显示的选项。下表显示此单片机带有的功能选项。

驱动数目	占空比	偏压	偏压类型	波形类型
36×4	1/4	1/3	C	A 或 B
34×6	1/6	1/3	C	A 或 B
32×8	1/8	1/3	C	A 或 B

LCD 驱动器输出选项

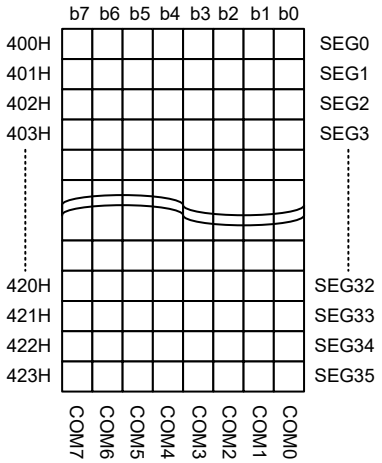
LCD 显示数据存储器

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 LCD 显示数据存储器。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。

该单片机为 LCD 显示提供一个嵌入式数据存储区域。这个区域位于 Sector 4 的 00H~23H。LCD 显示存储器能被读出和写入，可通过间接寻址模式，并使用 MP1L/MP1H 或 MP2L/MP2H 来进行，或者通过扩展指令直接寻址。如果使用间接寻址方式，当要存取 LCD 存储器时，首先要将 MP1H 或 MP2H 的值设为“04H”来选择对 Sector 4 操作。此后，用户可以通过 MP1L 或 MP2L 使用间接寻址方式来对存储区进行操作。选择了 Sector 4 之后，使用 MP1L 或 MP2L 可以对地址范围为 00H~23H 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。

当数据被写入此显示数据区域，这些数据自动地被 LCD 驱动器读取来产生相应的 LCD 驱动信号。把“1”或“0”写入显示存储器的相应位，可以控制显示或不显示。

下方的 LCD 存储器映射图显示了内部 LCD 存储如何映射到单片机显示的 SEG 和 COM 引脚。应注意，未使用的 LCD RAM 不可作为通用数据存储器使用，例如 LCD 为 1/4 占空比，则 COM4~COM7 只能读为“0”。



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} 的时钟源通过 SCC 寄存器中的 FSS 位选择来自于内部 LIRC 振荡器或者 LXT 振荡器。该方法可产生理想的 4kHz 频率的 LCD 时钟，以获得更好的 LCD 显示效果。

LCD 寄存器介绍

LCD 控制寄存器位于数据存储区，用于设置 LCD 驱动器的各种特性。该单片机有两个 LCD 控制寄存器，LCDC0 和 LCDC2。

LCDC0 寄存器中的 TYPE 位用于选择输出的 LCD 控制信号为 A 型或是 B 型。该寄存器中的 LCDP1~LCDP0 位用于选择 C 型 LCD 电源来提供适当的偏压。在应用中，选择匹配的 LCD 面板也可以降低偏压电流。LCDC0 寄存器中的 LCDEN 位只有当单片机工作于快速模式、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 LCD 显示将一直处于除能状态。

LCDC2 寄存器用于选择 C 型 LCD 充电泵时钟分频以及 LCD 占空比。

寄存器名称	位							
	7	6	5	4	3	2	1	0
LCDC0	TYPE	—	LCDP1	LCDP0	—	—	—	LCDEN
LCDC2	LCDPCK2	LCDPCK1	LCDPCK0	—	—	DTYC1	DTYC0	—

LCD 寄存器列表

- **LCDC0 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TYPE	—	LCDP1	LCDP0	—	—	—	LCDEN
R/W	R/W	—	R/W	R/W	—	—	—	R/W
POR	0	—	0	0	—	—	—	0

Bit 7 **TYPE**：LCD 波形类型选择
0：A 型
1：B 型

Bit 6 未定义，读为“0”

Bit 5~4 **LCDP1~LCDP0**：C 型 LCD 电源选择
00：电源来自 PLCD/V1/V2 外部引脚
01：电源来自 V_C 上的内部参考电压 V_{REFIN}
10：电源来自 V_B 上的内部电压 3V
11：电源来自 V_A 上的内部电压 V_{DD}
V_{REFIN} 为内部参考电压，其电压值约为 1.0V。

Bit 3~1 未定义，读为“0”

Bit 0 **LCDEN**：LCD 使能控制
0：除能
1：使能
在快速、低速和空闲模式下，LCD 使能 / 除能可由此位控制。在休眠模式下，LCD 始终关闭。

- **LCDC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	LCDPCK2	LCDPCK1	LCDPCK0	—	—	DTYC1	DTYC0	—
R/W	R/W	R/W	R/W	—	—	R/W	R/W	—
POR	0	0	0	—	—	0	0	—

Bit 7~5 **LCDPCK2~LCDPCK0**：C 型 LCD 充电泵时钟分频选择
000：250Hz (f_{SUB}/128)
001：500Hz (f_{SUB}/64)
010：1kHz (f_{SUB}/32)
011：2kHz (f_{SUB}/16)
100：4kHz (f_{SUB}/8)
101：8kHz (f_{SUB}/4)
110：16kHz (f_{SUB}/2)
111：16kHz (f_{SUB}/2)
这些频率选项是基于 32kHz LCD 时钟源计算所得。

Bit 4~3 未定义，读为“0”

Bit 2~1 **DTYC1~DTYC0**：LCD 占空比选择
00：1/4 Duty (使用 COM0~COM3)
01：1/6 Duty (使用 COM0~COM5)
10：1/8 Duty (使用 COM0~COM7)
11：未定义
未使用到的 COM 引脚可被配置为普通 I/O 或其它共用引脚功能。

Bit 0 未定义，读为“0”

LCD 电压源和偏压

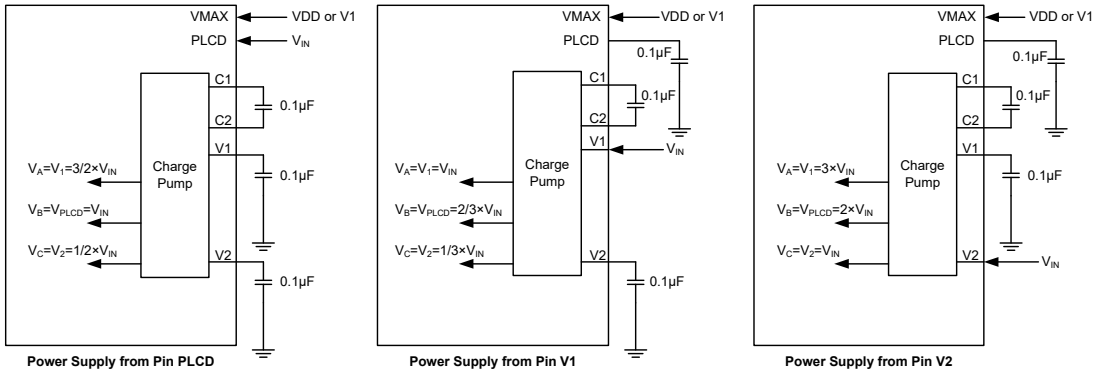
该 LCD 驱动器通过 C 型偏压方式产生几种电压值以产生时间振幅可变的信号。电压源可以通过 LCDC0 寄存器 LCDP1~LCDP0 位选择来自内部电源还是外部电源引脚 PLCD、V1 或 V2。C 型偏压使用内部充电泵电路，产生高于 PLCD 或 V2 引脚上的电压。这项特性在单片机提供电压小于 LCD 所需电压时非常有用。C 型充电泵时钟来源于 f_{SUB} 的分频，分频率是通过 LCDC2 寄存器的 LCDPCK2~LCDPCK0 位实现的。为了产生所需的电压值，必须要在引脚 C1 与 C2 之间连接充电泵电容。

对于 C 型 1/3 偏压电源方案，无论 LCD 电压源来自外部引脚输入还是内部电压，都要用到 V_{SS}、V_A、V_B 和 V_C 四种电压值。这些偏压值的大小取决于 LCD 电源连接方案。

LCD 电源		V_A 电压值	V_B 电压值	V_C 电压值
外部电源	V_{IN} 来自 V1 引脚	V_{IN}	$2/3\times V_{IN}$	$1/3\times V_{IN}$
	V_{IN} 来自 PLCD 引脚	$3/2\times V_{IN}$	V_{IN}	$1/2\times V_{IN}$
	V_{IN} 来自 V2 引脚	$3\times V_{IN}$	$2\times V_{IN}$	V_{IN}
内部电源	V_A (V_A=V_{DD})	V_{DD}	$2/3\times V_{DD}$	$1/3\times V_{DD}$
	V_B (V_B=3V)	$3/2\times 3V$	3V	$1/2\times 3V$
	V_C (V_C=V_{REFIN}) (注)	$3\times V_{REFIN}$	$2\times V_{REFIN}$	V_{REFIN}

