M9F6601

8BIT
TK+AD 型
FLASH MCU

Version 1.01

2025 年 8 月

磐 芯 电 子

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，64K*8 位 Flash 程序存储器，3680*8 位 RAM，最多 46 个双向 I/O 口，1 个 8 位 Timer 定时器/计数器，4 个 16 位 Timer 定时器/计数器，6 路 USART，1 路 SPI，1 路 IIC，1 路 8 位 PWM，2 路 16 位独立互补 PWM 时基 Timer1，6 路 16 位独立互补 EPWM， 18+3 路 12 位 AD 转换器，31 路触摸，LCD/LED 模块，比较器模块，循环冗余校验单元 CRC，支持多种系统工作模式和多个中断源。

1.1 特性

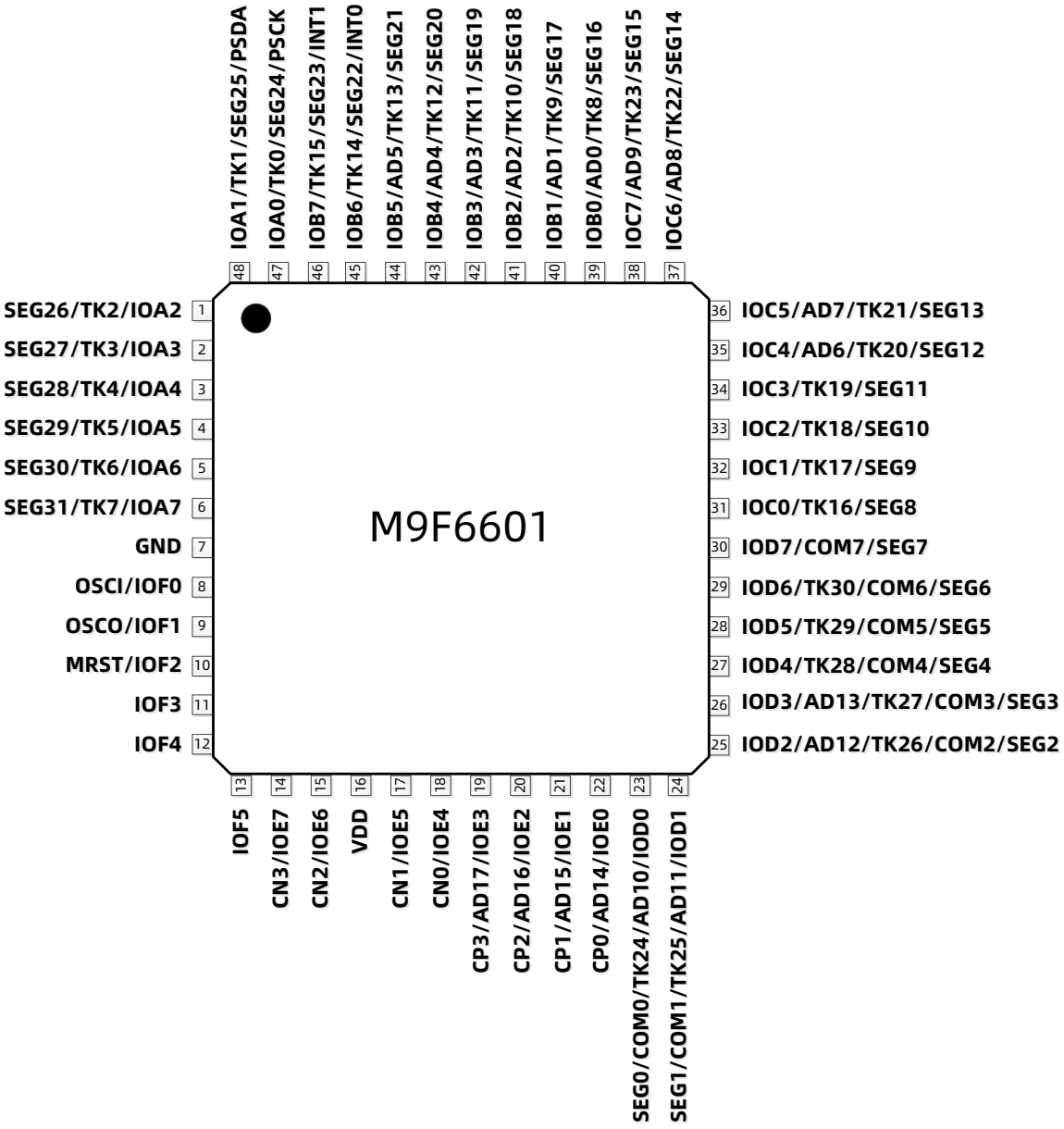
- CPU 特性
  - 高性能精简指令
  - 高速CPU
  - 31级堆栈缓存器
  - 支持查表指令
- ROM
  - 64K*8位的Flash程序存储器
  - 可编程代码保护(可分区设置)
  - IAP功能
- RAM
  - 3680*8位的数据存储器
- I/O 口
  - 最多46个双向I/O口
  - 所有端口可编程弱上拉、下拉
  - 所有端口电平变化中断
- 多路中断
  - 2个中断向量与2级优先级可选
  - 5个定时器中断
  - 2路可选电平外部中断
  - 支持所有端口电平变化中断
  - 其它外设中断
- 定时器 TC0
  - 8位计数/16位计数
  - PWM0：1路8位分辨率
  - PWM0：任意映射
  - 1-8倍后分频
- 系统时钟
  - 内部高速RC振荡器： 32MHz
  - 内部低速RC振荡器： 64KHz
  - 外部高速晶体振荡器：4-24MHz
  - 外部低速晶体振荡器：32.768KHz
- 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：可映射到任意端口
- 定时器 TC2/3/4
  - 16位计数
  - 门控功能
- EPWM 模块
  - 16位分辨率
  - 向上、向下、中心对齐、非中心对齐计数模式
  - 可设置成6路独立，3组互补PWM，3组同步
  - 可映射到任意端口
- USART0/1/2/3/4/5
  - 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）
  - 快速模式（400kbps）
  - 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
- 16 位硬件乘除法器
  - 16位x16位乘法运算
  - 32位/16位除法运算
- LCD
  - 最多8x24点
  - 2-8 COM任意选择
  - 电阻模式（可设VLCD电压）
  - 支持1/4、1/3、1/2偏置电压
- LED
  - 最多8x7+8x24点
  - 2-8 COM任意选择
  - LED支持恒流源驱动
  - 带调光LED驱动
- 看门狗定时器
- 循环冗余校验单元
  - 可自定义初始值，输入输出是否反转
  - 两个系统时钟输出校验值
- CDC 触摸按键模块
  - 最多 31 路通道
  - 无需外接电容
- 封装形式
  - LQFP48

1.2 引脚图

注：芯片仿真烧录口分别是VDD、PSDA、PSCK、GND。

1.2.1 LQFP48



注：（1）引脚图中未注明的 UASRT、I2C、SPI、PWM0、PWM1x、EPWMx、FBx、ADET 与 TxG 相关功能口可映射到任意端口上。
（2）SPI 只能整体在选择在端口高或低位上。
（3）以上可选端口功能控制介绍，详见端口 6.9 章节。

1.3 封装类型与引脚功能映射表

PACKAGE	IO	LED/LCD	ADC	Touch	Other Interface
LQFP48	46	$24_{SEG}X8_{COM}$	18+3	31+1	USART/I^2C/SPI/CMP

1.4 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
TK0	A	触摸按键通道 0
SEG24	O	LCD/LED 显示 Segment 信号输出引脚
IOA1	I/O	输入/输出 IO、SMT、上拉/下拉电阻
PSDA	I/O	编程/仿真串行数据
TK1	A	触摸按键通道 1
SEG25	O	LCD/LED 显示 Segment 信号输出引脚
IOA2	I/O	输入/输出 IO、SMT、上拉/下拉电阻
TK2	A	触摸按键通道 2
SEG26	O	LCD/LED 显示 Segment 信号输出引脚
IOA3	I/O	输入/输出 IO、SMT、上拉/下拉电阻
TK3	A	触摸按键通道 3
SEG27	O	LCD/LED 显示 Segment 信号输出引脚
IOA4	I/O	输入/输出 IO、SMT、上拉/下拉电阻
TK4	A	触摸按键通道 4
T0CKI	I	TC0 外部时钟输入
SEG28	O	LCD/LED 显示 Segment 信号输出引脚
IOA5	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
TK5	A	触摸按键通道 5
T1CKI	I	TC1 外部时钟输入
VREF	A	AD 外部参考电压输入
SEG29	O	LCD/LED 显示 Segment 信号输出引脚
IOA6	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
TK6	A	触摸按键通道 6
T2CKI	I	TC2 外部时钟输入
SEG30	O	LCD/LED 显示 Segment 信号输出引脚
IOA7	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
TK7	A	触摸按键通道 7
SEG31	O	LCD/LED 显示 Segment 信号输出引脚
IOB0	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD0	A	AD 通道 0
TK8	A	触摸按键通道 8
SEG16	O	LCD/LED 显示 Segment 信号输出引脚
IOB1	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD1	A	AD 通道 1
TK9	A	触摸按键通道 9
SEG17	O	LCD/LED 显示 Segment 信号输出引脚

名称	类型	说明
IOB2	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD2	A	AD 通道 2
TK10	A	触摸按键通道 10
SEG18	O	LCD/LED 显示 Segment 信号输出引脚
IOB3	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD3	A	AD 通道 3
TK11	A	触摸按键通道 11
SEG19	O	LCD/LED 显示 Segment 信号输出引脚
IOB4	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD4	A	AD 通道 4
TK12	A	触摸按键通道 12
SEG20	O	LCD/LED 显示 Segment 信号输出引脚
IOB5	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD5	A	AD 通道 5
TK13	A	触摸按键通道 13
SEG21	O	LCD/LED 显示 Segment 信号输出引脚
IOB6	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
TK14	A	触摸按键通道 14
SEG22	O	LCD/LED 显示 Segment 信号输出引脚
INT0	I	外部中断 0
IOB7	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
TK15	A	触摸按键通道 15
SEG23	O	LCD/LED 显示 Segment 信号输出引脚
INT1	I	外部中断 1
IOC0	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
TK16	A	触摸按键通道 16
SEG8	O	LCD/LED 显示 Segment 信号输出引脚
IOC1	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
TK17	A	触摸按键通道 17
SEG9	O	LCD/LED 显示 Segment 信号输出引脚
IOC2	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
TK18	A	触摸按键通道 18
SEG10	O	LCD/LED 显示 Segment 信号输出引脚
IOC3	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
TK19	A	触摸按键通道 19
SEG11	O	LCD/LED 显示 Segment 信号输出引脚
IOC4	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD6	A	AD 通道 6
TK20	A	触摸按键通道 20
SEG12	O	LCD/LED 显示 Segment 信号输出引脚

名称	类型	说明
IOC5	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD7	A	AD 通道 7
TK21	A	触摸按键通道 21
SEG13	O	LCD/LED 显示 Segment 信号输出引脚
IOC6	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD8	A	AD 通道 8
TK22	A	触摸按键通道 22
SEG14	O	LCD/LED 显示 Segment 信号输出引脚
IOC7	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD9	A	AD 通道 9
TK23	A	触摸按键通道 23
SEG15	O	LCD/LED 显示 Segment 信号输出引脚
IOD0	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD10	A	AD 通道 10
TK24	A	触摸按键通道 24
SEG0	O	LCD/LED 显示 Segment 信号输出引脚
COM0	O	LCD/LED 显示 COM 信号输出引脚
IOD1	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD11	A	AD 通道 11
TK25	A	触摸按键通道 25
SEG1	O	LCD/LED 显示 Segment 信号输出引脚
COM 1	O	LCD/LED 显示 COM 信号输出引脚
IOD2	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD12	A	AD 通道 12
TK26	A	触摸按键通道 26
SEG2	O	LCD/LED 显示 Segment 信号输出引脚
COM 2	O	LCD/LED 显示 COM 信号输出引脚
IOD3	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
AD13	A	AD 通道 13
TK27	A	触摸按键通道 27
SEG3	O	LCD/LED 显示 Segment 信号输出引脚
COM 3	O	LCD/LED 显示 COM 信号输出引脚
IOD4	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
TK28	A	触摸按键通道 28
SEG4	O	LCD/LED 显示 Segment 信号输出引脚
COM 4	O	LCD/LED 显示 COM 信号输出引脚
IOD5	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
TK29	A	触摸按键通道 29
SEG5	O	LCD/LED 显示 Segment 信号输出引脚
COM 5	O	LCD/LED 显示 COM 信号输出引脚

名称	类型	说明
IOD6 TK30 SEG6 COM 6	I/O A O O	输入/输出 IO、SMT、上拉电阻、下拉电阻 触摸按键通道 30 LCD/LED 显示 Segment 信号输出引脚 LCD/LED 显示 COM 信号输出引脚
IOD7 TK30 SEG7 T3CKI COM7	I/O A O I O	输入/输出 IO、SMT、上拉电阻、下拉电阻 触摸按键通道 31 LCD/LED 显示 Segment 信号输出引脚 TC3 外部时钟输入 LCD/LED 显示 COM 信号输出引脚
IOE0 AD14 CP0	I/O A A	输入/输出 IO、SMT、上拉电阻、下拉电阻 AD 通道 14 比较器 CMP 正相输入端口 0
IOE1 AD15 CP1	A A	输入/输出 IO、SMT、上拉电阻、下拉电阻 AD 通道 15 比较器 CMP 正相输入端口 1
IOE2 AD16 T4CKI CP2	I/O A I A	输入/输出 IO、SMT、上拉电阻、下拉电阻 AD 通道 16 TC4 外部时钟输入 比较器 CMP 正相输入端口 2
IOE3 AD17 CP3	I/O A A	输入/输出 IO、SMT、上拉电阻、下拉电阻 AD 通道 17 比较器 CMP 正相输入端口 3
IOE4 CN0	I/O A	输入/输出 IO、SMT、上拉电阻、下拉电阻 比较器 CMP 反相输入端口 0
IOE5 CN1	I/O A	输入/输出 IO、SMT、上拉电阻、下拉电阻 比较器 CMP 反相输入端口 1
IOE6 CN2	I/O A	输入/输出 IO、SMT、上拉电阻、下拉电阻 比较器 CMP 反相输入端口 2
IOE7 CN3	I/O A	输入/输出 IO、SMT、上拉电阻、下拉电阻 比较器 CMP 反相输入端口 3
IOF0 OSCI	I/O A	输入/输出 IO、SMT、上拉电阻、下拉电阻 外部晶体振荡器接口
IOF1 OSCO	I/O A	输入/输出 IO、SMT、上拉电阻、下拉电阻 外部晶体振荡器接口
IOF2 MRST	I/O I	输入/输出 IO、SMT、上拉电阻、下拉电阻 外部复位口(低电平有效，内建上拉到 VDD)
IOF3	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻
IOF4 CX	I/O O	输入/输出 IO、SMT、上拉电阻、下拉电阻 比较器 CMP 输出
IOF5	I/O	输入/输出 IO、SMT、上拉电阻、下拉电阻

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
0x00000 ~ 0x00007	用户区
0x00008 ~ 0x0018	中断向量
0x00019 ~ 0x0XXXX	用户区
0x0XXXX ~ 0x0XXXX	备份区
0x0XXXX ~ 0x0XXXX	类 EE
0x0XXXX ~ 0x0FFBF	Bootload 区
0x0FF80 ~ 0x0FFFF	Boot 数据区（1 个 Flash 页面）

APP 区(用户区): 配置字可设该区域大小, 仅能通过 BOOT 数据区写入, 可读取 APP 区和类 EE 区, 支持写入类 EE 区。

备份区：配置字可设该区域大小，仅能通过 BOOT 数据区写入，可读取 APP 区、类 EE 区及本区的数据。可写入类 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，分 BOOT 数据区和 BOOT_APP 区。若 BOOTS≥1，BOOT 数据区地址为最后一个页面，其余为 BOOT_APP 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），备份区写入数据。

具体见下表

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

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

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
备份区域=0 类 EE 区域=0 BOOT 区域=0	0x0000 ~ 0xFFFF	APP 区（用户区）
备份区域=24KB 类 EE 区域=2KB BOOT 区域=1KB	0x0000 ~ 0X93FF	APP 区（用户区）
	0X9400 ~ 0XF3FF	备份区
	0XF400 ~ 0xFBFF	类 EE 区
	0xFC00 ~ 0xFFFF	BOOT 区

注：（1）若配置字选择PC运行在类EE区复位功能，须根据配置字的类EE区分配地址，将类EE区起始地址的前两个地址占用，以防止被分配为程序区。
以上图为例：类EE区地址为0xF400~0xFBFF，则将0xF3FE提前占用
const unsigned int EE_OCCUPY_ADDR @0xF3FE;　　//类EE区前两个地址提前占用
（2）若配置字选择PC出BOOT后复位功能，必须将BOOT区结束地址（0XFFFE）地址定义占用，以防止被分配为程序区。
const unsigned int BOOT_OCCUPY_ADDR @0xFFFE;　//BOOT区后两个地址提前占用

2.1.1 复位向量（0000h）

M9F6601 有以下 6 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)
- 分区复位(EE 复位和 BOOT 复位)

发生上述任一种复位后，程序将从 0000h 或 BOOT 区（由配置字和寄存器决定）处开始运行，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）

M9F6601 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），16 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[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] **PCLATH[7:0]：** 程序计数器指针高字节锁存PC[15:8]

程序计数器（Program Counter，PC）指定欲取出执行的指令的地址。PC 为 16 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[15:8]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[4:0]				
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 **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 [4:0] **STKPTR[4:0]：** 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[15:8]							
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 [4:0] **TOSH[7:0]：** 栈顶寄存器高八位

返回地址堆栈允许最多 31 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 5 位的堆栈指针 STKPTR 来实现 31 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 31 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 32 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 32 次压栈的值，并且 STKPTR 将保持为 31。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

M9F6601 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，TBLPTRU 、TBLPTRH 和 TBLPTRL 组成 17 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     EXTE(DTAB)      ;获取数据表地址扩展位
        MOVAR     TBLPTRU         ;设置数据表高位指针
        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 3680 个字节。存储空间最多被分为 15 个存储区，每个存储区包含 256 个字节。数据存储器由通用寄存器（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[4:0]，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	GPR	-
BANK3	0x300-0x3FF	GPR	-
BANK4	0x400-0x4FF	GPR	-
BANK5	0x500-0x5FF	GPR	-
BANK6	0x600-0x6FF	GPR	-
BANK7	0x700-0x7FF	GPR	-
BANK8	0x800-0x8FF	GPR	-
BANK9	0x900-0x9FF	GPR	-
BANK10	0xA00-0xAFF	GPR	-
BANK11	0xB00-0xBFF	GPR	-
BANK12	0xC00-0xCFF	GPR	-
BANK13	0xD00-0xDFF	GPR	-
BANK14	0xE00-0xE5F	GPR	-
	0xE60-0xE7F	GPR(1)	LCD/LED RAM
	0xE80-0xEFF	SFR	-
BANK15	0xF00-0xF5F	SFR	-
	0xF60-0xFFF	SFR	快速操作寄存器

注：
(1) 表示此 GPR 地址被 LCD/LED 显示缓存占用。

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	0	0	0	0	0	0	0

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	0	0	0	0	0	0	0	0

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	IOFOR	F9Fh	TC2PRH	F7Fh	I2CCR
FFEh	TOSH	FDEh	POSTINC2	FBEh	IOEOR	F9Eh	TC2PRL	F7Eh	I2CSTA
FFDh	TOSL	FDDh	POSTDEC2	FBDh	IODOR	F9Dh	TC2CH	F7Dh	I2CDATA
FFCh	STKPTR	FDCh	PREINC2	FBCh	IOCOR	F9Ch	TC2CL	F7Ch	I2CADR
FFBh	-	FDBh	PLUSW2	FBBh	IOBOR	F9Bh	TC2GCR	F7Bh	SPICR
FFAh	PCLATH	FDAh	FSR2H	FBAh	IOAOR	F9Ah	T2CR	F7Ah	SPISTA
FF9h	PCL	FD9h	FSR2L	FB9h	EXB1	F99h	-	F79h	SPIDATA
FF8h	TBLPTRU	FD8h	STATUS	FB8h	EXB0	F98h	-	F78h	RX1REG
FF7h	TBLPTRH	FD7h	INTF4	FB7h	EXA3	F97h	-	F77h	TX1REG
FF6h	TBLPTRL	FD6h	INTF3	FB6h	EXA2	F96h	-	F76h	RX1CR
FF5h	TABLAT	FD5h	INTF2	FB5h	EXA1	F95h	-	F75h	TX1CR
FF4h	PRODH	FD4h	INTF1	FB4h	EXA0	F94h	PWM11DH	F74h	RX0REG
FF3h	PRODL	FD3h	INTF0	FB3h	MDUCON	F93h	PWM11DL	F73h	TX0REG
FF2h	OPTION	FD2h	INTCR4	FB2h	CRCDH	F92h	PWM10DH	F72h	RX0CR
FF1h	EECON1	FD1h	INTCR3	FB1h	CRCDL	F91h	PWM10DL	F71h	TX0CR
FF0h	EECON2	FD0h	INTCR2	FB0h	CRCIN	F90h	PWM1CR	F70h	-
FEFh	INDF0	FCFh	INTCR1	FAFh	CRCCR	F8Fh	TC1PRH	F6Fh	TKCNTH
FEEh	POSTINC0	FCEh	INTCR0	FAEh	TC3PRH	F8Eh	TC1PRL	F6Eh	TKCNTL
FEDh	POSTDEC0	FCDh	-	FADh	TC3PRL	F8Dh	TC1CH	F6Dh	TKCHS
FECh	PREINC0	FCCh	PCON	FACh	TC3CH	F8Ch	TC1CL	F6Ch	TKCHS3
FEBh	PLUSW0	FCBh	IOF	FABh	TC3CL	F8Bh	T1CR	F6Bh	TKCHS2
FEAh	FSR0H	FCAh	IOE	FAAh	TC3GCR	F8Ah	PWM0D	F6Ah	TKCHS1
FE9h	FSR0L	FC9h	IOD	FA9h	T3CR	F89h	PWM0CR	F69h	TKCHS0
FE8h	AREG	FC8h	IOC	FA8h	-	F88h	TC0CH	F68h	TKCCTRH
FE7h	INDF1	FC7h	IOB	FA7h	-	F87h	TC0CL	F67h	TKCCTRL
FE6h	POSTINC1	FC6h	IOA	FA6h	ADH	F86h	T0CR2	F66h	TKRCTR
FE5h	POSTDEC1	FC5h	OEF	FA5h	ADL	F85h	T0CR	F65h	TKCTR5
FE4h	PREINC1	FC4h	OEE	FA4h	ADCON4	F84h	FLBUFH	F64h	TKCTR4
FE3h	PLUSW1	FC3h	OED	FA3h	ADCON3	F83h	FLBUFL	F63h	TKCTR3
FE2h	FSR1H	FC2h	OEC	FA2h	ADCON2	F82h	FLOPEN	F62h	TKCTR2
FE1h	FSR1L	FC1h	OEB	FA1h	ADCON1	F81h	SCM	F61h	TKCTR1
FE0h	BSR	FC0h	OEA	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	IOFIPS	EFFh	CALLOCK	EDFh	BRGD5H
F5Eh	-	F3Eh	-	F1Eh	IOEIPS	EFEh	IRCCAL	EDEh	BRGD5L
F5Dh	MPADET	F3Dh	-	F1Dh	IODIPS	EFDh	IRCCAH	EDDh	BRGD4H
F5Ch	MPFB1	F3Ch	IOFODS	F1Ch	IOCIPS	EFCh	LVRSR	EDCh	BRGD4L
F5Bh	MPFB0	F3Bh	-	F1Bh	IOBIPS	EFBh	FCPU	EDBh	BRGD3H
F5Ah	MPEPWM5	F3Ah	IOEODS	F1Ah	IOAIPS	EFAh	VREFCAL	EDAh	BRGD3L
F59h	MPEPWM4	F39h	-	F19h	IOFICR	EF9h	MOSC	ED9h	BRGD2H
F58h	MPEPWM3	F38h	IODODS	F18h	IOEICR	EF8h	-	ED8h	BRGD2L
F57h	MPEPWM2	F37h	-	F17h	IODICR	EF7h	-	ED7h	BRGD1H
F56h	MPEPWM1	F36h	IOCODS	F16h	IOCICR	EF6h	-	ED6h	BRGD1L
F55h	MPEPWM0	F35h	-	F15h	IOBICR	EF5h	-	ED5h	BRGD0H
F54h	MPPWM11	F34h	IOBODS	F14h	IOAICR	EF4h	-	ED4h	BRGD0L
F53h	MPPWM10	F33h	-	F13h	TC4PRH	EF3h	IDEH	ED3h	RX5REG
F52h	MPPWM0	F32h	IOAODS	F12h	TC4PRL	EF2h	IDEL	ED2h	TX5REG
F51h	MPT4G	F31h	PUF	F11h	TC4CH	EF1h	VERH	ED1h	RX5CR
F50h	MPT3G	F30h	PUE	F10h	TC4CL	EF0h	VERL	ED0h	TX5CR
F4Fh	MPT2G	F2Fh	PUD	F0Fh	TC4GCR	EEFh	-	ECFh	RX4REG
F4Eh	MPSPI	F2Eh	PUC	F0Eh	T4CR	EEEh	-	ECEh	TX4REG
F4Dh	MPSDA	F2Dh	PUB	F0Dh	CMPC2	EEDh	-	ECDh	RX4CR
F4Ch	MPSCL	F2Ch	PUA	F0Ch	CMPC1	EECh	-	ECCh	TX4CR
F4Bh	MPRX5	F2Bh	PDF	F0Bh	CMPC0	EEBh	-	ECBh	RX3REG
F4Ah	MPTX5	F2Ah	PDE	F0Ah	-	EEAh	-	ECAh	TX3REG
F49h	MPRX4	F29h	PDD	F09h	-	EE9h	-	EC9h	RX3CR
F48h	MPTX4	F28h	PDC	F08h	SEGIOSD	EE8h	-	EC8h	TX3CR
F47h	MPRX3	F27h	PDB	F07h	SEGIOSC	EE7h	-	EC7h	RX2REG
F46h	MPTX3	F26h	PDA	F06h	SEGIOSB	EE6h	-	EC6h	TX2REG
F45h	MPRX2	F25h	ANSF	F05h	SEGIOSA	EE5h	-	EC5h	RX2CR
F44h	MPTX2	F24h	ANSE	F04h	LCDCR4	EE4h	INTP4	EC4h	TX2CR
F43h	MPRX1	F23h	ANSD	F03h	LCDCR3	EE3h	INTP3	EC3h	-
F42h	MPTX1	F22h	ANSC	F02h	LCDCR2	EE2h	INTP2	EC2h	-
F41h	MPRX0	F21h	ANSB	F01h	LCDCR1	EE1h	INTP1	EC1h	-
F40h	MPTX0	F20h	ANSA	F00h	LCDCR0	EE0h	INTP0	EC0h	-

数据寄存器映射表（E80h - EBFh）									
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
EBFh	-	E9Fh	-	-	-	-	-	-	-
EBEh	-	E9Eh	-	-	-	-	-	-	-
EBDh	-	E9Dh	EPWMDD5H	-	-	-	-	-	-
EBCh	-	E9Ch	EPWMDD5L	-	-	-	-	-	-
EBBh	-	E9Bh	EPWMDD4H	-	-	-	-	-	-
EBAh	-	E9Ah	EPWMDD4L	-	-	-	-	-	-
EB9h	-	E99h	EPWMDD3H	-	-	-	-	-	-
EB8h	-	E98h	EPWMDD3L	-	-	-	-	-	-
EB7h	-	E97h	EPWMDD2H	-	-	-	-	-	-
EB6h	-	E96h	EPWMDD2L	-	-	-	-	-	-
EB5h	EPWMDIF	E95h	EPWMDD1H	-	-	-	-	-	-
EB4h	EPWMDIE	E94h	EPWMDD1L	-	-	-	-	-	-
EB3h	EPWMUIF	E93h	EPWMDD0H	-	-	-	-	-	-
EB2h	EPWMUIE	E92h	EPWMDD0L	-	-	-	-	-	-
EB1h	EPWMZIF	E91h	EPWMD5H	-	-	-	-	-	-
EB0h	EPWMZIE	E90h	EPWMD5L	-	-	-	-	-	-
EAFh	EPWMPIF	E8Fh	EPWMD4H	-	-	-	-	-	-
EAEh	EPWMPIE	E8Eh	EPWMD4L	-	-	-	-	-	-
EADh	EPWMFBKD	E8Dh	EPWMD3H	-	-	-	-	-	-
EACh	EPWMFBKC	E8Ch	EPWMD3L	-	-	-	-	-	-
EABh	EPWMMASKD	E8Bh	EPWMD2H	-	-	-	-	-	-
EAAh	EPWMMASKE	E8Ah	EPWMD2L	-	-	-	-	-	-
EA9h	EPWMLOADEN	E89h	EPWMD1H	-	-	-	-	-	-
EA8h	EPWMCNTCLR	E88h	EPWMD1L	-	-	-	-	-	-
EA7h	EPWMCNTM	E87h	EPWMD0H	-	-	-	-	-	-
EA6h	EPWMCNTE	E86h	EPWMD0L	-	-	-	-	-	-
EA5h	EPWM45CKPS	E85h	EPWMP45H	-	-	-	-	-	-
EA4h	EPWM23CKPS	E84h	EPWMP45L	-	-	-	-	-	-
EA3h	EPWM01CKPS	E83h	EPWMP23H	-	-	-	-	-	-
EA2h	EPWMMPINV	E82h	EPWMP23L	-	-	-	-	-	-
EA1h	EPWMOE	E81h	EPWMP01H	-	-	-	-	-	-
EA0h	EPWMCON	E80h	EPWMP01L	-	-	-	-	-	-

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 128 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	-	-	-	-	-	CLRPL	ERASE	WRITE
读/写	-	-	-	-	-	R/W	R/W	R/W
复位后	0	1	0	0	0	0	0	0

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区起始地址低八位

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 缓存区起始地址高四位

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区和APP程序区擦写APP程序区
以及BOOT数据区写入EE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[7:0]: **FLOPEN [7：0]：软件复位向量控制位，需配合软复位操作**

FLOPEN [7：0] = 0XC0 ： 执行软复位操作后，复位向量为BOOT地址

FLOPEN [7：0] = 0X30 ： 执行软复位操作后，复位向量为0x00地址

FLOPEN [7：0] = 其它 ： 执行软复位操作后，复位向量受配置字控制

该寄存器要按以下步骤写0XAA值，否则值为复位值，

```
MOVIA     0x55
MOVAR     CALLOCK          ;// CALLOCK写入55H
MOVIA     0XAA
MOVAR     CALLOCK          ;// CALLOCK地址写入AAH
MOVIA     0XAA
MOVAR     FLOPEN           ;//写入AAH
...
```

注： 对非 IAP 相关寄存器（FLBUFL、FLBUFH）执行写操作后，FLOPEN 寄存器会在该操作后发生复位 。

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 128 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。

2.3.8 Flash 操作示例

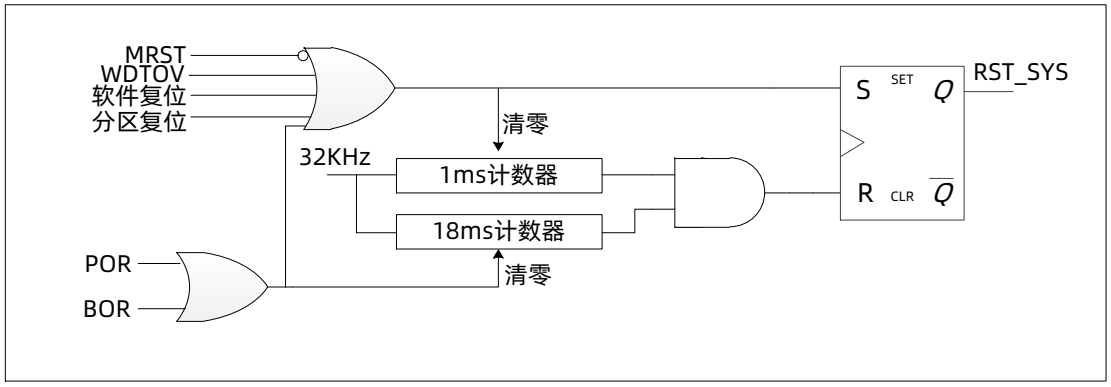
注：关于 Flash 应用请参照 DEMO 相关例程。

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)
- 分区复位(EE 复位和 BOOT 复位)

M9F6601有以上6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	TD	PD	APPRST	MCLR	POR	BOR
读/写	-	R/W	R	R	R	R	R	R
复位后	-	1	1	1	1	0	0	0

Bit 6 **SOFTRST：** 软复位
0 = 发生软复位
1 = 上电复位、掉电复位，软件清零

Bit 5 **TO**：超时位
0 = WDT发生溢出
1 = 上电复位或清除WDT或进入休眠模式

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

Bit 3 **APPRST：** 分区复位
0 = 发生分区复位（EE复位和BOOT复位）
1 = 未发生分区复位，软件置1

Bit 2 **MCLR**：复位口复位
0 = 发生复位口复位
1 = 未发生复位口复位，软件置1

Bit 1 **POR：** 上电复位位
0 = 发生上电复位
1 = 未发生上电复位，软件置1

Bit 0 **BOR：** 欠压复位位
0 = 发生欠压复位
1 = 未发生欠压复位，软件置1

3.3 LVRSR 复位电压选择

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LVRSR					LVRSR3	LVRSR2	LVRSR1	LVRSR0
读/写					R/W	R/W	R/W	R/W
复位后					x	x	x	x

bit[3:0]: **LVRSR[3:0]**：复位电压选择，初始值由配置字决定。

LVRSR[3:0]	复位电压选择
0111	2.2V
1000	2.5V
1010	2.8V
1111	3.4V

该寄存器要按以下步骤写入0X07值，

```
MOVIA       0X55
MOVAR       CALLOCK         ;CALLOCK写入55H
MOVIA       0XAA
MOVAR       CALLOCK         ;CALLOCK写入AAH
MOVIA       0X07
MOVAR       LVRSR           ;写入07H
```

4 系统时钟

4.1 概述

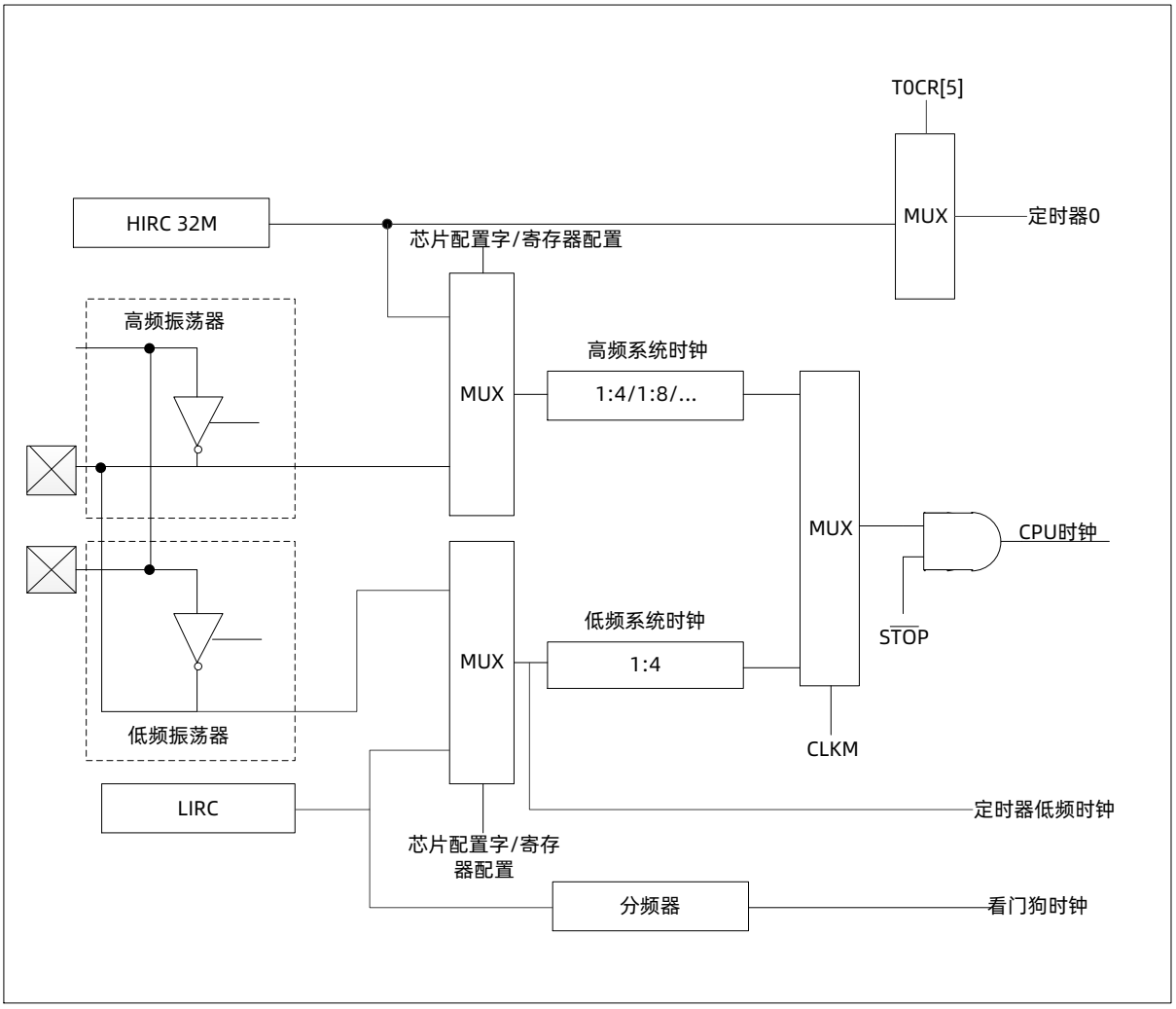
M9F6601支持双时钟系统：高速时钟和低速时钟。系统时钟的时钟源具有 5 种类型，可通过配置字或用户寄存器的设置进行时钟源、时钟分频选择。系统时钟源如下：

