乐升半导体
LEVETOP

LT165A

Uart TFT LCD Display Controller

串口屏控制芯片

完整版规格书

V1.3

www.levetop.cn
Levetop Semiconductor Co., Ltd

版本记录

版本	日期	描述
V1.0	2024/11/28	● 初版
V1.1	2025/09/16	● 移除 LT165B 相关信息
V1.2	2025/12/24	● 修改第 21.7.5 节 SCI Baud 率寄存器 ● 修改第 24.4.4 节从地址寄存器 ● 修改表 28-2：IO 电气参数表、表 28-3：电源特性
V1.3	2026/04/16	● 修改表 2-1：信号特性说明 增加引脚号

版权说明

目录

图 附 录

表附录

1. LT165A 介绍

1.1. 基本简介

LT165A 是一系列低成本高效能的 Uart 串口屏控制芯片，其内部采用 32bit RISC 核心架构，主要的功能就是提供 Uart 串口通讯，让主控端 MCU 透过简单的通讯指令就能轻易的将要显示到 TFT 屏的内容传递给 TFT 屏上的驱动器（Driver），LT165A 内部硬件及串口程序提供高速图像处理的功能，能够达到极佳的显示效率及降低主控端 MCU 处理图形显示的时间，LT165A 支持分辨率 320*240 以内的 8 位并口 MCU 屏、或是 SPI 串口的 TFT LCD 屏。

LT165A 内部的主频可达 150MHz，含有 32KB SRAM，除了提供串口通讯，也提供一个 QSPI Flash 接口，用来快速读取储存在外部 SPI Flash 的程序代码、图片、动画等信息，LT165A 可以配合 乐升半导体 研发的串口屏开发软件（UI_Editor）、模拟软件（UI_Emulator），直接在电脑上进行产品的 UI 显示界面开发，其所支持的显示功能包括图片显示、GIF 动画显示、循环图片显示、进度条显示、显示、文字符串显示、PWM (DMA 模式) 音讯播放，及结合触控功能的多变量控制显示，除了提升显示效率外，也大幅缩短 TFT 显示的开发周期。此外，LT165A 还提供另一组的 SCI（Uart）接口可以连接如蓝牙模块或 WiFi 模块，也有包括 CanBus、模拟输入 AIN、PWM 及 INT 中断等接口，还有电容触控输入接口，这些也可设置成普通 IO 接口，同时自带 RTC 时钟，丰富的功能增加了串口屏的实用与适用范围，芯片特性符合车规标准设计与应用。LT165A 可以支持双屏显示、或是并屏显示的应用，具有良好的流畅度与极高的性价比。

在許多小型電子產品上，LT165A 也可以将部分资源作为主控的 MCU，将主控及 TFT 显示功能由一颗 LT165A 来完成，它的显示功能非常适合用在低分辨率 TFT-LCD 屏的电子产品上，如用来取代原单色屏产品，或是增加产品质感、档次，不会造成产品原主控端 MCU 太多的负担，可应用于各式小型電子產品如智能家电、手持控制设备、工业控制板、电子仪器、检测设备、小电摩、个人医美、小型检测设备、充电设备、水电表、带屏智能音箱、机器人眼睛等产品。

LT165A 的工作频率高达 150MHz。温度范围为-40°C 至 105°C，工作电压为 3.3V。LT165A 封装的型号，如下所示：

表 1-1：LT165A 型号

型 号	封 装	SRAM	TFT Panel
LT165A	QFN-40	32KB	● 8-bits 8080 MCU 接口 TFT 屏 ● SPI 接口 TFT 屏

QFN-40 (5.0*5.0 mm²)

图 1-1：LT165A 外观图

1.2. 内部方块图

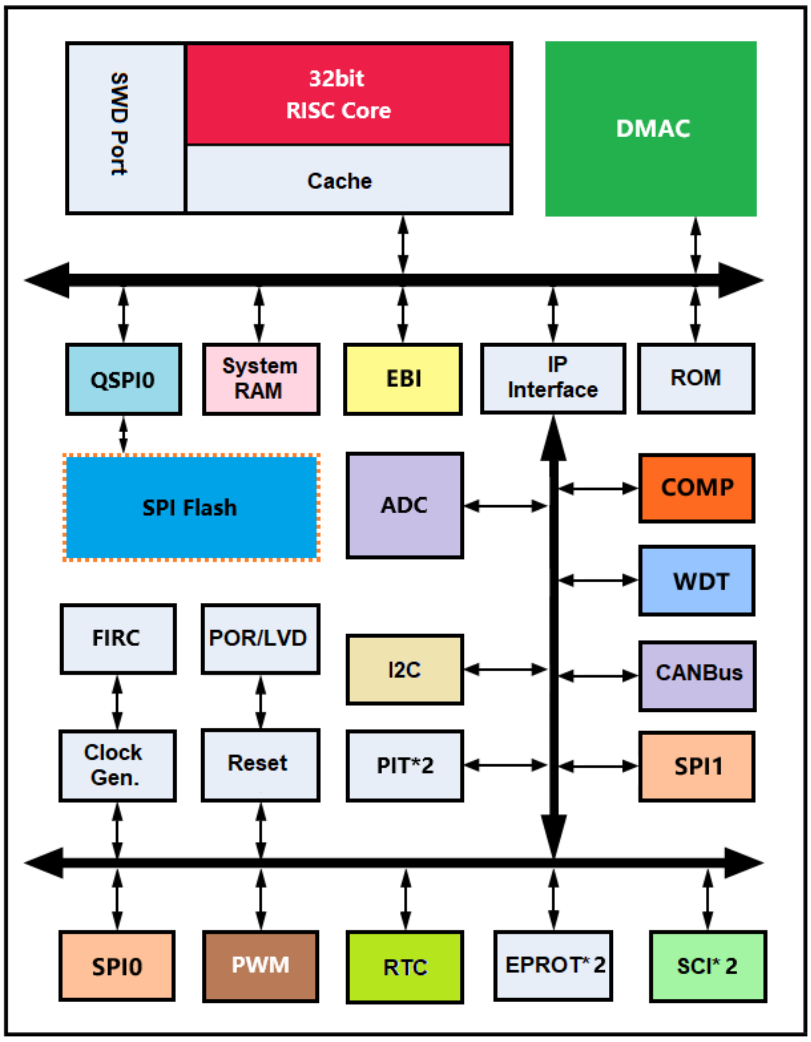


图 1-2：内部方块图

1.3. 功能说明

1.3.1. 32 位 RISC 核心

- 32 位加载/存储精简指令集计算机（RISC）架构，具有固定的 16 位指令长度
- 16 个 32 位通用寄存器文件
- 高效三阶段执行流水线，隐藏于应用软件之外
- 单周期指令执行，多个指令需要三个周期，分支则需要三个周期
- 支持字节/半字/字内存访问
- 嵌入式中断控制器，支持嵌套向量中断。
- 单周期 32 位 x32 位硬件整数乘法器阵列
- 3~13 周期硬件整数分频器阵列

1.3.2. 32KB SRAM

- 单周期字节、半字（16 位）和字（32 位）读写
- 两个用于改进特定应用性能的段
  - 系统 RAM0：16KB ， 地址范围从 0x0080_0000 到 0x0080_3FFF
  - 系统 RAM1：16KB ， 地址范围从 0x0080_4000 到 0x0080_7FFF

1.3.3. 6KB ROM

- 单周期字节、半字（16 位）和字（32 位）读取访问

1.3.4. 2KB 缓存

- 双向集合 — 关联式组织
- 两个 AHB 总线接口，一个主接口和一个从接口

1.3.5. 外部总线接口（EBI）

- 可编程等待状态-在访问终止前最多可编程设置 256 个等待状态
- 一个可编程异步低电平有效芯片选择器。
- 可编程芯片选择等待周期
- 为了与各种面板连接，最多可以对 256 个芯片选择施加周期进行编程。
- 可编程读/写断言周期
- 可编程读/写否定循环
- 支持 8 位端口大小。

- 支持 8040 标准总线

1.3.6. DMA 模块

- 16 个可编程通道，支持独立的 8、16 或 32 位单值或块传输
- 支持可变大小队列和循环队列
- 源地址寄存器和目标地址寄存器独立配置，以保持增量器恒定
- 由外围设备、CPU、周期性计时器中断或 DMA 通道请求发起的每次传输
- 可能来自 QSPI、QADC 的外围 DMA 请求源
- 每个 DMA 通道在完成单值或块传输后，可选择性地向 CPU 发送中断请求
- DMA 传输可在系统存储器和所有可访问的内存映射位置之间进行，包括外围设备和寄存器
- DMA 支持以下功能：
  - 散射聚集
  - 通道链接
  - 内部循环偏移量
  - 仲裁
  - 固定组，固定频道
  - Round Robin 组，固定信道
  - 轮询组，轮询通道
  - 固定组，轮询通道
  - 信道优先权
  - 取消通道传输
- 中断-DMA 为每个已实现的通道提供一个中断请求，并且有一个组合 DMA 错误中断，用于向系统标记传输错误

1.3.7. 重置模块

- 复位电路的内部电源
- 重置的五个来源：
  - 通电重置
  - 外部引脚
  - 软件重置
  - 看门狗定时器
  - 程序电压检测复位
- 状态标志指示上次重置的来源

1.3.8. 可编程中断计时器（PIT）

- 16 位计数器，其模数为“初始计数”寄存器
- 可选择为自由运行或倒计时
- 16 个可选预分档器 — 20 至 215
- 支持 DMA 接口

1.3.9. 看门狗定时器（WDT）

- 16 位计数器，其模数为“初始计数”寄存器
- 低功耗模式下的暂停选项
- 最长服务时间 2000ms

1.3.10. 实时时钟（RTC）

- 支持秒、分钟、小时和天计数器中的加载时间数据和读取时间数据
- 支持闹钟设置
- 中断源：
  - 秒、分钟、小时、日中断
  - 可编程闹钟中断
  - 1KHz/32KHz 周期性中断

1.3.11. EPORT 模块

- 每个 EPORT 有 8 个通道
- 上升/下降边沿选择
- 低/高灵敏度
- 中断引脚可配置为通用 I/O

1.3.12. SPI 模块

- 主模式和从属模式
- 有线或 OR 模式
- 选择性输出
- 模式故障错误标志，具有中央处理器（CPU）中断功能
- 双缓冲操作
- 具有可编程极性与相位的串行时钟

- 在睡眠模式期间控制 SPI 操作
- 降低驱动控制以降低功耗

1.3.13. SSI / QSPI 模块

- 串行主操作
- DMA 控制器接口
- 使 SSI 能够通过总线使用用于传输请求的握手接口与 DMA 控制器进行交互。
- 增强的 SPI 传输中的时钟伸缩支持
- 数据项大小（4 至 32 位） — 每个数据传输项的大小，由程序员控制
- 发送和接收 FIFO 缓冲器的深度可配置为 2 到 256 个字深。FIFO 宽度固定为 32 位
- 增强的 SPI 支持
- 执行就地（XIP）模式支持

1.3.14. SCI / UART 模块

- 全双工，标准非归零（NRZ）格式
- 可编程波特率（13 位模数分频器），可配置的过采样比为 4x 至 256x
- 中断，轮询操作：
  - 发送数据寄存器为空，传输完成
  - 接收数据寄存器已满
  - 接收溢出、奇偶校验错误、帧错误和噪声错误
  - 怠速转速检测器
  - 接收引脚上的有效边沿
  - 断路器检测支持的 LIN
  - 接收数据匹配
- 硬件奇偶校验生成和检查
- 可编程的 8 位、9 位或 10 位字符长度
- 可编程的 1 比特或 2 比特停止位
- 三种接收机唤醒方式：
  - 空闲线路唤醒
  - 标记唤醒地址
  - 接收数据匹配
- 自动地址匹配以减少 ISR 开销：
  - 地址标记匹配

- 空闲行地址匹配
- 地址匹配开始，地址匹配结束

- 可选的 13 位中断字符生成/11 位中断字符检测
- 支持 1、2、4、8、16、32、64 或 128 个空闲字符的可配置空闲长度检测
- 可选择的发射机输出和接收机输入极性
- 可选择的 IrDA 1.4 返回零反转（RZI）格式，具有可编程脉冲宽度
- 发送和接收的独立 FIFO 结构
  - 接收和发送请求可配置独立的水印
  - 如果接收 FIFO 为空，则接收器可选择在配置的空闲字符数之后断言请求

1.3.15. CAN 总线控制器

- 全面实施 CAN 协议规范，版本 2.0B
  - 标准数据和远程帧
  - 扩展数据和远程帧
  - 0 ~ 8 字节数据长度
  - 可编程比特率，最高可达 1 Mbit/s
  - 内容相关寻址
- 16 个数据长度为 0 到 8 字节的信息缓冲区
- 每个 MB 可配置为 Rx 或 Tx，均支持标准和扩展信息
- 每个信息缓冲区中的单个 Rx 掩码寄存器
- 包括用于 MB 存储的 288 字节(16 MB）SRAM
- 包括用于各个 Rx 掩码寄存器的 64 字节(16 MB）SRAM
- 功能齐全的 Rx FIFO，具有存储 6 帧容量和内部指针处理
- 强大的 Rx FIFO ID 过滤，能够将输入的 ID 与 8 个扩展、16 个标准或 32 个部分（8 位）ID 进行匹配，并具有单独掩码功能
- 可编程的 CAN 协议接口时钟源，总线时钟或晶体振荡器
- 未使用的 MB 和 Rx 掩码寄存器空间可用作通用 SRAM 空间
- 仅支持收听模式的功能
- 支持自检操作的可编程回环模式
- 可编程传输优先级方案：最低 ID、最低缓冲区号或最高优先级
- 基于 16 位自由运行计时器的时间戳
- 由特定信息同步的全局网络时间
- 掩蔽中断

- 与传输介质无关（假设为外部收发器）
- 由于采用了高优先级信息的仲裁方案，因而缩短了延迟时间
- 低功率模式
- 发送信息缓冲区的硬件取消

1.3.16. PWM 模块

- 四个通道的 PWM 控制器
- 可编程周期
- 可编程工作周期
- 2 台 Dead-Zone 发生器
- 捕获功能
- 引脚可配置为通用 I/O

1.3.17. ADC 模块

- 高性能
  - 12 位、10 位、8 位或 6 位可配置分辨率
  - ADC 转换时间：12 位分辨率（1 MHz）为 1.0 μs，10 位分辨率的转换时间为 0.88 μs，通过降低分辨率可以获得更快的转换时间。
  - 可编程采样时间
  - 数据与内置数据一致性进行对齐
  - DMA 支持
- 低功率
  - 应用程序可降低 PLCK 频率以进行低功耗操作，同时仍保持最佳 ADC 性能。例如，无论 PCLK 的频率如何，都保持 1.0 μs 转换时间。
  - 等待模式：在低频 PLCK 的应用中防止 ADC 溢出
  - 自动关闭模式：除在活动转换阶段外，ADC 会自动关闭。这大大降低了 ADC 的功耗。
- 模拟输入通道
  - 3 个外部模拟输入
  - 1 个通道用于内部温度传感器
- 可启动转换开始：
  - 软件
  - 通过具有可配置极性的硬件触发器
- 转换模式

- 可转换为单通道模式或扫描一系列通道。单模式下，每次触发时仅转换选定输入一次。
- 连续模式可连续选择输入
- 间断模式

- 在采样结束、转换结束、序列转换结束以及模拟看门狗或超频事件发生时中断生成。
- 模拟看门狗
- 单端和差分输入配置
  - 转换器使用内部参考或外部参考

1.3.18. I2C 模块

- 支持 7 位寻址。
- 支持标准模式、快速模式和高速模式
- 软件选项，可选择高速模式和标准/快速模式
- 与 I2C 总线 2.1 版标准的正常模式和快速模式兼容。
- 多主操作。
- 软件可编程，支持 64 种不同的串行时钟频率。
- 软件可选的确认位。
- 中断驱动的逐字节数据传输。
- 仲裁中断，自动切换主设备为从设备。
- 传输完成和读取配置中断。
- 启动和停止信号生成/检测。
- 重复的 START 信号生成。LT32A05_SPEC_ENG/V0.0
- 确认二进制位生成/检测。
- 总线繁忙检测。
- 系统时钟停止模式下，可选择从属地址接收使能
- 支持 SCL 或 SDA 线路的 gpio 功能

1.3.19. 模拟比较器

- 可编程响应时间
- 可编程滞回
- 支持模拟输入多路复用器，具有 9 个选择
- 两种可选输出：过滤或异步输出
- 可选择的上升/下降沿中断

1.3.20. 触摸传感器

- 支持四个触摸键
- 支持三种时钟模式，具有充电或放电功能
  - 频率范围为 369KHz 至 6MHz，具有固定分频器
  - PRS 1.5MHz 的频率符合正态分布
  - PRS 1.5MHz 的频率分布均匀
- 可编程计数器时钟频率为 24/12/6/4MHz
- 可编程计数器的宽度范围为 9 至 16 位
- 支持同步扫描模式

1.3.21. 电源管理单元（PMU）

- 支持芯片上 1.2V LDO，最大负载电流为 150mA
- 1.2V LDO 支持两种模式：低功耗、高功耗

1.3.22. 电压探测器

- 可编程电压检测器

1.3.23. 内部振荡器

- 用于看门狗和 PMU 的 128KHz 片上振荡器时钟
- 快速内部 RC 时钟，可用于系统时钟

1.3.24. 外部晶体振荡器

- 32.768KHz 外部晶体振荡器时钟，可用于 RTC
- 可作为系统时钟使用的快速外部晶体振荡器

1.4. 系统应用方块图

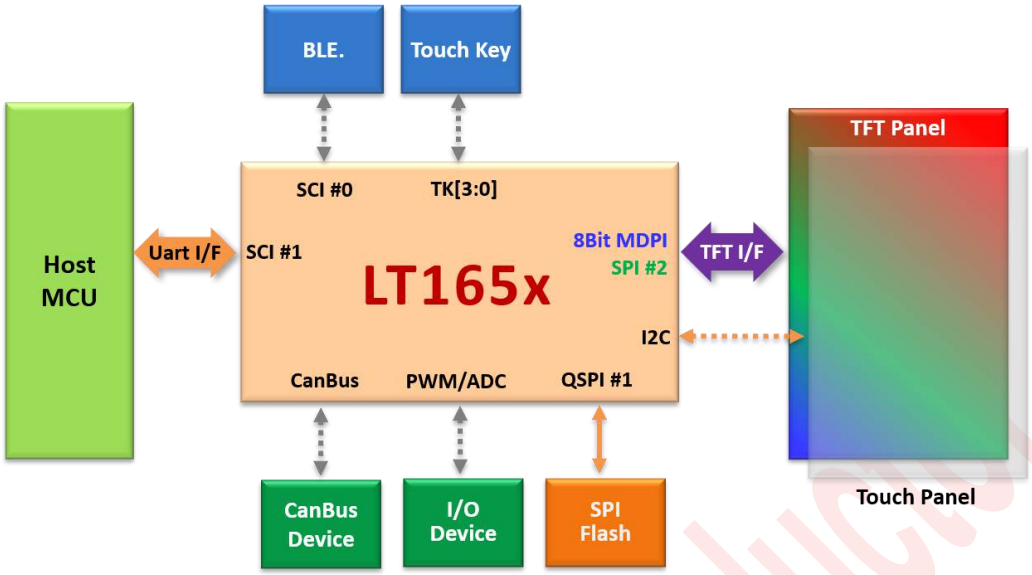


图 1-3：LT165A 系统应用方块图

1.5. 内存与寄存器配置

1.5.1. 简要说明

LT165A 的内建/外部内存、寄存器包括：

- 最高 128M Bytes 的外部 QSPI Flash 闪存空间
- 8K Bytes 内部 Boot ROM
- 32K Bytes 内部静态 SRAM
- 各模块内部寄存器

1.5.2. 内存地址映射

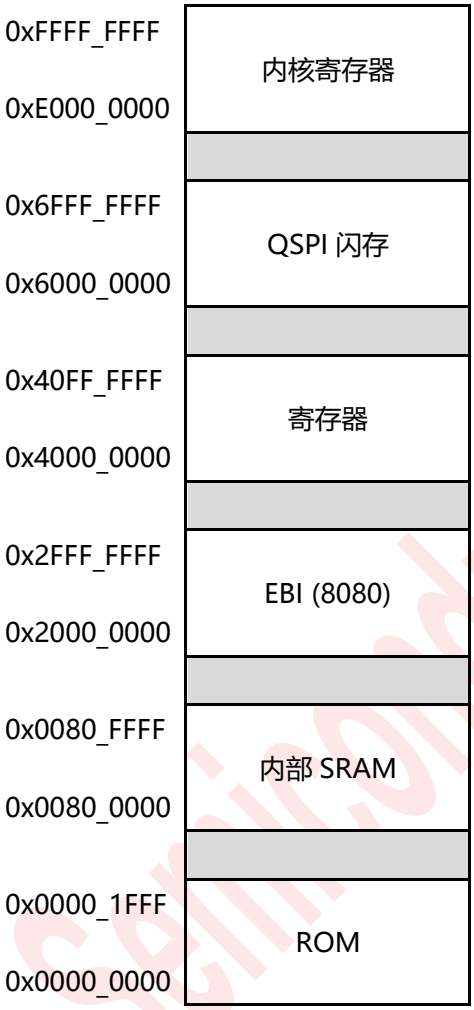


图 1-4：内存地址配置图

表 1-2：硬件模块和寄存器地址位置图

配置地址	最大区块	硬 件 模 块	章次
0x4000_0000	64KB	直接内存访问控制器（DMAC）	13
0x4001_0000	64KB	系统集成模块（SIM）	7
0x4002_0000	64KB	重置控制模块（RCM）	9
0x4003_0000	64KB	时钟控制模块（CLKM）	8
0x4004_0000	64KB	可编程中断计时器 0（PIT0）	16
0x4005_0000	64KB	可编程中断计时器 1（PIT1）	16
0x4006_0000	64KB	保留	-
0x4007_0000	64KB	保留	-
0x4008_0000	64KB	串行通信接口 1（SCI1）	21