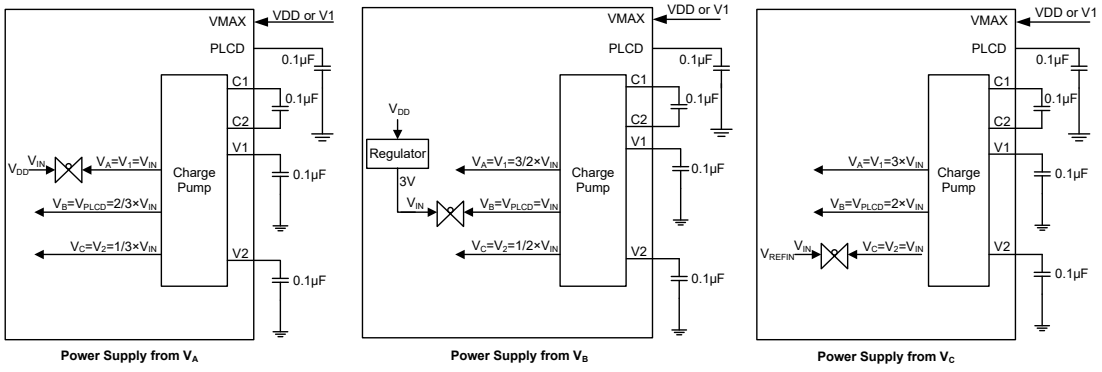
注：电压来自内部产生的参考电压，约为 1.0V

C 型偏压电源方案



注：VMAX 引脚必须连接最大电压值以防止引脚上漏电。

C 型偏压外部电源配置 – 1/3 Bias



注：VMAX 引脚必须连接最大电压值以防止引脚上漏电。

C 型偏压内部电源配置 – 1/3 Bias

连接到 VMAX 引脚的电压取决于加在 PLCD 引脚上的电压，但需要注意的是，要确保充电泵产生的内部电压值不能超过 V_{DD} 最大值，5.5V。

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

C 型偏压 VMAX 引脚连接方式

LCD 复位功能

LCDC0 寄存器中的 LCDEN 位取反后，与休眠功能进行“OR”逻辑或运算的结果可产生 LCD 内部复位。清除 LCDEN 位也会将 LCD 复位。若进入休眠模式前，即使 LCDEN 位为 1 使能 LCD 驱动，进入休眠模式后 LCD 仍被复位。

当 LCDEN 位设置为 1 使能 LCD 驱动功能接着发生单片机复位，则 LCD 驱动器将被复位，且在单片机复位过程中 COM 和 SEG 输出都处于浮空状态。LCD 复位操作所需时间为 $t_{RSTD}+t_{SST}$，这两个参数值可参考系统上电时间电气特性。

MCU 复位	休眠模式	LCDEN	LCD 复位	COM&SEG 电平
No	Off	1	No	正常工作
No	Off	0	Yes	低
No	On	x	Yes	高
Yes	x	x	Yes	浮空

“x”：无关

注：这里所说的 MCU 复位情况不包含 WDT 在空闲 / 休眠模式时的溢出复位。

LCD 复位功能

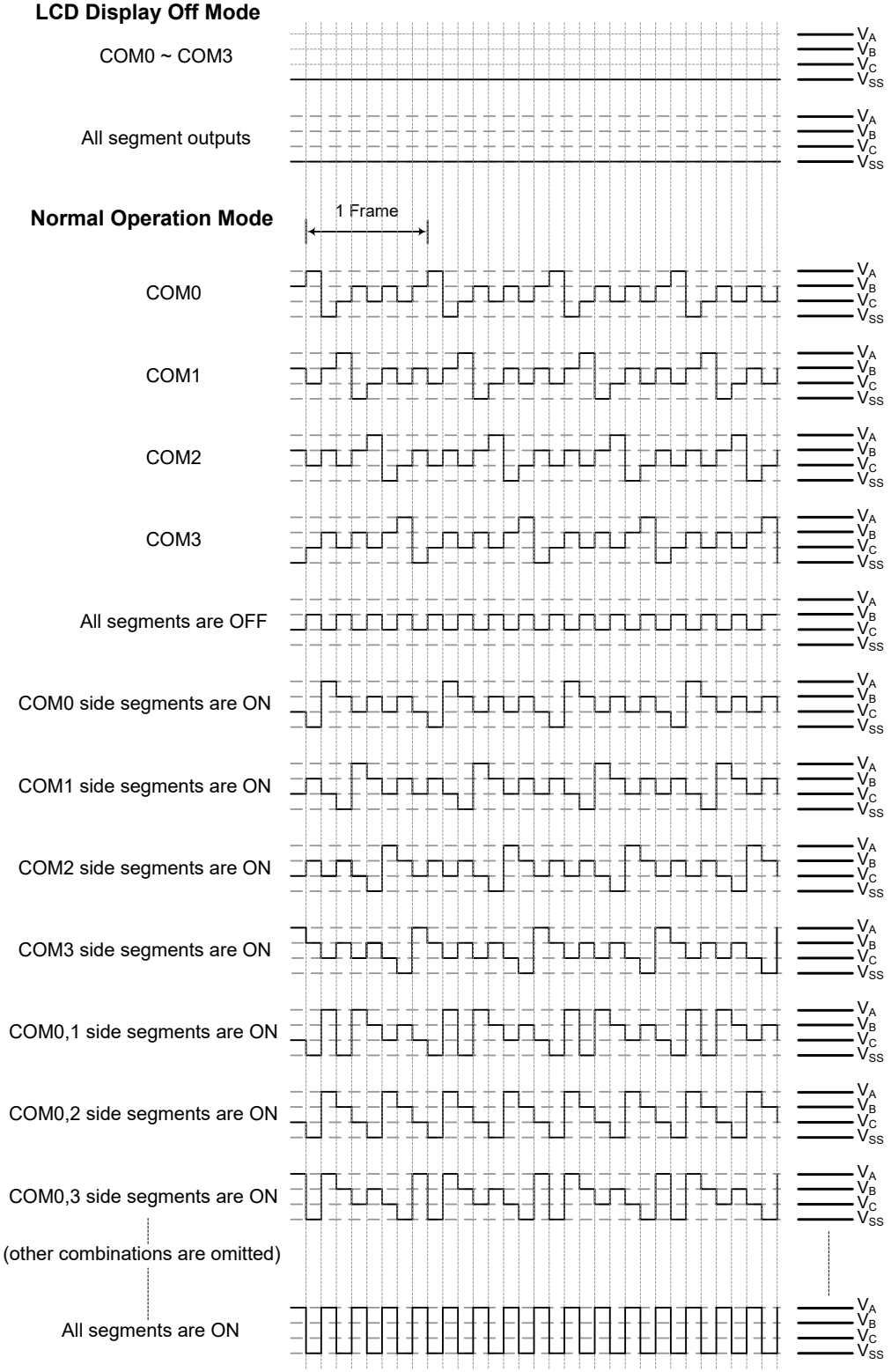
LCD 驱动输出

LCD 驱动器提供的 COM 和 SEG 输出数目，以及波形类型选项，取决于 LCD 控制位的设置。LCD 偏压类型为 C 型，固定为 1/3 偏压。

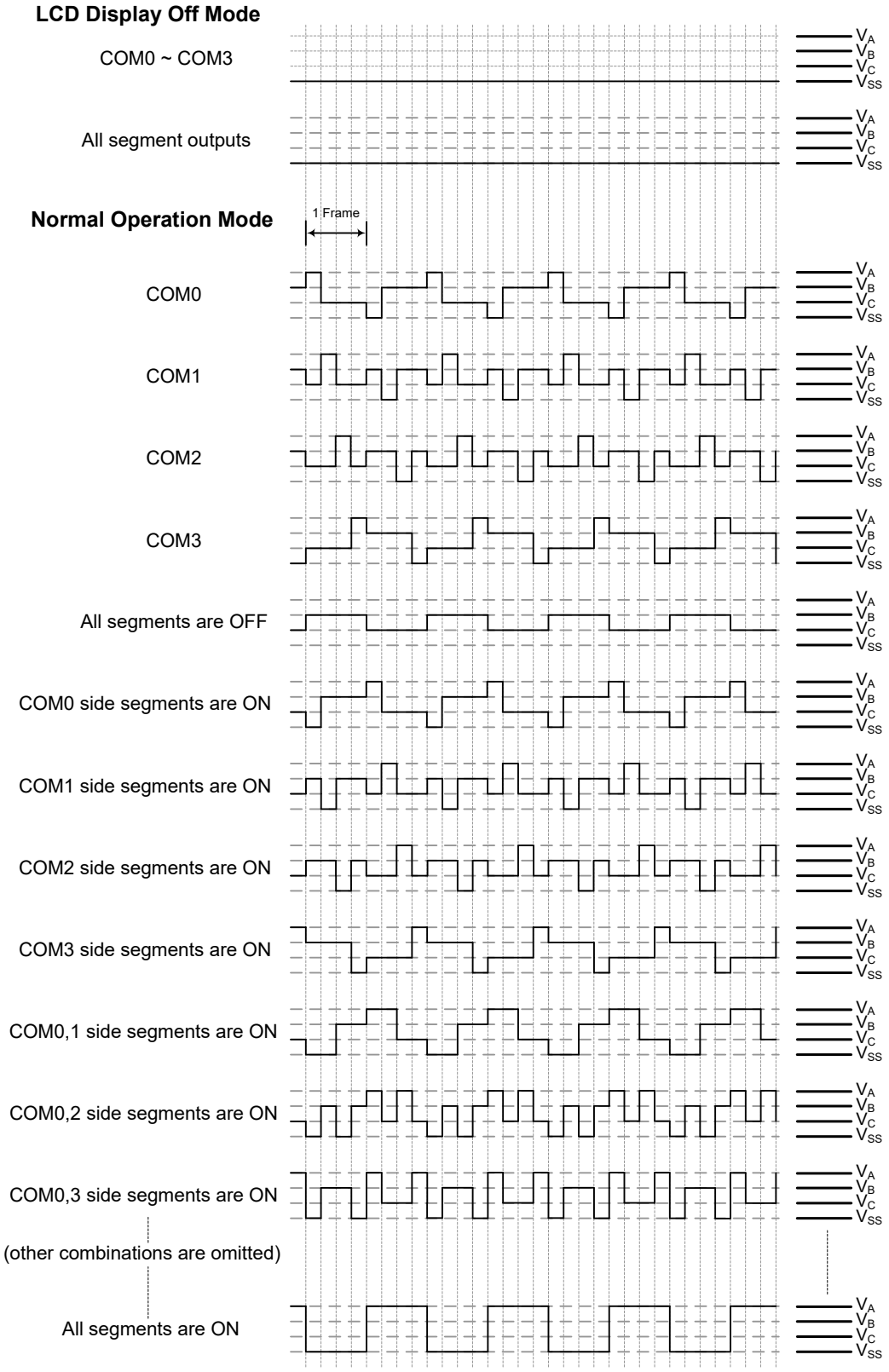
由于 LCD 基本性质的缘故，它们的像素点只能加上 AC 电压，如果加上 DC 电压，将会引起永久性的损害。因此 LCD 显示器的对比度由提供到每个像素的实际 RMS 电压控制，这个值等于 COM 引脚上的电压值减去 SEG 引脚上电压值的结果的 RMS 值。RMS 电压必须大于 LCD 的饱和电压，以便能打开像素点，但同时也要小于阈值电压，以便能关闭像素点。

