M9F6820

8BIT
TK+AD 型
FLASH MCU

Version 1.07

2026 年 05 月

磐 芯 电 子

本公司保留对产品在可靠性,功能和设计方面的改进作进一步说明的权利
数据手册的更改,恕不另行通知

http://www.masses-chip.com/

本公司不承担由本手册所涉及的产品或电路的运用和使用所引起的任何责任，本公司的产品不是专门设计来应用于外科植入、生命维持和任何本公司产品的故障会对个体造成伤害甚至死亡的领域。如果将本公司的产品应用于上述领域，即使这些是由本公司在产品设计和制造上的疏忽引起的，用户应赔偿所有费用、损失、合理的人身伤害或死亡所直接或间接产生的律师费用，并且用户保证本公司及其雇员、子公司、分支机构和销售商与上述事宜无关。

目录

1 产品简述

采用高速低功耗 CMOS 工艺设计开发的 8 位高性能精简指令单片机，16K*8 位 Flash 程序存储器，512*8 位 RAM，最多 18 个双向 I/O 口，1 个 8 位 Timer 定时器/计数器，2 个 16 位 Timer 定时器/计数器，2 路 USART，1 路 SPI，1 路 IIC，4 路 8 位 PWM，2 路 16 位 PWM，18+3 路 12 位 AD 转换器，18 路触摸，比较器模块，支持多种系统工作模式和多个中断源。

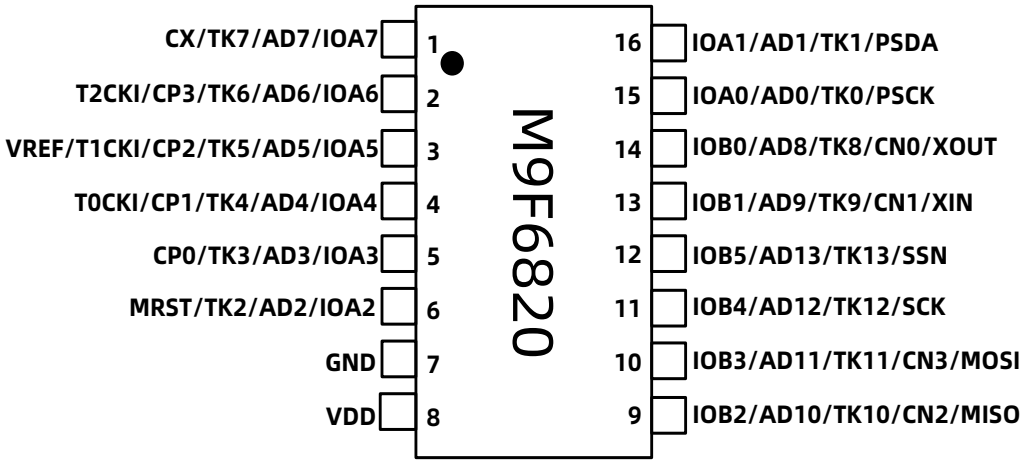
1.1 特性

- CPU 特性
  - 高性能精简指令
  - 高速CPU
  - 15级堆栈缓存器
  - 支持查表指令
- ROM
  - 16K*8位的Flash程序存储器
  - 可编程代码保护(可分区设置)
  - IAP功能
- RAM
  - 512*8位的数据存储器
- I/O 口
  - 最多18个双向I/O口
  - 所有端口可设置弱上拉下拉
  - 4级驱动
  - 所有端口电平变化中断
- 多路中断
  - 2个中断向量与2级优先级可选
  - 3个定时器中断
  - 2路可选电平外部中断
  - 支持所有端口电平变化中断
  - 其它外设中断
- 定时器 TC0
  - 8位计数/16位计数
  - PWM0： 4路8位分辨率
  - PWM0：可整组选择端口
  - 1-8倍后分频
- 定时器 TC1
  - 16位计数
  - 向上，向下，中心对齐计数模式
  - PWM1：2路16位分辨率
  - PWM1：独立互补模式
  - PWM1：可映射到任意端口
  - 驱动WS2812
- 定时器 TC2
  - 16位计数
  - 门控功能
- 看门狗定时器
- USART
  - 宽范围波特率
  - 支持半双工同步模式
  - 可映射到任意端口
- I2C
  - 主从模式
  - 7位地址
  - 标准模式（100kbps）
  - 可映射到任意端口
- SPI
  - 全双工模式
  - 主从模式
  - 多种波特率
- 系统时钟
  - 内部高速RC振荡器： 16MHz
  - 内部低速RC振荡器： 48KHz
  - 外部高速晶体振荡器：4-24MHz
  - 外部低速晶体振荡器：32.768KHz
- 系统工作模式
  - 普通模式：高低速时钟同时工作
  - 低速模式：仅低速时钟工作
  - 休眠模式：高低速时钟都停止工作
- 18+3 路 12 位 ADC
  - 18路外部输入
  - 1路内部电源电压检测VDD/4
  - 1路内部参考
  - 1路GND
- 比较器
  - 4个正相输入源
  - 4个反相输入源
  - 1路内部电源电压检测VDD/2
  - 1路内部GND电压检测
  - 1路可设内部参考电压检测
- CDC 触摸按键模块
  - 最多 18 路通道
  - 无需外接电容
- 封装形式
  - SOP16
  - SOP20
  - TSSOP20

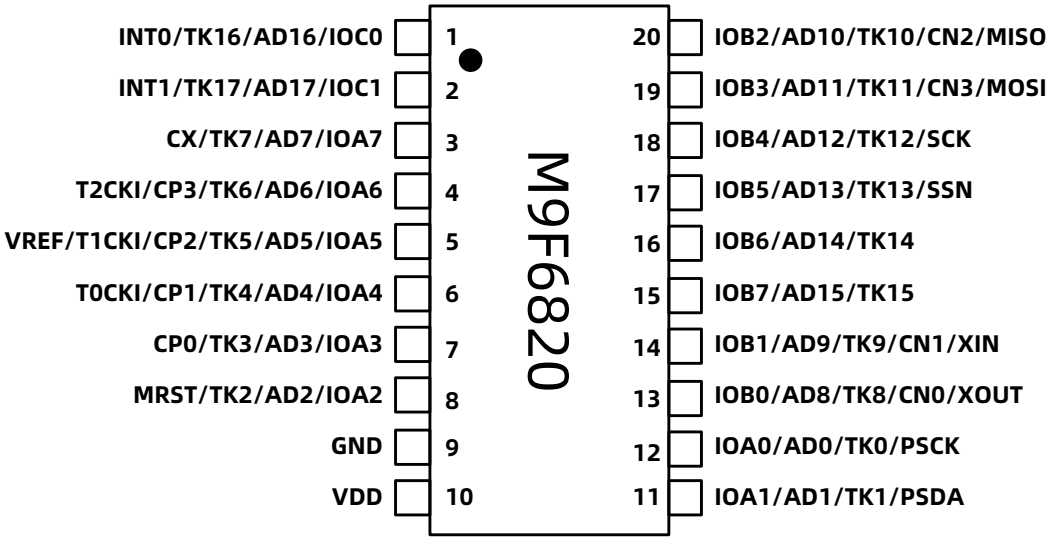
1.2 引脚图

> 注：芯片仿真烧录口分别是VDD、PSDA、PSCK、GND。

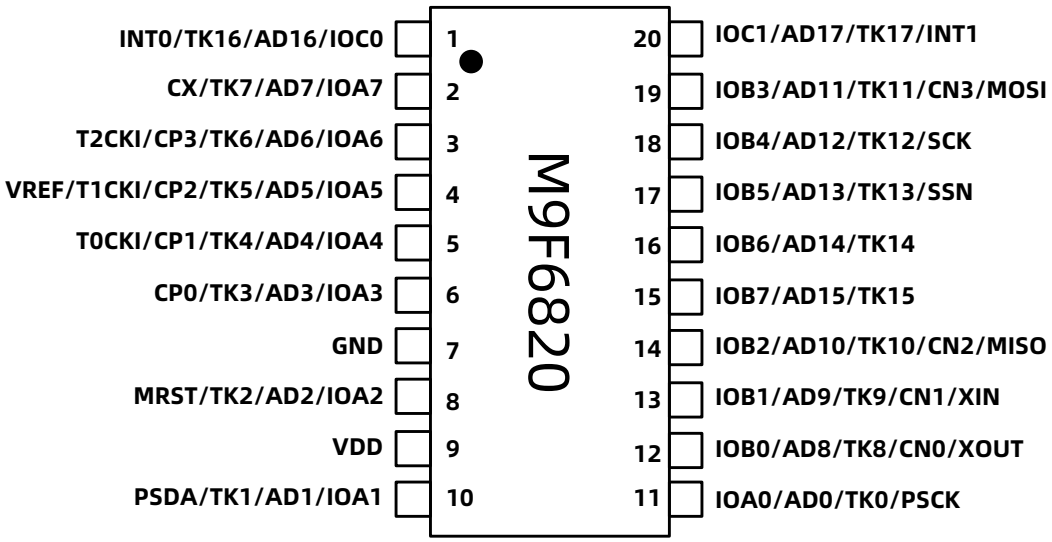
1.2.1 SOP16



1.2.2 SOP20



1.2.3 TSSOP20



注：

（1）引脚图中未注明的 UASRT、I2C、PWM1 与 T2G 相关功能口可映射到任意端口上。

（2）PWM00/PWM01/PWM02/PWM03 只能整体在选择在端口高位或低位上。

（3）以上可选端口功能控制介绍，详见端口 6.9 章节。

1.3 引脚描述

名称	类型	说明
VDD，GND	P	电源输入端
IOA0	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
PSCK	I	编程/仿真串行时钟输入
AD0	A	AD 通道 0
TK0	A	触摸按键通道 0
IOA1	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
PSDA	I/O	编程/仿真串行数据
AD1	A	AD 通道 1
TK1	A	触摸按键通道 1
IOA2	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD2	A	AD 通道 2
TK2	A	触摸按键通道 2
MRST	I	外部复位口(低电平有效，内建上拉到 VDD)
IOA3	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD3	A	AD 通道 3
TK3	A	触摸按键通道 3
CP0	A	比较器 CMP 正相输入端口 0
IOA4	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD4	A	AD 通道 4
TK4	A	触摸按键通道 4
CP1	A	比较器 CMP 正相输入端口 1
T0CKI	I	TC0 外部时钟输入
IOA5	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD5	A	AD 通道 5
TK5	A	触摸按键通道 5
CP2	A	比较器 CMP 正相输入端口 2
T1CKI	I	TC1 外部时钟输入
VREF	A	AD 外部参考电压输入
IOA6	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD6	A	AD 通道 6
TK6	A	触摸按键通道 6
CP3	A	比较器 CMP 正相输入端口 3
T2CKI	I	TC2 外部时钟输入

名称	类型	说明
IOA7	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD7	A	AD 通道 7
TK7	A	触摸按键通道 7
CX	O	比较器 CMP 输出
IOB0	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD8	A	AD 通道 8
TK8	A	触摸按键通道 8
CN0	A	比较器 CMP 反相输入端口 0
XOUT	A	外部晶体振荡器接口
IOB1	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD9	A	AD 通道 9
TK9	A	触摸按键通道 9
CN1	A	比较器 CMP 反相输入端口 1
XIN	A	外部晶体振荡器接口
IOB2	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD10	A	AD 通道 10
TK10	A	触摸按键通道 10
MISO	I/O	SPI 通讯数据口
CN2	A	比较器 CMP 反相输入端口 2
IOB3	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD11	A	AD 通道 11
TK11	A	触摸按键通道 11
MOSI	I/O	SPI 通讯数据口
CN3	A	比较器 CMP 反相输入端口 3
IOB4	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD12	A	AD 通道 12
TK12	A	触摸按键通道 12
SCK	I/O	SPI 通讯时钟口
IOB5	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD13	A	AD 通道 13
TK13	A	触摸按键通道 13
SSN	I	SPI 通讯片选口
IOB6	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD14	A	AD 通道 14
TK14	A	触摸按键通道 14
IOB7	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD15	A	AD 通道 15
TK15	A	触摸按键通道 15

名称	类型	说明
IOC0	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD16	A	AD 通道 16
TK16	A	触摸按键通道 16
INT0	I	外部中断 0
IOC1	I/O	输入/输出 IO、SMT、上拉/下拉电阻、变化中断
AD17	A	AD 通道 17
TK17	A	触摸按键通道 17
INT1	I	外部中断 1

注： I = 输入，O = 输出，I/O = 输入/输出，P = 电源，A = 模拟端口

2 中央处理器（CPU）

2.1 程序存储器

地址	说明
0x0000 ~ 0x0007	用户区
0x0008 ~ 0x0018	中断向量
0x0019 ~ 0xXXXX	用户区
0xXXXX ~ 0xXXXX	类 EE
0xXXXX ~ 0x3FBF	复位向量 Bootload 区
0x3FC0 ~ 0x3FFF	Boot 数据区（1 个 Flash 页面）

存储区域读写权限表

读		被访问地址 tblptr			
		APP	类 EE	BOOTAPP	BOOT 数据
PC 地址	APP	可读	可读	配置字设置	配置字设置
	BOOTAPP	可读	可读	可读	可读
	BOOT 数据	可读	可读	可读	可读

写		被访问地址 tblptr			
		APP	类 EE	BOOTAPP	BOOT 数据
PC 地址	APP	不可写	可写	不可写	不可写
	BOOTAPP	不可写	可写	不可写	可写
	BOOT 数据	配置字设置	可写	不可写	不可写

APP 区(用户区):配置字可设该区域大小,该区域只能通过 BOOT 数据区写入数据,该区域可以读取 APP 区和类 EE 区的数据。可以写入 EE 区。

类 EE 区:配置字可设该区域大小,所有区可以写入,读取。

BOOT 区:配置字可设该区域大小,BOOT 区由两部分组成,BOOT 数据区和 BOOTAPP 区,BOOT 数据区只要设置了 BOOTS≥1 地址就是最后一个页面,其他 BOOT 区空间均是 BOOTAPP 区。

BOOTAPP 区:可以向 BOOT 数据区写入数据。

BOOT 数据区:可以向 APP 区(用户区)写入数据。

下表列出了通过配置字设置各存储区域后所对应的区域大小:

配置字	地址	说明
类 EE 区域=0 BOOT 区域=0	0x0000 ~ 0x3FFF	APP 区(用户区)
类 EE 区域=2KB BOOT 区域=0	0x0000 ~ 0x37FF	APP 区(用户区)
	0x3800 ~ 0x3FFF	类 EE 区
类 EE 区域=2KB BOOT 区域=1KB	0x0000 ~ 0x33FF	APP 区(用户区)
	0x3400 ~ 0x3BFF	类 EE 区
	0x3C00 ~ 0x3FBF	BOOT 区
	0x3FC0 ~ 0x3FFF	Boot -2(1 个 Flash 页面)

2.1.1 复位向量（0000h）

M9F6820 有以下五种复位方式：

- 上电复位
- 看门狗复位
- 外部复位
- 欠压复位
- 软件复位(软复位指令：SRESET)

发生上述任一种复位后，程序将从 0000h 处重新开始执行，系统寄存器也将都恢复为初始默认值。

例：定义复位向量

```
        ORG         0000H
        GOTO        Main_Program        ;//跳转至用户程序开始
        ...
Main_Program:                               ;//用户程序开始
        ...
Main:
        ...
        GOTO        Main                ;//用户主程序循环
```

2.1.2 中断向量（0008h,0018h）

M9F6820 有高低两种中断向量地址。一旦有中断响应，程序计数器 PC 的当前值就会存入堆栈缓存器并跳转到相应的中断向量地址开始执行中断服务程序。

使用触摸库时，低优先级中断被触摸暂用，其它中断不再能使用低向量地址。

例：中断服务程序

```
            ORG         0000h
            GOTO        Main_Program            ;//跳转到程序开始
            ORG         0008h
            GOTO        Interrupt_High          ;//发生高优先级中断后，跳转到中断子程序
            ORG         0018h
            GOTO        Interrupt_Low           ;//发生低优先级中断后，跳转到中断子程序
Main_Program:
            ...
Main:
            ...
            GOTO        Main                ;//主程序循环

Interrupt_High:
            ...
            RETIE

Interrupt_Low:
            ...
            RETIE

            END
```

2.1.3 程序计数器

PCL 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PCL	PCL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PCL[7:0]：** 程序计数器指针低字节

用户将该 PCL 作为目的操作数做加法运算时（ADDRA PCL、ADCRA PCL），14 位 PC 值参与运算，运算结果写入 PC，实现程序的相对跳转；加法运算外的其它运算时，仅 PCL 参与运算，PCH 保持不变。PCH 不可寻址。

PCLATH 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PCLATH	-	-	PCLATH[5:0]					
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

Bit [5:0] **PCLATH[5:0]：** 程序计数器指针高字节锁存PC[13:8]

程序计数器（Program Counter，PC）指定欲取出执行的指令的地址。PC 为 14 位宽，保存在两个不同的 8 位寄存器中。存储低字节的寄存器称为 PCL 寄存器，该寄存器可读写。存储高字节的寄存器，即 PCH 寄存器，存储 PC[13:8]位；该寄存器不可直接读写。更新 PCH 寄存器的操作是通过 PCLATH 寄存器实现的。PCLATH 的内容通过执行写 PCL 的任何操作被传送到程序计数器。PC 是按字节寻址程序存储器的。为了防止 PC 不能正确获取字指令，需要将 PCL 的最低有效位固定取值为 0。PC 每次加 2 来寻址程序存储器中的顺序指令。CALL、RCALL、GOTO 和程序跳转指令直接写入程序计数器。对于这些指令，PCLATH 的内容不会传送到程序计数器。

2.1.4 堆栈

STKPTR 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STKPTR	STKUOV	STKDOV	-	-	STKPTR[3:0]			
读/写	R/W	R/W	-	-	R/W	R/W	R/W	R/W
复位后	0	0	-	-	0	0	0	0

Bit 7 **STKUOV：** 堆栈向上溢出
0 = 堆栈未向上溢出（软件清零或复位清零）
1 = 堆栈上溢出

Bit 6 **STKDOV：** 堆栈向下溢出
0 = 堆栈未向下溢出（软件清零或复位清零）
1 = 堆栈下溢出

Bit [3:0] **STKPTR[3:0]：** 堆栈指针

TOSL 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TOSL	TOS[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **TOSL[7:0]：** 栈顶寄存器低八位

TOSH 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TOSH	-	-	-	TOS[12:8]				
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

Bit [4:0] **TOSH[4:0]：** 栈顶寄存器高五位

返回地址堆栈允许最多 15 个程序调用和中断的任意组合。当执行 CALL、RCALL 指令或响应中断时，PC 值会被压入该堆栈。而在执行 RETURN、RETIA 或 RETIE 指令时，PC 值会从堆栈弹出。PCLATH 不受 RETURN 或 CALL 指令的影响。通过 16 位的 RAM 和一个 4 位的堆栈指针 STKPTR 来实现 15 字的堆栈操作。堆栈既不占用程序存储空间，也不占用数据存储空间。堆栈指针是可读写的，并且通过栈顶（TOS）特殊文件寄存器可以读写栈顶地址。也可以使用这些寄存器将数据压入堆栈或者从堆栈弹出。

执行 CALL 类型指令进行压栈操作，首先堆栈指针加 1，并且将 PC（PC 已经指向 CALL 之后的下一条指令）的内容写入堆栈指针所指向的地址单元。执行 RETURN 类型指令进行出栈操作，STKPTR 寄存器所指向的地址单元的内容会被传送给 PC，然后堆栈指针减 1。所有复位后，堆栈指针均会初始化为 0000。堆栈指针值 0000 不指向任何 RAM 单元，它仅仅是一个复位值。状态位指示堆栈是已满、上溢还是下溢。

有两个寄存器{TOSH:TOSL}用于保存由 STKPTR 寄存器所指向的堆栈单元的内容。这可以让用户在必要时实现软件堆栈。在 CALL、RCALL 或中断后，软件可以通过读取{TOSH:TOSL}寄存器来读取压入堆栈的值。这些值可以被存放到由用户定义的软件堆栈。返回时，软件将这些值存回{TOSH:TOSL}并执行返回。为防止意外的堆栈操作，访问堆栈时用户必须禁止全局中断允许（GIE）位。

STKPTR 寄存器包含堆栈指针值、STKUOV（堆栈满）状态位和 STKDOV（堆栈下溢）状态位。堆栈指针值可为 0 至 15 范围内的任意值。向堆栈压入值前，堆栈指针加 1；而从堆栈弹出值后，堆栈指针减

1。复位时，堆栈指针值为零。用户可以读写堆栈指针的值。向堆栈压入 PC 值 16 次（且没有值从堆栈弹出）后， STKUOV 位置 1。通过软件或 POR 将 STKUOV 位清零。压栈操作会覆盖第 16 次压栈的值，并且 STKPTR 将保持为 15。当堆栈弹出次数足够卸空堆栈时，下一次出栈会向 PC 返回一个零值，并将 STKUOV 位置 1，而堆栈指针则保持为零。STKUOV 位将保持置 1，直到由软件清零或发生 POR 为止。

2.1.5 SPUSH 和 SPOP 指令

由于栈顶是可以读写的，因此将值压入堆栈或从堆栈弹出而不影响程序的正常执行是非常理想的。指令集包含两条指令 SPUSH 和 SPOP，使用这两条指令可在软件控制下对堆栈顶 TOS 执行操作。然后就可以修改 TOSH 和 TOSL，将数据或返回地址压入堆栈。SPUSH 指令将当前的 PC 值压入堆栈。执行该指令会使堆栈指针加 1 并将当前的 PC 值装入堆栈。SPOP 指令通过将堆栈指针减 1 来放弃当前的 TOS 值，然后前一个入栈的值就成为了 TOS 值。

2.1.6 快速寄存器堆栈

M9F6820 为 STATUS、AREG 和 BSR 寄存器提供的快速寄存器堆栈具有从中断“快速返回”的功能。每个寄存器的堆栈只有一级且不可读写。当处理器转入中断向量处执行时，它装入对应寄存器的当前值。所有中断源都会将值压入堆栈寄存器。如果使用 RETIE，FAST 指令从中断返回，这些寄存器中的值就会被装回相关的寄存器。如果同时允许低优先级中断和高优先级中断，从低优先级中断返回时，无法可靠地使用堆栈寄存器。如果在为低优先级中断提供服务时，发生了高优先级中断，则低优先级中断存储在堆栈寄存器中的值将被覆盖。在为低优先级中断提供服务时，用户必须用软件保存关键寄存器的值。如果未使用中断优先级，所有中断都可以使用快速寄存器堆栈从中断返回。如果没有使用中断，快速寄存器堆栈可以用于在子程序调用结束后恢复 STATUS、AREG 和 BSR 寄存器。要在子程序调用中使用快速寄存器堆栈，必须执行 CALL label，FAST 指令将 STATUS、AREG 和 BSR 寄存器的内容存入快速寄存器堆栈。然后执行 RETURN，FAST 指令，从快速寄存器堆栈恢复这些寄存器。

2.1.7 查表

利用 RDT 指令可以读取程序区数据，TBLPTRH 和 TBLPTRL 组成 16 位表地址，读取数据到 TABLAT 寄存器。

例 1：查找 ROM 地址为“DTAB”的值

```
            MOVIA       HIGH(DTAB) ;获取数据表地址高位
            MOVAR       TBLPTRH         ;设置数据表高位指针
            MOVIA       LOW(DTAB)       ;获取数据表地址低位
            MOVAR       TBLPTRL         ;设置数据表低位指针
                                    ;若需读取表的其它数据，修改指针
            RDT*+                       ;读取表的第一个数据0x01，读取后指针+1
            MOVR        TABLAT,0        ;将数据0x01放在A
            ...
DTAB:
            DB          0x01,0x02,0x03,0x04,0x05,0x06,0x07.....
```

2.2 数据存储器

2.2.1 数据存储器结构

数据存储器是用静态 RAM 实现的。在数据存储器中，每个寄存器都有 12 位地址，允许数据存储器实现为最大 4096 个字节。存储空间最多被分为 16 个存储区，每个存储区包含 256 个字节。数据存储器由特殊功能寄存器(Special Function Register, SFR)和通用寄存器(General Purpose Register, GPR)组成。SFR 用于单片机和外设功能模块的控制和状态指示，GPR 则用于用户应用程序的数据存储和中间结果暂存。任何未实现的存储单元均读为 0。指令集和架构支持跨所有存储区的操作。可以通过直接、间接寻址模式访问整个数据存储器。确保能在一个周期中访问常用寄存器（SFR 和某些 GPR），本器件实现了一个快速操作存储区。该存储区是一个 256 字节的存储空间，它可实现对 SFR 和 GPR Bank 0 的低地址单元的快速访问，而无需使用存储区选择寄存器（Bank Select Register，BSR）。

2.2.2 BSR 页面选择寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BSR	-	-	-	-	BSR[3:0]			
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

Bit [3:0] **BSR[3:0]:** 直接寻址的页面选择

容量较大的数据存储器需要高效的寻址机制，以便对所有地址进行快速访问。理想状况下，这意味着不必为每次读写操作提供完整地址。本器件是使用 RAM 存储区分区机制实现快速访问的。这种机制将存储空间分成连续的 16 个 256 字节的存储区。根据不同的指令，可以通过完整的 12 位地址，或通过 8 位的低字节地址和 4 位存储区指针直接寻址每个存储单元。指令集中的大部分指令都使用存储区指针，也就是存储区选择寄存器（BSR）。BSR 保存单元地址的高 4 位，而指令本身则包括单元地址的低 8 位。只使用 BSR 的低 4 位（BSR[3:0]），不使用高 4 位；高 4 位总是读为 0 且不能被写入。可以通过使用 BANKBSR 指令直接装入 BSR。BSR 的值代表数据存储器中的存储区；指令中的 8 位指向存储区中的存储单元，可以将它看作距离存储区下边界的偏移量。由于最多可有 16 个寄存器共享同一个低位地址，用户必须非常小心以确保在执行数据读或写之前选择了正确的存储区。

当选择存储区时，只有已实现的存储区才可以读写。对未实现的存储区进行的写操作将被忽略，而读这些存储区会返回 0。虽然是这样，这些操作仍然会对 STATUS 寄存器起作用，就好像操作成功了一样。指令中只有 MOVRR 指令指定源寄存器和目标寄存器的完整 12 位地址。该指令在执行时完全忽略 BSR。所有其他指令仅包含作为操作数的低位地址，而且必须使用 BSR 或快速操作存储区来寻址目标寄存器。

2.2.3 快速操作寄存器

使用 BSR 和嵌入的 8 位地址，用户可以寻址数据存储器的整个空间，但这同时也意味着用户必须始终确保选择了正确的存储区。否则可能会从错误的单元读取数据或将数据写入错误的单元。如果本来是向 GPR 进行写操作，却将结果写入了 SFR，后果是非常严重的。但是在每次对数据存储器进行读或写操作时验证或更改 BSR 会严重影响工作效率。为了提高访问大多数常用数据存储单元的效率，现为数据存储器配置了快速操作存储区，这样可以允许用户访问被映射的存储区而无需指定 BSR。快速操作存储区由 Bank 0 的前 96 个字节（00h-5Fh）和 Bank 15 的后 160 个字节（60h-FFh）组成。地址较低的部分被称为“快速操作 RAM”，由 GPR 组成。地址较高的部分则被映射为器件的 SFR。这两个区域被连续地映射到快速操作存储区并且可以用一个 8 位地址进行线性寻址。包括快速操作 RAM 位（指令中的“a”参数）的指令使用快速操作存储区。当“a”等于 1 时，指令使用 BSR 和包含在操作码中的 8 位地址对数据存储器寻址。当“a”为 0 时，强制指令使用快速操作存储区地址映射，此时完全忽略 BSR 的当前值。此“强制”寻址模式可使指令在一个周期内对数据地址进行操作，而不需要首先更新 BSR。这意味着用户可以更高效地对 8 位地址为 60h 或以上的 SFR 进行取值和操作。地址为 60h 以下的快速操作 RAM 非常适合于存储那些用户可能需要快速访问的数据值，如直接计算结果或常用程序变量。快速操作 RAM 也可实现更加快速和高效的现场保护和变量切换代码。

2.2.4 数据存储器寻址模式

本器件主要有三种寻址模式：立即数寻址，直接寻址，间接寻址。
立即数寻址：主要是 MOVRR 指令时的寻址。
直接寻址：给出八位地址，根据 BSR 或者快速操作寄存器确定地址。
间接选址：有三组 12 位数据指针，FSR0，FSR1，FSR2。

寻址模式	地址组成
立即数寻址	INST[11:0]
直接寻址	BSR0[3:0]，INST[7:0]
间接寻址 0	FSR0H，FSR0L
间接寻址 1	FSR1H，FSR1L
间接寻址 2	FSR2H，FSR2L

页	地址	用途	说明
BANK0	0x000-0x05F	Access RAM	快速操作寄存器
	0x060-0x0FF	GPR	-
BANK1	0x100-0x1FF	GPR	-
BANK2	0x200-0x2FF	-	未实现，读为 0
BANK3	0x300-0x3FF	-	未实现，读为 0
BANK4	0x400-0x4FF	-	未实现，读为 0
BANK5	0x500-0x5FF	-	未实现，读为 0
BANK6	0x600-0x6FF	-	未实现，读为 0
BANK7	0x700-0x7FF	-	未实现，读为 0
BANK8	0x800-0x8FF	-	未实现，读为 0
BANK9	0x900-0x9FF	-	未实现，读为 0
BANK10	0xA00-0xAFF	-	未实现，读为 0
BANK11	0xB00-0xBFF	-	未实现，读为 0
BANK12	0xC00-0xCFF	-	未实现，读为 0
BANK13	0xD00-0xDFF	-	未实现，读为 0
BANK14	0xE00-0xEFF	-	未实现，读为 0
BANK15	0xF00-0xF5F	-	未实现，读为 0
	0xF60-0xFFF	SFR	快速操作寄存器

2.2.5 间接寻址 0/1/2

FSRxL 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSRxL	FSRxL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **FSRxL[7:0]：** FSRx 寄存器低七位地址（x = 0/1/2）

FSRxH 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSRxH	-	-	-	-	FSRxH[3:0]			
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

Bit [3:0] **FSRxH[3:0]：** FSR0 寄存器高四位地址（x = 0/1/2）

注：INDFx，POSTINCx，POSTDECx，PREINCx，PLUSWx 不是实际存在的寄存器，对这些寄存器的操作实际上是对以 FSRx 为地址的寄存器的操作。（x = 0/1/2，下同）

访问 INDFx 寄存器时，实现间接寻址 x，访问到的是{FSRxH:FSRxL}寄存器值作为数据指针所指向的寄存器内容，间接寻址模式 x 可寻址所有页面空间。

访问 POSTINCx 寄存器时，实现间接寻址模式 x，访问到的是{FSRxH:FSRxL}寄存器值作为数据指针所指向的寄存器内容，访问后，FSRx 的值自加一，间接寻址模式 x 可寻址所有页面空间。

访问 POSTDECx 寄存器时，实现间接寻址模式 x，访问到的是{FSRxH:FSRxL}寄存器值作为数据指针所指向的寄存器内容，访问后，FSRx 的值自减一，间接寻址模式 x 可寻址所有页面空间。

访问 PREINCx 寄存器时，实现间接寻址模式 x，访问前，FSRx 的值自加一，访问到的是{FSRxH:FSRxL}寄存器值作为数据指针所指向的寄存器内容，间接寻址模式 x 可寻址所有页面空间。

访问 PLUSWx 寄存器时，实现间接寻址模式 x，访问到的是{FSRxH:FSRxL}+AREG 寄存器（范围为-127 至 128）值作为数据指针所指向的寄存器内容，间接寻址模式可寻址所有页面空间。

2.2.6 8x8 硬件乘法器

本器件包含一个8* 8 硬件乘法器（是ALU的一部分）。该乘法器可在一个指令周期内执行无符号运算并产生一个16位运算结果，该结果存储在一对乘积寄存器{PRODH:PRODL}中。该乘法器执行的运算不会影响STATUS 寄存器中的任何标志。

PRODL 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PRODL	PRODL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	x	x	x	x	x	x	x

Bit [7:0] **PRODL[7:0]：** 乘法结果寄存器低八位

PRODH 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PRODH	PRODH[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	x	x	x	x	x	x	x	x

Bit [7:0] **PRODH[7:0]：** 乘法结果寄存器高八位

2.2.7 STATUS 状态寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STATUS	-	-	-	N	OV	Z	DC	C
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	x	x	x	x	x

Bit 4 **N：** 负标志，此位用于有符号的算术运算（以二进制补码方式进行），它可以表示结果是否为负（ALU MSB = 1）
0 = 运算结果为正
1 = 运算结果为负

Bit 3 **OV：** 溢出标志，此位用于有符号的算术运算（以二进制补码方式进行），它表明运算结果溢出了7位二进制数的，范围溢出导致符号位（bit 7）发生改变。
0 = 运算未溢出
1 = 运算结果溢出