配置地址	最大区块	硬 件 模 块	章次
0x4009_0000	64KB	串行通信接口 0（SCI0）	21
0x400A_0000	64KB	模拟比较器 0（COMP0）	26
0x400B_0000	64KB	保留	-
0x400C_0000	64KB	保留	-
0x400D_0000	64KB	脉冲宽度调制器 0（PWM0）	25
0x400E_0000	64KB	保留	-
0x400F_0000	64KB	边缘端口模块 0（EPORT0）	19
0x4010_0000	64KB	边缘端口模块 1（EPORT1）	19
0x4011_0000	64KB	模数转换器（ADC）	27
0x4012_0000	64KB	选项字节（OPB）	14
0x4013_0000	64KB	WatchDog 定时器（WDT）	17
0x4014_0000	64KB	实时控制器（RTC）	18
0x4015_0000	64KB	集成电路间（I2C）	24
0x4016_0000	64KB	触摸控制器	
0x4017_0000	64KB	横条开关（XBAR）	12
0x4018_0000	64KB	外部总线接口（EBI）	15
0x4019_0000	64KB	缓存模块（CACHEM）	11
0x401A_0000	64KB	保留	-
0x401B_0000	64KB	保留	-
0x401C_0000	64KB	CANBus 控制器（CANBC）	20
0x401D_0000	64KB	保留	-
0x401E_0000	64KB	串行外围接口模块- SPI0	23
0x401F_0000	64KB	串行外围接口模块- SPI1	23
0x6000_0000	64KB	同步串行接口 0（SSI0）- QSPI0	22
0xE000_0000	4KB	嵌入式中断控制器（EIC）	4
0xE000_1000	4KB	嵌入式可编程定时器（EPT）	6

2. 引脚信号描述

2.1. 芯片脚位图

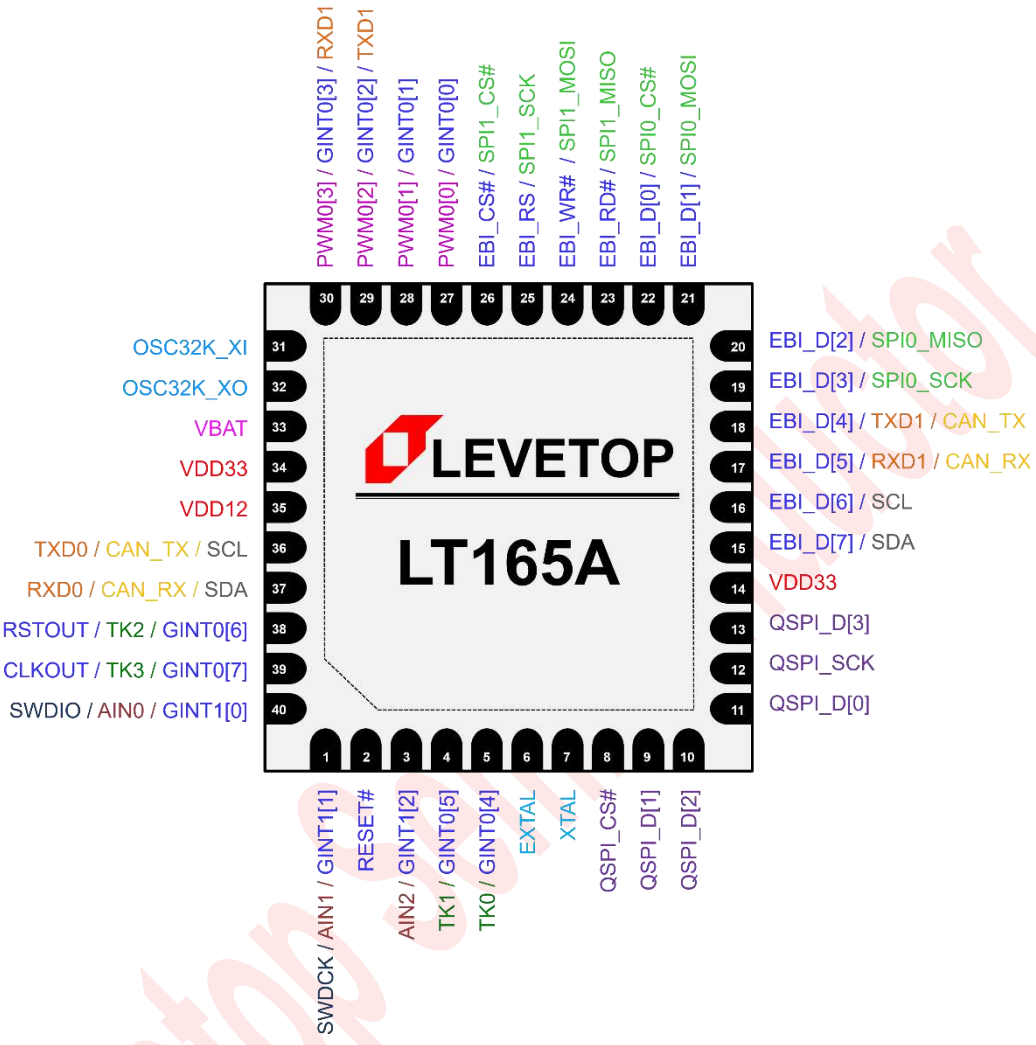


图 2-1：LT165A（QFN-40）脚位图

2.2. 信号特性总结

表 2-1：信号特性说明

引脚名称	复用信号 1	复用信号 2	GPIO	引脚号	Default State	Default Dir *1	Pullup *2	IO Type *3
SCI (2)								
TXD0	SCL	CAN_TX	GPIO29	36	HiZ	--	--	I/O
RXD0	SDA	CAN_RX	GPIO30	37	HiZ	--	--	I/O
QSPI (6)								
QSPI_CS#	--	--	GPIO19	8	HiZ	--	--	I/O
QSPI_D[0]	--	--	GPIO20	11	HiZ	--	--	I/O
QSPI_D[1]	--	--	GPIO21	9	HiZ	--	--	I/O
QSPI_D[2]	--	--	GPIO22	10	HiZ	--	--	I/O
QSPI_D[3]	--	--	GPIO23	13	HiZ	--	--	I/O
QSPI_SCK	--	--	GPIO24	12	HiZ	--	--	I/O
PWM0 (4)								
PWM0[0]	GINT0[0]	COMP0_OUT	GPIO0	27	HiZ	--	--	I/O
PWM0[1]	GINT0[1]	--	GPIO1	28	HiZ	--	--	I/O
PWM0[2]	GINT0[2]	TXD1	GPIO2	29	HiZ	--	--	I/O
PWM0[3]	GINT0[3]	RXD1	GPIO3	30	HiZ	--	--	I/O
ADC (1)								
AIN2	GINT1[2]	--	GPIO18	3	HiZ	--	--	I/O
Touch (2)								
TK0	GINT0[4]	--	GPIO4	5	HiZ	--	--	I/O
TK1	GINT0[5]	--	GPIO5	4	HiZ	--	--	I/O
Programming Port (2)								
SWDIO	GINT1[0]	AIN0	GPIO16	40	I	I	PullUp	I/O
SWDCK	GINT1[1]	AIN1	GPIO17	1	I	I	PullUp	I/O
CLOCK (5)								
EXTAL	--	--	--	6	HiZ/I	--/I	--	I
XTAL	--	--	--	7	HiZ/O	--/O	--	O
OSC32K_XI	--	--	--	31	I	I	-	I
OSC32K_XO	--	--	--	32	O	O	--	O
CLKOUT	GINT0[7]	TK3	GPIO7	39	O	O	--	I/O
RESET (2)								
RESET#	--	--	--	2	I	I	PullUp	I
RSTOUT	GINT0[6]	TK2	GPIO6	38	O	O	--	I/O
EBI (12)								

引脚名称	复用信号 1	复用信号 2	GPIO	引脚号	Default State	Default Dir *1	Pullup *2	IO Type *3
EBI_CS#	SPI1_CS#	--	GPIO25	26	HiZ	--	--	I/O
EBI_RS	SPI1_SCK	--	GPIO28	25	HiZ	--	--	I/O
EBI_RD#	SPI1_MISO	--	GPIO27	23	HiZ	--	--	I/O
EBI_WR#	SPI1_MOSI	--	GPIO26	24	HiZ	--	--	I/O
EBI_D[0]	SPI0_CS#	--	GPIO8	22	HiZ	--	--	I/O
EBI_D[1]	SPI0_MOSI	--	GPIO9	21	HiZ	--	--	I/O
EBI_D[2]	SPI0_MISO	--	GPIO10	20	HiZ	--	--	I/O
EBI_D[3]	SPI0_SCK	--	GPIO11	19	HiZ	--	--	I/O
EBI_D[4]	TXD1	CAN_TX	GPIO12	18	HiZ	--	--	I/O
EBI_D[5]	RXD1	CAN_RX	GPIO13	17	HiZ	--	--	I/O
EBI_D[6]	SCL	--	GPIO14	16	HiZ	--	--	I/O
EBI_D[7]	SDA	--	GPIO15	15	HiZ	--	--	I/O
Power Supply (5)								
VDD33	--	--	--	14, 34	P	--	--	--
VDD12	--	--	--	35	P	--	--	--
AVDD	--	--	--	--	P	--	--	--
VBAT	--	--	--	33	P	--	--	--
VSS	--	--	--	41(*)	G	--	--	--

提示：

1. “Defaul Dir”是指复位后的方向。“I”代表输入，“O”代表输出，“O（H）”代表输出高、“O（L）”表示输出低、“Hiz”表示输入和输出均已禁用，上拉/下拉也已禁用。
2. 当信号被设置为输出时，所有上拉和下拉都断开。
3. “IO TYPE”指的是引脚设计：“I”代表仅具有输入功能的引脚；“O”代表仅输出功能引脚；“I/O”代表引脚具有输入和输出的功能。

2.3. 信号描述

本章节提供引脚信号的简要说明，有关更多详细信息，请参阅特定模块部分。

表 2-2：引脚信号说明述

信号名称	引脚号	引脚说明
串口模块信号 0 (Serial Communications Interface – 0，SCI0)		
RXD0	37	**SCI 接收数据** 此信号用于 SCI0 接收器数据输入，当未配置为接收器操作时，也可复用于 SDA、CAN_RX 或是 GPIO30 信号。关于复用信号请参考表 2-1。
TXD0	36	**SCI 传输数据** 此信号用于 SCI0 发送器数据输出，当未配置为发送器操作时，也可复用于 SCL、CAN_RX 或是 GPIO29 信号。
串口模块信号 1 (SCI1)		
RXD1	17 / 30	**SCI 接收数据** 此信号用于 SCI1 接收器数据输入。 此信号有 2 个复用源，一个是与 EBI_D[5]、CAN_RX 或是 GPIO13 信号复用，另一个是与 PWM0[3]、GINT0[3]或是 GPIO3 信号复用。
TXD1	18 / 29	**SCI 传输数据** 此信号用于 SCI1 发送器数据输出。 此信号有 2 个复用源，一个是与 EBI_D[4]、CAN_TX 或是 GPIO12 信号复用，另一个是与 PWM0[2]、GINT0[2]或是 GPIO2 信号复用。
CAN Bus 模块信号		
CAN_RX	17 / 37	**Can Bus 接收数据** 此信号是来自 CANBus 收发器的接收引脚。显性状态由逻辑电平“0”表示。隐性状态由逻辑电平“1”表示。 此信号有 2 个复用源，当未配置为 CANBUS 操作时，一个是与 EBI_D[5]、RXD1 或是 GPIO13 信号复用，另一个是与 RXD0、SDA 或是 GPIO30 信号复用。

信号名称	引脚号	引脚说明
CAN_TX	18 / 36	**Can Bus 传输数据** 此信号是 CANBus 收发器的发送引脚。显性状态由逻辑电平“0”表示。隐性状态由逻辑电平“1”表示。 此信号有 2 个复用源，当未配置为 CANBUS 操作时，一个是与 EBI_D[4]、TXD1 或是 GPIO12 信号复用，另一个是与 TXD0、SCL 或是 GPIO29 信号复用。
外部总线（External Bus Interface，EBI）信号 外部总线接口负责控制内部总线和外部 8 位并口 MCU 显示屏间的信息传输。只有 LT165A 提供 EBI 接口支持 8 位 MCU 显示屏。		
EBI_CS#	26	**8 位 MCU 屏的片选信号** LT165A 提供 EBI 接口用来驱动 8080 MCU 接口的 TFT LCD 显示面板，此信号是外部总线接口的片选信号。 此信号与 SPI1_CS#或是 GPIO25 信号复用。
EBI_RS	25	**8 位 MCU 屏的寄存器选择信号** 此信号连接到 8Bit MCU 屏 RS 或 A0 信号。 此信号与 SPI1_SCK 或是 GPIO28 信号复用。
EBI_RD#	23	**8 位 MCU 屏的数据读取控制信号** 此信号为 LT165A 对外部 8Bit MCU 屏的数据读取控制信号。 此信号与 SPI1_MISO 或是 GPIO27 信号复用。
EBI_WR#	24	**8 位 MCU 屏的数据写入控制信号** 此信号为 LT165A 对外部 8Bit MCU 屏的数据写入控制信号。 此信号与 SPI1_MOSI 或是 GPIO26 信号复用。
EBI_D[0]	22	**8 位 MCU 屏的数据信号 0** 此信号为 LT165A 对外部 8Bit MCU 屏的数据 Bit0 传输信号。此信号与 SPI0_CS#或是 GPIO8 信号复用。
EBI_D[1]	21	**8 位 MCU 屏的数据信号 1** 此信号为 LT165A 对外部 8Bit MCU 屏的数据 Bit1 传输信号。此信号与 SPI0_MOSI 或是 GPIO9 信号复用。
EBI_D[2]	20	**8 位 MCU 屏的数据信号 2** 此信号为 LT165A 对外部 8Bit MCU 屏的数据 Bit2 传输信号。此信号与 SPI0_MISO 或是 GPIO10 信号复用。

信号名称	引脚号	引脚说明
EBI_D[3]	19	**8 位 MCU 屏的数据信号 3** 此信号为 LT165A 对外部 8Bit MCU 屏的数据 Bit3 传输信号。此信号与 SPI0_SCK 或是 GPIO11 信号复用。
EBI_D[4]	18	**8 位 MCU 屏的数据信号 4** 此信号为 LT165A 对外部 8Bit MCU 屏的数据 Bit4 传输信号。此信号与 TXD1、CAN_TX 或是 GPIO12 信号复用。
EBI_D[5]	17	**8 位 MCU 屏的数据信号 5** 此信号为 LT165A 对外部 8Bit MCU 屏的数据 Bit5 传输信号。此信号与 RXD1、CAN_RX 或是 GPIO13 信号复用。
EBI_D[6]	16	**8 位 MCU 屏的数据信号 6** 此信号为 LT165A 对外部 8Bit MCU 屏的数据 Bit6 传输信号。此信号与 SCL 或是 GPIO14 信号复用。
EBI_D[7]	15	**8 位 MCU 屏的数据信号 7** 此信号为 LT165A 对外部 8Bit MCU 屏的数据 Bit7 传输信号。此信号与 SDA 或是 GPIO15 信号复用。
I2C 控制信号		
SCL	16 / 36	**I2C 时钟** 此信号用于 I2C 时钟线信号。 此信号有 2 个复用源，当未配置为 I2C 操作时，一个是与 EBI_D[6] 或是 GPIO14 信号复用，另一个是与 TXD0、CAN_TX 或是 GPIO29 信号复用。
SDA	15 / 37	**I2C 数据** 此信号用于 I2C 数据线信号。 此信号有 2 个复用源，当未配置为 I2C 操作时，一个是与 EBI_D[7] 或是 GPIO15 信号复用，另一个是与 RXD0、CAN_RX 或是 GPIO30 信号复用。
四线 SPI 串口模块信号 (QSPI)		
QSPI_D[3:0]	13, 10, 9 11	**QSPI 数据输入/输出** 这些信号是 QSPI 在主模式下的数据输出或输入。
QSPI_CS#	8	**QSPI 片选输出** 此信号是 QSPI 在主模式和低电平有效模式下的片选信号。

信号名称	引脚号	引脚说明
QSPI_SCK	12	**QSPI 时钟输出** 此信号是 QSPI 在主模式下的串行时钟输出。
双线 SPI 串口模块信号 (SPI)		
SPI0_MOSI	21	**SPI #0 的数据输出信号** 此信号为第 1 组 SPI 输出数据。 此信号与 EBI_D[1] 或是 GPIO9 信号复用。
SPI0_MISO	20	**SPI #0 数据输入信号** 此信号为第 1 组 SPI 的读取数据输入。 此信号与 EBI_D[2] 或是 GPIO10 信号复用。
SPI0_CS#	22	**SPI #0 芯片选择信号** 此信号为第 1 组 SPI 的片选输出。 此信号与 EBI_D[0] 或是 GPIO8 信号复用。
SPI0_SCK	19	**SPI #0 串行时钟信号** 此信号为第 1 组 SPI 的时钟信号输出。 此信号与 EBI_D[3] 或是 GPIO11 信号复用。
SPI1_MOSI	24	**SPI #1 的数据输出信号** 此信号为第 2 组 SPI 输出数据。 此信号与 EBI_WR#或是 GPIO26 信号复用。
SPI1_MISO	23	**SPI #1 数据输入信号** 此信号为第 2 组 SPI 的读取数据输入。 此信号与 EBI_RD#或是 GPIO27 信号复用。
SPI1_CS#	26	**SPI #1 芯片选择信号** 此信号为第 2 组 SPI 的片选输出。 此信号与 EBI_CS#或是 GPIO25 信号复用。
SPI1_SCK	25	**SPI #1 串行时钟信号** 此信号为第 2 组 SPI 的时钟信号输出。 此信号与 EBI_RS 或是 GPIO28 信号复用。
Touch Key 信号		
TK0	5	**Touch Key 0 输入** 此信号也可复用于 GINT0[4] 或是 GPIO4。

信号名称	引脚号	引脚说明
TK1	4	**Touch Key 1 输入** 此信号也可复用于 GINT0[5] 或是 GPIO5。
TK2	38	**Touch Key 2 输入** 此信号也可复用于 RSTOUT、GINT0[6] 或是 GPIO6。
TK3	39	**Touch Key 3 输入** 此信号也可复用于 CLKOUT、GINT0[7] 或是 GPIO7。
EPORT 模块 0 信号		
GINT0[0]	27	**中断输入 / GPIO 输出入** 这些双向信号可用作外部中断源或 GPIO 使用。 此信号与 PWM0[0]、COMP0_OUT 或是 GPIO0 信号复用。
GINT0[1]	28	**中断输入 / GPIO 输出入** 这些双向信号可用作外部中断源或 GPIO 使用。 此信号与 PWM0[1] 或是 GPIO1 信号复用。
GINT0[2]	29	**中断输入 / GPIO 输出入** 这些双向信号可用作外部中断源或 GPIO 使用。 此信号与 PWM0[2]、TXD1 或是 GPIO2 信号复用。
GINT0[3]	30	**中断输入 / GPIO 输出入** 这些双向信号可用作外部中断源或 GPIO 使用。 此信号与 PWM0[3]、RXD1 或是 GPIO3 信号复用。
GINT0[4]	5	**中断输入 / GPIO 输出入** 这些双向信号可用作外部中断源或 GPIO 使用。 此信号与 TK0 或是 GPIO4 信号复用。
GINT0[5]	4	**中断输入 / GPIO 输出入** 这些双向信号可用作外部中断源或 GPIO 使用。 此信号与 TK1 或是 GPIO5 信号复用。
GINT0[6]	38	**中断输入 / GPIO 输出入** 这些双向信号可用作外部中断源或 GPIO 使用。 此信号与 RSTOUT、TK2 或是 GPIO6 信号复用。