- 内部高速RC振荡器HIRC32（32MHz）
- 外部高速振荡器HXT（4-24MHz）
- 内部低速RC振荡器LIRC（64KHz）
- 外部低速振荡器LXT（32.768KHz）

芯片上电默认使用配置字设置时钟源，若需要改变系统时钟源，用户可通过配置系统时钟选择寄存MOSC（按照操作步骤配置MOSC，详见下文）。系统时钟源选择HXT/LXT时，支持停振检测功能，详见下文。

注：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。

4.2 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

4.3 OSCM 系统状态控制寄存器

	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。

4.4 SCM 晶振监控寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SCM	-	-	-	-	-	-	SCMSTA	SCMEN
读/写	-	-	-	-	-	-	R	RW
复位后	-	-	-	-	-	-	0	0

Bit1 **SCMSTA**：停振状态位

1 = 停振；

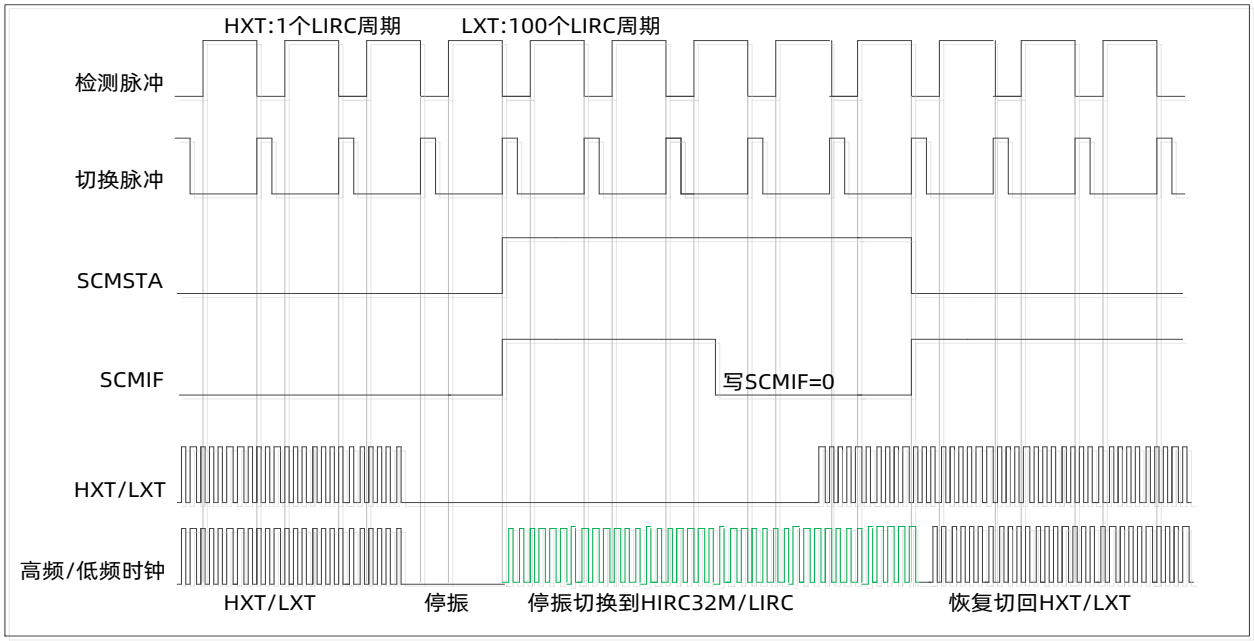
0 = 正常；

Bit0 **SCMEN**：停振检测模块使能

1 = 使能；

0 = 禁止

注：1、只要系统时钟源内含有HXT/LXT时，使能SCMEN就会开启晶振检测功能。
2、当系统时钟源不含有HXT/LXT时，使能晶振检测功能会异常，不可使用。
3、休眠模式下，晶振检测开启会强制开启LIRC时钟，会影响功耗。
4、当检测外部高频晶振频率低于1MHz则认为外部高频晶振失效
5、当检测外部低频晶振频率低于10KHz则认为外部低频晶振失效



当使用HXT做系统时钟时，在1个LIRC周期一旦检测到 HXT异常，下一个切换脉冲会强制启动HIRC32MHz作为高频时钟，停振状态位SCMSTA置1，若使能晶振检测中断则触发中断。当检测到HXT正常运行，则切回HXT作高频时钟，停振状态位SCMSTA清0，若使能晶振检测中断则触发中断。

当使用LXT做系统时钟时，在100个LIRC周期一旦检测到 LXT异常，下一个切换脉冲会强制启动LIRC作为低频时钟，停振状态位SCMSTA置1，若使能晶振检测中断则触发中断。当检测到LXT正常运行，则切回LXT作低频时钟，停振状态位SCMSTA清0，若使能晶振检测中断则触发中断。

4.5 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	256T
001	256T
010	128T
011	64T
100	32T
101	16T
110	8T
111	4T

注：FCPU的初始状态由配置字决定。

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	R/W
复位后	x	x	x	x	x	x	x	x

32MHz 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。

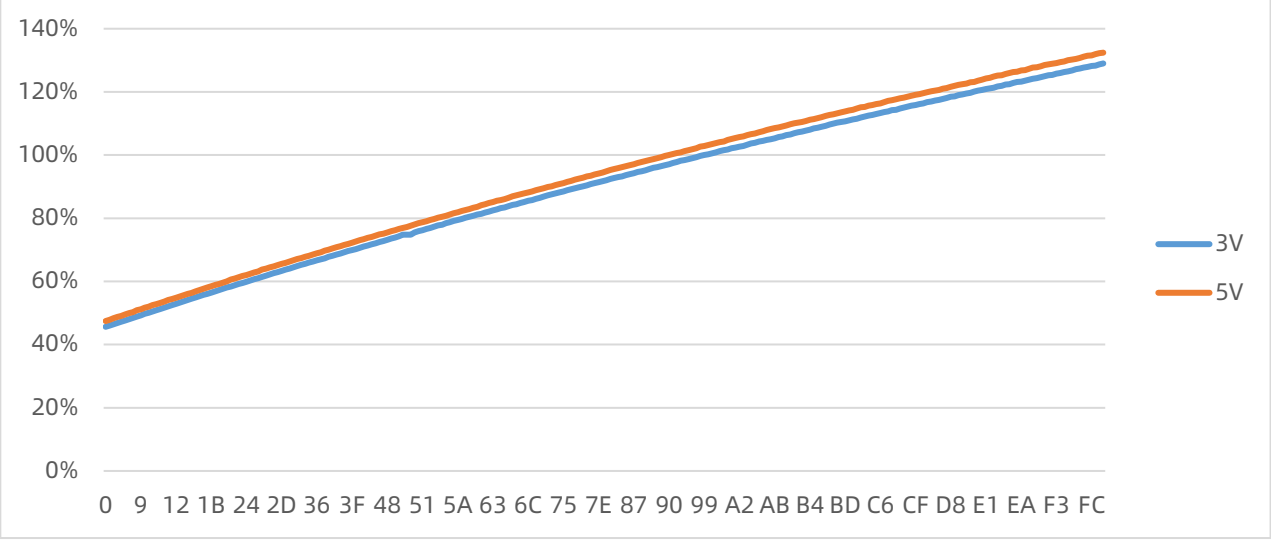
内置的双高频 RC 振荡器在芯片出厂后，频率将校准到 32MHz，IRCCAH 复位初值即为出厂校准值，但程序中可以通过特殊的写入流程来微调此频率以满足特定应用需求。

同时可以通过该特殊写入流程来调整 cpu 的分频。

例： 调整 32MHz 高频内部 IRC 频率

```
TASK_IRCCAH:
        BANKBSR     CAL_BANKSEL
        MOVIA       0x55
        MOVAR       CALLOCK          ;// CALLOCK写入0x55
        MOVIA       0xAA
        MOVAR       CALLOCK          ;// CALLOCK写入0xAA
        MOVIA       0x04
        MOVAR       IRCCAH           ;//写入04H
        ...
;//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
```

32MHZ HIRC 调整频率趋势图



注：具体值不做设计保证。

4.7 系统时钟的工作模式

4.7.1 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}）

（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_{DD2}）

4.7.2 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）。

4.7.3 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_{SP4}）

休眠模式可以由 IO 电平变化中断、外设中断或 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，中断能唤醒芯片但是不会进中断。

（4）如果想高频停止工作后进入休眠，需要在进休眠前保证 cdc 和 timer0 没有使用高频时钟，否则休眠时外设使用的高频时钟无法关闭。

4.8 系统时钟切换

高频时钟选择外部高频晶振 HXT，高频稳定时间：T_{HXT} X 752+高频起振时间

高频时钟选择内部 RC 振荡器 HIRC32MHz，高频稳定时间：$T_{HIRC32MHz}$ X 752 ≈ 16us

低频时钟选择外部低频晶振 LXT，低频稳定时间：T_{LXT} X 1024+低频起振时间

低频时钟选择内部 RC 振荡器 LIRC，低频稳定时间：T_{LIRC} X 368 ≈ 240us

当软件切换系统时钟源、检测晶振异常、振荡器关闭时，会清除对应的时钟稳定标志位（STBH\ STBL），只有稳定计数器达到对应个数时，振荡器稳定标志位（STBH\ STBL）才会置 1，此时才会接入 CPU 时钟。

时钟源切换时间：

软件切换系统时钟源，切换时间 = 目标振荡器稳定时间

HXT 异常切换时钟源，切换时间 = HIRC32MHz 稳定时间

LXT 异常切换时钟源，切换时间 = LIRC 稳定时间

高低频切换时间：

高频切低频：3 个 2MHz 切换时钟周期

低频切高频（STBH = 0）：高频振荡器稳定时间

低频切高频（STBH = 1）：3 个 2MHz 切换时钟周期

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

M9F6601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 GIE/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[GIE，GIEL]的值是 1，则允许高优先级的中断嵌套；响应高优先级时，GIE 和 GIEL 会被自动清零，同时将 GIE/GIE 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）。

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	IOCICR7	IOCICR6	IOCICR5	IOCICR4	IOCICR3	IOCICR2	IOCICR1	IOCICR0
IODICR	IODICR7	IODICR6	IODICR5	IODICR4	IODICR3	IODICR2	IODICR1	IODICR0
IOEICR	IOEICR7	IOEICR6	IOEICR5	IOEICR4	IOEICR3	IOEICR2	IOEICR1	IOEICR0
IOFICR	-	-	IOFICR5	IOFICR4	IOFICR3	IOFICR2	IOFICR1	IOFICR0
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:** IOA端口变化中断使能（x = A/B/C/D/E/F，y = 0-7）

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	TC4GIE	TC4IE	TC3GIE	TC3IE	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 **TC4GIE：** TC4门控中断使能位
0 = 屏蔽TC4门控中断
1 = 使能TC4门控中断

Bit 6 **TC4IE：** TC4溢出中断使能位
0 = 屏蔽TC4溢出中断
1 = 使能TC4溢出中断

Bit 5 **TC3GIE：** TC3门控中断使能位
0 = 屏蔽TC3门控中断
1 = 使能TC3门控中断

Bit 4 **TC3IE：** TC3溢出中断使能位
0 = 屏蔽TC3溢出中断
1 = 使能TC3溢出中断

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	TC4GIF	TC4IF	TC3GIF	TC3IF	TC2GIF	TC2IF	TC1IF	TC0IF
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 **TC4GIF：** TC4门控中断标志
0 = 未产生TC4门控中断
1 = 产生TC4门控中断

Bit 6 **TC4IF：** TC4溢出中断标志
0 = 未产生TC4溢出中断
1 = 产生TC4溢出中断

Bit 5 **TC3GIF：** TC3门控中断标志
0 = 未产生TC3门控中断
1 = 产生TC3门控中断

Bit 4 **TC3IF：** TC3溢出中断标志
0 = 未产生TC3溢出中断
1 = 产生TC3溢出中断

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

5.7 INTP0 中断优先级寄存器 0

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTP0	TC4GIP	TC4IP	TC3GIP	TC3IP	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 **TC4GIP：**TC4门控中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 6 **TC4IP：**TC4溢出中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 5 **TC3GIP：**TC3门控中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 4 **TC3IP：**TC3溢出中断优先级控制位
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 1 **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	IOFCHIE	IOECHIE	IODCHIE	IOCCHIE	IOBCHIE	IOACHIE
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 **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 **IOFCHIE：** 端口F变化中断使能位
0 = 屏蔽端口F变化中断
1 = 使能端口F变化中断

Bit 4 **IOECHIE：** 端口E变化中断使能位
0 = 屏蔽端口E变化中断
1 = 使能端口E变化中断

Bit 3 **IODCHIE：** 端口D变化中断使能位
0 = 屏蔽端口D变化中断
1 = 使能端口D变化中断

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	IOFCHIF	IOECHIF	IODCHIF	IOCCHIF	IOBCHIF	IOACHIF
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 **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 **IOFCHIF：** 端口F变化中断标志
0 = 未产生端口F变化中断
1 = 产生端口F变化中断

Bit 4 **IOECHIF：** 端口E变化中断标志
0 = 未产生端口E变化中断
1 = 产生端口E变化中断

Bit 3 **IODCHIF：** 端口D变化中断标志
0 = 未产生端口D变化中断
1 = 产生端口D变化中断

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	IOFCHIP	IOECHIP	IODCHIP	IOCCHIP	IOBCHIP	IOACHIP
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 **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 **IOFCHIP：** 端口F变化中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 4 **IOECHIP：** 端口E变化中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 3 **IODCHIP：** 端口D变化中断优先级控制位
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	RX3IE	TX3IE	RX2IE	TX2IE	RX1IE	TX1IE	RX0IE	TX0IE
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 **RX3IE：** USART3接收中断使能位
0 = 屏蔽UART3接收中断
1 = 使能UART3接收中断

Bit 6 **TX3IE：** USART3发送中断使能位
0 = 屏蔽UART3发送中断
1 = 使能UART3发送中断

Bit 5 **RX2IE：** USART2接收中断使能位
0 = 屏蔽UART2接收中断
1 = 使能UART2接收中断

Bit 4 **TX2IE：** USART2发送中断使能位
0 = 屏蔽UART2发送中断
1 = 使能UART2发送中断

Bit 3 **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 2 **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 1 **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 0 **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

5.12 INTF2 中断标志寄存器 2

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTF2	RX3IF	TX3IF	RX2IF	TX2IF	RX1IF	TX1IF	RX0IF	TX0IF
读/写	R	R	R	R	R	R	R	R
复位后	0	1	0	1	0	1	0	1

Bit 7 **RX3IF：** USART3接收中断标志
0 = 未产生UART3接收中断
1 = 产生UART3接收中断（对RX3REG进行读操作后自动清零）

Bit 6 **TX3IF：** USART3发送中断标志
0 = 未产生UART3发送中断（数据不为空，还在发送中）
1 = 产生UART3发送中断（数据为空，可发送下一个数据）

Bit 5 **RX2IF：** USART2接收中断标志
0 = 未产生UART2接收中断
1 = 产生UART2接收中断（对RX2REG进行读操作后自动清零）

Bit 4 **TX2IF：** USART2发送中断标志
0 = 未产生UART2发送中断（数据不为空，还在发送中）
1 = 产生UART2发送中断（数据为空，可发送下一个数据）

Bit 3 **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 2 **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 1 **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 0 **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）

5.13 INTP2 中断优先级寄存器 2

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTP2	RX3IP	TX3IP	RX2IP	TX2IP	RX1IP	TX1IP	RX0IP	TX0IP
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 **RX3IP：** UART3接收中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 6 **TX3IP：** UART3发送中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 5 **RX2IP：** UART2接收中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 4 **TX2IP：** UART2发送中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 3 **RX1IP：** UART1接收中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 2 **TX1IP：** UART1发送中断优先级控制位
0 = 高优先级
1 = 低优先级

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

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

5.14 INTCR3 中断控制寄存器 3

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCR3	ADIE	CMP0IE	I2CIE	SPIIE	RX5IE	TX5IE	RX4IE	TX4IE
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 **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 6 **CMP0IE：** 比较器中断使能位
0 = 屏蔽CMP中断
1 = 使能CMP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 **SPIIE：** SPI中断使能位
0 = 屏蔽SPI中断
1 = 使能SPI中断

Bit 3 **RX5IE：** USART5接收中断使能位
0 = 屏蔽UART5接收中断
1 = 使能UART5接收中断

Bit 2 **TX5IE：** USART5发送中断使能位
0 = 屏蔽UART5发送中断
1 = 使能UART5发送中断

Bit 1 **RX4IE：** USART4接收中断使能位
0 = 屏蔽UART4接收中断
1 = 使能UART4接收中断

Bit 0 **TX4IE：** USART4发送中断使能位
0 = 屏蔽UART4发送中断
1 = 使能UART4发送中断

5.15 INTF3 中断标志寄存器 3

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTF3	ADIF	CMP0IF	I2CIF	SPIF2	RX5IF	TX5IF	RX4IF	TX4IF
读/写	R/W	R/W	R	R	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	1	0	1

Bit 7 **ADIF：AD**中断标志
0 = 未产生AD转换中断
1 = 产生AD转换中断

Bit 6 **CMP0IF：**比较器中断使能位
0 = 未产生CMP中断
1 = 产生CMP中断

Bit 5 **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 **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

Bit 3 **RX5IF：**USART5接收中断标志
0 = 未产生UART5接收中断
1 = 产生UART5接收中断（对RX5REG进行读操作后自动清零）

Bit 2 **TX5IF：**USART5发送中断标志
0 = 未产生UART5发送中断（数据不为空，还在发送中）
1 = 产生UART5发送中断（数据为空，可发送下一个数据）

Bit 1 **RX4IF：**USART4接收中断标志
0 = 未产生UART4接收中断
1 = 产生UART5接收中断（对RX4REG进行读操作后自动清零）

Bit 0 **TX4IF：**USART4发送中断标志
0 = 未产生UART4发送中断（数据不为空，还在发送中）
1 = 产生UART5发送中断（数据为空，可发送下一个数据）

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。

5.16 INTP3 中断优先级寄存器 3

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTP3	ADIP	CMP0IP	I2CIP	SPIP	RX5IP	TX5IP	RX4IP	TX4IP
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 **ADIP：AD**中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 6 **CMP0IP：**CMP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 **SPIP：**SPI中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 3 **RX5IP：**UART5接收中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 2 **TX5IP：**UART5发送中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 1 **RX5IP：**UART4接收中断优先级控制位
0 = 高优先级
1 = 低优先级

Bit 0 **TX4IP：**UART4发送中断优先级控制位
0 = 高优先级
1 = 低优先级

5.17 INTCR4 中断控制寄存器 4

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCR4						SCMIE	LEDIE	TKIE
读/写						R/W	R/W	R/W
复位后						0	0	0

Bit 2 **SCMIE：** 系统时钟监控使能位
0 = 屏蔽系统时钟监控中断
1 = 使能系统时钟监控中断

Bit 1 **LEDIE：** LED中断使能位
0 = 屏蔽LED中断
1 = 使能LED中断

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

5.18 INTF4 中断标志寄存器 4

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTF4						SCM**IF**	LEDIF	TKIF
读/写						R/W	R/W	R/W
复位后						0	0	0

Bit 2 **SCMIF：** 系统时钟监控中断标志
0 = 未产生系统时钟监控中断
1 = 产生系统时钟监控中断

Bit 1 **LEDIF：** LED中断标志
0 = 未产生LED中断
1 = 产生LED中断

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

5.19 INTP4 中断优先级寄存器 4

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTP4					EPWMIP	SCMIP	LEDIP	TKIP
读/写					R/W	R/W	R/W	R/W
复位后					0	0	0	0

Bit 3 **EPWMIP：EPWM中断优先级控制位**
0 = 高优先级
1 = 低优先级

Bit 2 **SCMIP：系统时钟监控中断优先级控制位**
0 = 高优先级
1 = 低优先级

Bit 1 **LEDIP：LED中断优先级控制位**
0 = 高优先级
1 = 低优先级

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

5.20 EPWMPIE 中断控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMPIE						EPWM45PIE	EPWM23PIE	EPWM01PIE
读/写						R/W	R/W	R/W
复位后						0	0	0

Bit 2 **EPWM45PIE：EPWM45周期中断使能位**
0 = 屏蔽PWM45周期中断
1 = 使能PWM45周期中断

Bit 1 **EPWM23PIE：EPWM23周期中断使能位**
0 = 屏蔽PWM23周期中断
1 = 使能PWM23周期中断

Bit 0 **EPWM01PIE：EPWM01周期中断使能位**
0 = 屏蔽PWM01周期中断
1 = 使能PWM01周期中断

5.21 EPWMPIF 中断标志寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMPIF						EPWM45PIF	EPWM23PIF	EPWM01PIF
读/写						R/W	R/W	R/W
复位后						0	0	0

Bit 2 **EPWM45PIF：EPWM45周期中断使能位**

0 = 未产生中断

1 = 产生中断（软件清零）

Bit 1 **EPWM23PIF：EPWM23周期中断使能位**

0 = 未产生中断

1 = 产生中断（软件清零）

Bit 0 **EPWM01PIF：EPWM01周期中断使能位**

0 = 未产生中断

1 = 产生中断（软件清零）

5.22 EPWMZIE 中断控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMZIE						EPWM45ZIE	EPWM23ZIE	EPWM01ZIE
读/写						R/W	R/W	R/W
复位后						0	0	0

Bit 2 **EPWM45PIE：EPWM45零点中断使能位**

0 = 屏蔽零点中断

1 = 使能零点中断

Bit 1 **EPWM23PIE：EPWM23零点中断使能位**

0 = 屏蔽零点中断

1 = 使能零点中断

Bit 0 **EPWM01PIE：EPW01零点中断使能位**

0 = 屏蔽零点中断

1 = 使能零点中断

5.23 EPWMZIF 中断标志寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMZF						EPWM45ZIF	EPWM23ZIF	EPWM01ZIF
读/写						R/W	R/W	R/W
复位后						0	0	0

Bit 2 **EPWM45ZIF：**EPWM45零点中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

Bit 1 **EPWM23ZIF：**EPWM23零点中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

Bit 0 **EPWM01ZIF：**EPWM01零点中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

5.24 EPWMUIE 断控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMUE			EPWM5UE	EPWM4UE	EPWM3UE	EPWM2UE	EPWM1UE	EPWM0UE
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 **EPWM5UE：**EPWM5向上比较中断使能位
0 = 屏蔽向上比较中断
1 = 使能向上比较中断

Bit 4 **EPWM4UE：**EPWM4向上比较中断使能位
0 = 屏蔽向上比较中断
1 = 使能向上比较中断

Bit 3 **EPWM3UE：**EPWM3向上比较中断使能位
0 = 屏蔽向上比较中断
1 = 使能向上比较中断

Bit 2 **EPWM2UE：**EPWM2向上比较中断使能位
0 = 屏蔽向上比较中断
1 = 使能向上比较中断

Bit 1 **EPWM1UE：**EPWM1向上比较中断使能位
0 = 屏蔽向上比较中断
1 = 使能向上比较中断

Bit 0 **EPWM0UE：**EPWM0向上比较中断使能位
0 = 屏蔽向上比较中断
1 = 使能向上比较中断

5.25 EPWMUIF 断标志寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMUF			EPWM5UIF	EPWM4UIF	EPWM3UIF	EPWM2UIF	EPWM1UIF	EPWM0UIF
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 **EPWM5UIF：**EPWM5向上比较中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

Bit 4 **EPWM4UIF：**EPWM4向上比较中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

Bit 3 **EPWM3UIF：**EPWM3向上比较中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

Bit 2 **EPWM2UIF：**EPWM2向上比较中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

Bit 1 **EPWM1UIF：**EPWM1向上比较中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

Bit 0 **EPWM0UIF：**EPWM0向上比较中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

5.26 EPWMDIE 中断控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMDIE			EPWM5DIE	EPWM4DIE	EPWM3DIE	EPWM2DIE	EPWM1DIE	EPWM0DIE
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 **EPWM5DIE：**EPWM5向下比较中断使能位
0 = 屏蔽零点中断
1 = 使能零点中断

Bit 4 **EPWM4DIE：**EPWM4零向下比较中断使能位
0 = 屏蔽零点中断
1 = 使能零点中断

Bit 3 **EPWM3DIE：**EPWM3向下比较中断使能位
0 = 屏蔽零点中断
1 = 使能零点中断

Bit 2 **EPWM2DIE：**EPWM2向下比较中断使能位
0 = 屏蔽零点中断
1 = 使能零点中断

Bit 1 **EPWM1DIE：**EPWM1向下比较中断使能位
0 = 屏蔽零点中断
1 = 使能零点中断

Bit 0 **EPWM0DIE：**EPWM0向下比较中断使能位
0 = 屏蔽零点中断
1 = 使能零点中断

5.27 EPWMDIF 中断标志寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMDF			EPWM5DIF	EPWM4DIF	EPWM3DIF	EPWM2DIF	EPWM1DIF	EPWM0DIF
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 **EPWM5DIF：**EPWM5向下比较中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

Bit 4 **EPWM4DIF：**EPWM4向下比较中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

Bit 3 **EPWM3DIF：**EPWM3向下比较中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

Bit 2 **EPWM2DIF：**EPWM2向下比较中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

Bit 1 **EPWM1DIF：**EPWM1向下比较中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

Bit 0 **EPWM0DIF：**EPWM0向下比较中断使能位
0 = 未产生中断
1 = 产生中断（软件清零）

6 端口

6.1 数据寄存器 IOx（x = A/B/C/D/E/F）

	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	IOC7	IOC6	IOC5	IOC4	IOC3	IOC2	IOC1	IOC0
IOD	IOD7	IOD6	IOD5	IOD4	IOD3	IOD2	IOD1	IOD0
IOE	IOE7	IOE6	IOE5	IOE4	IOE3	IOE2	IOE1	IOE0
IOF			IOF5	IOF4	IOF3	IOF2	IOF1	IOF0
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

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/D/E/F）

	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	IOCOR7	IOCOR6	IOCOR5	IOCOR4	IOCOR3	IOCOR2	IOCOR1	IOCOR0
IODOR	IODOR7	IODOR6	IODOR5	IODOR4	IODOR3	IODOR2	IODOR1	IODOR0
IOEOR	IOEOR7	IOEOR6	IOEOR5	IOEOR4	IOEOR3	IOEOR2	IOEOR1	IOEOR0
IOFOR	-	-	IOFOR5	IOFOR4	IOFOR3	IOFOR2	IOFOR1	IOFOR0
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/C/D/E/F，y = 0-7）

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/D/E/F）

	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	OEC7	OEC6	OEC5	OEC4	OEC3	OEC2	OEC1	OEC0
OED	OED7	OED6	OED5	OED4	OED3	OED2	OED1	OED0
OEE	OEE7	OEE6	OEE5	OEE4	OEE3	OEE2	OEE1	OEE0
OEF	-	-	OEF5	OEF4	OEF3	OEF2	OEF1	OEF0
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/C/D/E/F，y = 0-7）

0 = 输入模式

1 = 输出模式

6.4 上拉控制寄存器 PUx（x = A/B/C/D/E/F）

	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	PUC7	PUC6	PUC5	PUC4	PUC3	PUC2	PUC1	PUC0
PUD	PUD7	PUD6	PUD5	PUD4	PUD3	PUD2	PUD1	PUD0
PUE	PUE7	PUE6	PUE5	PUE4	PUE3	PUE2	PUE1	PUE0
PUF			PUF5	PUF4	PUF3	PUF2	PUF1	PUF0
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/C/D/E/F，y = 0-7）

0 = 上拉关闭

1 = 上拉使能

6.5 下拉控制寄存器 PDx（x = A/B/C/D/E/F）

	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	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDD	PDD7	PDD6	PDD5	PDD4	PDD3	PDD2	PDD1	PDD0
PDE	PDE7	PDE6	PDE5	PDE4	PDE3	PDE2	PDE1	PDE0
PDF	-	-	PDF5	PDF4	PDF3	PDF2	PDF1	PDF0
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/C/D/E/F，y = 0-7）

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，IO 口将自动屏蔽输入功能（读该端口状态为 0），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/D/E/F）

	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	ANSC7	ANSC6	ANSC5	ANSC4	ANSC3	ANSC2	ANSC1	ANSC0
ANSD	ANSD7	ANSD6	ANSD5	ANSD4	ANSD3	ANSD2	ANSD1	ANSD0
ANSE	ANSE7	ANSE6	ANSE5	ANSE4	ANSE3	ANSE2	ANSE1	ANSE0
ANSF	-	-	ANSF5	ANSF4	ANSF3	ANSF2	ANSF1	ANSF0
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/C/D/E/F，y = 0-7）

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 IOxODS（x = A/B/C/D/E/F）

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOAODS	IOAODS7	IOAODS6	IOAODS5	IOAODS4	IOAODS3	IOAODS2	IOAODS1	IOAODS0
IOBODS	IOBODS7	IOBODS6	IOBODS5	IOBODS4	IOBODS3	IOBODS2	IOBODS1	IOBODS0
IOCODS	IOCODS7	IOCODS6	IOCODS5	IOCODS4	IOCODS3	IOCODS2	IOCODS1	IOCODS0
IODODS	IODODS7	IODODS6	IODODS5	IODODS4	IODODS3	IODODS2	IODODS1	IODODS0
IOEODS	IOEODS7	IOEODS6	IOEODS5	IOEODS4	IOEODS3	IOEODS2	IOEODS1	IOEODS0
IOFODS	-	-	IOFODS5	IOFODS4	IOFODS3	IOFODS2	IOFODS1	IOFODS0
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] **IOxODSy：**驱动能力选择（x = A/B/C/E/F，y = 0-7）

IOxODSy	IOxy 驱动电流选择
1	标准驱动(I_{OL1}\ I_{OH1})
0	小驱动(I_{OL2}\ I_{OH2})

Bit [7:0] **IODODSy：**IOD驱动能力选择（y = 0-7）

IODODSy	IODy 驱动电流选择
1	COM 口 N 驱动大电流(I_{OL3}\ I_{OH1})
0	小驱动(I_{OL4}\ I_{OH2})

注：对于不同的应用，用户需参考 23.3 IO 口拉灌电流特性章节。

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

	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	IOCIPS7	IOCIPS6	IOCIPS5	IOCIPS4	IOCIPS3	IOCIPS2	IOCIPS1	IOCIPS0
IODIPS	IODIPS7	IODIPS6	IODIPS5	IODIPS4	IODIPS3	IODIPS2	IODIPS1	IODIPS0
IOEIPS	IOEIPS7	IOEIPS6	IOEIPS5	IOEIPS4	IOEIPS3	IOEIPS2	IOEIPS1	IOEIPS0
IOFIPS	-	-	IOFIPS5	IOFIPS4	IOFIPS3	IOFIPS2	IOFIPS1	IOFIPS0
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/C/D/E/F，y = 0-7）

0 = SMT

1 = 1.5 V 或 1/2VDD（由FLIPCR寄存器控制）

注：IOFIPS[7:6] 复位值为 00b。

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/C/D/E/F，y = 0-7）

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，USARTx(x=0、1、2、3、4、5)、PWM0、PWM1x(x=0、1)、EPWMx(x=0、1、2、3、4、5)、TxG（x=2、3、4）、FB x(x=0、1)、ADET、I2C SDA、I2C SCL 可以映射到任意端口，SPI 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	优先级顺序	复用功能	复位后	复用说明
MPPWM0	1	PWM0	IOC[3]	全端口映射
MPSPI	2	MISO/ MOSI/ SCK/ SSN	IOB[7:4]	按端口整体选择
MPTX0	3	USART TX0	IOA[5]	全端口映射
MPTX1	4	USART TX1	IOB[3]	全端口映射
MPTX2	5	USART TX2	IOC[1]	全端口映射
MPTX3	6	USART TX3	IOE[4]	全端口映射
MPTX4	7	USART TX4	IOD[7]	全端口映射
MPTX5	8	USART TX5	IOA[2]	全端口映射
MPSDA	9	I2C SDA	IOC[6]	全端口映射
MPSCL	10	I2C SCL	IOC[5]	全端口映射
MPPWM10	11	PWM10	IOF[4]	全端口映射
MPPWM11	12	PWM11	IOF[5]	全端口映射
MPEPWM0	13	EPWM0	IOF[0]	全端口映射
MPEPWM1	14	EPWM1	IOF[1]	全端口映射
MPEPWM2	15	EPWM2	IOF[2]	全端口映射
MPEPWM3	16	EPWM3	IOF[3]	全端口映射
MPEPWM4	17	EPWM4	IOE[3]	全端口映射
MPEPWM5	18	EPWM5	IOE[2]	全端口映射
MPRX0	-	USART RX0	IOA[6]	全端口映射
MPRX1	-	USART RX1	IOB[1]	全端口映射
MPRX2	-	USART RX2	IOC[0]	全端口映射
MPRX3	-	USART RX3	IOE[5]	全端口映射
MPRX4	-	USART RX4	IOD[6]	全端口映射
MPRX5	-	USART RX5	IOA[3]	全端口映射
MPT2G	-	T2G	IOB[2]	全端口映射
MPT3G	-	T3G	IOC[2]	全端口映射
MPT4G	-	T4G	IOD[5]	全端口映射
MPFB0	-	FB0	IOA[6]	全端口映射
MPFB1	-	FB1	IOA[7]	全端口映射
MPADET	-	ADET	IOC[7]	全端口映射

6.9.2 全端口映射控制

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

Bit [5:4] **PORTSEL**[2:0]：映射端口选择位

PORTSEL[1:0]	IO 端口组选择
000	IOA
001	IOB
010	IOC
011	IOD
10x	IOE
11x	IOF

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

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

注：如需将 USART TX 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 SPI 复用控制

SPI 复用以四路为一组,下面的引脚为 SPI 的引脚,其他三路按顺序向下排。比如 MPSPI 设置为 00h，MISO 从 IOA0 输入，输出 MOSI 从 IOA1 输入，输出 SCK 从 IOA2 输入，输出 SSN 从 IOA3 输入，输出，当 MPSPI 设置为 61h 时，由于 IOF 口只到 IOF5 因此只有 IOF[3:0]可以使用。

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MPSPI	-	PORTSEL[2:0]			-	-	-	PINSEL2
读/写	-	R/W	R/W	R/W	-	-	-	R/W
复位后	-	0	0	0	-	-	-	0