Bit 2 **Z：** 零标志
0 = 算术/逻辑运算的结果非零
1 = 算术/逻辑运算的结果为零

Bit 1 **DC：** 辅助进位标志
0 = 加法运算时低四位没有进位，减法时有向高四位借位
1 = 加法运算时低四位有进位，减法时没有向高四位借位

Bit 0 **C：** 进位标志
0 = 加法运算后没有进位，减法时有借位
1 = 加法运算后有进位，减法时没有借位

2.2.8 系统寄存器定义

数据寄存器映射表（F60h – FFFh）									
地址	寄存器	地址	寄存器	地址	寄存器	地址	寄存器	地址	寄存器
FFFh	-	FDFh	INDF2	FBFh	-	F9Fh	-	F7Fh	I2CCR
FFEh	TOSH	FDEh	POSTINC2	FBEh	PUC	F9Eh	-	F7Eh	I2CSTA
FFDh	TOSL	FDDh	POSTDEC2	FBDh	PUB	F9Dh	-	F7Dh	I2CDATA
FFCh	STKPTR	FDCh	PREINC2	FBCh	PUA	F9Ch	-	F7Ch	I2CADR
FFBh	-	FDBh	PLUSW2	FBBh	-	F9Bh	-	F7Bh	SPICR
FFAh	PCLATH	FDAh	FSR2H	FBAh	PDC	F9Ah	-	F7Ah	SPISTA
FF9h	PCL	FD9h	FSR2L	FB9h	PDB	F99h	TC2PRH	F79h	SPIDATA
FF8h	-	FD8h	STATUS	FB8h	PDA	F98h	TC2PRL	F78h	
FF7h	TBLPTRH	FD7h	INTP2	FB7h	-	F97h	TC2CH	F77h	RX1REG
FF6h	TBLPTRL	FD6h	INTP1	FB6h	-	F96h	TC2CL	F76h	TX1REG
FF5h	TABLAT	FD5h	INTP0	FB5h	IOCODS1	F95h	TC2GCR	F75h	RX1CR
FF4h	PRODH	FD4h	INTF2	FB4h	IOCODS0	F94h	T2CR	F74h	TX1CR
FF3h	PRODL	FD3h	INTF1	FB3h	IOBODS1	F93h	TC1PRH	F73h	RX0REG
FF2h	OPTION	FD2h	INTF0	FB2h	IOBODS0	F92h	TC1PRL	F72h	TX0REG
FF1h	EECON1	FD1h	INTCR2	FB1h	IOAODS1	F91h	TC1CH	F71h	RX0CR
FF0h	EECON2	FD0h	INTCR1	FB0h	IOAODS0	F90h	TC1CL	F70h	TX0CR
FEFh	INDF0	FCFh	INTCR0	FAFh	-	F8Fh	-	F6Fh	TKCNTH
FEEh	POSTINC0	FCEh	-	FAEh	ANSC	F8Eh	T1CR	F6Eh	TKCNTL
FEDh	POSTDEC0	FCDh	-	FADh	ANSB	F8Dh	-	F6Dh	TKCHS
FECh	PREINC0	FCCh	-	FACh	ANSA	F8Ch	-	F6Ch	
FEBh	PLUSW0	FCBh	-	FABh		F8Bh	TC0CH	F6Bh	
FEAh	FSR0H	FCAh	IOC	FAAh		F8Ah	TC0CL	F6Ah	
FE9h	FSR0L	FC9h	IOB	FA9h		F89h	T0CR2	F69h	TKCHS2
FE8h	AREG	FC8h	IOA	FA8h		F88h	T0CR	F68h	TKCHS1
FE7h	INDF1	FC7h	-	FA7h		F87h	-	F67h	TKCHS0
FE6h	POSTINC1	FC6h	OEC	FA6h		F86h	FLBUFH	F66h	TKCCTR
FE5h	POSTDEC1	FC5h	OEB	FA5h	ADH	F85h	FLBUFL	F65h	TKRCTR
FE4h	PREINC1	FC4h	OEA	FA4h	ADL	F84h	FLOPEN	F64h	TKCTR4
FE3h	PLUSW1	FC3h	-	FA3h	-	F83h	-	F63h	TKCTR3
FE2h	FSR1H	FC2h	IOCOR	FA2h	ADCON2	F82h	-	F62h	TKCTR2
FE1h	FSR1L	FC1h	IOBOR	FA1h	ADCON1	F81h	PCON	F61h	TKCTR1
FE0h	BSR	FC0h	IOAOR	FA0h	ADCON0	F80h	OSCM	F60h	TKCTR0

注：空白处为未实现地址，读为 0，无法写。

数据寄存器映射表（EC0h – F5Fh）									
地址	寄存器	地址	寄存器	地址	寄存器	地址	寄存器	地址	寄存器
F5Fh		F3Fh	FLIPCR	F1Fh	-	EFFh	CALLOCK	EDFh	-
F5Eh		F3Eh		F1Eh	-	EFEh	IRCCAL	EDEh	-
F5Dh		F3Dh		F1Dh	PWM11DH	EFDh	IRCCAH	EDDh	-
F5Ch		F3Ch		F1Ch	PWM11DL	EFCh	LVR	EDCh	-
F5Bh		F3Bh		F1Bh	PWM10DH	EFBh	FCPU	EDBh	-
F5Ah		F3Ah		F1Ah	PWM10DL	EFAh	VREFCAL	EDAh	-
F59h		F39h		F19h	-	EF9h		ED9h	-
F58h		F38h		F18h	PWM1CR	EF8h	-	ED8h	-
F57h		F37h	-	F17h	PWM03D	EF7h	-	ED7h	-
F56h		F36h	IOCICR	F16h	PWM03CR	EF6h	-	ED6h	-
F55h		F35h	IOBICR	F15h	PWM02D	EF5h		ED5h	-
F54h		F34h	IOAICR	F14h	PWM02CR	EF4h		ED4h	
F53h		F33h	-	F13h	PWM01D	EF3h	IDEH	ED3h	
F52h		F32h	IOCIPS	F12h	PWM01CR	EF2h	IDEL	ED2h	
F51h		F31h	IOBIPS	F11h	PWM00D	EF1h	VERSIONH	ED1h	
F50h		F30h	IOAIPS	F10h	PWM00CR	EF0h	VERSIONL	ED0h	
F4Fh		F2Fh		F0Fh		EEFh		ECFh	
F4Eh		F2Eh	CMPC2	F0Eh		EEEh		ECEh	
F4Dh		F2Dh	CMPC1	F0Dh		EEDh		ECDh	
F4Ch		F2Ch	CMPC0	F0Ch		EECh		ECCh	
F4Bh	MPT2G	F2Bh		F0Bh		EEBh		ECBh	
F4Ah	MPPWM0	F2Ah		F0Ah		EEAh		ECAh	
F49h	MPPWM11	F29h		F09h		EE9h		EC9h	
F48h	MPPWM10	F28h		F08h		EE8h		EC8h	
F47h		F27h		F07h		EE7h		EC7h	
F46h		F26h		F06h		EE6h		EC6h	
F45h	MPSDA	F25h		F05h		EE5h		EC5h	
F44h	MPSCL	F24h		F04h		EE4h		EC4h	
F43h	MPRX1	F23h	BRGD1H	F03h		EE3h		EC3h	
F42h	MPTX1	F22h	BRGD1L	F02h		EE2h		EC2h	
F41h	MPRX0	F21h	BRGD0H	F01h		EE1h		EC1h	
F40h	MPTX0	F20h	BRGD0L	F00h		EE0h		EC0h	

注：空白处为未实现地址，读为 0，无法写。

2.3 Flash 自编程（IAP）

Flash 编程只能通过页写操作，即 Flash 写操作每次都是对一个页面的地址进行写操作，每一页的大小为 64 个字节。读为自由地址读取，可直接指针操作。

2.3.1 EECON1 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	PRGT4	PRGT3	PRGT2	PRGT1	PRGT0	CLRPL	ERASE	WRITE
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	1	0	0	0	0	0	0

Bit [7:3] **PRGT[4:0]**：编程、擦除时间控制，默认为为01000b
编程时间=（1364+21*PRGT）us
擦除时间=（2208+42*PRGT）us

Bit 2 **CLRPL**：IAP清除Flash内部缓冲区操作，IAP写入操作必须置1
0 = 不清除Flash内部缓冲区
1 = 清除Flash内部缓冲区

Bit 1 **ERASE**：IAP擦除操作
0 = 不允许IAP擦除操作
1 = 允许IAP擦除操作

Bit 0 **WRITE**：IAP写入操作
0 = 不允许IAP写入操作
1 = 允许IAP写入操作

2.3.2 EECON2 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON2	WERR	-	-	-	EELOCK3	EELOCK2	EELOCK1	-
读/写	R	-	-	-	W	W	W	-
复位后	0	-	-	-	0	0	0	-

Bit 7 **WERR**：Flash写操作出错标志，写EECON1时该位自动置0。

Bit 3 **EELOCK3**：解锁流程1

Bit 2 **EELOCK2**：解锁流程2

Bit 1 **EELOCK1**：解锁流程3

2.3.3 FLBUFL

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FLBUFL	FLBUFL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

{FLBUFH[3:0], FLBUFL[7:0]} :Flash buf区起始地址

2.3.4 FLBUFH

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FLBUFH	-	-	-	-	FLBUFH[3:0]			
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

{FLBUFH[3:0], FLBUFL[7:0]} ：Flash buf区起始地址

例如{FLBUFH[3:0], FLBUFL[7:0]} = 0x100,则Flash 缓存区为0x100 - 0x13F

2.3.5 FLOPEN

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FLOPEN	FLOPEN [7：0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **FLOPEN** [7：0]：FLASH操作允许寄存器

FLOPEN [7：0] = 0X55： 允许写类EE区

FLOPEN [7：0] = 0XAA： 允许BOOT区或者APP程序区（APPM[1:0]！=00时）擦写APP程序区

FLOPEN [7：0] = 0X5A： 允许BOOT区（INFOM = 0时有效）擦写配置区

FLOPEN [7：0] = 其它 ： 不允许擦写APP程序区与类EE区

Bit [0] **FLOPEN[0]：** 复位向量控制位（VRESETS = 1时有效）

0 = 复位向量为BOOT地址

1 = 复位向量为0x0000地址

该寄存器要按以下步骤写0XAA值，否则值为复位值，

```
BANKBSR     0X0E
MOVIA       0X55
MOVAR       CALLOCK         ;CALLOCK写入55H
MOVIA       0XAA
MOVAR       CALLOCK         ;CALLOCK写入AAH
MOVIA       0XAA
MOVAR       FLOPEN          ;FLOPEN写入AAH
```

2.3.6 Flash 写入缓存器

由{FLBUFH[3:0], FLBUFL[7:0]}指定起始地址，大小为 64 字节的数据存储器为 Flash 写入缓存器，操作方式与普通 RAM 方式相同。Flash 写入缓存器暂用通用寄存器的 RAM。

2.3.7 INFO0 区修改方法

修改配置字区只能在 BOOT 中进行，对应地址如下。以{FLBUFH[3:0], FLBUFL[7:0]} = 0x060 为例

RAM 地址	对应 OPT	RAM 地址	对应
0x060	OPT0L	0x80	~OPT0L
0x061	OPT0H	0x81	~OPT0H
0x062	OPT1L	0x82	~OPT1L
0x063	OPT1H	0x83	~OPT1H
0x064	OPT2L	0x84	~OPT2L
0x065	OPT2H	0x85	~OPT2H
0x066	OPT3L	0x86	~OPT3L
0x067	OPT3H	0x87	~OPT3H
0x068	OPT4L	0x88	~OPT4L
0x069	OPT4H	0x89	~OPT4H
0x6a-0x7f	-	0x8a-0x9f	-

注：对应寄存器一定要取反写入。

2.3.8 Flash 操作示例

例：擦 Flash 操作

```
        BCLR        OPTION,GIE          ;操作前要关中断
        BANKBSR     0X0E
        MOVIA       0x55                ;FLOPEN 写入时序
        MOVAR       CALLOCK
        MOVIA       0xAA
        MOVAR       CALLOCK
        MOVIA       0x55                ;允许操作类EE区
        MOVAR       FLOPEN

        BSET        EECON1,1            ;擦除Flash使能
        MOVR        TBLPTRHBUF,A        ;写高位地址
        MOVAR       TBLPTRH
        MOVR        TBLPTRLBUF,A        ;写低位地址
        MOVAR       TBLPTRL
        BSET        EECON2,3            ;Flash解锁流程1
        BSET        EECON2,2            ;Flash解锁流程2
        BSET        EECON2,1            ;Flash解锁流程3
        WDT*                            ;写入，CPU暂停，擦除结束后再运行下一条指令
        NOP
        ...
```

例： Flash页写操作

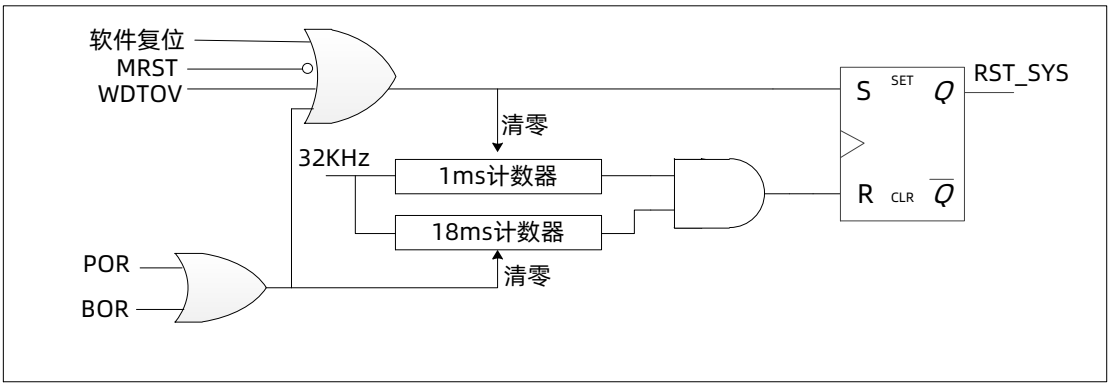
```
        BCLR    OPTION,GIE          ;操作前要关中断
        MOVIA   HIGH(BUFADDR)       ;指定Flash缓存区起始地址
        MOVAR    FLBUFL
        MOVIA   LOW(BUFADDR)
        MOVAR    FLBUFH
        BANKBSR HIGH(BUFADDR)
        MOVIA   DATA0
        MOVAR   BUFADDR +0
        ...
        MOVIA   DATA63
        MOVAR   BUFADDR +63         ;写Flash页面缓冲区64个字节

        BANKBSR 0X0E
        MOVIA   0x55                ;FLOPEN 写入时序
        MOVAR   CALLOCK
        MOVIA   0xAA
        MOVAR   CALLOCK
        MOVIA   0x55                ;允许操作类EE区
        MOVAR   FLOPEN

        BSET    EECON1,2            ;清缓存区
        BSET    EECON1,0            ;写Flash使能

        MOVIA   HIGH(DTAB)          ;写Flash页高位地址
        MOVAR   TBLPTRH
        MOVIA   LOW(DTAB)           ;写Flash页低位地址
        MOVAR   TBLPTRL
        BSET    EECON2,3            ;Flash解锁流程1
        BSET    EECON2,2            ;Flash解锁流程2
        BSET    EECON2,1            ;Flash解锁流程3
        WDT*                        ;写入，CPU暂停，写入结束后再运行下一条指令
        NOP
        ...                         ;必要时软件校验是否写入成功
```

3 复位

3.1 复位方式

- 上电复位（POR）（复位时间 5V 供电下为 20ms 左右，3V 供电下 35ms 左右）
- 外部复位（MCLRB Reset）
- 欠压复位（BOR）
- 看门狗定时器复位（WDT Reset）
- 软件复位(软复位指令：SRESET)

M9F6820有以上5种复位方式，任何一种复位都会使PC程序计数器清零，让程序从0000h或BOOT区（由配置字和寄存器决定）处开始运行，并且使系统寄存器值复位，默认从0000h开始运行。



3.2 PCON 复位状态寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PCON	-	SOFTRST	TO	PD	APPRST	MCLR	POR	BOR
读/写	-	R/W	R	R	R	R	R	R
复位后	0	0	1	1	0	0	0	1

Bit 6 **SOFTRST**: 软复位
1= 上电复位、掉电复位（软件置一）
0 = 发生软复位

Bit 5 **TO**: 超时位
1 = 上电复位或清除WDT
0 = WDT发生溢出

Bit 4 **PD**: 掉电位
1 = 上电复位或清除WDT
0 = 进入休眠模式

Bit3 **APPRST**: 分区复位
1=未发生分区复位（EE复位和BOOT复位）
0=发生分区复位（软件置一）

Bit2 **MCLR**: 外部复位位
1=未发生外部复位（硬件清零）
0=发生外部复位（软件置一）

Bit1 **POR**： 上电复位位
1=未发生上电复位（硬件清零）
0=发生上电复位（软件置一）

Bit0 **BOR**：欠压复位位
1=未发生欠压复位（硬件清零）
0=发生欠压复位（软件置一）

3.3 复位地址说明

上电时，配置 VRESETS=1，则直接会从 BOOT 区运行。当出 BOOT 区后，对寄存器 BOOTEN 操作可以让程序通过软件复位从 0 地址或者 BOOT 区开始执行。

当配置 VRESETS=0 时，BOOTEN 操作无效，只能从 0 地址开始执行。

4 系统时钟

4.1 概述

M9F6820支持双时钟系统：高速时钟和低速时钟。高速时钟由外部高速晶体振荡器（HXT）或内置高速RC振荡器（HIRC，16MHz）提供；低速时钟由外部低速晶体振荡器（LXT，如：32768Hz）或内置的低速RC振荡器（LIRC，48KHz）提供。两种时钟都可作为系统时钟源Fosc，系统工作在低速模式时，Fosc 2分频后为一个指令周期。低频系统时钟源和高频系统源可根据芯片配置字进行配置。

注：
（1）工作时勿在进行高低频切换同时 STOP CPU 操作，可能会造成系统紊乱。
（2）使用触摸功能时，高速时钟只能选择内部 HIRC 16MHz 时钟。

4.2 OSCM 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCM	STBH	STBL	-	STOP	CLKM	STPH	-	STPL
读/写	R	R	-	R/W	R/W	R/W	-	R/W
复位后	x	x	-	0	x	1	-	1

Bit 7 **STBH：**高频振荡器稳定标志

Bit 6 **STBL：**低频振荡器稳定标志

Bit 5 **Reserved：**保留，请保持为0

Bit 4 **STOP：**CPU工作状态标志位
0 = CPU正常工作，所有复位唤醒
1 = CPU停止工作

Bit 3 **CLKM：**系统时钟状态标志位
0 = CPU运行于高频时钟
1 = CPU运行于低频时钟

Bit 2 **STPH：**高频振荡器控制
0 = 休眠状态或低速模式下高速振荡器仍然工作
1 = 休眠状态或低速模式下关闭高频振荡器

Bit 0 **STPL：**低频振荡器控制
0 = 休眠状态下低频振荡器仍然工作
1 = 休眠状态下低频振荡器停止工作

注：CLKM 的初始状态由配置字决定。

4.3 系统时钟的工作模式

4.3.1 普通模式

普通模式有两种分别是：

（1）系统时钟选择高频时钟源，STOP = 0。（电流特性参考电性参数表 I_{DD1}、 I_{DD2}）

（2）系统时钟选择低频时钟源，STOP = 0。（电流特性参考电性参数表 I_{DD3}）

4.3.2 绿色模式

绿色模式有两种分别是：

（1）高频时钟工作，低频时钟工作，STOP = 1。（电流特性参考电性参数表 I_{SP1}）

（2）高频时钟停止，低频时钟工作，STOP = 1。（电流特性参考电性参数表 I_{SP2}）

绿色模式可以由所有中断或 WDT 唤醒，此模式下只停止 CPU，外设依旧可以运行（外设所选时钟需开启）。

4.3.3 休眠模式

休眠模式只有一种是：

（1）高频时钟停止，低频时钟停止，STOP = 1。（电流特性参考电性参数表 I_{SP3}）

休眠模式可以由外部中断、IO 变化中断或 WDT 唤醒。

注：

（1）省电建议，程序运行时跑高频，快速跑完程序然后进休眠，此时休眠下需设置高频时钟停止工作。

（2）各工作模式的工作电流参考电性参数表。

（3）绿色和休眠模式下，如果总中断不开启，所有中断唤醒能唤醒芯片但是不会进中断。

4.4 FCPU 系统时钟分频寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FCPU	-					FCPU2	FCPU1	FCPU0
读/写	-					RW	RW	RW
复位后	-					X	X	X

Bit[2:0] **FCPU[2:0]**：高频时钟分频选择位

FCPU[2:0]	高频时钟分频选择位
000	128T
001	128T
010	64T
011	32T
100	16T
101	8T
110	4T
111	2T

注：FCPU的初始状态由配置字决定。

例：调整CPU时钟分频

```
TASK_FCPU:
        BANKBSR     0X0E
        MOVIA       0x55
        MOVAR       CALLOCK         ;// CALLOCK写入0x55
        MOVIA       0xAA
        MOVAR       CALLOCK         ;// CALLOCK写入0xAA
        MOVIA       0x00
        MOVAR       FCPU            ;// FCPU写入0x00
        ...
```

4.5 IRCCAL 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IRCCAL	-	IRCCAL[6:0]						
读/写	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	x	x	x	x	x	x	x

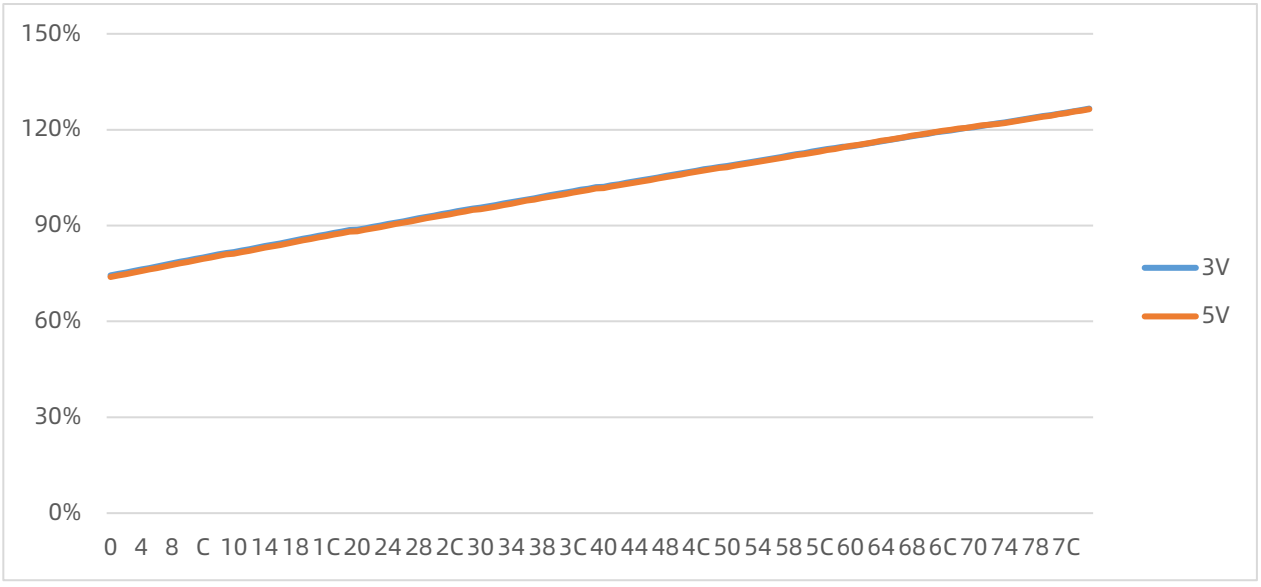
24MHz 高频内部 RC 振荡频率校准寄存器。复位后，初值为出厂校准值。

内置的双高频 RC 振荡器在芯片出厂后，频率将校准到 16MHz 和 24MHz，IRCCAH 和 IRCCAL 复位初值即为出厂校准值，但程序中可以通过特殊的写入流程来微调此频率以满足特定应用需求。

例： 调整 24MHz 高频内部 IRC 频率

```
TASK_IRCCAL:
        BANKBSR     0X0E
        MOVIA       0x55
        MOVAR       CALLOCK          ;// CALLOCK写入0x55
        MOVIA       0xAA
        MOVAR       CALLOCK          ;// CALLOCK写入0xAA
        MOVIA       0x5A
        MOVAR       IRCCAL           ;// IRCCAL写入0x5A
        ...
        ;//若需继续在IRCCAH寄存器内写入其他值需要重复以上所有步骤
```

24MHZ HIRC 调整频率趋势图



注：趋势仅供参考。

4.6 IRCCAH 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IRCCAH	-	IRCCAH[6:0]						
读/写	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	x	x	x	x	x	x	x

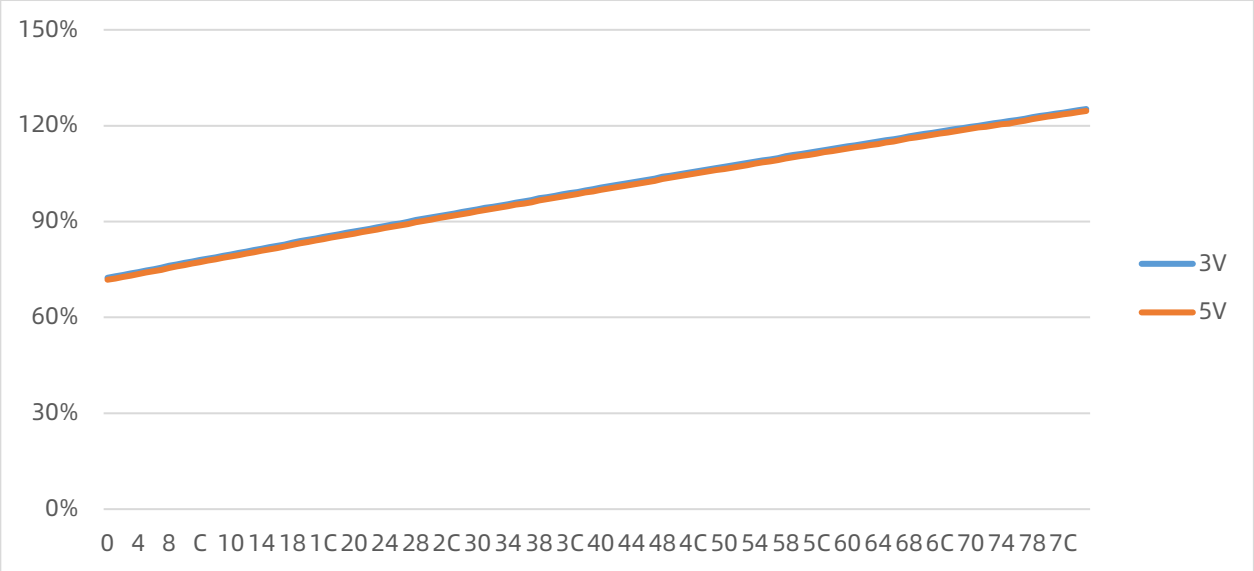
16MHz 高频内部 RC 振荡频率校准寄存器。复位后，初值为出厂校准值。

内置的双高频 RC 振荡器在芯片出厂后，频率将校准到 16MHz 和 24MHz，IRCCAH 和 IRCCAL 复位初值即为出厂校准值，但程序中可以通过特殊的写入流程来微调此频率以满足特定应用需求。

例： 调整 16MHz 高频内部 IRC 频率

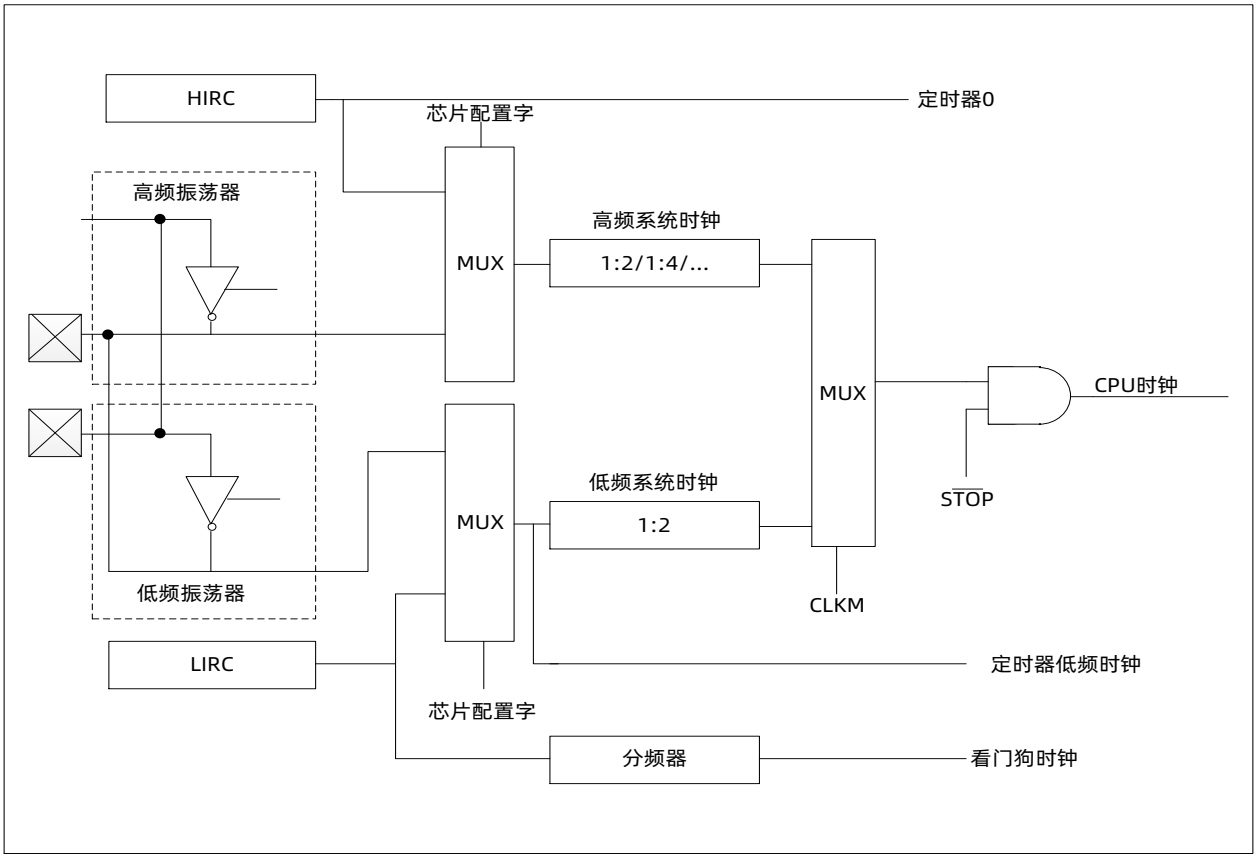
```
TASK_IRCCAH:
        BANKBSR     0X0E
        MOVIA       0x55
        MOVAR       CALLOCK          ;// CALLOCK写入0x55
        MOVIA       0xAA
        MOVAR       CALLOCK          ;// CALLOCK写入0xAA
        MOVIA       0x5A
        MOVAR       IRCCAH           ;// IRCCAH写入0x5A
        ...
;//若需继续在IRCCAH寄存器内写入其他值需要重复以上所有步骤
```

HIRC 调整频率趋势图



注：趋势仅供参考。

4.7 系统时钟结构框图



	高速运行模式 (CLKM = 0)	低速运行模式 (CLKM = 1)	休眠模式 (STOP = 1)
高频振荡器	运行	由 STPH 决定	由 STPH 决定
低频振荡器	运行	运行	由 STPL 决定
WDT	配置字决定	由配置字决定	由配置字决定
TC0/TC1	TXEN	若选择高速时钟， 需 STPH=0	高速时钟源&STPH=0 低速时钟源&STPL=0

4.8 系统时钟高低频切换

高频振荡器稳定计数器：64 Clocks（内部 IRC 模式）/1024 Clocks（外部高频振荡器模式）
低频振荡器稳定计数器：16 Clocks（内部 RC 模式）/1024 Clocks（外部低频振荡器模式）
高低频切换时间：
高频切低频：1 个低频时钟周期+1 个高频时钟周期
低频切高频（STBH = 0）：1 个低频时钟周期+高频振荡器起振时间+高频振荡器稳定时间
低频切高频（STBH = 1）：1 个低频时钟周期+1 个高频时钟周期
唤醒时间：
CLKM = 0&STBH = 0，唤醒时间 = 高频振荡器起振时间+高频振荡器稳定时间
CLKM = 0&STBH = 1，唤醒时间 = 高频振荡器稳定时间
CLKM = 1&STBL = 0，唤醒时间 = 低频振荡器起振时间+低频振荡器稳定时间
CLKM = 1&STBL = 1，唤醒时间 = 低频振荡器稳定时间

5 中断

5.1 概述

M9F6820可通过配置字选择位选择单优先级还是多优先级，单优先级是高优先级，所有中断默认都是高优先级。多优先级有两种优先级，高优先级和低优先级，通过INTPn寄存器进行控制。同级及高级优先级中断响应时不允许低级优先级中断嵌套，低级优先级中断允许高级优先级中断嵌套响应。

响应低优先级中断时，GIEL 会自动清零，同时将 GIEH/GIEL 状态压栈，执行中断时由于 GIEL 已被清零不允许响应同级中断，若 GIEH 的值是 1，则允许高优先级的中断嵌套；响应高优先级时，GIEH 和 GIEL 会被自动清零，同时将 GIEH/GIEH 状态压栈。

所有中断信号都可以唤醒 CPU（在不使能全局中断的情况下也可以唤醒）。

注：使用外部中断 INT0、INT1 要注意，当中断触发和进休眠的操作（即 STOP 从 0 到 1）同时发生时，会导致外部中断 INT0、INT1 无效且无法唤醒休眠；
解决方法：如需使用外部中断尽量使用 IO 变化中断，如果必须使用外部中断 INT0、INT1，必须要开启 WDT 作为唤醒源。

5.2 中断向量分配

高优先级中断：0008h
低优先级中断：0018h

5.3 OPTION 配置寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION	GIEH	GIEL	-	-	MINT1[1:0]		MINT0[1:0]	
读/写	R/W	R/W	-	-	R/W	R/W	R/W	R/W
复位后	0	0	-	-	0	0	0	0

Bit 7 **GIEH：**全局中断高优先级中断控制位

0 = 屏蔽高优先级中断

1 = 使能高优先级中断，中断响应后自动清零

Bit 6 **GIEL：**全局中断低优先级中断控制位

0 = 屏蔽低优先级中断

1 = 使能低优先级中断，中断响应后自动清零

Bit [3:2] **MINT1[1:0]**： INT1中断模式选择

MINT1[1:0]	INT1 中断模式选择
00	上升沿中断
01	下降沿中断
1x	变化中断

Bit [1:0] **MINT0[1:0]**：INT0中断模式选择

MINT0[1:0]	INT0 中断模式选择
00	上升沿中断
01	下降沿中断
1x	变化中断

注：同时使能 GIEL 和 GIEH，高优先中断可嵌套低优先级中断，但同级中断不可嵌套。

5.4 IO 端口变化中断使能寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOAICR	IOAICR7	IOAICR6	IOAICR5	IOAICR4	IOAICR3	IOAICR2	IOAICR1	IOAICR0
IOBICR	IOBICR7	IOBICR6	IOBICR5	IOBICR4	IOBICR3	IOBICR2	IOBICR1	IOBICR0
IOCICR							IOCICR1	IOCICR0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **IOxICRy:** IOX端口变化中断使能（x = A/B，y = 0-7，x = C，y = 0-1）

0 = 屏蔽IOxy口电平变化中断

1 = 使能IOxy口电平变化中断

5.5 INTCR0 中断控制寄存器 0

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCR0	RX0IE	TX0IE	I2CIE	CMP0IE	TC2GIE	TC2IE	TC1IE	TC0IE
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **RX0IE：** USART0接收中断使能位
0 = 屏蔽UART0接收中断
1 = 使能UART0接收中断