信号名称	引脚号	引脚说明
GINT0[7]	39	**中断输入 / GPIO 输出入** 这些双向信号可用作外部中断源或 GPIO 使用。 此信号与 CLKOUT、TK3 或是 GPIO7 信号复用。
EPORT 模块 1 信号		
GINT1[0]	40	**中断输入 / GPIO 输出入** 这些双向信号可用作外部中断源或 GPIO 使用。 此信号与 SWDIO、AIN0 或是 GPIO16 信号复用。
GINT1[1]	1	**中断输入 / GPIO 输出入** 这些双向信号可用作外部中断源或 GPIO 使用。 此信号与 SWDCK、AIN1 或是 GPIO17 信号复用。
GINT1[2]	3	**中断输入 / GPIO 输出入** 这些双向信号可用作外部中断源或 GPIO 使用。 此信号与 AIN2 或是 GPIO18 信号复用。
PWM 模块 0 信号		
PWM0[3:0]	30, 29, 28, 27	**PWM 输出** 这些输出信号可用作 PWM0 输出、GINT0[3:0] 或 GPIO[3:0] 使用。
ADC 模拟信号输入		
AIN2	3	**模拟信号输入** 此模拟信号用作 ADC 模拟输入通道。 当未配置为模拟输入时，此信号也可用于 GINT1[2] 或是 GPIO18。
AIN1	1	**模拟信号输入** 此模拟信号用作 ADC 模拟输入通道。 此信号与 SWDCK、GINT1[1] 或是 GPIO17 复用。
AIN0	40	**模拟信号输入** 此模拟信号用作 ADC 模拟输入通道。 此信号与 SWDIO、GINT1[0] 或是 GPIO16 复用。
程序烧录信号		
SWDCK	1	**程序烧录时钟** 此输入信号是用于对内部闪存进行程序烧录时的时钟信号。 此信号也可复用于 GINT1[1]、AIN1 或是 GPIO17。

信号名称	引脚号	引脚说明
SWDIO	40	**程序烧录数据** 此信号是用于对内部闪存进行程序烧录时的数据信号。 此信号也可复用于 GINT1[0]、AIN0 或是 GPIO16。
时钟信号		
EXTAL	6	**系统时钟信号源** 此引脚连接至外部 12Mhz 晶振
XTAL	7	**系统时钟信号源** 此引脚连接至外部 12Mhz 晶振
OSC32K_XI	31	**RTC 晶振输入** 此引脚连接至外部 32.768Khz 晶振。
OSC32K_XO	32	**RTC 晶振输出** 此引脚连接至外部 32.768Khz 晶振。
CLKOUT	39	**系统时钟信号输出** 此输出信号反映内部系统时钟。 当未配置为时钟输出时，此信号也可可复用于 GINT0[7]、TK3 或是 GPIO7。
复位输入信号		
RESET#	2	**复位输入信号** 当 RST# = 0 时，将对内部 MCU 产生复位动作，除了少数由 POR 才能复位的寄存器外，大多数由 MCU 控制的寄存器将回复到默认值。
RSTOUT	38	**复位输出信号** 此输出信号指示内部复位控制器已在对芯片进行复位。 0 = 芯片处于复位状态 1 = 芯片未复位状态 当未配置为复位输出时，此信号也可复用于 GINT0[6]、TK2 或是 GPIO6。
电源与接地信号 这些信号为芯片提供系统电源和接地。需确保芯片可以获得足够的电流能力。所有电源信号必须具有足够的旁路电容，以抑制高频噪声。		
VDD33	14, 34	**3.3V 电源输入** 供给 I/O 与 LDO 的电源

信号名称	引脚号	引脚说明
VDD12	35	**1.2V LDO 电源输出（Core）** 供内核数字电路的电源，此引脚必须外接一个 1uF 和一个 0.1uF 滤波电容到地。
AVDD	--	**3.3V ADC 等模拟电路的电源输入**
VBAT	33	**RTC 电源输入** **独立的 RTC 电源（电池）输入**
VSS	41(*)	**GND 接地**

提示：这是散热焊盘（Thermal Pad Zone）必须接到 VSS 或是 GND 。在做 PCB 布局时需要特别注意焊盘的焊接面设计，详细请参考第 30.2 节的说明。

2.4. LT165A 资源表

表 2-2：LT165A 资源表

功 能		LT165A
项 目	说 明	
TFT LCD 屏	8bit 8080 接口	√
	SPI 接口	√
MCU 内核、内存	内核	32-bits RISC
	速度	150MHz
	SRAM 容量	32KB
其他模块接口	Uart 接口	√ (x2)
	SPI 接口	√ (x2)
	QSPI 接口	√ (x1)
	PWM 输出	√ (x4)
	Can Bus	√ (x1)
	ADC 输入	√ (x3)
	Touch Key 接口	√ (x4)
	CTP I2C 接口	√
	GPIO 接口	√ (x11)
	RTC 时钟	√
应用与升级	UI_Editor-II	√
	UI_Emulator-II	√
	Uart 口升级	√
	二次开发	√
电源与封装	电源	3.3V
	封装	QFN-40

3. 硬件接口

3.1. 主控端 MCU 通讯接口

LT165A 和主控 MCU 的通讯模式是透过 UART 接口，两边的 UART 接口 TX 及 RX 必须交叉对接，参考下图所示。如果连接的距离很长还需要增加 RS232 驱动芯片，以避免信号衰减影响通讯。串口通讯的软件设置和通讯协议可参考乐升半导体的的串口屏应用手册（UI_Editor-II_CH_Vxx.pdf）。

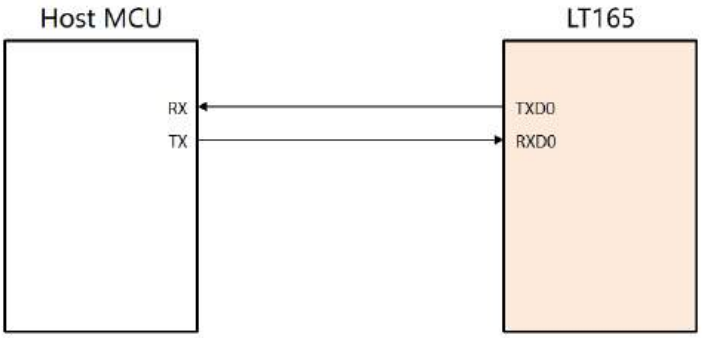


图 3-1：LT165A 的串口和主控 MCU 的通讯

3.2. TFT LCD 屏的控制接口

LT165A 提供了一组 8080 总线的 MCU 屏接口（External Bus Interface, EBI），用于驱动并行 8 位 8080 接口的 TFT LCD 面板，参考原理图如下图 3-2 所示：

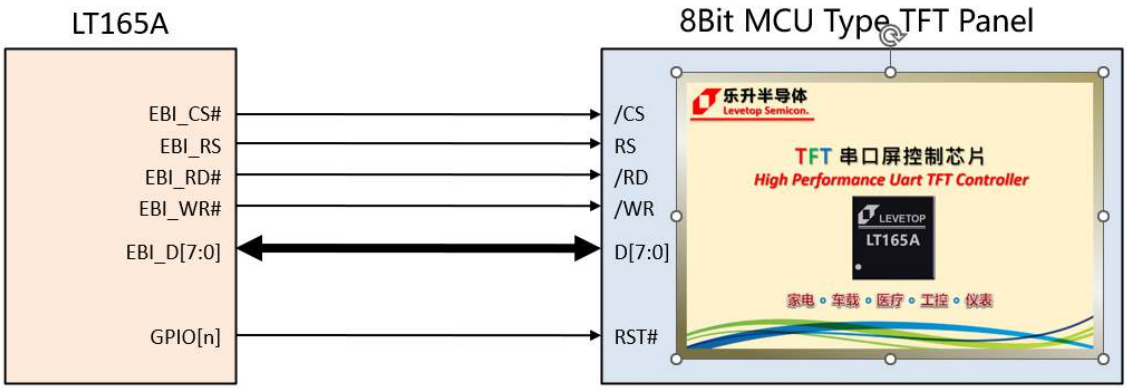


图 3-2：LT165A 与 8 位 8080 并行接口的 TFT 屏连接

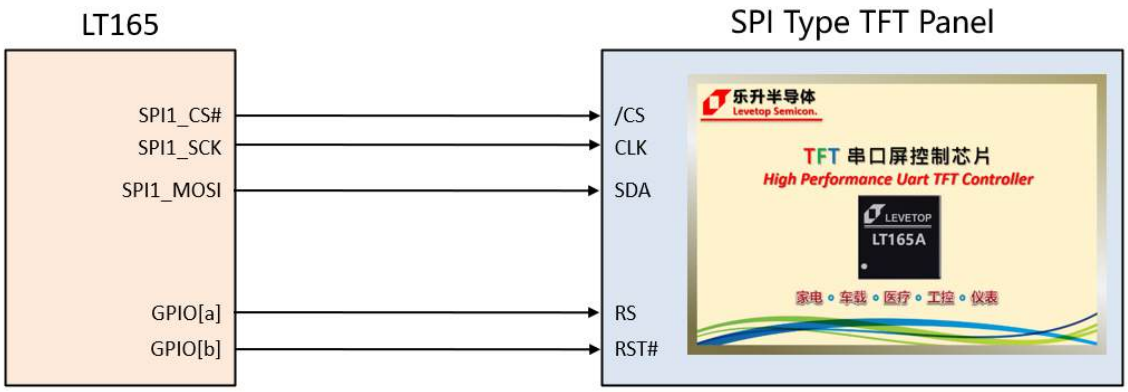


图 3-3：LT165A 与 SPI 串行接口的 TFT 屏连接

3.3. QSPI 接口

LT165A 有一组 QSPI 的接口，用来连接外部的 QSPI Flash，这个外部的 Flash 是被用做储存程序代码、显示图片、动画、文字和其他信息。当 LT165A 收到主控经由 Uart 接口发送过来的串口指令，它会根据指令从 QSPI Flash 提取图片或者其他显示相关信息，传输到 LCD 屏上，参考原理图如下图所示。

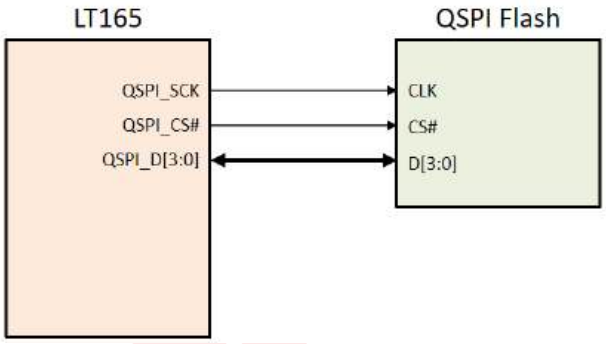


图 3-4：LT165A 连接 QSPI Flash 原理图

3.4. LCD 触控屏接口

LT165A 有一个 ADC 输入模块和 I2C 的控制电路，可以直接与外部 LCD 上面的触控屏连接，参考原理图如下：

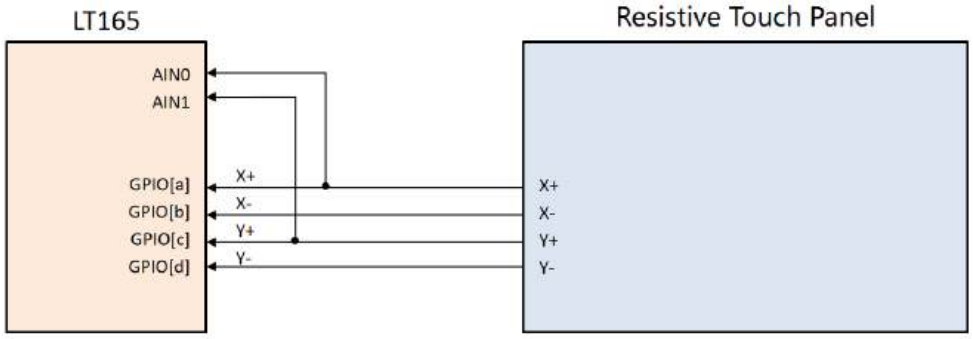


图 3-5：LT165A 连接到电阻触控屏

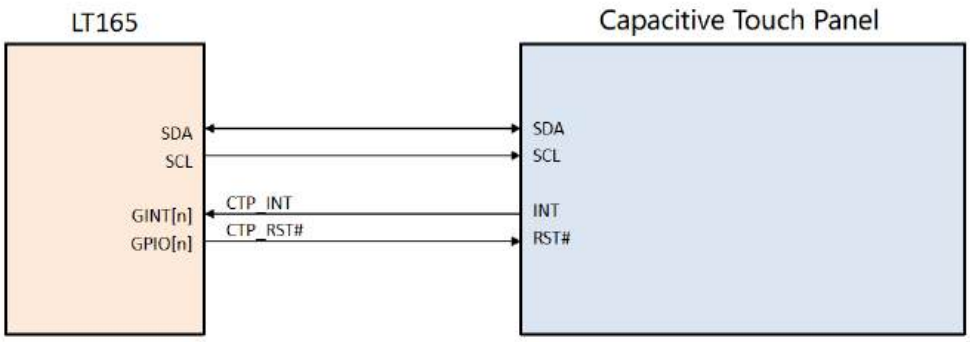


图 3-6：LT165A 连接到电容触控屏

3.5. 时钟信号源接口

LT165A 需要一个外部的 12MHz 晶振作为内部的系统时钟来源，参考原理图如下：

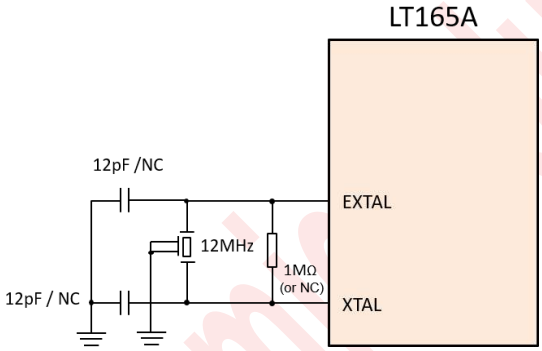


图 3-7：外部的 12MHz 晶振原理图

3.6. Can Bus 接口

LT165A 支持 Can Bus 协议，提供一组 Can Bus 接口，再经过一个 Can Bus 驱动芯片与外部通讯，参考原理图如下：

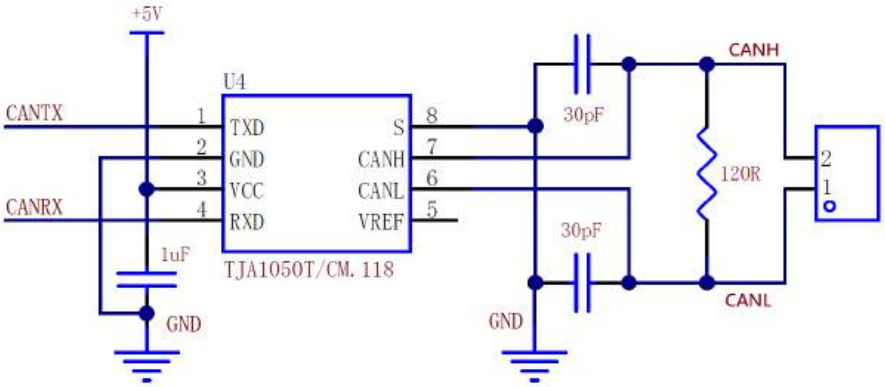


图 3-8：Canbus 驱动原理图

3.7. LCD 背光控制接口

LT165A 可以使用 PWM 中的一个通道来提供背光控制的信号 - "BL_PWM"，用来控制 TFT LCD 屏的背光或是背光的升压电路，参考原理图如下两个范例：

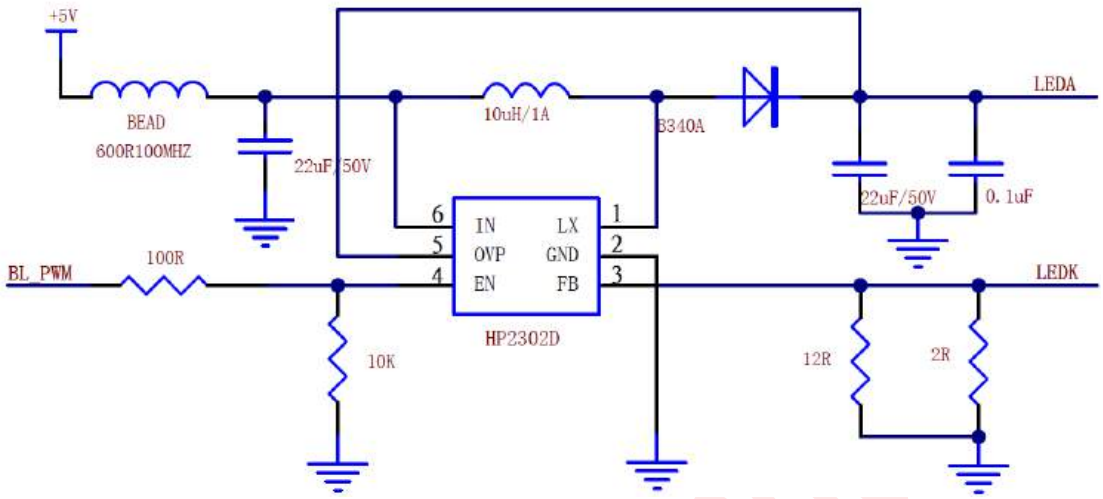


图 3-9：TFT LCD 背光控制参考原理图 - 1

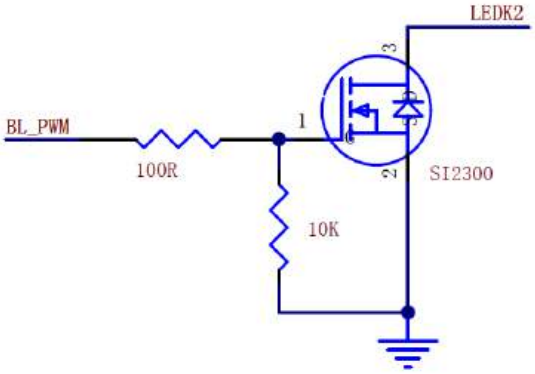


图 3-10：TFT LCD 背光控制参考原理图 - 2

3.8. RTC（Real Time Clock）的时钟源与电源

LT165A 内部有 RTC（Real Time Clock）时钟模块，如果使用这个 RTC 时钟，需要提供一 32.768KHz 的晶振电路，RTC 是独立供电，如果在外部电源关闭时仍保持 RTC 继续运行可以加上一外部电池电源，参考原理图如下：

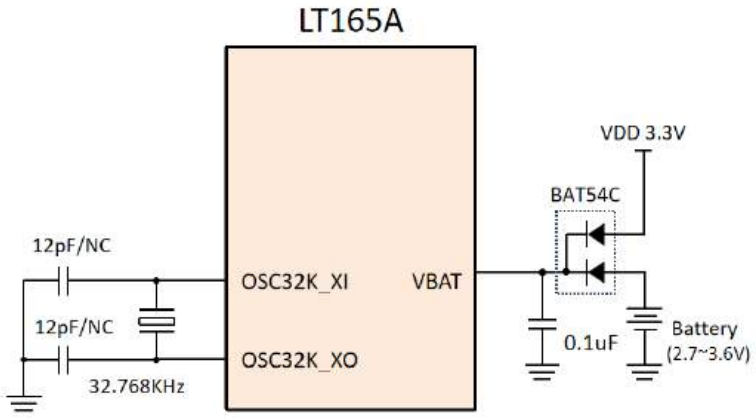


图 3-11：RTC 时钟源与外部电池电源的参考原理图

3.9. 复位（Reset）

LT165A 的硬件复位来源有 2 种，2 种复位都会经过内部时钟做同步处理：

- 电源开启复位（Power on Reset）
- 外部复位输入信号（External Reset Pin，RESET#）

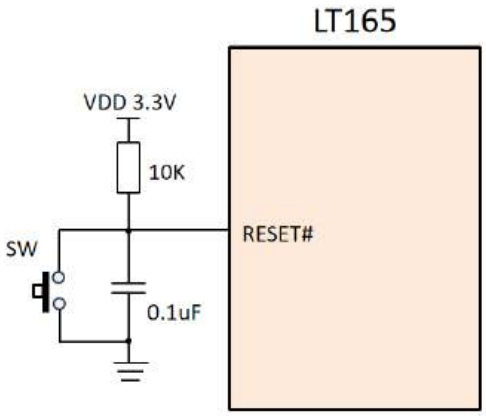


图 3-12：外部复位参考原理图

4. 嵌入式中断控制器（EIC）

本节介绍 MCU 的嵌入式中断控制器。

4.1. 介绍

中断控制器收集来自多个中断源的请求，并为 CPU 中断逻辑提供接口。

4.2. 特性

中断控制器模块的特性包括：

- 可配置中断源，最多 32 个
- 每个中断源有 32 个独特的可编程优先级
- 根据优先级，独立启用/禁用待处理中断
- 每个中断源都有一个固定的向量号
- 支持灵敏度级别和脉冲中断
- 支持 SuspendTrap 功能
- 支持软件重置

4.3. 内存映射和寄存器

本小节描述内存映射（参见表 4-1)和寄存器。

4.3.1. 内存映射

EIC 模块基地址（EIC_baseADDR）在 RISC 内核内部参数中定义，其默认值为 0xE000_0000。各 EIC 寄存器的实际地址由 EIC_BASEADDR 加上各寄存器的偏移地址构成。内核内部模块占用 64K 地址空间，系统应避免将其他寄存器映射到 EIC_BASEADDR 至 EIC_BASEADDR+0x0000_FFFF 之间的区域。