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 COM 口的个数，也称为底板或 COM。例如，占空比为 1/4，表示 COM 的数目为 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 LCDC 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。

C 型，4-COM, 1/3 Bias

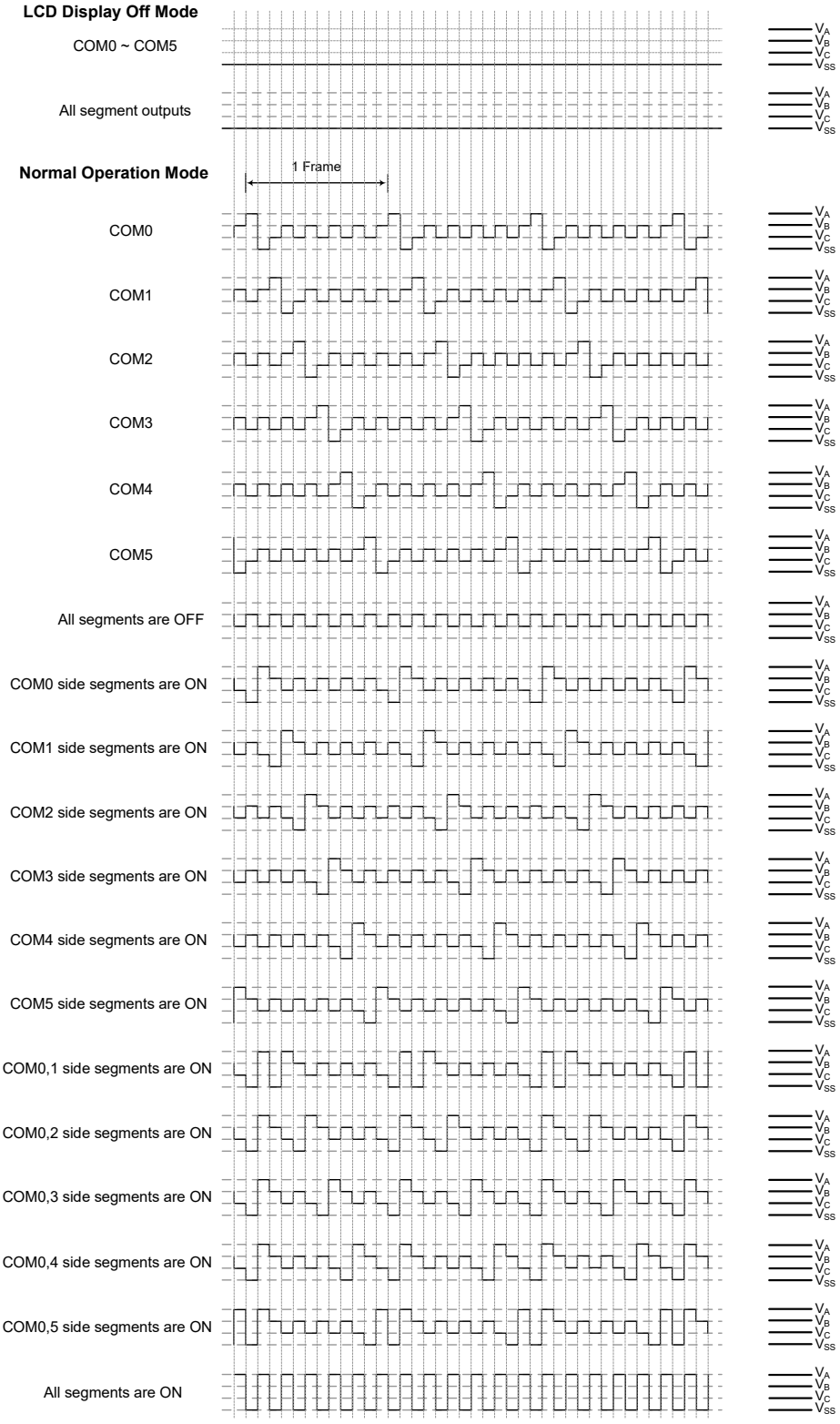


LCD 驱动输出 – A 型，1/4 Duty, 1/3 Bias

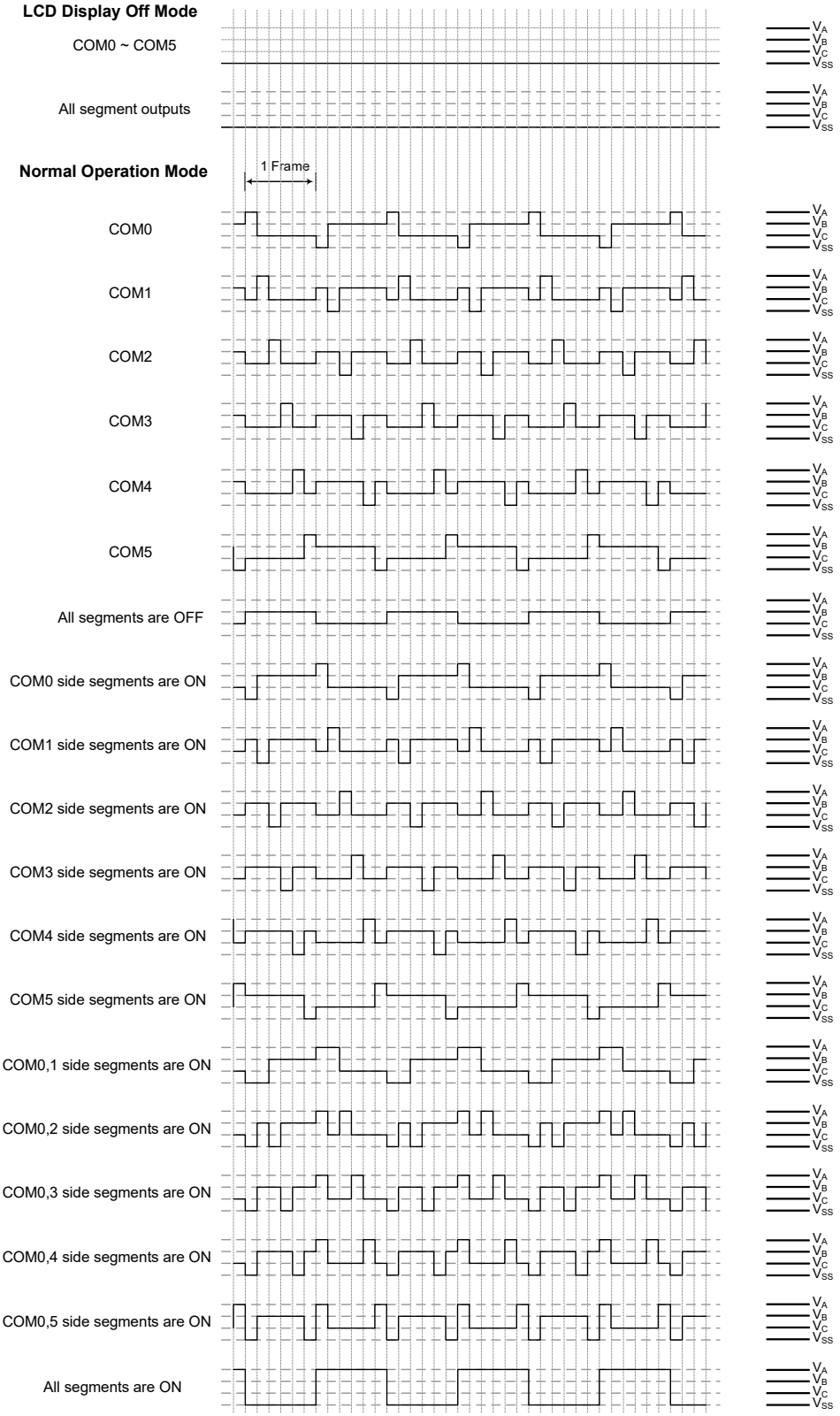


LCD 驱动输出 – B 型 , 1/4 Duty, 1/3 Bias

C 型 , 6-COM, 1/3 Bias

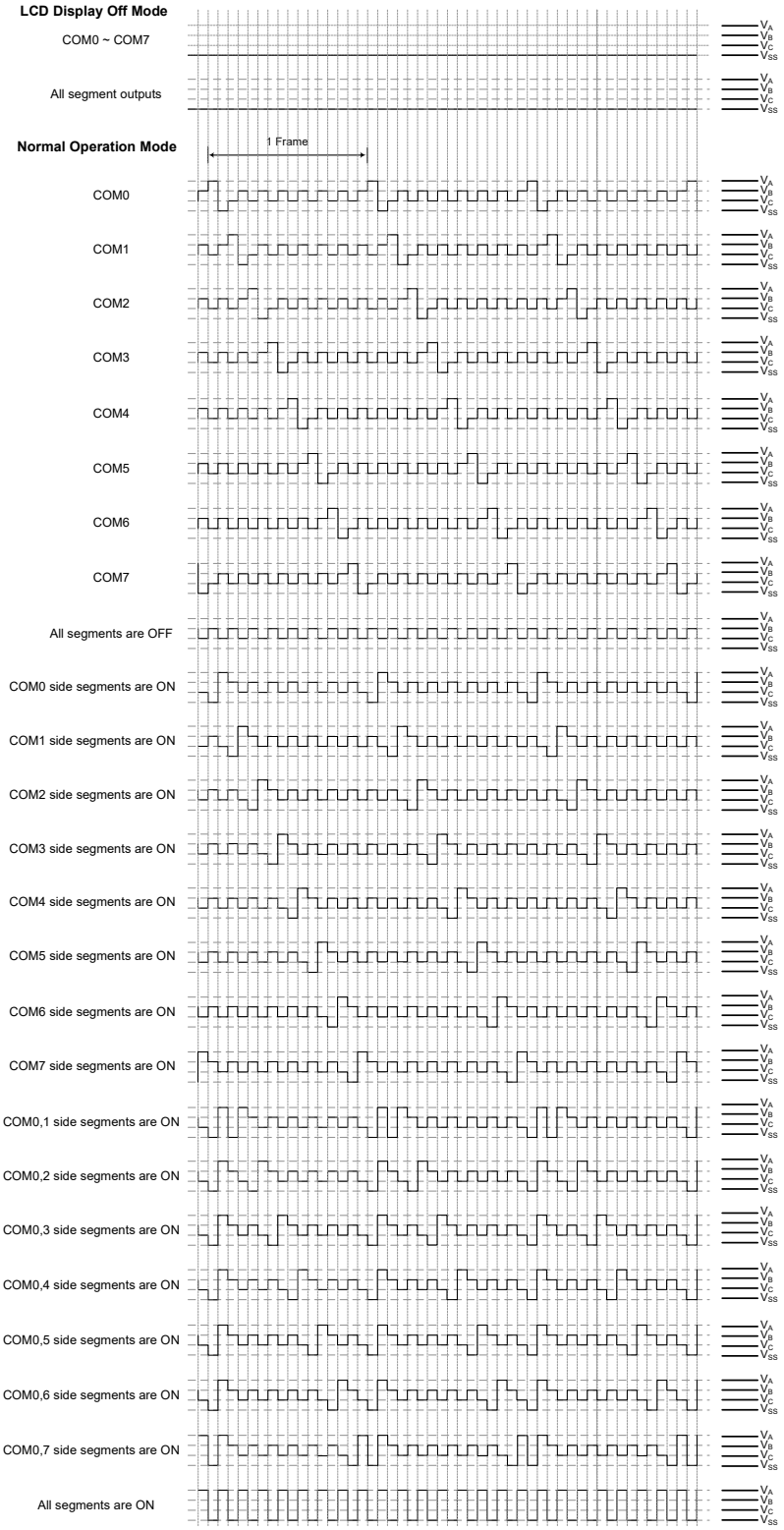