Bit 6 **TX0IE：** USART0发送中断使能位
0 = 屏蔽UART0发送中断
1 = 使能UART0发送中断

Bit 5 **I2CIE：** I2C中断使能位
0 = 屏蔽I2C中断
1 = 使能I2C中断

Bit 4 **CMP0IE：** 比较器中断使能位
0 = 屏蔽CMP中断
1 = 使能CMP中断

Bit 3 **TC2GIE：** TC2门控中断使能位
0 = 屏蔽TC2门控中断
1 = 使能TC2门控中断

Bit 2 **TC2IE：** TC2溢出中断使能位
0 = 屏蔽TC2溢出中断
1 = 使能TC2溢出中断

Bit 1 **TC1IE：** TC1溢出中断使能位
0 = 屏蔽TC1溢出中断
1 = 使能TC1溢出中断

Bit 0 **TC0IE：** TC0溢出中断使能位
0 = 屏蔽TC0溢出中断
1 = 使能TC0溢出中断

5.6 INTF0 中断标志寄存器 0

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTF0	RX0IF	TX0IF	I2CIF	CMP0IF	TC2GIF	TC2IF	TC1IF	TC0IF
读/写	R	R	R	R/W	R/W	R/W	R/W	R/W
复位后	0	1	0	0	0	0	0	0

Bit 7 **RX0IF：** USART0接收中断标志
0 = 未产生UART0接收中断
1 = 产生UART0接收中断（对RX0REG进行读操作后自动清零）

Bit 6 **TX0IF：** USART0发送中断标志
0 = 未产生UART0发送中断（数据不为空，还在发送中）
1 = 产生UART0发送中断（数据为空，可发送下一个数据）

Bit5 **I2CIF：** I2C中断使能位(清除I2CCR的SI位)
0 = 未产生IIC中断
1 = 产生IIC中断

Bit 4 **CMP0IF：** CMP中断标志
0 = 未产生CMP中断
1 = 产生CMP中断

Bit 3 **TC2GIF：** TC2门控中断标志
0 = 未产生TC2门控中断
1 = 产生TC2门控中断

Bit 2 **TC2IF：** TC2溢出中断标志
0 = 未产生TC2溢出中断
1 = 产生TC2溢出中断

Bit 1 **TC1IF：** TC1溢出中断标志
0 = 未产生TC1溢出中断
1 = 产生TC1溢出中断

Bit 0 **TC0IF：** TC0溢出中断标志
0 = 未产生TC0溢出中断
1 = 产生TC0溢出中断

注：除 USART 之外的所有中断标志位需软件清零。

5.7 INTP0 中断优先级寄存器 0

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTP0	RX0IP	TX0IP	I2CIP	CMP0IP	TC2GIP	TC2IP	TC1IP	TC0IP
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **RX0IP：** UART0接收中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 6 **TX0IP：** UART0发送中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 5 **I2CIP：** I2C中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 4 **CMP0IP：** CMP中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 3 **TC2GIP：** TC2门控中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 2 **TC2IP：** TC2溢出中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 0 **TC1IP：** TC1溢出中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 0 **TC0IP：** TC0溢出中断优先级控制位
0 = 高优先级
1 = 低优先级

5.8 INTCR1 中断控制寄存器 1

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCR1	INT1IE	INT0IE	TKIE	ADIE	-	IOCCHIE	IOBCHIE	IOACHIE
读/写	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
复位后	0	0	0	0	-	0	0	0

Bit 7 **INT1IE：** 外部端口中断1使能位
0 = 屏蔽外部端口中断1
1 = 使能外部端口中断1

Bit 6 **INT0IE：** 外部端口中断0使能位
0 = 屏蔽外部端口中断0
1 = 使能外部端口中断0

Bit 5 **TKIE：** 触摸中断使能位
0 = 屏蔽触摸中断
1 = 使能触摸中断

Bit 4 **ADIE：** AD转换中断使能位
0 = 屏蔽AD转换中断
1 = 使能AD转换中断

Bit 2 **IOCCHIE：** 端口C变化中断使能位
0 = 屏蔽端口C变化中断
1 = 使能端口C变化中断

Bit 1 **IOBCHIE：** 端口B变化中断使能位
0 = 屏蔽端口B变化中断
1 = 使能端口B变化中断

Bit 0 **IOACHIE：** 端口A变化中断使能位
0 = 屏蔽端口A变化中断
1 = 使能端口A变化中断

5.9 INTF1 中断标志寄存器 1

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTF1	INT1IF	INT0IF	TKIF	ADIF	-	IOCCHIF	IOBCHIF	IOACHIF
读/写	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
复位后	0	0	0	0	-	0	0	0

Bit 7 **INT1IF：** 外部端口中断1标志
0 = 未产生外部端口中断1
1 = 产生外部端口中断1

Bit 6 **INT0IF：** 外部端口中断0标志
0 = 未产生外部端口中断0
1 = 产生外部端口中断0

Bit 5 **TKIF：** TK中断标志
0 = 未产生触摸中断
1 = 产生触摸中断

Bit 4 **ADIF：AD**中断标志
0 = 未产生AD转换中断
1 = 产生AD转换中断

Bit 2 **IOCCHIF：** 端口C变化中断标志
0 = 未产生端口C变化中断
1 = 产生端口C变化中断

Bit 1 **IOBCHIF：** 端口B变化中断标志
0 = 未产生端口B变化中断
1 = 产生端口B变化中断

Bit 0 **IOACHIF：** 端口A变化中断标志
0 = 未产生端口A变化中断
1 = 产生端口A变化中断

注：所有中断标志位需软件清零。

5.10 INTP1 中断优先级寄存器 1

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTP1	INT1IP	INT0IP	TKIP	ADIP	-	IOCCHIP	IOBCHIP	IOACHIP
读/写	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
复位后	0	0	0	0	-	0	0	0

Bit 7 **INT1IP：** 外部端口中断1优先级控制位
0 = 高优先级
1 = 低优先级

Bit 6 **INT0IP：** 外部端口中断0优先级控制位
0 = 高优先级
1 = 低优先级

Bit 5 **TKIP：** 触摸中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 4 **ADIP：AD**中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 2 **IOCCHIP：** 端口C变化中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 1 **IOBCHIP：** 端口B变化中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 0 **IOACHIP：** 端口A变化中断优先级控制位
0 = 高优先级
1 = 低优先级

5.11 INTCR2 中断控制寄存器 2

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCR2	-	-	-	-	-	RX1IE	TX1IE	SPIIE
读/写	-	-	-	-	-	R/W	R/W	R/W
复位后	-	-	-	-	-	0	0	0

Bit 2 **RX1IE：** USART1接收中断使能位
0 = 屏蔽UART1接收中断
1 = 使能UART1接收中断

Bit 1 **TX1IE：** USART1发送中断使能位
0 = 屏蔽UART1发送中断
1 = 使能UART1发送中断

Bit 0 **SPIIE：** SPI中断使能位
0 = 屏蔽SPI中断
1 = 使能SPI中断

5.12 INTF2 中断标志寄存器 2

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTF2	-	-	-	-	-	RX1IF	TX1IF	SPIF2
读/写	-	-	-	-	-	R	R	R
复位后	-	-	-	-	-	0	1	0

Bit 2 **RX1IF：** USART1接收中断标志
0 = 未产生UART1接收中断
1 = 产生UART1接收中断（对RX1REG进行读操作后自动清零）

Bit 1 **TX1IF：** USART1发送中断标志
0 = 未产生UART1发送中断（数据不为空，还在发送中）
1 = 产生UART1发送中断（数据为空，可发送下一个数据）

Bit 0 **SPIF2：** SPI中断标志
0 = 清除对应SPIF或MODF后自动清0
1 = 产生SPI中断

注：SPIF2 为 SPIF&MODF 或逻辑产生，只读，故清除此标志需清除对应被置 1 的 SPIF 或 MODF。

5.13 INTP2 中断优先级寄存器 2

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTP2					-	RX1IP	TX1IP	SPIP
读/写					-	R/W	R/W	R/W
复位后					-	0	0	0

Bit 2 **RX1IP：** UART1接收中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 1 **TX1IP：** UART1发送中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 0 **SPIP：** SPI中断优先级控制位
0 = 高优先级
1 = 低优先级

6 端口

6.1 数据寄存器 IOx（x = A/B/C）

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOA	IOA7	IOA6	IOA5	IOA4	IOA3	IOA2	IOA1	IOA0
IOB	IOB7	IOB6	IOB5	IOB4	IOB3	IOB2	IOB1	IOB0
IOC	-	-	-	-	-	-	IOC1	IOC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

注：如读 IOA 的值读取的是引脚电平（模拟端口设置寄存器必需关闭，默认端口是模拟使能），向 IOA 写值是写入输出寄存器。

6.2 输出锁存寄存器 IOxOR（x = A/B/C）

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOAOR	IOAOR7	IOAOR6	IOAOR5	IOAOR4	IOAOR3	IOAOR2	IOAOR1	IOAOR0
IOBOR	IOBOR7	IOBOR6	IOBOR5	IOBOR4	IOBOR3	IOBOR2	IOBOR1	IOBOR0
IOCOR	-	-	-	-	-	-	IOCOR1	IOCOR0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	x	x	x	x	x	x	x	x

Bit [7:0] **IOxORy：**输出锁存（x = A/B，y = 0-7； x = C，y = 0-1）

0 = 输出为0

1 = 输出为1

注：如对 IOAOR 寄存器的读操作是对 IOA 输出的读，对 IOAOR 寄存器的写操作是对 IOA 输出的写。

6.3 输出方向寄存器 OEx（x = A/B/C）

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OEA	OEA7	OEA6	OEA5	OEA4	OEA3	OEA2	OEA1	OEA0
OEB	OEB7	OEB6	OEB5	OEB4	OEB3	OEB2	OEB1	OEB0
OEC	-	-	-	-	-	-	OEC1	OEC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **OExy：** 输出/输出使能（x = A/B，y = 0-7； x = C，y = 0-1）

0 = 输入模式

1 = 输出模式

6.4 上拉控制寄存器 PUx（x = A/B/C）

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PUA	PUA7	PUA6	PUA5	PUA4	PUA3	PUA2	PUA1	PUA0
PUB	PUB7	PUB6	PUB5	PUB4	PUB3	PUB2	PUB1	PUB0
PUC	-	-	-	-	-	-	PUC1	PUC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PUxy：** 上拉使能（x = A/B，y = 0-7； x = C，y = 0-1）

0 = 上拉关闭

1 = 上拉使能

注：IIC 开漏输出，输出高需要设置上拉电阻。

6.5 下拉控制寄存器 PDx（x = A/B/C）

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PDA	PDA7	PDA6	PDA5	PDA4	PDA3	PDA2	PDA1	PDA0
PDB	PDB7	PDB6	PDB5	PDB4	PDB3	PDB2	PDB1	PDB0
PDC	-	-	-	-	-	-	PDC1	PDC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PDxy：** 下拉使能（x = A/B，y = 0-7； x = C，y = 0-1）

0 = 下拉关闭

1 = 下拉使能

注：同一 IO 口上下拉电阻同时打开时，此时端口电平接近于 VDD/2。

6.6 模拟端口设置寄存器 ANSx（x = A/B/C）

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSA	ANSA7	ANSA6	ANSA5	ANSA4	ANSA3	ANSA2	ANSA1	ANSA0
ANSB	ANSB7	ANSB6	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0
ANSC	-	-	-	-	-	-	ANSC1	ANSC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	1	1	1	1	1	1	1	1

Bit [7:0] **ANSxy：** 端口类型设置（x = A/B，y = 0-7； x = C，y = 0-1）

0 = 数字端口

1 = 模拟端口功能（数字输入功能被屏蔽，即端口状态MCU无法读取）

注：模拟端口模式，仅数字输入功能被屏蔽，但可输出。

6.7 驱动电流选择寄存器 IOxODS1/ IOxODS0（x = A/B/C）

驱动电流选择寄存器 1

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOAODS1	IOAODS17	IOAODS16	IOAODS15	IOAODS14	IOAODS13	IOAODS12	IOAODS11	IOAODS10
IOBODS1	IOBODS17	IOBODS16	IOBODS15	IOBODS14	IOBODS13	IOBODS12	IOBODS11	IOBODS10
IOCODS1	-	-	-	-	-	-	IOCODS11	IOCODS10
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

驱动电流选择寄存器 0

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOAODS0	IOAODS07	IOAODS06	IOAODS05	IOAODS04	IOAODS03	IOAODS02	IOAODS01	IOAODS00
IOBODS0	IOBODS07	IOBODS06	IOBODS05	IOBODS04	IOBODS03	IOBODS02	IOBODS01	IOBODS00
IOCODS0	-	-	-	-	-	-	IOCODS01	IOCODS00
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **IOxODS1y，IOxODS0y：** 驱动能力选择（x = A/B，y = 0-7； x = C，y = 0-1）

IOxODS1y，IOxODS0y	IOxy 驱动电流选择
00	最大(I_{OL1}\ I_{OH1})
01	次大(I_{OL2}\ I_{OH2})
10	次小(I_{OL3}\ I_{OH3})
11	最小(I_{OL4}\ I_{OH4})

6.8 翻转电平设置寄存器 IOxIPS/FLIPCR（x = A/B/C）

翻转控制寄存器 IOxIPS

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOAIPS	IOAIPS7	IOAIPS6	IOAIPS5	IOAIPS4	IOAIPS3	IOAIPS2	IOAIPS1	IOAIPS0
IOBIPS	IOBIPS7	IOBIPS6	IOBIPS5	IOBIPS4	IOBIPS3	IOBIPS2	IOBIPS1	IOBIPS0
IOCIPS	-	-	-	-	-	-	IOCIPS1	IOCIPS0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	1	1	1	1	1	1	1	1

Bit [7:0] **IOxIPSy：**翻转电平选择（x = A/B，y = 0-7； x = C，y = 0-1）

0 = SMT

1 = 1.5 V 或 1/2VDD（由FLIPCR寄存器控制）

注：IOCIPS[7:2] 复位值为 000000b。

翻转控制寄存器 FLIPCR

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FLIPCR	-	-	-	-	-	-	-	FLIPCR0
读/写	-	-	-	-	-	-	-	R/W
复位后	-	-	-	-	-	-	-	0

Bit 0 **FLIPCR0：**IO 口翻转控制，在 IOxIPSy 为 1 时有效（x = A/B，y = 0-7； x = C，y = 0-1）

0 = 翻转电平为 1/2VDD

1 = 翻转电平为 1.5V

6.9 数字功能可选端口控制模块

引脚图中未标明的数字功能中，USART、I2C 和 PWM1，T2G 可以映射到任意端口，PWM0 只能整体选择端口。

6.9.1 端口复用控制寄存器表

寄存器名称	复用功能	复用说明
MPTX0	USART TX	全端口映射
MPRX0	USART RX	全端口映射
MPTX1	USART TX	全端口映射
MPRX1	USART RX	全端口映射
MPSDA	I2C SDA	全端口映射
MPSCL	I2C SCL	全端口映射
MPPWM10	PWM10	全端口映射
MPPWM11	PWM11	全端口映射
MPT2G	T2G	全端口映射
MPPWM0	MPPWM0x（x = 0-3）	按端口整体选择

6.9.2 全端口映射控制

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	-	-	PORTSEL[1:0]		-	PINSEL[2:0]		
读/写	-	-	R/W	R/W	-	R/W	R/W	R/W
复位后	-	-	0	0	-	0	0	0

Bit [5:4] **PORTSEL**[1:0]：映射端口选择位

PORTSEL[1:0]	IO 端口组选择
00	IOA
01	IOB
10	IOC

Bit [2:0] **PINSEL**[2:0]：映射管脚号选择位(x = A、B、C)

PINSEL[2:0]	IO 端口位选择
000	IOx0
001	IOx1
010	IOx2
011	IOx3
100	IOx4
101	IOx5
110	IOx6
111	IOx7

如需将USART TX0端口映射到IOB3，则可如下操作：

```
MPTX0 = 0x13;    //TX0将映射到IOB3口所在的管脚。
```

6.9.3 PWM0 复用控制

PWM0 复用以四路为一组，下面的引脚为四路 PWM00 的引脚，其他三路按顺序向下排。比如 MPPWM0 设置为 00h，PWM00 从 IOA0 输出 PWM01 从 IOA1 输出 PWM02 从 IOA2 输出 PWM03 从 IOA3 输出，当 MPPWM0 设置为 20h 时，由于 IOC 口只到 IOC1 因此 PWM02、PWM03 不输出。

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MPPWM0	-	-	PORTSEL[1:0]		-	PINSEL2	-	-
读/写	-	-	R/W	R/W	-	R/W	-	-
复位后	-	-	0	0	-	0	-	-

Bit [5:4] **PORTSEL[1:0]：** PWM0 输出端口 IOx 选择(x = A、B、C)

PORTSEL[1:0]	IO 口选择
00	IOA
01	IOB
10	IOC

Bit 2 **PINSEL2：** PWM0y 输出端口高低位选择（y = 0-3）

0 = PWM00/01/02/03 输出端口依次为 IOx0/1/2/3

1 = PWM00/01/02/03 输出端口依次为 IOx4/5/6/7

注：由于 IOC 端口无 IOC2&IOC3 口，故当 PWM0 选择 IOC 高位时，PWM02&PWM03 将无法输出。

7 定时器 0(TC0/PWM0)

7.1 概述

TC0 为带有可设置 1-128 预分频器及 1- 8 后分频器和周期寄存器的 8 位/16 位定时计数器，具有休眠状态下唤醒功能。

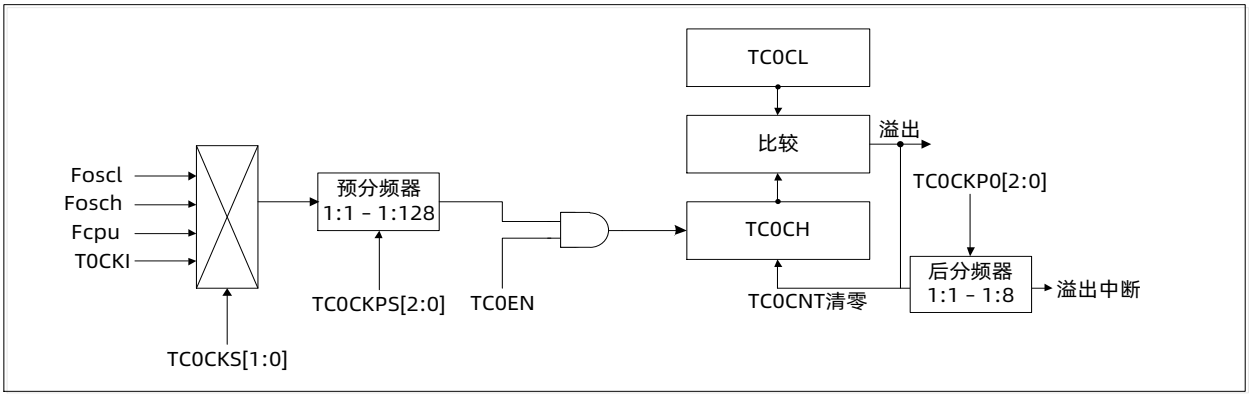
在 8 位模式下，TC0CL 作为 TC0 的周期寄存器器，TC0 使能后，TC0CH 递加，当 TC0CH 与 TC0CL 数值相等，TC0 溢出，将 TC0CH 清零重新开始计数，同时将中断标志位 TC0IF 置 1。

在 16 位模式下，{TC0CH:TC0CL}作为 16 位的计数器，TC0 使能后，16 位计数器递加，当计数值等于 0xFFFF 时，16 位计数器将清零重新开始计数，同时将中断标志位 TC0IF 置 1。

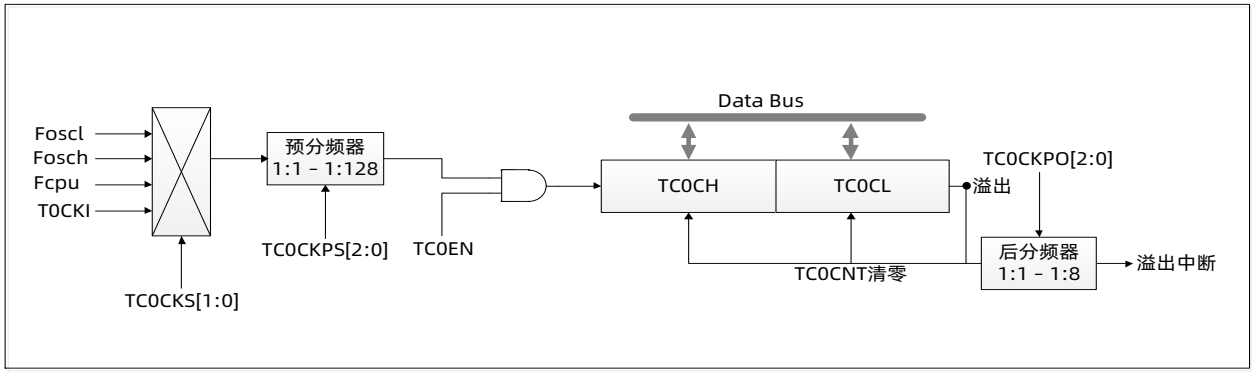
- 可选择时钟源：高频系统时钟 F_{OSCH}、低频系统时钟 F_{OSCL}、指令时钟 F_{CPU} 或外部时钟 T0CKI
- 可选择 8 位模式和 16 位模式，使能 PWM0 时，只能在 8 位模式
- 预分频比多级可选，最大可选择 1:128
- 后分频比多级可选，最大可选择 1:8
- 溢出中断功能
- 溢出中断唤醒功能（当输入频率选择 F_{OSCH} 或 F_{OSCL} 时，若所选择的时钟源振荡器一直工作，此时 TC0 在休眠状态下依然工作，溢出中断可唤醒 CPU）

TC0 框图

8 位模式



16 位模式



7.2 T0CR 控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T0CR	TC0EN	TC0MOD	TC0OSCS	TC0CKS[1:0]		TC0CKPS[2:0]		
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **TC0EN**：TC0模块使能位

0 = 关闭TC0

1 = 使能TC0

Bit 6 **TC0MOD：**TC0模式选择位

0 = 8位模式

1 = 16位模式

Bit 5 **TC0OSCS：**TC0内部高频时钟选择（系统时钟选择外部晶振时，此控制位无效）

0 = 24MHz（使用触摸功能，禁止选择）

1 = 16MHz

Bit [4:3] **TC0CKS[1:0]**：TC0时钟源选择

TC0CKS[1:0]	TC0 时钟源选择
00	F_{OSCL}(低频系统时钟)
01	F_{OSCH}(高频系统时钟)
10	F_{CPU}
11	T0CKI

Bit [2:0] **TC0CKPS[2:0]**：TC0预分频比选择

TC0CKPS[2:0]	TC0 预分频比
000	1:1
001	1:2
010	1:4
011	1:8
100	1:16
101	1:32
110	1:64
111	1:128

7.3 T0CR2 TC0 控制寄存器 2

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T0CR2	-	-	-	-	-	TC0CKPO[2:0]		
读/写	-	-	-	-	-	R/W	R/W	R/W
复位后	-	-	-	-	-	0	0	0

Bit [2:0] **TC0CKPO[2:0]**：TC0 后分频，在原来预分频的基础上再分频

TC0CKPO[2:0]	TC0 后分频比
000	1:1
001	1:2
010	1:3
011	1:4
100	1:5
101	1:6
110	1:7
111	1:8

7.4 TC0CL TC0 计数器低位

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC0CL	TC0CL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	x	x	x	x	x	x	x	x

7.5 TC0CH TC0 计数器高位

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC0CH	TC0CH[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	x	x	x	x	x	x	x	x

7.6 PWM0xCR PWM 控制寄存器（x = 0,1,2,3）

PWM0x 可以复用到很多 IO 口，其复用以四路为一组，具体见 6.9 小节。

PWM0xCR 控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM00CR	PWM00EN	PWM00OE	PWM00M	-	-	-	-	-
PWM01CR	PWM01EN	PWM01OE	PWM01M	-	-	-	-	-
PWM02CR	PWM02EN	PWM02OE	PWM02M	-	-	-	-	-
PWM03CR	PWM03EN	PWM03OE	PWM03M	-	-	-	-	-
读/写	R/W	R/W	R/W	-	-	-	-	-
复位后	0	0	0	-	-	-	-	-

Bit 7 **PWM0xEN**：PWM0x使能位
0 = 关闭PWM
1 = 使能PWM（TC0必须选择8位模式）

Bit 6 **PWM0xOE**：PWM0x输出控制
0 = PWM信号不从管脚输出，管脚用做IO
1 = PWM信号从管脚输出

Bit 5 **PWM0xM**：PWM0x输出极性控制
0 = PWM高电平有效
1 = PWM低电平有效

7.7 PWM0xD PWM 占空比寄存器（x = 0,1,2,3）

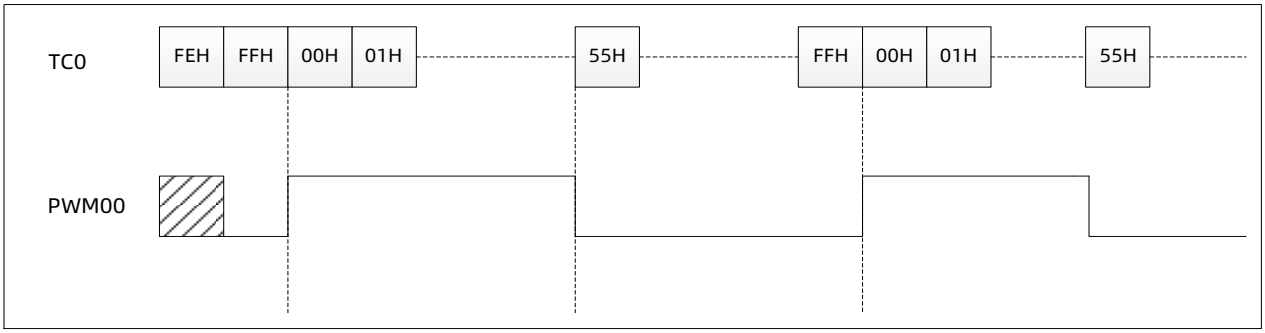
PWM0xD 占空比寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM00D	PWM00D[7:0]							
PWM01D	PWM01D[7:0]							
PWM02D	PWM02D[7:0]							
PWM03D	PWM03D[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PWM0xD：**PWM0x占空比控制寄存器

7.8 PWM0x 波形实例（x = 0,1,2,3）

例：PWM00CR = 11100000B，PWM00D = 55h，TC0CL = FFh。



8 定时器 1（TC1/PWM1）

8.1 概述

TC1 为带有可设置 1-128 预分频器及周期寄存器的 16 位定时计数器，具有休眠状态下唤醒功能。

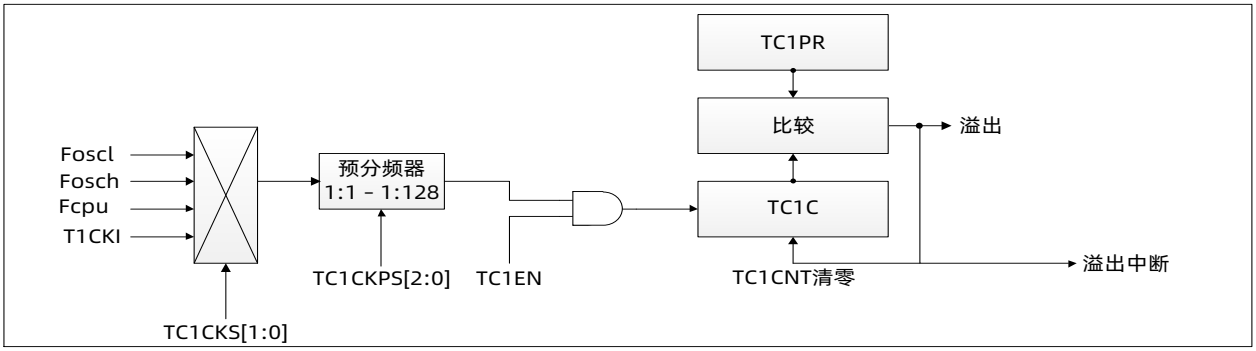
向上计数模式，TC1 使能后，TC1C 递增，当 TC1C 计数值与 TC1PR 相等时，TC1 溢出，将 TC1CH 清零重新开始计数，同时将中断标志位 T1IF 置 1。

向下计数模式，TC1 使能后，TC1C 递减，当 TC1C 计数值为 0 时，TC1 溢出，TC1C 将重新载入 TC1PR 的值开始递减计数，同时将中断标志位 T1IF 置 1。

中间对齐方式，TC1 使能后，TC1C 递增计数，当 TC1C 递增到 TC1PR 时，产生溢出中断，然后 TC1C 开始递减，递减到 0 时，再次产生溢出中断，同时开始递增。

- 可选择时钟源：高频系统时钟 F_{OSCH}、低频系统时钟 F_{OSCL}、指令时钟 F_{CPU} 或外部时钟 T1CKI
- 16 位周期寄存器
- 预分频比多级可选，最大可选择 1:128
- 溢出中断功能
- 溢出中断唤醒功能（当输入频率选择 F_{OSCH} 或 F_{OSCL} 时，若所选择的时钟源振荡器一直工作，此时 TC1 在休眠状态下依然工作，溢出中断可唤醒 CPU。）

TC1 框图



8.2 T1CR 控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CR	TC1EN	TC1MOD[1:0]		TC1CKS[1:0]		TC1CKPS[2:0]		
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **TC1EN**：TC1模块使能位

0 = 关闭TC1

1 = 使能TC1

Bit [6:5] **TC1MOD[1:0]**：TC1模式选择位

TC1MOD[1:0]	TC1 模式选择
00	递增模式
01	递减模式
1x	中心对齐模式

Bit [4:3] **TC1CKS[1:0]**：TC1时钟源选择

TC1CKS[1:0]	TC1 时钟源选择
00	F_{OSCL}(低频系统时钟)
01	F_{OSCH}(高频系统时钟)
10	F_{CPU}
11	T1CKI

Bit [2:0] **TC1CKPS[2:0]**：TC1预分频比选择

TC1CKPS[2:0]	TC1 预分频比
000	1:1
001	1:2
010	1:4
011	1:8
100	1:16
101	1:32
110	1:64
111	1:128

8.3 TC1CL TC1 计数器低位

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC1CL	TC1CL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	x	x	x	x	x	x	x	x

8.4 TC1CH TC1 计数器高位

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC1CH	TC1CH[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	x	x	x	x	x	x	x	x

8.5 TC1PRL TC1 周期寄存器低位

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC1PRL	TC1PRL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	1	1	1	1	1	1	1	1

8.6 TC1PRH TC1 周期寄存器高位

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC1PRH	TC1PRH[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	1	1	1	1	1	1	1	1

8.7 PWM1CR 控制寄存器

PWM1 输出端口可以复用到任意 IO 端口，详见章节 6.9。

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1CR	PWM10EN	PWM10OE	PWM10M	PWM11EN	PWM11OE	LEDFLAG	LEDBSY	LED
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **PWM10EN**：PWM10模块使能位
0 = 关闭PWM10
1 = 使能PWM10

Bit 6 **PWM10OE**：PWM10波形输出使能位
0 = 端口用作IO
1 = 端口输出PWM10波形

Bit 5 **PWM10M**：PWM10输出模式（LED=1）
0 = 正常数据（输出码型先高后低）
1 = 数据反向（输出码型先低后高）
PWM10M：PWM10输出模式（LED=0）
0 = 高电平有效
1 = 低电平有效

Bit 4 **PWM11EN**：PWM11模块使能位（LED=0）
0 = 关闭PWM11
1 = 使能PWM11

Bit 3 **PWM11OE**：PWM11波形输出使能位（LED=0）
0 = 端口用作IO
1 = 端口输出PWM11波形

Bit 2 **LEDFLAG**：LED发送标志（LED=1）
0 = 空闲
1 = 正在发送
LEDFLAG：PWM11输出模式（LED=0）
0 = 低电平有效
1 = 高电平有效

Bit 1 **LEDBSY**：LED数据缓冲标志（LED=1）
0 = 可写入发送数据
1 = 缓冲区和数据区满，不可写入数据。
LEDBSY：PWM1模式（LED=0）
0 = 独立输出
1 = 互补输出

Bit0 **LED**：级联LED驱动
0 = 不使能
1 = 使能

8.8 PWM1xD 数据寄存器（x = 0,1）

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1xDH	PWM1xDH[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1xDL	PWM1xDL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

8.9 PWM 死区控制寄存器

当 PWM1CR 设置为互补模式时，PWM10D 作为数据寄存器，PWM11D 作为死区寄存器，PWM11DH 前死区，PWM11DL 后死区。

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM11DH	PWM11DH[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PWM11DH[7:0]**：前死区宽度设置

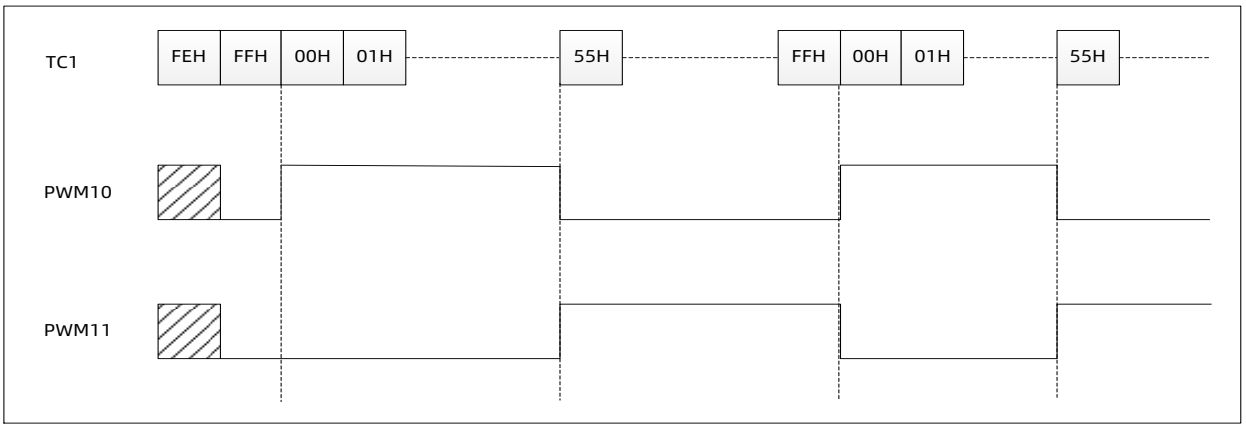
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM11DL	PWM11DL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PWM11DL[7:0]**：后死区宽度设置