表 4-1：中断控制器模块内存映射

地址偏移量	Bit[31:24]	Bit[23:16]	Bit[15:8]	Bit[7:0]	访问权限 [1]
0x0000_0000	中断控制状态寄存器（ICSR）				S/U
0x0000_0004	保留				S/U
0x0000_0008	保留				S/U
0x0000_000C	保留				S/U
0x0000_0010	中断使能寄存器（IER）				S/U
0x0000_0014	保留				S/U
0x0000_0018	中断待处理寄存器（IPSR）				S/U
0x0000_001C	中断待处理清除寄存器（IPCR）				S/U
0x0000_0020 | 0x0000_003C	未实现(2)				—
优先级选择寄存器（PLSR0-PLSR31）					
0x0000_0040	PLSR3	PLSR2	PLSR1	PLSR0	S/U
0x0000_0044	PLSR7	PLSR6	PLSR5	PLSR4	S/U
0x0000_0048	PLSR11	PLSR10	PLSR9	PLSR8	S/U
0x0000_004C	PLSR15	PLSR14	PLSR13	PLSR12	S/U
0x0000_0050	PLSR19	PLSR18	PLSR17	PLSR16	S/U
0x0000_0054	PLSR23	PLSR22	PLSR21	PLSR20	S/U
0x0000_0058	PLSR27	PLSR26	PLSR25	PLSR24	S/U
0x0000_005C	PLSR31	PLSR30	PLSR29	PLSR28	S/U
0x0000_0060	系统优先级选择寄存器（SYSPLSR）				S/U
0x0000_0064 | 0x0000_007C	未实现(2)				—

提示：

1. RISC 内核中，寄存器在任何情况下都可以访问。
2. 访问未实现的地址位置没有效果，并导致循环终止传输错误。

4.3.2. 寄存器

本小节包含对中断控制器模块寄存器的描述。

4.3.2.1. 中断控制状态寄存器

32 位中断控制寄存器（ICSR）反映中断控制器输出到 CPU 的状态。

偏移地址：0x0000

	31	30	29	28	27	26	25	24
R	SRST	0	0	SetPTrap	ClrPTrap	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	VEC6	VEC5	VEC4	VEC3	VEC2	VEC1	VEC0
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 4-1：中断控制状态寄存器（ICSR）

SRST — 软件复位位

只读位用于创建软件复位请求。设置此位将在 SYSRESETREQ 信号上生成脉冲。读取总是返回 0。

SetPTrap — 设置 PendTrap 位

读/写位用于创建一个待处理的软件中断。该操作类似于执行“trap”指令。但是，在退出所有更高优先级的例外/中断之前，不会进入待处理的软件中断。当进入软件中断时，该位将被自动清除。复位也将清除此位。

读取时：

1 = 软件中断处于等待状态

0 = 软中断未待处理

写入时：

1 = 将软件中断设为待处理

0 = 无影响

ClrPTrap — 清除 PendTrap 位

读写 ClrDSI 位用于取消待处理的软件中断（PendTrap）。复位将清除此位。

读取时：

1 = 软件中断处于等待状态

0 = 软中断未待处理

写入时：

1 = 取消待处理的软件中断

0 = 无影响

VEC[6:0] — 中断向量数域

只读的 VEC[6:0] 字段包含 7 位中断向量号。复位清零 VEC[6:0]。

4.3.2.2. 中断使能寄存器

读写 32 位中断使能寄存器（IER）单独启用当前任何作为正常中断源分配给每个优先级的待处理中断。启用已断言请求的中断源将导致该请求变为待处理，并且如果尚未发出，则会断言对 CPU 的请求。

偏移地址：0x0010

	31	30	29	28	27	26	25	24
R W	IE31	IE30	IE29	IE28	IE27	IE26	IE25	IE24
RESET:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R W	IE23	IE22	IE21	IE20	IE19	IE18	IE17	IE16
RESET:	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R W	IE15	IE14	IE13	IE12	IE11	IE10	IE9	IE8
RESET:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 4-2：中断使能寄存器（IER）

IE[31:0] — 中断使能字段

读写 IE[31:0] 字段允许将对应优先级的源发出的中断请求作为中断请求处理。复位操作会清除 IE[31:0]。

1 = 中断请求已启用

0 = 禁用中断请求

4.3.2.3. 中断拨片设置寄存器

偏移地址：0x0018

	31	30	29	28	27	26	25	24
R W	SetPend31	SetPend30	SetPend29	SetPend28	SetPend27	SetPend26	SetPend25	SetPend24
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R W	SetPend23	SetPend22	SetPend21	SetPend20	SetPend19	SetPend18	SetPend17	SetPend16
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R W	SetPend15	SetPend14	SetPend13	SetPend12	SetPend11	SetPend10	SetPend9	SetPend8
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R W	SetPend7	SetPend6	SetPend5	SetPend4	SetPend3	SetPend2	SetPend1	SetPend0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 4-3：中断保持寄存器（IPSR）

SetPend[31:0] — 中断暂停设置字段

读写 SetPend[31:0] 字段将待处理状态设置为与关联中断相关联，并指示关联中断是否处于待处理状态。重置操作会清除 SetPend[31:0]。

我们读到：

1 = 相关中断处于等待状态

0 = 相关的中断不处于待处理状态

上面写道：

1 = 将相关中断的状态更改为待处理

0 = 无影响

4.3.2.4. 中断周期清除寄存器

偏移地址：0x001C

	31	30	29	28	27	26	25	24
R W	ClrPend31	ClrPend30	ClrPend29	ClrPend28	ClrPend27	ClrPend26	ClrPend25	ClrPend24
RESET:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R W	ClrPend23	ClrPend22	ClrPend21	ClrPend20	ClrPend19	ClrPend18	ClrPend17	ClrPend16
RESET:	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R W	ClrPend15	ClrPend14	ClrPend13	ClrPend12	ClrPend11	ClrPend10	ClrPend9	ClrPend8
RESET:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	ClrPend7	ClrPend6	ClrPend5	ClrPend4	ClrPend3	ClrPend2	ClrPend1	ClrPend0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 4-4：中断暂停清除寄存器（IPCR）

ClrPend[31:0] — 中断暂停清除字段

读/写 ClrPend[31:0] 字段清除了与相关中断的等待状态，并指明相关中断是否处于等待状态。重置将清除 ClrPend[31:0]。

我们读到：

1 = 相关中断处于等待状态

0 = 相关中断未处于待处理状态

上面写道：

1 = 将相关中断的状态更改为“非待处理”

0 = 无影响

4.3.2.5. 优先级选择寄存器

读/写 8 位优先级选择寄存器（PLSRx）是 32 个读/写、8 位优先级选择寄存器 PLSR0 ~ PLSR31，每个中断源一个。PLSRx 寄存器为中断源 x 分配一个优先级。

偏移地址：0x0040 ~ 0x005C

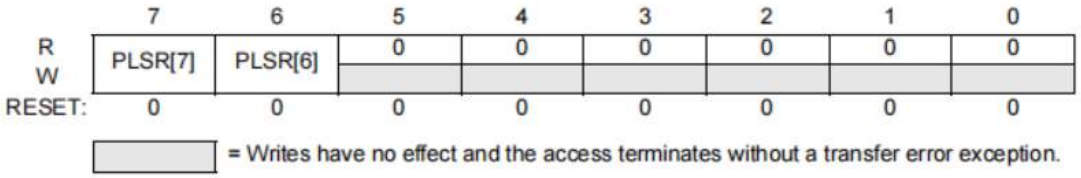


图 4-5：优先级选择寄存器（PLSR0 ~ PLSR31）

PLSRx[7:6] — 优先级选择字段

IRQ0~31 的默认优先级值为 0~31。数值越小，优先级越高，默认情况下优先级排序为：IRQ0>IRQ1>...>IRQ31。不过用户可以通过设置 PLSRx[7:6] 来调整中断优先级，实际优先级值等于默认值乘以 PLSRx[7:6] 的 64 倍。举个例子，如果 PLSR1[7:6] 设为 2，那么 IRQ1 的优先级值就是 1+2×64 = 129，这意味着 IRQ1 的优先级比所有数值更低的 IRQ 都要低。

表 4-2：优先级值调整

PLSRx[7:6]	优先级值
00	0
01	64
10	128
11	192

4.3.2.6. 系统优先级选择寄存器

偏移地址：0x0060

	31	30	29	28	27	26	25	24
R	EPTPRI[7]	EPTPRI[6]	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	SIPRI[7]	SIPRI[6]	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	0	0						
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 4-6：系统优先级选择寄存器（SYSPLSR）

EPTPRI[7:6] — EPT 优先级选择字段

EPT 中断的默认优先级为-2。这意味着 EPT 中断的优先级默认高于其他常规 IRQ 和系统软件中断（PendTrap）。数值越小，优先级越高。不过用户可以通过设置 PRI[7:6] 来调整 EPT 中断的优先级。实际优先级值等于默认值加上 PRI[7:6] 乘以 64。例如，当 PRI[7:6] 设置为 2 时，EPT 中断的优先级值计算为-2+2×64 = 126。

表 4-3：EPT 优先级值调整

EPTPRI[7:6]	EPT 优先级值
00	0
01	64
10	128
11	192

SIPRI[7:6] — 软件中断优先级选择字段

软件中断（PendTrap）的默认优先级为-1。这意味着 EPT 中断的默认优先级高于其他常规 IRQ。数值越小，优先级越高。不过用户可以通过设置 SIPRI[7:6] 来调整软件中断的优先级。实际优先级值等于默认值加上 SIPRI[7:6] 乘以 64。例如当 SIPRI[7:6] = 2 时，软件中断的优先级值计算为-1+2×64 = 127。

表 4-4：软件中断优先级值调整

SIPRI[7:6]	软件中断优先级值
00	0
01	64
10	128
11	192

4.4. 功能说明

EIC 支持级别敏感中断和脉冲中断，中断源号为 1 到 32。

由于以下原因之一，中断变为待处理：

- EIC 检测到中断信号处于激活状态，而对应的中断未激活。
- EIC 检测到中断信号的上升沿

待处理中断保持待处理状态，直至出现下列情况之一：

- 处理器进入中断服务程序。这将中断状态从“待处理”更改为“激活”。然后：
  - 对于级别敏感的中断，当处理器从 ISR 返回时，EIC 对中断信号进行采样。如果信号被断言，则中断状态更改为“等待”，这可能会导致处理器立即重新进入 ISR。否则，中断状态更改为“非活动”。
  - 对于脉冲中断，EIC 会持续监测中断信号。若检测到脉冲信号，则中断状态将切换为“待处理”和“激活”。此时，当处理器从 ISR 返回时，中断状态会变为“待处理”，这可能导致处理器立即重新进入 ISR。如果处理器在 ISR 期间未检测到脉冲信号，那么当其从 ISR 返回时，中断状态将转为“非激活”。
- 软件将写入对应的中断暂停清除寄存器位。

4.4.1. 无冲突的中断处理

如果中断是脉冲式的，中断的状态将变为“等待”，如果没有冲突，中断将使处理器立即进入 ISR。当处理器从 ISR 返回时，中断的状态将变为“非活动”。

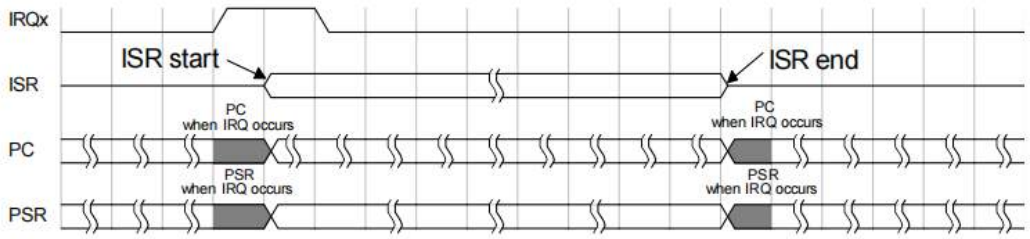


图 4-7：无冲突的单脉冲中断

对于层级敏感型中断，当信号被激活时，中断状态会变为“待处理”。若无冲突，该中断将立即触发处理器进入中断服务例程（ISR）。当处理器从 ISR 返回后，EIC 将继续采样中断信号。若信号未被清除，处理器将重新进入 ISR；否则，中断状态将转为“非活动”。

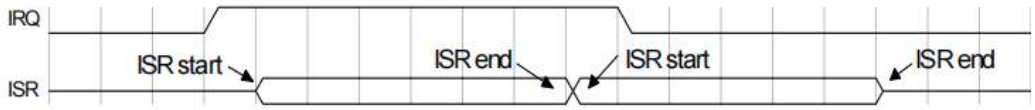


图 4-8：无冲突的电平敏感型中断

4.4.2. 中断与冲突

当两个中断信号同时被置为有效时，中断仲裁器将判断哪一个具有更高的优先级。例如，如果 IRQx 的优先级高于 IRQy，处理器将进入 ISRx 并且 IRQy 变为等待状态。在处理器从 ISRx 返回后，处理器将立即进入 ISRy。

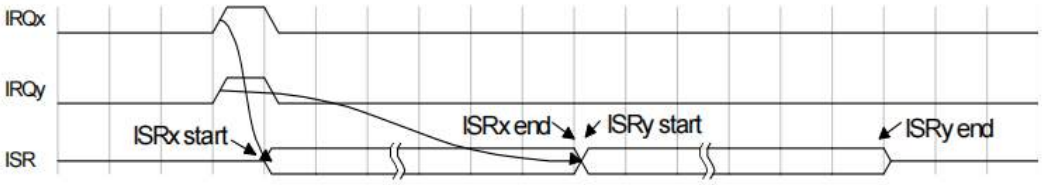


图 4-9：两个中断同时发生

如果在处理中断期间另一个中断信号发生，则会出现两种情况：

1、声明的中断优先级低于处理中断的优先级。在这种情况下，声明的中断状态将保持待处理状态，直到处理中断结束。

2、被断言的中断优先级高于处理中断优先级，此时高优先级中断处理将嵌套在低优先级中断例程中。

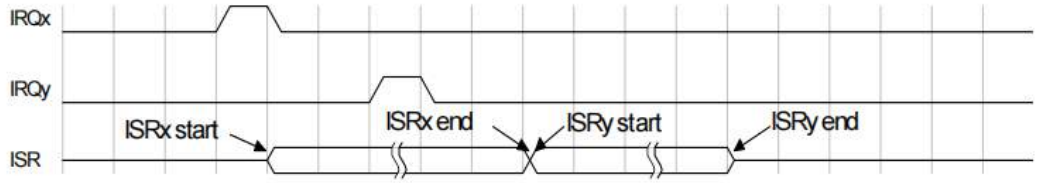


图 4-10：带冲突的低优先级中断发生

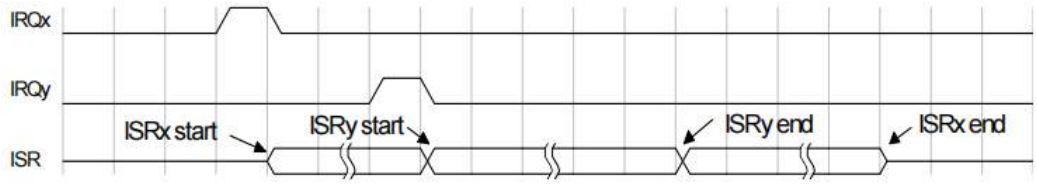


图 4-11：具有冲突的高优先级中断发生

4.4.3. Pend Trap 功能

ICSR 中的 SetPTrap / ClrPTrap 位用于在处理更高优先级中断时创建/取消“待处理”软件中断请求。处理器从更高优先级中断返回后，处理器将接受“待处理”软件中断。

通常，Pend Trap 功能用于 OS 任务被动切换。

4.5. 中断

中断控制器为每个中断源分配一个编号，如表 4-5 所示。

表 4-5：中断源分配

中断来源	模块	旗帜	来源描述	旗帜清除机制
0	ADC			
1	QSPI0	XRXOIS	XIP 接收 FIFO 溢出	
		RXFIS	接收 FIFO 全中断	
		RXOIS	接收 FIFO 溢出	
		RXUIS	接收 FIFO 下溢	
		TXOIS	发送器 FIFO 溢出	
		TXEIS	发送器 FIFO 空中断	
2	SCI0	TDRE	传输数据寄存器为空	
		TC	传输完成	
		TXOF	发送器缓冲区溢出	
		LBKDIF	LIN 检测中断报警	
		IDLE	闲置线路标志	
		RXEDGIF	RXD0 引脚有效边沿	
		RDRF	接收数据寄存器满	
		MA1F	Match 1 Flag	
		MA2F	Match 2 Flag	
		OR	接收器超限标志	
		NF	噪声标志	
		FE	框架错误标志	
		PF	奇偶校验错误标志	
		RXUF	接收器缓冲区下溢	
3	COMP0	CPRIF		
		CPFIF		
4	Reserved			
5	DMAC	DONE[0]		写入 DONE[0] = 1
		DONE[1]		写入 DONE[1] = 1
		...		...
		DONE[14]		写入 DONE[14] = 1

中断来源	模块	旗帜	来源描述	旗帜清除机制
		DONE[15]		写入 DONE[15] = 1
		DMA_ESR[CPE]	组优先级错误	将通道编号写入
		DMA_ESR[GPE]	信道优先级错误	将通道编号写入
		DMA_ESR[SAE]	源地址错误	将通道编号写入
		DMA_ESR[SOE]	源偏移错误	将通道编号写入
		DMA_ESR[DAE]	目标地址错误	将通道编号写入
		DMA_ESR[DOE]	目标偏移错误	将通道编号写入
		DMA_ESR[NCE]	Nbytes/Citer 配置错误	将通道编号写入
		DMA_ESR[SGE]	散射/聚集配置错误	将通道编号写入
		DMA_ESR[SBE]	源总线错误	将通道编号写入
		DMA_ESR[DBE]	目的地巴士错误	将通道编号写入
6	WDT0	IF		
7	PWM0	PIFR[0]		
		PIFR[1]		
		PIFR[2]		
		PIFR[3]		
8	Reserved			
9	PIT0	PIF	PIT 标志	Writing a 1 to it or writing to PMR
10	PIT1	PIF	PIT 标志	Writing a 1 to it or writing to PMR
11	Reserved			
12	Reserved			
13	RTC	Day_intf	日脉冲标志	
		Hou_intf	时钟脉冲标志	
		Min_intf	分钟脉冲标志	
		Sec_intf	秒脉冲标志	
		Ala_intf	闹钟标志	
		1KHz_intf	1KHz 脉冲标志	
		32KHz_intf	32KHz 脉冲标志	
14	TOUCH			
15	I2C	I2C Flag	I2C 标志	
16	Reserved			
17	PVD	PVDO	PVD 标志	

中断来源	模块	旗帜	来源描述	旗帜清除机制
18	CANBUS	CAN_IFRH[BUF15:BUF0]	CAN 缓冲器 15 ~ 0 中断	通过将其写入“1”来清除该位
19	CANBUS	CAN_ESR[BOFF_INT]	CAN 总线中断	通过将其写入“1”来清除该位
20	CANBUS	CAN_ESR[E RR_INT]	CAN 错误中断	通过将其写入“1”来清除该位
21	CANBUS	CAN_ESR[TWRN_INT]	CAN 发送警告中断	通过将其写入“1”来清除该位
22	CANBUS	CAN_ESR[RWRN_INT]	CAN 可以收到警告中断	通过将其写入“1”来清除该位
23	CANBUS	CAN_ESR[WKUP_INT]	唤醒中断	通过将其写入“1”来清除该位
24	Reserved			
25	Reserved			
26	SPI0	MODF	模式故障	
		EOTF	传输完成	Write-One Clear
		TXFTO	TXFIFO 超时	Write-One Clear
		TXFOVF	TXFIFO 溢出	Write-One Clear
		TXFUDF	TXFIFO 下溢	Write-One Clear
		TXFSER	TXFIFO 服务	Write-One Clear
		RXFTO	RXFIFO 超时	Write-One Clear
		RXFOVF	RXFIFO 溢出	Write-One Clear
		RXFUDF	RXFIFO 下溢	Write-One Clear
		RXFSER	后进先出服务	Write-One Clear
27	SPI1	MODF	模式故障	
		EOTF	传输完成	Write-One Clear
		TXFTO	TXFIFO 超时	Write-One Clear
		TXFOVF	TXFIFO 溢出	Write-One Clear
		TXFUDF	TXFIFO 下溢	Write-One Clear
		TXFSER	TXFIFO 服务	Write-One Clear
		RXFTO	RXFIFO 超时	Write-One Clear
		RXFOVF	RXFIFO 溢出	Write-One Clear