Bit [5:4] **PORTSEL[2:0]:** SPI 输出端口 IOx 选择(x = A、B、C、D、E、F)

MPSPI [6:4]	IO 口选择
000	IOA
001	IOB
010	IOC
011	IOD
100	IOE
101	IOF

Bit 2 **PINSEL2:** SPI 输出端口高低位选择

0 = MISO/ MOSI/ SCK/ SSN 输出端口依次为 IOx0/1/2/3

1 = MISO/ MOSI/ SCK/ SSN 输出端口依次为 IOx4/5/6/7

注：（1）由于 IOF 端口无 IOF6&IOF7 口，故当 SPI 无法选择 IOF 高位。

（2）如需将 MPSPI 端口映射到 IOB[7:4]，则可如下操作：

```
MPSPI = 0x11;    //MISO 在 IOB[4]  MOSI 在 IOB[5]  SCK 在 IOB[6]  SSN 在 IOB[7]
```

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，具有绿色模式下唤醒功能。

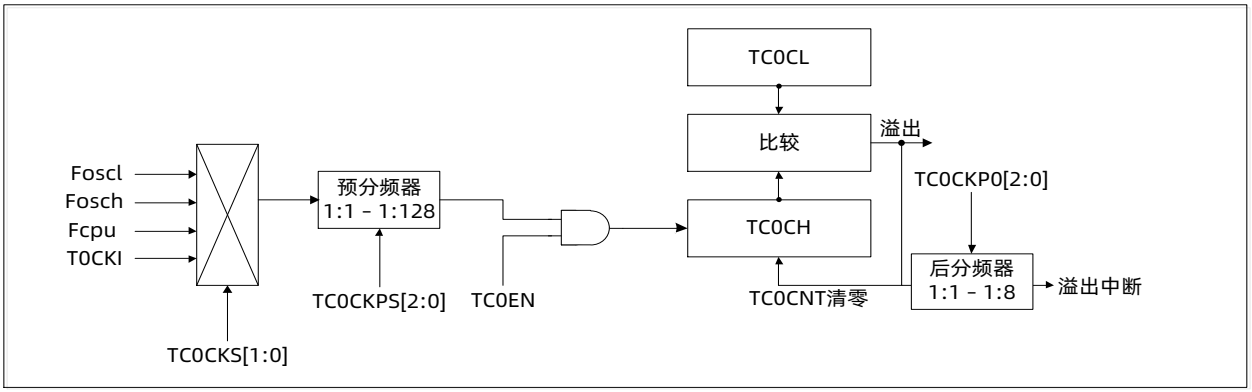
在 8 位模式下，TC0CL 作为 TC0 的周期寄存器器，TC0 使能后，TC0CH 递加，当 TC0CH 与 TC0CL 数值相等，TC0 溢出，将 TC0CH 清零重新开始计数，同时将中断标志位 TC0IF 置 1。

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。

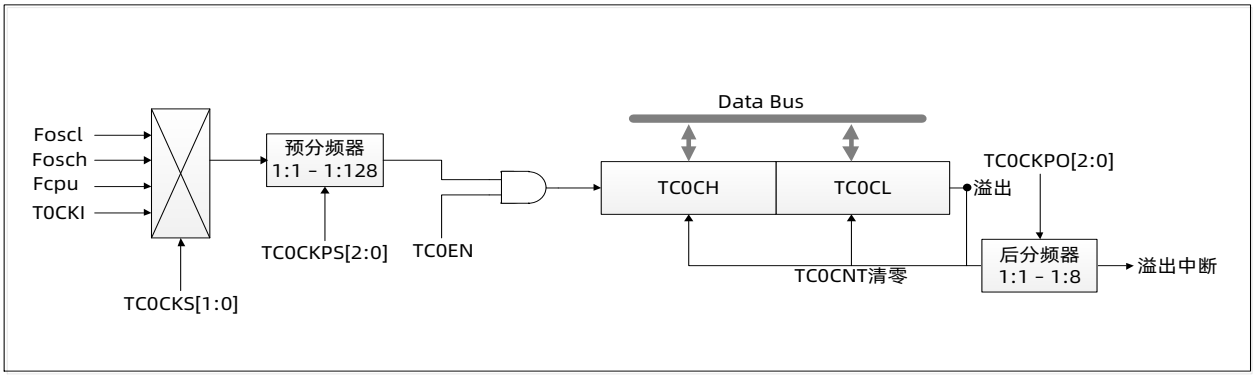
- 可选择时钟源：高频系统时钟 F_{OSCH}、低频系统时钟 F_{OSCL}、指令时钟 F_{CPU} 或外部时钟 T0CKI
- 可选择 8 位模式和 16 位模式，使能 PWM0 时，只能在 8 位模式
- 预分频比多级可选，最大可选择 1:128
- 后分频比多级可选，最大可选择 1:8
- 溢出中断功能
- 溢出中断唤醒功能（当输入频率选择 F_{OSCH} 或 F_{OSCL} 时，若所选择的时钟源振荡器一直工作，此时 TC0 在休眠状态下依然工作，溢出中断可唤醒 CPU）

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 = 32MHz

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

注：如果使能了 TC0，在高频停止进绿色或休眠模式前，需要关闭 TC0。否则高频时钟将无法关闭。

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 PWM0CR PWM 控制寄存器

PWM0 可以复用到任意 IO 口。

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM0CR	PWM0EN	PWM0OE	PWM0M	-	-	-	-	-
读/写	R/W	R/W	R/W	-	-	-	-	-
复位后	0	0	0	-	-	-	-	-

Bit 7 **PWM0EN**：PWM0使能位
0 = 关闭PWM
1 = 使能PWM（TC0必须选择8位模式）

Bit 6 **PWM0OE**：PWM0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 **PWM0M**：PWM0输出极性控制
0 = PWM高电平有效
1 = PWM低电平有效

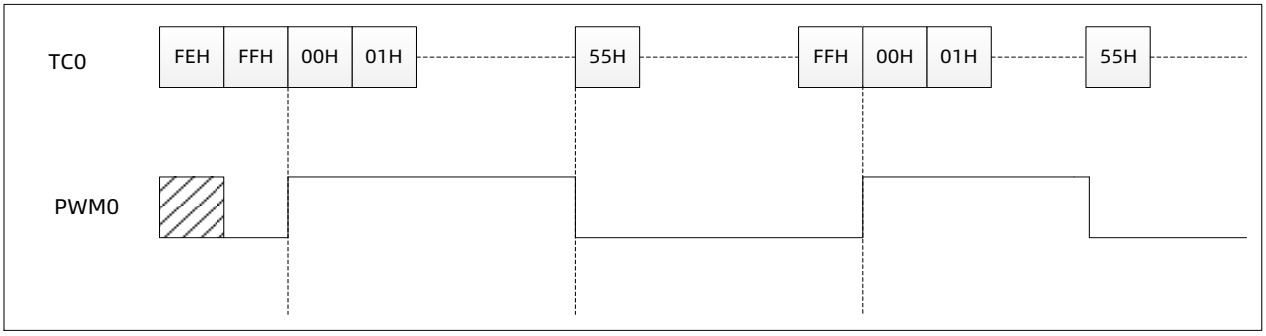
7.7 PWM0D PWM 占空比寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM0D	PWM0D[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] **PWM0D**：PWM0占空比控制寄存器

7.8 PWM0 波形实例

例：PWM0CR = 11100000B，PWM0D = 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，具有绿色模式下唤醒功能。

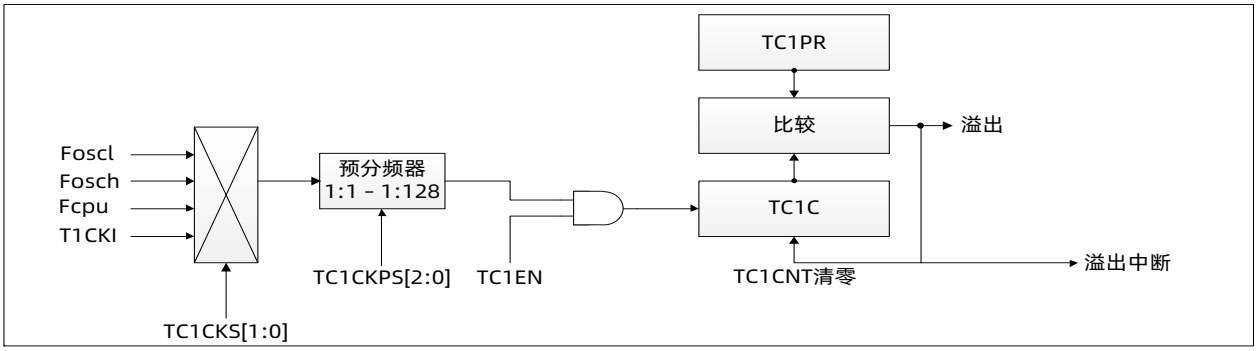
向上计数模式，TC1 使能后，TC1C 递增，当 TC1C 计数值与 TC1PR 相等时，TC1 溢出，将 TC1CH 清零重新开始计数，同时将中断标志位 T1IF 置 1。

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 在绿色模式下依然工作，溢出中断可唤醒 CPU。）
- 可驱动 WS2812 基于 TC1 时基。

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

注：中心对齐模式下，TC1MOD0 为只读模式，表示目前定时器的状态（增或减）。

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	x	x	x	x	x	x	x	x

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	x	x	x	x	x	x	x	x

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	LEDFLG	LEDBSY0	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发送标志（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

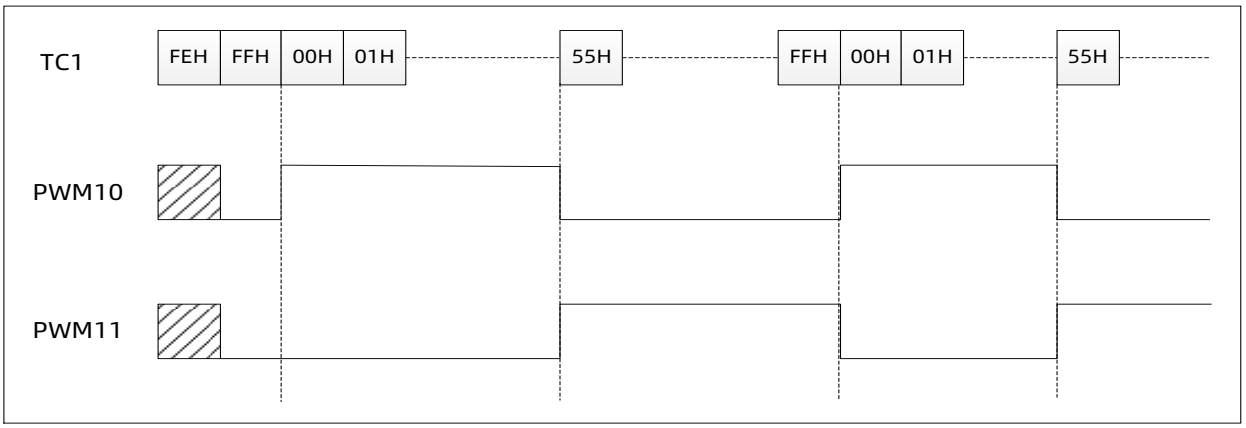
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM11DL	PWM11DL[7:0]							
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PWM11DL[7:0]**：后死区宽度设置