LCD 驱动输出 – A 型 , 1/6 Duty, 1/3 Bias

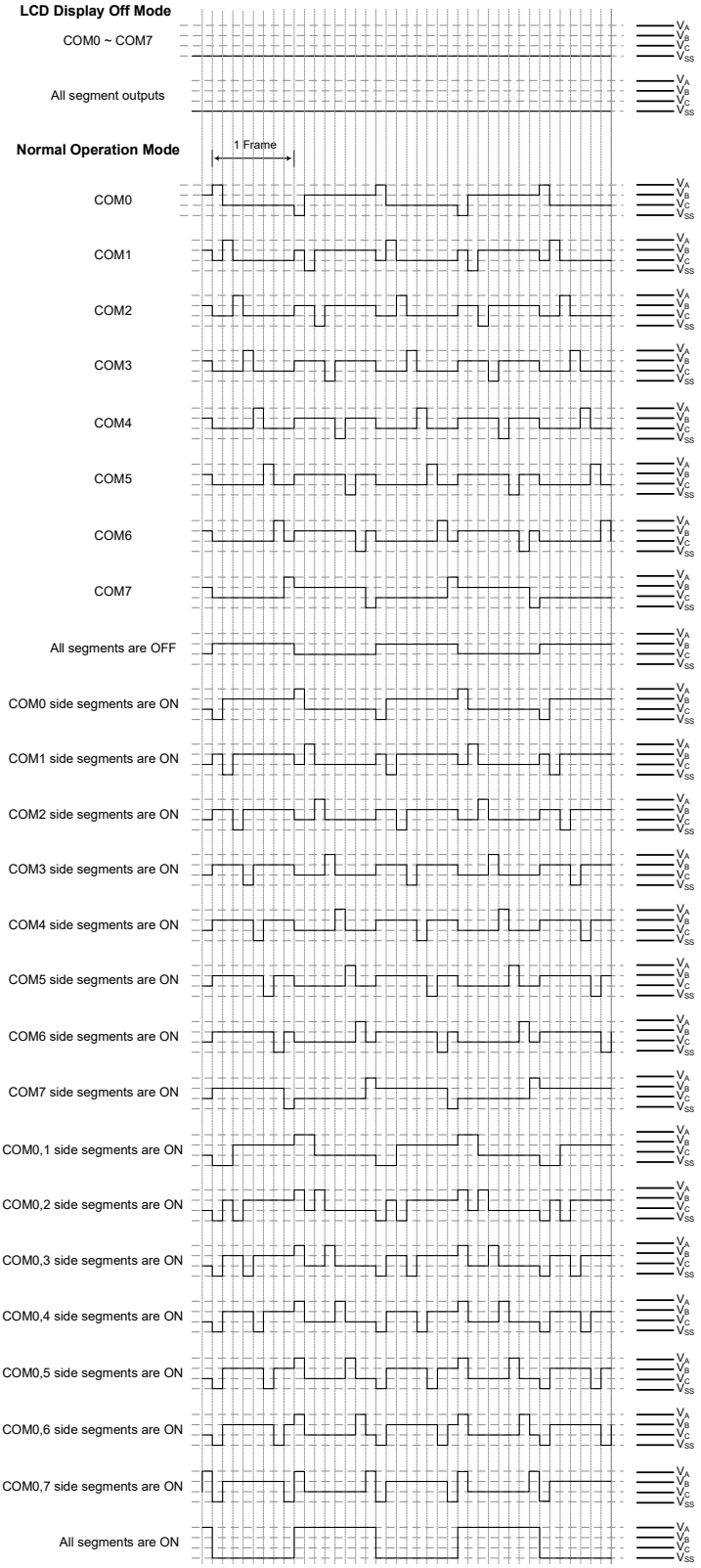


LCD 驱动输出 – B 型 , 1/6 Duty, 1/3 Bias

C 型 , 8-COM, 1/3 Bias



LCD 驱动输出 – A 型 , 1/8 Duty, 1/3 Bias



LCD 驱动输出 – B 型 , 1/8 Duty, 1/3 Bias

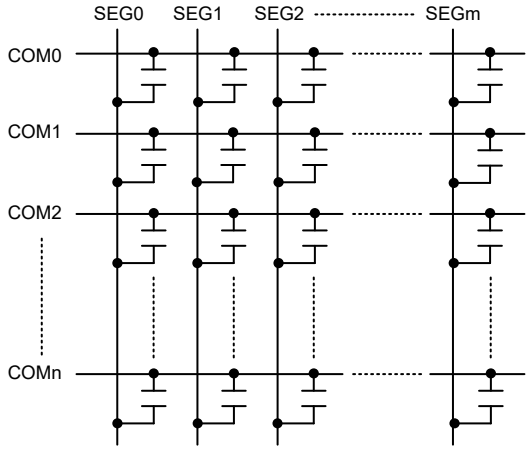
编程注意事项

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休眠模式后所发生的变化。可以通过将 LCDC0 控制寄存器中的 LCD 使能控制位 LCDEN 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。