中断来源	模块	旗帜	来源描述	旗帜清除机制
		RXFUDF	RXFIFO 下溢	Write-One Clear
		RXFSER	后进先出服务	Write-One Clear
28	SCI1	TDRE	传输数据寄存器空标志	
		TC	传输完成	
		TXOF	发送器缓冲区溢出标志	
		LBKDIF	LIN 检测中断报警	
		IDLE	闲置线路标志	
		RXEDGIF	RXD1 引脚有效边沿	
		RDRF	接收数据寄存器满	
		MA1F	Match 1 Flag	
		MA2F	Match 2 Flag	
		OR	接收器超限标志	
		NF	噪声标志	
		FE	框架错误标志	
		PF	奇偶校验错误标志	
		RXUF	接收器缓冲区下溢标志	
29	Reserved			
30	EPORT0	EPF0	边缘端口 0 标志 0	写入 EPF0 = 1
		EPF1	边缘端口 0 标志 1	写入 EPF1 = 1
		EPF2	边缘端口 0 标志 2	写入 EPF2 = 1
		EPF3	边缘端口 0 标志 3	写入 EPF3 = 1
		EPF4	边缘端口 0 标志 4	写入 EPF4 = 1
		EPF5	边缘端口 0 标志 5	写入 EPF5 = 1
		EPF6	边缘端口 0 标志 6	写入 EPF6 = 1
		EPF7	边缘端口 0 标志 7	写入 EPF7 = 1
31	EPORT1	EPF0	边缘端口 1 标志 0	写入 EPF0 = 1
		EPF1	边缘端口 1 标志 1	写入 EPF1 = 1
		EPF2	边缘端口 1 标志 2	写入 EPF2 = 1

5. RISC 核心简介

本文档描述了基于 M*Core 指令集/体系结构的 RISC Core 微处理器的功能，该微处理器专为极低功耗和成本敏感的嵌入式控制应用而设计。

RISC 内核采用新型的三阶段 Pipeline 冯·诺依曼架构(von Neumann architecture)，体积更小，功耗更低。

RISC Core 还集成了一个 EIC（嵌入式中断控制器），以减少系统面积。

外部总线接口协议为 AHB-lite，RISC 内核设计中可提供更多的配置选项，利用这些可配置的选项，可以更灵活地进行性能、功能和成本之间的权衡，RISC 内核的门数根据不同的配置从 12K 到 20K 不等。

5.1. 特性

RISC 内核的主要特性如下：

- 32 位加载/存储精简指令集计算机（RISC）体系结构，具有固定的 16 位指令长度
- 16 个 32 位通用寄存器文件
- 高效的三级执行管道(Pipeline)，对应用软件不可见
- 单周期指令执行：支持多条指令，分支操作仅需三个周期
- 支持字节/半字/字内存访问
- 嵌入式中断控制器，支持嵌套向量中断和低功耗模式唤醒
- 单周期 32 位×32 位硬件整数乘法器阵列
- 3~13 周期硬件整数分频器阵列
- AHB-轻量级外部总线

5.2. 微架构概要

RISC 内核采用三级流水线来执行指令。指令获取、指令译码/寄存器文件读取、执行/写回阶段以重叠方式运行，允许大多数指令在单个时钟周期内执行。

为源操作数和指令结果提供了 16 个通用寄存器，其中寄存器 R15 用作链接寄存器，用于保存子程序调用的返回地址，寄存器 R0 按照约定与当前堆栈指针值相关联。

该设计采用双入口 32 位指令缓冲器，通过每个时钟周期从内存中获取两条指令进行预取，最多可缓存三条指令，从而有效减少或消除总线资源与数据内存访问之间的冲突。统一总线架构无需依赖昂贵的双总线结构，即可同时满足指令传输和数据传输的带宽需求。

内存读写操作支持字节、半字和整型（32 位）数据类型，其中字节和半字加载数据会自动进行零扩展。这些指令可采用流水线技术处理，使短序列操作实现单周期高效吞吐量。数据依赖型操作可在两个时钟周期内完成。多寄存器加载/存储指令可显著降低上下文保存与恢复操作的开销，其执行时间仅为（N+1）个时钟周期，其中 N 表示需要传输的寄存器数量。

单个条件码/进位(C)位用于条件测试，并支持实现超过 32 位的算术和逻辑运算。通常情况下，C 位仅通过显式的测试/比较操作被置位，而非作为常规指令执行过程中的副产品。但当需要将条件设置与实际计算操作相结合时，这一规则就会出现例外情况。

5.3. 编程模型

RISC 内核编程模型为两种特权模式：监督者和用户单独定义，使用 HPROT[1] 位来指示特权模式。

程序通过指定模式访问寄存器。用户程序只能访问其专属的用户模式寄存器，而处于监督模式运行的系统软件则能访问所有寄存器，并通过控制寄存器执行监督功能。这种设计有效限制了用户程序对特权信息的访问权限，操作系统则通过协调各程序的运行活动，为用户程序提供管理服务支持。

所有指令均可在任一模式下执行。用户程序还可执行停止、暂停或等待指令。陷阱#n 指令为用户提供对操作系统服务的受控访问。为防止用户程序以非受控方式进入监督器模式，可修改程序状态寄存器（PSR）中 S 位的指令具有特权。

当 PSR 中的 S 位被设置时，处理器在监督模式下执行指令。与指令相关的总线周期根据模式指示监督或用户访问。

处理器在常规用户模式下运行时采用用户编程模型。当进入异常处理模式时，处理器会从用户模式切换至监督模式。异常处理过程会将程序状态寄存器（PSR）的当前值保存到堆栈内存，并设置 PSR 中的 S 位，从而强制处理器切换至监督模式。若需恢复原操作模式，系统例程可执行 RTE（异常返回）指令，该指令会触发指令流水线的清空操作，并从对应地址空间重新加载指令。

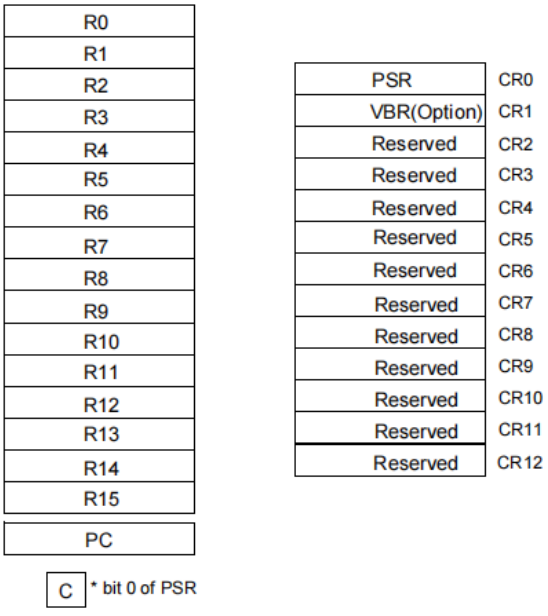


图 5-1：编程模型

编程模型中展示的寄存器（见图 5-1)负责操作数存储与控制功能。用户编程模型包含 16 个通用 32 位寄存器、32 位程序计数器（PC）以及条件/进位(C)位。其中 C 位在 PSR 寄存器中以第 0 位实现。按照行业惯例，寄存

器 R15 作为子程序调用的链接寄存器，而寄存器 R0 通常用作当前堆栈指针。

5.4. 数据格式摘要

整数运算单元支持的标准二进制补码数据格式作为操作数类型。每条指令的操作数长度有两种编码方式：一种是直接在指令中明确定义（如加载/存储指令），另一种则是通过指令操作类型隐式定义（如索引操作、字节提取）。通常情况下，指令会作用于源操作数(s)的全部 32 位，并生成对应的 32 位结果。

内存的字节顺序取决于处理器配置（参见图 5-2），可分为大端序和小端序两种模式。在大端序模式（默认工作模式）下，字 0 的最高有效字节（字节 0）位于地址 0 处。而在小端序模式中，字 0 的最高有效字节则位于地址...寄存器中的位从第 31 位开始编号，其中第 31 位是最高有效位（参见图 5-3）。按照行业惯例，寄存器字节 0 始终是最高有效字节，无论采用何种字节顺序模式。只有在执行 xtrb[0-3] 指令时，这个问题才会显现。

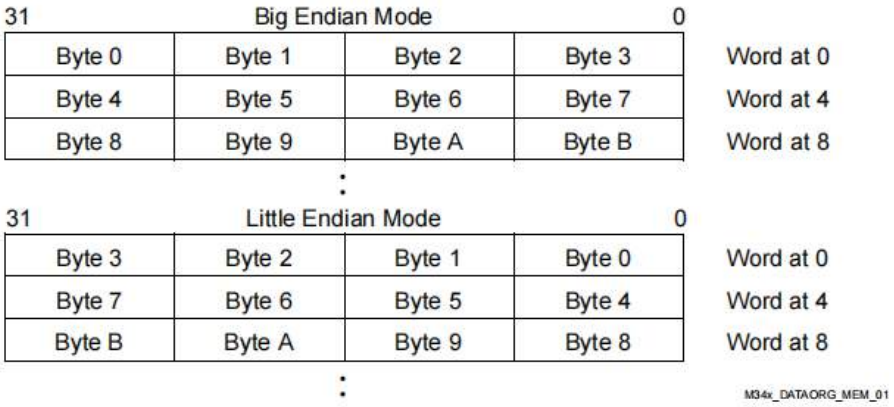


图 5-2：内存中的数据结构

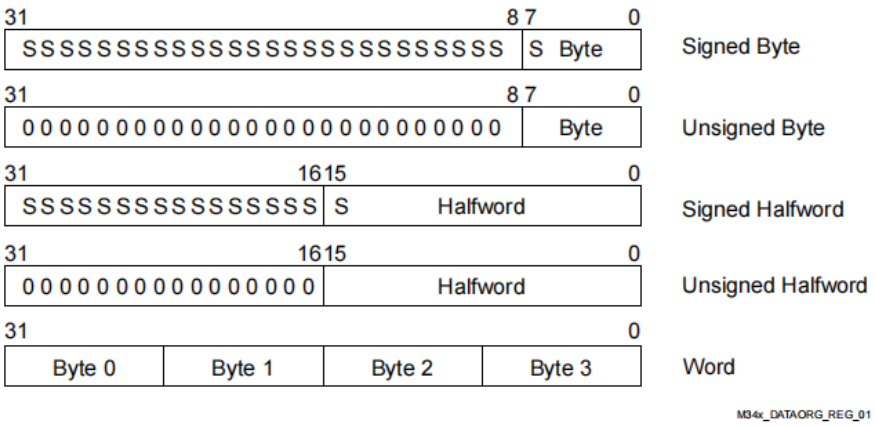


图 5-3：寄存器中的数据结构

5.5. 操作数寻址能力

RISC 内核通过加载和存储指令访问所有内存操作数，实现通用寄存器（GPR）与内存之间的数据传输。在加载和存储指令中采用+4 位寄存器缩放偏移寻址模式，用于寻址字节、半字或 32 位字数据。

加载和存储多个指令允许将 16 个 GPR 中的子集传输到或从寄存器 R0（默认的堆栈指针）所指向的基地址。

加载和存储寄存器象限指令使用寄存器间接寻址，将寄存器象限传输到内存或从内存传输到寄存器象限。

5.6. 指令集概述

该指令集专为支持高级语言而设计，并针对最常执行的指令进行了优化。它不仅提供了一套标准的算术和逻辑指令，还支持位操作、字节提取、数据移动、控制流修改等功能。此外，还包含少量条件执行指令，这些指令有助于减少短条件分支的使用。

表 5-1 提供了按字母顺序排列的 RISC 核心指令集列表。

表 5-1：RISC 核心指令集

助记符号	描述
ABS	Absolute Value
ADDC	Add with C bit
ADDI	Add Immediate
ADDU	Add Unsigned
AND	Logical AND
ANDI	Logical AND Immediate
ANDN	AND NOT
ASR	Arithmetic Shift Right
ASRC	Arithmetic Shift Right, update C bit
ASRI	Arithmetic Shift Right Immediate
BCLRI	Clear Bit
BF	Branch on Condition False
BGENI	Bit Generate Immediate
BGENR	Bit Generate Register
BKPT	Breakpoint
BMASKI	Bit Mask Immediate
BR	Branch
BREV	Bit Reverse
BSETI	Bit Set Immediate
BSR	Branch to Subroutine
BT	Branch on Condition True

助记符号	描述
CLRF CLRT CMPHS CMPLT CMPLTI	Clear Register on Condition False Clear Register on Condition True Compare Higher or Same Compare Less-Than Compare Less-Than Immediate
CMPNE CMPNEI	Compare Not Equal Compare Not Equal Immediate
DECF DECGT DECLT DECNE DECT DIVS[1] DIVU[1] DOZE	Decrement on Condition False Decrement Register and Set Condition if Result Greater-than Zero Decrement Register and Set Condition if Result Less-than Zero Decrement Register and Set Condition if Result Not Equal to Zero Decrement On Condition True Divide Signed Integers Divide Unsigned Integers Doze
FF1[1] INCF	Find First One Increment on Condition False
INCT IXH IXW	Increment On Condition True Index Halfword Index Word
JAVASW JMP JMPI JSR JSRI	Java interpreter switch Jump Jump Indirect Jump to Subroutine Jump to Subroutine Indirect
LD [BHW] LDM LDQ LRW LSL, LSR LSLC, LSRC LSLI, LSRI	Load Load Multiple Registers Load Register Quadrant Load Relative Word Logical Shift Left and Right Logical Shift Left and Right, update C bit Logical Shift Left and Right by Immediate
MFCR MOV MOVI MOVF MOVT MTCR MULSH MULT	Move from Control Register Move Move Immediate Move on Condition False Move on Condition True Move to Control Register Multiply signed Halfwords Multiply
MVC MVCV	Move C bit to Register Move Inverted C bit to Register
NOT	Logical Complement

助记符号	描述
OR	Logical Inclusive-OR
ROTLI RSUB RSUBI	Rotate Left by Immediate Reverse Subtract Reverse Subtract Immediate
RTE RFI	Return from Exception Return from Interrupt
SEXTB SEXTH ST[BHW] STM STQ	Sign-extend Byte Sign-extend Halfword Store Store Multiple Registers Store Register Quadrant
STOP SUBC SUBU SUBI	Stop Subtract with C bit Subtract Subtract Immediate
SYNC	Synchronize
TRAP TST TSTNBZ	Trap Test Operands Test for No Byte Equal Zero
WAIT	Wait
XOR XSR XTRB0 XTRB1 XTRB2 XTRB3	Exclusive OR Extended Shift Right Extract Byte 0 Extract Byte 1 Extract Byte 2 Extract Byte 3
ZEXTB ZEXTH	Zero-extend Byte Zero-extend Halfword

提示：

1. 当前版本中未实现。

6. 嵌入式可编程定时器（EPT）

6.1. 介绍

嵌入式可编程计时器（EPT）是一个 24 位计时器，可在最少处理器干预的情况下提供精确的定时中断。该计时器可以基于重置值进行倒计数，也可以作为自由运行的递减计数器。

EPT 中断可触发异常（向量号 = 24)。

通过将参数“EPT”清除为 0，可移除 EPT 模块，从而减少核心栅极数量。

6.2. 内存映射和寄存器

6.2.1. 内存映射

EPT 基地址定义为 EIC_BASEADDR + 0x1000，默认的基地址是 0xE000_1000，表 6-1 显示了 EPT 寄存器的偏移地址，EPT 模块占用 4K 地址区域。

表 6-1：可编程定时器模块内存映射

地址偏移量	Bit[31:24]	Bit[23:16]	Bit[15:8]	Bit[7:0]	访问权限
0x0000_0000	EPT 控制和状态寄存器（EPTCSR）				S/U
0x0000_0004	EPT 重载寄存器（EPTRLD）				S/U
0x0000_0008	EPT 计数寄存器（EPTCNT）				S/U
0x0000_000C	保留				S/U

6.2.2. 寄存器

本小节包含对 EPT 模块寄存器的描述。

6.2.2.1. EPT 控制状态寄存器

偏移地址：0x0000

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	CNTFLAG
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	CLKSRC	INTEN	CNTEN
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 6-1：EPT 控制状态寄存器（EPTCSR）

CNTFLAG — 倒计时至 0 标志

只读位表示计时器计数为 0。它将由 HRESETn 重置。

1 = 计时器倒计时至 0。

0 = 计时器仍在倒计时。

CLKSRC — 计数时钟源选择

读写位用于选择计数时钟源。它将由 HRESETn 重置。

1 = 核心时钟。

0 = 外部参考时钟。

INTI — EPT 中断请求使能

读/写位用于在计时器倒计时至 0 时启用 EPT 的中断，它将由 reset 重置。

1 = 当计时器倒计时至 0 时，发生 EPT 异常请求。

0 = 当计时器倒数至 0 时，不会出现 EPT 例外请求。

CNTEN — 反向启用

读/写位用于启用 EPT 的计数器。它将由 reset 重置。

1 = 启用计数器

0 = 禁用计数器

6.2.2.2. EPT 重载寄存器

偏移地址：0x0004

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	x	x	x	x	x	x	x	x

	23	22	21	20	19	18	17	16
R	RLD[23]	RLD[22]	RLD[21]	RLD[20]	RLD[19]	RLD[18]	RLD[17]	RLD[16]
W								
RESET:	x	x	x	x	x	x	x	x

	15	14	13	12	11	10	9	8
R	RLD[15]	RLD[14]	RLD[13]	RLD[12]	RLD[11]	RLD[10]	RLD[9]	RLD[8]
W								
RESET:	x	x	x	x	x	x	x	x

	7	6	5	4	3	2	1	0
R	RLD[7]	RLD[6]	RLD[5]	RLD[4]	RLD[3]	RLD[2]	RLD[1]	RLD[0]
W								
RESET:	x	x	x	x	x	x	x	x

= Writes have no effect and the access terminates without a transfer error exception.

图 6-2：EPT 重载寄存器（EPTRLD）

RLD[23:0] — 重载值

读写 RLD[23:0] 字段指定计时器倒计数至 0 时的重置值。该寄存器没有复位值。RLD 值可以是 0x0000_0001 ~ 0x00FF_FFFF 范围内的任意值。值 0 无效。要生成具有 N 个时钟周期的周期计时器，请将 RLD 设置为 N-1。

6.2.2.3. EPT 计数寄存器

偏移地址：0x0008

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	x	x	x	x	x	x	x	x

	23	22	21	20	19	18	17	16
R	CNT[23]	CNT[22]	CNT[21]	CNT[20]	CNT[19]	CNT[18]	CNT[17]	CNT[16]
W								
RESET:	x	x	x	x	x	x	x	x

	15	14	13	12	11	10	9	8
R	CNT[15]	CNT[14]	CNT[13]	CNT[12]	CNT[11]	CNT[10]	CNT[9]	CNT[8]
W								
RESET:	x	x	x	x	x	x	x	x

	7	6	5	4	3	2	1	0
R	CNT[7]	CNT[6]	CNT[5]	CNT[4]	CNT[3]	CNT[2]	CNT[1]	CNT[0]
W								
RESET:	x	x	x	x	x	x	x	x

= Writes have no effect and the access terminates without a transfer error exception.