8.10 PWM1 波形示例

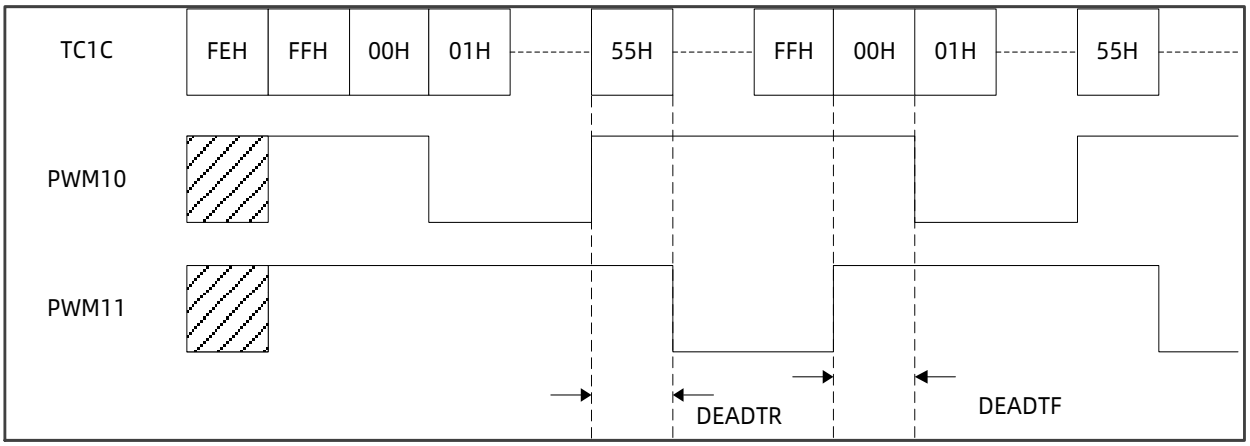
8.10.1 互补 PWM1 输出

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



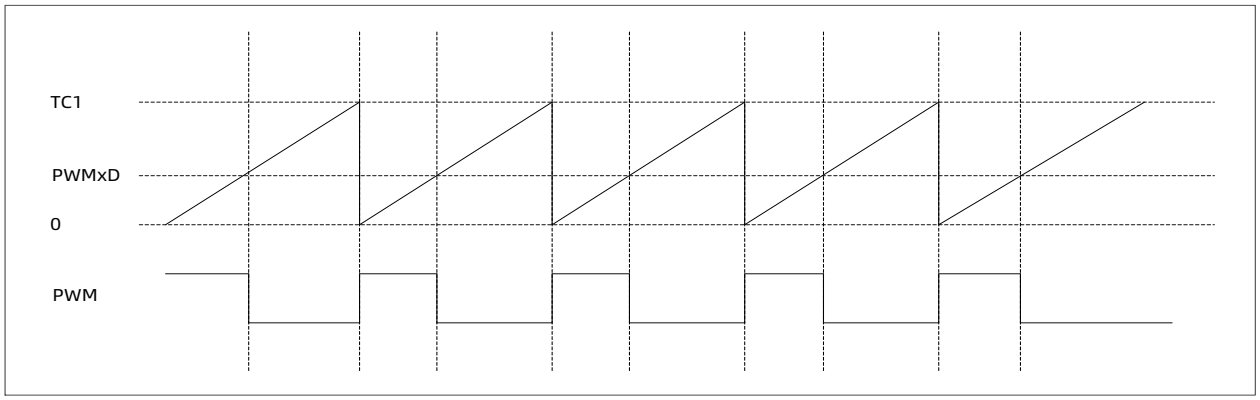
8.10.2 带死区的互补 PWM1 输出

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

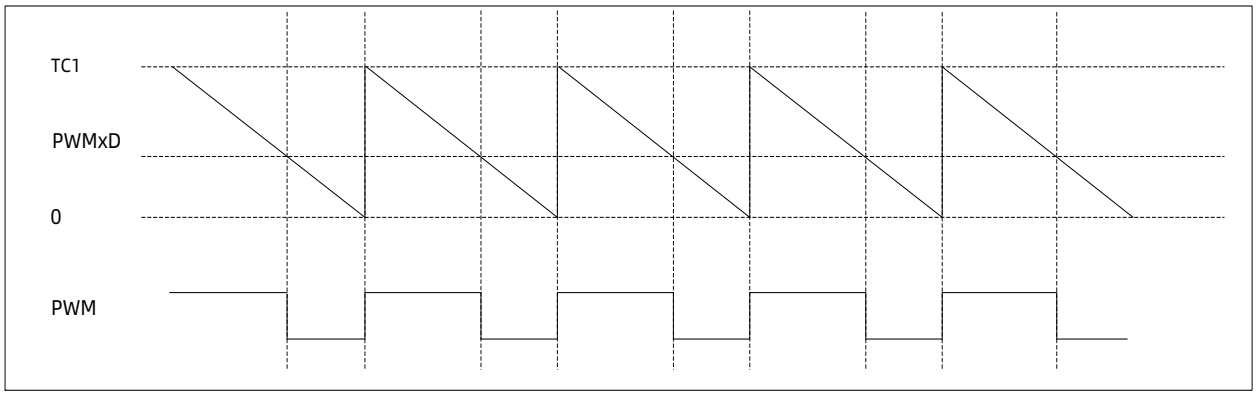


8.10.3 PWM 波形图

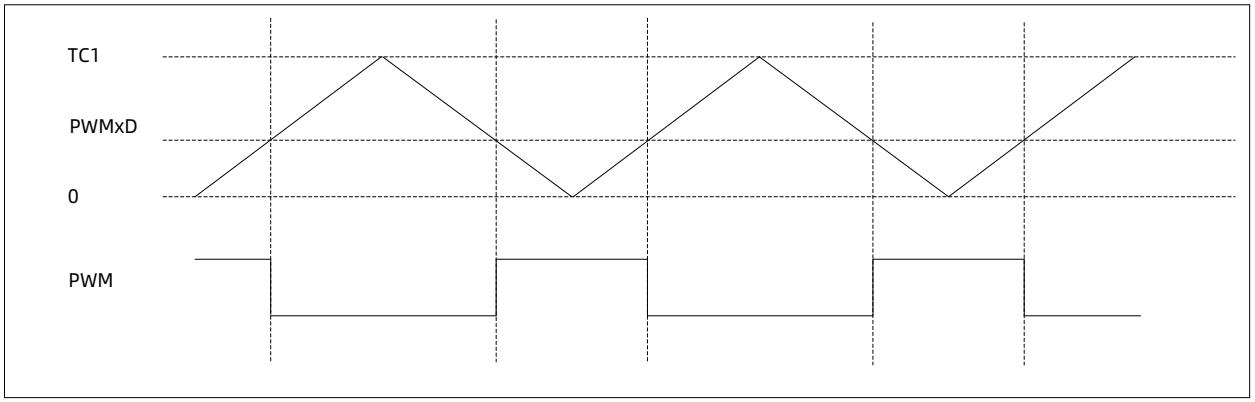
TC1 递增模式



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;
```

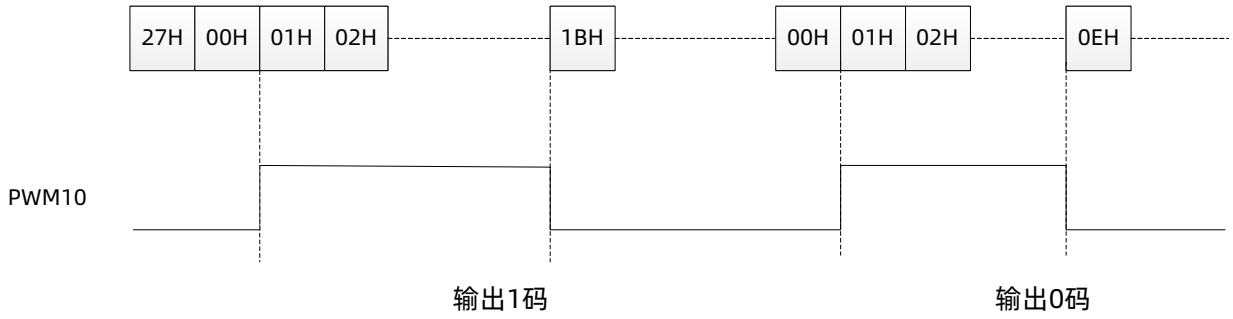
8.10.4 级联 LED 驱动

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；

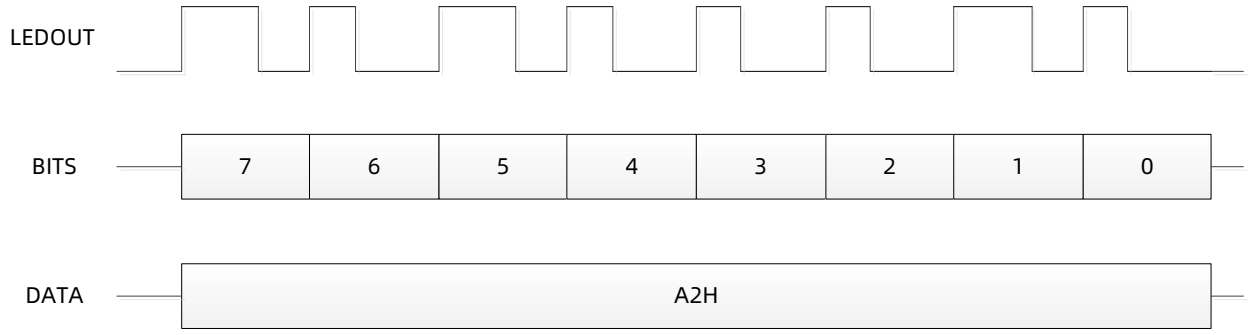
LED 使能，PWM11 自动关闭。

例： F_{OSC} = 32MHz，TC1PR=27H，PWM10DL=0DH，PWM10DH=1AH，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。

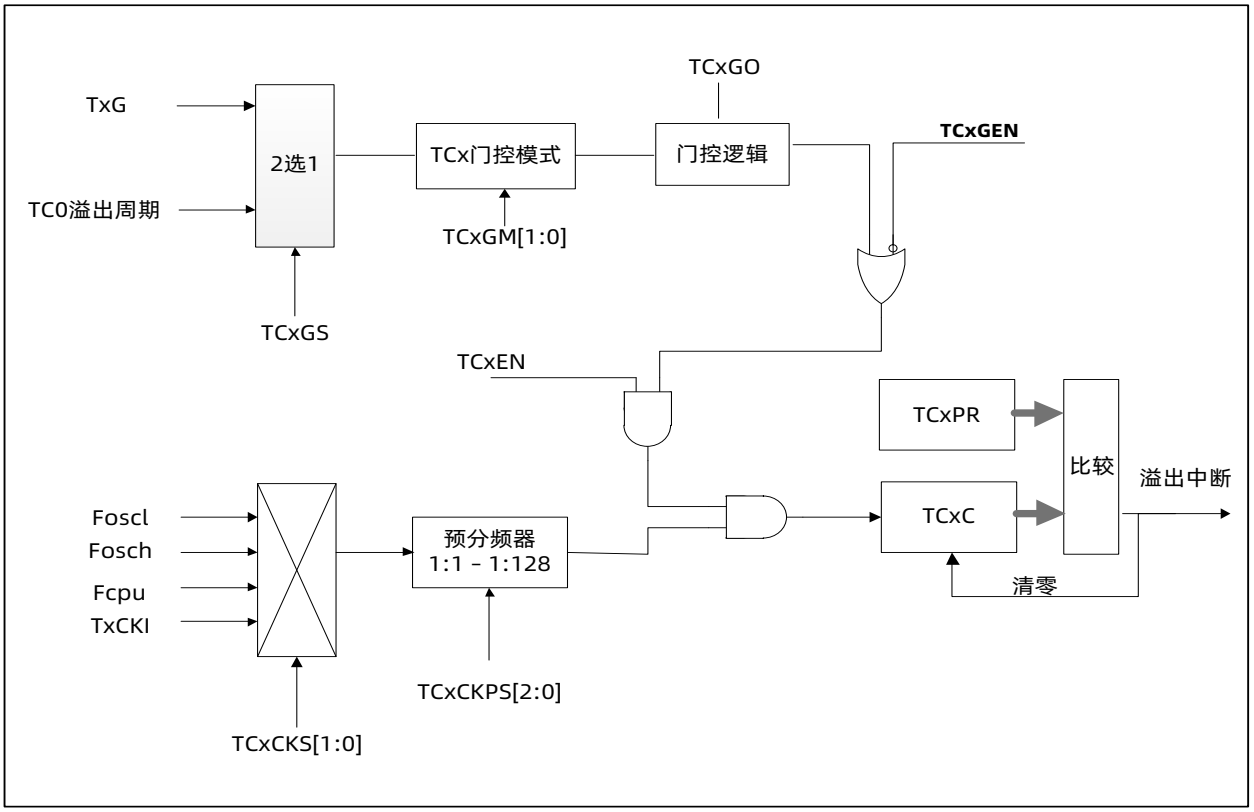
9 定时器 2/3/4（TC2/3/4）

9.1 概述

TC2/3/4 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，具有绿色模式下唤醒功能。

- 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 TxCKI(x=2、3、4)
- 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 TxG(x=2、3、4)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 TCx(x=2、3、4)在绿色模式下依然工作，溢出中断可唤醒 CPU。）

TCx(x=2、3、4)框图



9.2 TxCR 控制寄存器(x=2、3、4)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TxCR	TCxEN	-	-	TCxCKS[1:0]		TCxCKPS[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 **TCxEN：** TCx模块使能位

0 = 关闭TCx

1 = 使能TCx

Bit [4:3] **TCxCKS[1:0]：** TCx时钟源选择位

TCxCKS[1:0]	TCx 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	TxCKI

Bit [2:0] **TCxCKPS[2:0]：** TCx预分频比选择

TCxCKPS[2:0]	TCx 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 TCxCL 计数器低位(x=2、3、4)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TCxCL	TCxCL[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 TCxCH 计数器高位(x=2、3、4)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TCxCH	TCxCH[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 TCxPRL 周期寄存器低位(x=2、3、4)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TCxPRL	TCxPRL[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

9.6 TCxPRH 周期寄存器高位(x=2、3、4)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TCxPRH	TCxPRH[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

9.7 TCxGCR 门控控制寄存器(x=2、3、4)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TCxGCR	TCxGEN	TCxGO	-	-	-	TCxGS	TCxGM[1:0]	
读/写	R/W	R/W	-	-	-	R/W	R/W	R/W
复位后	0	0	-	-	-	0	0	0

Bit 7 **TCxGEN：** TCx门控模式使能位

0 = 关闭门控功能，TCx是否计数仅受TCxGEN控制

1 = 使能门控功能

Bit 6 **TCxGO：** 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 **TCxGS：** TCx门控源选择位(x=2、3、4)

TCxGS[1:0]	TCx 门控选择
0	TxG (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] **TCxGM[1:0]：** TCx门控模式选择位（只有TxG可选择）

TCxGM[1:0]	TCx 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

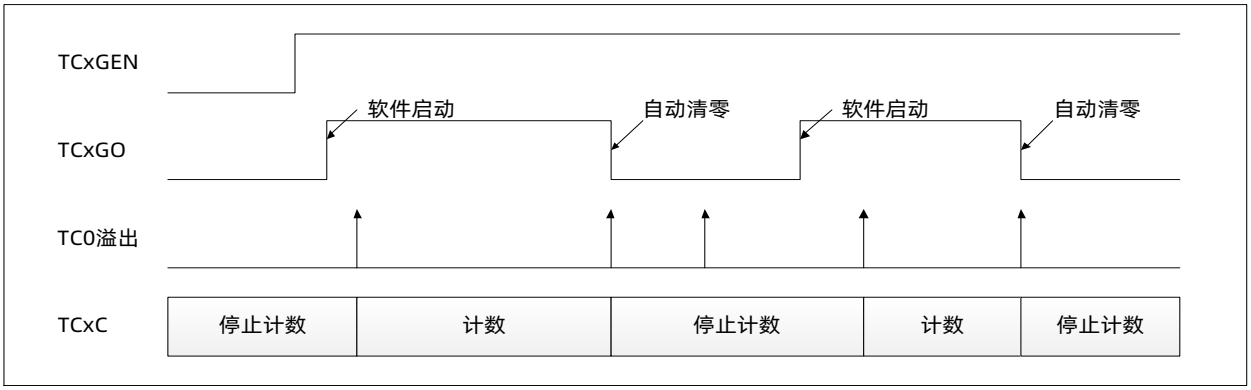
注：

（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。

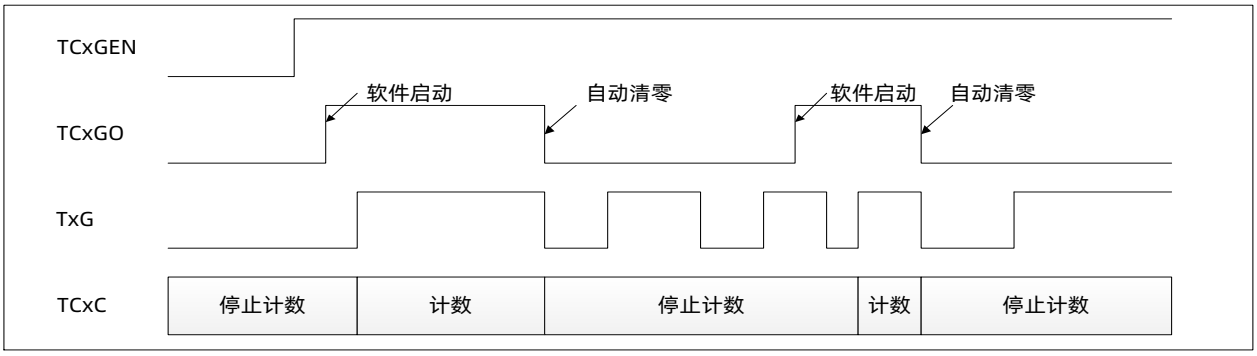
9.7.1 门控-TC0 溢出周期

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。



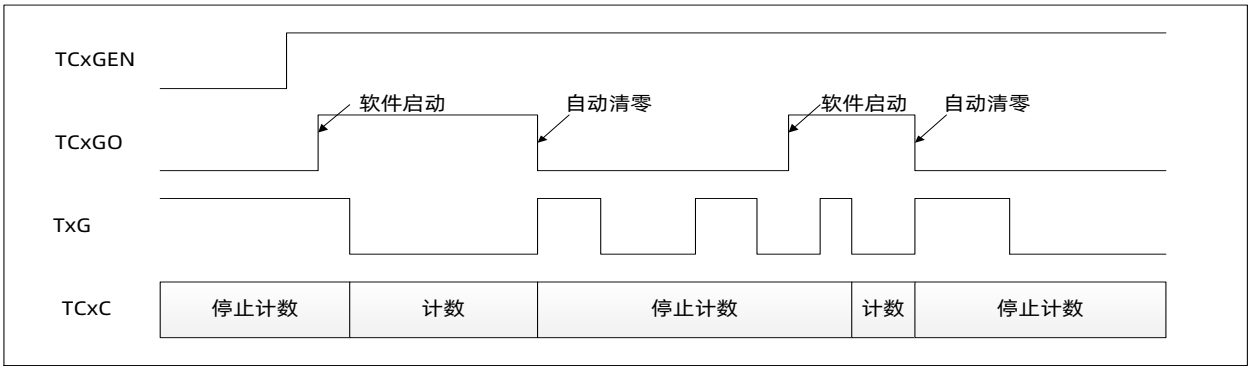
9.7.2 门控-上升沿到下降沿模式

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。



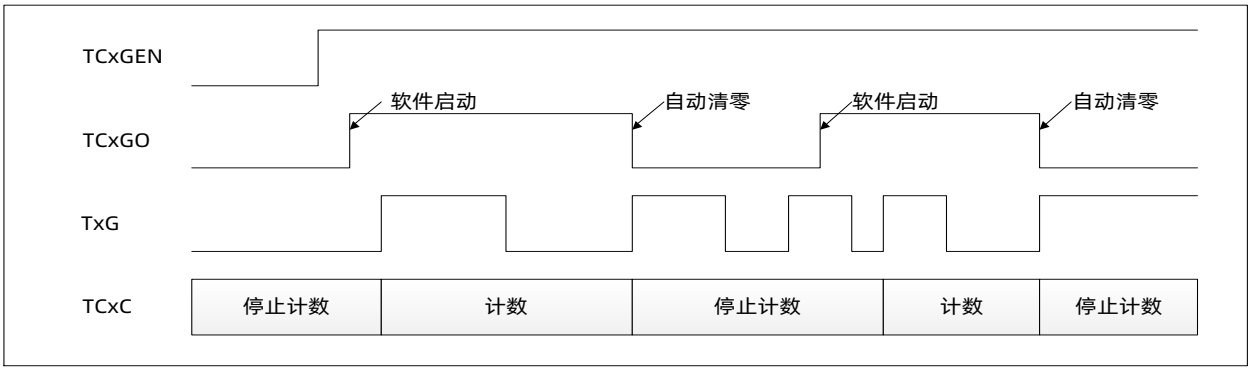
9.7.3 门控-下降沿到上升沿模式

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。



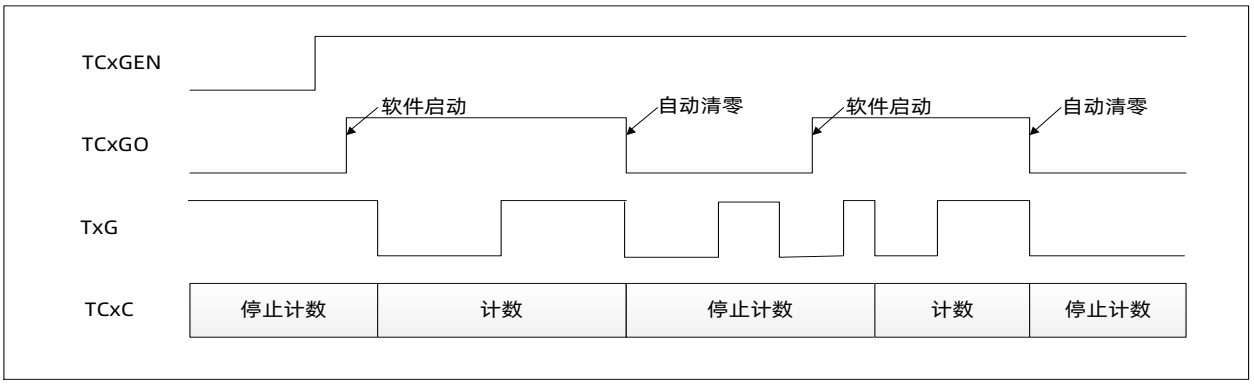
9.7.4 门控-上升沿到上升沿模式

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 EPWM 模块

10.1 概述

增强型 PWM 模块具有 6 路 PWM 发生器,可以配置成 3 组同步 PWM 输出，或 3 对分别带有死区编程发生器的互补 PWM 输出，其中 EPWM0-EPWM1，EPWM2-EPWM3，EPWM4-EPWM5 分别为一对。

每一对 EPWM 共用一个 16 位的 周期寄存器、各自可以配置 16 位的占空比寄存器（比较数据寄存器和向下比较数据寄存器)，用以配置 EPWM 输出的周期和调节占空比。且每一对 EPWM 具有独立的时钟分频控制寄存器，每一对 EPWM 共用可设置最大 128 预分频器。

每路 EPWM 可配置成边沿对齐计数或中心对齐计数模式,中心对齐计数模式下还可以设置为对称计数或非对称计数两种方式。每路 EPWM 可设置单次模式（产生一个 PWM 信号周期）或者自动装载（连续输出 EPWM 波形）输出，其输出极性也可通过输出极性控制器设置。

增强型 PWM 还支持掩码输出功能、硬件刹车保护功能、中断功能。6 路 PWM 发生器总共提供 25 个中断标志，包括零点中断、向上比较中断、周期中断、向下比较中断、刹车中断，这些中断共用一个中断向量入口。

10.2 EPWMCON 控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMCON	EPWMEN	-	-	EPWMMODE[1:0]		GROUPEN	CNTMOD[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 **EPWMEN**：EPWM模块总使能位

0 = 关闭EPWM模块

1 = 使能EPWM模块

Bit [4:3] **EPWMMODE[1:0]：**PWM的模式控制位：

PWMMODE[1:0]	模式选择
00	独立模式
01	互补模式
10	同步模式

同步模式下，EPWM1,3,5的周期占空比及时钟分频由EPWM0，2，4决定。除了输出控制位EPWMxOE外全部不再受自己控制器控制。

Bit [2] **GROUPEN**：EPWM成组功能使能

1 = EPWM0控制PWM0，EPWM2，EPWM4。EPWM1控制EPWM1，EPWM3，EPWM5.

0 = 所有EPWM通道信号相互独立。

（EPWM0，2，4输出相同，EPWM1，3，5输出相同。除了输出控制位EPWMxOE外全部不再受自己控制器控制。）

Bit [1:0] **CNTMOD[1:0]**：计数器模式选择位

CNTMOD[1:0]	计数器模式选择
00	递增模式
01	递减模式
10	中心对齐对称模式
11	中心非对齐对称模式

10.3 EPWM 输出控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMOE	-	-	EPWM5OE	EPWM4OE	EPWM3OE	EPWM2OE	EPWM1OE	EPWM0OE
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	x	x	x	x	x	x

Bit 5 **EPWM5OE**：EPWM5输出控制
0 = EPWM信号不从管脚输出，管脚用做IO
1 =EPWM信号从管脚输出

Bit 4 **EPWM4OE**：EPWM4输出控制
0 = EPWM信号不从管脚输出，管脚用做IO
1 = EPWM信号从管脚输出

Bit 3 **EPWM3OE**：EPWM3输出控制
0 =EPWM信号不从管脚输出，管脚用做IO
1 =EPWM信号从管脚输出

Bit 2 **EPWM2OE**：EPWM2输出控制
0 =EPWM信号不从管脚输出，管脚用做IO
1 = EPWM信号从管脚输出

Bit 1 **EPWM1OE**：EPWM1输出控制
0 = EPWM信号不从管脚输出，管脚用做IO
1 =EPWM信号从管脚输出

Bit 0 **EPWM0OE**：EPWM0输出控制
0 =EPWM信号不从管脚输出，管脚用做IO
1 = EPWM信号从管脚输出

10.4 EPWM0/1 时钟预分频控制寄存器 EPWM01CKPS

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWM01CKPS	-	-	-	-	-	EPWM01CKPS[2:0]		
读/写	-	-	-	-	-	R/W	R/W	R/W
复位后	-	-	-	-	-	0	0	0

Bit [2:0] **EPWM01CKPS [2:0]**：EPWM通道0/1预分频控制位

EPWM01CKS[2:0]	EPWM01 预分频
000	F_{osc}
001	F_{osc} /2
010	F_{osc} /4
011	F_{osc} /8
100	F_{osc} /16
101	F_{osc} /32
110	F_{osc} /64
111	F_{osc} /128

10.5 EPWM2/3 时钟预分频控制寄存器 EPWM23CKPS

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWM23CKPS						EPWM23CKPS[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 [2:0] **EPWM23CKPAS [2:0]**：EPWM通道2/3预分频控制位

EPWM23CKS[2:0]	EPWM23 预分频
000	F_{osc}
001	F_{osc} /2
010	F_{osc} /4
011	F_{osc} /8
100	F_{osc} /16
101	F_{osc} /32
110	F_{osc} /64
111	F_{osc} /128

10.6 EPWM4/5 时钟预分频控制寄存器 EPWM45CKPS

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWM45CKPS						EPWM45CKPS[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 [2:0] **EPWM45CKPS [2:0]**：EPWM通道4/5预分频控制位

EPWM45CKS[2:0]	EPWM45 预分频
000	F_{osc}
001	F_{osc} /2
010	F_{osc} /4
011	F_{osc} /8
100	F_{osc} /16
101	F_{osc} /32
110	F_{osc} /64
111	F_{osc} /128

10.7 EPWMLOADEN 数据加载使能控制位

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMLOADEN						EPWM45LD	EPWM23LD	EPWM01LD
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 [2:0] **EPWMxLD**：EPWM通道n的数据加载使能位(x=01,23,45)（加载完成后硬件清零）;

0 = 不使能加载。不建议置1后软件清0，可能出现无法预料的情况。

1 = 使能加载周期，占空比数据(EPWMPx、EPWMDx、EPWMDDx)。仅在EPWMxCNTM=1时使用。建议在EPWMxLD=0时写周期占空比数据缓存，若在EPWMxLD=1时写可能出现无法预料的情况。

10.8 EPWMMPINV 输出极性控制

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMMPINV			EPWM5M	EPWM4M	EPWM3M	EPWM2M	EPWM1M	EPWM0M
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 [5:0] **EPWMxM**：PWM输出极性控制（x=0-5)

0 = PWM高电平有效，空闲状态下为低

1 = PWM低电平有效，空闲状态下为高

进入互补时通道1,3,5极性自动与通道0,2,4相反，且EPWMxM（x=1,3,5）的控制与其他相反

10.9 EPWMCNTM 计数模式寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMCNTM						EPWM45CNTM	EPWM23CNTM	EPWM01CNTM
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 [2:0] **EPWMxCNTM**：EPWMx计数模式控制位（x=01,23,45)

0 = 单次模式

1 = 自动加载模式

10.10 EPWMCNTE 计数器使能控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMCNTE						EPWM45CNTE	EPWM23CNTE	EPWM01CNTE
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 [2:0] **EPWMxCNTE**：PWMx计数模式使能控制位（x=01,23,45)

1 = EPWMx计数器开启（EPWMx开始工作）

0 = EPWMx计数器停止（软件写0则计数器停止并清掉计数器值）

（刹车触发则该位硬件清0；单次模式完成该位硬件清0）

置1后，在第一个计数器周期会从周期、占空比数据寄存器缓存加载数据，此时EPWM输出为空闲。

10.11 EPWMCNTCLR 计数器模式控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMCNTCLR						EPWM45CNTCLR	EPWM23CNTCLR	EPWM01CNTCLR
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 [2:0] **EPWMxCNTCLR**：PWM通道n计数器清零控制位(x=01,23,45)（硬件自动清零）;

0 = 写0无效

1 = EPWMx计数器清0

在EPWM输出过程中清零计数器可能导致非期望的EPWM输出。

10.12 EPWMP 周期数据寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMPxH	EPWMPxH [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] **EPWMPxH**：PWM周期寄存器高8位(x=01,23,45)；
具有一级缓存，读为缓存值

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMPxL	EPWMPxL[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] **EPWMPxL**：PWM周期寄存器低8位(x=01,23,45)；
具有一级缓存，读为缓存值

10.13 EPWMD 比较数据寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMDxH	EPWMDxH [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] **EPWMDxH**：PWM占空比寄存器高8位(x=0,1,2,3,4,5)；
具有一级缓存，读为缓存值

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMDxL	EPWMDxL[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] **EPWMDxL**：PWM占空比寄存器低8位(x=0,1,2,3,4,5)；
具有一级缓存，读为缓存值

10.14 EPWMDD 向下比较数据寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMDDxH	EPWMDDxH [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] **EPWMDDxH**：PWM向下比较寄存器高8位(x=0,1,2,3,4,5)；
具有一级缓存，读为缓存值
仅在中心非对齐对称模式（EPWMCON[1:0]=2'B11）下使用

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMDDxL	EPWMDDxL[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] **EPWMDDxL**：PWM向下比较寄存器低8位(x=0,1,2,3,4,5)；
具有一级缓存，读为缓存值
仅在中心非对齐对称模式（EPWMCON[1:0]=2'B11）下使用

10.15 EPWMDT 死区延时数据寄存器

当 EPWM 设置成互补模式的时候。EPWMxD 作为比较数据寄存器，EPWMyD 作为死区寄存器，EPWMyDH 是后死区，EPWMyDL 是前死区。（x=0，2，4.y=1，3，5）

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMD1H	EPWMD1H[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] **EPWM1DH：** 互补时 EPWM01 的后死区宽度设置

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMD1L	EPWMD1L[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] **EPWMD1L：** 互补时 EPWM01 的前死区宽度设置

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMD3H	EPWMD3H[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] **EPWMD3H:** 互补时 EPWM23 的后死区宽度设置

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMD3L	EPWMD3L [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] **EPWMD3L：** 互补时 EPWM23 的前死区宽度设置

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMD5H	EPWMD5H [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] **EPWMD5H：** 互补时 EPWM45 的后死区宽度设置

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMD5L	EPWMD5L [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] **EPWMD5L：** 互补时 EPWM45 的前死区宽度设置

10.16 EPWMMASKE 掩码控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMMASKE			EPWM5MASKE	EPWM4MASKE	EPWM3MASKE	EPWM2MASKE	EPWM1MASKE	EPWM0MASKE
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 [5:0] **EPWMxMASKE**：EPWMx掩码控制使能位（x=0-5）

0 = EPWMx通道禁止掩码数据输出；