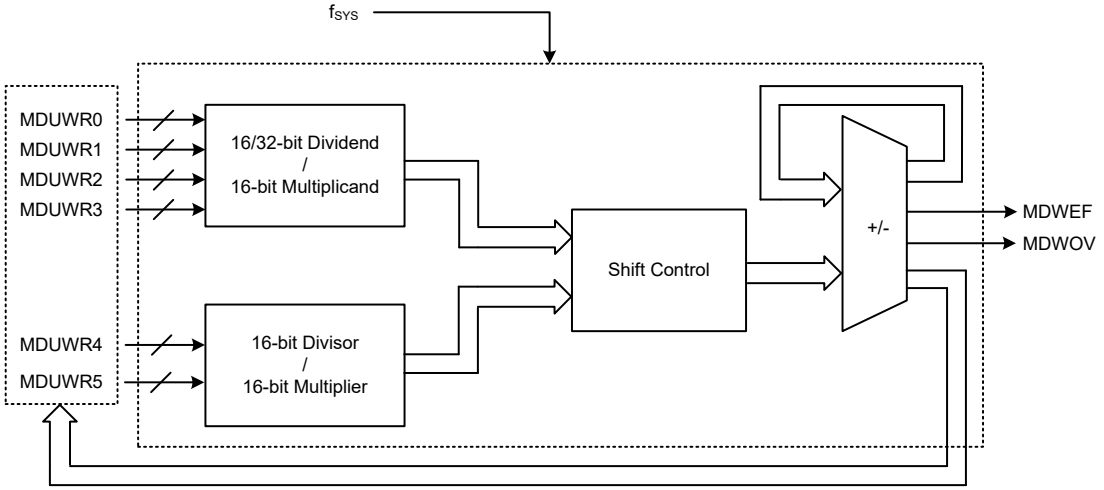
要注意当上电复位后，LCDEN 位会被清零，显示功能关闭。



LCD 面板等效电路

16 位乘除法单元 – MDU

该单片机内置一个 16 位乘除法单元，即 MDU，是 16 位无符号乘法与 32 位 / 16 位无符号除法器。此乘除法器可取代软件执行乘除法运算，节省大量运算时间、程序及数据占用空间，降低单片机负载，以达到提升整体系统性能。



16-bit MDU 方框图

MDU 寄存器

乘法或除法操作通过一系列寄存器以指定的方式，指定的读写次序实现。状态寄存器 MDUWCTRL 可指示运算操作的状态。六个数据寄存器每个寄存器充当的角色取决于要执行的运算操作。

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

MDU 寄存器列表

- **MDUWRn 寄存器 (n=0~5)**

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

“x”：未知

Bit 7~0 **D7~D0**：16-bit MDU 数据寄存器 n
此寄存器的具体使用取决于当前 MDU 的执行运算。

● **MDUWCTRL 寄存器**

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

Bit 7 **MDWEF**：16-bit MDU 错误标志位
0：正常
1：异常
如果在运算过程中 MDUWRn 寄存器被改写或被读取，MDWEF 位由硬件自动置位。当操作完成且 MDWEF 位为 1 时，可通过读取 MDUWCTRL 寄存器将此位清零。

Bit 6 **MDWOV**：16-bit MDU 溢出标志位
0：无溢出发生
1：乘法结果大于 FFFFH 或除数为 0
每完成一次运算，硬件自动更新此位以指示当前运算的情况。

Bit 5~0 未定义，读为“0”

MDU 操作

乘除法单元用于乘法运算还是除法运算取决于寄存器 MDUWR0~MDUWR5 的写入顺序。无论是被乘数，被除数，乘数或除数值的写入，都是要先写入低字节数据再写入高字节数据到对应的 MDU 数据寄存器。当按指定的次序写入数据到 MDUWRn 数据寄存器直到对 MDUWR5 寄存器写入数据后，才开始执行乘法或除法操作。需要强调的是，写入数据到这些 MDUWRn 寄存器时，并非一定要不间断写入但必须要依照正确的写入次序。因此在对数据寄存器依指定次序写入时，允许中间插入非写入 MDUWRn 的其它指令或中断等。

对 MDUWRn 数据寄存器写入顺序与乘除法的对应关系如下：

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

当按当前运算指定的次序完成对数据寄存器的写入后，MDU 开始执行对应的运算。不同运算所需的时间是不同的。在运算过程中，MDUWRn (n=0~5) 寄存器禁止被写或被读。每次运算完成后，需读取 MDUWCTRL 寄存器判断运算状态是否正确。若正确，则可按照指定的次序读取 MDUWRn 寄存器以得到最终的运算结果。

各运算所需的时间如下所示：

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

运算完成后，计算结果储存在 MDU 数据寄存器中，必须按指定的次序读取这些寄存器。需要强调的是，读取 MDUWRn 数据寄存器时，并非一定要不间断读取但必须要依照正确的读取次序。因此在对数据寄存器依指定次序读取数据时，允许中间插入非读 MDUWRn 的其它指令或中断等。

读取 MDUWRn 数据寄存器与乘除法的对应关系如下：

- 32-bit/16-bit 除法运算：商数值依序读取 MDUWR0 到 MDUWR3；余数值依序读取 MDUWR4、MDUWR5
- 16-bit/16-bit 除法运算：商数值依序读取 MDUWR0、MDUWR1；余数值依序读取 MDUWR4、MDUWR5
- 16-bit×16-bit 乘法运算：乘积值依序读取 MDUWR0 到 MDUWR3

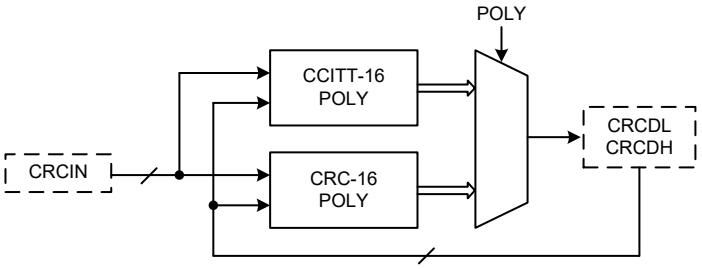
下述表格总结了 MDU 不同运算操作时的数据寄存器读写次序及运算时间。注意，在 MDU 操作未完成之前，单片机不可进入空闲或休眠模式，否则 MDU 操作失败。

运算 / 项目	32-bit / 16-bit 除法	16-bit / 16-bit 除法	16-bit × 16-bit 乘法
写次序 最先写 ↓ ↓ ↓ ↓ 最后写	写入被除数 Byte 0 到 MDUWR0 写入被除数 Byte 1 到 MDUWR1 写入被除数 Byte 2 到 MDUWR2 写入被除数 Byte 3 到 MDUWR3 写入除数 Byte 0 到 MDUWR4 写入除数 Byte 1 到 MDUWR5	写入被除数 Byte 0 到 MDUWR0 写入被除数 Byte 1 到 MDUWR1 写入除数 Byte 0 到 MDUWR4 写入除数 Byte 1 到 MDUWR5	写入被乘数 Byte 0 到 MDUWR0 写入乘数 Byte 0 到 MDUWR4 写入被乘数 Byte 1 到 MDUWR1 写入乘数 Byte 1 到 MDUWR5
运算时间	$17\times t_{SYS}$	$9\times t_{SYS}$	$11\times t_{SYS}$
读次序 最先读 ↓ ↓ ↓ ↓ 最后读	从 MDUWR0 读取商数 Byte 0 从 MDUWR1 读取商数 Byte 1 从 MDUWR2 读取商数 Byte 2 从 MDUWR3 读取商数 Byte 3 从 MDUWR4 读取余数 Byte 0 从 MDUWR5 读取余数 Byte 1	从 MDUWR0 读取商数 Byte 0 从 MDUWR1 读取商数 Byte 1 从 MDUWR4 读取余数 Byte 0 从 MDUWR5 读取余数 Byte 1	从 MDUWR0 读取乘积结果 Byte 0 从 MDUWR1 读取乘积结果 Byte 1 从 MDUWR2 读取乘积结果 Byte 2 从 MDUWR3 读取乘积结果 Byte 3

MDU 运算总结

循环冗余校验 – CRC

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



CRC 方框图

CRC 寄存器

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

寄存器名称	位							
	7	6	5	4	3	2	1	0
CRCCR	—	—	—	—	—	—	—	POLY
CRCIN	D7	D6	D5	D4	D3	D2	D1	D0
CRCDL	D7	D6	D5	D4	D3	D2	D1	D0
CRCDH	D15	D14	D13	D12	D11	D10	D9	D8

CRC 寄存器列表

- **CRCCR 寄存器**

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

Bit 7~1　未定义，读为“0”