图 6-3：EPT 计数器寄存器（EPTCNT）

CNT[23:0] — EPT 计数器值

只读寄存器指示 EPT 计时器的当前计数值。该寄存器没有复位值。

读取将返回 EPT 计数器的当前值。向该寄存器写入任何值都会将计数器值清零，并且将 CNTFLAG 清零。

6.3. 功能说明

启用后，EPT 会从 RLD 设定的值开始倒计时至零，并在下一个时钟周期对 RLD 中的数值进行循环加载。随后通过后续时钟周期继续倒计时，当将 RLD 写入零值时，系统将在下次循环加载时禁用计数器。当计数归零时，CLFAG 位将被置为 1，此时若 INTEN 功能已启用，ETP 将触发 ETP 中断。

读取 CSR 时，CFLAG 位被清除为 0。向 CNT 写入任何值时，CFLAG 位也被清除为 0。

6.3.1. 计数器计时

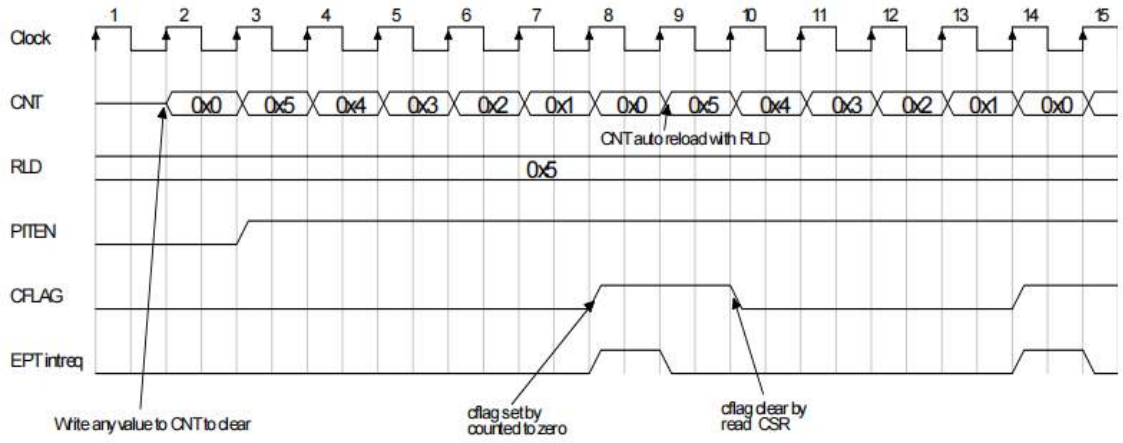


图 6-4：EPT 计数计时

7. 系统集成模块（SIM）

7.1. 介绍

系统集成模块（SIM）负责控制引脚配置、通用输入输出接口（GPIO）及内部外设复用功能。引脚配置模块负责管理 I/O 引脚的静态电气特性，而 GPIO 模块则为 MCU 的 I/O 引脚提供统一且独立的输入输出控制。SIM 通过外设总线与核心组件进行通信。图 7-1 展示了 SIM 及其与其他系统组件接口的方块图。需要特别说明的是，引脚接口/环形引脚模块和外设 I/O 通道均位于 SIM 外部。

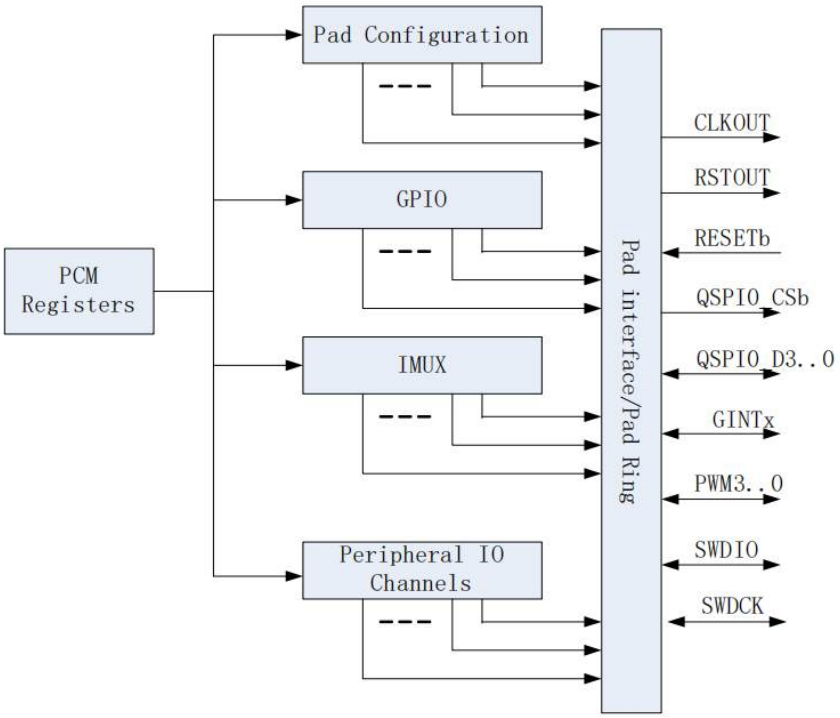


图 7-1：SIM 方块图

7.2. 特性

- 系统配置
  - 插槽配置控制
- GPIO
  - 31 个 I/O 引脚上的 GPIO 功能
  - 为每个 GPIO 引脚提供专用输入和输出寄存器
- 外部输入多路复用
  - 允许选择 UART 发送器/接收器引脚
  - 允许选择 CAN 总线接收器引脚
  - 允许选择 I2C 接收器引脚
- 配置唤醒功能
- 从外部闪存读取数据/指令时，配置加密/解密功能

7.3. 操作模式

7.3.1. 正常模式

在正常模式下，SIM 提供寄存器接口和控制端口配置、输入多路复用和 GPIO 的逻辑。

7.3.2. 调试模式

调试模式下的 SIM 操作与正常模式下的操作相同。

7.4. 内存映射

表 7-1 是 SIM 寄存器的地址映射。

表 7-1：SIM 地址映射

地址偏移量	寄存器	寄存器位数
0x0000 ~ 0x003F	引脚配置寄存器 0（SIM_PCR0） — 引脚配置寄存器 30 [1]	16
0x0040 ~ 0x03FF	保留	—
0x0400 ~ 0x041F	GPIO 引脚数据输出寄存器 0 ~ 3（SIM_GPDO0_3） — GPIO 引脚数据输出寄存器 28 ~ 31（SIM_GPDO28_31）[1]	8
0x0420 ~ 0x04FF	保留	—
0x0600 ~ 0x061F	GPIO 引脚数据输入寄存器 0 ~ 3（SIM_GPDI0_3） — GPIO 引脚数据输入寄存器 28 ~ 31（SIM_GPDI28_31）[1]	8
0x0620 ~ 0x06FF	保留	—
0x0800	SIM_PGPDO0 — 并行 GPIO 引脚数据输出寄存器 0	32
0x0804 ~ 0x08FF	保留	—
0x0900	SIM_PGPDI0 — 并行 GPIO 引脚数据输入寄存器 0	32
0x0904 ~ 0x08FF	保留	—
0x0A00	SIM_MPGPDO0 — 隐藏的并行 GPIO 引脚数据输出寄存器 0	32
0x0A04	SIM_MPGPDO1 — 隐藏的并行 GPIO 引脚数据输出寄存器 1	32
0x0A08 ~ 0x0BFF	保留	—
0x0C00	WKUPC — 唤醒配置寄存器	32
0x0C04	QSPIXIPCR — QSPI XIP 模式配置寄存器	32
0x0C08	QSPIKEYR — QSPI 32 位密钥寄存器	32
0x0C0C ~ 0x0FFF	保留	—

提示：在指定封装中，该存储空间存在 I/O 引脚不可用的间隙。

7.5. 寄存器说明

7.5.1. 片存储器配置寄存器（SIM_PCR）

以下各小节定义了所有设备引脚的寄存器配置寄存器（Pad Configuration Registers），用于配置引脚功能、方向及静态电气特性。本文档内容涉及特定引脚或引脚组的激活位与字段，以及寄存器的复位状态。需注意的是，下文所述寄存器配置寄存器（PCRs）的复位状态均以引导程序运行前的状态为准。该引导程序可根据当前复位配置对部分寄存器配置寄存器进行调整。

图 7-2 为 PCR 图谱示例，请注意以下内容：

- 寄存器位编号顺序遵循最高有效位为第 15 位的规则。字段位范围则相反 — 最低有效位称为第 0 位。
- 位 0 到 3 和位 12 到 13 是保留示例位。它们是只读的，且始终返回值 0。
- PCR 是 16 位寄存器，但可以读取或写入 32 位值，并以 32 位地址边界进行对齐。

偏移地址：0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	PA		OBE[1]	IBE[2]	DSC[2:0]			ODE	0	HYS	WPE	WPS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

= Unimplemented or Reserved

提示：

1. OBE 位在 GPIO 中很重要
2. IBE 位在 GPIO 中很重要

图 7-2：PCR 图谱示例

表 7-2 列出了并描述了各寄存器中包含的字段，不是所有字段都出现在每个寄存器中，但每个字段在其所在的每个寄存器中具有相同的功能。

表 7-2：SIM_PCR 字段描述

<table>
<tr><th>字段</th><th>描述</th></tr>
<tr><td>Reserved</td><td>寄存器图中用阴影表示保留字段。</td></tr>
<tr><td>PA</td><td>引脚分配
选择多路复用引脚的功能。
<table><tr><th>PA Value[1]</th><th colspan="2">Pin Function</th></tr><tr><td>0b11</td><td>P</td><td>Primary function</td></tr><tr><td>0b01</td><td>A1</td><td>Alternate function</td></tr><tr><td>0b10</td><td>A2</td><td>Alternate function2</td></tr><tr><td>0b00</td><td>G</td><td>GPIO</td></tr></table></td></tr>
<tr><td>OBE[2,3]</td><td>输出缓冲器启用
将该引脚用作输出，并驱动输出缓冲器使能信号。

0：禁用引脚的输出缓冲器。
1：为引脚启用输出缓冲器。</td></tr>
<tr><td>IBE[2,3]</td><td>输入缓冲区启用
使该引脚作为输入启用，并驱动输入缓冲器启用信号。

0：禁用引脚的输入缓冲器。
1：启用了引脚的输入缓冲器。

输入/输出功能（I/O）的位定义在不同 PCR 中各有差异。当某个 I/O 功能仅作为输入或输出时，无需设置 IBE 和 OBE 位即可启用相应输入或输出功能。若 I/O 功能支持双向切换，则需根据需要设置 IBE 和 OBE 位（输入时 IBE 置 1，输出时 OBE 置 1）。对于动态改变传输方向的I/O功能（如外部数据总线），其输入/输出切换由内部机制处理，此时IBE和OBE位将不再产生影响。

对于所有 GPIO 功能可用的引脚，如果将该引脚配置为输出并且设置了 IBE 位，则该引脚的实际值将反映在相应的 GPDIx_x 寄存器中。如果将引脚配置为输出并且取消了 IBE 位，则可降低噪声和功耗。</td></tr>
</table>

字段	描述
DSC[2:0][4]	驱动控制强度 控制引脚驱动强度。驱动强度控制与具有快速 I/O 的引脚有关。 垫片类型。 000：6mA 驱动强度 001：13mA 驱动强度 010：19mA 驱动强度 011：26mA 驱动强度 100：32mA 驱动强度 101：39mA 驱动强度 110：45mA 驱动强度 111：52mA 驱动强度
HYS[5]	输入滞后 控制是否为引脚启用滞后功能。 0：禁用引脚的滞后。 1：为引脚启用滞后。
WPE[6]	弱上拉/下拉功能已启用 控制是否为引脚启用/禁用弱上拉/下拉装置。默认情况下，上拉/下拉装置处于启用状态。 0：禁用软垫的弱拉装置。 1：启用垫片的弱拉装置。
WPS[6]	弱上拉/下拉选择 控制启用弱上拉/下拉设备时，是否在引脚中使用弱上拉/下拉设备。 WKPCFG 引脚用于确定复位期间启用上拉或下拉器件。WPS 位确定复位后使用弱上拉或下拉器件，或者用于 WKPCFG 引脚未确定复位弱上/下拉状态的引脚。 0：为该垫启用下拉功能。 1：为引脚启用上拉功能。

提示：

1. PA 字段的长度可能因寄存器的不同而有所变化。对于少于三个比特的 PA 字段，应删除这些值中相

应数量的前导零。

2. 在 I/O 功能仅为输入或输出的情况下，无需设置 IBE 和 OBE 位即可启用引脚 I/O。
3. 对于动态改变方向的 I/O 功能，例如 SCL/SDA，输入和输出之间的切换由外围设备内部处理，并且忽略 IBE 和 OBE 位。
4. 如果将引脚配置为输入，则 ODE、SRC 和 DSC 位不适用。SRC 在此设备中没有意义。
5. 如果引脚被配置为输出，则 HYS 位不适用。HYS 在此设备中没有意义。
6. 当引脚配置为输出时，无论 PCR 中的 WPE 或 WPS 设置如何，弱内部上/下拉功能都将被禁用。

该设备上存在多个输入信号，这些信号通过多个外部引脚作为输入源。对于这些引脚而言，任何时候只能有一个输入源处于激活状态。为实现这一功能，系统为控制这些引脚的 PCR 寄存器预设了优先级，通过多路复用输入源来确保仅有一个输入源处于激活状态。具有 PCR 优先级的多源输入配置详见表 7-3。

表 7-3：多源输入的 PCR 优先级

Function	PCR 编号 最高优先级 → 最低优先级	
TXD1	2	12
RXD1	3	13
CAN_RX	13	30
SCL	14	29
SDA	15	30

表 7-4：SIM_PCRn 设置

PCRn	Offset Address	Primary Function	Alt1	Alt2	GPIO	SIM_PCRn Default Settings							
						PA[1:0]	OBE	IBE	DSC	ODE	HYS	WPE	WPS
0	0x02	PWM0	GINT0[0]	COMP0_OUT	GPIO0	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
1	0x00	PWM1	GINT0[1]		GPIO1	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
2	0x06	PWM2	GINT0[2]	TXD1	GPIO2	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
3	0x04	PWM3	GINT0[3]	RXD1	GPIO3	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
4	0x0A	TK0	GINT0[4]		GPIO4	2'b11	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
5	0x08	TK1	GINT0[5]		GPIO5	2'b11	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
6	0x0E	RSTOUT	GINT0[6]	TK2	GPIO6	2'b11	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
7	0x0C	CLKOUT	GINT0[7]	TK3	GPIO7	2'b11	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
8	0x12	EBI_DB0	SPI0_CS#		GPIO8	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
9	0x10	EBI_DB1	SPI0_MOSI		GPIO9	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
10	0x16	EBI_DB2	SPI0_MISO		GPIO10	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
11	0x14	EBI_DB3	SPI0_SCK		GPIO11	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
12	0x1A	EBI_DB4	TXD1	CAN_TX	GPIO12	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
13	0x18	EBI_DB5	RXD1	CAN_RX	GPIO13	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
14	0x1E	EBI_DB6	SCL		GPIO14	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
15	0x1C	EBI_DB7	SDA		GPIO15	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
16	0x22	SWDIO	GINT1[0]	AIN0	GPIO16	2'b11	1'b0	1'b0	3'h4	1'b0	1'b1	1'b1	1'b1
17	0x20	SWDCK	GINT1[1]	AIN1	GPIO17	2'b11	1'b0	1'b0	3'b0	1'b0	1'b1	1'b1	1'b1
18	0x26	AIN2	GINT1[2]		GPIO18	2'b11	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
19	0x24	QSPI0_CS#			GPIO19	2'b00	1'b0	1'b0	3'h7	1'b0	1'b0	1'b0	1'b0
20	0x2A	QSPI0_D0			GPIO20	2'b00	1'b0	1'b0	3'h7	1'b0	1'b0	1'b0	1'b0
21	0x28	QSPI0_D1			GPIO21	2'b00	1'b0	1'b0	3'h7	1'b0	1'b0	1'b0	1'b0
22	0x2E	QSPI0_D2			GPIO22	2'b00	1'b0	1'b0	3'h7	1'b0	1'b0	1'b0	1'b0
23	0x2C	QSPI0_D3			GPIO23	2'b00	1'b0	1'b0	3'h7	1'b0	1'b0	1'b0	1'b0
24	0x32	QSPI0_SCK			GPIO24	2'b00	1'b0	1'b0	3'h7	1'b0	1'b0	1'b0	1'b0
25	0x30	EBI_CS#	SPI1_CS#		GPIO25	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
26	0x36	EBI_WR#	SPI1_MOSI		GPIO26	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
27	0x34	EBI_RD#	SPI1_MISO		GPIO27	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
28	0x3A	EBI_RS	SPI1_SCK		GPIO28	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
29	0x38	TXD0	SCL	CANTX	GPIO29	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
30	0x3E	RXD0	SDA	CANRX	GPIO30	2'b00	1'b0	1'b0	3'b0	1'b0	1'b0	1'b0	1'b0
31	0x3C	CLKMODE				2'b11	1'b0	1'b0	3'b0	1'b0	1'b1	1'b1	1'b1

7.5.2. GPIO 引脚数据输出寄存器（SIM_GPDO0_3 ~ SIM_GPDO28_31）

SIM_GPDOx_x 寄存器的定义如图 7-3 至图 7-10 所示。32 个 PDO 位（其中 31 位已使用）分别对应相同 GPIO 引脚编号的引脚。例如，PDO0 是 PWM0_GINT0[0]_GPIO0 引脚的数据输出位，而 PDO30 则是 RXD0_SDA_CAN_RX_GPIO30 引脚的数据输出位。该存储空间中存在空隙，表示封装中未包含这些引脚。

SIM_GPDOx_x 寄存器通过软件写入，用于驱动外部 GPIO 引脚输出数据。每个寄存器字节对应一个独立的外部 GPIO 引脚，可实现各引脚状态的自主控制。若相关引脚已通过引脚配置寄存器设置为输入模式，则对 SIM_GPDOx_x 寄存器的写入操作不会改变引脚状态。当 GPIO 引脚方向从输入切换为输出时，SIM_GPDOx_x 寄存器的数值将自动同步到 GPIO 引脚，无需额外软件更新。