1 = EPWMx通道使能掩码数据输出；（正常输出PWM波形）

10.17 EPWMMASKD 掩码数据寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMMASKD			EPWM5MASKD	EPWM4MASKD	EPWM3MASKD	EPWM2MASKD	EPWM1MASKD	EPWM0MASKD
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 [5:0] **PWMxMASKD**：PWMx掩码数据位（x=0-5）

0 = PWMx输出为低

1 = PWMx输出为高

10.18 EPWMFBKC 刹车控制寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMFBKC	PWMFBIE	PWMFBF	PWM5FBCCE	PWMFBKSW	PWMFB1ES	PWMFB0ES	PWMFB1EN	PWMFB0EN
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 **EPWMFBIE**:PWM 刹车中断屏蔽位;
1 = 使能中断;
0 = 禁止中断。

Bit6 **EPWMFBF**:PWM 刹车标志位（写0清除）;
1 = 产生了刹车操作（PWM输出刹车数据寄存器的值）;
0 = 没有产生刹车操作。
从刹车中断恢复流程：确认所有刹车源无效，清除刹车标志，使能计数器

Bit5 **EPWMFBCCE**:PWM 刹车时是否清掉所有通道计数器选择位;
1=刹车时计数器的值不会清掉;
0=刹车时清掉通道计数器值。

Bit4 **EPWMFBKSW**:PWM 软件刹车信号启动位;
1=PWM产生软件刹车信号;
0=禁止。

Bit3 **EPWMFB1ES**:PWM 外部硬件刹车通道（FB1）触发电平选择位;
1=高电平;
0=低电平。

Bit2 **EPWMFB0ES**:PWM 外部硬件刹车通道（FB0）触发电平选择位;
1=高电平;
0=低电平。

Bit1 **EPWMFB1EN**:PWM 外部硬件刹车通道（FB1）使能位;
1=使能;
0=禁止。

Bit0 **PWMFB0EN**:PWM 外部硬件刹车通道（FB0）使能位;
1=使能;
0=禁止。

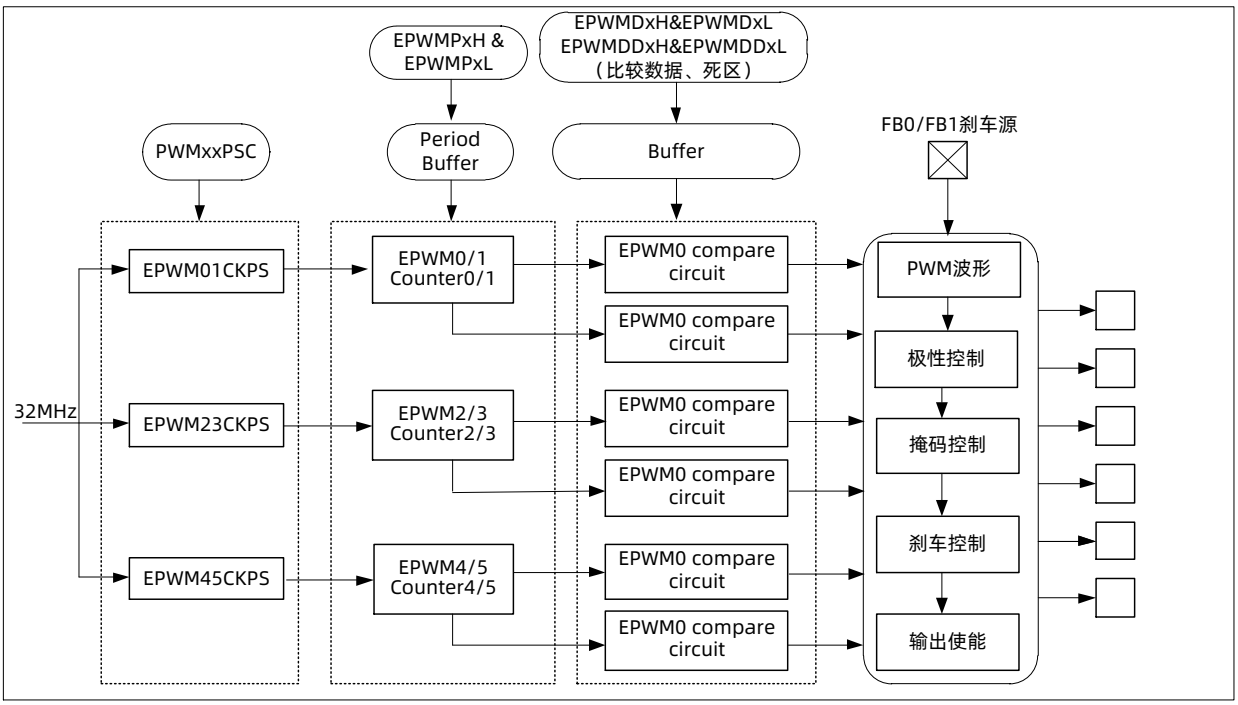
10.19 EPWM 刹车数据寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EPWMFBKD			EPWM5FBKD	EPWM4FBKD	EPWM3FBKD	EPWM2FBKD	EPWM1FBKD	EPWM0FBKD
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 [5:0] **EPWMxFBKD**：PWMx刹车数据位（x=0-5）
0 = PWMx输出为低
1 = PWMx输出为高

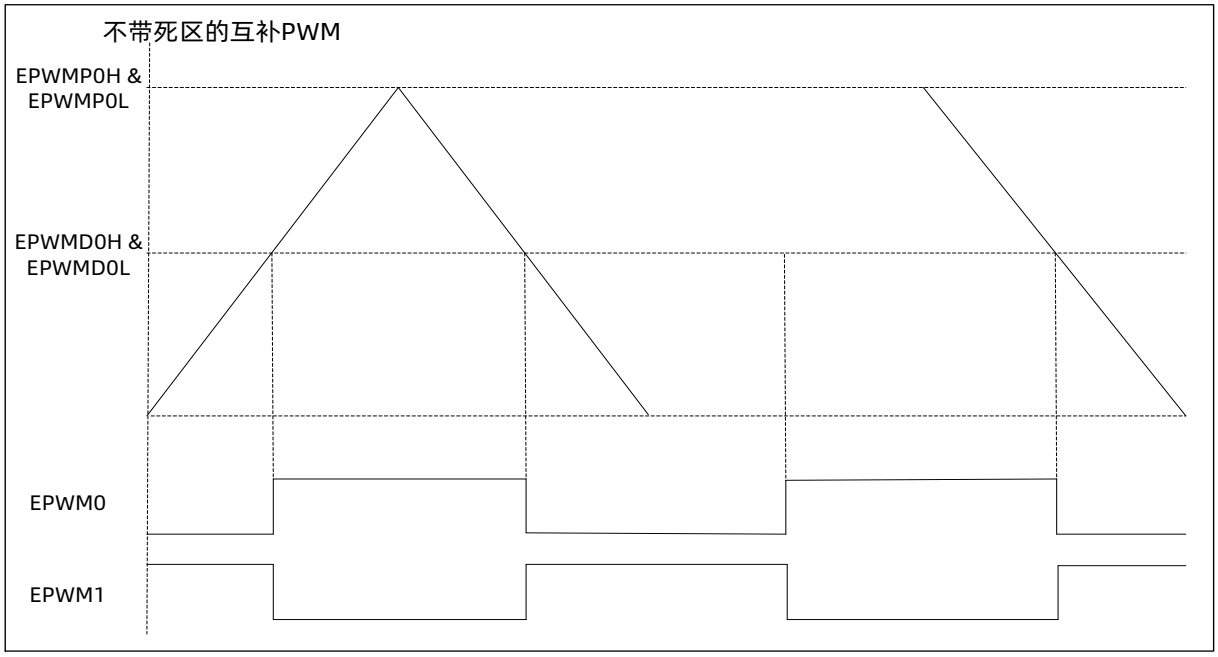
10.20 EPWMx(x=0/1/2/3/4/5)波形示例

10.20.1 功能描述

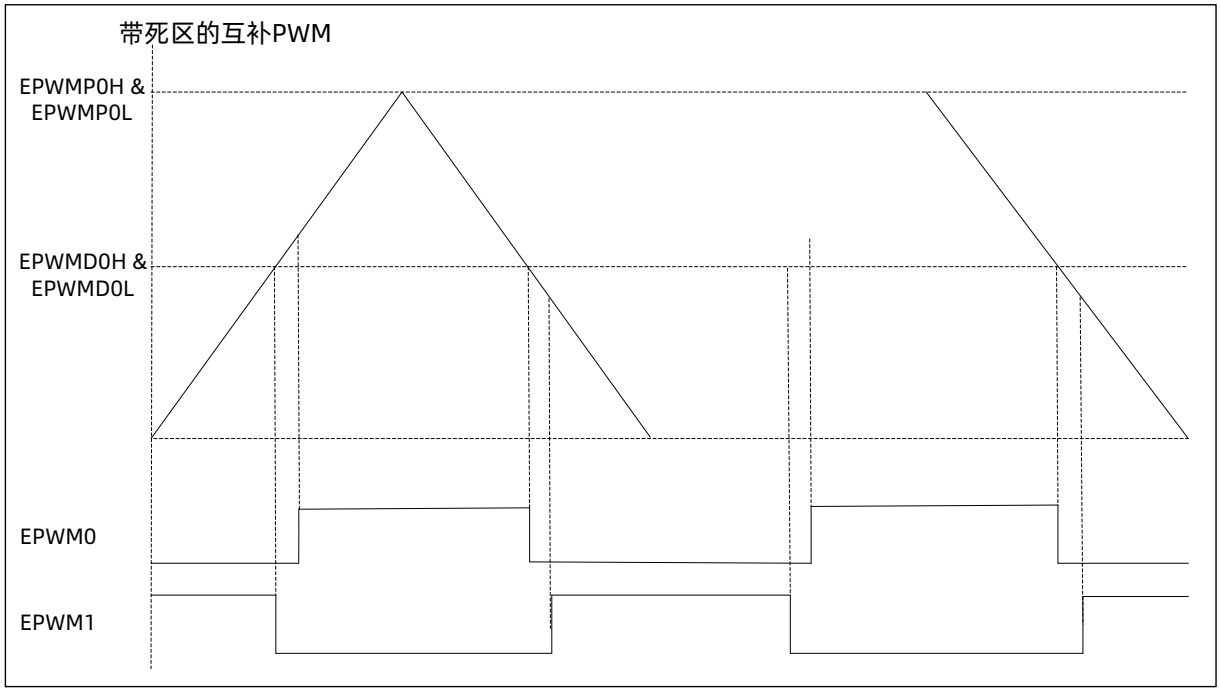


10.20.2 互补 PWMx(x=0/1/2/3/4/5)输出

不带死区

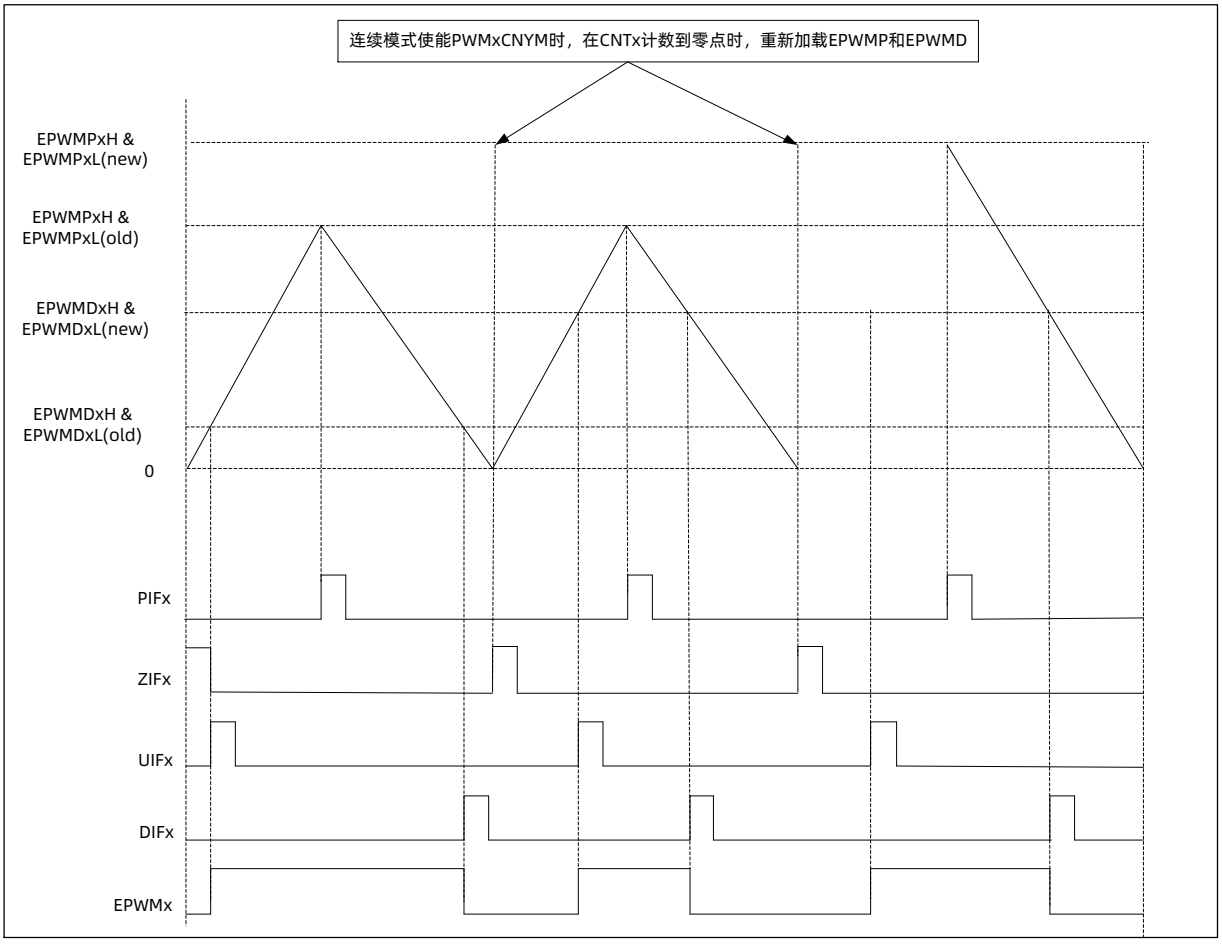


带死区

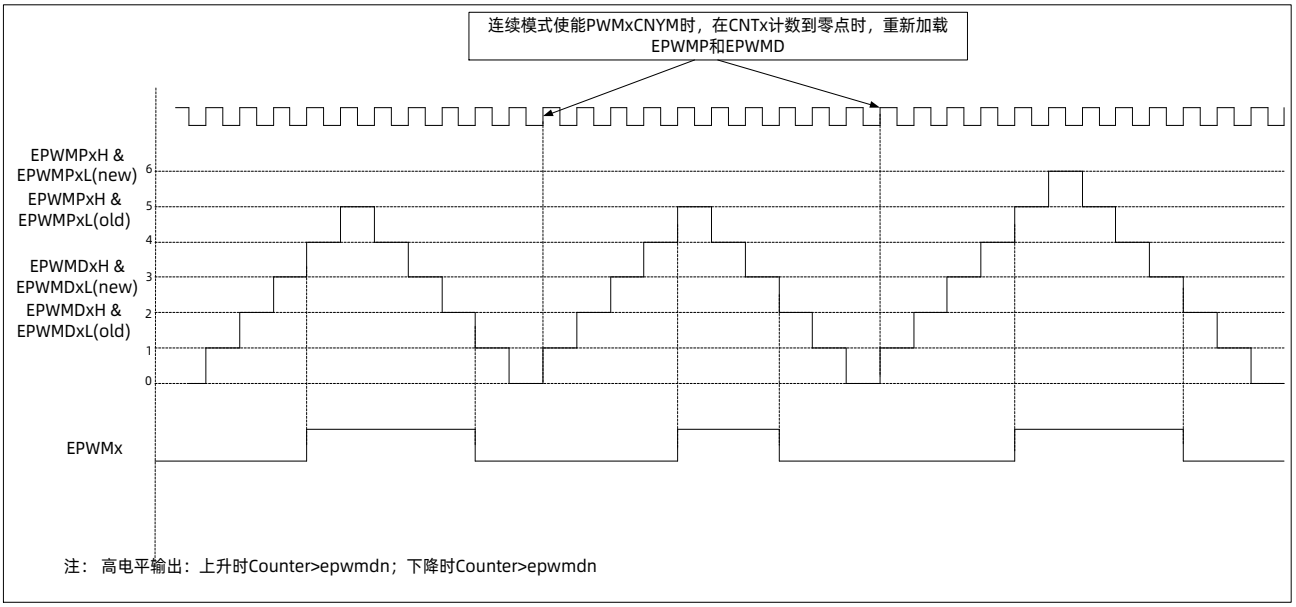


10.20.3 PWM 波形图

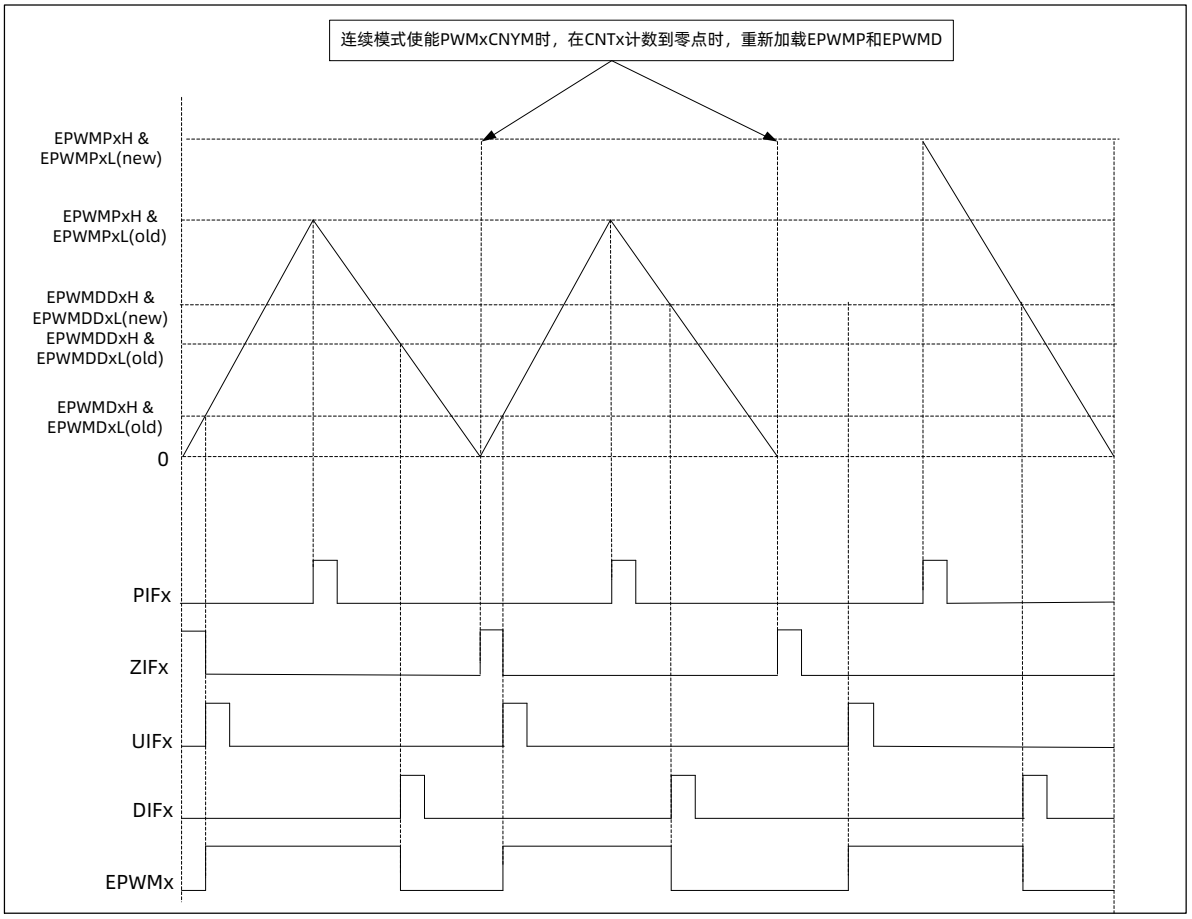
中心对齐对称计数模式



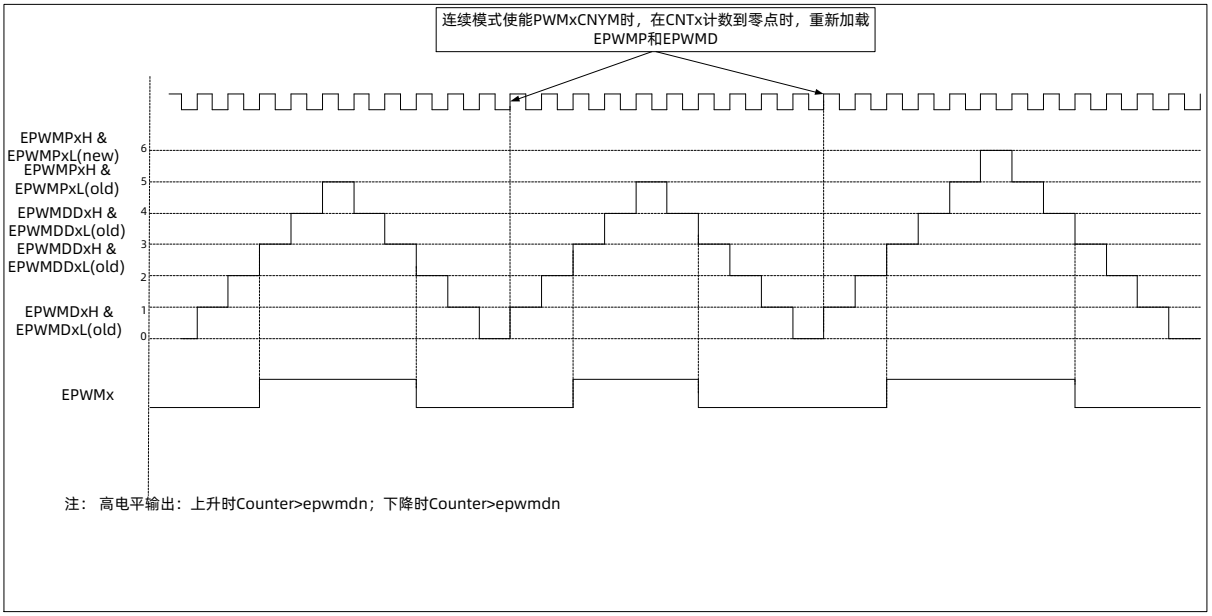
中心对齐对称计数波形



中心非对齐对称模式



中心非对齐对称计数波形



注：高电平输出：上升时Counter>epwmdn；下降时Counter>epwmdn

11 通用串行通讯口（USART0/1/2/3/4/5）

11.1 概述

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。

11.2 TXxCR 发送控制寄存器(x=0/1/2/3/4/5)

	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 **TxMCLR：**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

TxSPD[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

11.3 TXxREG 发送数据寄存器(x=0/1/2/3/4/5)

	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

11.4 RXxCR 接收控制寄存器(x=0/1/2/3/4/5)

	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 = 上升沿发送数据
1 = 下降沿发送数据

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

11.5 RXxREG 接收数据寄存器(x=0/1/2/3/4/5)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RXxREG	RXxREG[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

11.6 BRGDxH 波特率寄存器高位(x=0/1/2/3/4/5)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BRGDxH	SBYTEx	ODxEN	STOP1	-	-	-	BRGDx[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 **ODxEN：** 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] **STOP1: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

11.7 BRGDxL 波特率寄存器低位(x=0/1/2/3/4/5)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BRGDxL	BRGDx[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

11.8 USART 使用说明

11.8.1 波特率设置

通过设置 BRGxD 和 SPDx（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}/((BRGDx+1)×n)

常用波特率设置（F_{OSC} = 32MHz）

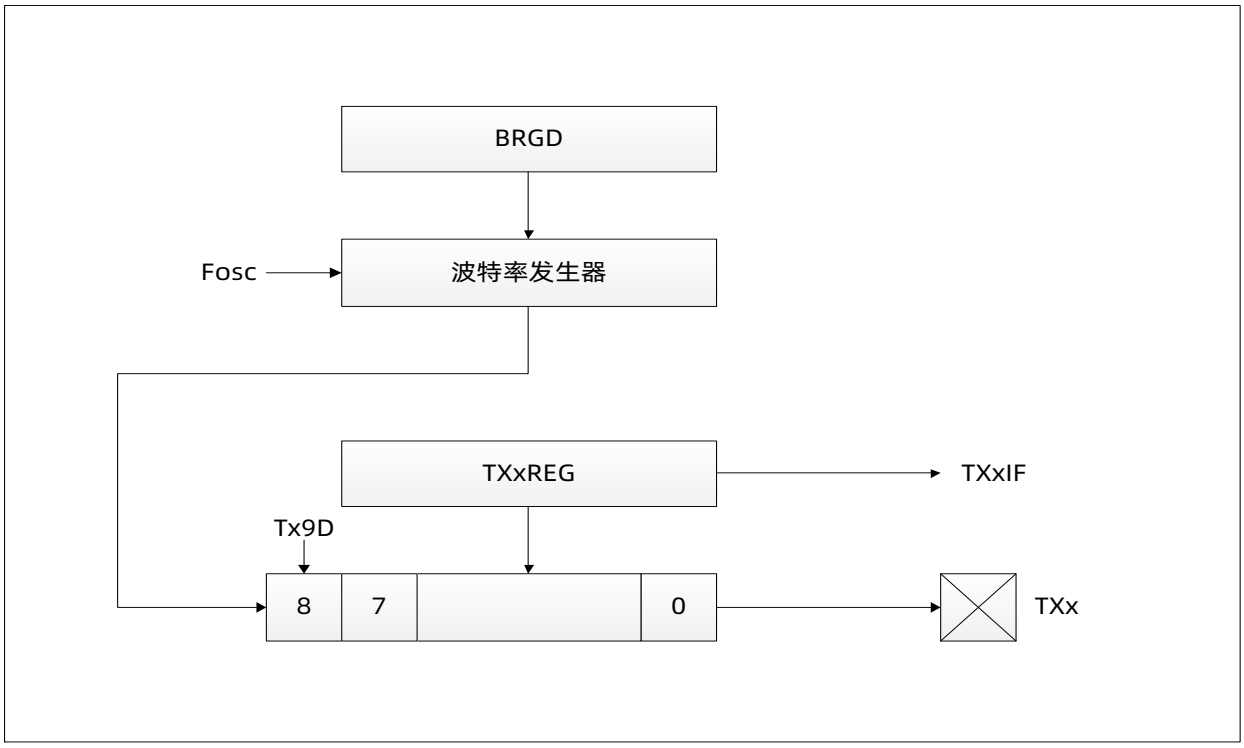
目标波特率	波特率时钟源分频比	BRGD	偏差
300	256	0x1A0	-0.08%
600	256	0xCF	0.16%
1200	64	0x1A0	-0.08%
2400	16	0x340	0.04%
4800	16	0x1A0	-0.08%
9600	4	0x340	0.04%
19200	4	0x1A0	-0.08%
38400	4	0xCF	0.16%
57600	4	0x8A	-0.08%
115200	4	0x44	0.64%

11.8.2 异步发送(x=0/1/2/3/4/5)

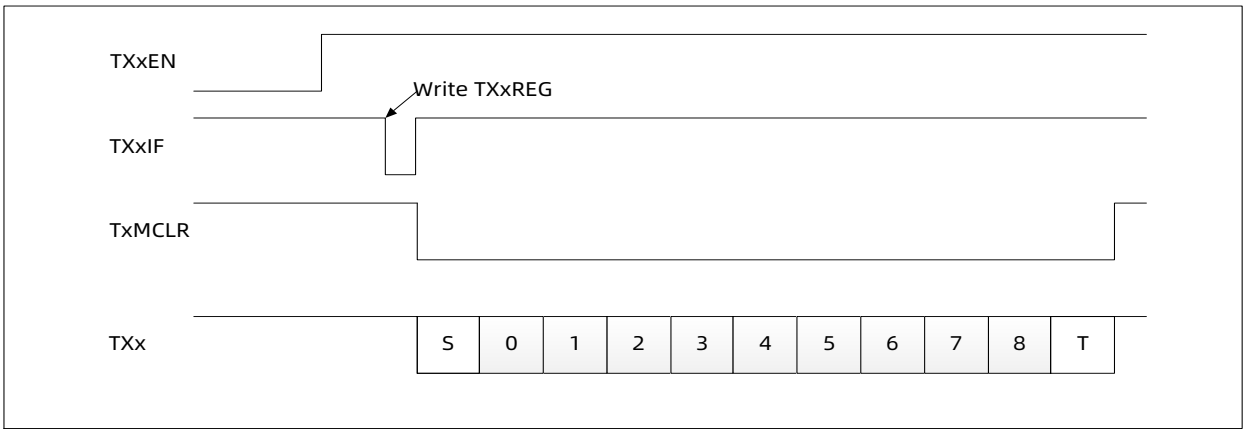
当 TXxEN 使能时，TXxIF 中断标志为 1 说明 TXxREG 发送寄存器为空，TxMCLR = 1 说明发送移位寄存器为空，发送器处于空闲状态。

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，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。

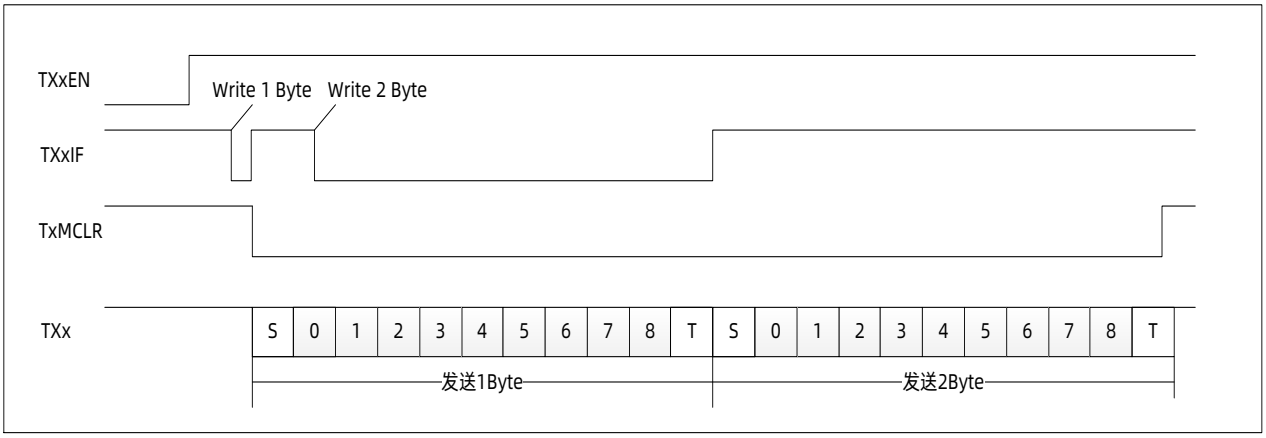
当 TXxIF 为 0 时写入 TXxREG，将覆盖上次写入数据。



单字节发送：



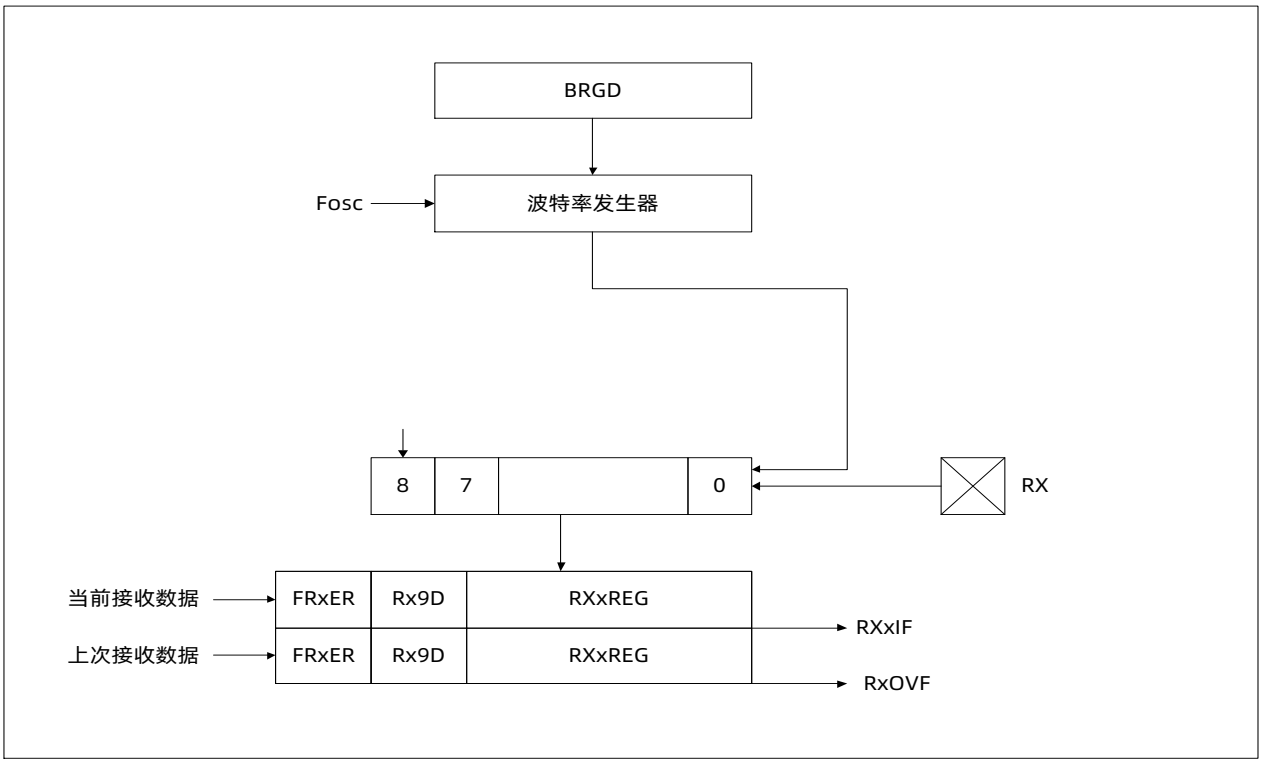
多字节发送：

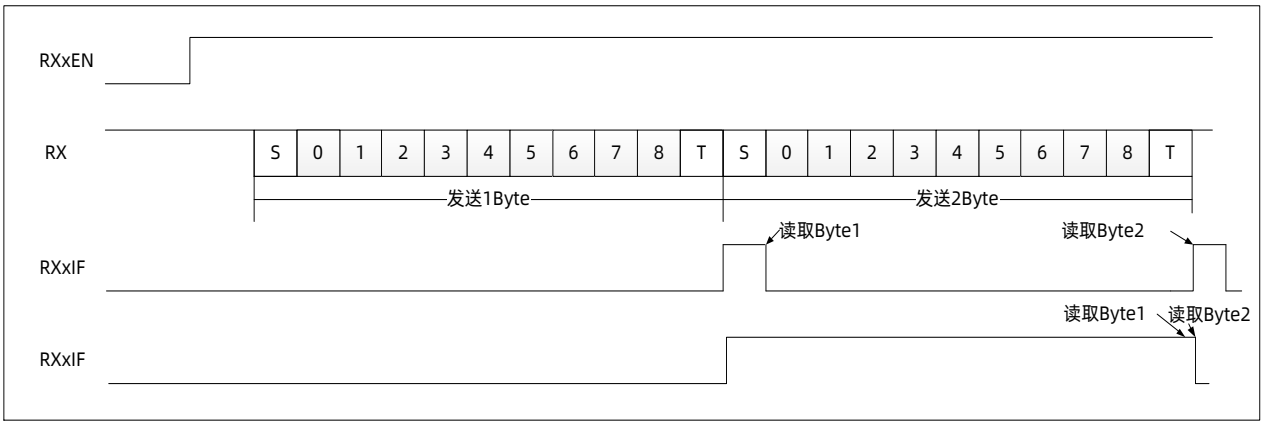


STEP1：设置波特率 TxSPD = X，BRGDx = X，TxSYNC = 0
STEP2：设置 TXxEN = 1，设置数据模式 TxL9 = X
STEP3：写入数据高位 TxD9
STEP4：写入 TXxREG，启动发送
STEP5：当 TXxIF = 1 时，写入 TXxREG 发送下一个字节
STEP6：重复 STEP5，直到该帧数据发送完成

11.8.3 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。





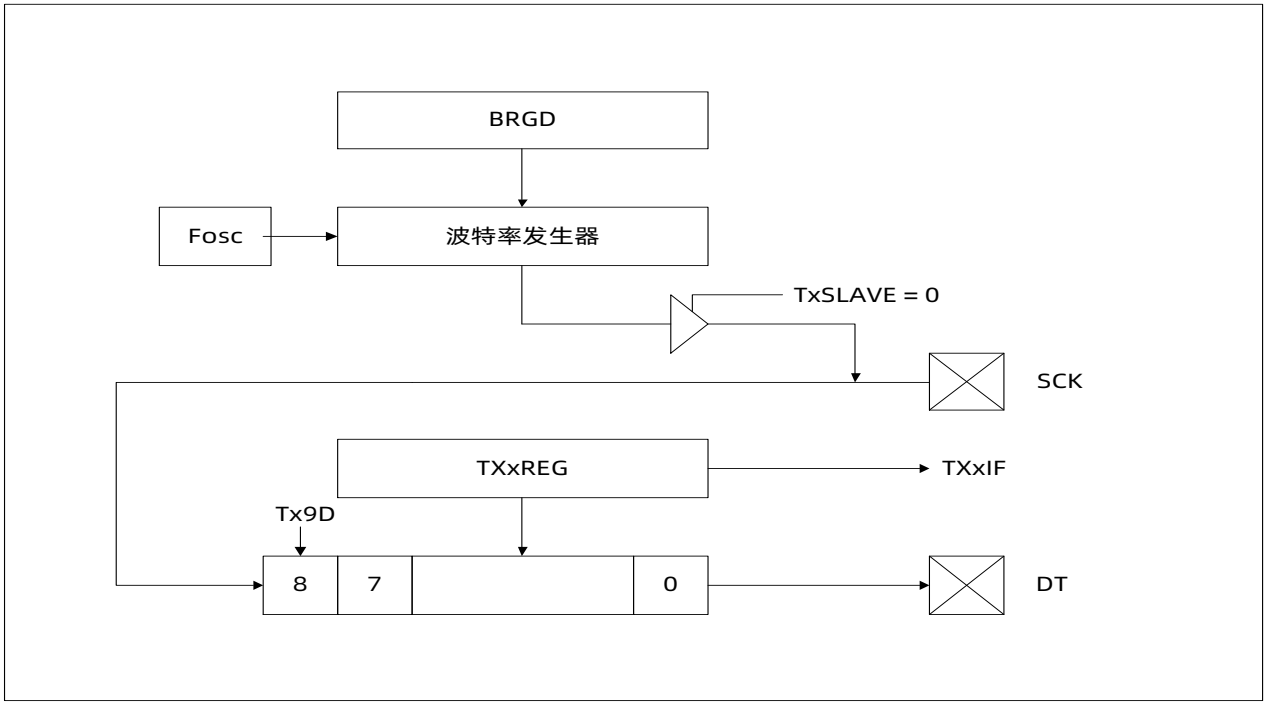
STEP1：设置波特率 TxSPD = X，BRGDx = X，TxSYNC = 0
STEP2：设置 RXxEN = 1
STEP3：等待接收完成 RXxIF = 1
STEP4：判断 FRxER = 0，若为 1，帧格式错误，舍弃数据
STEP5：读取 RxD9
STEP6：读取 RXxREG，重复 3-6

11.8.4 同步发送

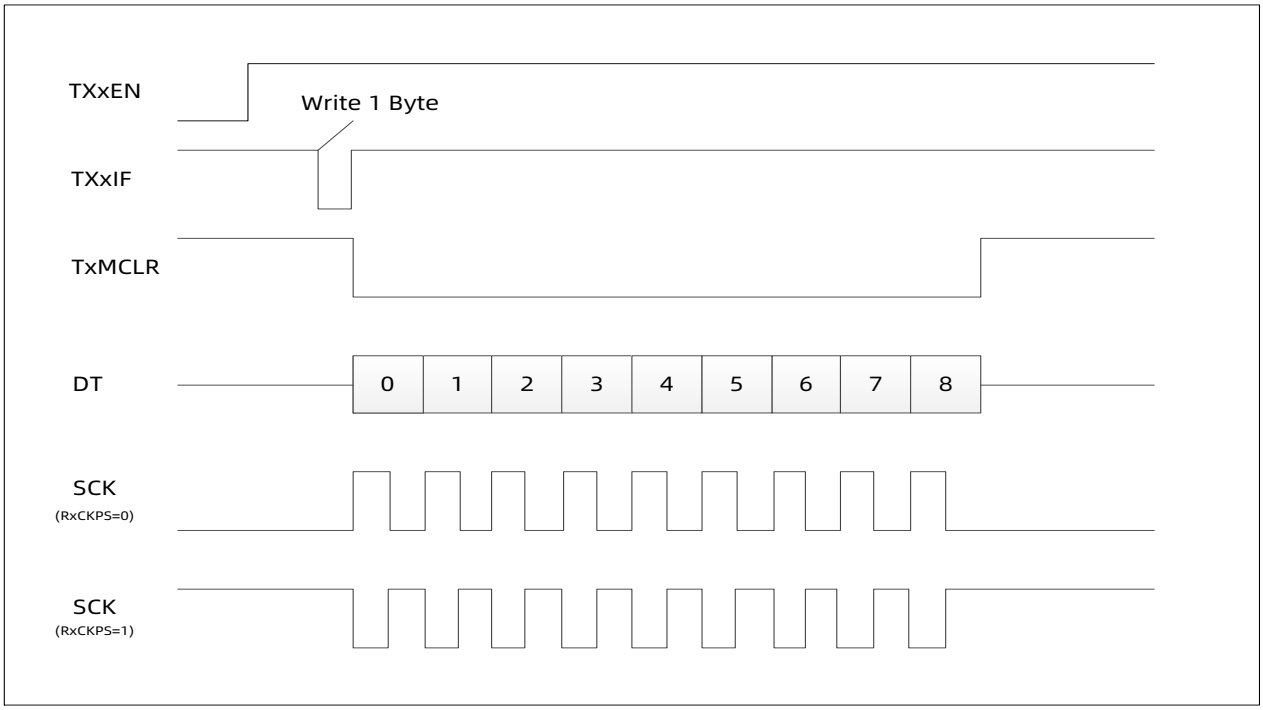
当 TXxEN=1，TxSYNC=1 时，使能同步发送功能。 CKPS 选择发送时钟极性，TXxIF 中断标志为 1 说明 TXxREG 发送寄存器为空，TxMCLR=1 说明发送移位寄存器为空，发送器处于空闲状态。

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。

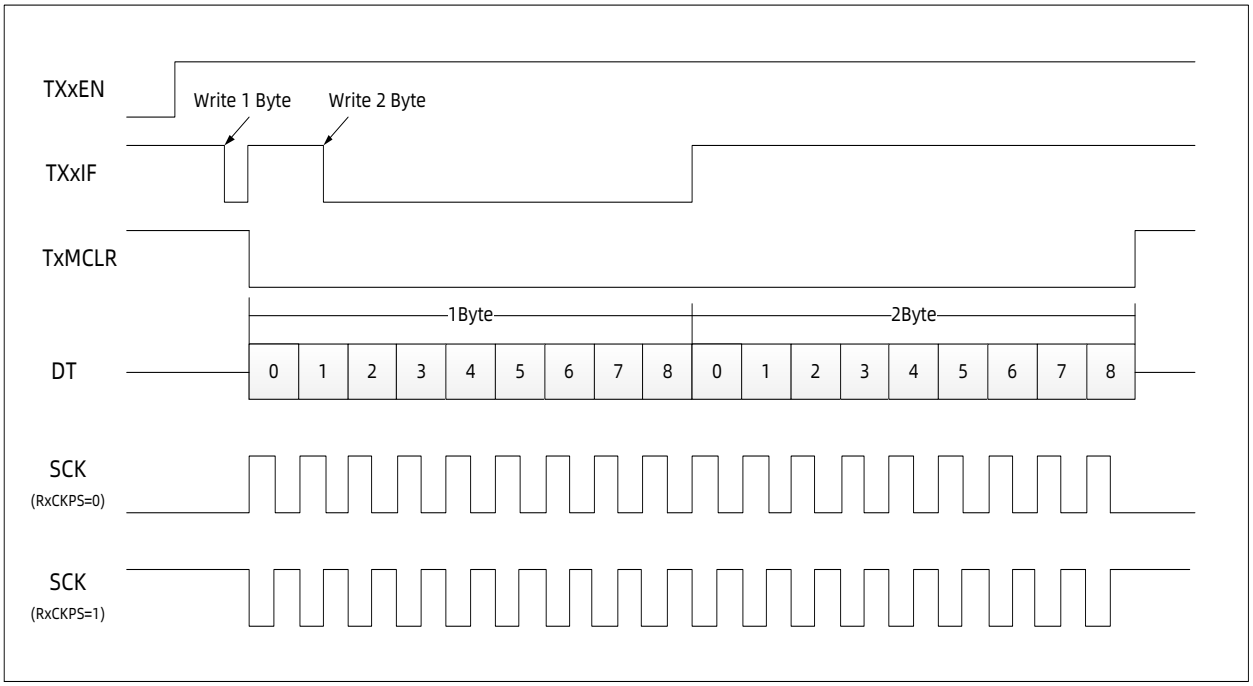
当 TXxIF 为 0 时写入 TXxREG，将覆盖上次写入数据。



单字节发送：



多字节发送：



参考操作步骤 TxSLAVE = 0：

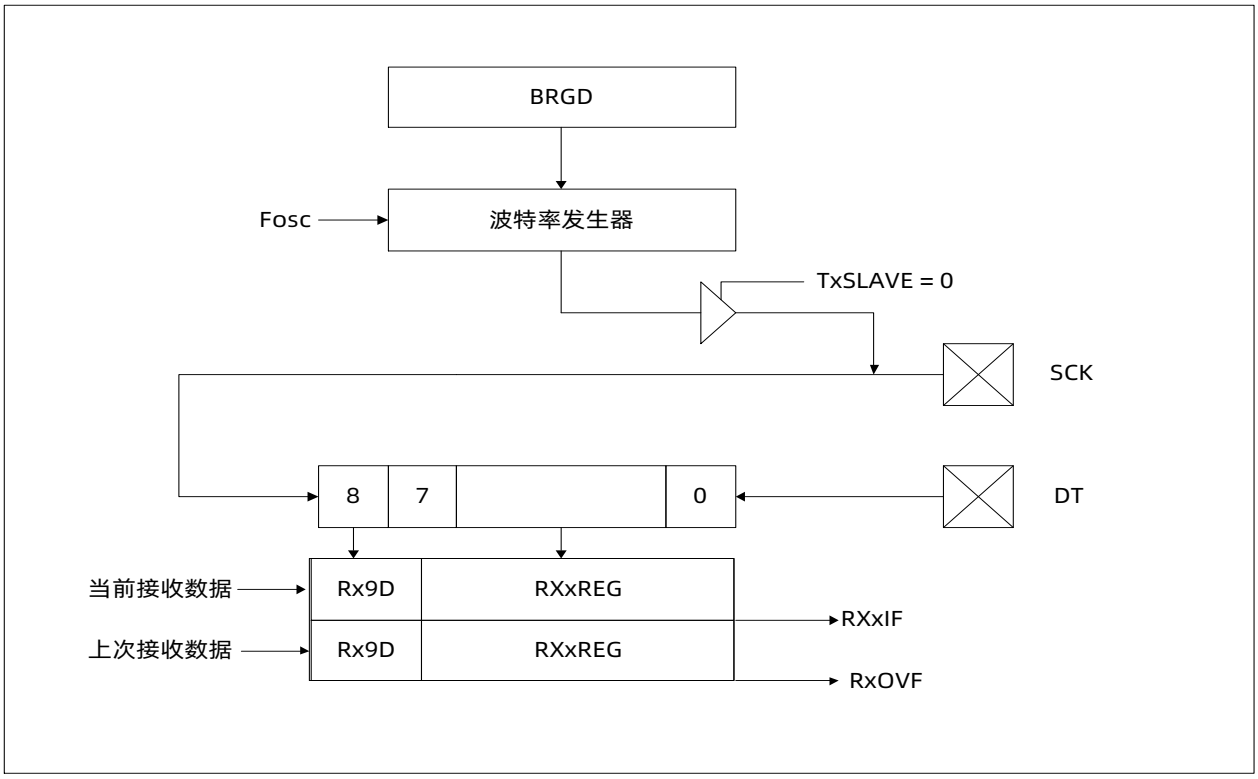
STEP1：设置波特率 TxSPD = X，BRGDx = X，TxSYNC = 1
STEP2：设置 TXxEN = 1，设置数据模式 TxL9 = X
STEP3：写入数据高位 TxD9
STEP4：写入 TXxREG，启动发送
STEP5：当 TXxIF = 1 时，写入 TXxREG 发送下一个字节
STEP6：重复 STEP5，直到该帧数据发送完成

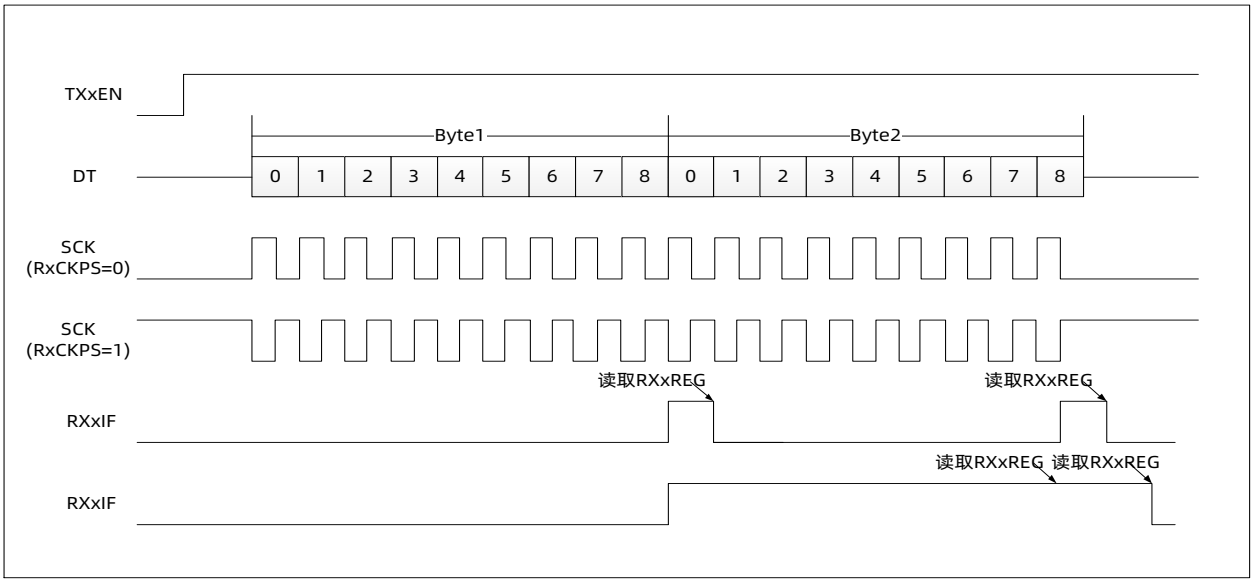
参考操作步骤 TxSLAVE = 1：

STEP1：设置波特率 TxSPD = X，BRGDx = 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

11.8.5 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 TxSPD = X，BRGDx = 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 RxD9

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 TxSPD = X，BRGDx = 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 RxD9

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

11.8.6 唤醒及休眠模式下通讯

TXxIE 置 1 时，TXxIF 中断标志可唤醒 CPU。

RXxIE 置 1 时，RXxIF 中断标志可唤醒 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。

12 I2C

12.1 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。

12.2 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。

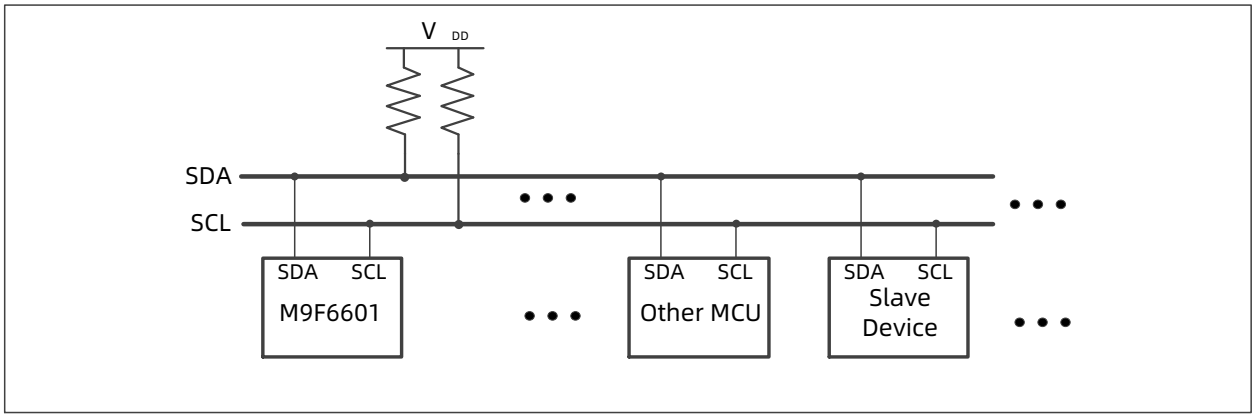


图 11.2-1 I2C 总线连接图

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。

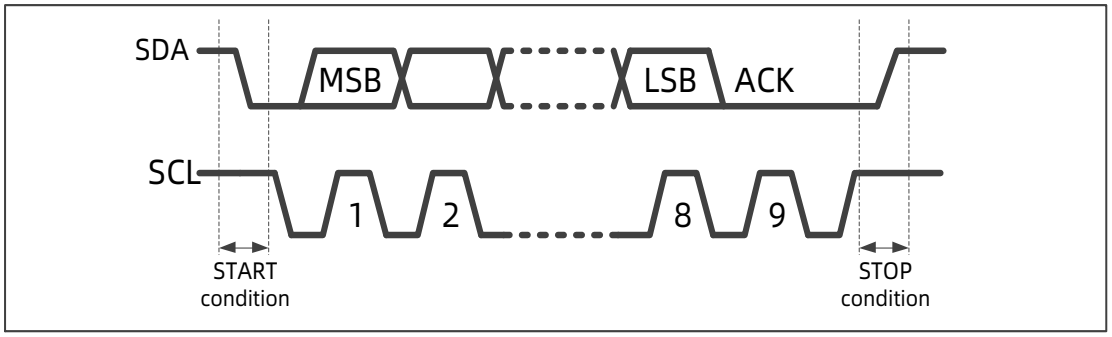


图 11.2-2 I2C 总线数据协议图

12.2.1 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。

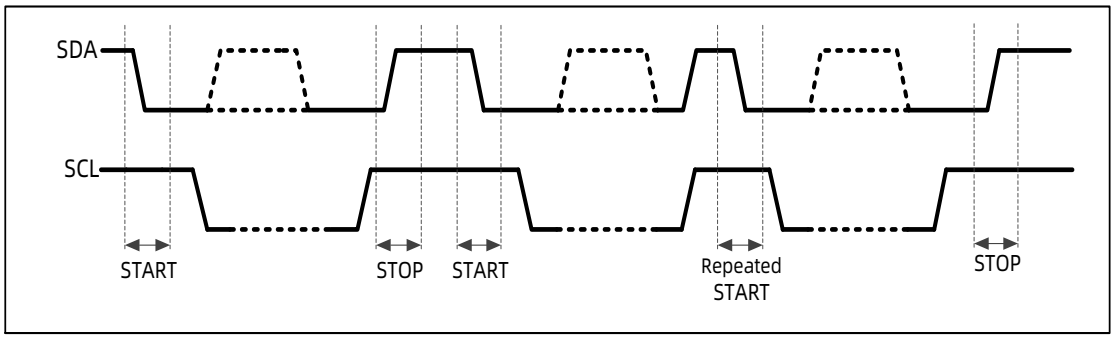


图 11.2-3　　起始 START、重复起始 Repeated START 和停止 STOP 信号

12.2.2 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。

12.2.3 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。

12.2.4 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。

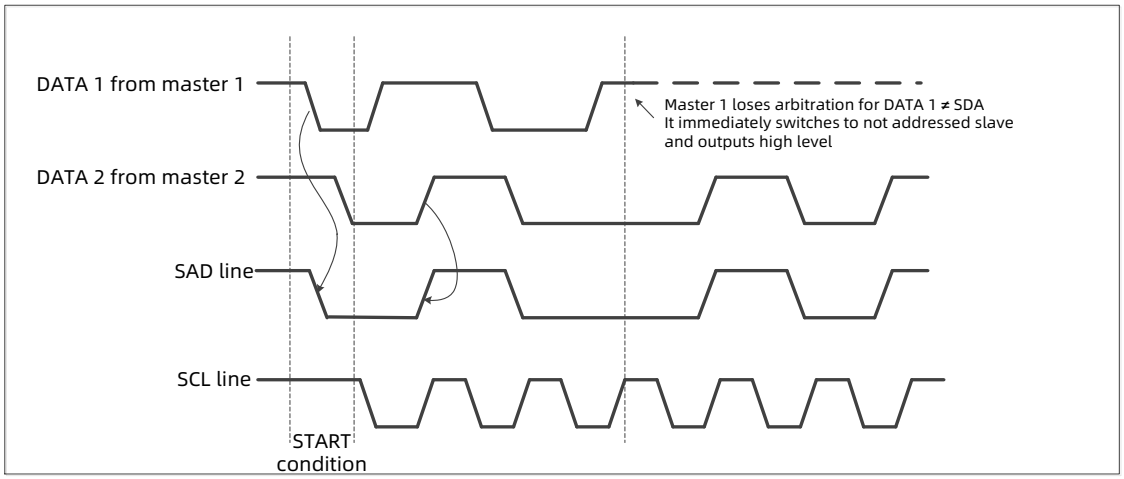


图 11.2-4 两台主机仲裁过程

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。

12.3 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。

12.3.1 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。

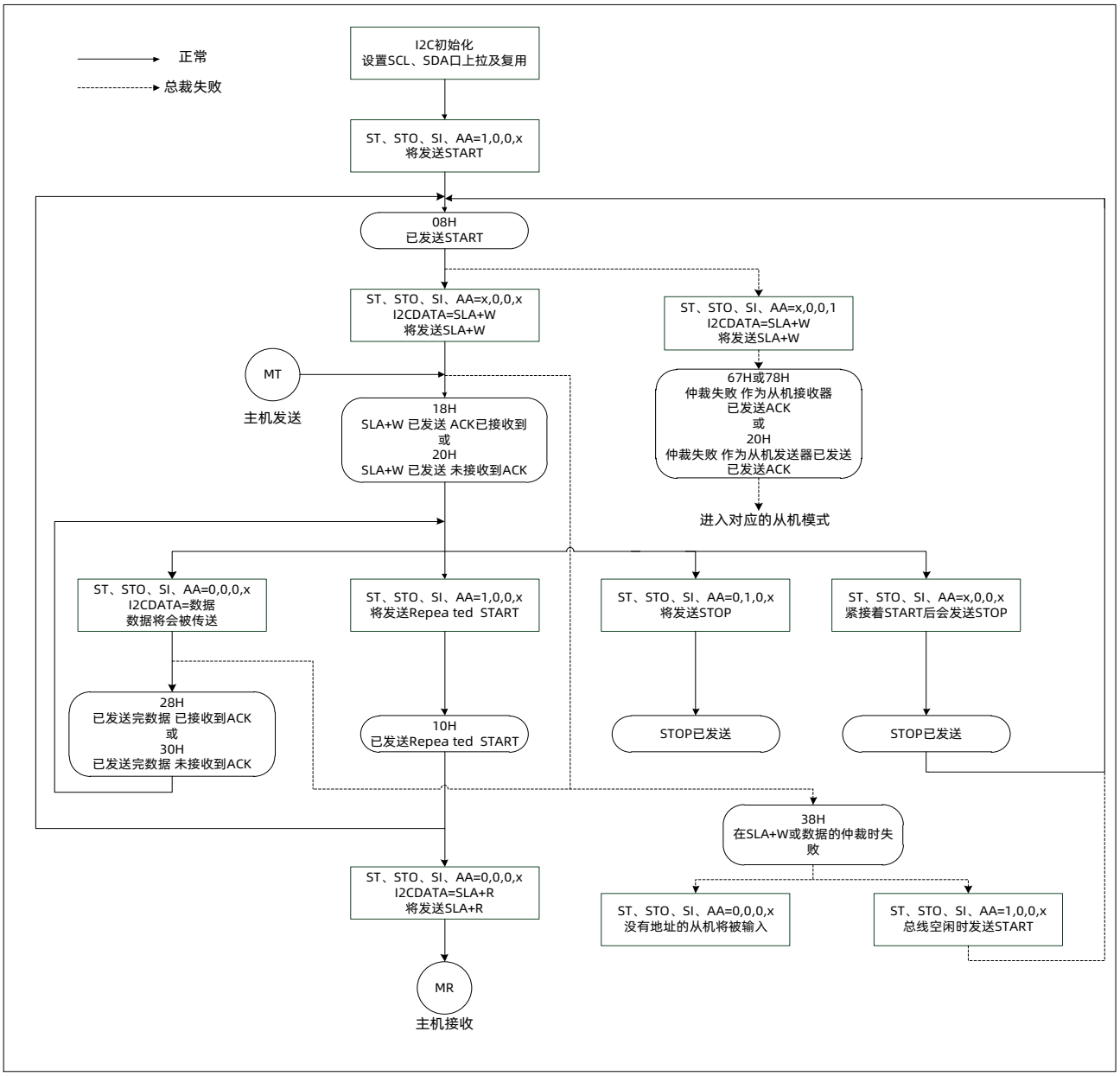


图 11.3-1 主机发送模式流程与状态图

12.3.2 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。

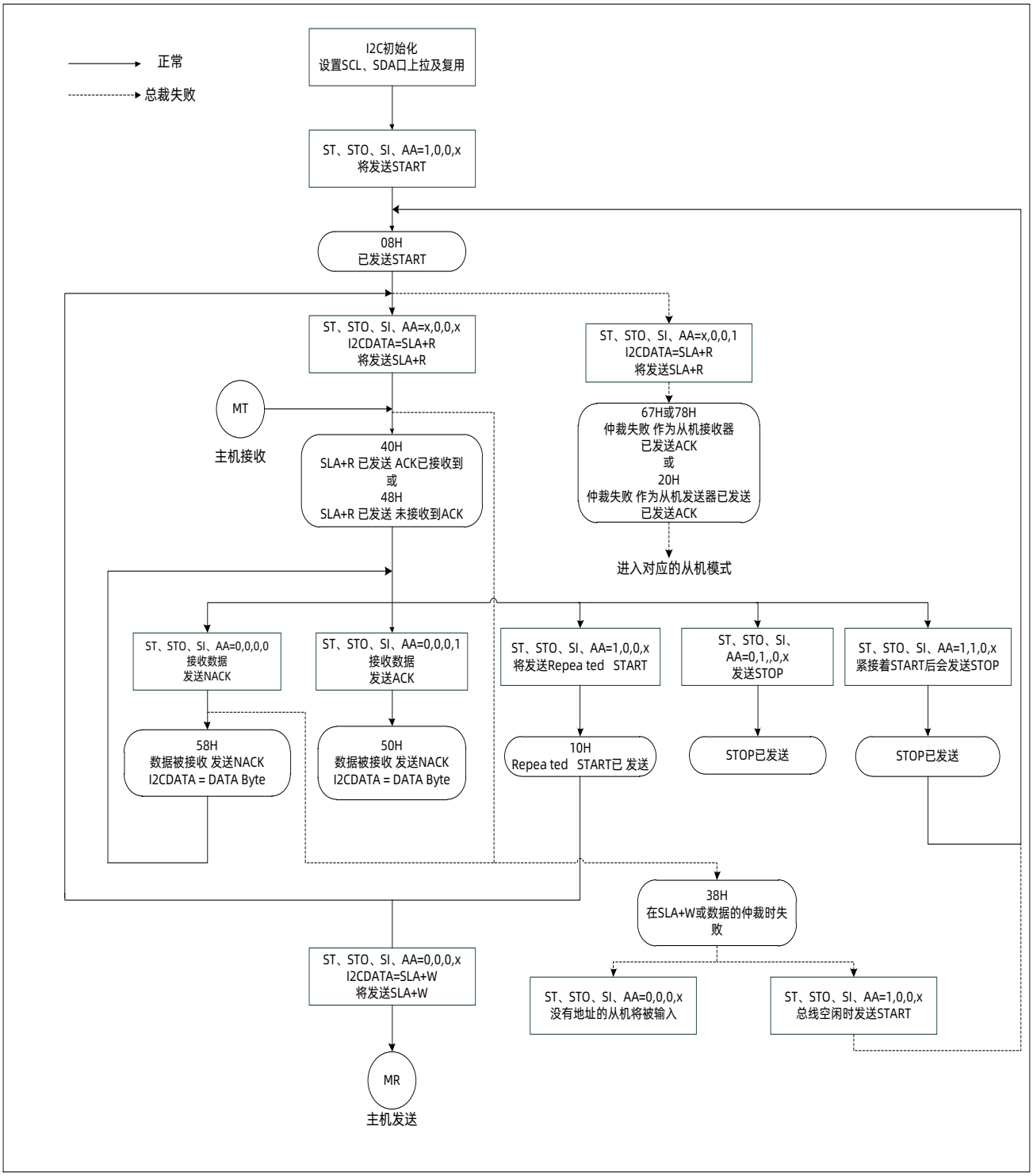


图 11.3-2　　主机接收模式流程与状态图

12.3.3 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。

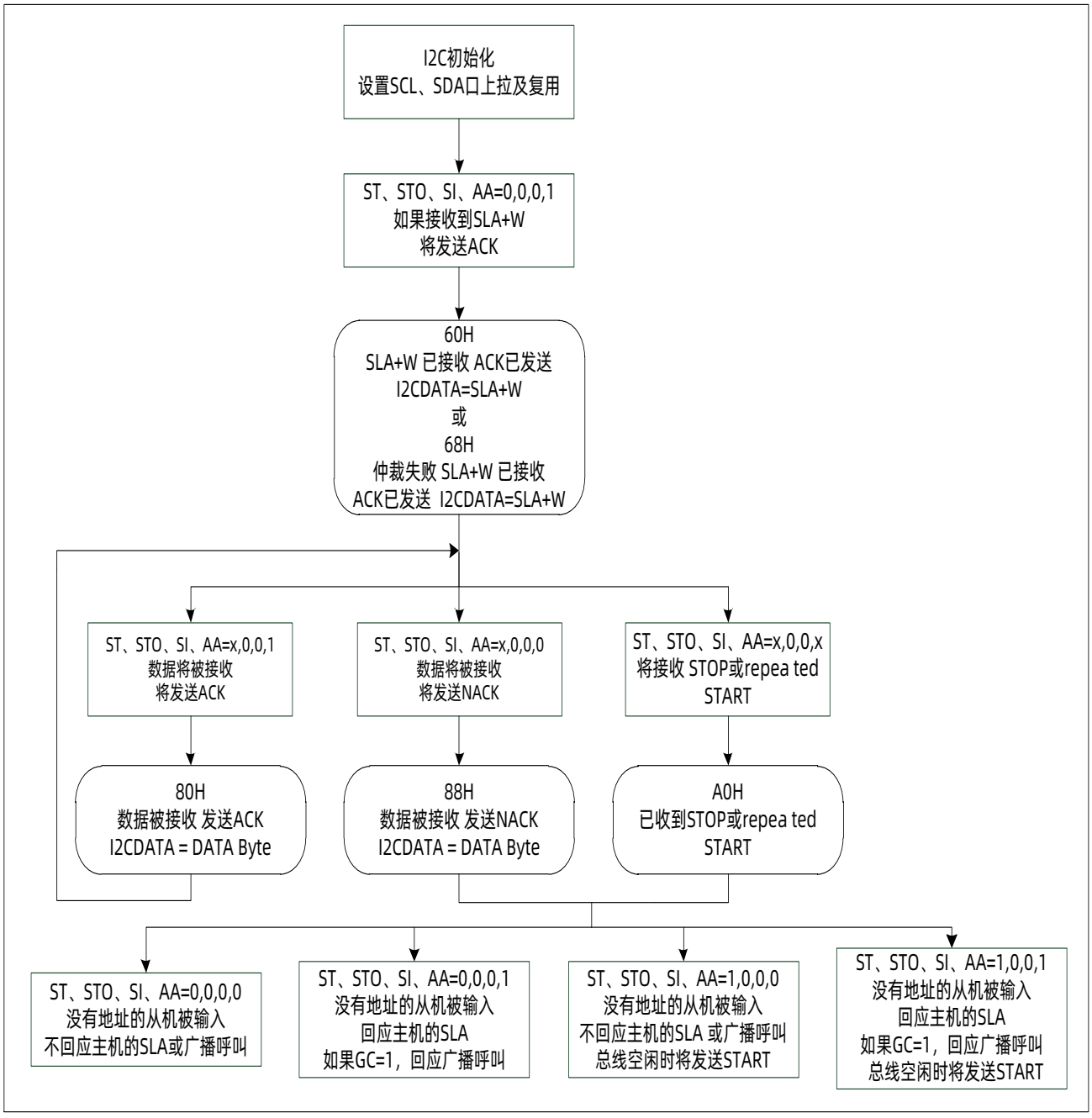


图 11.3-3　　从机接收模式流程与状态图

12.3.4 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。

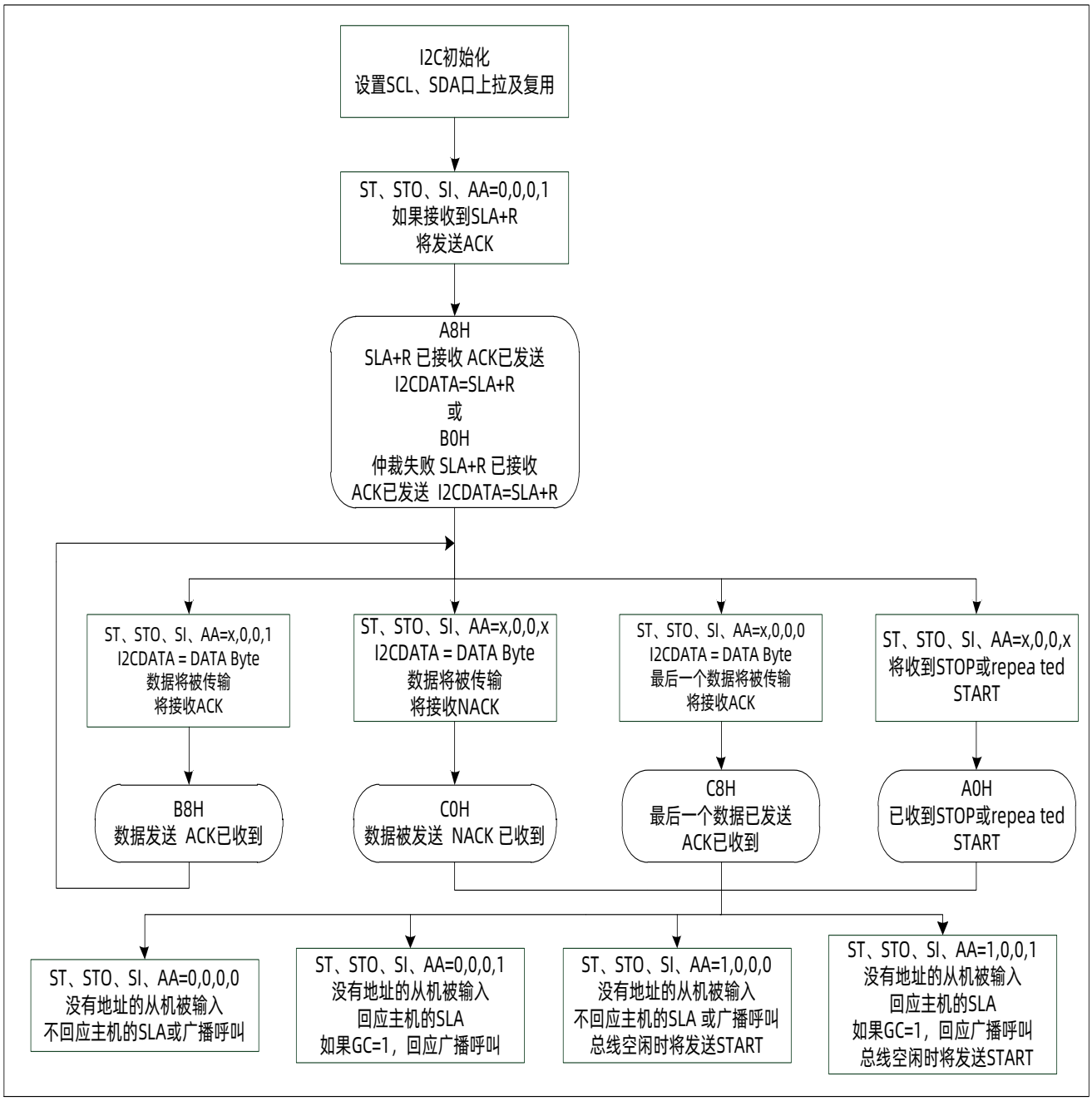


图 11.3-4 从机发送模式流程与状态图

12.3.5 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。

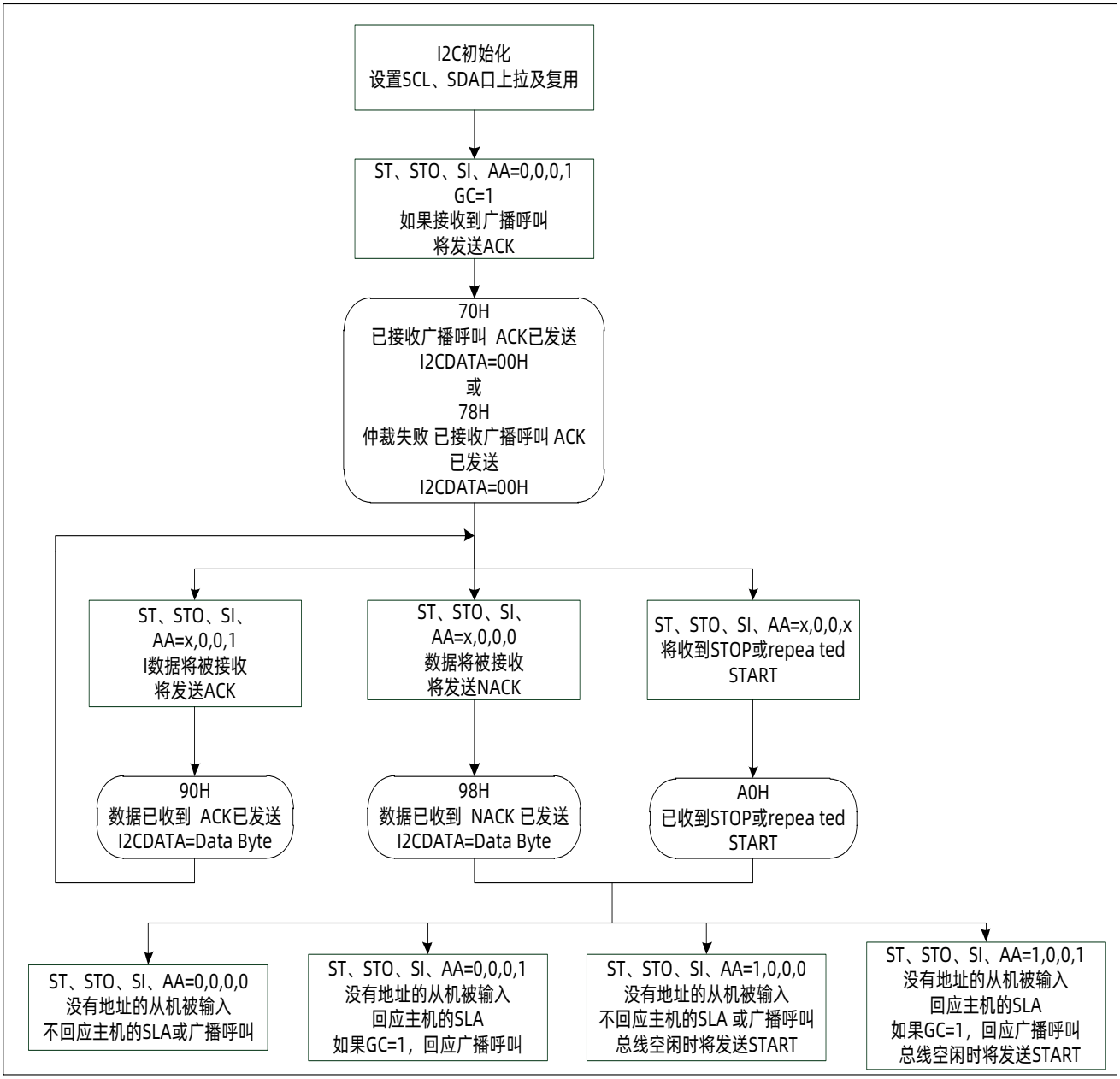


图 11.3-5 广播呼叫模式流程与状态图

12.3.6 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。

12.4 I2C 控制寄存器

12.4.1 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
000	F_{osc}/32	100	F_{osc}/160
001	F_{osc}/40	101	F_{osc}/192
010	F_{osc}/60	110	F_{osc}/240
011	F_{osc}/128	111	F_{osc}/320

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

12.4.2 I2C 状态寄存器

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

12.4.3 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

12.4.4 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

13 SPI

13.1 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

13.2 SPI 框图

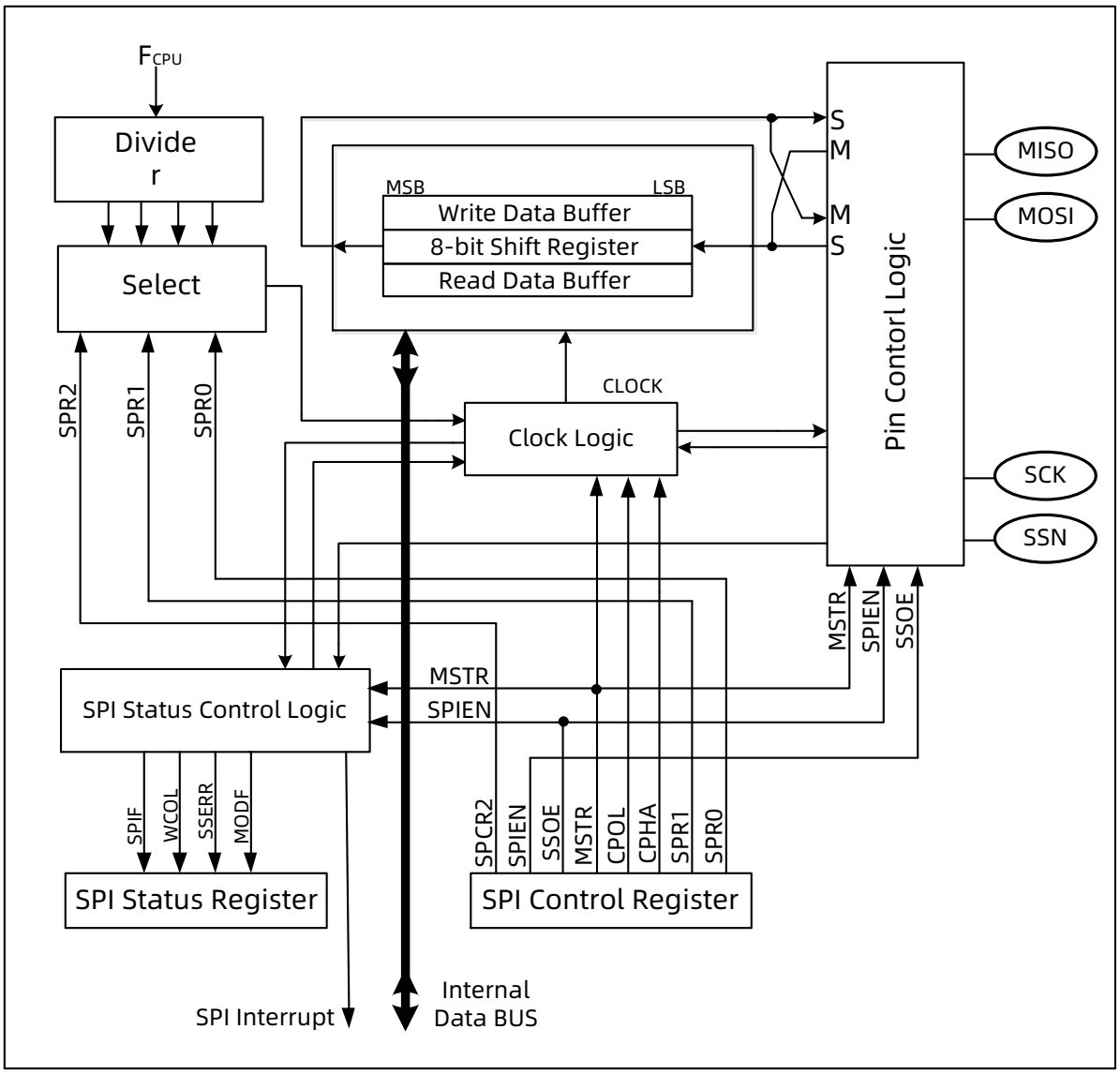


图 12.1　SPI 结构图

13.3 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。

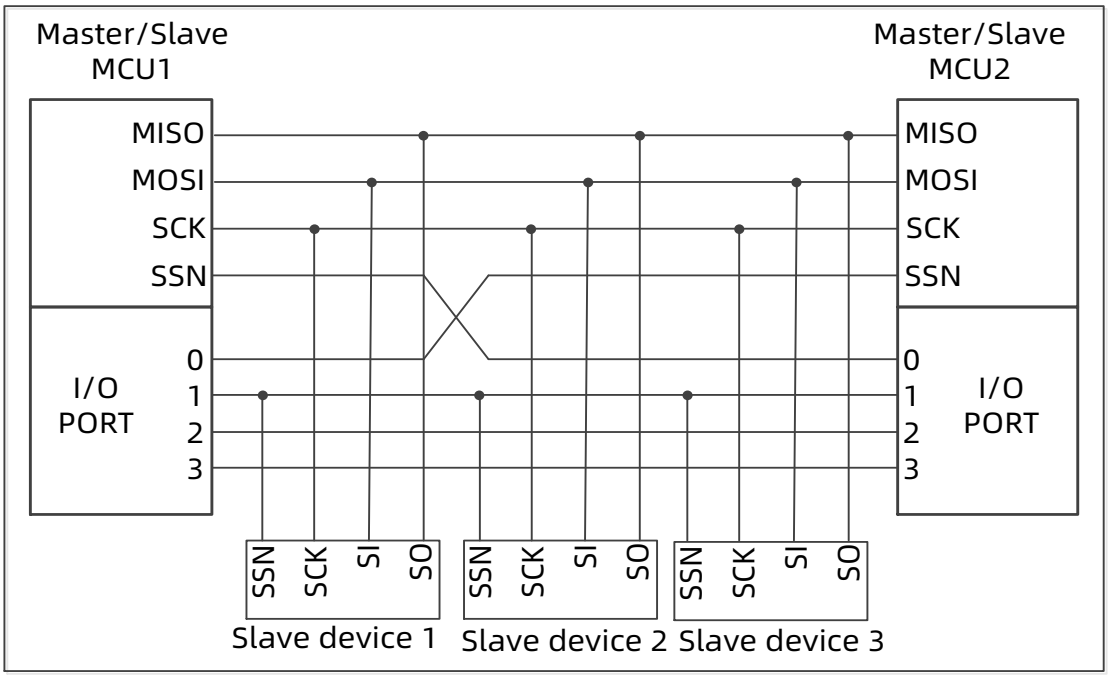


图 12.2-1　SPI 多主机多从机连接图

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。

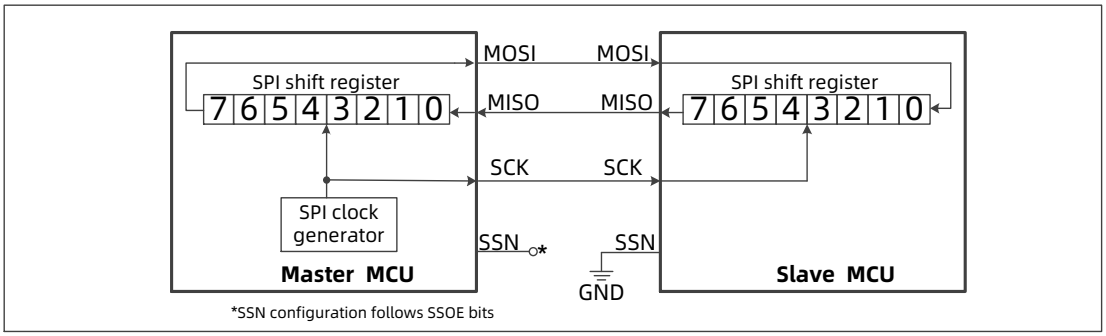


图 12.2-2 SPI 单主机单从机连接图

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。

13.4 SPI 寄存器

13.4.1 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] **SPCR[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

13.4.2 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

13.4.3 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

13.5 工作模式

13.5.1 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。

13.5.2 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。

13.6 时钟格式和数据传输

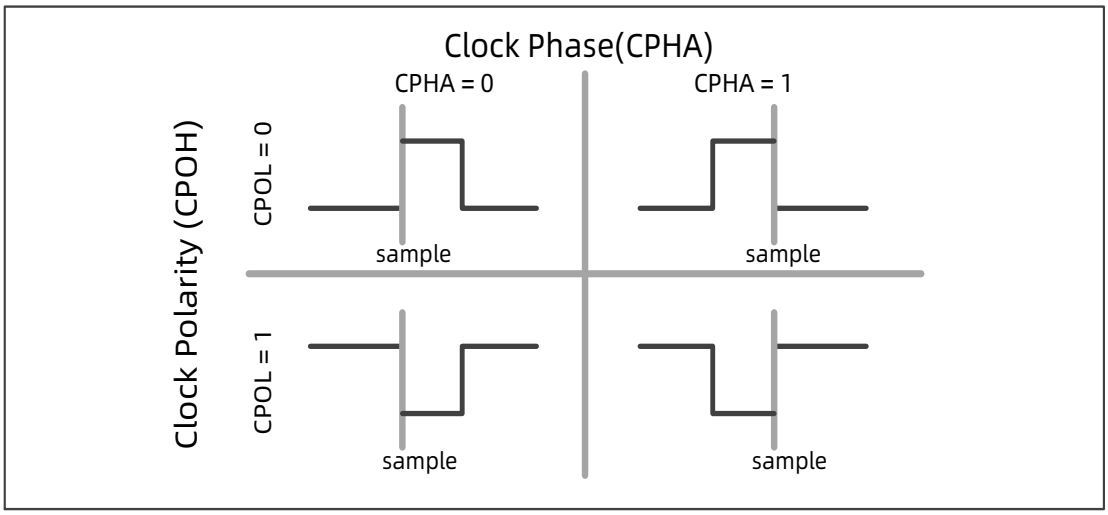


图 12.3 SPI 时钟格式图

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。

13.7 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。

13.8 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。

14 触摸按键（CDC）

31 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。

15 模数转换器(ADC)

15.1 概述

M9F6601有18路外部通道（AIN0~AIN17）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。

15.2 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	1	1	0	0

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 = AIN0 ~ AIN17
10010 = VDD/4
10011 = VREF
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。

15.3 ADCON1 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	VHS2	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 [6:4] **ADCKS[2:0]：** ADC 时钟源选择位

ADCKS[2:0]	ADC 时钟源选择
000	F_{cpu}
001	$F_{cpu}/2$
010	$F_{cpu}/4$
011	$F_{cpu}/8$
100	$F_{cpu}/16$
101	$F_{cpu}/32$
110	$F_{cpu}/64$
111	

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

Bit [7,1:0] **VHS[2:0]：** ADC 内建基准电平选择位。

VHS[2:0]	内建 VREF 基准电平
000	关闭内部参考
001	2.0V
010	3.0V
011	4.0V
100	关闭内部参考
101	VREF+
110	
111	

注：

（1）若由 VHS[1:0]控制选择的内部 VREF 电平高于 VDD，内部 VREF 为 VDD。

例：VHS[1:0] = 11（内部 VREF = 4.0V），VDD = 3.0V，则实际内部 VREF = 3.0V。

15.4 ADCON2 寄存器

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

Bit 7 **ADNCF：**电源噪声过大标志输出

Bit 4 **Reserved：**保留，请保持为0

Bit [3:0] **ADVOS[3:0]：**ADC失调补偿寄存器

15.5 ADCON3 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON3	ADCEX	-	ADTGS1	ADTGS0	ADEGS1	ADEGS0	ADDLY9	ADDLY8
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 **ADCEX:** ADC硬件触发使能位

1 = 使能

0 = 禁止

Bit [5:4] **ADTGS[1:0]:** ADC硬件触边源选择位

ADTGS [1:0]	ADC 硬件触发源选择位
00	PG0（EPWM0）
01	PG2（EPWM2）
10	PG4（EPWM4）
11	端口引脚（ADET）

Bit [3:2] **ADEGS[1:0]:** ADC硬件触发边沿选择位

ADEGS [1:0]	ADC 硬件触发边沿选择位
00	下降沿
01	上升沿