8.11 PWM1 波形示例

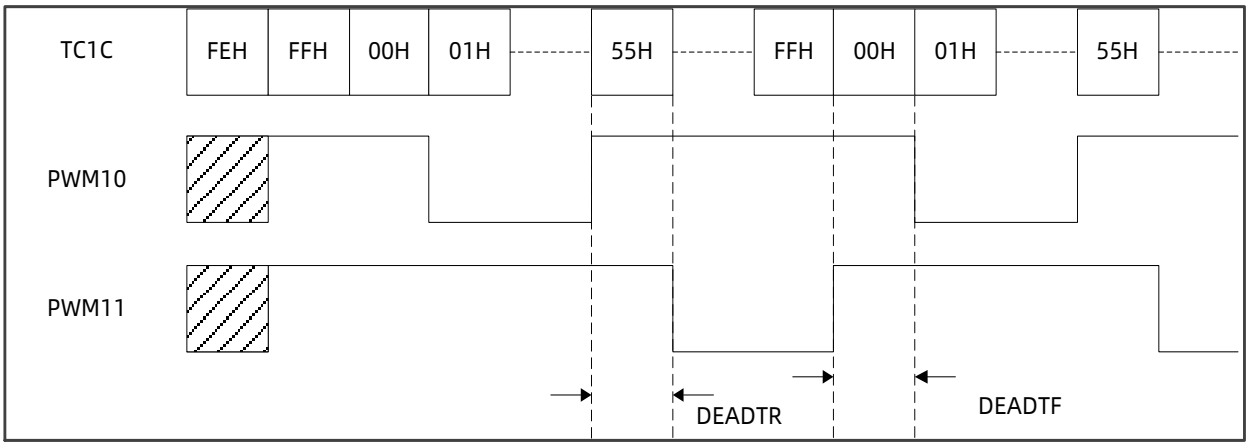
8.11.1 互补 PWM1 输出

例：PWM1CR = 11011010B，PWM10DH = 00h，PWM10DL = 55h，TC1PRL = FFh，TC1PRH = 00h
PWM11DH = 00h，PWM11DL = 00h。



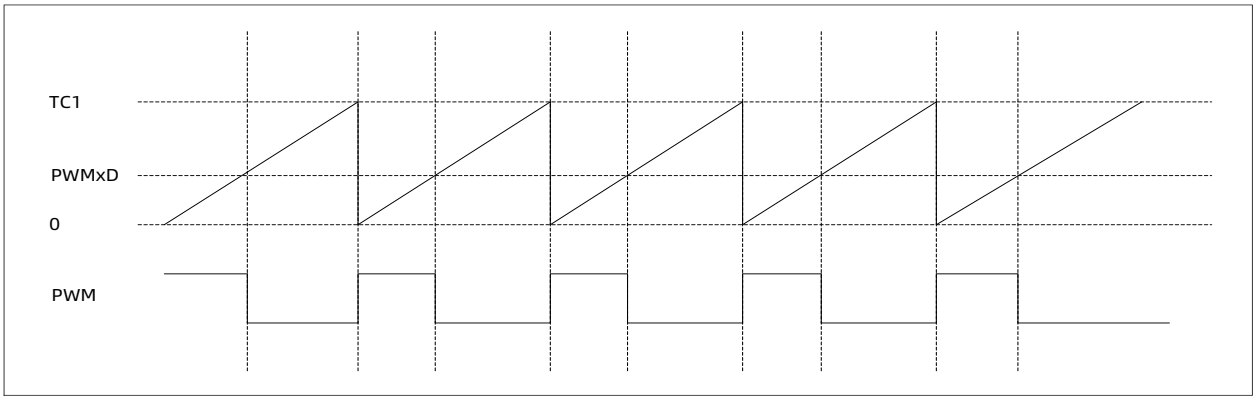
8.11.2 带死区的互补 PWM1 输出

例： PWM1CR = 11111110B，PWM10DH = 00h，PWM10DL = 55h，TC1PRL = FFh，TC1PRH = 00h，
PWM11DH = 01h，PWM11DL = 01h。

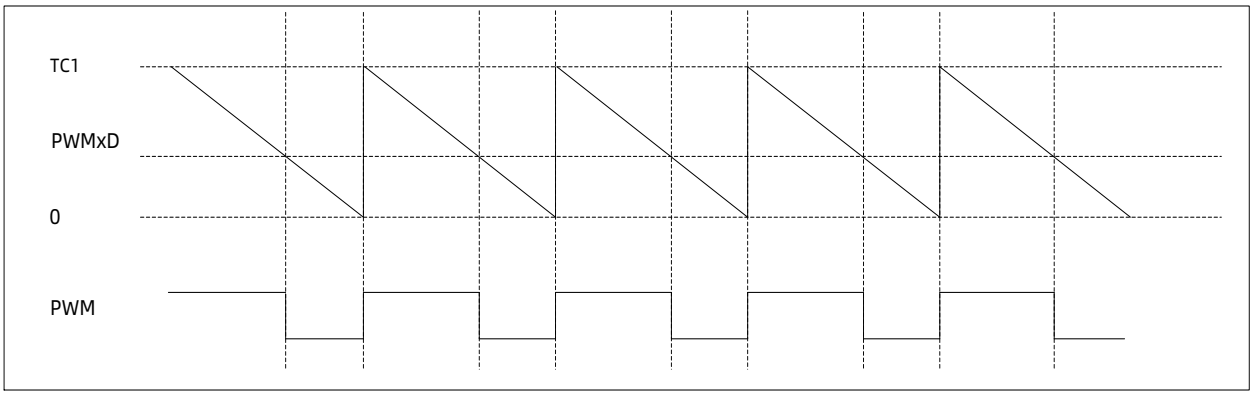


8.11.3 PWM 波形图

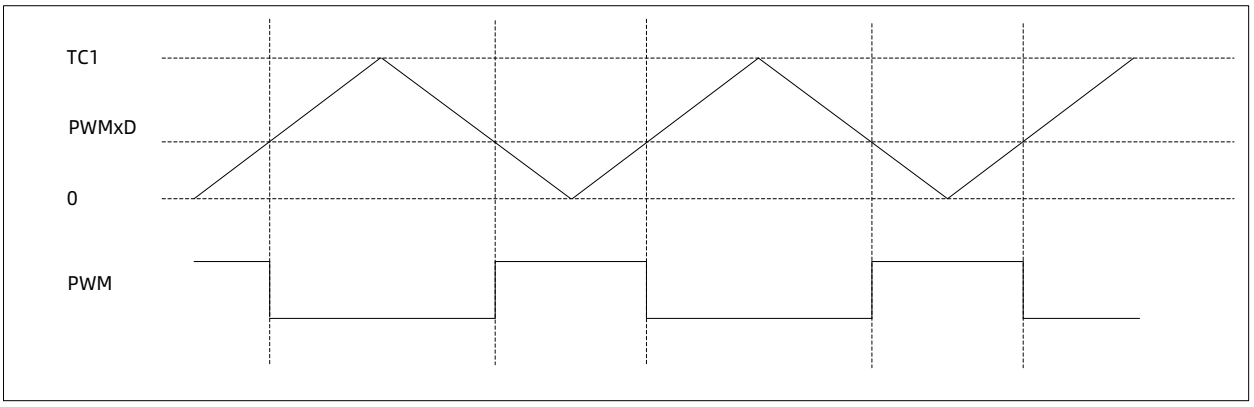
TC1 递增模式



TC1 递减模式



TC1 中心对齐模式



注：请不要在 PWM 模块使能时修改 PWM1，因为修改会立刻生效，导致当前这个周期出错。

示例代码：

```
//1、设置新周期
    TC1PRL = New_Period_L;
    TC1PRH = New_Period_H;
//2、清零计数器
    TC1CL = 0;
    TC1CH = 0;
```

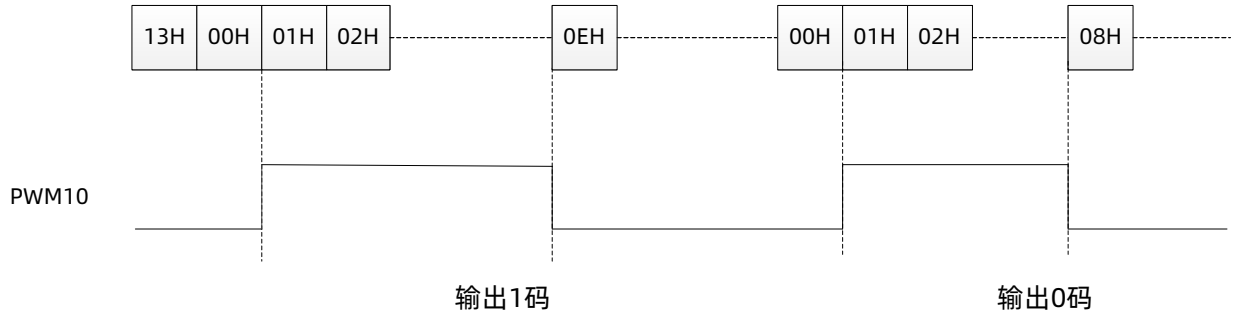
8.12 级联 LED 驱动

8.12.1 功能描述

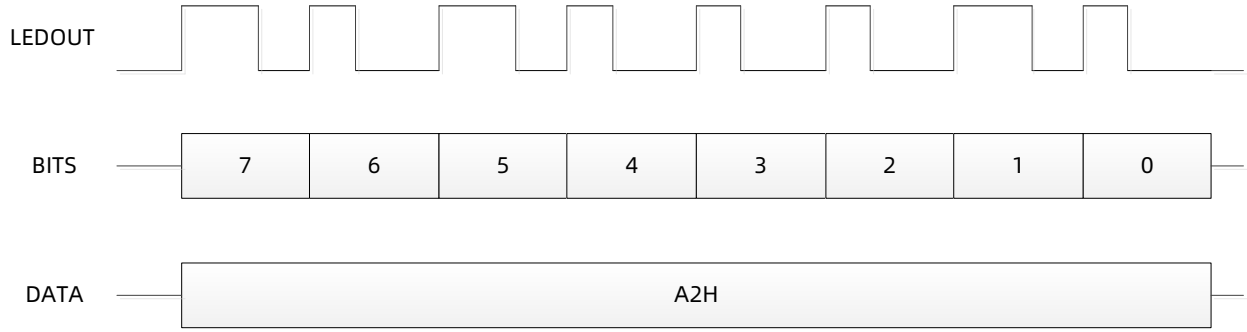
单线级联 LED 驱动模式下，位周期由 TC1PR 控制，0 码的高电平宽度由 PWM10DL 控制,1 码的高电平宽度由 PWM10DH 控制，发送数据寄存器为 PWM11DL(有一级缓存)，LEDBSY=0 时，可以写入数据；LED 使能，PWM11 自动关闭。

例：F_{OSC}=16MHz，TC1PR=13H，PWM10DL=07H，PWM10DH=0DH，PWM1CR=C1H，写入数据为 A2H。

输出 0 码和输出 1 码时序图



级联 LCD 时序图



注：关于级联 LED 驱动应用请参照 DEMO 相关例程。

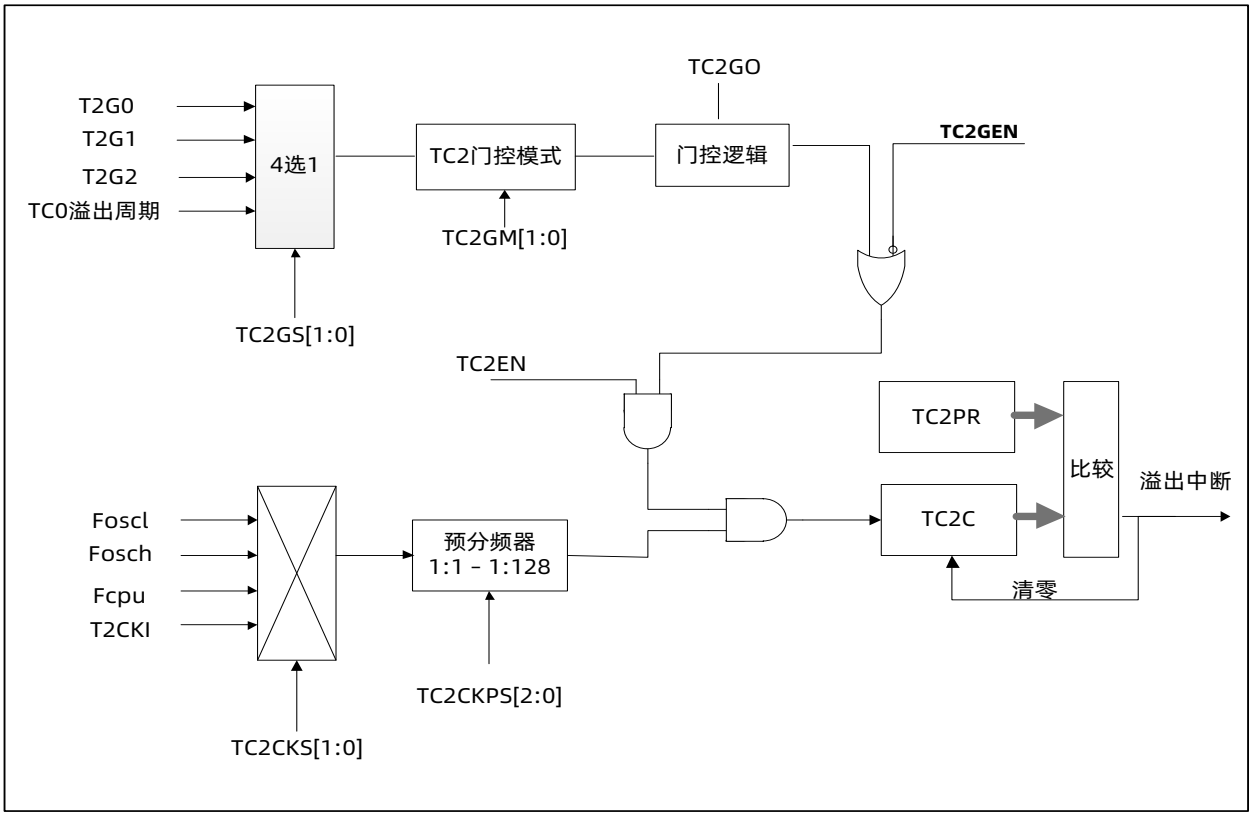
9 定时器 2（TC2）

9.1 概述

TC2 为带门控功能的可设置 1-128 预分频器及周期寄存器的 16 位定时计数器，具有休眠状态下唤醒功能。

- 可选择时钟源：高频系统时钟 F_{OSCH}、低频系统时钟 F_{OSCL}、指令时钟 F_{CPU} 或 T2CKI
- 可选择预分频比，最大 1:128
- 门控模式可选择门控源：外部门控信号 T2G0、T2G1、T2G2 或 TC0 溢出信号
- 门控和溢出中断功能
- 溢出中断唤醒功能（当输入频率选择 F_{OSCH} 或 F_{OSCL} 输出时，若所选择的时钟源振荡器一直工作，此时 TC2 在休眠状态下依然工作，溢出中断可唤醒 CPU。）

TC2 框图



9.2 T2CR 控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T2CR	TC2EN	-	-	TC2CKS[1:0]		TC2CKPS[2:0]		
读/写	R/W	-	-	R/W	R/W	R/W	R/W	R/W
复位后	0	-	-	0	0	0	0	0

Bit 7 **TC2EN：** TC2模块使能位

0 = 关闭TC2

1 = 使能TC2

Bit [4:3] **TC2CKS[1:0]：** TC2时钟源选择位

TC2CKS[1:0]	TC2 计数时钟源选择
00	F_{OSCL}(低频系统时钟)
01	F_{OSCH}(高频系统时钟)
10	F_{CPU}
11	T2CKI

Bit [2:0] **TC2CKPS[2:0]：** TC2预分频比选择

TCxCKPS[2:0]	TC2 预分频比
000	1:1
001	1:2
010	1:4
011	1:8
100	1:16
101	1:32
110	1:64
111	1:128

9.3 TC2CL 计数器低位

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC2CL	TC2CL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	x	x	x	x	x	x	x	x

9.4 TC2CH 计数器高位

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC2CH	TC2CH[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	x	x	x	x	x	x	x	x

9.5 TC2PRL 周期寄存器低位

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC2PRL	TC2PRL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	1	1	1	1	1	1	1	1

9.6 TC2PRH 周期寄存器高位

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC2PRH	TC2PRH[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	1	1	1	1	1	1	1	1

9.7 TC2GCR 门控控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC2GCR	TC2GEN	TC2GO	-	-	-	TC2GS	TC2GM[1:0]	
读/写	R/W	R/W	-	-	-	R/W	R/W	R/W
复位后	0	0	-	-	-	0	0	0

Bit 7 **TC2GEN：** TC2门控模式使能位

0 = 关闭门控功能，TC2是否计数仅受TC2GEN控制

1 = 使能门控功能

Bit 6 **TC2GO：** 启动门控控制位

0 = 完成门控计数，自动清零

1 = 启动门控

Bit 2 **TC2GS:：** TC2门控源选择位

TC2GS[1:0]	TC2 门控选择
0	T2G（任意管脚映射）
1	TC0 溢出周期（只支持上升沿到上升沿模式）

Bit [1:0] **TC2GM[1:0]：** TC2门控模式选择位（T2G下可选择）

TC2GM[1:0]	TC2 门控模式选择
00	上升沿到下降沿
01	下降沿到上升沿
10	上升沿到上升沿
11	下降沿到下降沿

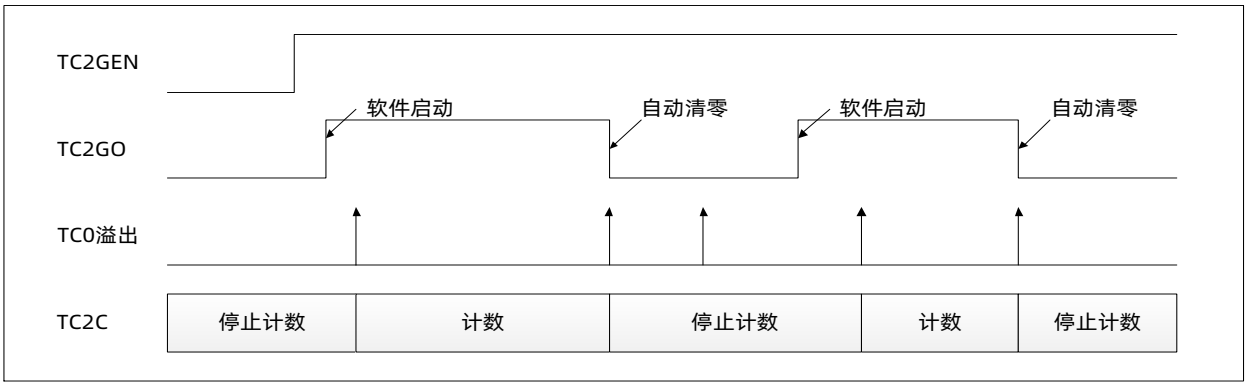
注：

（1）启动门控前需先将门控使能，不可同时置 1。

（2）选择 TC0 溢出信号作为门控源时，若 TC0 是 8 位模式，则 TC0CL 不能为 0。

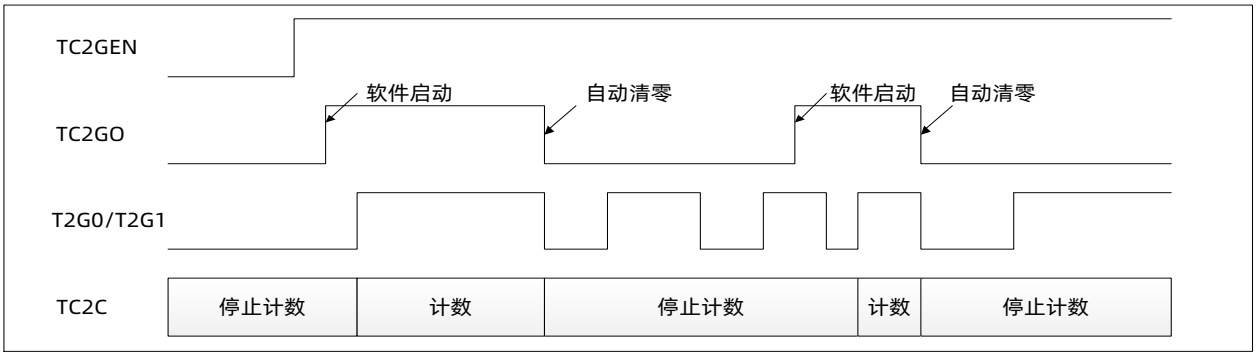
9.7.1 门控-TC0 溢出周期

上升沿到上升沿模式：启动门控计数后，门控逻辑从第一次 TC0 溢出开始计数，第二次 TC0 溢出停止计数，如下图。



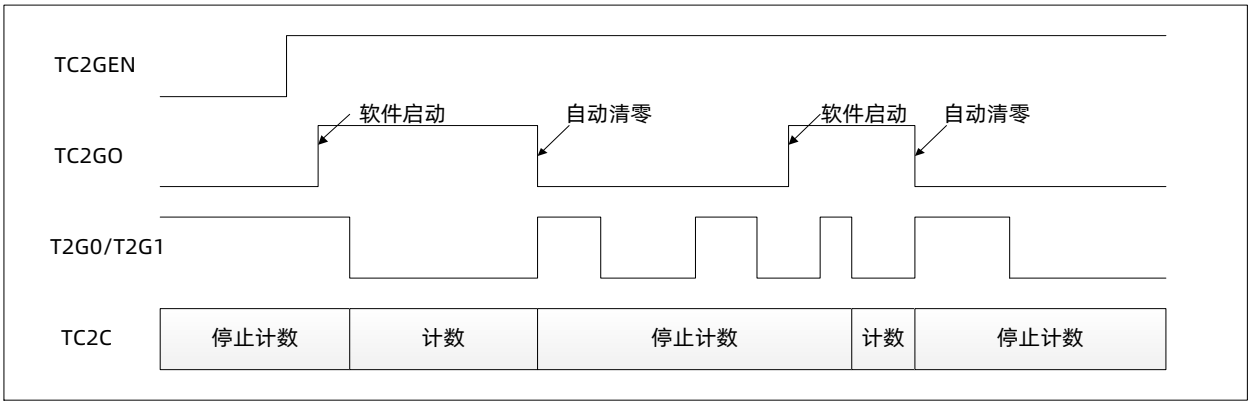
9.7.2 门控-上升沿到下降沿模式

上升沿到下降沿模式：启动门控计数后，门控逻辑从捕获到的第一个上升沿开始计数，然后捕获到下降沿停止计数，如下图。



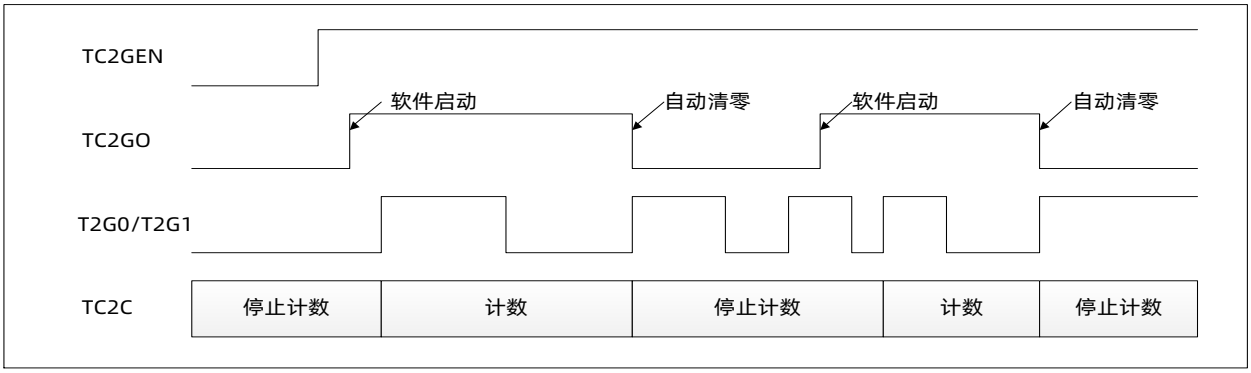
9.7.3 门控-下降沿到上升沿模式

下降沿到上升沿模式：启动门控计数后，门控逻辑从捕获到的第一个下降沿开始计数，然后捕获到上升沿停止计数，如下图。



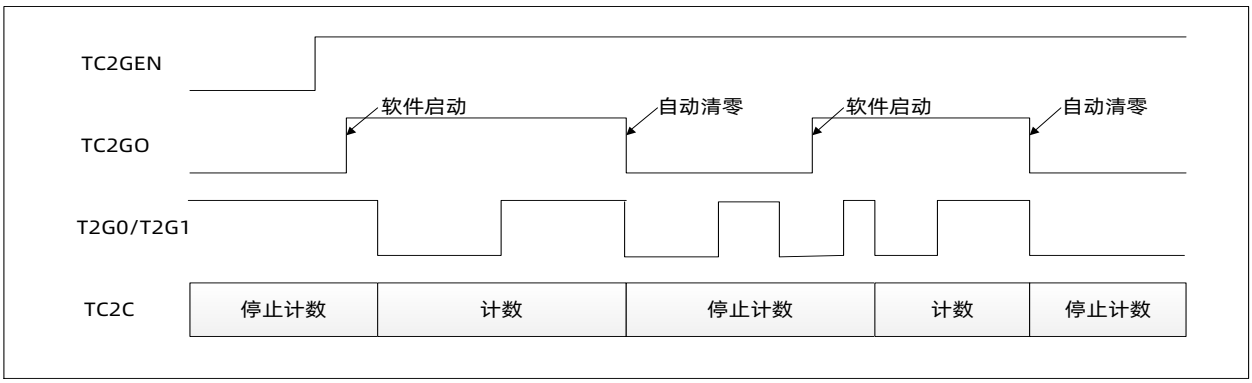
9.7.4 门控-上升沿到上升沿模式

上升沿到上升沿模式：启动门控计数后，门控逻辑从捕获到的第一个上升沿开始计数，然后捕获到第二个上升沿停止计数，如下图。



9.7.5 门控-下降沿到下降沿模式

下降沿到下降沿模式：启动门控计数后，门控逻辑从捕获到的第一个下降沿开始计数，然后捕获到第二个下降沿停止计数，如下图。



10 通用串行通讯口（USART）

10.1 概述

M9F6820 有两个 USART，支持异步全双工模式和同步半双工模式。
USART 的 TX、RX 口可以复用到任意 IO 口，详见章节 6.9。

10.2 TXxCR 发送控制寄存器(x = 0,1)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXxCR	TXxEN	TxMCLR	TxSYNC	TxL9	TxSLAVE	TxSPD[1:0]		TxD9
读/写	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	1	0	0	0	0	0	0

Bit 7 **TXxEN：**使能发送
0 = 屏蔽USART发送功能
1 = 使能USART发送功能

Bit 6 **TMxCLR：**发送寄存器空标志
0 = 正在发送数据，移位寄存器不空
1 = 数据已发送，移位寄存器空

Bit 5 **TxSYNC：**同步模式
0 = 选择异步模式
1 = 选择同步模式

Bit 4 **TxL9：**数据长度选择
0 = 8位数据
1 = 9位数据

Bit 3 **TxSLAVE：**同步发送/接收模式
0 = Master
1 = Slave

Bit [2:1] **TxSPD[1:0]：**波特率时钟源分频比

SPD[1:0]	波特率分频比(n)
00	4
01	16
10	64
11	256

Bit 0 **TxD9：**发送数据第9位数据

10.3 TXxREG 发送数据寄存器(x = 0,1)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXxREG	TXxREG[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	x	x	x	x	x	x	x	x

10.4 RXxCR 接收控制寄存器(x = 0,1)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RXxCR	RXxEN	RxCKPS	-	RxL9	SRxEN	RxOVF	FRxER	RxD9
读/写	R/W	R/W	-	R/W	R/W	R	R	R
复位后	0	0	-	0	0	0	0	x

Bit 7 **RXxEN：** 使能接收
0 = 屏蔽USART接收功能
1 = 使能USART接收功能

Bit 6 **RxCKPS：** 同步模式时钟模式选择
0 = 下降沿发送数据
1 = 上升沿发送数据

Bit 4 **RxL9：** 数据长度选择
0 = 8位数据
1 = 9位数据

Bit 3 **SRxEN：** 同步接收开始
0 = 停止同步接收
1 = 开始同步接收，单字节接收模式下接收完一个字节自动清零

Bit 2 **RxOVF：** 接受缓冲区溢出标志
0 = 接收缓冲区未发生溢出
1 = 接收缓冲区溢出，读缓冲区自动清零

Bit 1 **FRxER：** 接收数据格式错
0 = 当前接收数据未发生格式错
1 = 当前接收数据格式错（未收到停止位）

Bit 0 **RxD9：** 接收数据第9位数据

10.5 RXxREG 接收数据寄存器(x = 0,1)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RXxREG	RXxREG[7:0]							
读/写	R	R	R	R	R	R	R	R
复位后	x	x	x	x	x	x	x	x

10.6 BRGDxH 波特率寄存器高位(x = 0,1)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BRGDxH	SBYTEx	ODENx	STOPx1	-	-	-	BRGDxH[9:8]	
读/写	R/W	W	R/W	-	-	-	R/W	R/W
复位后	0	0	0	-	-	-	0	0

Bit 7 **SBYTEx：** 同步接收模式选择

0 = 多字节接收

1 = 单字节接收，接收完一个字节后自动清除SREN

Bit 6 **ODENx：** 开漏输出使能

0 = 不使能

1 = 使能

注：开漏使能时，并非真正意义上的开漏，即管脚允许接入的高电平依旧需要在芯片允许范围内。只是芯片发送数据位为 1（高电平）时，不是内部推挽输出的 1（高电平），此时输出 1（高电平）需外接电阻上拉置需要的 1（高电平）。

Bit [5] **STOPx1:停止位选择**

0 = 2位停止位

1 = 1位停止位

Bit [1:0] **BRGDxH[9:8]：** 10位波特率寄存器高2位

10.7 BRGDxL 波特率寄存器低位(x = 0,1)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BRGDxL	BRGDxL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **BRGDxL[7:0]：** 10位波特率寄存器低8位

10.8 USART 使用说明

10.8.1 波特率设置

通过设置 BRGD 和 SPD（分频比 n）来获得所需的波特率。
波特率计算公式： 目标波特率 = F_{OSC}/((BRGD+1)×n)

常用波特率设置（F_{OSC} = 16MHz）

目标波特率	波特率时钟源分频比	BRGD	偏差
300	256	0xCF	0.16%
600	256	0x67	0.16%
1200	64	0xCF	0.16%
2400	64	0x67	0.16%
4800	16	0xCF	0.16%
9600	16	0x67	0.16%
19200	4	0xCF	0.16%
38400	4	0x67	0.16%
57600	4	0x44	0.64%
115200	4	0x22	-0.79%

常用波特率设置（Fosc = 24MHz）

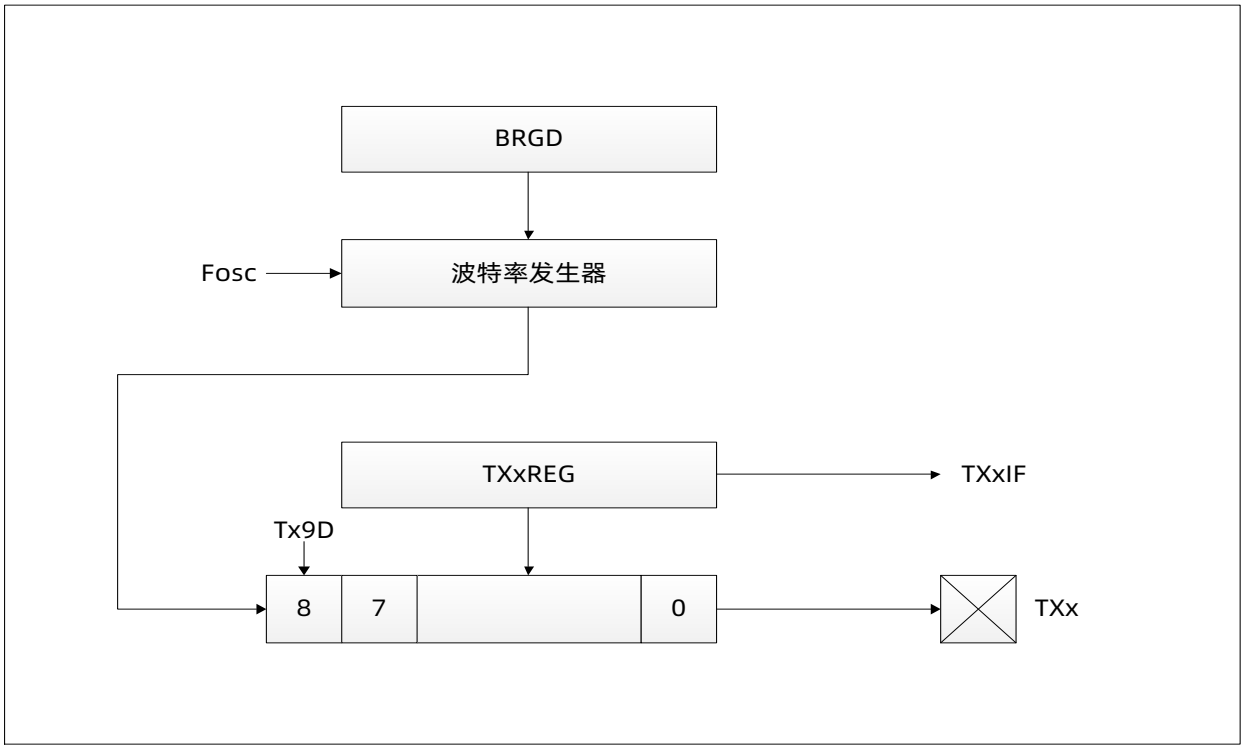
目标波特率	波特率时钟源分频比	BRGD	偏差
300	256	0x137	0.16%
600	256	0x9B	0.16%
1200	64	0x137	0.16%
2400	64	0x9B	0.16%
4800	16	0x137	0.16%
9600	16	0x9B	0.16%
19200	4	0x137	0.16%
38400	4	0x9B	0.16%
57600	4	0x67	0.16%
115200	4	0x33	0.16%

10.8.2 异步发送

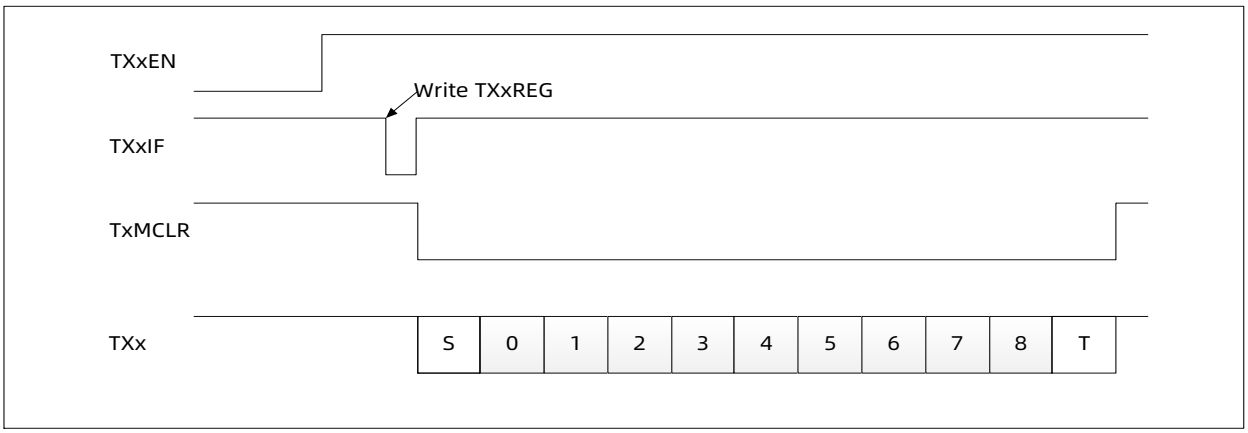
当 TXxEN 使能时，TXxIF 中断标志为 1 说明 TXxREG 发送寄存器为空，TxMCLR = 1 说明发送移位寄存器为空，发送器处于空闲状态。