当引脚被相应的 PCR 配置为主要或备用功能时，写入 SIM_GPDOx_x 寄存器的操作不会对相应引脚的状态产生影响。

偏移地址：0x0400

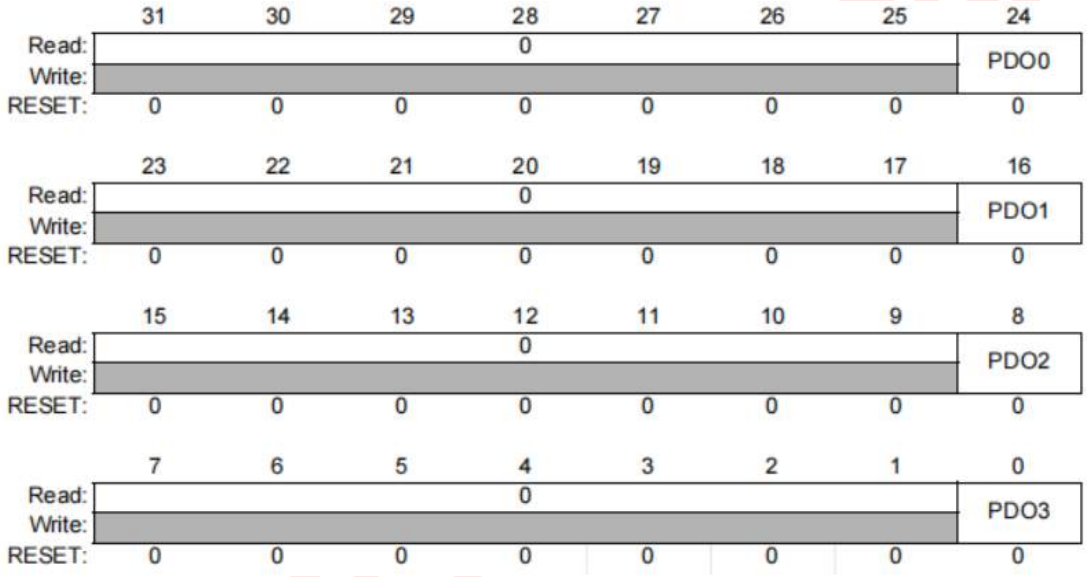


图 7-3：GPIO 引脚数据输出寄存器 0 ~ 3（SIM_GPDO0 ~ SIM_GPDO3）

偏移地址：0x0404

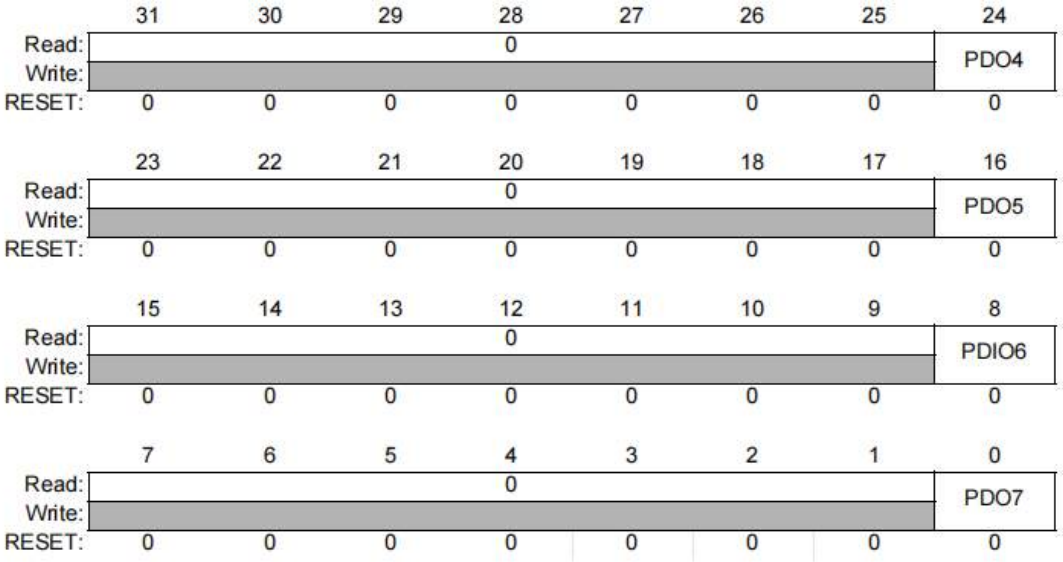


图 7-4：GPIO 引脚数据输出寄存器 4 ~ 7（SIM_GPDO4 ~ SIM_GPDO7）

偏移地址：0x0408

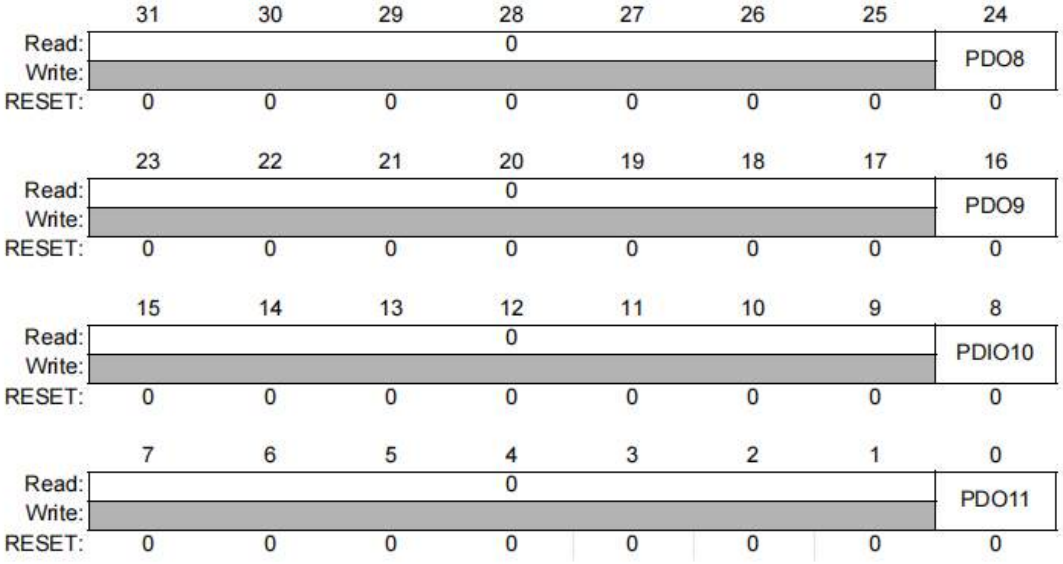


图 7-5：GPIO 引脚数据输出寄存器 8 ~ 11（SIM_GPDO8 ~ SIM_GPDO11）

偏移地址：0x040C

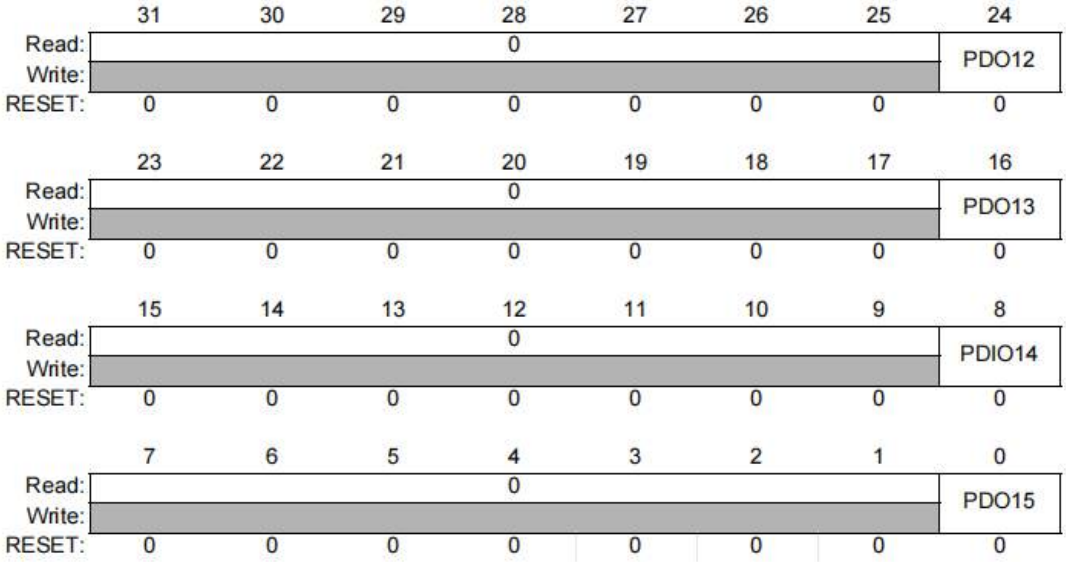


图 7-6：GPIO 引脚数据输出寄存器 12 ~ 15（SIM_GPDO12 ~ SIM_GPDO15）

偏移地址：0x0410

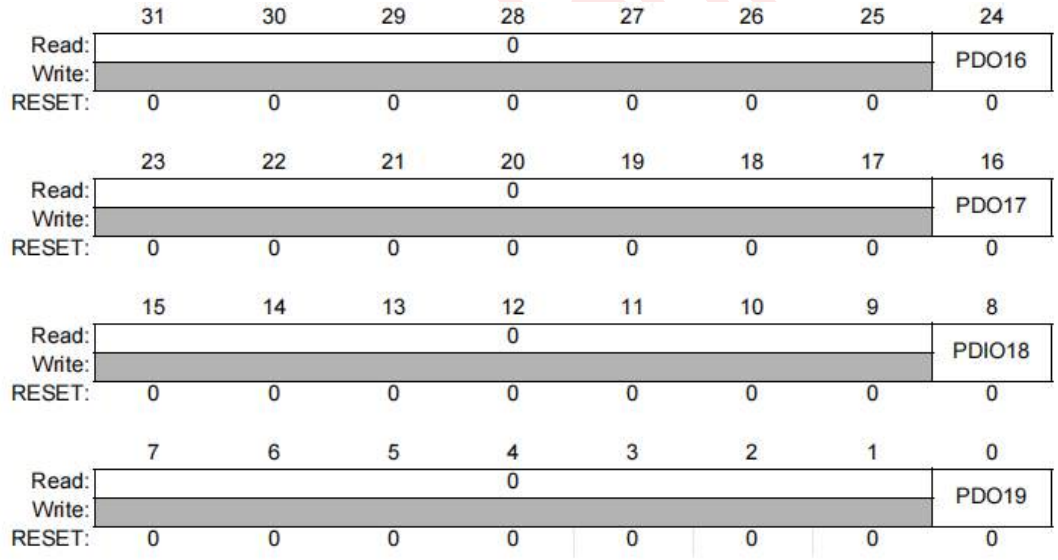


图 7-7：GPIO 引脚数据输出寄存器 16 ~ 19（SIM_GPDO16 ~ SIM_GPDO19）

偏移地址：0x0414

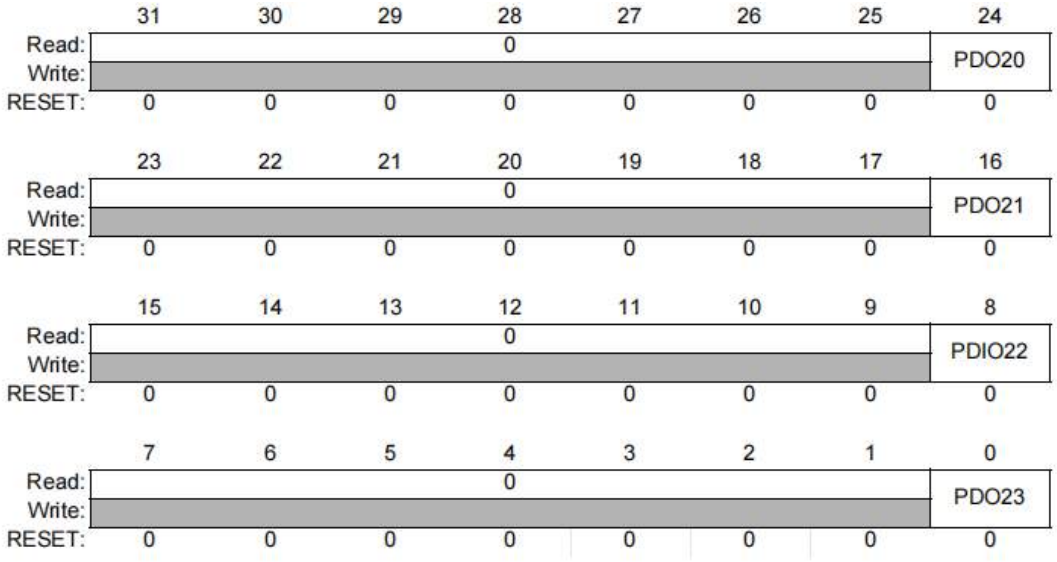


图 7-8：GPIO 引脚数据输出寄存器 20 ~ 23（SIM_GPDO20 ~ SIM_GPDO23）

偏移地址：0x0418

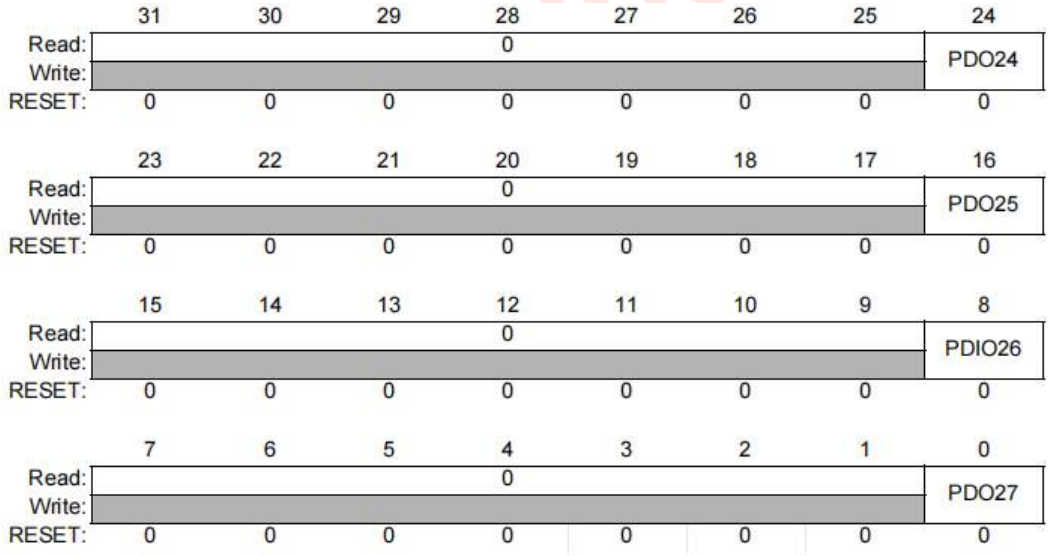


图 7-9：GPIO 引脚数据输出寄存器 24 ~ 27（SIM_GPDO24 ~ SIM_GPDO27）

偏移地址：0x041C

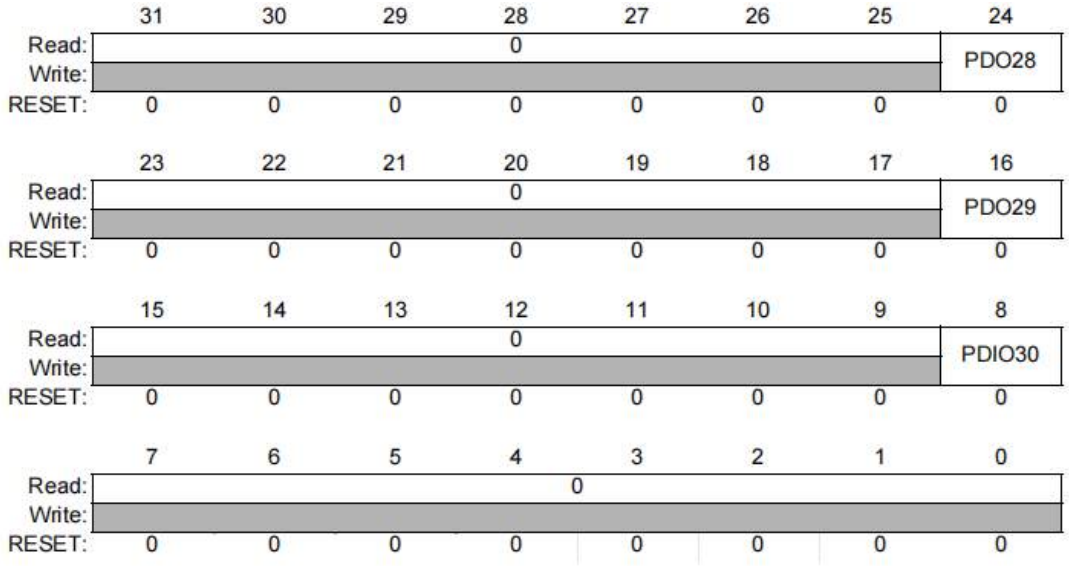


图 7-10：GPIO 引脚数据输出寄存器 28 ~ 31（SIM_GPDO28 ~ SIM_GPDO31）

表 7-5：SIM_GPDOx 寄存器字段说明

字段	描述
PDOx	引脚数据输出 该位存储要驱动到与寄存器关联的外部 GPIO 引脚上的数据。 1：当 GPIO 引脚被配置为输出时，VOH 由外部引脚驱动。 0：当外部 GPIO 引脚被配置为输出时，VOL 将被驱动。

7.5.3. GPIO 引脚数据输入寄存器（SIM_GPDI0_3 ~ SIM_GPDI_28_31）

SIM_GPDIx_x 寄存器的定义如图 7-11 至图 7-18 所示。32 个 GPDI 位（实际使用 31 位）分别对应相同 GPIO 引脚编号的引脚。例如，GPDI0 是 PWM0_GINT0[0]_GPIO0 引脚的数据输入位，而 PDI30 则是 RXD0_SDA_CAN_RX_GPIO30 引脚的数据输入位。该存储空间中存在空隙区域，表示封装中未配置该引脚。

SIM_GPDIx_x 寄存器是只读寄存器，用于让软件读取外部 GPIO 引脚的输入状态。每个寄存器字节对应一个外部 GPIO 引脚的输入状态。当 GPIO 引脚被配置为输出模式，并且在对应的引脚配置寄存器中设置了输入缓冲使能（IBE）位时，SIM_GPDIx_x 寄存器将实时反映该输出引脚的实际状态。

偏移地址：0x0600

	31	30	29	28	27	26	25	24
Read:	0							PDI0
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0							PDI1
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0							PDI2
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0							PDI3
Write:								
RESET:	0	0	0	0	0	0	0	0

图 7-11：寄存器 0 ~ 3 中的 GPIO 引脚数据（SIM_GPDI0 ~ SIM_GPDI3）

偏移地址：0x0604

	31	30	29	28	27	26	25	24
Read:	0							PDI4
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0							PDI5
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0							PDI6
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0							PDI7
Write:								
RESET:	0	0	0	0	0	0	0	0

图 7-12：寄存器 4 ~ 7 中的 GPIO 引脚数据（SIM_GPDI4 ~ SIM_GPDI7）

偏移地址：0x0608

	31	30	29	28	27	26	25	24
Read:	0							PDI8
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0							PDI9
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0							PDI10
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0							PDI11
Write:								
RESET:	0	0	0	0	0	0	0	0

图 7-13：寄存器 8 ~ 11 中的 GPIO 引脚数据（SIM_GPDI8 ~ SIM_GPDI11）

偏移地址：0x060C

	31	30	29	28	27	26	25	24
Read:	0							PDI12
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0							PDI13
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0							PDI14
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0							PDI15
Write:								
RESET:	0	0	0	0	0	0	0	0

图 7-14：寄存器 12 ~ 15 中的 GPIO 引脚数据（SIM_GPDI12 ~ SIM_GPDI15）

偏移地址：0x0610

	31	30	29	28	27	26	25	24
Read:	0							PDI16
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0							PDI17
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0							PDI18
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0							PDI19
Write:								
RESET:	0	0	0	0	0	0	0	0

图 7-15：寄存器 16 ~ 19 中的 GPIO 引脚数据（SIM_GPDI16 ~ SIM_GPDI19）

偏移地址：0x0614

	31	30	29	28	27	26	25	24
Read:	0							PDI20
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0							PDI21
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0							PDI22
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0							PDI23
Write:								
RESET:	0	0	0	0	0	0	0	0

图 7-16：寄存器 20 ~ 23 中的 GPIO 引脚数据（SIM_GPDI20 ~ SIM_GPDI23）

偏移地址：0x0618

	31	30	29	28	27	26	25	24
Read:	0							PDI24
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0							PDI25
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0							PDI26
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0							PDI27
Write:								
RESET:	0	0	0	0	0	0	0	0

图 7-17：寄存器 24 ~ 27 中的 GPIO 引脚数据（SIM_GPDI24 ~ SIM_GPDI27）

偏移地址：0x061C

	31	30	29	28	27	26	25	24
Read:	0							PDI28
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0							PDI29
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0							PDI30
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0							
Write:								
RESET:	0	0	0	0	0	0	0	0

图 7-18：寄存器 28 ~ 31 中的 GPIO 引脚数据（SIM_GPDI28 ~ SIM_GPDI31）

表 7-6：SIM_GPDIx 寄存器字段说明

字段	描述
PDIx	插入数据输入 该位将输入状态存储在与寄存器关联的外部 GPIO 引脚上。 1：引脚上的信号大于或等于 VIH。 0：引脚上的信号小于或等于 VIL。

7.5.4. 并行 GPIO 引脚数据输出寄存器（SIM_PGPDO0）

软件通过写入 PGPDOx 寄存器来驱动外部 GPIO 引脚输出数据。这些寄存器访问的 GPIO 引脚与 SIM_GPDO0 ~ SIM_GPDO31 位寄存器相同，SIM_GPDO 寄存器应直接映射到这些寄存器。例如：第 31 位的 SIM_PGPDO0 对应 SIM_GPDO0 的第 7 位，第 30 位的 SIM_PGPDO0 对应 SIM_GPDO1 的第 7 位，以此类推。图 7-19 展示了 SIM_GPDOx 与 SIM_PGPDOn 寄存器之间的对应关系对照表。

偏移地址：0x0800

	31	30	29	28	27	26	25	24
R W	PGPDO 0	PGPDO 1	PGPDO 2	PGPDO 3	PGPDO 4	PGPDO 5	PGPDO 6	PGPDO 7
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R W	PGPDO 8	PGPDO 9	PGPDO 10	PGPDO 11	PGPDO 12	PGPDO 13	PGPDO 14	PGPDO 15
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R W	PGPDO 16	PGPDO 17	PGPDO 18	PGPDO 19	PGPDO 20	PGPDO 21	PGPDO 22	PGPDO 23
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R W	PGPDO 24	PGPDO 25	PGPDO 26	PGPDO 27	PGPDO 28	PGPDO 29	PGPDO 30	PGPDO 31
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 7-19：并行 GPIO 引脚数据输出寄存器（SIM_PGPDO0）

表 7-7：SIM_PGPDO0 字段描述

字段	描述
PGPDOx	将数据引脚输出 该位存储要驱动到与寄存器关联的外部 GPIO 引脚上的数据。 1：当 GPIO 引脚被配置为输出时，VOH 由外部引脚驱动。 0：当外部 GPIO 引脚被配置为输出时，VOL 由该引脚驱动。

7.5.5. 并行 GPIO 引脚数据输入寄存器（SIM_PGPDI0）

GPDIx 寄存器是只读寄存器，用于读取外部 GPIO 引脚的输入状态。这些寄存器访问与 SIM_GPDI0 - SIM_GPDI31 位寄存器相同的 GPIO 引脚。SIM_GPDI 寄存器应直接映射到这些寄存器。有关查找表和示例，请参阅 7.5.4 节。

GPIO 读写应将逻辑地址解码为与正常 GPIO 相同的物理地址。这样，当对应 PCR 中的 IBE 被置为有效时，GPDI 和对应的 PGPDI 寄存器都应在引脚状态改变时更新。

偏移地址：0x0900

	31	30	29	28	27	26	25	24
R	PGPDI 0	PGPDI 1	PGPDI 2	PGPDI 3	PGPDI 4	PGPDI 5	PGPDI 6	PGPDI 7
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	PGPDI 8	PGPDI 9	PGPDI 10	PGPDI 11	PGPDI 12	PGPDI 13	PGPDI 14	PGPDI 15
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	PGPDI 16	PGPDI 17	PGPDI 18	PGPDI 19	PGPDI 20	PGPDI 21	PGPDI 22	PGPDI 23
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	PGPDI 24	PGPDI 25	PGPDI 26	PGPDI 27	PGPDI 28	PGPDI 29	PGPDI 30	PGPDI 31
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 7-20：并行 GPIO 引脚数据输入寄存器（SIM_PGPDI0）

表 7-8：SIM_PGPDI0 字段描述

字段	描述
PGPDIx	插入数据输入 该位将输入状态存储在与寄存器关联的外部 GPIO 引脚上。 1：引脚上的信号大于或等于 VIH。 0：引脚上的信号小于或等于 VIL。

7.5.6. 掩蔽并行 GPIO 引脚数据输出寄存器（SIM_MPGPDO0 - SIM_MPGPDO1）

MPGPDOx 寄存器由软件写入，用于通过外部 GPIO 引脚输出数据（这些寄存器是只读寄存器，读取时返回 0；必须通过访问对应的 SIM_GPDO 寄存器才能读取）。这些寄存器使用的 GPIO 引脚与 SIM_GPDO0-SIM_GPDO31 位寄存器相同。SIM_MPGPDO 寄存器的最高 16 位应直接映射到这些寄存器。例如：SIM_MPGPDO0 位 0 对应 SIM_GPDO15 位 7，SIM_MPGPDO0 位 1 对应 SIM_GPDO14 位 7，，....，SIM_MPGPD1 位 0 对应 SIM_GPDO31 位 7。最低 16 位则是 MASK 字段定义的 GPIO 引脚对应值。被屏蔽的并行 GPIO 读写操作应解码逻辑地址，使其与常规 GPIO 的物理地址保持一致。