10	EPWM 的周期点
11	EPWM 的零点

Bit [1:0] **ADDLY[9:8]:** ADC硬件触发延时数据[9:8]位

15.6 ADCON4 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON4	ADDLY8	ADDLY8	ADDLY8	ADDLY8	ADDLY8	ADDLY8	ADDLY8	ADDLY8
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] **ADDLY[7:0]:** ADC硬件触发延时数据低8位

15.7 ADH/ADL AD 结果寄存器

15.7.1 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

15.7.2 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

15.8 AD 转换时间

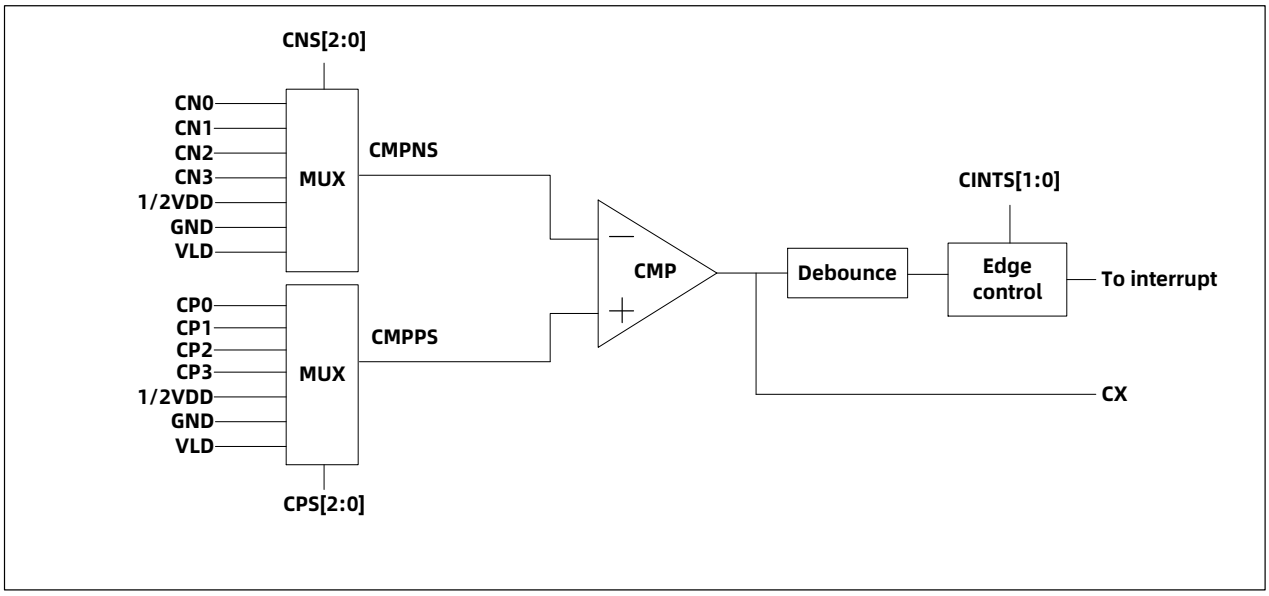
注：12 位 AD 转换时间 = 1/(ADC clock)*16 sec

16 比较器（CMP）

16.1 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。

16.2 比较器框图



16.3 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：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。

16.4 CMPC1 比较器控制寄存器 1

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CMPC1	CMPOEN	CMPHIEN	-	CMPVLD[4:0]				
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

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 **CMPHIEN：** 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

16.5 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/W	R/W	R/W
复位后	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_{CPU} * 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。

17 LCD 模块

17.1 概述

M9F6601 LCD 驱动模块由 LCD 控制寄存器以及数据寄存器组成，可以驱动 LCD 段码。

- 可选择 2-8COM
- 可调节帧频
- 1/2、1/3、1/4BIAS 电压可选
- 内部电阻模式（可设 VLCD 电压）

17.2 LCDCR0 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCR0	LCDEN	LCDM	LCDCKS	BIAS		CIOS[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 **LCDEN**: LCD 功能使能位

0 = 屏蔽 LCD 功能

1 = 使能 LCD 功能

Bit 6 **LCDM**: LCD 驱动模式选择

0 = B 类驱动模式

1 = A 类驱动模式

Bit 5 **LCDCKS**: LCD 时钟选择

0 = Fosch

1 = Foscl

Bit [4:3] **BIAS**: LCD BIAS 选择

00 = 1/2BIAS

01 = 1/3BIAS

10 = 1/4BIAS

Bit [2:0] **CIOS**: LCD Duty&IO 选择位

CIOS[2:0]	LCD Duty & IO 选择
000	Com0-7 作为 IO 口
001	Com0-1 作为 Com 口,Com2-7 作为 IO 口
010	Com0-2 作为 Com 口,Com3-7 作为 IO 口
011	Com0-3 作为 Com 口,Com4-7 作为 IO 口
100	Com0-4 作为 Com 口,Com5-7 作为 IO 口
101	Com0-5 作为 Com 口,Com6-7 作为 IO 口
110	Com0-6 作为 Com 口,Com7 作为 IO 口
111	Com0-7 作为 Com 口

17.3 LCDCR1 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCR1	LCDPR7	LCDPR6	LCDPR5	LCDPR4	LCDPR3	LCDPR2	LCDPR1	LCDPR0
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] **LCDPR:** 预分频

LCDCKS=0：（建议 LED 模式下使用）

帧频 = Fosch/((LCDPR+1)*256*m*n)

LCDCKS=1:（建议 LCD 低功耗时使用）

帧频 = Foscl/((LCDPR+1)*m*n)

注：m= Duty（com 数）；n=seg 数（LCD 模式）。

17.4 LCDCR2 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCR2	LEDEN	LEDOM	LEDMOD	LEDGS	LCDSPEED			
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 [3:0] **LCDSPEED:** LCD 小电阻开启时间

Tspd = (LCDSPEED*Tlcd)us (LCDSPEED！=0xF)

Tspd = 全开 (LCDSPEED =0xF)

17.5 LCDCR3 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCR3	LEDSOUTMOD	LEDCOUTMOD	RAMMAP	VLCD4	VLCD3	VLCD2	VLCD1	VLCD0
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 5 **RAMMAP:** LCD RAM 映射方式

0 = 方式 1

1 = 方式 2

Bit [4:0] **VLCD**: VLCD 电压选择

VLCD[4:0]	VLCD	VLDOS[4:0]	VLCD
00000	0.50 VDD	01001	0.78 VDD
00001	0.53 VDD	01010	0.81 VDD
00010	0.56 VDD	01011	0.84 VDD
00011	0.59 VDD	01100	0.88 VDD
00100	0.63 VDD	01101	0.91 VDD
00101	0.66 VDD	01110	0.94 VDD
00110	0.69 VDD	01111	0.97 VDD
00111	0.72 VDD	1xxxx	VDD
01000	0.75 VDD		

17.6 LCDCR4 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCR4				-	LCDMODIF	-	OPAIS1	OPAIS0
读/写	-	-	-	-	R/W	-	R/W	R/W
复位后	-	-	-	0	0	-	0	0

Bit 3 **LCDMODIF:** LCD 中断运行模式控制位

0 = LEDIF 为 1 时，LCD 扫描继续

1 = LEDIF 为 1 时，LCD 扫描停止，需要将 LEDIF 清 0 开始下一帧扫描

Bit [1:0] **OPAIS:** OP 档位选择

OPAIS [1:0]	OP 档位选择
00	1uA
01	2uA
10	3uA
11	4uA

17.7 SEGIOSX 端口设置（X=A/B/C/D）

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGIOSA	SEGIOSA7	SEGIOSA6	SEGIOSA5	SEGIOSA4	SEGIOSA3	SEGIOSA2	SEGIOSA1	SEGIOSA0
SEGIOSB	SEGIOSB7	SEGIOSB6	SEGIOSB5	SEGIOSB4	SEGIOSB3	SEGIOSB2	SEGIOSB1	SEGIOSB0
SEGIOSC	SEGIOSC7	SEGIOSC6	SEGIOSC5	SEGIOSC4	SEGIOSC3	SEGIOSC2	SEGIOSC1	SEGIOSC0
SEGIOSD	SEGIOSD7	SEGIOSD6	SEGIOSD5	SEGIOSD4	SEGIOSD3	SEGIOSD2	SEGIOSD1	SEGIOSD0
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] **SEGIOSxn:** SEG口选择（x=A/B/C/D,n=0-7）

0 = 作为IO

1 = 作为SEG

注：

（1）在未开启 LCD 和 LED 的情况下，可以控制 SEGIOSx 控制 IOx（x=A/B/C/D）使用电流源模式。

（2）寄存器具体地址参考<2.2.8 系统寄存器定义>

17.8 LCDDSxx：LCD 显示数据寄存器

LCD RAM 映射方式 1

地址		Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
E60H	COM0	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	-
E61H		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E62H		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E63H		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E64H	COM1	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	-	SEG0
E65H		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E66H		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E67H		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E68H	COM2	SEG7	SEG6	SEG5	SEG4	SEG3	-	SEG1	SEG0
E69H		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E6AH		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E6BH		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E6CH	COM3	SEG7	SEG6	SEG5	SEG4	-	SEG2	SEG1	SEG0
E6DH		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E6EH		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E6FH		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E70H	COM4	SEG7	SEG6	SEG5	-	SEG3	SEG2	SEG1	SEG0
E71H		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E72H		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E73H		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E74H	COM5	SEG7	SEG6	-	SEG4	SEG3	SEG2	SEG1	SEG0
E75H		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E76H		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E77H		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E78H	COM6	SEG7	-	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
E79H		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E7AH		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E7BH		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E7CH	COM7	-	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
E7DH		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E7EH		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E7FH		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24

LCD RAM 映射方式 2：

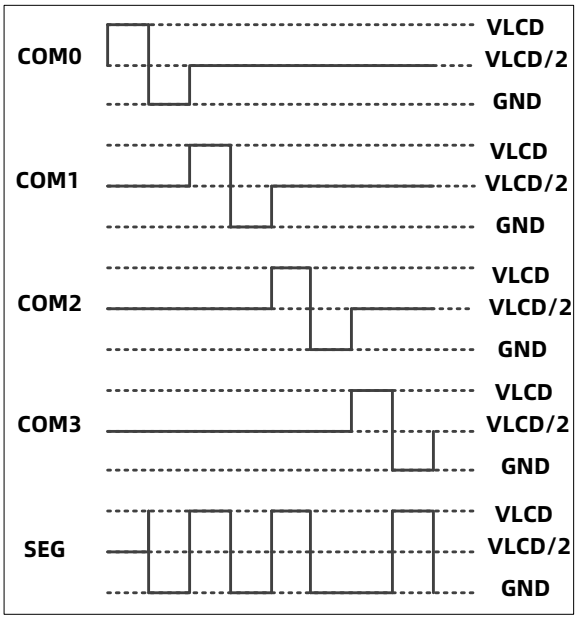
寄存器	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0
LCDDS00	E60H	C7S0	C6S0	C5S0	C4S0	C3S0	C2S0	C1S0	C0S0
LCDDS01	E61H	C7S1	C6S1	C5S1	C4S1	C3S1	C2S1	C1S1	C0S1
LCDDS02	E62H	C7S2	C6S2	C5S2	C4S2	C3S2	C2S2	C1S2	C0S2
LCDDS03	E63H	C7S3	C6S3	C5S3	C4S3	C3S3	C2S3	C1S3	C0S3
LCDDS04	E64H	C7S4	C6S4	C5S4	C4S4	C3S4	C2S4	C1S4	C0S4
LCDDS05	E65H	C7S5	C6S5	C5S5	C4S5	C3S5	C2S5	C1S5	C0S5
LCDDS06	E66H	C7S6	C6S6	C5S6	C4S6	C3S6	C2S6	C1S6	C0S6
LCDDS07	E67H	C7S7	C6S7	C5S7	C4S7	C3S7	C2S7	C1S7	C0S7
LCDDS08	E68H	C7S8	C6S8	C5S8	C4S8	C3S8	C2S8	C1S8	C0S8
LCDDS09	E69H	C7S9	C6S9	C5S9	C4S9	C3S9	C2S9	C1S9	C0S9
LCDDS10	E6AH	C7S10	C6S10	C5S10	C4S10	C3S10	C2S10	C1S10	C0S10
LCDDS11	E6BH	C7S11	C6S11	C5S11	C4S11	C3S11	C2S11	C1S11	C0S11
LCDDS12	E6CH	C7S12	C6S12	C5S12	C4S12	C3S12	C2S12	C1S12	C0S12
LCDDS13	E6DH	C7S13	C6S13	C5S13	C4S13	C3S13	C2S13	C1S13	C0S13
LCDDS14	E6EH	C7S14	C6S14	C5S14	C4S14	C3S14	C2S14	C1S14	C0S14
LCDDS15	E6FH	C7S15	C6S15	C5S15	C4S15	C3S15	C2S15	C1S15	C0S15
LCDDS16	E70H	C7S16	C6S16	C5S16	C4S16	C3S16	C2S16	C1S16	C0S16
LCDDS17	E71H	C7S17	C6S17	C5S17	C4S17	C3S17	C2S17	C1S17	C0S17
LCDDS18	E72H	C7S18	C6S18	C5S18	C4S18	C3S18	C2S18	C1S18	C0S18
LCDDS19	E73H	C7S19	C6S19	C5S19	C4S19	C3S19	C2S19	C1S19	C0S19
LCDDS20	E74H	C7S20	C6S20	C5S20	C4S20	C3S20	C2S20	C1S20	C0S20
LCDDS21	E75H	C7S21	C6S21	C5S21	C4S21	C3S21	C2S21	C1S21	C0S21
LCDDS22	E76H	C7S22	C6S22	C5S22	C4S22	C3S22	C2S22	C1S22	C0S22
LCDDS23	E77H	C7S23	C6S23	C5S23	C4S23	C3S23	C2S23	C1S23	C0S23
LCDDS24	E78H	C7S24	C6S24	C5S24	C4S24	C3S24	C2S24	C1S24	C0S24
LCDDS25	E79H	C7S25	C6S25	C5S25	C4S25	C3S25	C2S25	C1S25	C0S25
LCDDS26	E7AH	C7S26	C6S26	C5S26	C4S26	C3S26	C2S26	C1S26	C0S26
LCDDS27	E7BH	C7S27	C6S27	C5S27	C4S27	C3S27	C2S27	C1S27	C0S27
LCDDS28	E7CH	C7S28	C6S28	C5S28	C4S28	C3S28	C2S28	C1S28	C0S28
LCDDS29	E7DH	C7S29	C6S29	C5S29	C4S29	C3S29	C2S29	C1S29	C0S29
LCDDS30	E7EH	C7S30	C6S30	C5S30	C4S30	C3S30	C2S30	C1S30	C0S30
LCDDS31	E7FH	C7S31	C6S31	C5S31	C4S31	C3S31	C2S31	C1S31	C0S31

注：LED RAM 和 LCD RAM 共享，不能同时使用。

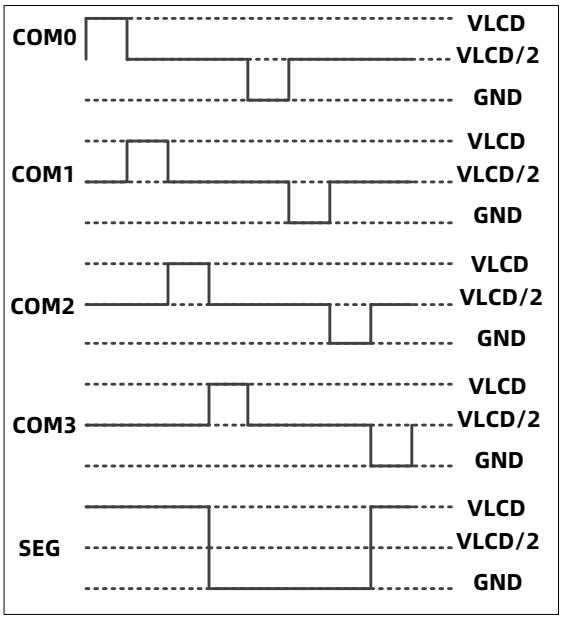
17.9 LCD 驱动波形

1/4Duty、1/2BIAS（SEG 输入 08）

A 类驱动模式

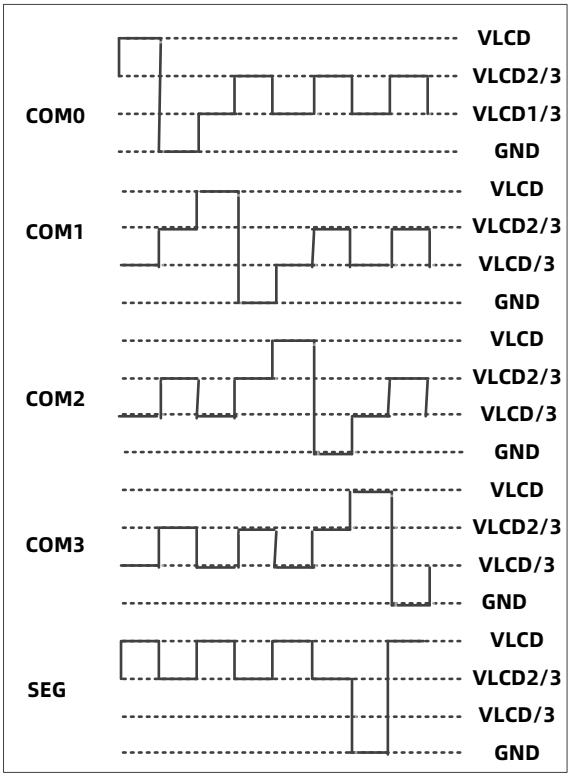


B 类驱动模式

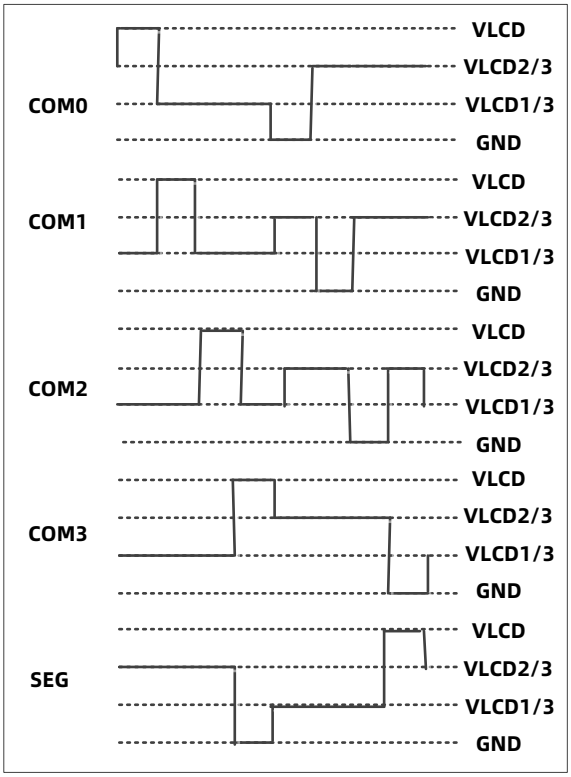


1/4Duty、1/3BIAS（SEG 输入 08）

A 类驱动模式

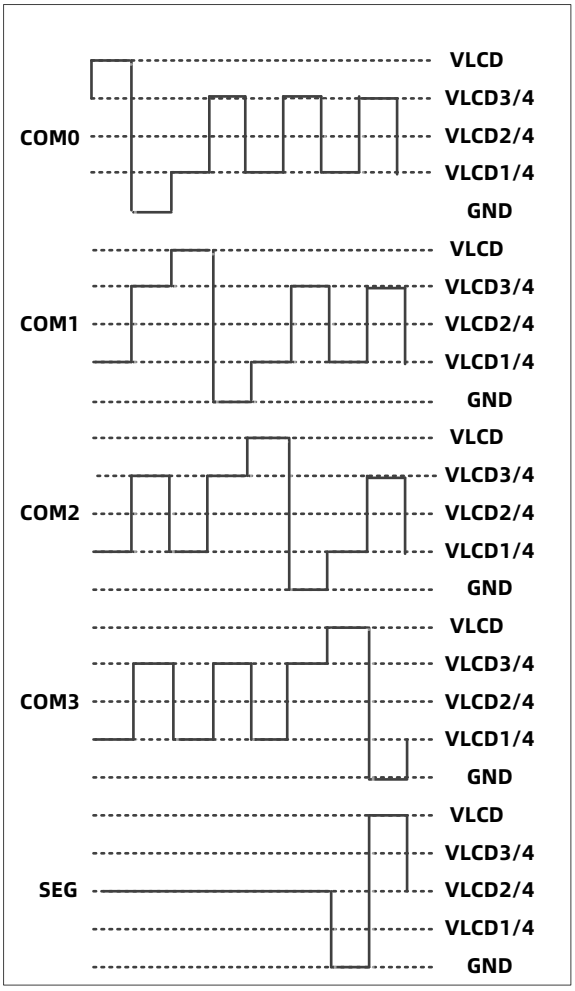


B 类驱动模式

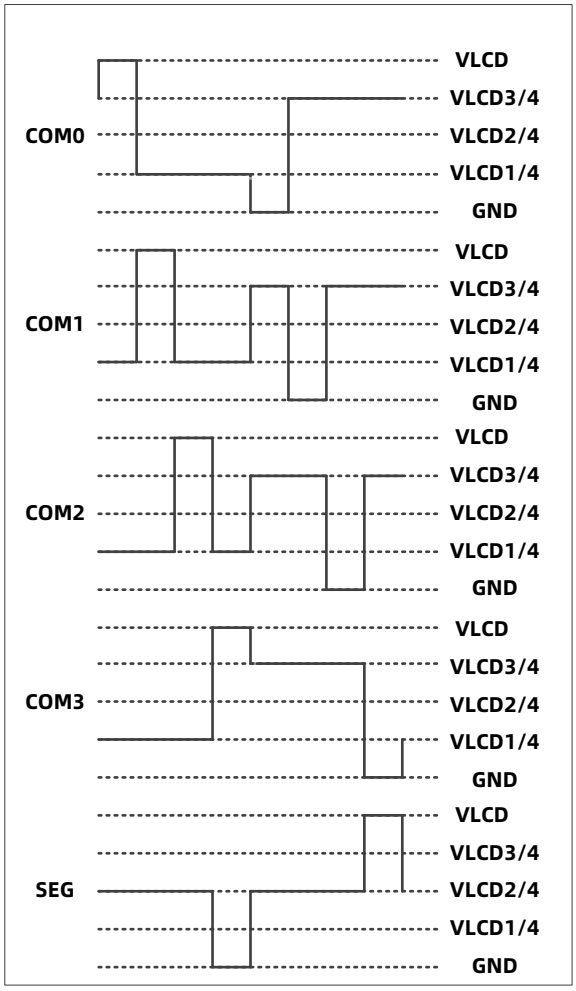


1/4Duty、1/4BIAS（SEG 输入 08）

A 类驱动模式



B 类驱动模式



17.10 LCD 操作示例

注：关于 LCD 应用请参照 DEMO 相关例程。

18 LED 模块

18.1 概述

M9F6601 LED 驱动模块由 LED 控制寄存器以及数据寄存器组成，可以驱动 LED 段码。

- 可选择 2-8COM
- 可调节帧频
- LED 模式支持多级亮度设置
- LED 模式支持 N 个 COM 口推 N x (N-1)个 LED
- 带调光LED驱动
- LED支持恒流源驱动
- 两种运行模式

亮灭 LED 模式：当 LED 工作在亮灭 LED 模式时，每一个 LEDRAM 位控制一个 LED 灯，当 LEDRAM 位为 0 时，LED 熄灭，当 LEDRAM 位为 1 时，LED 点亮；在 LED 一帧或者一个 COM 扫描结束后，LED 驱动器对应的中断标志位 LEDIF 标志位置 1。

调光 LED 模式：当 LED 驱动器工作在调光 LED 模式时，每一个 LEDRAMbyte 控制正在扫描的 COM 周期内 SEG 的占空比；该占空比总共可以 256 档可选；当 LEDRAMbyte 为 0xff 时，SEG 输出最大占空比，当 LEDRAMbyte 为 0x00 时，SEG 输出最小占空比；当 LEDRAMbyte 为 0x00-0xff 中间值时，SEG 输出相对应的占空比；针对 LEDRAMbyte 的修改，会在下一个 COM 扫描周期生效；在扫描完成一帧后，LED 驱动器对应的中断标志位 LEDIF 会置 1；

18.2 LCDCR0 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCR0	LCDEN	LCDM	LCDCKS	BIAS		CIOS[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 5 **LCDCKS：** LED 时钟选择

0 = Fosch

1 = Foscl

Bit [2:0] **CIOS：** LED COM 选择位

CIOS[2:0]	LED COM 选择位
000	Com0-7 作为 IO 口
001	Com0-1 作为 Com 口,Com2-7 作为 IO 口
010	Com0-2 作为 Com 口,Com3-7 作为 IO 口
011	Com0-3 作为 Com 口,Com4-7 作为 IO 口
100	Com0-4 作为 Com 口,Com5-7 作为 IO 口
101	Com0-5 作为 Com 口,Com6-7 作为 IO 口
110	Com0-6 作为 Com 口,Com7 作为 IO 口
111	Com0-7 作为 Com 口

注：（1）LED 模式下，LCDEN 必须写 0，LCDM 无效。
（2）LED 调光模式下，CIOS 选择当前选中 COM，未选中 COM 为 IO 端口状态。

18.3 LCDCR1 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCR1	LCDPR[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] **LCDPR：** 预分频

LCDCKS=0：（建议 LED 模式下使用）

帧频 = Fosch/((LCDPR+1)*256*m*n)

LCDCKS=1：（建议 LCD 低功耗时使用）

帧频 = Foscl/((LCDPR+1)*m*n)

注：m= com 数；n=seg 分组数（LED 模式）。

18.4 LCDCR2 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCR2	LEDEN	LEDOM	LEDMOD	LEDGS	LEDDRIV			