空闲状态时写入 TXxREG，写入数据将立即装载到发送移位寄存器中，此时，TXxIF 为 1，TxMCLR=0，发送器进入发送状态。此时再次写入 TXxREG，TXIF 将清零，说明 TXxREG 有未发送数据，发送移位寄存器发送完毕后，TXxREG 数据将自动载入发送移位寄存器继续发送，且 TXxIF 为空。

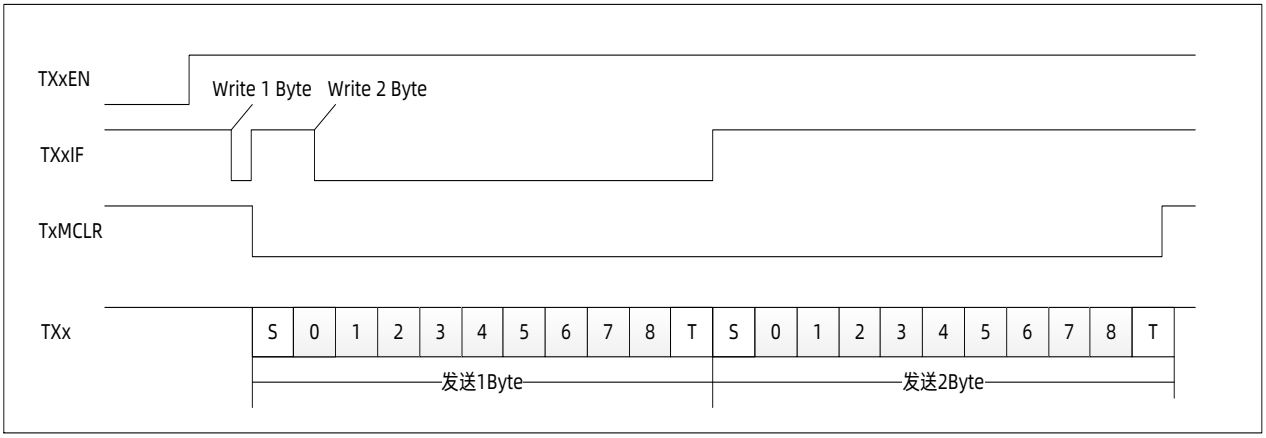
当 TXxIF 为 0 时写入 TXxREG，将覆盖上次写入数据。



单字节发送：



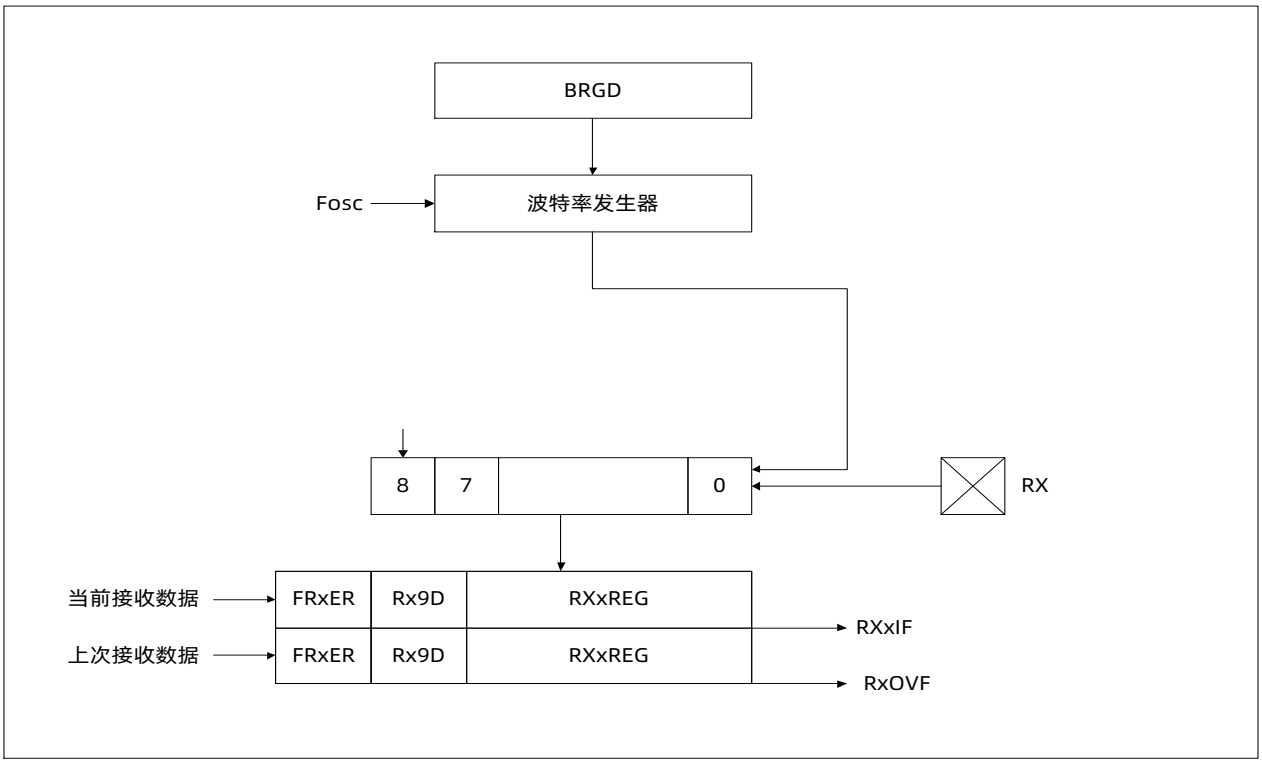
多字节发送：

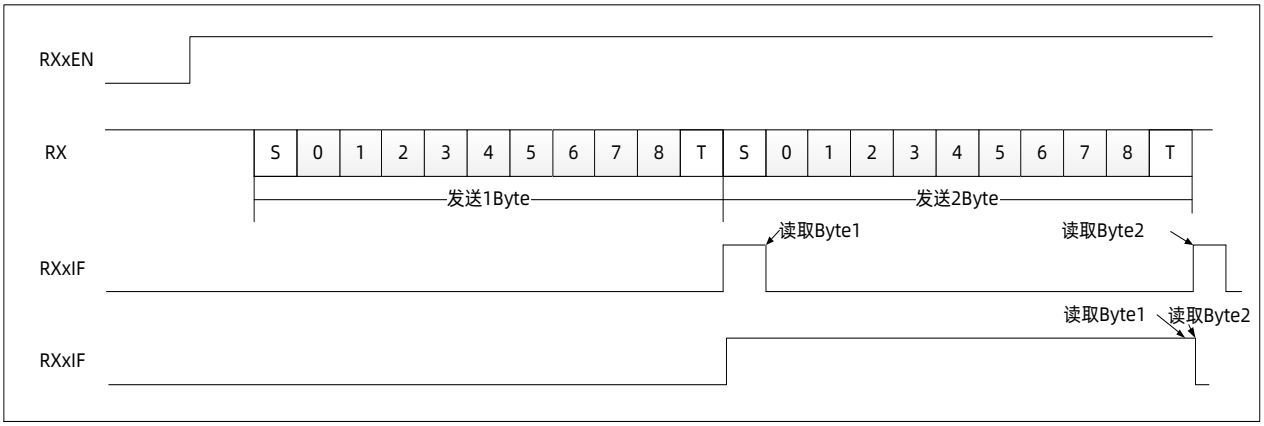


STEP1：设置波特率 SSPD = X，BRGD = X，TxSYNC = 0
STEP2：设置 TXxEN = 1，设置数据模式 TX9 = X
STEP3：写入数据高位 TXD9
STEP4：写入 TXxREG，启动发送
STEP5：当 TXxIF = 1 时，写入 TXxREG 发送下一个字节
STEP6：重复 STEP5，直到该帧数据发送完成

10.8.3 异步接收

设置异步模式，使能 RXxEN，开始启动异步接收。RX 管脚处于高电平时，接收器处于空闲状态，当检测到 RX 变为低电平，接收器检测该低电平是否有效起始位，若为有效起始位，则启动数据时钟恢复电路和数据恢复电路进行接收。1 个数据接收完成后，RXxIF 置 1，当接收 3 个数据未读取，RxOVF 置 1，同时舍弃第三个接收数据。完全读取 RXxREG 后 RXxIF 自动清零。





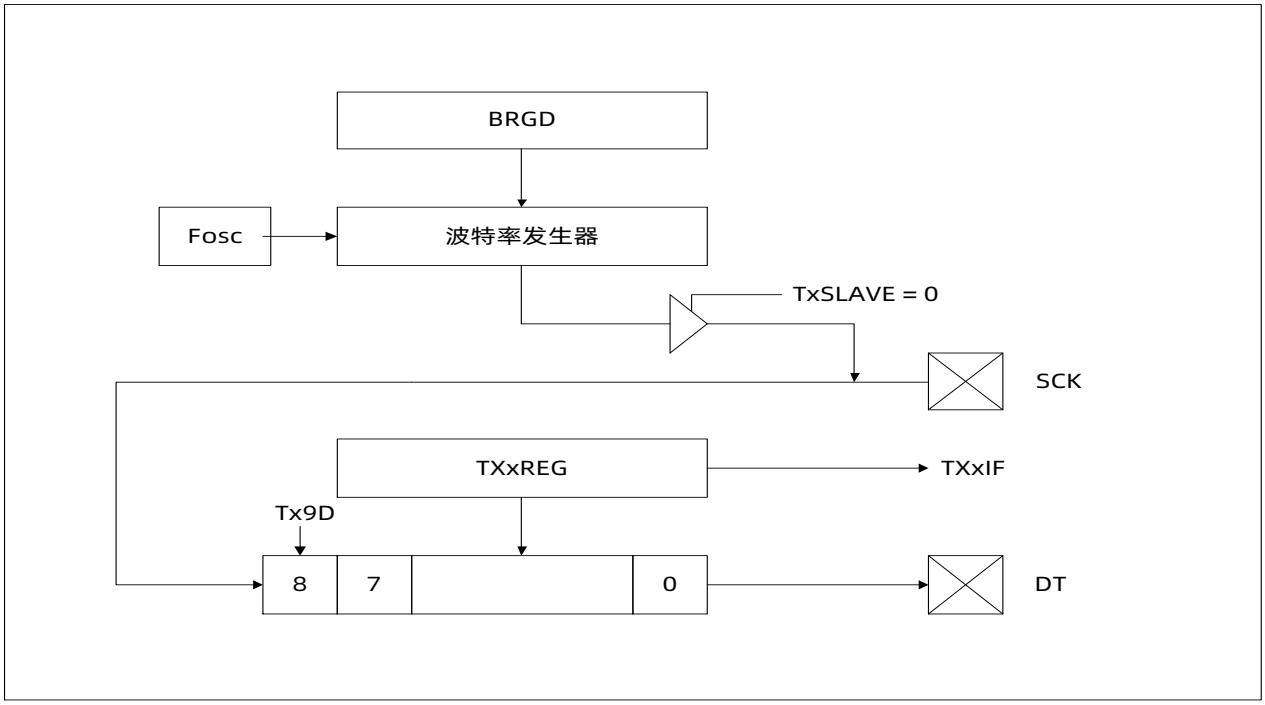
STEP1：设置波特率 SSPD = X，BRGD = X，TxSYNC = 0
STEP2：设置 RXxEN = 1
STEP3：等待接收完成 RXxIF = 1
STEP4：判断 FRxER = 0，若为 1，帧格式错误，舍弃数据
STEP5：读取 Rx9D
STEP6：读取 RXxREG，重复 3-6

10.8.4 同步发送

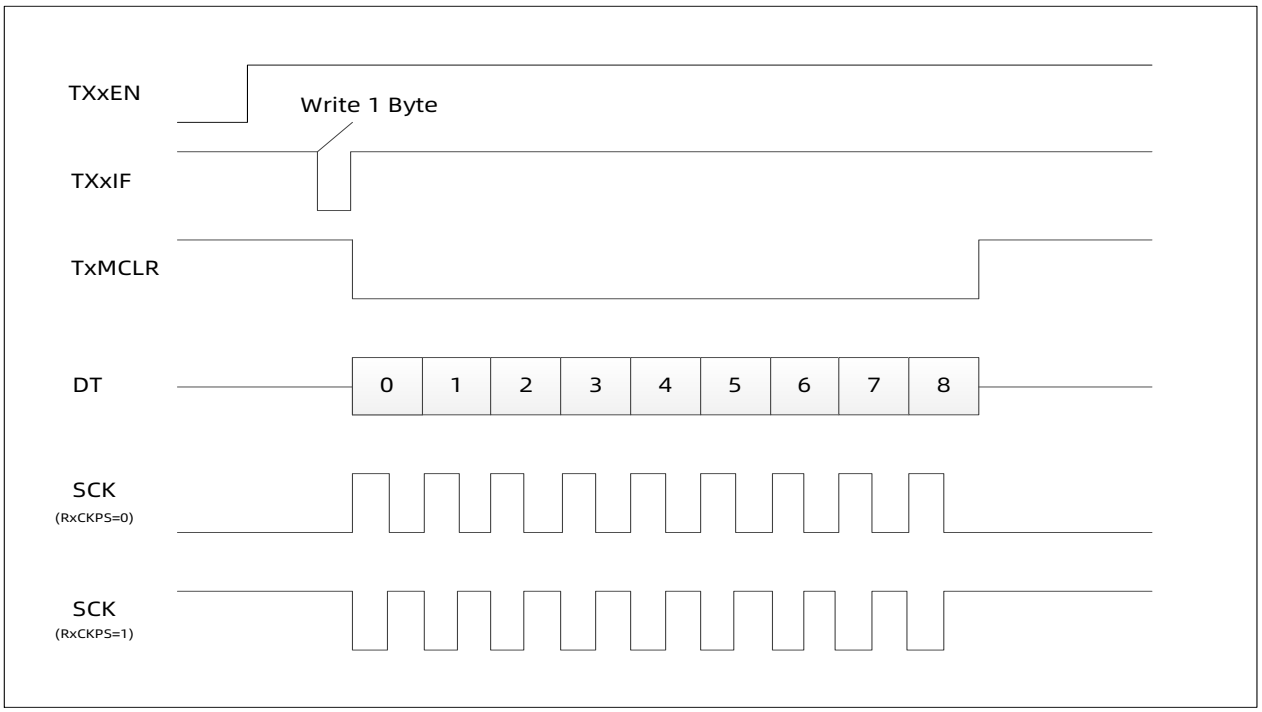
当 TXxEN=1，TxSYNC=1 时，使能同步发送功能。 RxCKPS 选择发送时钟极性， TXxIF 中断标志为 1 说明 TXxREG 发送寄存器为空，TxMCLR=1 说明发送移位寄存器为空，发送器处于空闲状态。

空闲状态写入 TXxREG，写入数据将立即装载到发送移位寄存器中，此时，TXxIF 为 1，TxMCLR=0，发送器进入发送状态。此时再次写入 TXxREG，TXxIF 将清零，说明 TXxREG 有未发送数据，发送移位寄存器发送完毕后，TXxREG 数据将自动载入发送移位寄存器继续发送，且 TXxIF 为空。

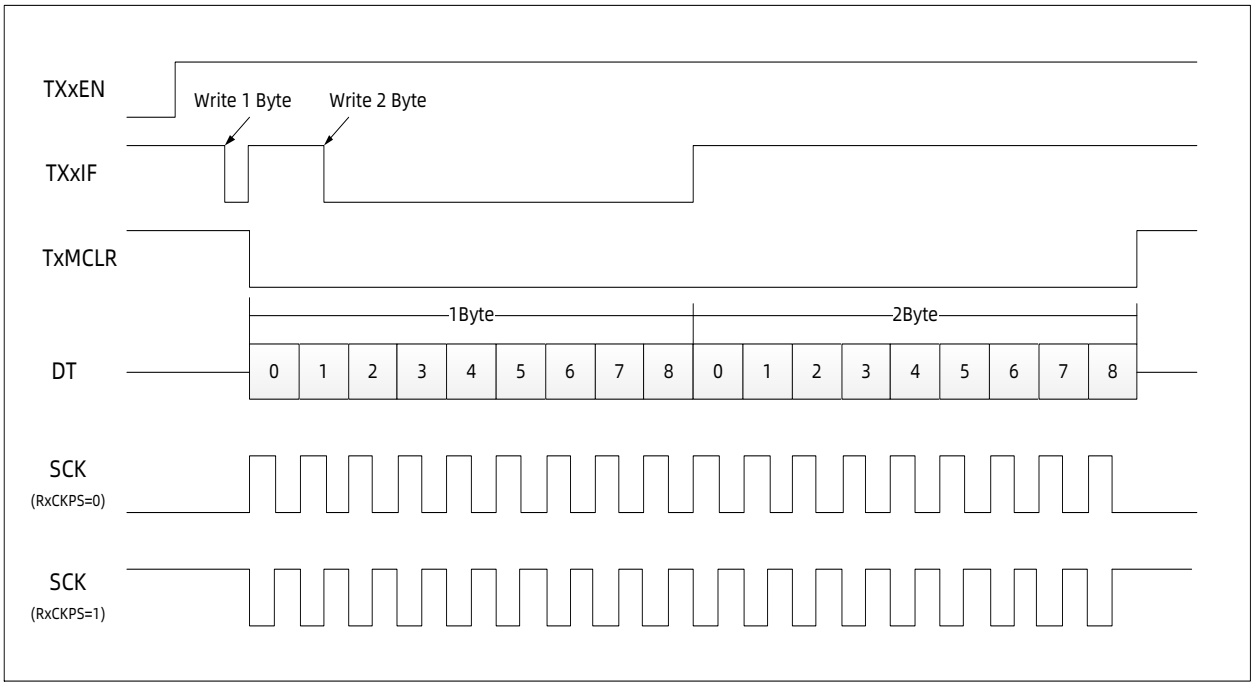
当 TXxIF 为 0 时写入 TXxREG，将覆盖上次写入数据。



单字节发送：



多字节发送：



参考操作步骤 TxSLAVE = 0：

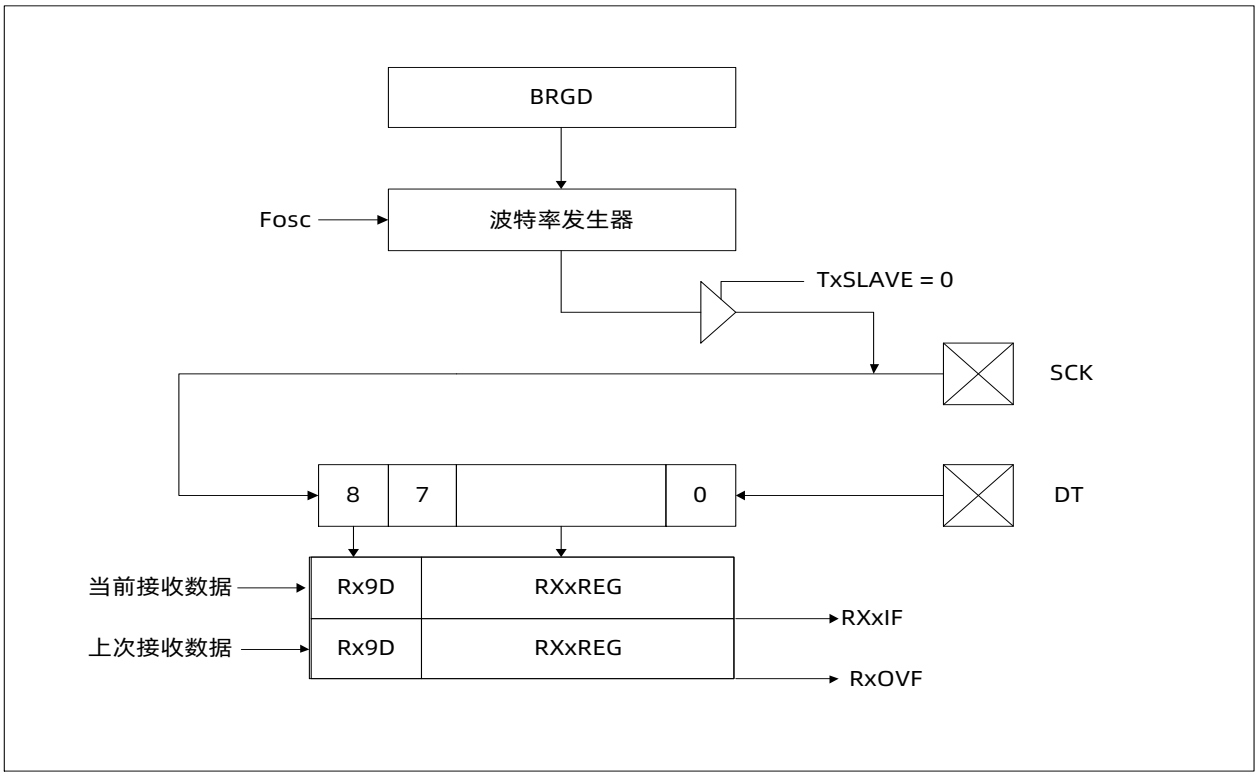
STEP1：设置波特率 SSPD = X，BRGD = X，TxSYNC = 1
STEP2：设置 TXxEN = 1，设置数据模式 TxL9 = X
STEP3：写入数据高位 TxD9
STEP4：写入 TXxREG，启动发送
STEP5：当 TXxIF = 1 时，写入 TXxREG 发送下一个字节
STEP6：重复 STEP5，直到该帧数据发送完成

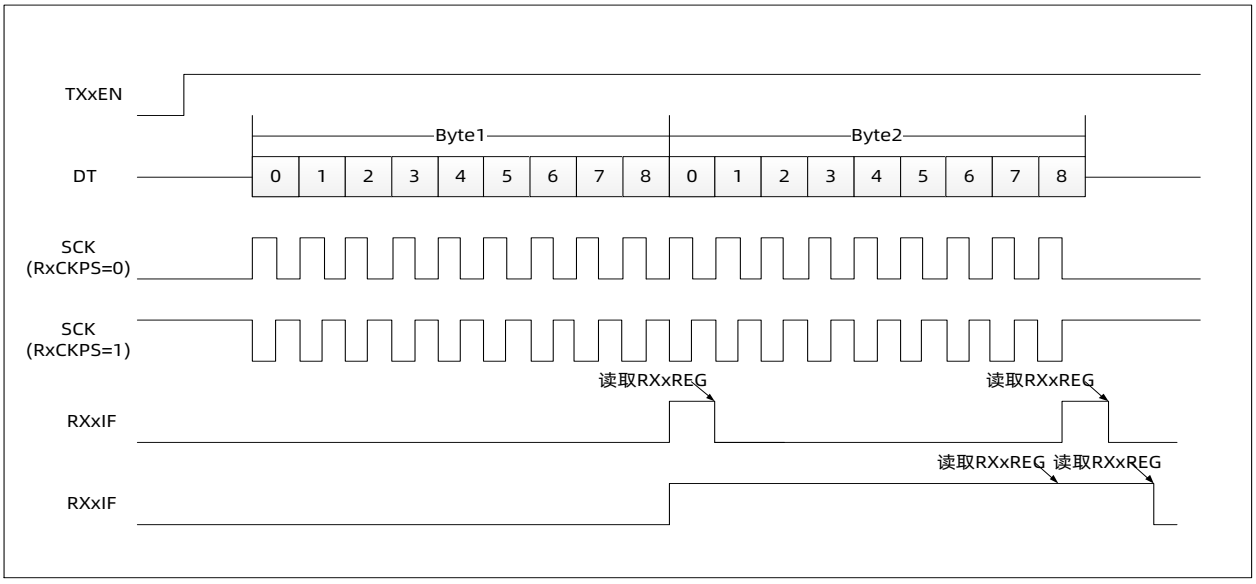
参考操作步骤 TxSLAVE = 1：

STEP1：设置波特率 SSPD=X，BRGD=X，TxSYNC=1
STEP2：设置 TXxEN=1，设置数据模式 TxL9=X
STEP3：当 TXxIF=1 时，写入数据高位 TxD9
STEP4：写入 TXxREG 等待发送下一个字节
STEP5：重复 STEP3-4，直到该帧数据发送完成

10.8.5 同步接收

设置同步 TxSYNC=1 模式，使能 RXxEN，开始启动异步接收。RX 管脚处于高电平时，接收器处于空闲状态，当检测到 RX 变为低电平，接收器检测该低电平是否有效起始位，若为有效起始位，则启动数据时钟恢复电路和数据恢复电路进行接收。1 个数据接收完成后，RXxIF 置 1，当接收 3 个数据未读取，RxOVF 置 1，同时舍弃第三个接收数据。完全读取 RXxREG 后 RXxIF 自动清零。





参考操作步骤 TxSLAVE=0：

STEP1：设置波特率 SSPD=X，BRGD=X，TxSYNC=0

STEP2：设置 RXxEN=1

STEP3：写 SRxEN 启动接收

STEP4：等待接收完成 RXxIF=1

STEP5：读取 Rx9D

STEP6：读取 RXxREG，单字节接收（SBYTEx=1）重复 3-6；多字节接收（SBYTEx=0）重复 4-6

参考操作步骤 TxSLAVE=1：

STEP1：设置波特率 SSPD=X，BRGD=X，TxSYNC=0

STEP2：设置 RXxEN=1

STEP3：写 SRxEN 启动接收

STEP4：等待接收完成 RXxIF=1

STEP5：读取 Rx9D

STEP6：读取 RXxREG，单字节接收（SBYTEx=1）重复 3-6；多字节接收（SBYTEx=0）重复 4-6

10.8.6 唤醒及休眠模式下通讯

TXxIE 置 1 时，TXxIF 中断标志唤醒 CPU

RXxIE 置 1 时，RXxIF 中断标志唤醒 CPU

异步接收时，检测到 START 位将自动使能高频振荡器，接收完成后唤醒 CPU。

同步接收时，若作为主机，则休眠状态下部工作；作为从机，则接收 1 个字节完成后唤醒 CPU。

11 I2C

11.1 概述

I2C 总线作为微控制器与 I2C 设备之间的串行接口。连接在 I2C 总线上的多个设备之间可以相互交换信息。

I2C 总线使主机和从机之间数据可以双向传输。没有中心主机，通过仲裁同时允许多主机系统。同步串行时钟允许器件之间通过一条串行总线以不同位率传输。I2C 总线支持 4 种传输模式，包括主机发送模式，主机接收模式，从机发送模式和从机接收模式。I2C 接口仅支持 7 位寻址模式和广播呼叫。

输出方式为开漏输出。可以通过配置上拉控制寄存器开启上拉电阻。

I2C 的 SDA、SCL 口可以复用到任意 IO 口，详见章节 6.9。

11.2 功能描述

对于双向传输操作，SDA 和 SCL 引脚必须开漏模式。该接口最基本的操作是执行线与功能。当一个或多个 I2C 器件输出 0 时，I2C 总线上为低电平。当所有 I2C 器件输出 1 时，产生高电平，允许上拉电阻将总线拉高。

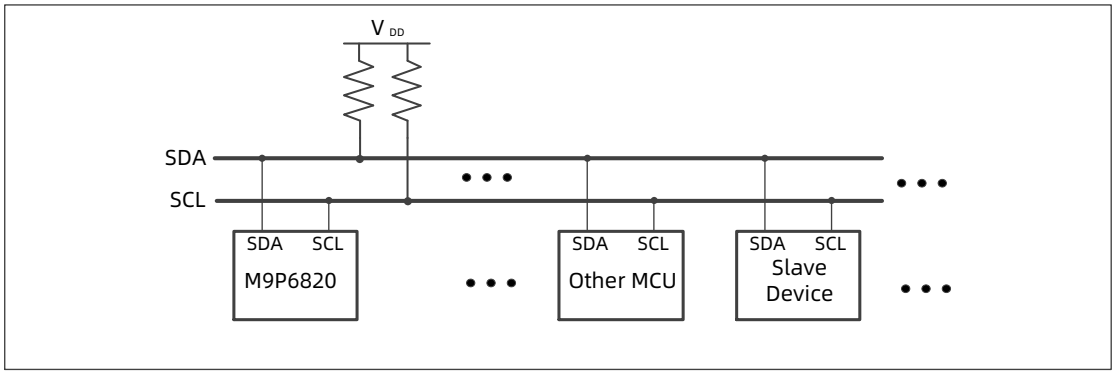


图 11.2-1 I2C 总线连接图

当两线都为高时，I2C 空闲。同时，任何器件可作为主机占用总线并在产生起始信号后传输，在发送停止信号结束传输之前，总线视为忙。主机产生所有串行时钟脉冲和起始与停止信号。然而如果总线上没有起始信号，所有器件均作为没被寻址的从机。硬件寻找自己的从机地址或广播呼叫地址。(广播呼叫地址检测由 GC 使能或禁止。)如果接收到的地址匹配时，请求中断。

I2C 总线上的每个传输为 9 位长度，由 8 位数据（先 MSB）和一个应答位组成。每次传输的字节数(一次有效的 START 与停止信号之间)不受限止，但每个字节后都跟随着一个应答位。主机产生 8 个时钟脉冲发送 8 位数据，在 SCL 总线上的第 8 个下降沿，器件将 SDA 改变输出为输入，并在第 9 个时钟脉冲读应答值。第 9 个时钟脉冲之后，如果下一次接收还没有准备好，数据接收器件将 SCL 总线拉低，从而迫使下一字节的传输暂停。当接收器释放 SCL 总线，数据传输继续。

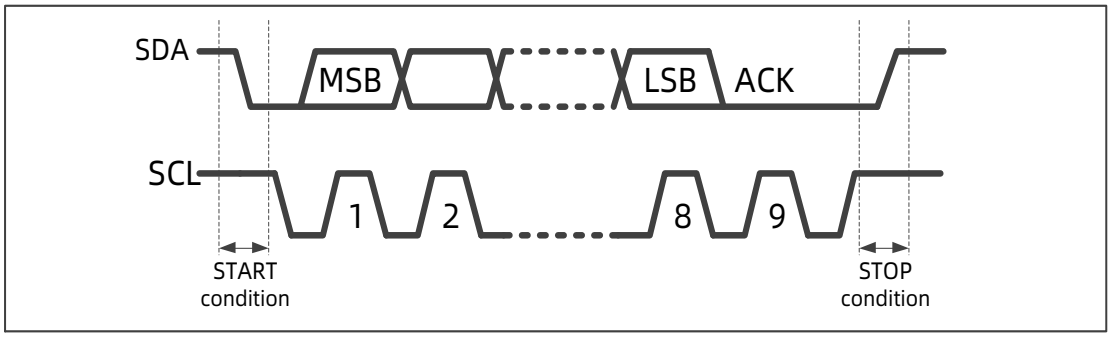


图 11.2-2 I2C 总线数据协议图

11.2.1 起始 START 和停止 STOP 信号

I2C 总线协议定义两个状态开始和结束传输，起始(S) 和停止(P) 信号。起始信号：当 SCL 为高时，在 SDA 总线上有从高到低的电平变化。停止信号：当 SCL 为高时，在 SDA 上有从低到高的电平变化。起始或停止信号常由主机产生，在起始信号产生之后 I2C 总线视为忙，在停止信号之后，I2C 总线视为空闲。停止信号出现后，主机设备将释放控制权并返回为无寻址从机。因此，原来寻址的从机将变成未被寻址的从机，I2C 总线空闲并等待下一个起始信号。

通常由主机产生停止信号中止数据传输，然而，如果主机仍希望在总线上通信，就会重复产生起始信号(重复 START) 和地址。传输中可能存在着各种结合的读/写格式。

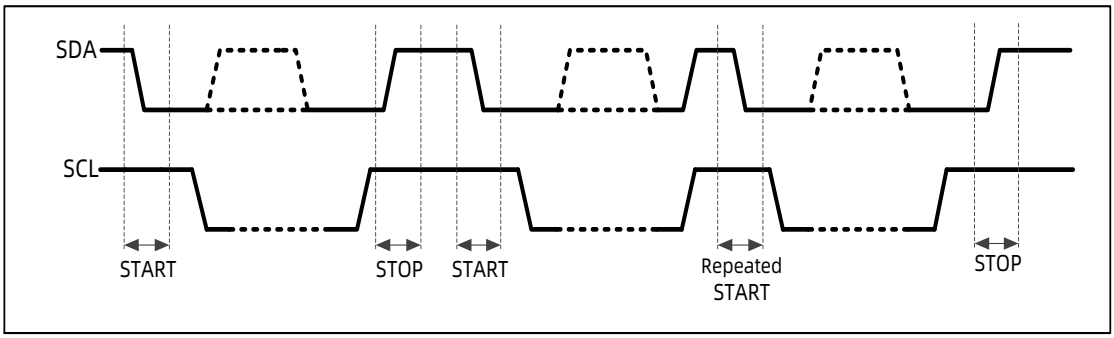


图 11.2-3　起始 START、重复起始 Repeated START 和停止 STOP 信号

11.2.2 7 位地址数据格式

产生起始信号之后，应该由主机传输一个字节的特殊数据，包括在第 8 位数据方向位(R/W)之后的 7 位从机地址(SLA)，寻址目标从机并决定数据流的方向。如果 R/W 位为 0，表示主机向所选从机写信息，如果该位为 1，表示主机从从机读取信息。一个地址包由从机地址和读(R)或写(W) 位组成，分别称之为 SLA+R 或 SLA+W。一次传输基本上由一个起始信号，一个 SLA+R/W，一个或多个数据包和一个停止信号组成。在 SLA+R/W 指定从机地址后，第二个和之后的 8 位数据字节由主机或从机地址根据 R/W 位配置。

“广播呼叫”是个例外，它可以通过将第一个字节的数据全部赋值为 0 来寻址所有器件。广播呼叫用于当主机希望发送相同信息到几个从机时情况。当该地址在使用时，其他器件根据软件配置可能响应应答或忽略。如果器件响应广播呼叫，其操作就像从机接收器模式。