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

- **CRCIN 寄存器**

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

Bit 7~0　**D7~D0**：CRC 输入数据寄存器

● **CRCDL 寄存器**

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

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

● **CRCDH 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 **D15~D8**：16-bit CRC 校验和高字节数据寄存器

CRC 操作

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

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

● CRC-CCITT：$X^{16}+X^{12}+X^5+1$

● CRC-16：$X^{16}+X^{15}+X^2+1$

CRC 计算

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

CRC 计算步骤

1. 清除校验和寄存器对 CRCDH 和 CRCDL。
2. 对 8-bit 输入数据字节和 16-bit CRCSUM 高字节进行异或操作，其结果称为临时 CRCSUM。
3. 将临时 CRCSUM 值左移一位，并向最低有效位 LSB 填入“0”。
4. 检查在步骤 3 中完成移位后的临时 CRCSUM 值。

   若 MSB 为“0”，则该移位后的临时 CRCSUM 将作为新的临时 CRCSUM。否则，对步骤 3 中移位后的临时 CRCSUM 和数据“8005H”进行异或操作。该操作结果将作为新的临时 CRCSUM。

   应注意的是对于 CRC-16 多项式，用于异或操作的数据为“8005H”，而对于 CRC-CCITT 多项式用于异或操作的数据则为“1021H”。
5. 重复步骤 3 到步骤 4，直到输入数据字节的所有位都经过计算。
6. 重复步骤 2 到步骤 5，直到所有输入数据字节都经过计算。此时，最新的计算结果则为最终的 CRC 校验和 CRCSUM。

CRC 计算范例

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

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

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

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

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

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

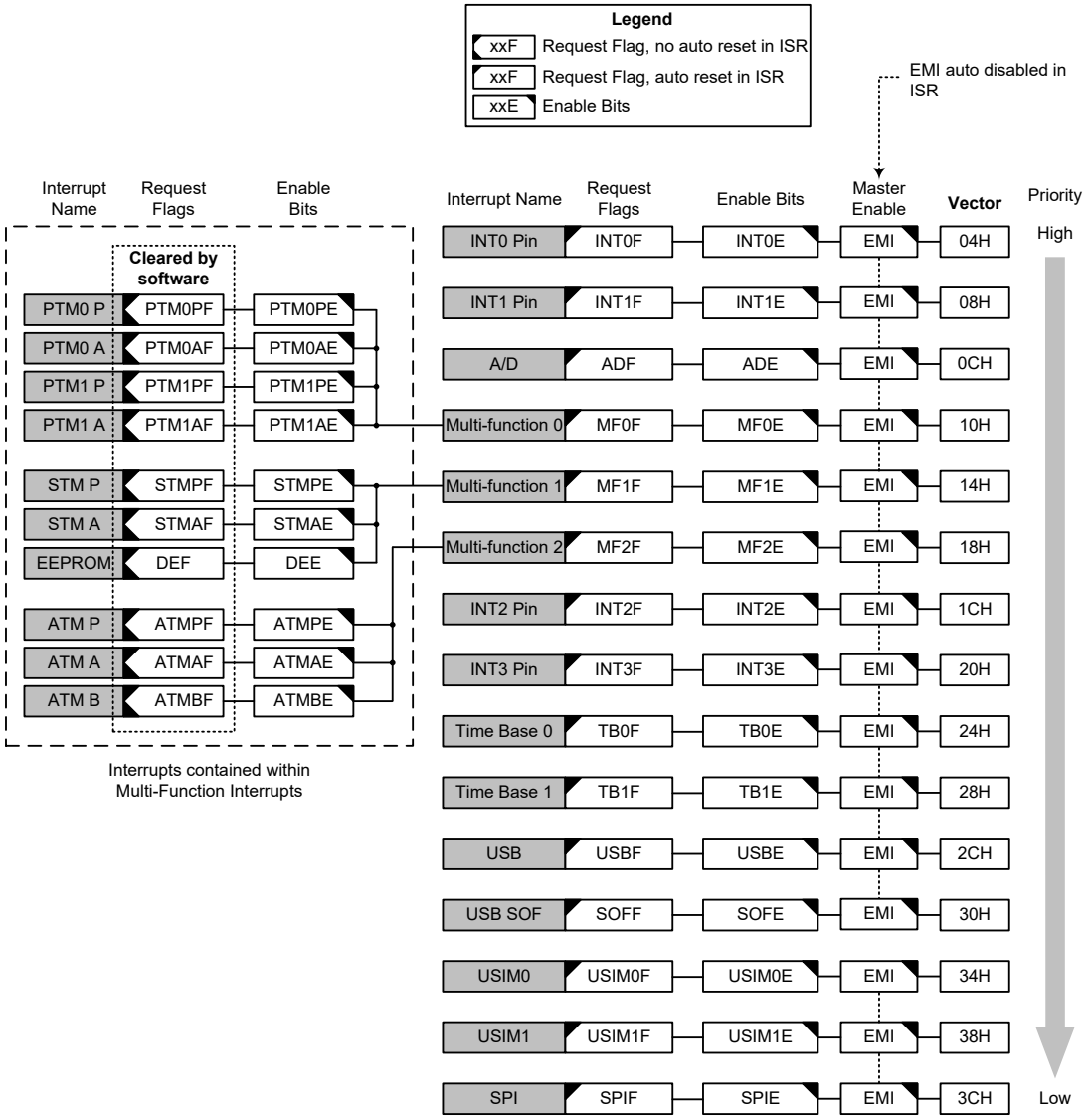
注程序存储器 CRC 校验和计算范例

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

中断

中断是单片机一个重要功能。当外部事件或内部功能如发生定时器模块或 A/D 转换器转换结束等，并且产生中断时，系统会暂时中止当前的程序而转到执行相对应的中断服务程序。此单片机提供多个外部中断和内部中断功能，外部中断由 INT0~INT3 引脚动作产生，而内部中断由各种内部功能产生，如定时器模块、时基、EEPROM、USIM、SPI 和 A/D 转换器等等。

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，而有些中断则共用多功能中断向量。



中断结构

中断寄存器

中断控制基本上是在一定单片机条件发生时设置请求标志位，应用程序中中断使能位的设置是通过位于专用数据存储器中的一系列寄存器控制的。寄存器总的分为三类。第一类是 INTC0~INTC3 寄存器，用于设置基本的中断；第二类

是 MFI0~MFI2 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 引脚	INTnE	INTnF	n=0~3
A/D 转换器	ADE	ADF	—
多功能中断	MFnE	MFnF	n=0~2
时基	TBnE	TBnF	n=0~1
USB	USBE	USBF	—
USB 帧起始	SOFE	SOFF	—
USIM	USIMnE	USIMnF	n=0~1
SPI	SPIE	SPIF	—
EEPROM 擦 / 写操作	DEE	DEF	—
STM	STMPE	STMPF	—
	STMAE	STMAF	—
PTM	PTMnPE	PTMnPF	n=0~1
	PTMnAE	PTMnAF	n=0~1
ATM	ATMPE	ATMPF	—
	ATMAE	ATMAF	—
	ATMBE	ATMBF	—

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	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
INTC1	INT2F	MF2F	MF1F	MF0F	INT2E	MF2E	MF1E	MF0E
INTC2	USBF	TB1F	TB0F	INT3F	USBE	TB1E	TB0E	INT3E
INTC3	SPIF	USIM1F	USIM0F	SOFF	SPIE	USIM1E	USIM0E	SOFE
MFI0	PTM1AF	PTM1PF	PTM0AF	PTM0PF	PTM1AE	PTM1PE	PTM0AE	PTM0PE
MFI1	—	DEF	STMAF	STMPF	—	DEE	STMAE	STMPE
MFI2	—	ATMBF	ATMAF	ATMPF	—	ATMBE	ATMAE	ATMPE

中断寄存器列表

● **INTEG 寄存器**

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

Bit 7~6 **INT3S1~INT3S0**：INT3 脚中断边沿控制位
00：除能
01：上升沿
10：下降沿
11：双沿

Bit 5~4 **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	—	ADF	INT1F	INT0F	ADE	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 **ADF**：A/D 转换器中断请求标志位
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 **ADE**：A/D 转换器中断控制位
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	INT2F	MF2F	MF1F	MF0F	INT2E	MF2E	MF1E	MF0E
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 **INT2F**：INT2 中断请求标志位
0：无请求
1：中断请求

Bit 6 **MF2F**：多功能中断 2 请求标志位
0：无请求
1：中断请求

Bit 5 **MF1F**：多功能中断 1 请求标志位
0：无请求
1：中断请求

Bit 4 **MF0F**：多功能中断 0 请求标志位
0：无请求
1：中断请求

Bit 3 **INT2E**：INT2 中断控制位
0：除能
1：使能

Bit 2 **MF2E**：多功能中断 2 控制位
0：除能
1：使能

Bit 1 **MF1E**：多功能中断 1 控制位
0：除能
1：使能

Bit 0 **MF0E**：多功能中断 0 控制位
0：除能
1：使能

● **INTC2 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	USBF	TB1F	TB0F	INT3F	USBE	TB1E	TB0E	INT3E
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 **USBF**：USB 中断请求标志位
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 **INT3F**：INT3 中断请求标志位
0：无请求
1：中断请求