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 **LEDEN**: LED 功能使能位
0 = 屏蔽 LED 功能
1 = 使能 LED 功能

Bit 6 **LEDOM**: LED SEG 输出模式
0 = BUFF 驱动模式（IO）
1 = 电流源模式

Bit 5 **LEDMOD**: LED 模式选择
0 = 调光 LED 模式
1 = 亮灭 LED 模式

Bit 4 **LEDGS**: LEDSEG 单次扫描个数
0 = 4 个 SEG
1 = 8 个 SEG

LEDGS	GROP0	GROP1	GROP2	GROP3	GROP4	GROP5	GROP6	GROP7
0	SEG0 -3	SEG4-7	SEG8-11	SEG12-15	SEG16-19	SEG20-23	SEG24-27	SEG28-31
1	SEG0-7	SEG8-15	SEG16-23	SEG24-31				

注：某个 GROP 的 SEG 全部为选取，将不对改组进行扫描，尽量选取相同 GROP 内的 SEG。

Bit [3:0] **LEDDRIV:** LED 电流源调节
当 LED 使能时，SEG 电流源驱动选择

LEDDRIV[3:0]	输出电流	LEDDRIV[3:0]	输出电流
0000	2mA	1000	18 mA
0001	4 mA	1001	20 mA
0010	6 mA	1010	22 mA
0011	8 mA	1011	24 mA
0100	10 mA	1100	26 mA
0101	12 mA	1101	28 mA
0110	14 mA	1110	30 mA
0111	16 mA	1111	关闭

注：使用 LED 电流源调节时，需要将对应 SEG 口的驱动能力设置为“1”，具体参考 6.7 章节。

18.5 LCDCR3 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCR3	LEDSOUTMOD	LEDCOUTMOD	RAMMAP	VLCD4	VLCD3	VLCD2	VLCD1	VLCD0
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 **LEDSOUTMOD:** LEDSEGOUT 无效 PAD 状态

0 = 无效时为高阻态

1 = 无效时 SEG 为 0

Bit 6 **LEDCOUTMOD:** LEDCOMOUT 无效 PAD 状态

0 = 无效时为高阻态

1 = 无效时 COM 为 1

Bit 5 **RAMMAP:** LED 亮灭模式下 RAM 映射方式

0 = 方式 1

1 = 方式 2

注：8*7 模式下，要设置 COM 无效时为高组态。

18.6 LCDCR4 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCR4				-	LCDMODIF	-	OPAIS1	OPAIS0
读/写	-	-	-	-	R/W	-	R/W	R/W
复位后	-	-	-	0	0	-	0	0

Bit 4 **Reserved:** 保留，如果使用 LED 时请保持为 1

Bit 3 **LCDMODIF:** 亮灭模式下 LED 中断运行模式控制位

0 = LEDIF 为 1 时，LED 扫描继续

1 = LEDIF 为 1 时，LED 扫描停止，需要将 LEDIF 清 0 开始下一帧扫描

18.7 SEGIOSx 端口设置（x=A/B/C/D）

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGIOSA	SEGIOSA7	SEGIOSA6	SEGIOSA5	SEGIOSA4	SEGIOSA3	SEGIOSA2	SEGIOSA1	SEGIOSA0
SEGIOSB	SEGIOSB7	SEGIOSB6	SEGIOSB5	SEGIOSB4	SEGIOSB3	SEGIOSB2	SEGIOSB1	SEGIOSB0
SEGIOSC	SEGIOSC7	SEGIOSC6	SEGIOSC5	SEGIOSC4	SEGIOSC3	SEGIOSC2	SEGIOSC1	SEGIOSC0
SEGIOSD	SEGIOSD7	SEGIOSD6	SEGIOSD5	SEGIOSD4	SEGIOSD3	SEGIOSD2	SEGIOSD1	SEGIOSD0
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] **SEGIOSxn:** SEG口选择（x=A/B/C/D,n=0-7）

0 = 作为IO

1 = 作为SEG

注：

（1）在未开启 LCD 和 LED 的情况下，可以控制 SEGIOSx 控制 IOx（x=A/B/C/D）使用电流源模式。

（2）寄存器具体地址参考<2.2.8 系统寄存器定义>

18.8 LCDDSxx：LED 显示数据寄存器

LED 亮灭模式 RAM 映射方式 1

地址		Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
E60H	**COM0**	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	-
E61H		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E62H		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E63H		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E64H	**COM1**	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	-	SEG0
E65H		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E66H		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E67H		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E68H	**COM2**	SEG7	SEG6	SEG5	SEG4	SEG3	-	SEG1	SEG0
E69H		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E6AH		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E6BH		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E6CH	**COM3**	SEG7	SEG6	SEG5	SEG4	-	SEG2	SEG1	SEG0
E6DH		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E6EH		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E6FH		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E70H	**COM4**	SEG7	SEG6	SEG5	-	SEG3	SEG2	SEG1	SEG0
E71H		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E72H		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E73H		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E74H	**COM5**	SEG7	SEG6	-	SEG4	SEG3	SEG2	SEG1	SEG0
E75H		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E76H		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E77H		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E78H	**COM6**	SEG7	-	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
E79H		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E7AH		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E7BH		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
E7CH	**COM7**	-	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
E7DH		SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
E7EH		SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
E7FH		SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24

LED 亮灭模式 RAM 映射方式 2：

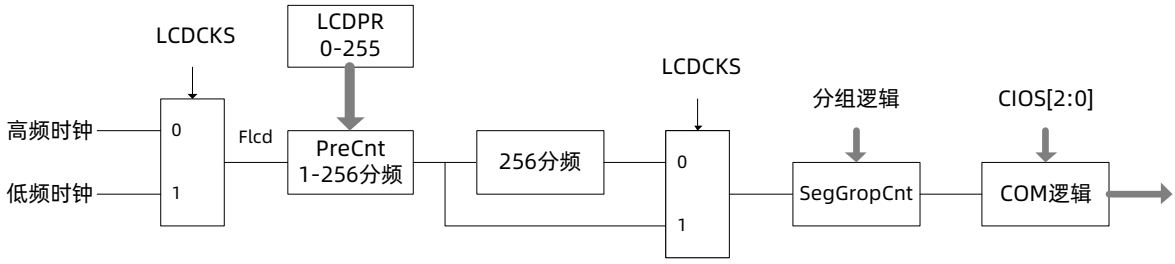
寄存器	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0
LCDDS00	E60H	C7S0	C6S0	C5S0	C4S0	C3S0	C2S0	C1S0	C0S0
LCDDS01	E61H	C7S1	C6S1	C5S1	C4S1	C3S1	C2S1	C1S1	C0S1
LCDDS02	E62H	C7S2	C6S2	C5S2	C4S2	C3S2	C2S2	C1S2	C0S2
LCDDS03	E63H	C7S3	C6S3	C5S3	C4S3	C3S3	C2S3	C1S3	C0S3
LCDDS04	E64H	C7S4	C6S4	C5S4	C4S4	C3S4	C2S4	C1S4	C0S4
LCDDS05	E65H	C7S5	C6S5	C5S5	C4S5	C3S5	C2S5	C1S5	C0S5
LCDDS06	E66H	C7S6	C6S6	C5S6	C4S6	C3S6	C2S6	C1S6	C0S6
LCDDS07	E67H	C7S7	C6S7	C5S7	C4S7	C3S7	C2S7	C1S7	C0S7
LCDDS08	E68H	C7S8	C6S8	C5S8	C4S8	C3S8	C2S8	C1S8	C0S8
LCDDS09	E69H	C7S9	C6S9	C5S9	C4S9	C3S9	C2S9	C1S9	C0S9
LCDDS10	E6AH	C7S10	C6S10	C5S10	C4S10	C3S10	C2S10	C1S10	C0S10
LCDDS11	E6BH	C7S11	C6S11	C5S11	C4S11	C3S11	C2S11	C1S11	C0S11
LCDDS12	E6CH	C7S12	C6S12	C5S12	C4S12	C3S12	C2S12	C1S12	C0S12
LCDDS13	E6DH	C7S13	C6S13	C5S13	C4S13	C3S13	C2S13	C1S13	C0S13
LCDDS14	E6EH	C7S14	C6S14	C5S14	C4S14	C3S14	C2S14	C1S14	C0S14
LCDDS15	E6FH	C7S15	C6S15	C5S15	C4S15	C3S15	C2S15	C1S15	C0S15
LCDDS16	E70H	C7S16	C6S16	C5S16	C4S16	C3S16	C2S16	C1S16	C0S16
LCDDS17	E71H	C7S17	C6S17	C5S17	C4S17	C3S17	C2S17	C1S17	C0S17
LCDDS18	E72H	C7S18	C6S18	C5S18	C4S18	C3S18	C2S18	C1S18	C0S18
LCDDS19	E73H	C7S19	C6S19	C5S19	C4S19	C3S19	C2S19	C1S19	C0S19
LCDDS20	E74H	C7S20	C6S20	C5S20	C4S20	C3S20	C2S20	C1S20	C0S20
LCDDS21	E75H	C7S21	C6S21	C5S21	C4S21	C3S21	C2S21	C1S21	C0S21
LCDDS22	E76H	C7S22	C6S22	C5S22	C4S22	C3S22	C2S22	C1S22	C0S22
LCDDS23	E77H	C7S23	C6S23	C5S23	C4S23	C3S23	C2S23	C1S23	C0S23
LCDDS24	E78H	C7S24	C6S24	C5S24	C4S24	C3S24	C2S24	C1S24	C0S24
LCDDS25	E79H	C7S25	C6S25	C5S25	C4S25	C3S25	C2S25	C1S25	C0S25
LCDDS26	E7AH	C7S26	C6S26	C5S26	C4S26	C3S26	C2S26	C1S26	C0S26
LCDDS27	E7BH	C7S27	C6S27	C5S27	C4S27	C3S27	C2S27	C1S27	C0S27
LCDDS28	E7CH	C7S28	C6S28	C5S28	C4S28	C3S28	C2S28	C1S28	C0S28
LCDDS29	E7DH	C7S29	C6S29	C5S29	C4S29	C3S29	C2S29	C1S29	C0S29
LCDDS30	E7EH	C7S30	C6S30	C5S30	C4S30	C3S30	C2S30	C1S30	C0S30
LCDDS31	E7FH	C7S31	C6S31	C5S31	C4S31	C3S31	C2S31	C1S31	C0S31

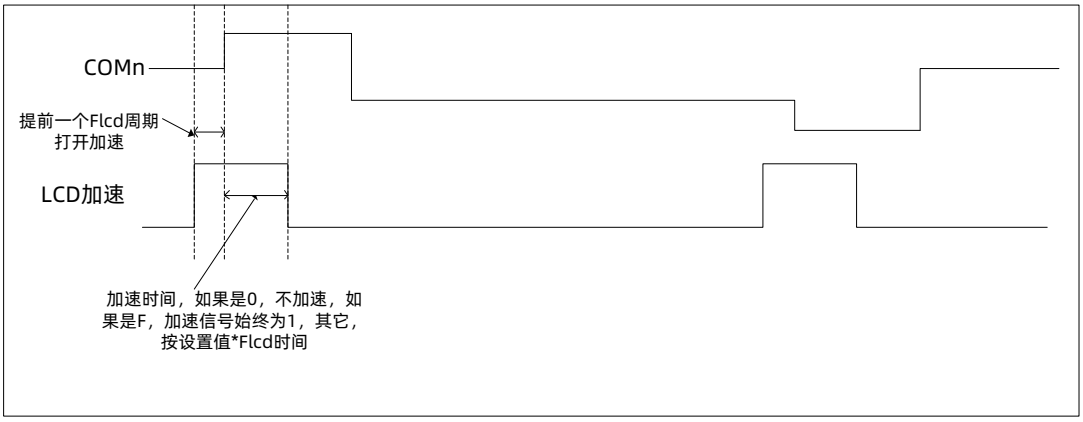
LED 调光模式下，每个寄存器的数据控制 SEG 占空比

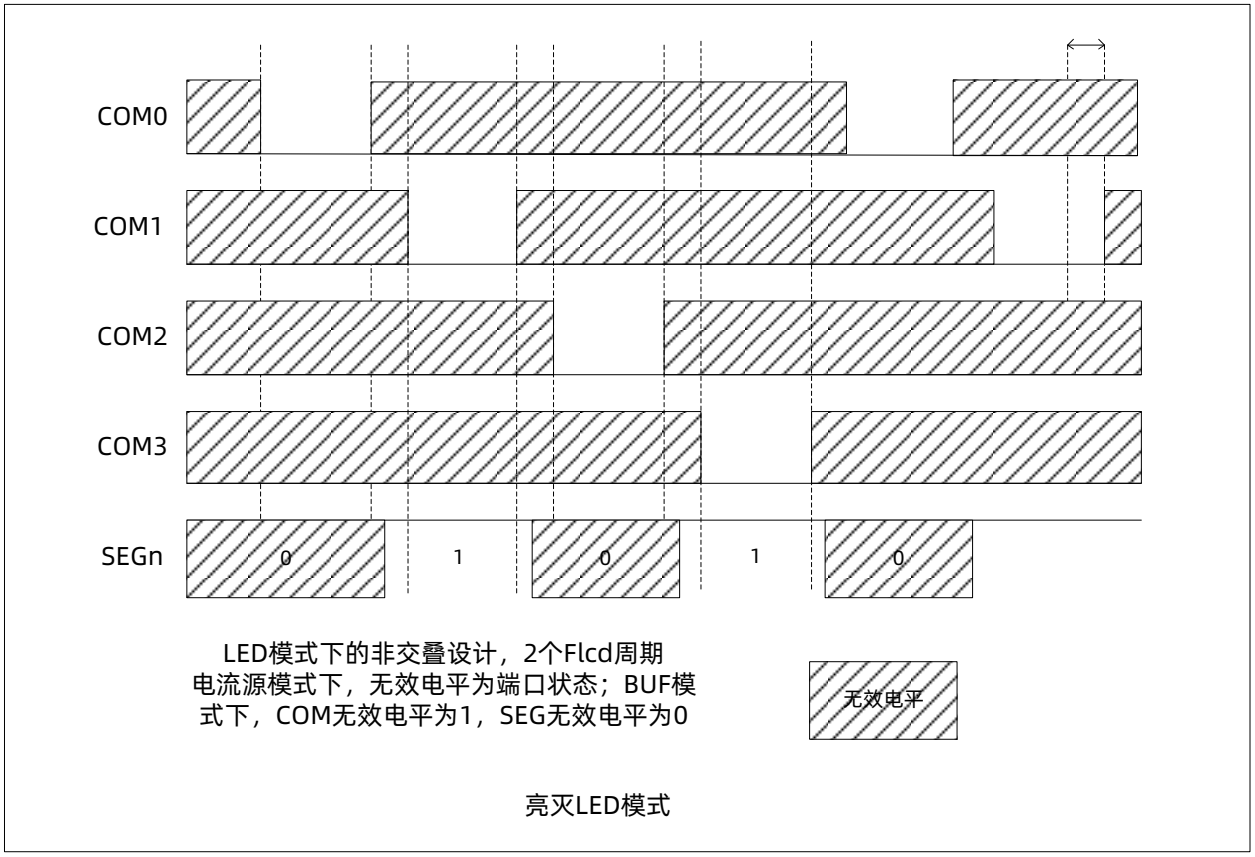
寄存器	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEG00DUTY	E60H	DY0.7	DY0.6	DY0.5	DY0.4	DY0.3	DY0.2	DY0.1	DY0.0
SEG01DUTY	E61H	DY1.7	DY1.6	DY1.5	DY1.4	DY1.3	DY1.2	DY1.1	DY1.0
SEG02DUTY	E62H	DY2.7	DY2.6	DY2.5	DY2.4	DY2.3	DY2.2	DY2.1	DY2.0
SEG03DUTY	E63H	DY3.7	DY3.6	DY3.5	DY3.4	DY3.3	DY3.2	DY3.1	DY3.0
SEG04DUTY	E64H	DY4.7	DY4.6	DY4.5	DY4.4	DY4.3	DY4.2	DY4.1	DY4.0
SEG05DUTY	E65H	DY5.7	DY5.6	DY5.5	DY5.4	DY5.3	DY5.2	DY5.1	DY5.0
SEG06DUTY	E66H	DY6.7	DY6.6	DY6.5	DY6.4	DY6.3	DY6.2	DY6.1	DY6.0
SEG07DUTY	E67H	DY7.7	DY7.6	DY7.5	DY7.4	DY7.3	DY7.2	DY7.1	DY7.0
SEG08DUTY	E68H	DY8.7	DY8.6	DY8.5	DY8.4	DY8.3	DY8.2	DY8.1	DY8.0
SEG09DUTY	E69H	DY9.7	DY9.6	DY9.5	DY9.4	DY9.3	DY9.2	DY9.1	DY9.0
SEG10DUTY	E6AH	DY10.7	DY10.6	DY10.5	DY10.4	DY10.3	DY10.2	DY10.1	DY10.0
SEG11DUTY	E6BH	DY11.7	DY11.6	DY11.5	DY11.4	DY11.3	DY11.2	DY11.1	DY11.0
SEG12DUTY	E6CH	DY12.7	DY12.6	DY12.5	DY12.4	DY12.3	DY12.2	DY12.1	DY12.0
SEG13DUTY	E6DH	DY13.7	DY13.6	DY13.5	DY13.4	DY13.3	DY13.2	DY13.1	DY13.0
SEG14DUTY	E6EH	DY14.7	DY14.6	DY14.5	DY14.4	DY14.3	DY14.2	DY14.1	DY14.0
SEG15DUTY	E6FH	DY15.7	DY15.6	DY15.5	DY15.4	DY15.3	DY15.2	DY15.1	DY15.0
SEG16DUTY	E70H	DY16.7	DY16.6	DY16.5	DY16.4	DY16.3	DY16.2	DY16.1	DY16.0
SEG17DUTY	E71H	DY17.7	DY17.6	DY17.5	DY17.4	DY17.3	DY17.2	DY17.1	DY17.0
SEG18DUTY	E72H	DY18.7	DY18.6	DY18.5	DY18.4	DY18.3	DY18.2	DY18.1	DY18.0
SEG19DUTY	E73H	DY19.7	DY19.6	DY19.5	DY19.4	DY19.3	DY19.2	DY19.1	DY19.0
SEG20DUTY	E74H	DY20.7	DY20.6	DY20.5	DY20.4	DY20.3	DY20.2	DY20.1	DY20.0
SEG21DUTY	E75H	DY21.7	DY21.6	DY21.5	DY21.4	DY21.3	DY21.2	DY21.1	DY21.0
SEG22DUTY	E76H	DY22.7	DY22.6	DY22.5	DY22.4	DY22.3	DY22.2	DY22.1	DY22.0
SEG23DUTY	E77H	DY23.7	DY23.6	DY23.5	DY23.4	DY23.3	DY23.2	DY23.1	DY23.0
SEG24DUTY	E78H	DY24.7	DY24.6	DY24.5	DY24.4	DY24.3	DY24.2	DY24.1	DY24.0
SEG25DUTY	E79H	DY25.7	DY25.6	DY25.5	DY25.4	DY25.3	DY25.2	DY25.1	DY25.0
SEG26DUTY	E7AH	DY26.7	DY26.6	DY26.5	DY26.4	DY26.3	DY26.2	DY26.1	DY26.0
SEG27DUTY	E7BH	DY27.7	DY27.6	DY27.5	DY27.4	DY27.3	DY27.2	DY27.1	DY27.0
SEG28DUTY	E7CH	DY28.7	DY28.6	DY28.5	DY28.4	DY28.3	DY28.2	DY28.1	DY28.0
SEG29DUTY	E7DH	DY29.7	DY29.6	DY29.5	DY29.4	DY29.3	DY29.2	DY29.1	DY29.0
SEG30DUTY	E7EH	DY30.7	DY30.6	DY30.5	DY30.4	DY30.3	DY30.2	DY30.1	DY30.0
SEG31DUTY	E7FH	DY31.7	DY31.6	DY31.5	DY31.4	DY31.3	DY31.2	DY31.1	DY31.0

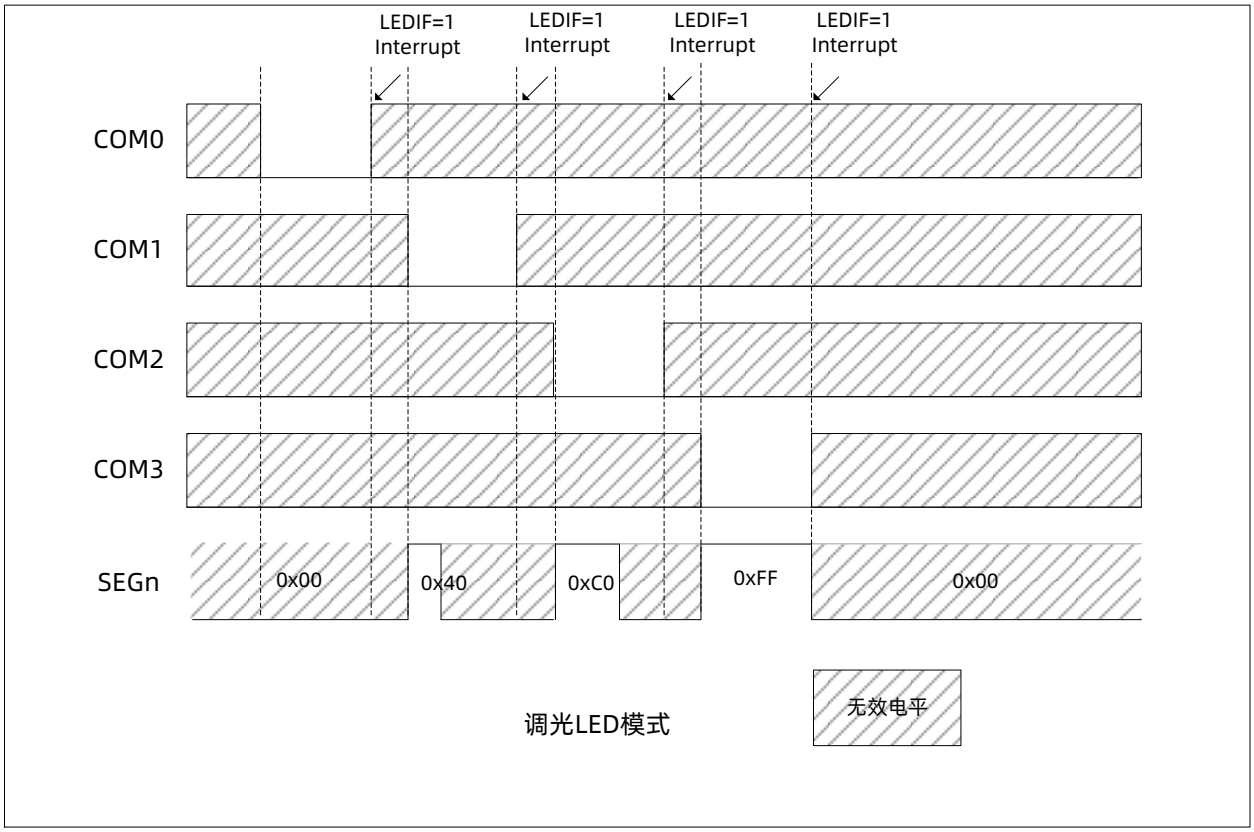
注：LED RAM 和 LCD RAM 共享，不能同时使用。

18.9 LED 驱动波形

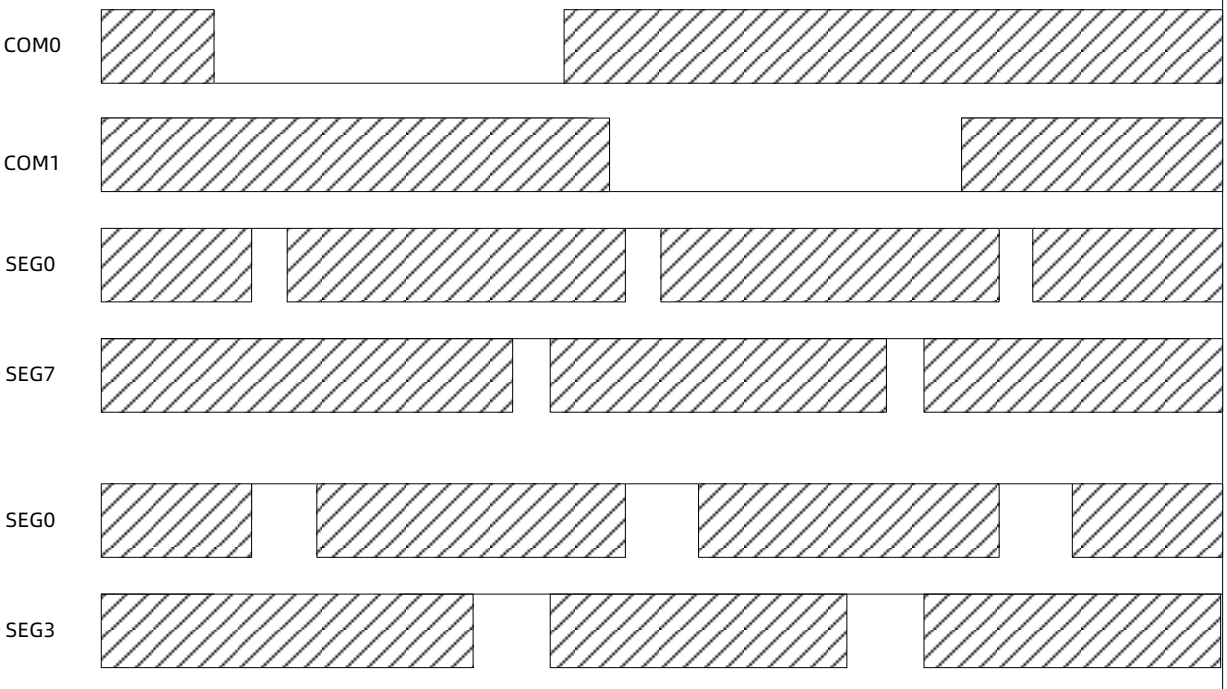

LCDCKS
LCDPR
0-255
LCDCKS
分组逻辑
CIOS[2:0]
高频时钟
低频时钟
0
1
Flcd
PreCnt
1-256分频
256分频
0
1
SegGropCnt
COM逻辑


COMn
提前一个Flcd周期
打开加速
LCD加速
加速时间，如果是0，不加速，如果是F，加速信号始终为1，其它，按设置值*Flcd时间


COM0
COM1
COM2
COM3
SEGn
0
1
0
1
0
LED模式下的非交叠设计，2个Flcd周期
电流源模式下，无效电平为端口状态；BUF模式下，COM无效电平为1，SEG无效电平为0
无效电平
亮灭LED模式



调光LED模式

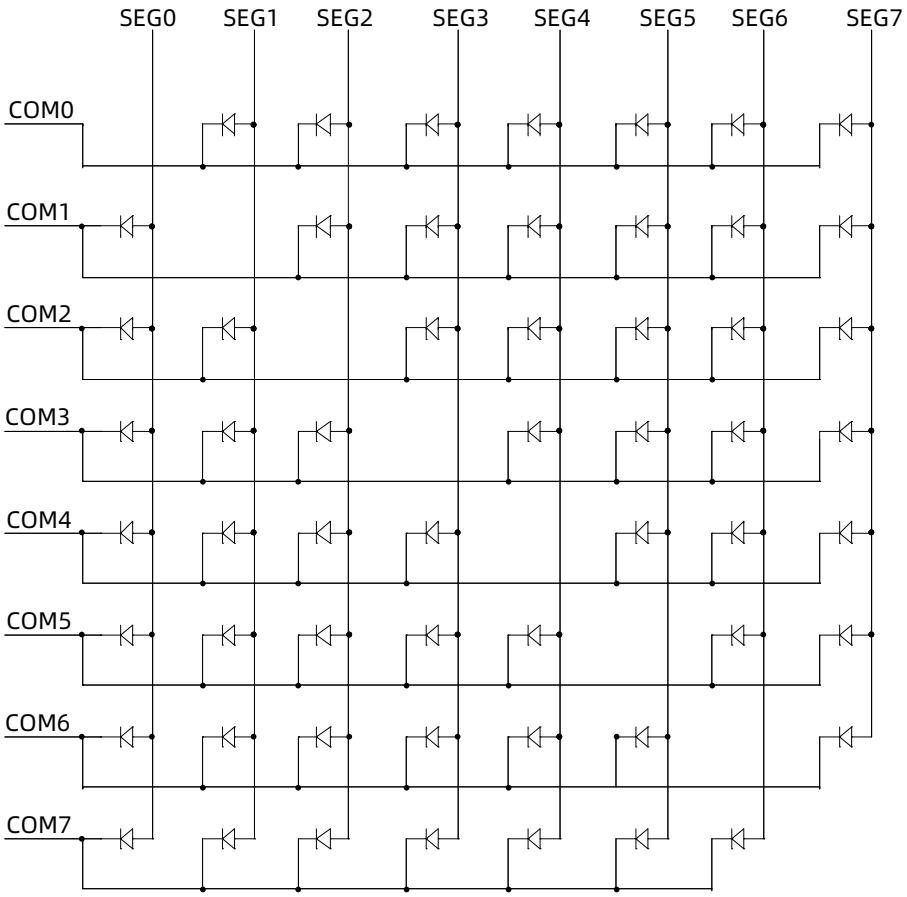


LED模式下，每次4个或8个SEG有效，最大分组，8组；如果某组内没有SEG使能，改组的有效时间为0；SEG选用时，尽量使有效组降低，以增加LCD的显示最大亮度

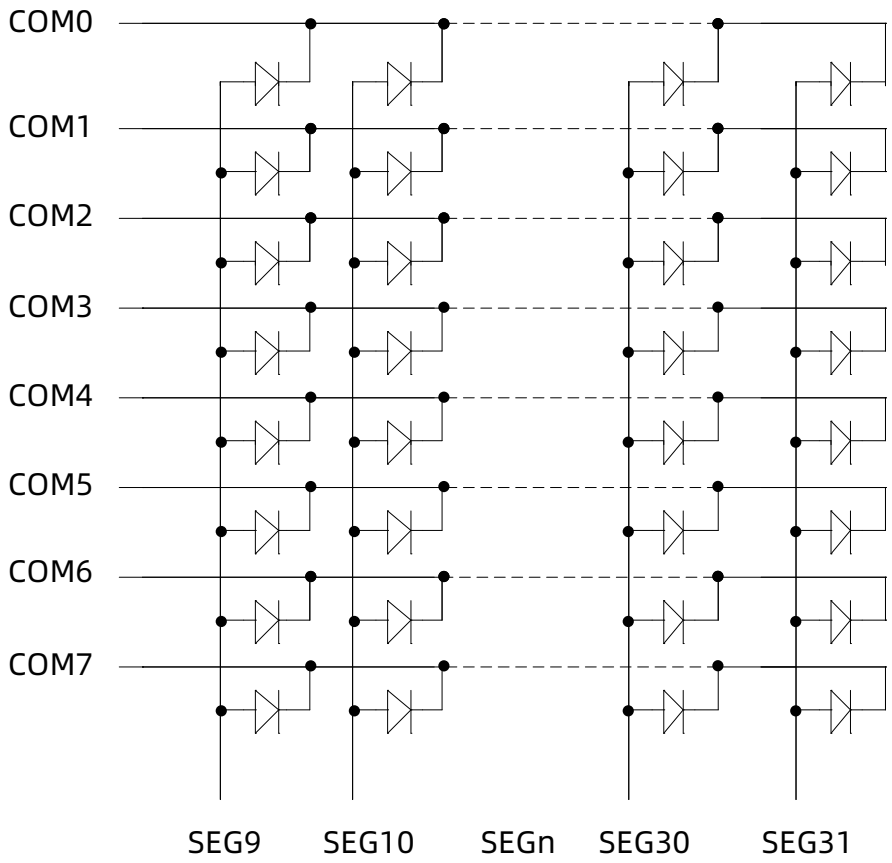
LED分组

18.10 LED 逻辑图

8*7LED 接法逻辑图：



8*24 LED 接法逻辑图：



18.11 LED 操作示例

注：关于 LED 应用请参照 DEMO 相关例程。

19 乘除法器

19.1 概述

M9F6601 提供 1 个 16 位硬件乘除法器，由 EXA0~EXA3，EXB0，EXB1 以及运算控制寄存器 OPERCON。乘除法器不占用 CPU 周期，运算由硬件完成，速度比软件实现快几十倍，可代替软件进行 16 位 x16 位乘法运算。32 位/16 位除法运算，提高程序运行效率。

19.2 MDUCON 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MDUCON	OPERS	MD	-	-	-	-	-	-
读/写	R/W	R/W	R	R	R	R	R	R
复位后	0	0	0	0	0	0	0	0

Bit7 **OPERS**： 运算开始触发控制

0 = 计算完成

1 = 写入 1 后开始计算，计算完成自动清 0

注：乘除法器运算转换所需时间为 16/f_{CPU}。

Bit6 **MD：** 乘除法运算选择

0 = 乘法运算

1 = 除法运算

19.3 EXAx 扩展累加器(x=0/1/2/3)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EXAx	EXAx							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit[7:0] **EXAx：** 扩展累加器

19.4 EXBx 扩展 B 寄存器(x=0/1)

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EXBx	EXBx							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit[7:0] **EXBx：** 扩展B寄存器

19.5 乘除法器使用说明

19.5.1 乘法运算

	字节 3	字节 2	字节 1	字节 0
被乘数 16bit	-	-	EXA1	EXA0
乘数 16bit	-	-	EXB1	EXB0
乘积 32bit	EXA3	EXA2	EXA1	EXA0

乘法运算例程：