在数据传输过程中，SDA 总线上的数据必须在时钟为高的期间内保持稳定，数据总线仅在 SCL 为低时改变。

11.2.3 应答

任何传输字节的第 9 个 SCL 脉冲被视作应答信号(ACK)。通过将 SDA 拉低，允许接收器件(可以为主机或从机)对发送器件的响应(可以为主机或从机)。应答由主机产生相关时钟脉冲，发送器件必须在应答时钟脉冲期间释放 SDA 总线控制，ACK 为低电平有效信号，在时钟脉冲为高时将 SDA 总线拉低，通知发送器接收器已经接收到数据。通常，被寻址的接收器在接收到一个字节之后要求产生一个 ACK，当从机接收器没有应答(NACK)从机地址，SDA 线必须由从机拉离高电平，以让主机产生停止或重复起始信号。

如果从机接收器应答了从机地址，将切换到不寻址从机模式，不再接收任何数据，该从机保持 SDA 总线为高，主机应该产生停止信号或复位起始信号。

如果主机接收器传输时，由于主机在传输时控制字节数目，就必须向从机发送器标记数据的结束，而不是在最后一个字节产生应答信号。从机发送器切换到不寻址模式，并释放 SDA 总线，允许主机产生停止信号或复位起始信号。

11.2.4 仲裁

主机仅在总线空闲时开始传输，可能是两个或更多主机产生起始信号。在这种情况下，当 SCL 为高时，就要在 SDA 总线上有仲裁。仲裁期间，相互竞争的第一个主机设备在 SDA 上置 1(高)，其他主机发送 0(低)，由于其电平与自身电平不匹配而关闭其数据输出，仲裁失败的主机立即切换成不被寻址从机，并检测自身的从机地址以判断是否被仲裁胜出的主机寻址，它也释放 SDA 为高电平，以防影响胜出主机开始数据传输。但是，仲裁失败的主机继续在 SCL 上产生时钟脉冲，直到它失去仲裁的最后字节输入。如果检测的地址与仲裁失败的主机的自身从机地址相匹配，它就会切换到被寻址的从机模式。

输出数据后，所有主机进行仲裁以持续侦测 SDA 总线。如果从 SDA 总线上读取的值与主机输出的值不匹配时，就失去仲裁。注，当一个主机在其他主机输出低时，自身输出一个高的 SDA 值时就可能失去仲裁，只要有一个主机保存，仲裁就将继续，这可能会占用很多位。如果几个主机试图寻址相同从机时，仲裁将继续输入数据包。

仲裁可以超过几位，第一阶段是比较地址位，如果主机试图寻址相同器件，仲裁继续比较数据位或应答位。

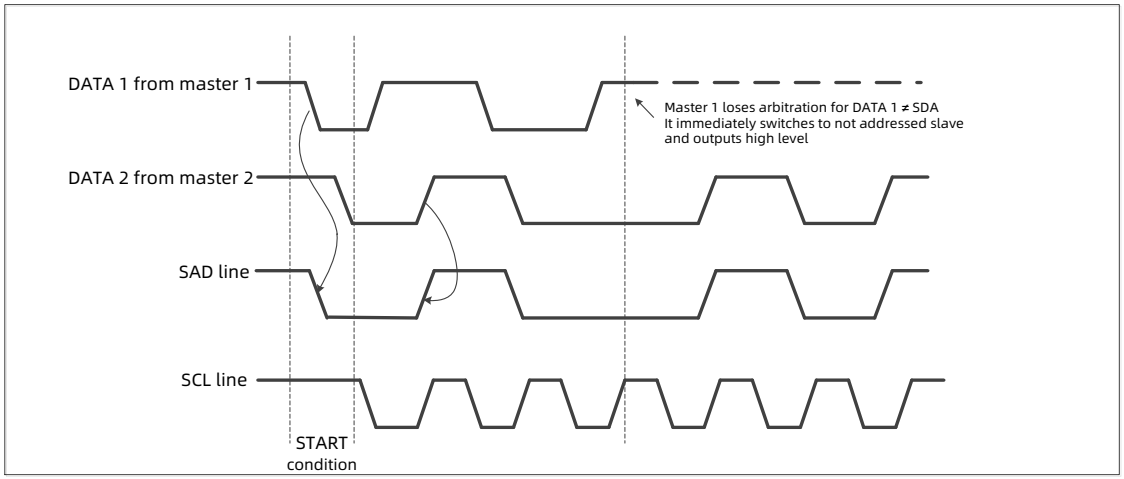


图 11.2-4 两台主机仲裁过程

I2C 总线的这种仲裁机制，让总线上的设备可以有多个主机，而且没有优先等级。从机不介入仲裁。

11.3 工作模式

I2C 协议定义了四种模式：主机发送，主机接收，从机发送和从机接收。还有一种特殊模式广播呼叫模式，其操作方式与从机接收模式类似。

11.3.1 主机发送模式

在主机发送模式下，向从机接收器发送数据。主机通过 CR[2:0]设置期望时钟速率并向 I2CEN 写 1 使能 I2C 总线，设置 ST 为 1 进入主机发送模式，只要总线空闲，硬件将测试总线并产生起始信号，成功产生起始信号后，SI 标志将置 1 且 I2CSTA 的状态码为 08h，之后就是给 I2DATA 载入目标从机地址和数据方向位“写” (SLA+W)，SLA+W 开始传输时 SI 位必须清零。

在 SLA+W 字节被发送且由被寻址的从机器件返回应答(ACK)后，SI 标志再次置 1 且 I2CSTA 读出为 18h，依据用户定义的通信协议持续发送数据，全部数据发送完成后，主机可以通过设置 STO 发送停止信号并清 SI 中止传输，在没有发送停止信号下重复起始信号（重复 START）将立即开始另一轮传输。

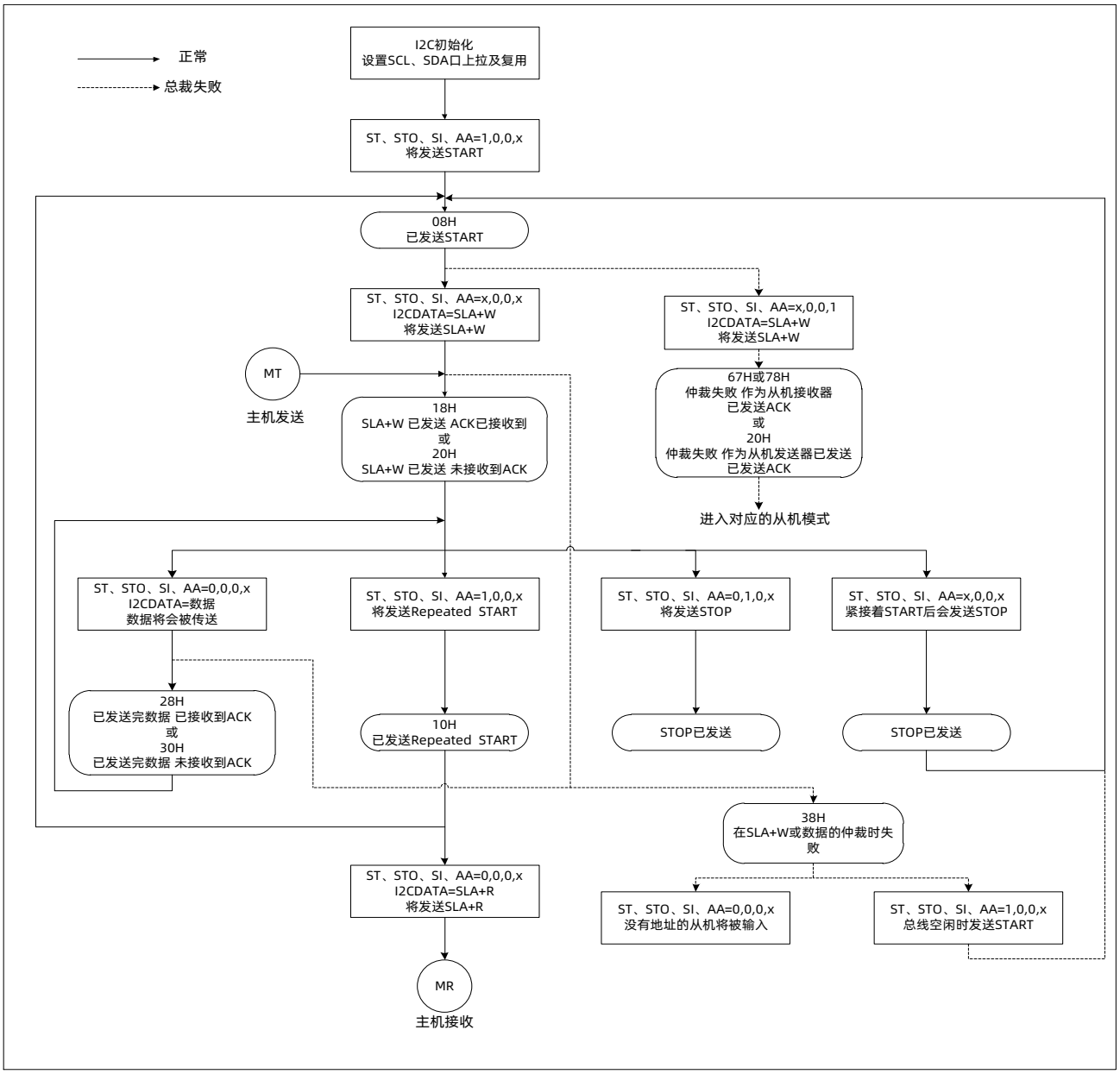


图 11.3-1 主机发送模式流程与状态图

11.3.2 主机接收模式

在主机接收模式下，从从机发送器接收数据。传输开始与主机发送模式相似，在起始信号之后，I2DATA 应该加载目标从机地址和数据方向位“读”(SLA+R)，SLA+R 字节发送后，且返回应答位 AA，重新置位 SI 标志且 I2CSTA 读出为 40h，SI 标志应该被清零以便开始接收从机发送过来的数据，如果 AA 置 1，主机接收器接收到数据后将应答从机发送器，如果清零 AA，主机接收器接收到数据后将不会应答从机，并释放从机发送器为不被寻址的从机，然后，主机产生停止信号中止传输或重复起始信号开始另一轮传输。

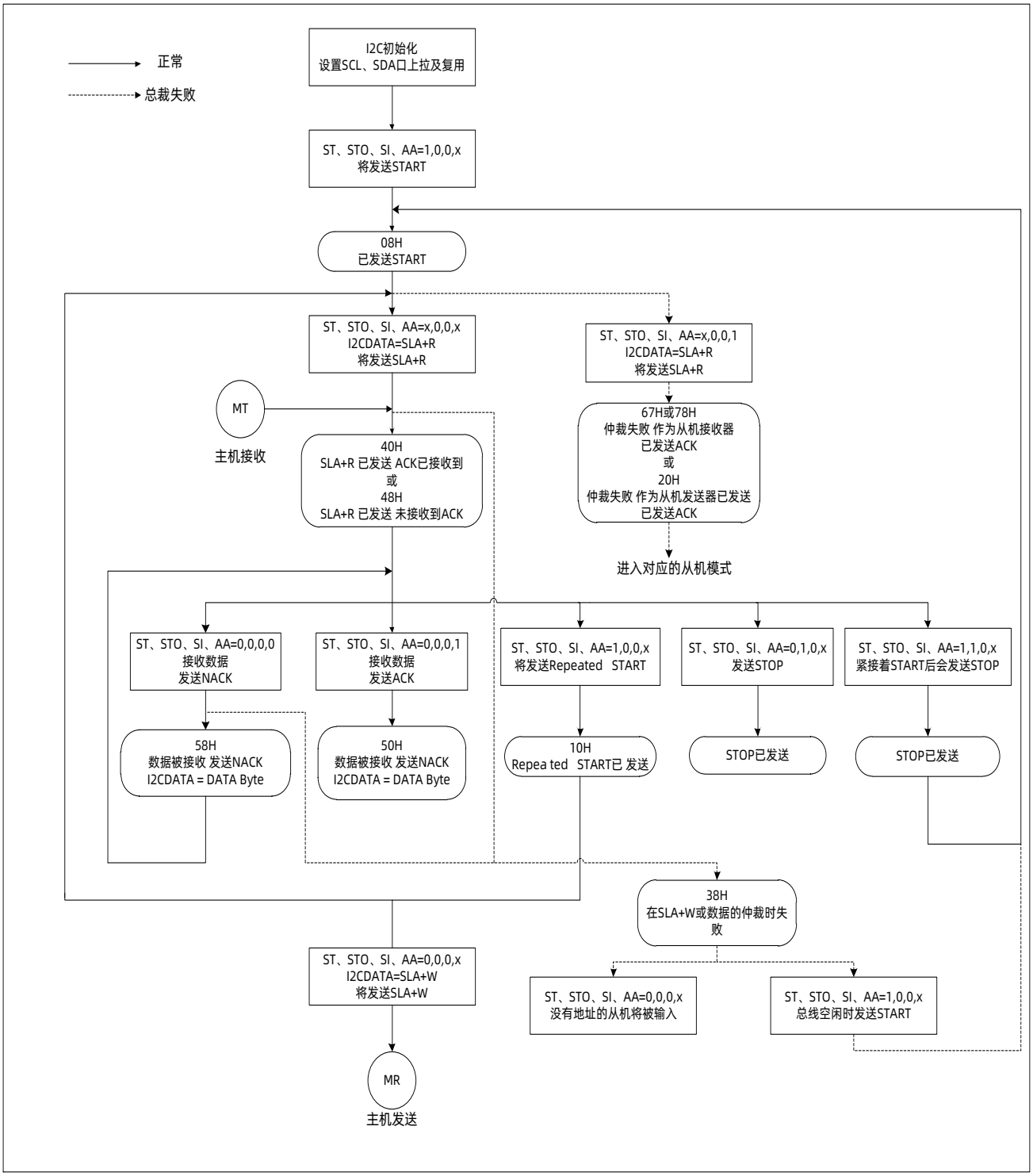


图 11.3-2　主机接收模式流程与状态图

11.3.3 从机接收模式

在从机接收模式下，从主机发送器接收数据。发送开始之前，I2CADR 必须装载响应器件的地址，以让主机寻址，从机模式下 CR[2:0]无效，AA 位必须设置使能应答自身从机地址或广播呼叫，完成以上初始过程后，I2C 等待自身地址被寻址与数据方向位“写” (SLA+W)或被广播呼叫寻址。如果在仲裁失败时，也可以进入从机接收模式。

在从机被 SLA+W 寻址后，需清 SI 标志以便接收主机发送过来的数据，传输期间，如果 AA 位为 0，从机将在下一次接收到的数据字节之后返回无应答 NACK，从机也不被寻址并与主机分离，不能接收 I2CDATA 的任何字节，而保持当前接收到的数据。

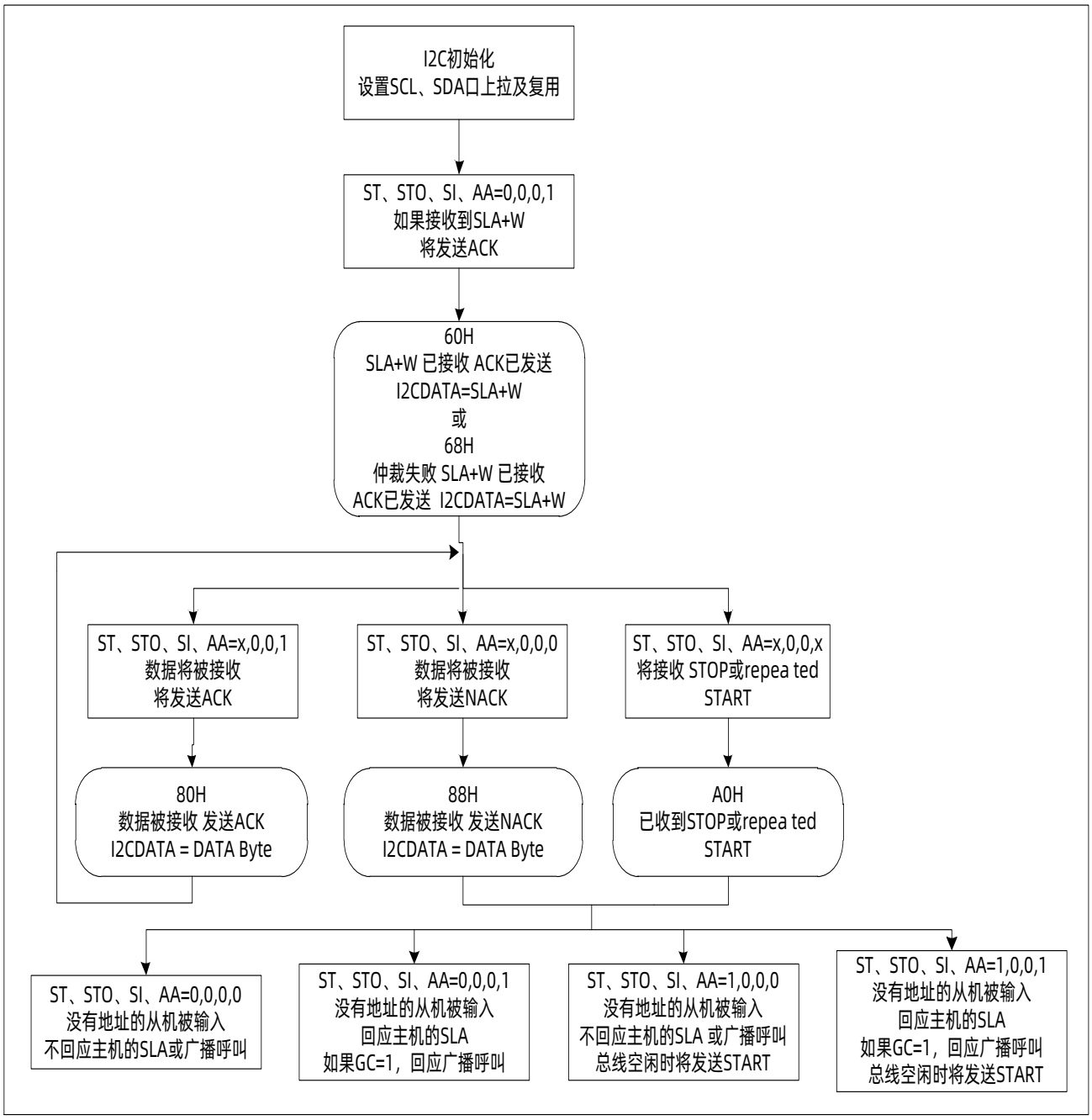


图 11.3-3　　从机接收模式流程与状态图

11.3.4 从机发送模式

在从机发送模式下，发送几个字节数据到主机接收器。确定 I2CADR 和 I2CCR 的值之后，I2C 等待自己的地址被寻址“读”(SLA+R)。如果仲裁失败后，也可以进入从机发送模式。

在从机被 SLA+R 寻址后，应该清 SI 标志以便传输数据到主机发送器，通常主机接收器将在从机发送每个字节数据之后返回应答，如果没有接收到应答，如果继续传输将发送全 1，并成为不被寻址的从机，如果在传输中清了 AA 标志，从机发送最后一个字节数据，下一次传输数据全为 1，从机变为未定址从机。

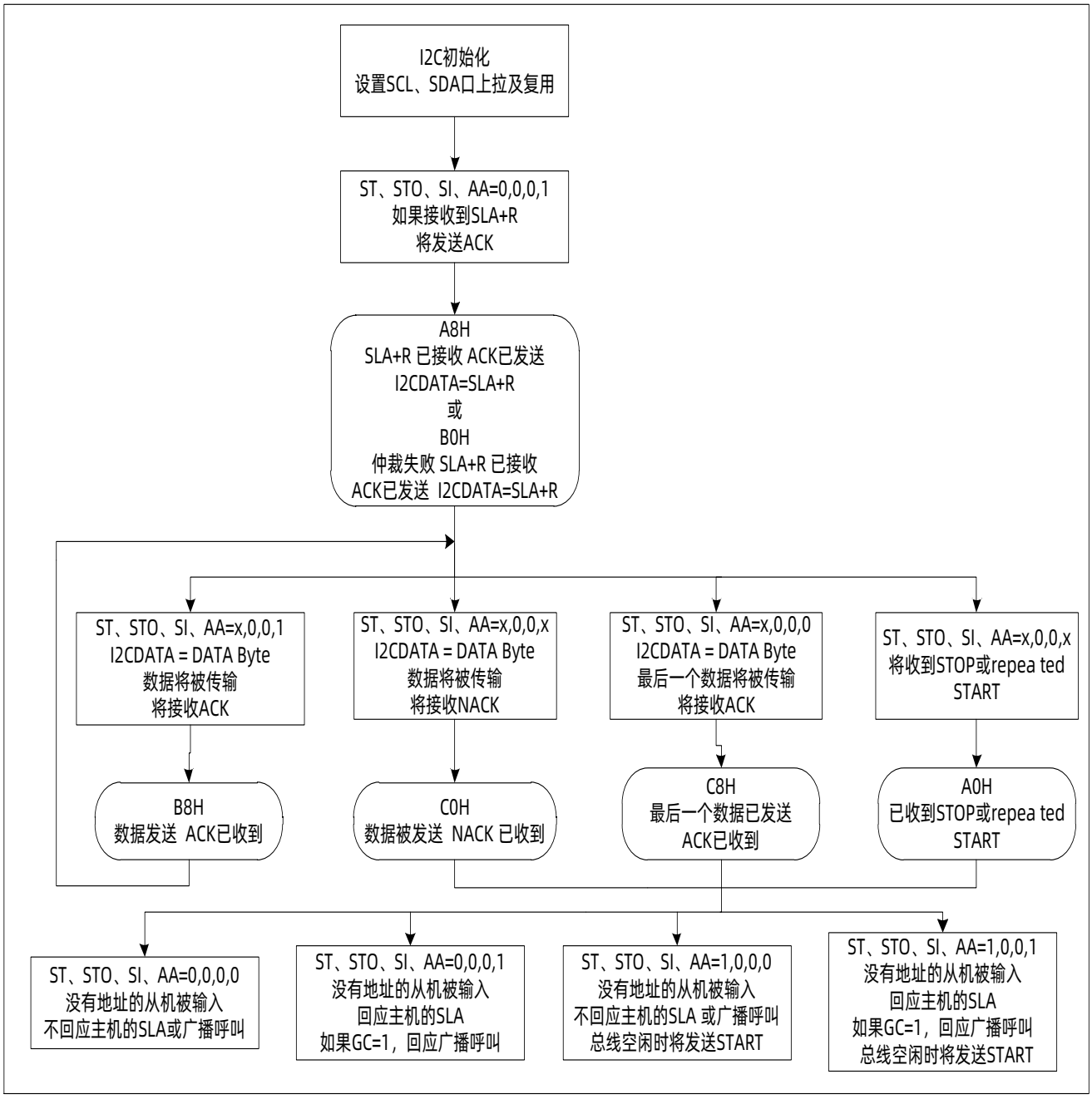


图 11.3-4 从机发送模式流程与状态图

11.3.5 广播呼叫

广播呼叫是从机接收模式的一种特殊情况，即从机地址和数据方向位全为 0，且 GC 及 AA 都置 1，使能接收广播呼叫模式。被广播呼叫寻址的从机在正常从机接收模式的 I2CSTA 里有不同状态码，如果仲裁失败也可以进入广播呼叫模式。

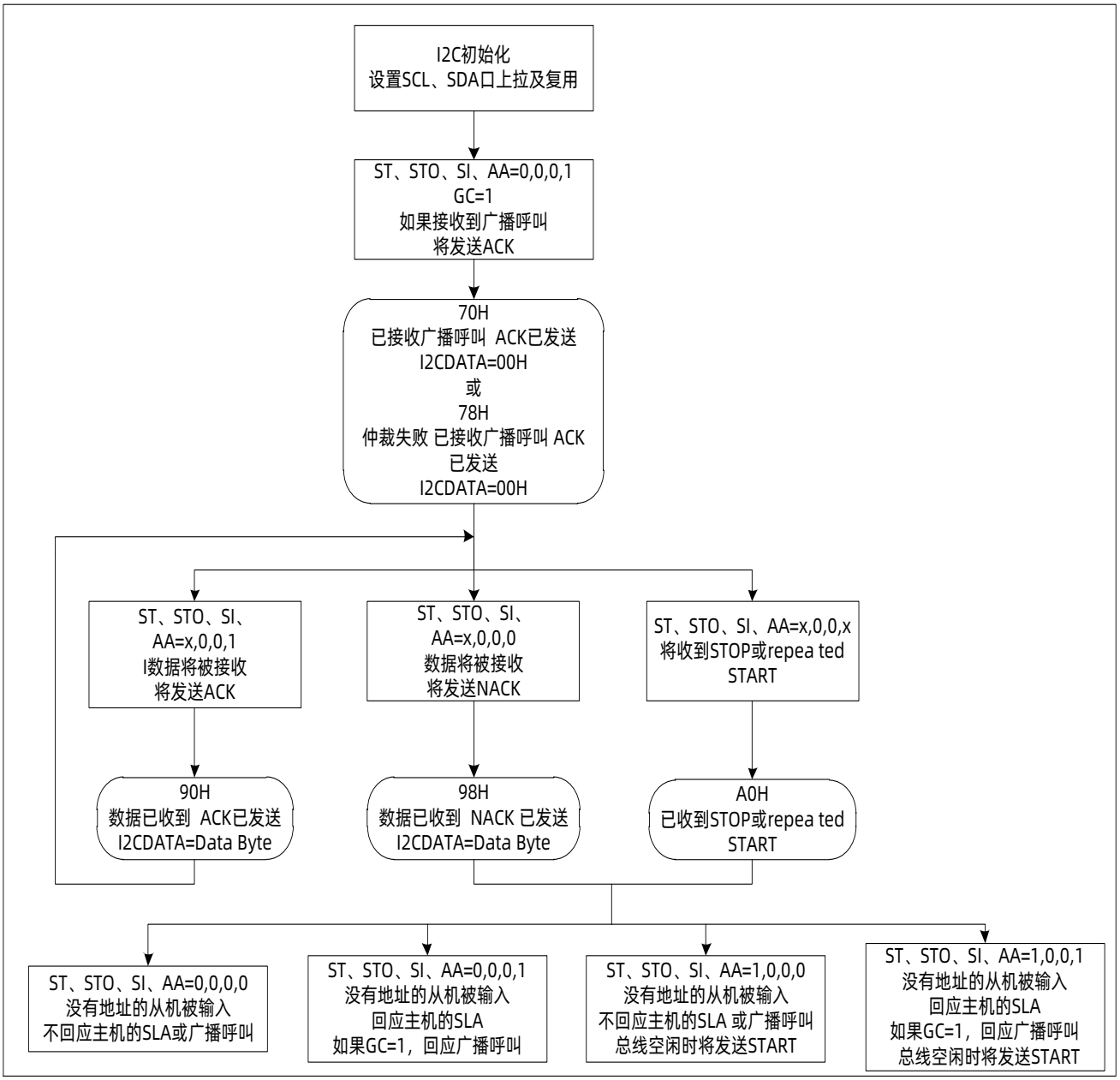


图 11.3-5 广播呼叫模式流程与状态图

11.3.6 其它状态

有两个 I2CSTA 状态码与 25 个定义状态不一致，即前面提到的 F8h 和 00h 状态。

第一个状态码 F8h 表示在每次传输期间没有得到相关信息，同时，SI 标志为 0 且没有 I2C 中断请求。

另一个标志码 00h 意味在传输过程中发生错误，总线错误是由 START 或停止信号暂时出现在一个非法的位置，如地地址字节里第 2 位换到第 8 位，或数据字节包括应答位，当出现总线错误时，SI 标志立即置 1，当在 I2C 总线上检测到总线错误，工作器件立即切换到不被寻址从机模式，释放 SDA 和 SCL 总，置位 SI 标志，将 00H 载入 I2CSTA。要从总线错误恢复，STO 位必须设置为逻辑 1 且 SI 必须清零，然后，STO 由硬件清零且在没有停止信号就释放 I2C 总线。

特例：如果没有成功产生 START 或重复起始信号，I2C 总线被 SDA 的低电平阻挡，如一个从机器件没有位同步，可以通过在 SCL 总线上发送额外时钟脉冲解决这个问题。当 ST 位置 1 时，I2C 硬件发送额外时钟脉冲，但是由于 SDA 被拉低，不能产生起始信号，当 SDA 总线最终被释放，发送一个普通的 START 条件，进入状态 08h，继续进行串行传输。当 SDA 为低，如果发送重复起始信号，I2C 硬件也执行以上相同的动作。此情况下，在成功发送起始信号后，进入状态 08h，而不是进入 10h。注软件不能解决这类总线问题。

11.4 I2C 控制寄存器

11.4.1 I2C 控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
I2CCR	CR2	I2CEN	ST	STO	SI	AA	CR1	CR0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7，Bit [1:0] **CR[2:0]：** 时钟速度控制位

CR[2:0]	时钟速度控制		CR[2:0]	时钟速度控制	
	Fosc=24M	Fosc=16M		Fosc=24M	Fosc=16M
000	750 Kbit/s	500Kbit/s	100	150 Kbit/s	100Kbit/s
001	600 Kbit/s	400 Kbit/s	101	125 Kbit/s	83.3 Kbit/s
010	400 Kbit/s	266.7 Kbit/s	110	100 Kbit/s	66.7 Kbit/s
011	187.5 Kbit/s	125 Kbit/s	111	75 Kbit/s	50 Kbit/s

Bit 6 **I2CEN：** 模块使能位

0 = 关闭 I2C

1 = 使能 I2C

Bit 5 **ST：** I2C 产生起始信号

0 = 不产生起信号

1 = 总线空闲，将产生起始信号，总线忙，等待总线停止信号后，再产生起始信号。主机模式下，I2C 准备好，在发送或接收一个或多个字节时，将在总线上产生一一个重复起始信号。

注：ST 可在任何时候置 1，但在从机模式下，ST 不能在检测到总线起始信号或重复起始信号后自动由硬件清零，必需由软件清零。

Bit 4 **STO：** I2C 产生停止信号

0 = 不产生停止信号

1 = 主机模式时产生停止信号，当检测到总线上出现停止信号。I2C 硬件清除 STO 标志。STO 标示的设置也用于将 I2C 设备从错误状态（I2CSTA 为 00h）恢复，此条件下，没有停止信号发送 I2C 总线上。若 ST 和 STO 都置 1，且在主机模式下设备为原始的，I2C 总线将产生停止信号并立即伴随着起始信号。如果设备为从机模式，置 STO 恢复到非寻址从机，STO 将会硬件清零。

Bit 3 **SI：** I2C 中断标志位

0 = 未产生中断

1 = 产生中断（状态码 F8h 不会产生此标志）

注：I2C 所有 26 种状态（释放总线 F8H 除外）出现一种，硬件置 1，用户可根据 ST 数值来判断 I2C 执行到哪一步。

Bit 2 **AA：** 应答位

0 = 应答 NACK

1 = 应答 ACK

11.4.2 I2C 状态寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
I2CSTA	I2CSTA7	I2CSTA6	I2CSTA5	I2CSTA4	I2CSTA3	-	-	-
读/写	R	R	R	R	R	-	-	-
复位后	1	1	1	1	1	-	-	-

Bit [7:3] **I2CSTA[7:3]:**

主机模式		从机模式	
状态	**描述**	**状态**	**描述**
0x08	开始信号	0xA0	从机发送重复开始或停止应答信号
0x10	主机重复开始信号	0xA8	从机发送地址应答
0x18	主机发送地址应答	0xB0	从机发送仲裁失败
0x20	主机发送地址无应答	0xB8	从机发送数据应答
0x28	主机发送数据应答	0xC0	从机发送数据无应答
0x30	主机发送数据无应答	0xC8	从机发送最后数据应答
0x38	主机仲裁失败	0x60	从机接收地址应答
0x40	主机接收地址应答	0x68	从机接收仲裁失败
0x48	主机接收地址无应答	0x80	从机接收数据应答
0x50	主机接收数据应答	0x88	从机接收地址无应答
0x58	主机接收数据无应答	0x70	广播呼叫模式地址应答
0x00	总线错误	0x78	广播呼叫模式仲裁失败
-	-	0x90	广播呼叫模式数应答
-	-	0x98	广播呼叫模式数据无应答
0xF8		释放总线	

11.4.3 I2C 数据寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
I2CDATA	I2CDATA[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

11.4.4 I2C 从机地址寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
I2CADR	I2CADR[7:1]							GC
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:1] **I2CADR[7:1]**：I2C从机地址

主机模式：无效

从机模式：存放7位从机地址

Bit 0 **GC：**广播呼叫位

主机模式：无效

从机模式：0 = 广播呼叫模式忽略，不响应

1 = 如果AA置1，参与广播呼叫模式，若AA清零，忽略广播呼叫

12 SPI

12.1 概述

SPI 模块允许在 MCU 和外设(包括其他 MCU)之间进行全双工、同步、串行通信。该模块可以被编程为作为主设备或从设备工作。

- 全双工模式
- 三线同步传输
- 主从模式
- 多个 SPI 波特率
- 可编程极性和相位的串行时钟
- 具有 MCU 中断能力和主模式故障错误标志
- 写入冲突标志保护
- 8 位数据首先传输最高位（MSB），最后传输最低位（LSB）
- 从机模式通过控制端口来控制从机设备