Bit 3 **USBE**：USB 中断控制位
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 **INT3E**：INT3 中断控制位
0：除能
1：使能

- **INTC3 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	SPIF	USIM1F	USIM0F	SOFF	SPIE	USIM1E	USIM0E	SOFE
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 **SPIF**：SPI 中断请求标志位
0：无请求
1：中断请求

Bit 6 **USIM1F**：USIM1 中断请求标志位
0：无请求
1：中断请求

Bit 5 **USIM0F**：USIM0 中断请求标志位
0：无请求
1：中断请求

Bit 4 **SOFF**：USB SOF 中断请求标志位
0：无请求
1：中断请求

Bit 3 **SPIE**：SPI 中断控制位
0：除能
1：使能

Bit 2 **USIM1E**：USIM1 中断控制位
0：除能
1：使能

Bit 1 **USIM0E**：USIM0 中断控制位
0：除能
1：使能

Bit 0 **SOFE**：USB SOF 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	PTM1AF	PTM1PF	PTM0AF	PTM0PF	PTM1AE	PTM1PE	PTM0AE	PTM0PE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PTM1AF**：PTM1 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求
注意，当中断响应时此位必须通过应用程序清零。

Bit 6 **PTM1PF**：PTM1 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求
注意，当中断响应时此位必须通过应用程序清零。

Bit 5 **PTM0AF**：PTM0 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求
注意，当中断响应时此位必须通过应用程序清零。

Bit 4 **PTM0PF**：PTM0 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求
注意，当中断响应时此位必须通过应用程序清零。

Bit 3 **PTM1AE**：PTM1 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 2 **PTM1PE**：PTM1 比较器 P 匹配中断控制位
0：除能
1：使能

Bit 1 **PTM0AE**：PTM0 比较器 A 匹配中断控制位
0：除能
1：使能

Bit 0 **PTM0PE**：PTM0 比较器 P 匹配中断控制位
0：除能
1：使能

● **MFI1 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	—	DEF	STMAF	STMPF	—	DEE	STMAE	STMPE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 未定义，读为“0”

Bit 6 **DEF**：数据 EEPROM 中断请求标志位
0：无请求
1：中断请求
注意，当中断响应时此位必须通过应用程序清零。

Bit 5 **STMAF**：STM 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求
注意，当中断响应时此位必须通过应用程序清零。

Bit 4 **STMPF**：STM 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求
注意，当中断响应时此位必须通过应用程序清零。

Bit 3 未定义，读为“0”

Bit 2 **DEE**：数据 EEPROM 中断控制位
0：除能
1：使能

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

Bit 0 **STMPE**：STM 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	—	ATMBF	ATMAF	ATMPF	—	ATMBE	ATMAE	ATMPE
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 **ATMBF**：ATM 比较器 B 匹配中断请求标志位
0：无请求
1：中断请求
注意，当中断响应时此位必须通过应用程序清零。

Bit 5 **ATMAF**：ATM 比较器 A 匹配中断请求标志位
0：无请求
1：中断请求
注意，当中断响应时此位必须通过应用程序清零。

Bit 4 **ATMPF**：ATM 比较器 P 匹配中断请求标志位
0：无请求
1：中断请求
注意，当中断响应时此位必须通过应用程序清零。

Bit 3 未定义，读为“0”

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

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

Bit 0 **ATMPE**：ATM 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。

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，执行优先级如中断结构图所示。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。

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 引脚发生所选的边沿跳转时，外部中断请求标志 INT0F~INT3F 被置位产生外部中断请求。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。外部中断引脚和通用 I/O 口共用，如果相应寄存器中的中断使能位被置位，并且通过引脚共用寄存器选择外部中断脚，此引脚将被作为外部中断脚使用。此时必须通过设置端口控制寄存器中的相关位，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引脚发生所选边沿跳转时，将调用外部中断向量子程序。当响应外部中断服务子程序时，中断请求标志位 INT0F~INT3F 会自动复位且 EMI 位会被清零以除能其它中断。注意，即使此引脚被用作外部中断输入，其上拉电阻选择仍保持有效。

寄存器 INTEG 被用来选择触发外部中断的有效的边沿类型。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。

A/D 转换器中断

A/D 转换动作的结束控制 A/D 转换器中断。当 A/D 转换器中断请求标志 ADF 被置位，即 A/D 转换过程完成时，A/D 转换器中断请求发生。若要跳转到相应中断向量地址，总中断控制位 EMI、A/D 转换器中断使能位 ADE 需先被置位。当中断使能，堆栈未满且 A/D 转换动作结束时，将调用 A/D 转换器中断向量子程序。当 A/D 转换器中断响应时，ADF 标志将自动清除，EMI 将被自动清零以除能其它中断。

多功能中断

此单片机中有 3 个多功能中断，与其它中断不同，它没有独立源，但由其它现有的中断源构成，即 TM 中断和 EEPROM 擦 / 写中断。

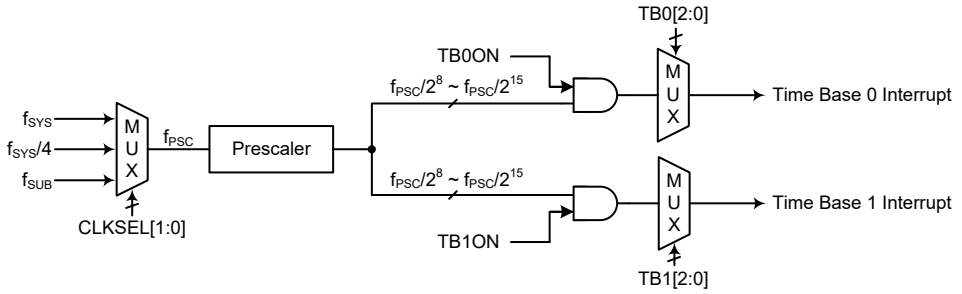
当多功能中断中任何一种中断请求标志 MFnF 被置位，多功能中断请求产生。当中断使能，堆栈未满，包括在多功能中断中的任意一个中断发生时，将调用多功能中断向量中的一个子程序。当响应中断服务子程序时，相关的多功能请求标志位会自动复位且 EMI 位会自动清零以除能其它中断。

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，必须由应用程序清零。

时基中断

时基中断提供一个固定周期的中断信号，由各自的定时器功能产生溢出信号控制。当出现这种情况时其各自的中断请求标志 TB0F 或 TB1F 被置位，中断请求发生。若要跳转到其相应的中断向量地址，总中断使能位 EMI 和时基使能位 TB0E 或 TB1E 需先被置位。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 TB0F 或 TB1F 会自动复位且 EMI 位会被清零以除能其它中断。

时基中断的目的是提供一个固定周期的中断信号。其时钟源 f_{PSC} 来源于内部时钟源 f_{SYS}，$f_{SYS}/4$ 或 f_{SUB}，经过一个分频器，其分频比可通过配置 TB0C 和 TB1C 寄存器中的位选择以获得更长的中断周期。时钟源控制时基中断周期，分别通过 PSCR 寄存器中的 CLKSEL[1:0] 位进行选择。



时基中断

- **PSCR 寄存器**

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

Bit 7~2　未定义，读为“0”

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

- **TB0C 寄存器**

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	—	—	—	—	TB02	TB01	TB00
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7　**TB0ON**：时基 0 控制位
0：除能
1：使能

Bit 6~3　未定义，读为“0”

Bit 2~0　**TB02~TB00**：时基 0 溢出周期选择位
000：$2^8/f_{PSC}$
001：$2^9/f_{PSC}$
010：$2^{10}/f_{PSC}$
011：$2^{11}/f_{PSC}$
100：$2^{12}/f_{PSC}$

101：$2^{13}/f_{PSC}$
110：$2^{14}/f_{PSC}$
111：$2^{15}/f_{PSC}$

- **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 控制位
0：除能
1：使能
Bit 6~3 未定义，读为“0”
Bit 2~0 **TB12~TB10**：时基 1 溢出周期选择位
000：$2^{8}/f_{PSC}$
001：$2^{9}/f_{PSC}$
010：$2^{10}/f_{PSC}$
011：$2^{11}/f_{PSC}$
100：$2^{12}/f_{PSC}$
101：$2^{13}/f_{PSC}$
110：$2^{14}/f_{PSC}$
111：$2^{15}/f_{PSC}$

USB 中断

有几种 USB 情况可以产生 USB 中断。当其中一种情况发生，将会产生一个中断脉冲来引起单片机注意。这些情况包括 USB 暂停，USB 恢复，USB 复位和 USB 端点 FIFO 访问事件。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和 USB 中断使能位 USBE 需先被置位。当中断使能，堆栈未满且上述任一情况发生时，可跳转至 USB 中断向量子程序中执行。当 USB 中断响应，相应的中断请求标志位 USBF 会自动复位且 EMI 位会被清零以除能其它中断。

USB 帧起始中断

USB 帧起始中断即 USB SOF 中断。当检测到一个 USB 的帧起始信号，相应的中断请求标志位 SOFF 被置位时，USB SOF 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和 USB 帧起始中断使能位 SOFE 需先被置位。当中断使能，堆栈未满且检测到一个 USB 的帧起始信号时，可跳转至相关中断向量子程序中执行。当 USB SOF 中断响应，相应的中断请求标志位 SOFF 会自动复位且 EMI 位会被清零以除能其它中断。