```
unsigned int   multiplier_data;
unsigned int   multiplicand_data;
unsigned long result_data;

EXB0 = multiplier_data & 0xFF;       // 乘数低8位
EXB1 = multiplier_data >> 8;         // 乘数高8位
EXA0 = multiplicand_data & 0xFF;     // 被乘数低8位
EXA1 = multiplicand_data >> 8;       // 被乘数高8位

MDUCON = 0x80;                       // 启动乘法运算
while(MDUCON & 0X80);                // 等待运算完成
result_data = ((u32)EXA3 << 24)|((u32)EXA2 << 16)|((u32)EXA1 << 8)|EXA0;//读取32位结果
```

19.5.2 除法运算

	字节 3	字节 2	字节 1	字节 0
被除数 32bit	EXA3	EXA2	EXA1	EXA0
除数 16bit	-	-	EXB1	EXB0
商 32bit	EXA3	EXA2	EXA1	EXA0
余数 16bit			EXB1	EXB0

除法运算例程：

```
unsigned long   dividend_data;              // 32位被除数
unsigned int    divisor_data;               // 16位除数
unsigned long   quotient_data;              // 32位商
unsigned int    remainder_data;             // 16位余数

EXA0 = dividend_data & 0xFF;                //  被除数最低字节
EXA1 = (dividend_data >> 8) & 0xFF;         //  被除数次低字节
EXA2 = (dividend_data >> 16) & 0xFF;        //  被除数次高字节
EXA3 = dividend_data >> 24;                 //  被除数最高字节
EXB0 = divisor_data & 0xFF;                 //  除数低8位
EXB1 = divisor_data >> 8;                   //  除数高8位

MDUCON = 0xC0;                              //  启动除法运算
while (MDUCON & 0x80);                      //  等待运算完成

quotient_data=((u32)EXA3 << 24)|((u32)EXA2 << 16)|((u32)EXA1 << 8)|EXA0; // 32位商
remainder_data = ((u16)EXB1 << 8) | EXB0;       // 16位余数
```

20 循环冗余校验单元(CRC)

20.1 概述

为了保证运行过程中的安全和提高 CPU 运行效率，内置了一个硬件 CRC 模块，执行 CRC16-CCITT 标准。该 CRC 模块能在 CPU 运行中作为外围功能进行 CRC 运算。通用 CRC 模块通过程序指定要确认的数据进行 CRC 校验，不限于代码闪存区而能用于多用途的检查。

硬件 CRC 参数模型：

CRC算法名称	CRC16-CCITT
多项式公式	$X^{16}+X^{12}+X^5+1$
数据宽度	16
初始值	0x0000
结果异或值	0x0000
输入值反转	True
输出值反转	True
LSB/MSB	LSB

20.2 CRCCR 寄存器

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

Bit0 **CRCBIT：** CRC BIT 翻转控制位

0 = MSB first

1 = LSB first

20.3 CRCIN 寄存器

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CRCIN	CRCIN[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] **CRCIN[7:0]**： CRC 数据输入寄存器

注：写入数据自动进行 CRC 计算

注：后续寄存器地址设定的时候，尽量放在快速 sfr

20.4 CRCDL 寄存器

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

Bit[7:0] **CRCDL[7:0]:** CRC 结果寄存器低位

20.5 CRCDH 寄存器

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

Bit[7:0] **CRCDH[7:0]:** CRC 结果寄存器高位

注:写入数据时为写 CRC 计算初始值

读出数据时为 CRC 计算结果值

20.6 CRC 功能描述

在写 CRCIN 寄存器后经过 1 个系统时钟，将 CRC 的运算结果保存到 CRCDL/CRCDH 寄存器。如有需要，则在写入覆盖之 前需读取前一次的运算数据，否则会被新的运算结果覆盖。

注：在进行新一轮的 CRC 计算时，需要对 CRC 初始值进行赋值，以防止上次计算结果的影响。

例：单字节 CRC 计算发送数据“0x55”，首先给 CRC 初始值进行赋值，再给 CRCIN 寄存器写值，写完后从 CRCDH/CRCDL 寄存器读取为 CCRCDH = 0x05,CRCDL = 0x28。寄存器操作如下。

```
CRCCR = 1;                  //Bit 反转
CRCDH = 0;
CRCDL = 0;                  //初始化初始值
CRCIN = 0x55;               //写入要计算的数据
Resulth = CRCDH;
Resultl = CRCDL;            //读取数据
```

例：多字节 CRC 计算发送数据“0x12345678”，首先给 CRC 初始值进行赋值，再给 CRCIN 寄存器按照“0x12”、“0x34”、“0x56”、“0x78” 的顺序写入，写完后从 CRCDH/CRCDL 寄存器读取为 CCRCDH = 0x67,CRCDL = 0xF0。寄存器操作如下。

```
CRCCR = 1;                  //Bit 反转
CRCDH = 0;
CRCDL = 0;                  //初始化初始值
CRCIN = 0x12;               //按顺序写入要计算的数据
CRCIN = 0x34;
CRCIN = 0x56;
CRCIN = 0x78;
Resulth = CRCDH;
Resultl = CRCDL;            //读取数据
```

21 看门狗（WDT）

21.1 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 4ms、16ms、64ms、 256ms、512ms、1s、2s 或 4s。

000	4ms	100	512ms
001	16ms	101	1s
010	64ms	110	2s
011	256ms	111	4s

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。

22 芯片配置字（OPTION BIT）

烧录选项	内容	说明
振荡器模式	HIRC(32MHz)+LIRC	双系统时钟
	HIRC(32MHz)+LXT	
	HXT(32MHz)+LIRC	
FCPU	4T(Vlvr>2.4V)	高频模式下 CPU 速度选择
	8T	
	16T	
	32T	
	64T	
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=1.8V(FCPU<4T)	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=2.2V(FCPU<4T)	
	LVR=2.5V	
	LVR=2.8V	
	LVR=3.4V	
进仿真模式选择	A[0]和 A[1]为普通口	
	A[0]和 A[1]只做 PSDA/PSCK 使用	
低频晶振供电选择	芯片 VDD 供电	
	VCRY 供电	
芯片代码加密	不使能	
	使能	
WDT 溢出时间	4ms	VDD=5V 典型值
	16ms	
	64ms	
	256ms	
	512ms	
	1s	
	2s	
	4s	

烧录选项	内容	说明
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	
APP 读写备份区允许	不允许	
	允许	
APP 读取 BOOT 区	不允许	
	允许	
BOOT 数据区写 APP 区允许	不允许	
	允许	
BOOT 数据区写备份区允许	不允许	
	允许	
BOOTAPP 区写 BOOT 数据区允许	不允许	
	允许	
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	
备份区域大小	无备份	类 EE 区起始地址=0x10000-备份区大小-BOOT 区大小-类 EE 区大小 例：备份区=24KB，类 EE 区=2KB，BOOT 区=1KB 备份区起始地址=0x10000-0x400*24-0x400*2-0x400*1=0x9400 类 EE 区起始地址=0x10000-0x400*2-0x400*1=0XF400 BOOT 区起始地址=0xFC00
	24KB	
	24.5KB	
	25KB	
	25.5KB	
	26KB	
	26.5KB	
	27KB	
	27.5KB	
	28KB	
	28.5KB	
	29KB	
	29.5KB	
	30KB	

烧录选项	内容	说明
类 EE 区域大小	无类 EE	
	0.5KB	
	1KB	
	1.5KB	
	2KB	
	2.5KB	
	3KB	
	3.5KB	
	4KB	
	4.5KB	
	5KB	
	5.5KB	
	6KB	
	6.5KB	
	7KB	
	7.5KB	
	8KB	
BOOT 区域大小	无 BOOT 区	括号中为 BOOT 区起始地址
	1KB(0xFC00)	
	2KB(0xF800)	
	3KB(0xF400)	
	4KB(0xF000)	
	5KB(0xEC00)	
	6KB(0xE800)	
	7KB(0xE400)	
	8KB(0xE000)	
	9KB(0xDC00)	
	10KB(0xD800)	
	11KB(0xD400)	
	12KB(0xD000)	
	13KB(0xCC00)	
	14KB(0xC800)	
	15KB(0xC400)	
	16KB(0xC000)	

23 电性参数

23.1 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。

23.2 直流特性

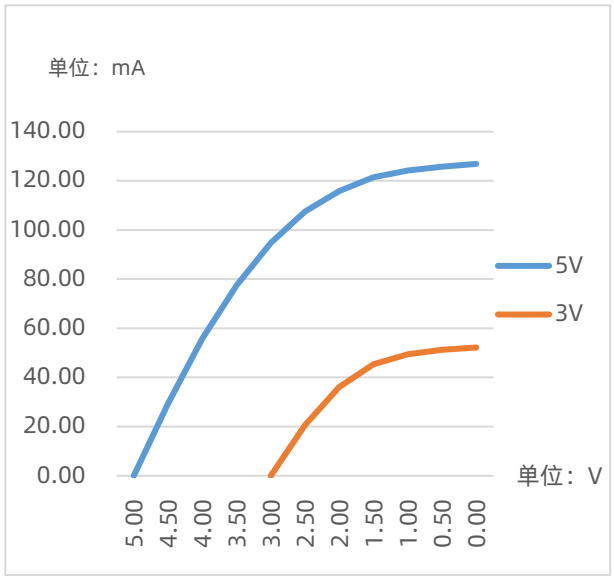
符号	参数	测试条件		最小值	典型值	最大值	单位
		VDD	条件（常温 25℃）				
VDD	工作电压	—	Fosc = 32MHz，32T	2.2	-	5.5	V
I_{DD1}	工作电流 1	3V	高频运行（HIRC=32MHz） 低频运行（LIRC=64KHz） F_{CPU}=HIRC/4T 全速工作	-	2.8	-	mA
		5V		-	4.4	-	
I_{DD2}	工作电流 2	3V	高频停止 低频运行（LIRC=64KHz）	-	11	-	μA
		5V		-	13	-	
I_{SP1}	静态电流 1	3V	高频运行（HIRC=32MHz） 低频运行（LIRC=64KHz） STOP =1 无唤醒源	-	0.5	-	mA
		5V		-	0.8	-	
I_{SP2}	静态电流 2	3V	高频停止 低频运行（LIRC=64KHz） STOP =1 无唤醒源	-	3	-	μA
		5V		-	5	-	
I_{SP3}	静态电流 3	3V	高频停止 外部低频运行(LXT=32768HZ) STOP=1 无唤醒源	-	3.0	-	
		5V		-	7	-	
I_{SP4}	静态电流 4	3V	高频停止 低频停止 STOP =1 无唤醒源	-	0.8	-	
		5V		-	0.9	-	

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℃）				
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	
I_{PH}	上拉电阻	5V	V_{IN} = GND	-	15	-	kΩ
		3V	V_{IN} = GND	-	25	-	
I_{PL}	下拉电阻	5V	V_{IN} = VDD	-	45	-	
		3V	V_{IN} = VDD	-	81	-	
I_{OL1}	输出灌电流	5V	输出口，Vout=GND+0.6V (IOA/IOB/IOC/ IOE/IOF)		35		mA
		3V			24		
I_{OL2}	输出灌电流	5V			10.5		
		3V			7		
I_{OL3}	输出灌电流	5V	输出口，Vout=GND+0.6V (IOD)		94		
		3V			72		
I_{OL4}	输出灌电流	5V			12		
		3V			9		
I_{OH1}	输出拉电流	5V	输出口，Vout=VDD-0.6V (IOA/IOB/IOC/ IOD / IOE/IOF)		41		
		3V			25		
I_{OH2}	输出拉电流	5V			10		
		3V			6		

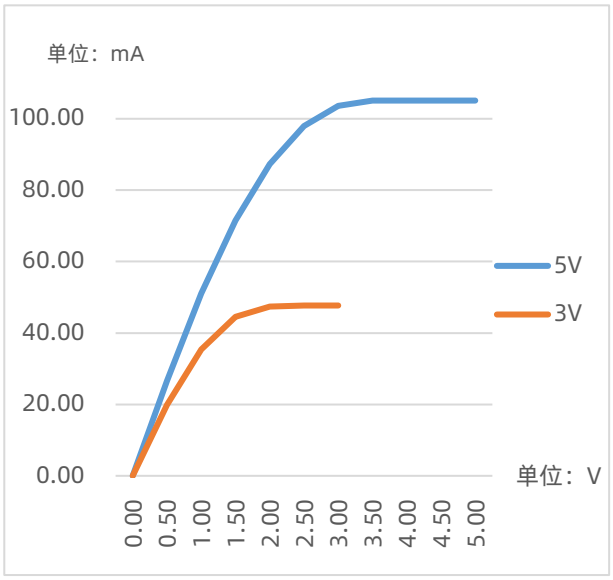
注：具体值不做设计保证。

23.3 IO 口拉灌电流特性

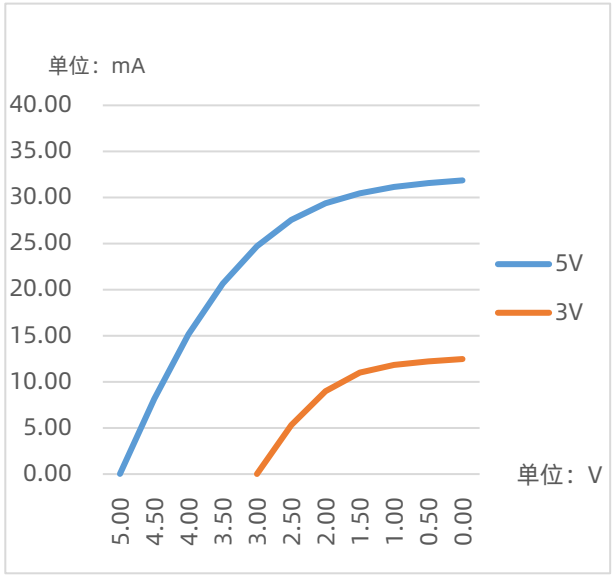
IOH_1



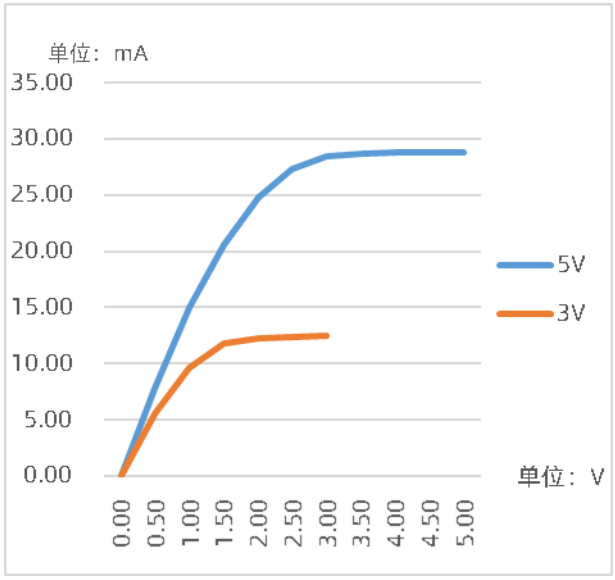
IOL_1



IOH_2

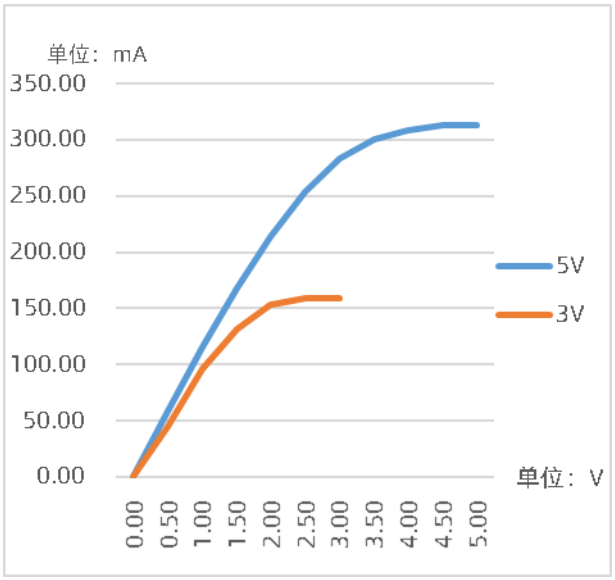


IOL_2

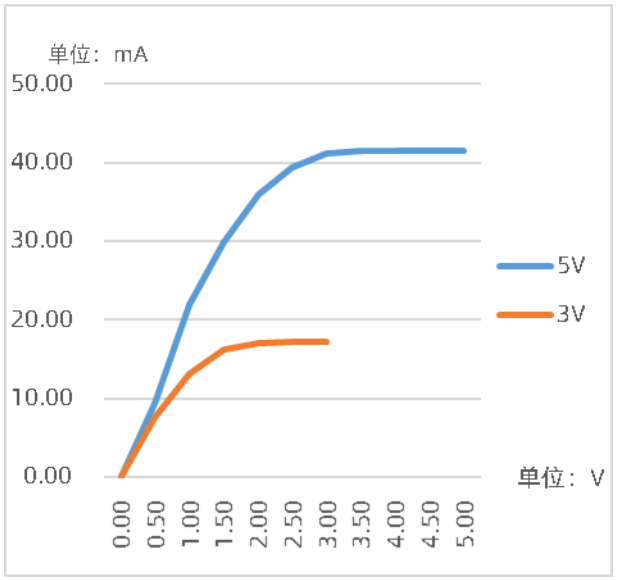


注：具体值不做设计保证。

IOL_3



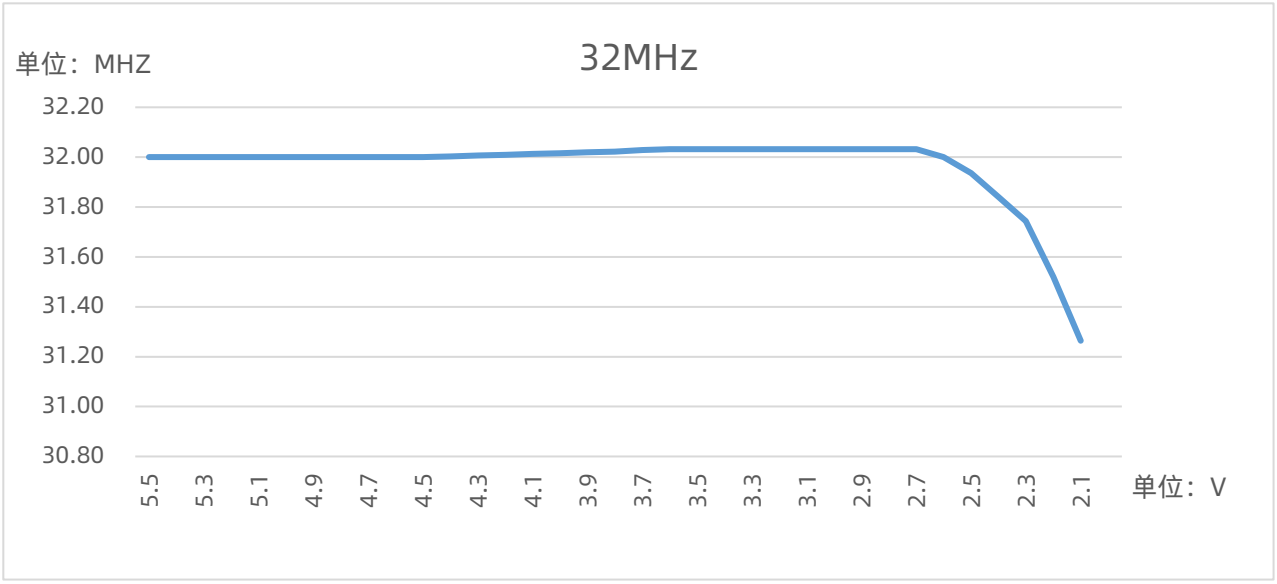
IOL_4



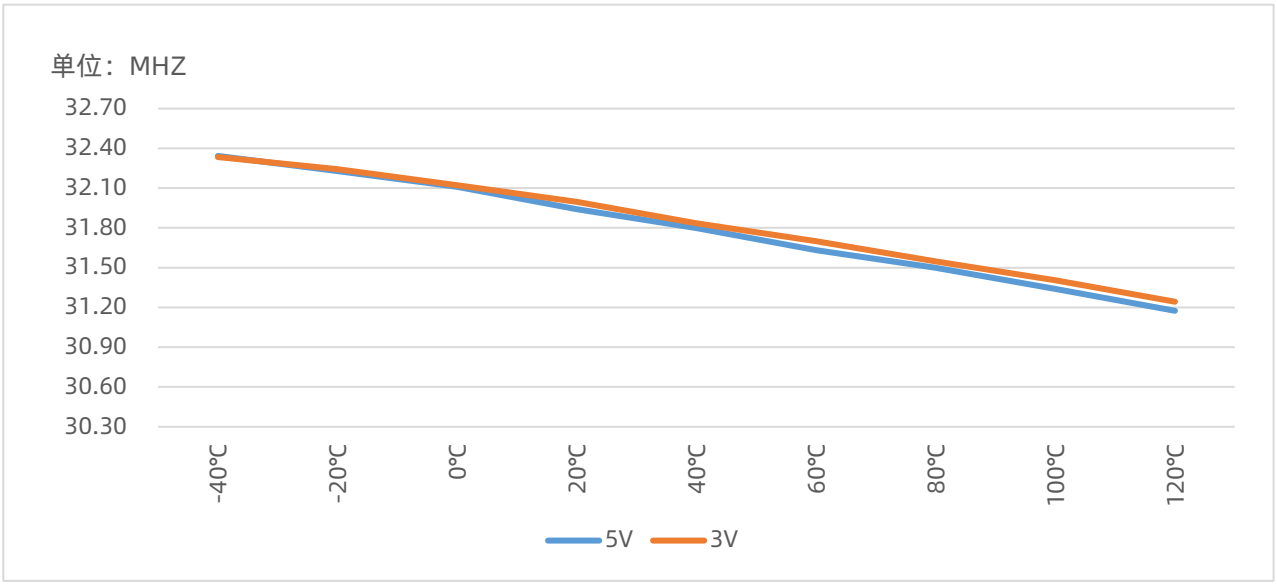
注：具体值不做设计保证。

23.4 系统时钟特性

常温下（25℃），高频时钟（32MHZ）随电压变化典型曲线



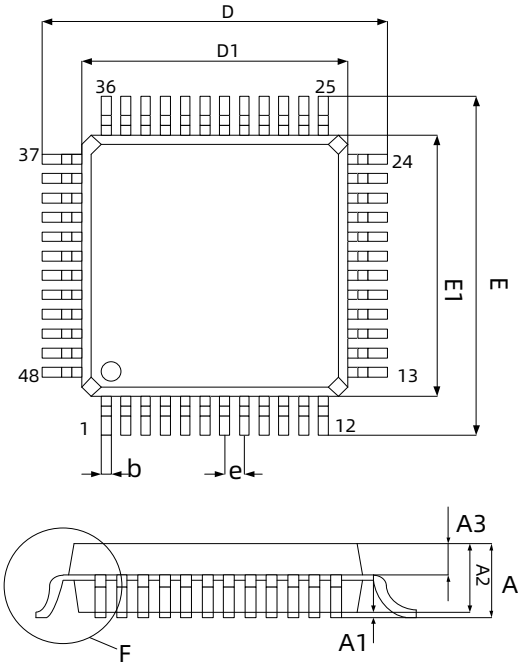
高频时钟（32MHZ）随温度变化典型曲线

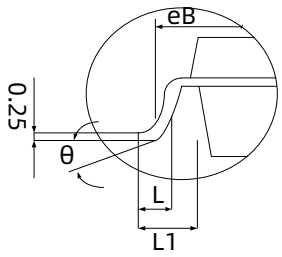


注：具体值不做设计保证 。

24 封装信息

24.1 LQFP48





符号	单位（mm）		
	最小	正常	最大
A	---	---	1.600
A1	0.050	---	0.150
A2	1.350	1.400	1.450
A3	0.600	---	0.640
b	0.180	---	0.260
D	8.800	9.000	9.200
D1	6.900	7.000	7.100
e	0.500BSC		
E	8.800	9.000	9.200
E1	6.900	7.000	7.100
L	0.400	---	0.650
L1	1.000REF		
eB	8.100	---	8.250
θ	0°	---	8°

25 指令集简述

25.1 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。

25.2 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

25.3 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	-	
	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/GIE	
	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。

26 修正记录

版本	日期	描述
V1.00	2025-08-20	初版
V1.01	2025-08-29	勘