12.2 SPI 框图

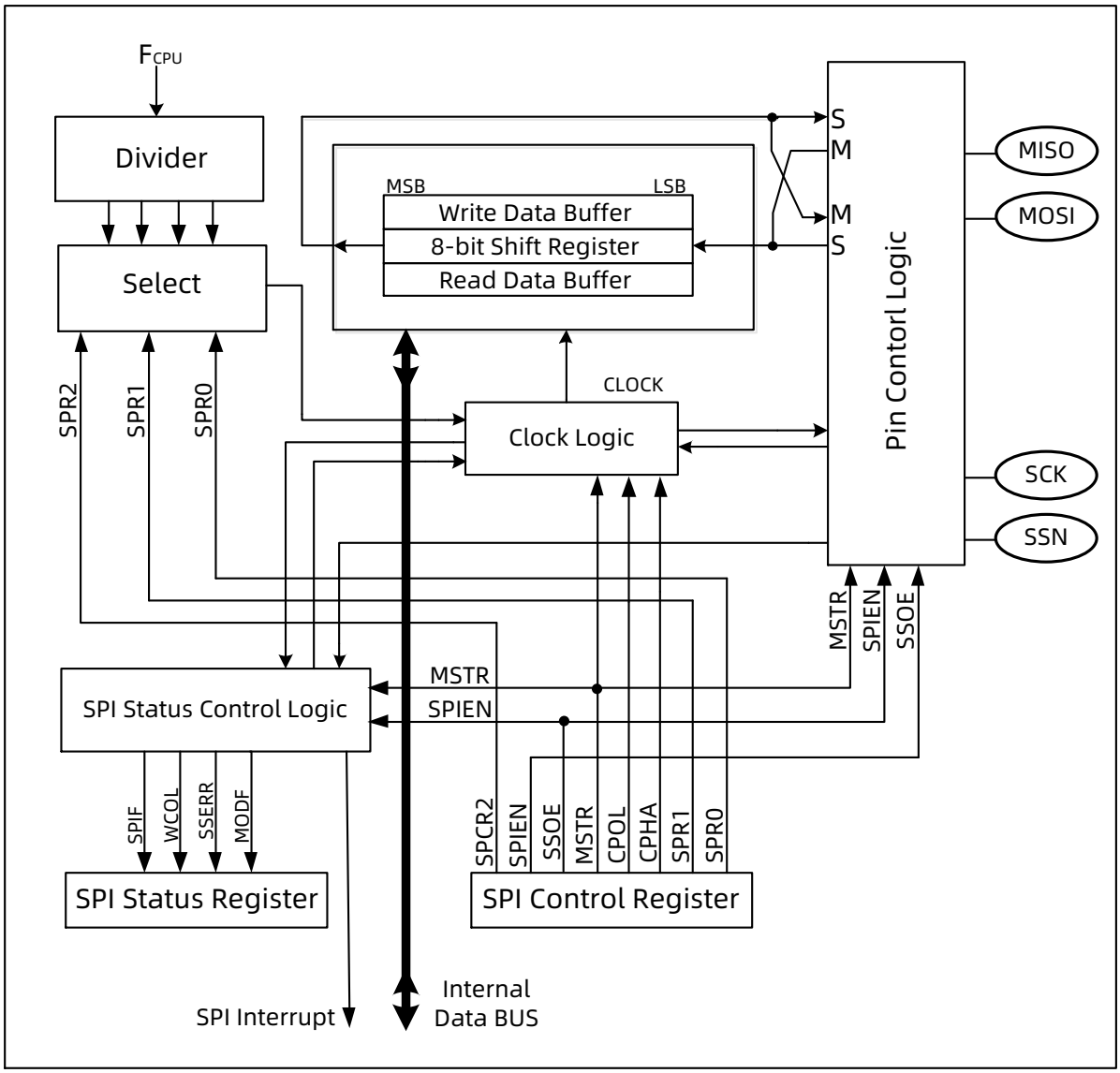


图 12.1　SPI 结构图

12.3 功能描述

SPI 结构图展示了 SPI 的体系结构。SPI 寄存器板块是 SPI 模块的主要组成部分，包括逻辑控制，波特率控制和管脚逻辑控制，SPI 包括移位寄存器和读出数据缓冲器，传送数据是单缓冲器，接收数据是双缓冲器。在传送完成之前传送的数据不能写入移位装置。

SPI 需要四个管脚主进/从出(MISO)，主出/从进(MOSI)，移位时钟(SCK)，和从机选择(SSN)。MOSI 脚用于传输主机到从机的 8 位数据，MOSI 是一个主机设备的输出引脚，从机设备的输入引脚。相应的，MISO 用于接收从机到主机的串行数据。

SCK 引脚为主机模式下的时钟输出，从机模式的时钟输入。移位时钟用于 MOSI 和 MISO 脚之间数据传输的时钟同步。位移时钟由主机输出，一组 SPI 传输系统上只能有一个主机以避免设备冲突。建议该管脚设置为史密特触发输入模式。

每路从机外设通过设定从机选择脚(SSN)使能。当需要读取任何从机时，该信号脚必须保持低。当 SSN 为高，从机读写将被禁止。若为多从机模式，在同一时刻必须只有一个从机保持此低电位，对于主机 SSN 脚不做任何用途，可配置为普通端口另做他用。SSN 可用于多主机模式下错误侦测功能。

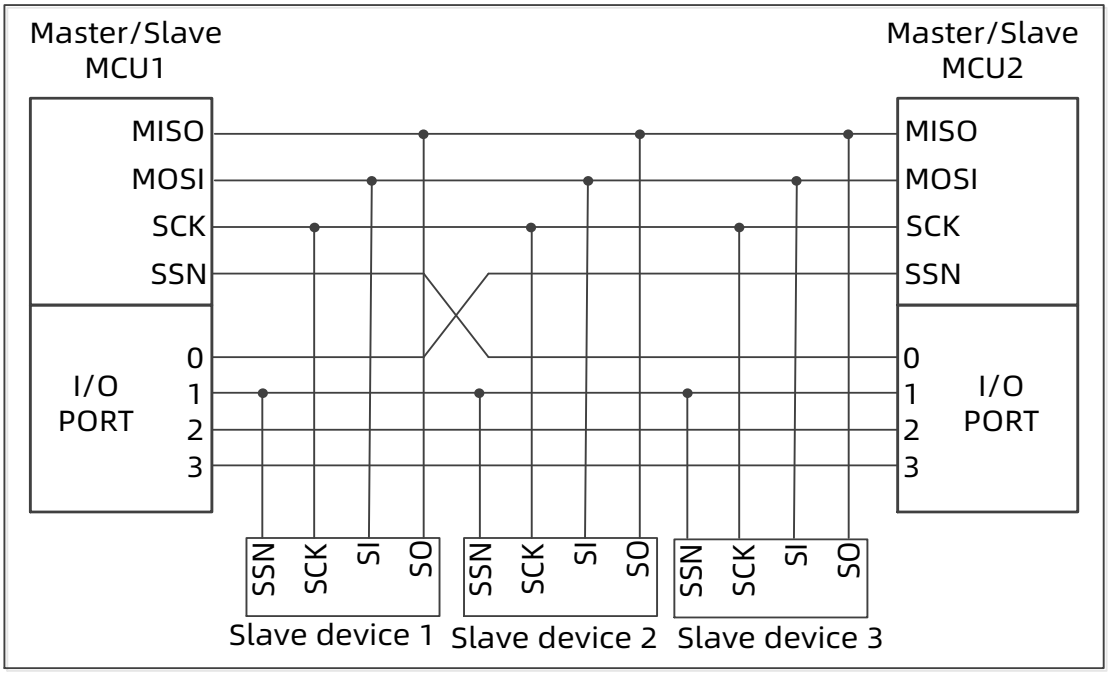


图 12.2-1　SPI 多主机多从机连接图

典型的 SPI 设备通信总线通常为 3 信号线相连,，MOSI ~ MOSI，MISO ~ MISO 和 SCK ~ SCK. 主机通过四线并行连接的方式，每根 SSN 线分别控制每个从机。MCU1 和 MCU2 可以任意定义为主/从机模式. 需配置为主机模式侦测功能避免多主机冲突。

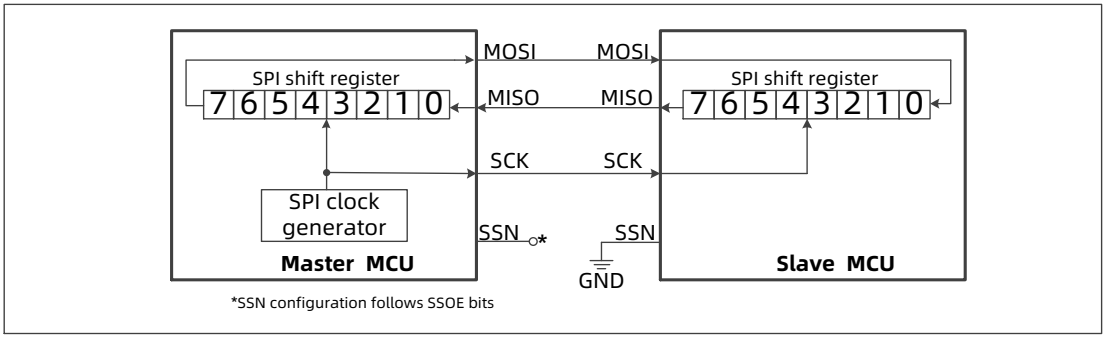


图 12.2-2 SPI 单主机单从机连接图

在传输时，主机通过 MOSI 线向从机发送数据。同时， 主机也通过 MISO 线由从机接收数据。此时主机和从机的两个数据寄存器可被视为一个 16 位的循环位移寄存器。因此，当主机向从机某地址送数据时，从机内该地址内的数据同时也由从机推向主机。传输进行了交换的动作。

12.4 SPI 寄存器

12.4.1 SPI 控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPICR	SPCR2	SPIEN	SSOE	MSTR	CPOL	CPHA	SPCR1	SPCR0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	1	0	0

Bit 7，Bit [1:0] SP**CR[2:0]：** 时钟速度控制位

SP**CR[2:0]**	**时钟速度控制位**
000	$F_{osc}/2$
001	$F_{osc}/4$
010	$F_{osc}/8$
011	$F_{osc}/16$
100	$F_{osc}/32$
101	$F_{osc}/64$
110	$F_{osc}/128$
111	主模式下不产生时钟，当 CPOL 为 1 时，SCKO 输出为高阻态，否则为低电平

Bit 6 **SPIEN：** SPI 使能位
0 = 不使能
1 = 使能

Bit 5 **SSOE：** SSN 控制
0 = SSN 有效
1 = SSN 无效

注：从机模式下，当 CPHA 为 0 时，此位没用。当此位置一，MODF 中断请求就不会产生。

Bit 4 **MSTR：** 主从控制
0 = 作为从机
1 = 作为主机

Bit 3 **CPOL：** 时钟极性选择
0 = 空闲状态 SCK 为低
1 = 空闲状态 SCK 为高

Bit 2 **CPHA：** 时钟相位选择
0 = 第一个时钟沿采样
1 = 第二个时钟沿采样

12.4.2 SPI 状态寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPISTA	SPIF	WCOL	SSERR	MODF	-	-	-	-
读/写	R	R	R	R	-	-	-	-
复位后	0	0	0	0	-	-	-	-

Bit 7 **SPIF：** 数据传输完成标志

0 = 数据传输未完或置 1 后读取 SPIDATA 请 0

1 = 数据传输完成

Bit 6 **WCOL：** 写冲突位

0 = 未发生写冲突事件或置 1 后读取 SPISTA 请 0

1 = 发生写冲突事件

Bit 5 **SSERR：** 从机接收错误标志

0 = 未发生模式错误或 SPI 重新使能

1 = 发生从机接收错误，在从机接收过程中 SSN 被取消，硬件置 1，清除 SPEN，此位才能清除 0

Bit 4 **MODF：** 模式错误标志

0 = 未发生模式错误或置 1 后读取 SPISTA 请 0

1 = 发生模式错误，SSN 电平与 SPI 模式不一致，硬件置 1 且立即切回从机模式

12.4.3 SPI 数据寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPIDATA	SPIDATA[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

12.5 工作模式

12.5.1 主机模式

对 MSTR 位置 1，芯片作为主机模式开始 SPI 传输模块开始工作。整个 SPI 系统中只允许一个主机启动传输。每次传输总是由主机发起，对主机 SPIDAT 寄存器的写开始传送。在 SCK 控制下在 MOSI 管脚传送数据。8 位数据传输完毕，SPIF 由硬件自动置位以示完成一个字节数据传输，同时由从机接收到的数据传送到 SPIDAT。从 SPIDAT 读出数据后，用户才可以清除 SPIF。

12.5.2 从机模式

设定 MSTR 为 0，SPI 将工作在从机模式。当作为从机模式时，SCK 管脚变为输入脚，它将被另外一个主机的 SPI 设备控制，SSN 管脚需要设置为输入，同样，在数据传输完成前保持低电平状态。如果 SSN 变为高电平，SPI 将被迫进入闲置状态。如果 SSN 管脚在传输的过程被置高，那么传输将被取消，同时接受数据的缓存区也将进入闲置状态。

在从机模式下，数据在 MOSI 管脚从主机向从机流动，在 MISO 管脚从从机向主机流动。根据 SCK 的时钟控制数据由主机位移传入，每次一个字节传输完成 SPIF 置 1，此时读取 SPIDATA 寄存器即为该字节内容。对 SPIDATA 的读实际上就是对缓冲器的读。为了防止缓冲器溢出和由于溢出导致的数据丢失，SPIF 必须在数据第二次从移位寄存器向读缓冲器传送前清零。

12.6 时钟格式和数据传输

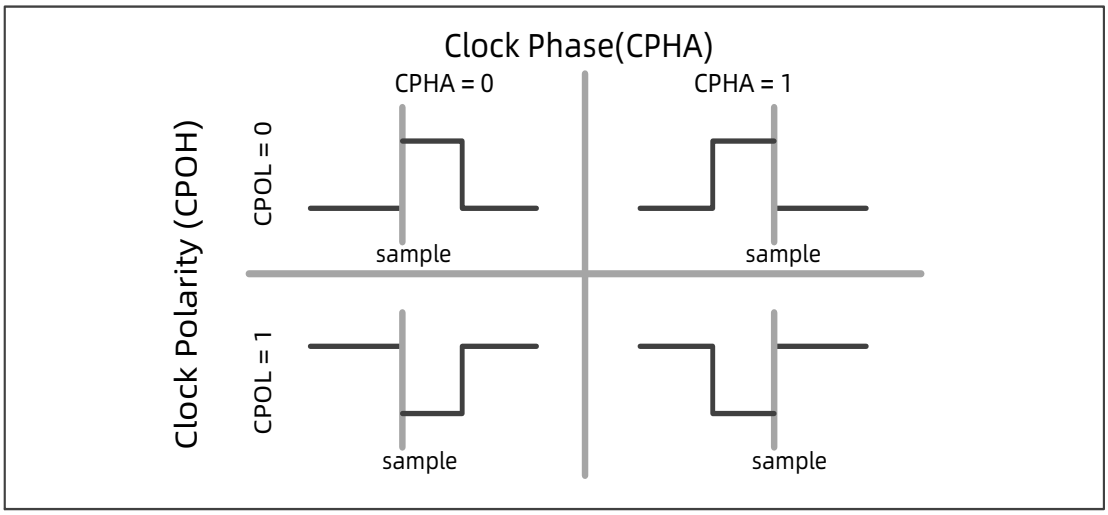


图 12.3 SPI 时钟格式图

为了适应各种各样的同步串行外设，SPI 提供时钟极性位 CPOL 和时钟相位位 CPHA 用以控制。如图 SPI 时钟格式所示，CPOL 和 CPHA 组和出四种不同的时钟格式。CPOL 位表示空闲状态时 SCK 脚电位。CPHA 位表示是由 MOSI 或由 MISO 上那条线的边缘采样。在同一系统上的主从设备中，CPOL 和 CPHA 的应该是相同的，传输不同的数据格式，将产生随机错误结果。

在 SPI 传输中，总是由主机启动传输。如果 SPI 被选定作为主模机式（MSTR = 1）并且打开传输（SPIEN =1），对主机的 SPI 数据寄存器（SPIDATA）写入内容将启动 SPI 时钟和数据传输。传出一个字节的同时会接受一个字节的内容，此后 SPI 时钟停止，主机和从机的 SPIF 同时被置 1。如果 SPI 中断使能位设置为 1，全局中断使能，芯片将执行中断服务程序。

关于从机模式下，SSN 信号需要注意。如时钟格式图所示，CPHA=0 时，第一个 SCK 边沿为 MSB 的采样点。因此，从机必须在 SCK 第一个采样边沿出现之前先把 MSB 传出。SSN 的下降沿可用于准备 MISO 的 MSB。因此，每次成功串行传输一个字节后，该引脚必须切换先高然后低，每个成功逐次串行字节之间。此外，如果从将数据写入 SPI 数据寄存器（SPIDATA）时，如果 SSN 为低电位，则会发生写冲突错

当 CPHA = 1，采样边沿位于 SPCLK 时钟的第二个边沿。从机使用的第一个 SCK 时钟转移的 MSB，而不是 SSN 的下降沿。因此，在每次成功传输时 SSN 可以始终保持低电位保持低之间的转移。此格式更适合单主机单从机的结构使用。从机的 SSN 可以不连接在 SPI 系统中，直接接地。

12.7 模式故障侦测

在一个 SPI 网络中，当不止一个设备有可能成为主机时，为减少数据传输错误，模式故障侦测功能是非常有用的。模式故障侦测发现 SSN 由其它设备拉低，说明系统上有一个从机试图寻找主机地址并把注主机认为成从机。此时，硬件自动将 SPICR 的 MSTR 和 SPIEN 清除，从而 SPI 功能关闭，同时使错误侦测标志 MODF 置 1，如果之前已打开中断，则会进入中断向量。

12.8 写冲突错误

写冲突检测显示当正在进行一次传送时，设备正在试图写数据到 SPIDATA。SPIDATA 在传送方不是双缓冲器，对 SPIDATA 的写被直接写进 SPI 移位寄存器，如果这种写在转移过程中被误用，将发生一个写冲突错误(WCOL 将被置位)。如果转移连续稳定没有受到干扰，那么导致错误的写数据是没有写进移位装置。一次写冲突通常是一个从机错误，原因是当主机开始一次传送时主机知道传送正在进行，所以主机没有理由产生写冲突错误，尽管 SPI 逻辑可以在主机和从机之间进行写冲突检测。WCOL 标志用软件清除。

SPI 是信号对于接收数据，是双向缓冲的。在前一笔数据传输完成之前不能对传输缓冲写入新的数据。否则对 SPIDATA 写入内容会引起写冲突错误。对于发送端 SPIDATA 不是双缓冲的，此时对 SPIDATA 写入数据被直接写入到 SPI 移位寄存器。一旦产生写冲突错误，WCOL 将通过硬件设置为 1 表示写冲突。在这种情况下，当前的数据传输继续其传送，而新的数据将会丢失。虽然 SPI 逻辑可以检测写在主机模式和从机模式的冲突，写冲突通常是从机错误引起的，因为当主机启动传输时，从机并没有指示一个从机没有指示标志。在从机接收模式，对 SPDATA 写入内容，也会引起写冲突错误。

13 触摸按键（CDC）

13.1 概述

18 路触摸按键通道，内置电容无需外接，可替代机械式触摸按键，实现防水防尘，简单易用的操作接口。

注：CDC 功能请使用官网对应的库。

14 模数转换器(ADC)

14.1 概述

M9F6820有18路外部通道（AD0~AD17）和3路内部通道（VDD/4、VREF和GND）12位分辨率的A/D转换器，可以将模拟信号转换成12位数字信号。进行AD转换时，首先要选择输入通道，然后启动AD转换。转换结束后，系统自动将转换结果存入寄存器ADH和ADL中。

注：MCU电源口VDD&GND口并联104电容，104电容位置应紧靠IC，电源走线也应先进入104电容再进入MCU。

14.2 ADCON0 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADEN	ADS	ADFM	CHS[4:0]				
读/写	R/W	R/W	R/W	R/W	RW	R/W	R/W	R/W
复位后	0	0	0	1	0	0	1	1

Bit 7 **ADEN：** ADC使能控制位
0 = 关闭ADC
1 = 使能ADC

Bit 6 **ADS：** ADC 启动位
0 = 停止，转换完成自动清零
1 = 开始（每次写入1将重新启动ADC）

Bit 5 **ADFM：** 数据格式选择位
0 = 左对齐，即ADRES = {ADH[7:0]，ADL[7:4]}；ADL[3:0] = 0
1 = 右对齐，即ADRES = {ADH[3:0]，ADL[7:0]}；ADH[7:4] = 0

Bit [4:0] **CHS[4:0]：** ADC 输入通道选择位
00000 ~ 10001 = AD0 ~ AD17
10010 = VDD/4
10011= VREF
10100= GND
10101 = GND

注：需要采集的外部通道，用户应设置对应的端口为输入模式并且将端口设为模拟端口。

14.3 ADCON1 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	AGEN	ADCKS[2:0]			VREMS[1:0]		[VHS1:0]	
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **AGEN：** 增益控制位

0 = 关闭增益控制

1 = 开启增益控制（VREMS=00时，此位无效）

Bit [6:4] **ADCKS[2:0]：** ADC 时钟源选择位

ADCKS[2:0]	ADC 时钟源选择
000	Fcpu
001	Fcpu/2
010	Fcpu/4
011	Fcpu/8
100	Fcpu/16
101	Fcpu/32
110	Fcpu/64
111	Fcpu/128

Bit [3:2] **VREMS[1:0]：** ADC 参考电压模式选择位

VREMS[1:0]	ADC 参考电压模式
00	VDD
01	内部参考电压
10	外部参考电压
11	内部参考与外部参考连接

B it [7,1:0] **VHS[1:0]：**

AGEN = 0 时为 ADC 内建基准电平选择位

VHS[1:0]	内建 VREF 基准电平
00	关闭内部参考
01	2.0V
10	3.0V
11	4.0V

AGEN = 1 时为增益选择位，将通道信号放大后进行 ADC 转换

VHS[1:0]	放大倍数
00	关闭内部参考
01	11.4 倍
10	
11	

注：对应 VHS[1:0]取值后 VDD 供电应高于选择的内部参考+0.5V 或以上。
例：VHS[1:0] = 11（内部 VREF = 4.0V），则 VDD 要大于 4.5V。

14.4 ADCON2 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON2	-	-	-	-	ADVOS[3:0]			
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

Bit [3:0] **ADVOS[3:0]:** ADC失调补偿寄存器

14.6 ADH/ADL AD 结果寄存器

14.6.1 左对齐，ADFM = 0

ADH 结果寄存器高字节

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADH	ADREG[11:4]							
读/写	R	R	R	R	R	R	R	R
复位后	x	x	x	x	x	x	x	x

Bit [7:0] **ADREG[11:4]：** 12 位 ADC 结果寄存器高 8 位

ADL 结果寄存器低字节

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADL	ADREG[3:0]				-	-	-	-
读/写	R	R	R	R	R	R	R	R
复位后	x	x	x	x	x	x	x	x

Bit [7:4] **ADREG[3:0]：** 12 位 ADC 结果寄存器低 4 位

14.6.2 右对齐，ADFM = 1

ADH 结果寄存器高字节

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADH	-	-	-	-	ADREG[11:8]			
读/写	R	R	R	R	R	R	R	R
复位后	0	0	0	0	0	0	0	0

Bit [3:0] **ADREG[11:8]：** 12 位 ADC 结果寄存器高 4 位

ADL 结果寄存器低字节

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADL	ADREG[7:0]							
读/写	R	R	R	R	R	R	R	R
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **ADREG[7:0]：** 12 位 ADC 结果寄存器低 8 位

14.7 AD 转换时间

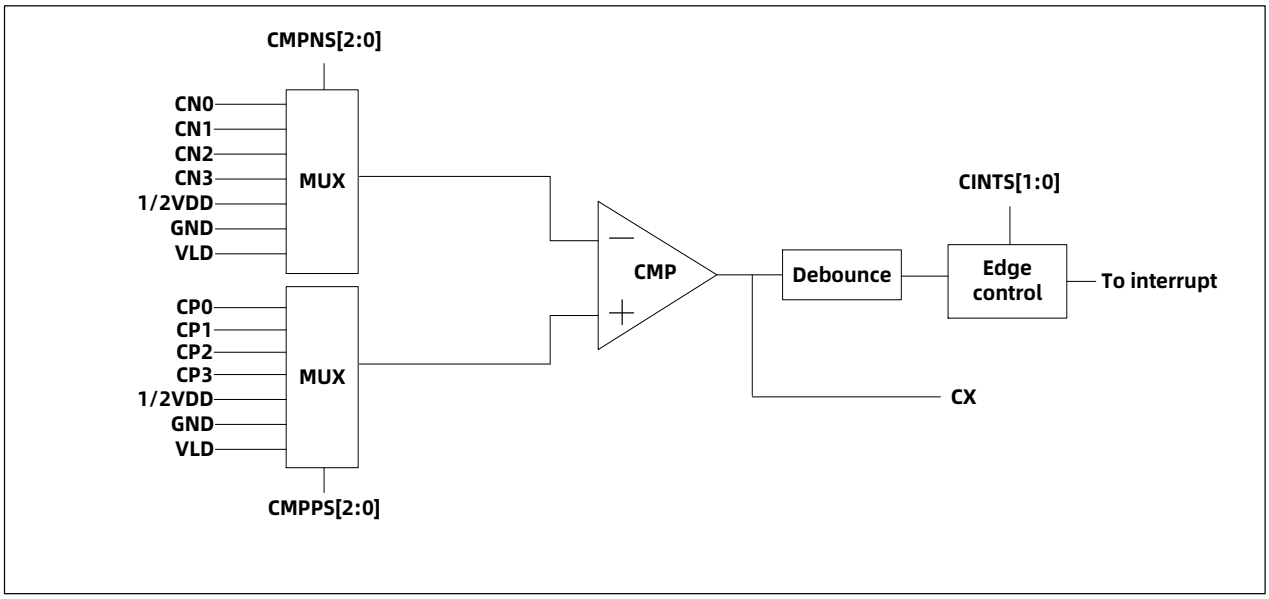
注：12 位 AD 转换时间 = 1/(ADC clock)*16 sec

15 比较器（CMP）

15.1 概述

比较器具有多种输入源、多种参考电压、输出极性可选择、多种输出中断触发和输出信号可唤醒等功能，增强了使用的灵活性，适应各种广泛的应用。

15.2 比较器框图



15.3 CMPC0 比较器控制寄存器 0

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CMPC0	CMPEN	CMPOUT	CMPNS[2:0]			CMPPS[2:0]		
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **CMPEN：** 比较器使能控制位

0 = 关闭比较器

1 = 使能比较器

Bit 6 **CMPOUT：** 比较器输出位

0 = CP脚输入电压小于CN脚

1 = CP脚输入电压大于CN脚

Bit [5:3] **CMPNS[2:0]：** 比较器反相输入信号选择位

CMPNS[2:0]	输入信号选择
000	CN0
001	CN1
010	CN2
011	CN3
100	1/2VDD
101	GND
110	VLD
111	未定义

Bit [2:0] **CMPPS[2:0]：** 比较器正相输入信号选择位

CMPPS[2:0]	输入信号选择
000	CP0
001	CP1
010	CP2
011	CP3
100	1/2VDD
101	GND
110	VLD
111	未定义

注：设计者必须在使能比较器中断之前将比较器使能，以避免未知的中断发生。

15.4 CMPC1 比较器控制寄存器 1

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CMPC1	CMPOEN	C0PVRC	-	CMPVLD[4:0]				
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **CMPOEN：** 比较器输出使能

0 = 关闭比较器信号输出

1 = 比较器信号从端口输出

Bit 6 **C0PVRC：** P端信号输入控制

0 = 无影响

1 = 不论CMPPS为何值，CP0通道导通

Bit 5 **Reserved：** 必需保持为0

Bit [4:0] **CMPVLD[4:0]：** VLD电压选择位

CMPVLD [4:0]	电压（V）	CMPVLD [4:0]	电压（V）	CMPVLD [4:0]	电压（V）	CMPVLD [4:0]	电压（V）
00000	1.25	01000	1.65	10000	2.05	11000	2.45
00001	1.30	01001	1.70	10001	2.10	11001	2.50
00010	1.35	01010	1.75	10010	2.15	11010	2.55
00011	1.40	01011	1.80	10011	2.20	11011	2.60
00100	1.45	01100	1.85	10100	2.25	11100	2.65
00101	1.50	01101	1.90	10101	2.30	11101	2.70
00110	1.55	01110	1.95	10110	2.35	11110	2.75
00111	1.60	01111	2.00	10111	2.40	11111	2.80

15.5 CMPC2 比较器控制寄存器 2

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CMPC2	CINTS[1:0]		-	-	-	DEB[2:0]		
R/W	R/W	R/W	R	R	R	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:6] **CINTS[1:0]：** 比较器中断触发类型选择

CINTS[1:0]	触发类型
00	下降沿触发
01	上升沿触发
1X	双边沿触发

Bit [2:0] **DEB[2:0]：** 比较器逻辑输出滤波设置

DEB[2:0]	滤波设置
000	关闭
001	4
010	8
011	16
100	32
101	64
110	128
111	256

注：

（1）滤波时间为 Tdeb = T_{HIRC} * DEB*3/4。

（2）端口的 Cx 输出不经过滤波。

（3）滤波算法简介：

系统时钟对比较器的输出进行采样，为 1 时滤波计数器加 1。为零时滤波计数器减 1，滤波器初值为设置值的一半，当滤波器计数器>DEB*3/4 时。滤波结果为 1；滤波器计数器<DEB*1/4 时，滤波结果为 0。

16 看门狗（WDT）

16.1 概述

看门狗定时器的时钟为内部独立的低速 RC 时钟源。
芯片配置字中可选择看门狗定时器的工作模式，共三种模式可选择：
（1）始终开启 WDT 功能,即在 STOP 模式下仍然工作，溢出可唤醒 STOP
（2）使能：绿色或休眠模式下关闭，即 STOP 下关闭
（3）屏蔽 WDT 功能，即始终关闭
芯片配置字中可选择看门狗的溢出时间，可选择时间分别为 4.5ms、18ms、72ms 或 288ms。

注：看门狗正常溢出后，程序复位到 0000H，但是在休眠模式下看门狗溢出程序是继续往下运行。

17 芯片配置字（OPTION BIT）

烧录选项	内容	说明
振荡器模式	HIRC(16M)+LIRC	双系统时钟
	HIRC(16M)+LXT	
	HXT+LIRC	
FCPU	2T(Vlvr>3.1V)	高频模式下 CPU 速度选择
	4T(Vlvr>2.1V)	
	8T(Vlvr>1.7V)	
	16T	
	32T	
	64T	
	128T	
外部复位端口选择	作为外部复位端口	
	作为 IO 端口	
启动模式选择	低速启动	
	高速启动	
复位向量	复位向量为 0x0000	
	复位向量为 BOOT	
复位电压选择	LVR=1.4V(FCPU<8T)	系统高速运行时，请选择相应较高的 LVR 电压，以保证系统的可靠性
	LVR=1.6V(FCPU<8T)	
	LVR=1.8V(FCPU<4T)	
	LVR=2.0V(FCPU<4T)	
	LVR=2.1V(FCPU<4T)	
	LVR=2.2V(FCPU<2T)	
	LVR=2.4V(FCPU<2T)	
	LVR=2.5V(FCPU<2T)	
	LVR=2.7V(FCPU<2T)	
	LVR=2.8V(FCPU<2T)	
	LVR=3.0V(FCPU<2T)	
	LVR=3.1V(FCPU<2T)	
	LVR=3.3V	
	LVR=3.5V	
	LVR=3.6V	

烧录选项	内容	说明
中断向量选择	唯一中断向量 0x0008	
	双中断向量	
WDT 使能选择	屏蔽 WDT 功能	
	使能，绿色或休眠模式下关闭	
	始终开启 WDT 功能	
WDT 溢出时间	4.5ms	VDD=5V 典型值
	18ms	
	72ms	
	288ms	
APP 写允许	不允许 APP 写入	
	允许 APP 写入	
APP 读取 BOOT 区	不允许 APP 读取 BOOT 区	
	允许 APP 读取 BOOT 区	
PC 运行在类 EE 是否复位	PC 可以在类 EE 运行	
	PC 不能运行在类 EE	
PC 出 BOOT 后是否复位	不复位	
	复位	
类 EE 区域大小	无类 EE	类 EE 区起始地址=0x4000-BOOT 区大小-类 EE 区大小 例：类 EE 区=2KB，BOOT 区=1KB 类 EE 区起始地址=0x4000-0x400*2-0x400*1=0x3400 BOOT 区起始地址=0x3C00
	0.5KB	
	1KB	
	1.5KB	
	2KB	
	2.5KB	
	3KB	
	3.5KB	
	4KB	
BOOT 区域大小	无 BOOT 区	括号中为 BOOT 区起始地址
	0.5KB(0x3E00)	
	1KB(0x3C00)	
	1.5KB(0x3A00)	
	2KB(0x3800)	
晶振驱动选择	低驱动	
	高驱动	
低频晶振供电选择	芯片 VDD 供电	
	VCRY 供电	
	BIAS 供电	
芯片代码加密	不使能	
	使能	

18 电性参数

18.1 极限参数

储存温度.....-50℃ ~ 125℃
工作温度.....-40℃ ~ 85℃
电源供应电压.....0V ~ 5.5V
端口输入电压.....GND-0.3V ~ VDD+0.3V

注：如果器件工作条件超出上述极限参数，将造成器件永久性破坏。如果在极限参数最大值上长时间工作，器件稳定性会受到影响。为保障器件稳定运行请在规定范围内工作。

18.2 直流特性

符号	参数	测试条件		最小值	典型值	最大值	单位
		VDD	条件（常温 25℃）				
VDD	工作电压	—	Fosc = 16MHz，16T	1.4	-	5.5	V
I_{DD1}	工作电流 1	3V	高频运行（HIRC=16M） 低频运行（LIRC=48K） F_{CPU}=HIRC/16T 全速工作	-	0.3	-	mA
		5V		-	0.5	-	
I_{DD2}	工作电流 2	3V	高频运行（HIRC=24M） 低频运行（LIRC=48K） F_{CPU}=HIRC/16T 全速工作	-	0.5	-	mA
		5V		-	0.7	-	
I_{DD3}	工作电流 3	3V	高频停止 低频运行（LIRC=48K）	-	12	-	μA
		5V		-	17	-	
I_{SP1}	静态电流 1	3V	高频运行（HIRC=16M） 低频运行（LIRC=48K） STOP =1 无唤醒源	-	0.2	-	mA
		5V		-	0.3	-	
I_{SP2}	静态电流 2	3V	高频停止 低频运行（LIRC=48K） STOP =1 无唤醒源	-	3	-	μA
		5V		-	8	-	
I_{SP3}	静态电流 3	3V	高频停止 低频停止 STOP =1 无唤醒源	-	1.0	-	μA
		5V		-	2.5	-	

符号	参数	测试条件		最小值	典型值	最大值	单位
		VDD	条件（常温 25℃）				
I_{SP4}	静态电流 4	3V	高频停止 外部低频运行 (LXT=32768HZ) STOP=1 无唤醒源	-	3.0	-	μA
		5V		-	6.5	-	
V_{IL1}	输入低电平	3V	SMT	0	-	0.2VDD	V
V_{IH1}	输入高电平	3V		0.8VDD	-	VDD	
V_{IL2}	输入低电平	5V		0	-	0.2VDD	
V_{IH2}	输入高电平	5V		0.8VDD	-	VDD	
V_{IL3}	输入低电平	3V	1/2VDD	0	-	0.4VDD	
V_{IH3}	输入高电平	3V		0.6VDD	-	VDD	
V_{IL4}	输入低电平	5V		0	-	0.4VDD	
V_{IH4}	输入高电平	5V		0.6VDD	-	VDD	
V_{IL5}	输入低电平	3V	1.5V	0	-	0.9	
V_{IH5}	输入高电平	3V		1.3	-	VDD	
V_{IL6}	输入低电平	5V		0	-	1.3	
V_{IH6}	输入高电平	5V		1.7	-	VDD	
R_{PH}	上拉电阻	5V	V_{IN} = GND	-	15	-	KΩ
		3V	V_{IN} = GND	-	25	-	
R_{PL}	下拉电阻	5V	V_{IN} = VDD	-	40	-	
		3V	V_{IN} = VDD	-	71	-	
I_{OL1}	输出灌电流	5V	输出口，Vout=GND+0.6V (IOA/ IOB/IOC [0:1])		43		mA
		3V			27		
I_{OL2}	输出灌电流	5V			37		
		3V			23		
I_{OL3}	输出灌电流	5V			12		
		3V			8		
I_{OL4}	输出灌电流	5V			4		
		3V			2.5		
I_{OH1}	输出拉电流	5V	输出口，Vout=VDD-0.6V (IOA/ IOB/IOC [0:1])		29		
		3V			20		
I_{OH2}	输出拉电流	5V			25		
		3V			15		
I_{OH3}	输出拉电流	5V			9		
		3V			6		
I_{OH4}	输出拉电流	5V			4.5		
		3V			3		