通用串行接口模块中断

通用串行接口模块中断，即 USIM0 和 USIM1 中断。当 USIMn 中断标志位 USIMnF 置位时，产生中断请求。由于 USIMn 可工作在三个模式下：SPI 模式、I^2C 模式和 UART 模式，USIMnF 标志位置位可由不同情况触发，取决于所选择的接口模式。

若选择 SPI 或 I^2C 模式，当一个字节数据已由 SPI/I^2C 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 USIMnF 被置位，USIMn 中断请求产生。若选择 UART 模式，USIMn 中断由几种 UART 传输条件控制。当发送器为空、发送器空闲、接收器数据有效、接收器溢出、地址检测和 URXn/UTXn 引脚唤醒，USIMn 中断请求标志 USIMnF 被置位，USIMn 中断请求产生。

若要程序跳转到相应中断向量地址，总中断控制位 EMI 和通用串行接口模块中断使能位 USIMnE 需先被置位。当中断使能，堆栈未满且以上任一种情况发生时，将调用相应的 USIMn 中断向量子程序。当响应中断服务子程序时，通用串行接口模块中断标志位 USIMnF 会自动复位且 EMI 将被自动清零以除能其它中断。

注意，当 USIMn 中断是由 UART 接口触发产生的，当中断响应后，UnUSR 寄存器里的标志位只有在对 UART 执行特定动作时才会被清零，详细参考 USIM 章节的 UART 接口部分。

SPI 接口中断

当一个字节数据已由 SPI 接口接收或发送完，中断请求标志 SPIF 被置位，SPI 中断请求产生。若要程序跳转到相应中断向量地址，总中断控制位 EMI 和串行接口中断使能位 SPIE 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，将调用 SPI 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 SPIF 会自动复位且 EMI 位会被清零以除能其它中断。

EEPROM 中断

EEPROM 中断属于多功能中断。当擦 / 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 擦 / 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

标准型和周期型 TM 各包含两个中断，分别来自比较器 P 或比较器 A 匹配，而音频型 TM 包含三个中断，分别来自比较器 P、比较器 A 或比较器 B 匹配。这些 TM 中断都属于多功能中断。标准型和周期型 TM 有两个中断请求标志位及两个使能位，而音频型 TM 有三个中断请求标志位及三个使能位。当 TM 比较器 P、A 或 B 匹配情况发生时，相应 TM 中断请求标志被置位，TM 中断请求产生。

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

中断唤醒功能

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

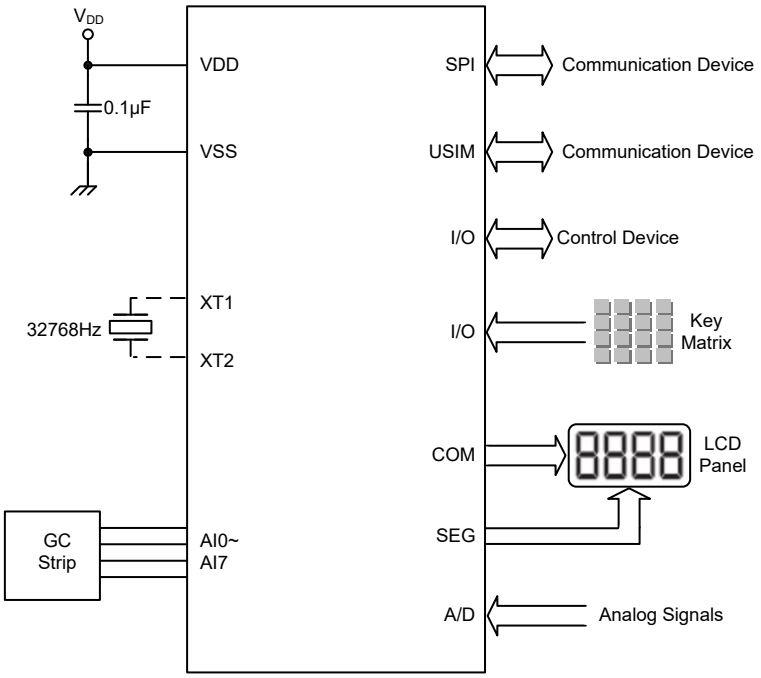
建议在中断服务子程序中不要使用“CALL 子程序”指令。中断通常发生在不可预料的情况或是需要立刻执行的某些应用。假如只剩下一层堆栈且没有控制好中断，当“CALL 子程序”在中断服务子程序中执行时，将破坏原来的控制序列。

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。

应用电路



指令集

简介

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

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
其它指令			
NOP	空指令	1	无
CLR [m]	清除数据存储器	1 注	无
SET [m]	置位数据存储器	1 注	无
CLR WDT	清除看门狗定时器	1	TO, PDF
SWAP [m]	交换数据存储器的高低字节，结果放入数据存储器	1 注	无
SWAPA [m]	交换数据存储器的高低字节，结果放入 ACC	1	无
HALT	进入暂停模式	1	TO, PDF

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

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
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

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

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

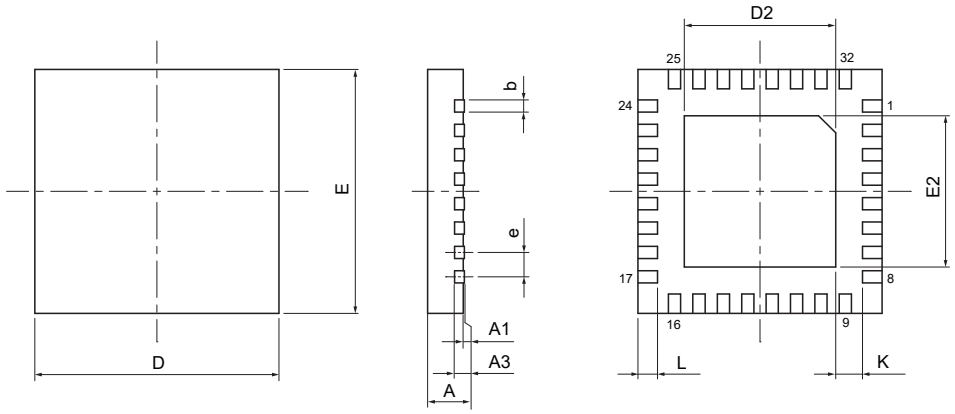
封装信息

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

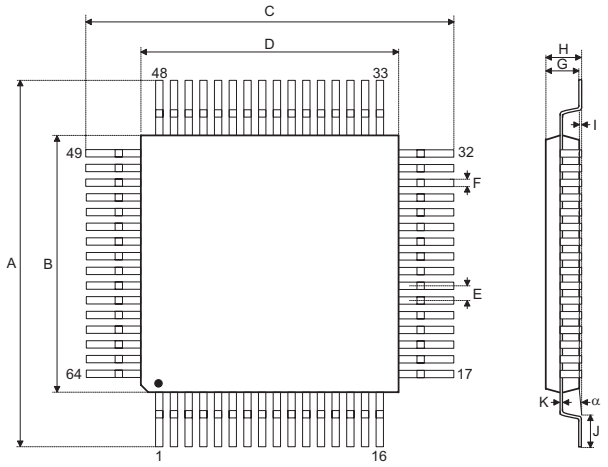
SAW Type 32-pin QFN (4mm×4mm×0.75mm) 外形尺寸



符号	尺寸 (单位：inch)		
	最小值	典型值	最大值
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3		0.008 REF	
b	0.006	0.008	0.010
D		0.157 BSC	
E		0.157 BSC	
e		0.016 BSC	
D2	0.100	—	0.108
E2	0.100	—	0.108
L	0.010	—	0.018
K	0.008	—	—

符号	尺寸 (单位：mm)		
	最小值	典型值	最大值
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3		0.203 REF	
b	0.15	0.20	0.25
D		4.00 BSC	
E		4.00 BSC	
e		0.40 BSC	
D2	2.55	—	2.75
E2	2.55	—	2.75
L	0.25	—	0.45
K	0.20	—	—

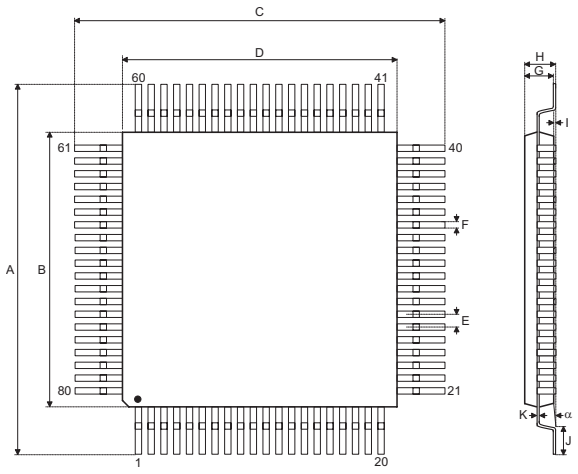
64-pin LQFP (7mm×7mm) 外形尺寸



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		0.016 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		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°