18.3 交流特性

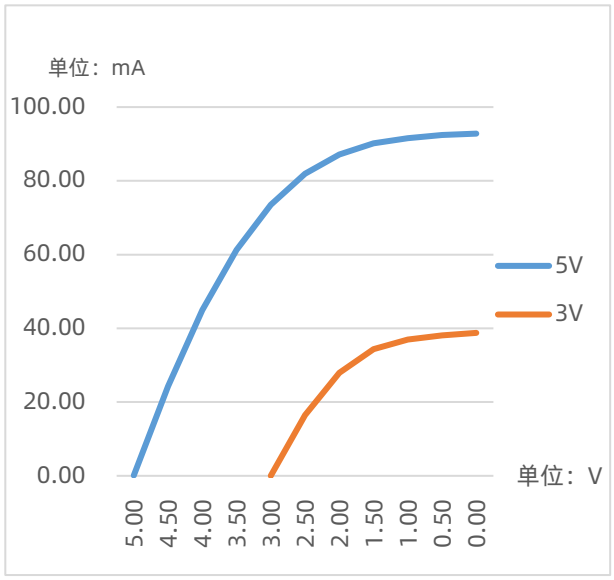
常温(25℃)下，测试了 5 种不同等效串联电阻（ESR）类型的 32.768KHZ 晶振，在以下推荐的两种晶振配置字下所对应的不同负载电容的起振时间、过秒和电流数据：

晶振配置字参数	测试条件			起振时间 (mS)	过秒 (M/S)	电流 (uA)
	等效串联电阻 ESR 类型(KΩ)	负载电容（pF）	VDD			
晶振驱动选择:高驱动 供电选择:芯片 VDD 供电	30	15	5V	140	152	8.7
			3V	250	26	3.4
	40	15	5V	150	77	9.5
			3V	270	15	3.6
	50	15	5V	180	80	10
			3V	340	19	3.8
	60	24	5V	300	56	10.8
			3V	710	10	4.3
	70	24	5V	250	59	10.8
			3V	520	15	4.2
晶振驱动选择:高驱动 供电选择:BIAS 供电	30	15	5V	940	23	7.8
			3V	1000	21	6.6
	40	15	5V	1100	10	7.8
			3V	1200	9	6.8
	50	15	5V	1500	5	8.1
			3V	1800	3	7.1
	60	24	5V	2800	-10	8
			3V	2800	-11	6.9
	70	24	5V	2900	-4	7.6
			3V	2900	7	6.4

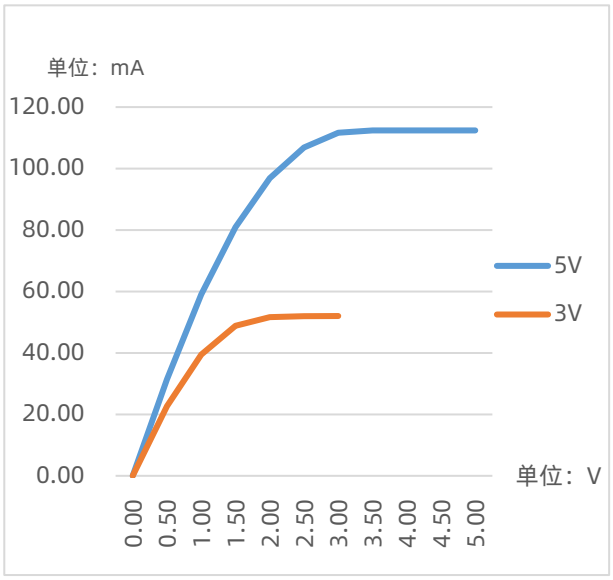
注：（1）晶振供电选择为 VDD 供电时起振时间会偏小，但是过秒会随电压变化而变化。
晶振供电选择为 BIAS 供电时起振时间会偏大，但是过秒不会随电压变化而变化。
（2）高驱动：可以快速起振，但消耗更多电流。
低驱动：功耗低，但起振时间较长。
（3）数据仅供参考，具体值不做设计保证。

18.4 IO 口拉灌电流特性曲线

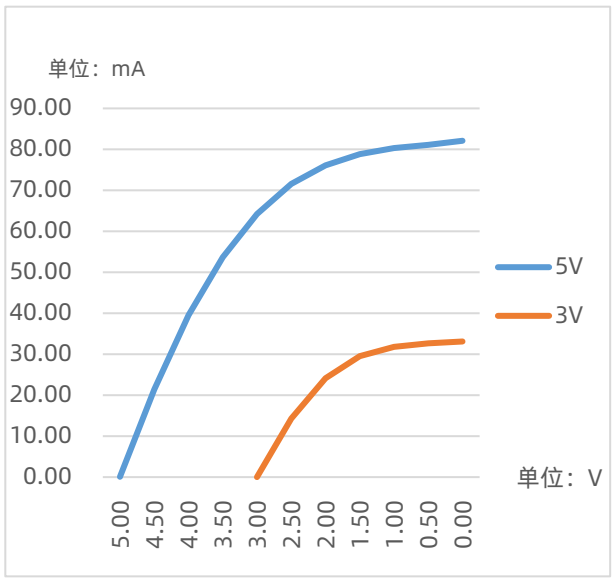
IOH_1



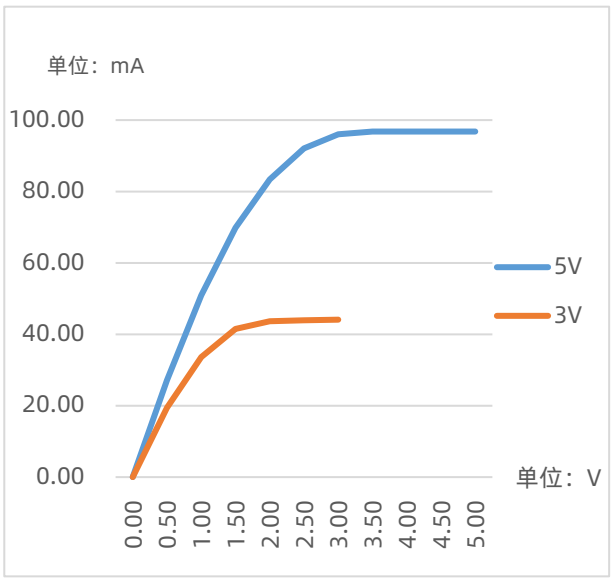
IOL_1



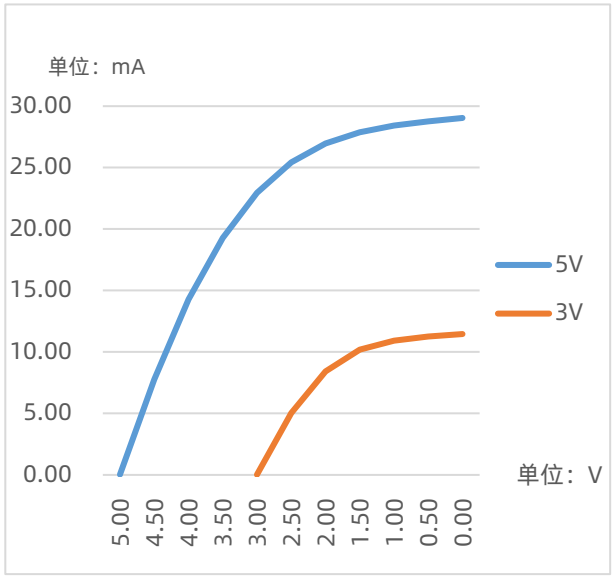
IOH_2



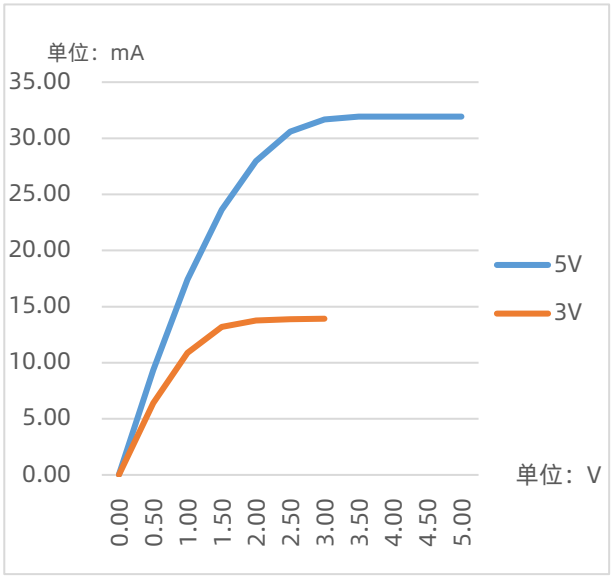
IOL_2



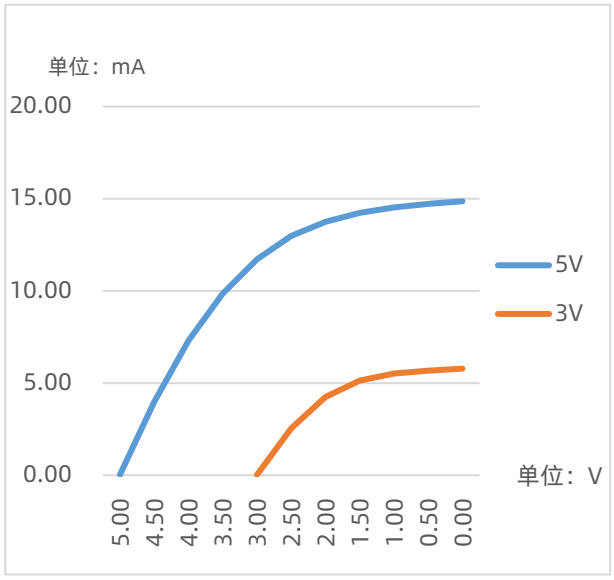
IOH_3



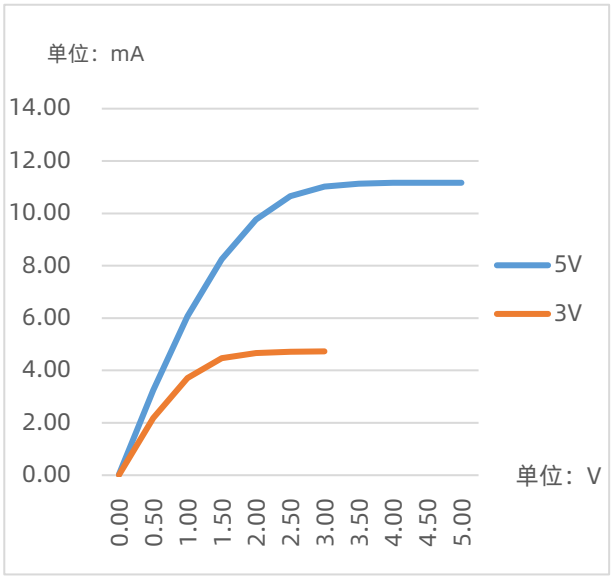
IOL_3



IOH_4

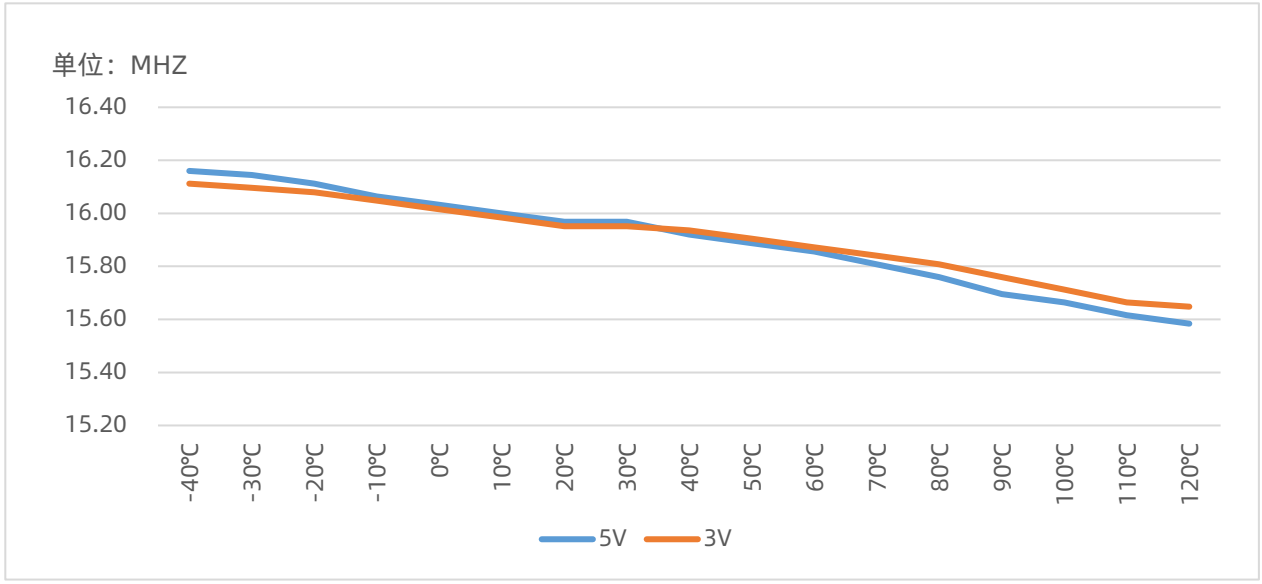


IOL_4

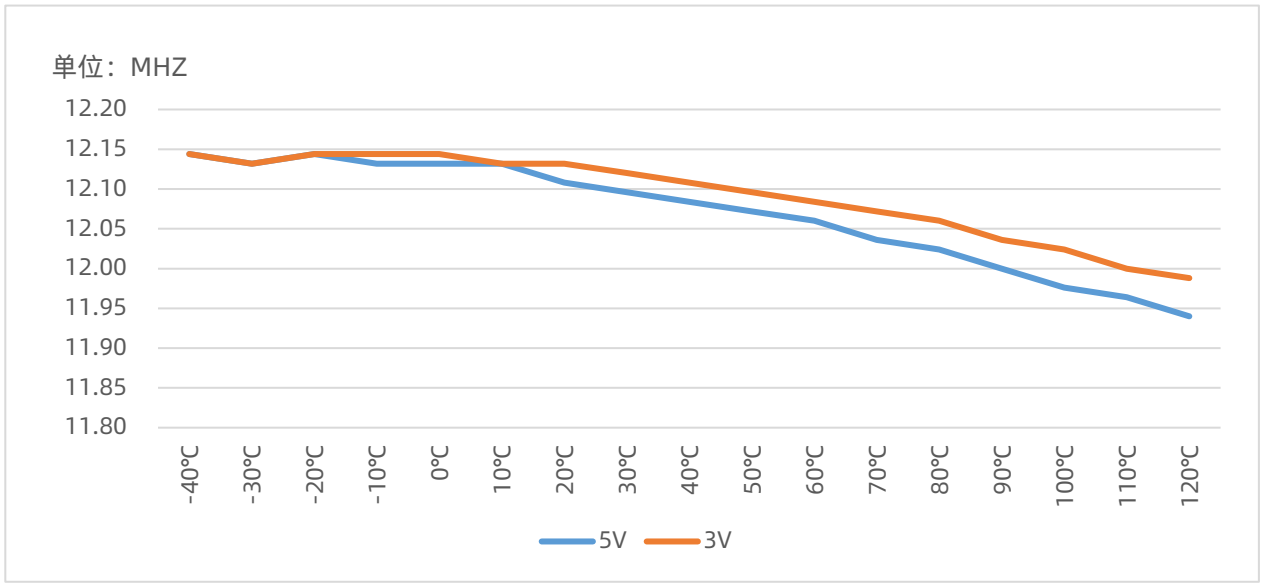


18.5 系统时钟特性

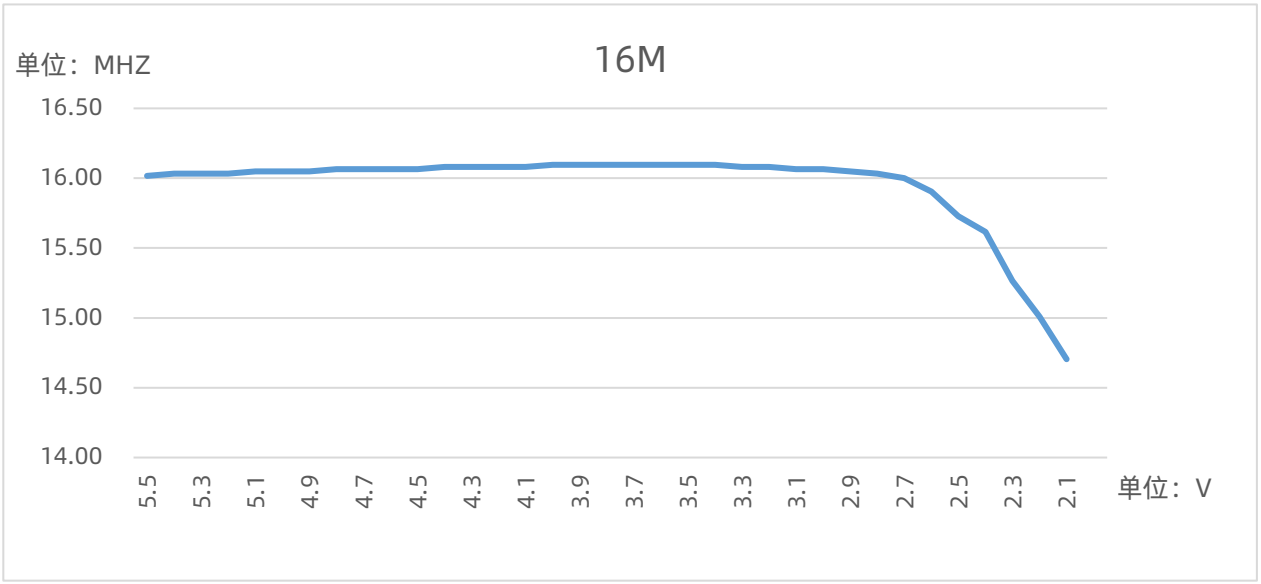
高频时钟（16MHZ）随温度变化典型曲线



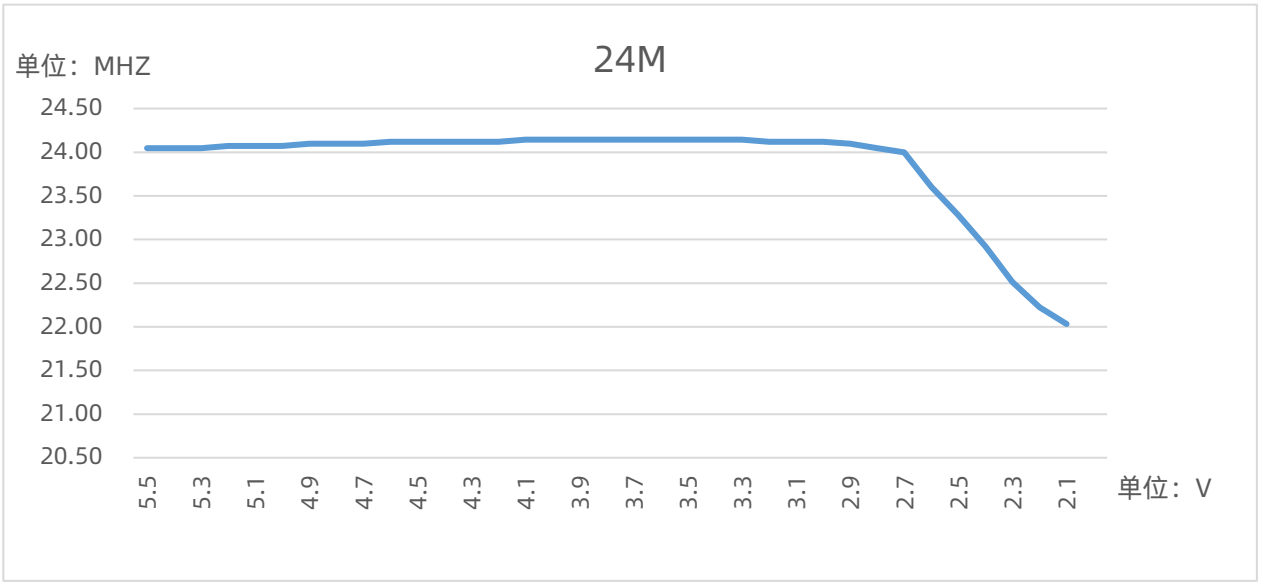
高频时钟（24MHZ）随温度变化典型曲线



常温下（25℃），高频时钟（16MHZ）随电压变化典型曲线

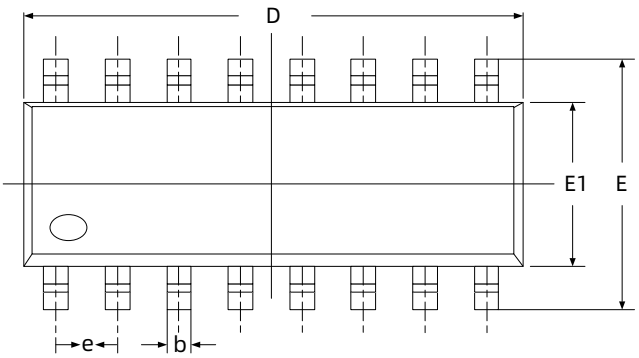


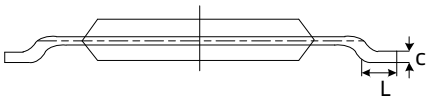
常温下（25℃），高频时钟（24MHZ）随电压变化典型曲线

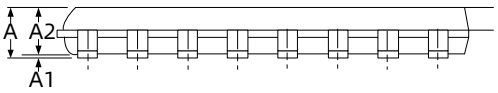


19 封装信息

19.1 SOP16

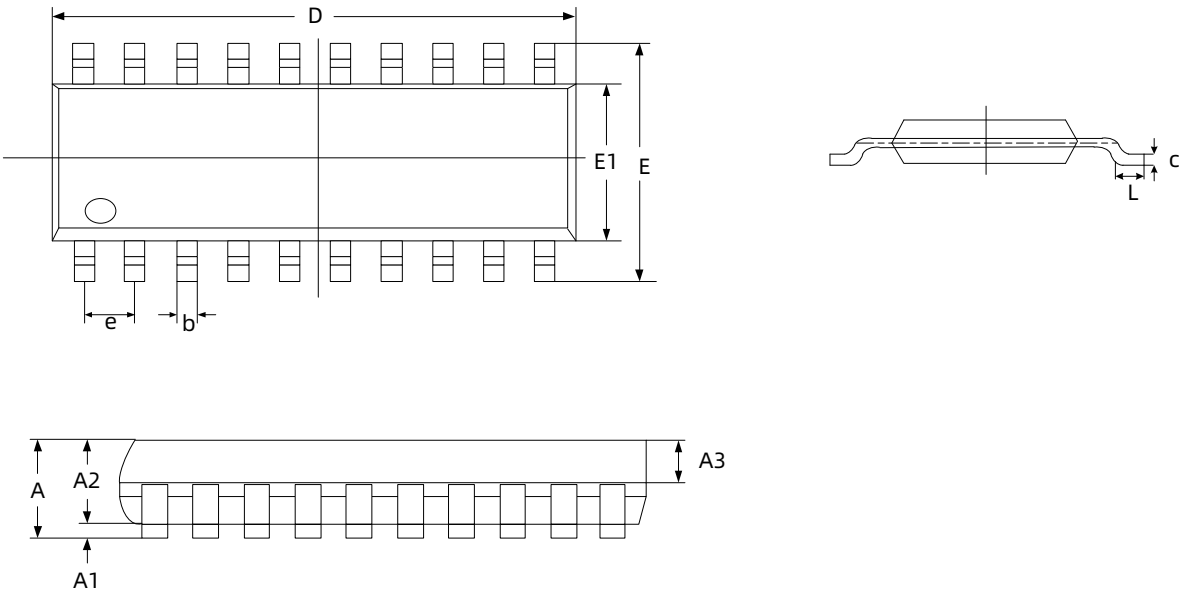






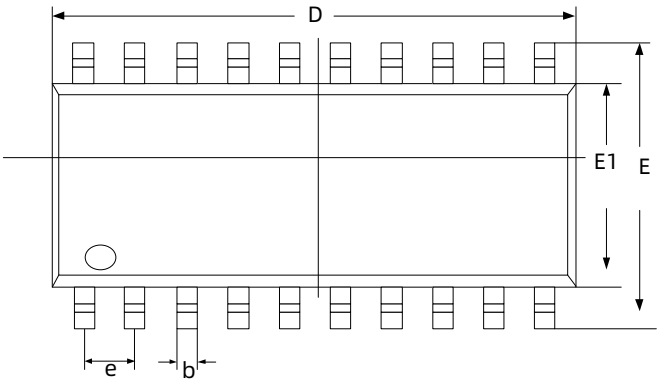
符号	单位（mm）		
	最小	正常	最大
A	1.35	-	1.75
A1	-	-	0.25
A2	1.35	1.40	1.50
b	0.30	-	0.51
D	9.75	9.90	10.05
e	1.27BSC		
E1	3.70	3.90	4.10
E	5.75	6.00	6.25
L	0.40	0.60	0.80
c	-	0.20	-

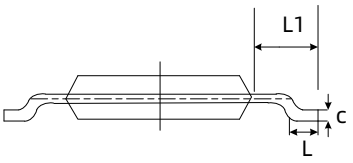
19.2 SOP20

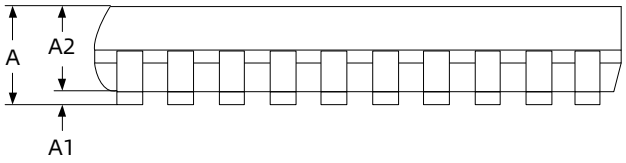


符号	单位（mm）		
	最小	正常	最大
A	2.25	-	2.65
A1	-	-	0.30
A2	2.25	2.30	2.35
b	0.30	-	0.51
D	12.50	12.80	13.00
e	1.27BSC		
E	10.00	10.30	10.60
E1	7.40	7.50	7.60
L	0.50	0.80	1.10
c	0.20	-	0.30

19.3 TSSOP20







符号	单位（mm）		
	最小	正常	最大
A	-	-	1.20
A1	-	-	0.15
A2	0.80	0.95	1.05
b	0.18	-	0.30
D	6.35	6.45	6.55
e	0.65BSC		
E	6.30	6.400	6.60
E1	4.20	4.35	4.50
L	0.45	0.60	0.75
L1	1.00REF		
c	0.09	-	0.20

20 指令集简述

20.1 概述

M9指令集是一种精简指令集（RISC），指令宽度为16位，由操作码和0~2个操作数组成。指令按照功能可分为5类：字节操作类指令、位操作类指令、控制操作类指令、立即数操作类指令和存储器操作类指令。

一个指令周期由 1 个系统时钟周期组成,除非条件测试结果为真或指令执行改变了程序计数器的值，否则执行所有的指令都只需要一个指令周期。对于上诉两种特征情况，指令执行需要两个指令周期。

任何一条指定文件寄存器，作为指令的一部分都会进行读-修改-写操作。读寄存器、修改数据并根据指令或目标寄存器标识符“d”存储结果。即使是写寄存器的指令也将先对改寄存器进行读操作。

20.2 符号说明

符号	范围	说明	符号	范围	说明
R/r	000h-fffh	8 位文件寄存器地址	PC	-	PC 指针
Rs	000h-fffh	12 位文件寄存器源地址	C	-	进位标志
Rd	000h-fffh	12 位文件寄存器目标地址	DC	-	半进位标志
A	-	AREG 寄存器	DC	-	半进位标志
a		快速操作 RAM 位	Z	-	零标志
BSR		存储选择寄存器	OV	-	溢出标志
b	0-7	位地址	N	-	负标志
K/k		立即数、常数或标号	d	0-1	目的操作数定义
TOS	-	栈顶	TO	-	超时位
Label	-	标号名称	PD	-	掉电位
s	-	快速调用选择位	GIE	-	总中断使能位

20.3 M9 指令集表

指令集表中：d=1，目的操作数为 R；d=0，目的操作数为 A。

指令类型	助记符	指令说明	周期数	影响标志位	备注
字节操作类指令	ADDAR R, d, a	A 与 R 相加	1	C, DC, Z, OV, N	1, 2
	ADCAR R, d, a	A 带进位与 R 相加	1	C, DC, Z, OV, N	1, 2
	ANDAR R, d, a	A 和 R 作逻辑与运算	1	Z, N	1, 2
	CLRR R, a	将 R 清零	1	Z	2
	COMR R, d, a	对 R 取反	1	Z, N	1, 2
	JERA R, a	R 与 A 比较，相等跳过	1（2 或 3）	-	3
	JGRA R, a	R 与 A 比较，大于跳过	1（2 或 3）	-	3
	JLRA R, a	R 与 A 比较，小于跳过	1（2 或 3）	-	1
	DECR R, d, a	R 递减 1	1	C, DC, Z, OV, N	1, 2, 3
	DJZR R, d, a	R 递减 1，为 0 跳过	1（2 或 3）	-	1, 2, 3
	DJNZR R, d, a	R 递减 1，非 0 跳过	1（2 或 3）	-	1
	INCR R, d, a	R 递增 1	1	C, DC, Z, OV, N	1, 2, 3
	JZR R, d, a	R 递增 1，为 0 跳过	1（2 或 3）	-	3
	JNZR R, d, a	R 递增 1，非 0 跳过	1（2 或 3）	-	1
	ORAR R, d, a	A 与 R 作逻辑或运算	1	Z, N	1
	MOVR R, d, a	传送 R	1	Z, N	1
	MOVRR Rs,Rd	从源 Rs 送到目标 Rd	2	-	
	MOVAR R,a	将 A 中的内容送入 R	1	-	
	MULAR R,a	A 与 R 相乘	1	-	1
	NEGR R,a	对 R 取补	1	C, DC, Z, OV, N	
	RLR R, d, a	R 带进位循环左移	1	C, Z, N	1
	RLNCR R, d, a	R 循环左移（无进位）	1	Z, N	
	RRR R, d, a	R 带进位循环右移	1	C, Z, N	
	RRNCR R, d, a	R 循环右移（无进位）	1	Z, N	
	SETR R,a	将 R 全置为 1	1	-	1
	SBCAR R, d, a	A 减去 R（带借位）	1	C, DC, Z, OV, N	
	SUBRA R, d, a	R 减去 A	1	C, DC, Z, OV, N	1
	SBCRA R, d, a	R 减去 A（带借位）	1	C, DC, Z, OV, N	
	SWAPR R, d, a	对 R 进行半字节交换	1	-	3
	JREZ R,a	测试 R，为 0 则跳过	1（2 或 3）	-	1
	XORAR R, d, a	A 和 R 作逻辑异或运算	1	Z, N	
位操作指令	BCLR R, b, a	将 R 中的指定位清零	1	-	1
	BSET R, b, a	将 R 中的指定位置 1	1	-	1
	JBTS0 R, b, a	测试 R 的位，为 0 跳过	1（2 或 3）	-	2, 3
	JBTS1 R, b, a	测试 R 的位，为 1 跳过	1（2 或 3）	-	2, 3
	BNEG R, b, a	将 R 中的指定位取反	1	-	1

指令类型	助记符	指令说明	周期数	影响标志位	备注
控制操作类指令	RJBC Label	如果有进位则跳转	1（2）	-	
	RJBN Label	如果为负则跳转	1（2）	-	
	RJBNC Label	如果没有进位则跳转	1（2）	-	
	RJBNN Label	如果不为负则跳转	1（2）	-	
	RJBNOV Label	如果未溢出则跳转	1（2）	-	
	RJBNZ Label	如果不为零则跳转	1（2）	-	
	RJBOV Label	如果溢出则跳转	1（2）	-	
	RGOTO Label	无条件跳转	2	-	
	RJBZ Label	如果为零则跳转	1（2）	-	
	CALL k, s	调用子程序	2	-	
	CLRWDT	清除 WDT 寄存器	1	PD，TO	
	DAA	对 A 进行十进制调整	1	C	
	GOTO k	跳转到地址	2	-	
	NOP	无操作	1	-	
	NOP	无操作	1	-	3
	SPOP	弹出堆栈栈顶 TOS	1	-	
	SPUSH	压入堆栈栈顶 TOS	1	-	
	RCALL Label	相对调用	2	-	
	SRESET	软件器件复位	1	所有	
	RETIE s	中断返回并允许中断	2	GIE/GIEH	
	RETIA k	返回并将 k 送入 A	2	-	
	RETURN s	从子程序返回	2	-	
立即数操作指令	ADDIA k	A 与立即数 k 相加	1	C, DC, Z, OV, N	
	ANDIA k	立即数 k 和 A 作逻辑与运算	1	Z, N	
	ORIA k	立即数 k 和 A 作逻辑或运算	1	Z, N	
	LDFSR R,k	传送立即数 k 到 FSR	2	-	
	BANKBSR k	将立即数 k 送入 BSR	1	-	
	MOVIA k	将立即数 k 送入 A	1	-	
	MULIA k	立即数 k 与 A 相乘	1	-	
	SUBIA k	立即数 k 减去 A	1	C, DC, Z, OV, N	
	XORIA k	立即数 k 与 A 作逻辑异或运算	1	Z, N	
存储器操作指令	RDT*	表读	2	-	
	RDT*+	表读，后递增		-	
	RDT*-	表读，后递减		-	
	RDT+*	表读，预递增		-	
	WDT*	表写	2	-	

注：（1）当端口寄存器修改自身时，修改时使用的值是引脚上的当前值。例如，如果将一引脚配置为输入，其对应数据锁存器中的值将为 1，但此时若有外部器件将该引脚驱动为低电平，则被写回数据锁存器的数据值将是 0。

（2）如果程序计数器（PC）被修改或者条件测试为真，则该指令需要两个周期。第二个周期执行一条 NOP 指令。

21 修正记录

版本	日期	描述
V1.00	2023-12-11	初版
---	---	---
V1.07	2026-05-11	优化封装信息参数、添加交流特性、添加 TSSOP20 封装