偏移地址：0x0A00

	31	30	29	28	27	26	25	24
R								
W	MASK0	MASK1	MASK2	MASK3	MASK4	MASK5	MASK6	MASK7
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R								
W	MASK8	MASK9	MASK10	MASK11	MASK12	MASK13	MASK14	MASK15
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R								
W	DATA0	DATA1	DATA2	DATA3	DATA4	DATA5	DATA6	DATA7
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R								
W	DATA8	DATA9	DATA10	DATA11	DATA12	DATA13	DATA14	DATA15
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 7-21：掩蔽并行 GPIO 引脚数据输出寄存器（SIM_MPGPDO0）

偏移地址：0x0A04

	31	30	29	28	27	26	25	24
R								
W	MASK16	MASK17	MASK18	MASK19	MASK20	MASK21	MASK22	MASK23
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R								
W	MASK24	MASK25	MASK26	MASK27	MASK28	MASK29	MASK30	MASK31
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R								
W	DATA16	DATA17	DATA18	DATA19	DATA20	DATA21	DATA22	DATA23
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R								
W	DATA24	DATA25	DATA26	DATA27	DATA28	DATA29	DATA30	DATA31
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 7-22：掩蔽并行 GPIO 引脚数据输出寄存器（SIM_MPGPDO1）

表 7-9：SIM_MPGPDO0 ~ SIM_MPGPDO31 字段说明

字段	描述
MASK*x*	Pin Data Out。控制对相应 GPDO 的写入访问权限。 0：GPDO 定义的上一个值为“维持”。 1：对应的 GPDO 用 DATA 字段定义的值进行写入。
DATAx	当引脚被配置为输出时，Pin Data Out.将数据存储到由该寄存器控制的外部 GPIO 引脚上。 0：低电平值被驱动到对应 GPIO 引脚的 pad 接口数据输出信号上。 1：当引脚被配置为输出时，逻辑高值被驱动到对应 GPIO 引脚的 pad 接口数据 out 信号上。

7.5.7. WKUPC — 唤醒配置寄存器

偏移地址：0x0C00

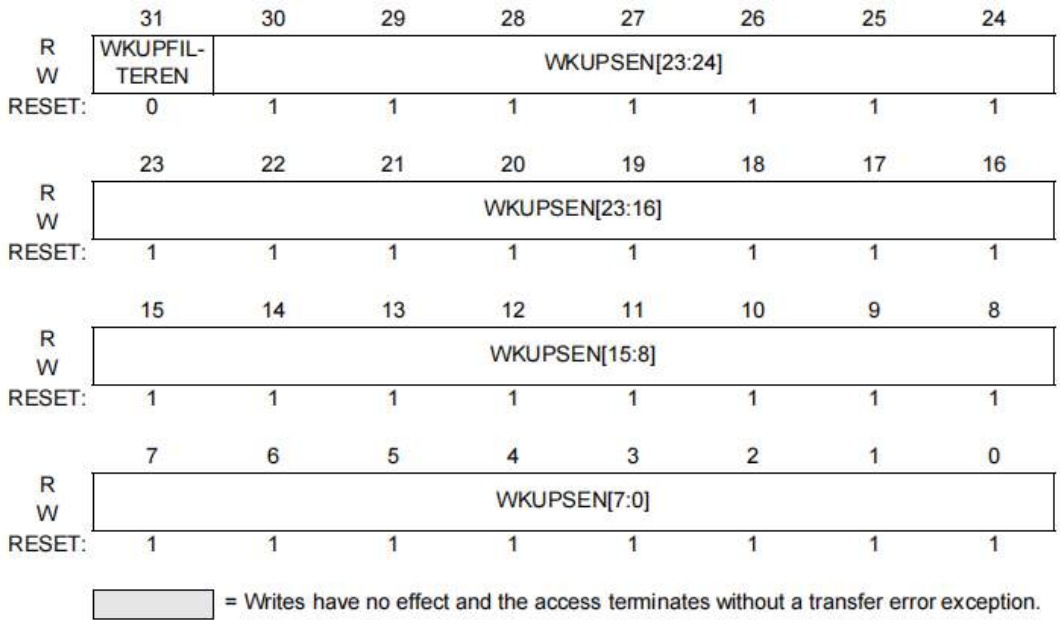


图 7-23：WKUPC — 唤醒配置寄存器

WKUPFILTEREN — 唤醒源过滤器启用

如果设置了 WKUPFILTEREN，唤醒源将通过一个滤波器来消除毛刺，然后唤醒芯片的待机模式。

1 = 启动源过滤器已启用

0 = 唤醒源过滤器已禁用

WKUPSEN — 唤醒源启用

该字段控制是否将对应的源用作唤醒芯片待机模式的源，如果设置，则将对应的源用作唤醒源。

表 7-10 显示了 WKUPSEN 和对应的唤醒源。

表 7-10：WKUPSEN 和对应的唤醒源

WKUPSEN	Wakeup Source
WKUPSEN[30]	Reserved
WKUPSEN[29]	Reserved
WKUPSEN[28]	Reserved
WKUPSEN[27]	RTC interrupt
WKUPSEN[26]	PVD interrupt
WKUPSEN[25]	Reserved
WKUPSEN[24]	Reserved
WKUPSEN[23]	COMP0 interrupt
WKUPSEN[22]	Reserved
WKUPSEN[21]	Reserved
WKUPSEN[20]	I2C
WKUPSEN[19]	WDT0 interrupt
WKUPSEN[18]	JTAG POWERON REQEST
WKUPSEN[17]	Reset# pin
WKUPSEN[16]	WDT0 reset
WKUPSEN[15]	Reserved
WKUPSEN[14]	Reserved
WKUPSEN[13]	Reserved
WKUPSEN[12]	Reserved
WKUPSEN[11]	Reserved
WKUPSEN[10]	int1[2]
WKUPSEN[9]	int1[1]
WKUPSEN[8]	int1[0]
WKUPSEN[7]	int0[7]
WKUPSEN[6]	int0[6]
WKUPSEN[5]	int0[5]
WKUPSEN[4]	int0[4]
WKUPSEN[3]	int0[3]
WKUPSEN[2]	int0[2]
WKUPSEN[1]	int0[1]
WKUPSEN[0]	int0[0]

7.5.8. QSPIXIPMCFR — QSPI XIP 模式配置寄存器

QSPIXIPMCFR 寄存器是一个可读写的寄存器。

偏移地址：0x0C04

	31	30	29	28	27	26	25	24
Read:	0	0	0	0	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0	0	0	0	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0	0	0	0	0	0	QSPI0_DATA_ENCR_EN	QSPI0_XIPEN
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0	0	0	0	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 7-24：QSPIXIPMCFR-QSPI XIP 模式配置寄存器

QSPI*_XIPEN — QSPI* XIP 模式启用控制位

1 = QSPI* XIP 模式已启用

0 = QSPI* XIP 模式已禁用

QSPI*_DATA_ENCR_EN — QSPI* XIP 传输数据加密功能启用控制位

1 = QSPI* XIP 传输数据加密功能已启用

0 = QSPI* XIP 传输数据加密功能已禁用

7.5.9. QSPILKEYR — QSPI 32 位密钥寄存器

QSPIKEYR 寄存器是一个可写寄存器，读取时总是返回 0。

偏移地址：0x0C08

	31	30	29	28	27	26	25	24
Read:	0	0	0	0	0	0	0	0
Write:	QSPI_KEY[31:24]							
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0	0	0	0	0	0	0	0
Write:	QSPI_KEY[23:16]							
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0	0	0	0	0	0	0	0
Write:	QSPI_KEY[15:8]							
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0	0	0	0	0	0	0	0
Write:	QSPI_KEY[7:0]							
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 7-25：QSPIKEYR-QSPI 32 位密钥寄存器

7.6. 功能说明

以下部分概述了 SIM 操作功能。

7.6.1. 引脚配置

SIM 卡中的引脚配置寄存器（PCR）允许软件控制外部引脚的静态电气特性。通过 PCRs 可以指定引脚的复用功能、上拉或下拉器件的选择、I/O 信号的上升速率、输出引脚的开漏模式、输入引脚的滞后效应以及总线信号的驱动强度。

7.6.2. GPIO 操作

该设备的所有 GPIO 功能均由 SIM 提供。每个具有 GPIO 功能的设备引脚在 SIM 中都有一个对应的引脚配置寄存器，其中为该引脚选择了 GPIO 功能。此外，每个具有 GPIO 功能的设备引脚都有一个输入数据寄存器（SIM_GPDIx_x）和一个输出数据寄存器（SIM_GPDOx_x）。

8. 时钟和电源控制模块（CLKM）

8.1. 介绍

时钟模块包含：

- FIRC：快速内部 150MHz RC 振荡器
- PLL：内部锁相环时钟
- FXOSC：外部高速晶体振荡器（范围为 8MHz 至 20MHz）
- SIRC：内部低速 128KHz 振荡器
- SXOSC：外部低速晶体振荡器（32768Hz）
- 状态和控制寄存器
- 时钟和电源控制逻辑

8.2. 特性

时钟模块的特性包括：

- 三个系统时钟源
  - 快速内部 150MHz RC 振荡器
  - 内部 PLL 时钟
  - 外置高速晶体振荡器（FXOSC）
- 用于 IPS、系统、触摸屏和 ADC 时钟的单个分频器
- 支持低功耗模式
- 可通过设置 MSCR 单独停止模块

8.3. 时钟结构

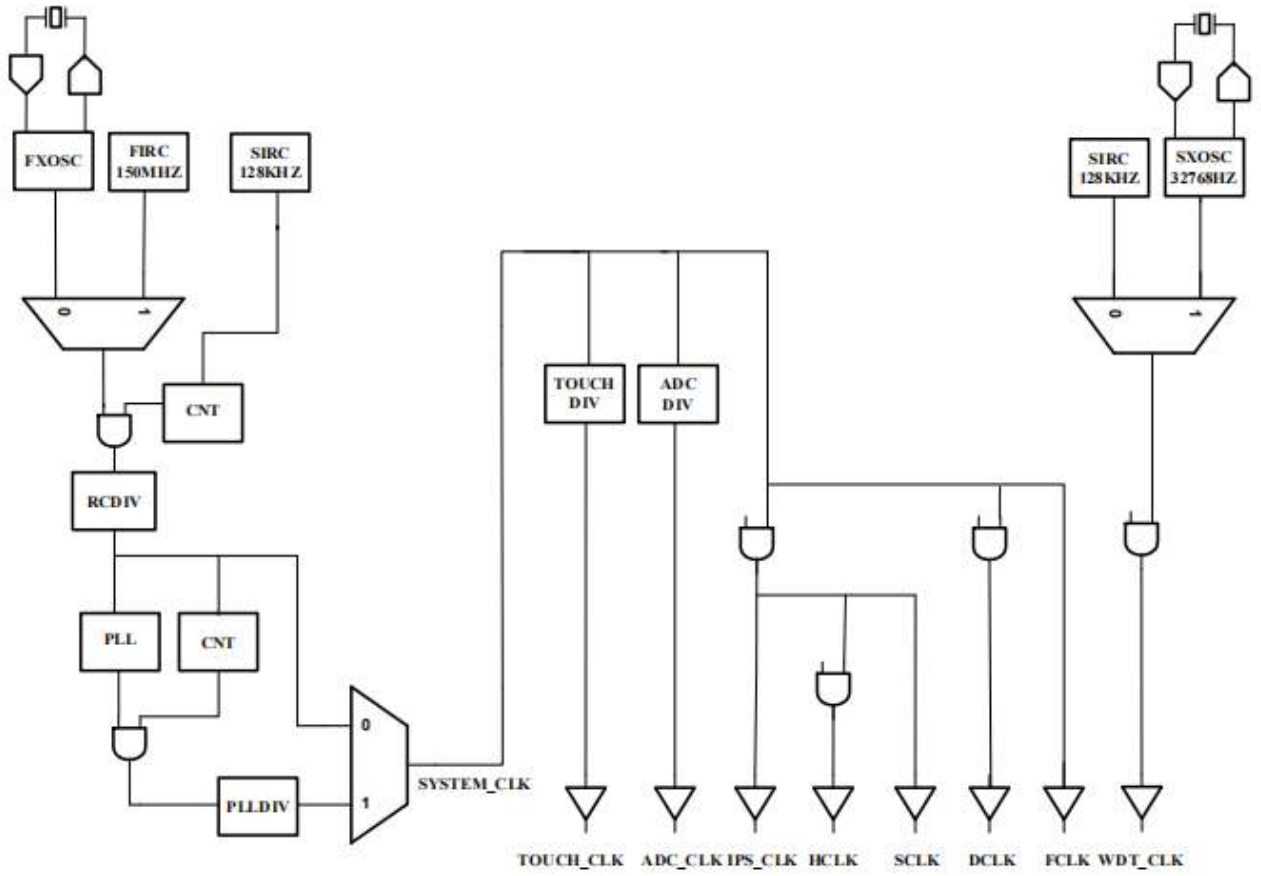


图 8-1：时钟结构

8.4. 时钟源选择

系统时钟源可以是内部 PLL 时钟、快速内部 150MHz RC 振荡器（FIRC）或外部高速晶体振荡器（FXOSC）。时钟源选择取决于 SYNCR 寄存器的 PLLEN 位。如果该位被设置，则内部 PLL 是系统时钟源；否则，系统时钟源为快速内部

150MHz RC 振荡器（FIRC）或外部高速晶体振荡器（FXOSC）由 CCR[CLKMD] 位决定。

8.4.1. 低功耗选项

8.4.1.1. 等待和打盹模式

在等待和打盹模式下，系统时钟将发送给外围设备并启用嵌入式闪存，而 CPU、ROM 和 SRAM 的时钟将停止。每个模块都可以在模块级别或通过设置 MSCR 来禁用模块时钟。

8.4.1.2. 停止模式

在停止模式下，所有系统时钟均被禁用。

提示：在停止模式下，不要对 EFLASH 进行编程或擦除。

8.5. 内存映射和寄存器

时钟编程模型由以下寄存器组成：

- 合成器控制寄存器（SYNCR）
- 低速振荡器控制寄存器（LOSCCR）
- PLL 配置和状态寄存器（PLLCSR）
- 模块停止控制寄存器（MSCR）
- EPT 外部和 CLKOUT 时钟源控制寄存器（ECCSCR）
- OSC Bist 测试配置寄存器 1（OBTCR1）
- OSC Bist 测试配置寄存器 2（OBTCR2）
- OSC Bist 测试控制寄存器（OBTCR）
- OSC BIST 测试计数器寄存器（OBTCNTR）
- OSC BIST 测试结果寄存器（OBTRR）

8.5.1. 模块内存映射

表 8-1：时钟存储器映射

地址偏移量	Bit[31:16]	Bit[15:0]	访问权限 [1]
0x0000	合成器控制寄存器（SYNCR）		S
0x0004	低速振荡器控制寄存器（LOSCCR）		S
0x0008	PLL 配置和状态寄存器（PLLCSR）		S
0x000C	模块停止控制寄存器（MSCR）		S
0x0010	EPT 外部和 CLKOUT 时钟源控制寄存器（ECCSCR）		S
0x0014	OSC Bist 测试配置寄存器 1（OBTCR1）		S
0x0018	OSC Bist 测试配置寄存器 2（OBTCR2）		S
0x001C	OSC Bist 测试控制寄存器（OBTCR）		S
0x0020	OSC BIST 测试计数器寄存器（OBTCNTR）		S
0x0024	OSC BIST 测试结果寄存器（OBTRR）		S

提示：S = 仅限主管访问。用户模式下对仅限主管访问地址位置的访问无效，并导致循环终止传输错误。

8.5.2. 寄存器说明

本小节描述时钟模块寄存器。

8.5.2.1. 合成器控制寄存器

合成器控制寄存器（SYNCR）始终处于可读/写状态。

偏移地址：0x0000

	31	30	29	28	27	26	25	24
R	SYNCTEST[1:0]		PLLDIV[3:0]				PLLOCKM	ENLOW-POWER
W								
RESET:	0	0	0	0	0	1	0	1

	23	22	21	20	19	18	17	16
R	RCDIV[5:0]						TOUCHDIV[1:0]	
W								
RESET:	0	0	Note1	0	Note1	Note1	1	0

	15	14	13	12	11	10	9	8
R	ADCDIV[3:0]				LOSCEN	PLL-SRCEN	PLLEN	SLEEP
W								
RESET:	0	0	1	0	1	0	0	0

	7	6	5	4	3	2	1	0
R	0	CLKOUT-SEL	STBYMD[1:0]		0	ADCEN	0	LOSCL-PEN
W								
RESET:	0	0	1	1	0	1	0	1

= Writes have no effect and the access terminates without a transfer error exception.

提示：1. 由 CCR[CLKMD] 位的值决定

图 8-2：合成器控制寄存器（SYNCR）

SYNCTEST[1:0] — SYNCR 写入访问序列输入

SYNCR 寄存器的可写位不能随意更改，除非按照正确的序列写入。正确的写入顺序是：2‘b01 → 2’b10 → 2‘b11。按照这个顺序写入这两个位后，这两个位的值为 2’b11，此时 SYNCR 寄存器的可写位就可以随意修改了。只有当写入 2‘b00 时，这些位才会被清零，因为此时寄存器的值等于 2’b11。其他任何写入操作都不会改变状态，只会让寄存器保持 2‘b11 的状态。

RCDIV[5:0] — FIRC150M 或 FXOSC 时钟分频器

该字段设置 FIRC150M 或 FXOSC 时钟的分频值。当 CCR[CLKMD] 位被设置时，默认值为 6‘b001011（分频 12），否则为 6’b000000（分频 1）。其他分频值参考表 8-2。

表 8-2：FIRC150M 或 FXOSC 分频器

RCDIV[5:0]	除数值
000000	除以 1
000001	除以 2
000010	除以 3
000011	除以 4
000100	除以 5
000101	除以 6
…	…
…	…
111111	除以 64

提示：当 CCR[CLKMD] 为清零时，RCDIV[5:0] 被固定为 6'h0（只读）。

PLDIV[3:0] — PLL 时钟分频器

该字段设置 PLL 时钟的分频值，默认值为 4'b0000（分频为 2）。其他分频值参考表 8-3。

表 8-3：PLL 时钟分频器

PLDIV[3:0]	除数值
0000	除以 2
0001	除以 2
0010	除以 4
0011	除以 6
…	…
1101	除以 26
1110	除以 28
1111	除以 30

提示：系统时钟频率不应大于 150MHz。

TOUCHDIV[1:0] — TOUCH 48MHz 时钟分频器

该字段设置 TOUCH 48MHz 时钟的分频值。默认值为 2'b10（除以 3）。其他分频值参考表 8-4。

表 8-4：触摸屏 48MHz 分频器

触摸屏[1:0]	除数值
00	除以 1
01	除以 2
10	除以 3
11	除以 4

ADCDIV[3:0] — ADC 时钟分频器

该字段设置 ADC 时钟的分频值，默认值为 4'b0000（分频为 1）。其他分频值参考表 8-5。

表 8-5：ADC 时钟分频器

ADCDIV[3:0]	除数值
0000	除以 1
0001	除以 2
0010	除以 3
0011	除以 4
0100	除以 5
0101	除以 6
0110	除以 7
0111	除以 8
1000	除以 9
1001	除以 10
1010	除以 11
1011	除以 12
1100	除以 13
1101	除以 14
1110	除以 15
1111	除以 16

LOSCEN — 内部低速 128KHz 振荡器使能位

1 = 启用内部低速 128KHz 振荡器

0 = 禁用内部低速 128KHz 振荡器

PLPS cen — 当 PLLEN 从 0 变更为 1 时，系统时钟是否停止。

1 = 系统时钟不会停止，系统时钟源来自 FXOSC

直到 PLL 锁定

0 = 系统时钟将停止，直至 PLL 锁定

PLLEN-PLL 已启用控制位。

1 = PLL 被启用，系统时钟源为 PLL

0 = PLL 被禁用且系统时钟源为 FXOSC SLEEP-芯片睡眠模式控

制位

设置 SLEEP 位，芯片将进入由 STBYMD[1:0] 指示的待机模式。其操作与“stop”指令相同。

提示：SLEEP 仅在 STBYMD[1] = 1‘b1 时有效。这与停止指令的情况不同，因为停止指令始终有效。

CLKOUTSEL — 时钟输出选择位

表 8-6：CLKOUTSEL 模式

CLKOUTSEL 引脚	时钟输出
0	系统时钟输出分频器
1	128KHz 时钟

STBYMD[1:0] — 睡眠操作控制位

STBYMD[1:0] 控制时钟源、系统时钟操作和睡眠模式下的 LDO 状态如表 8-7 所示。

表 8-7：睡眠模式下的睡眠操作控制位

	睡眠模式下的操作				
STBYMD	ADC 时钟	触摸时钟	系统时钟	时钟源	LDO
00	启用	启用	禁止	启用	正常的
01	禁止	禁止	禁止	启用	正常的
10	禁止	禁止	禁止	禁止	正常的
11	禁止	禁止	禁止	禁止	待机

LOSCLPEN — 内部低速 128KHz 振荡器低功耗使能

如果设置了 LOSCLPE，那么在芯片进入待机模式时，内部低速 128KHz 振荡器将停止工作。

1 = 启用内部低速 128KHz 振荡器的低功耗模式

0 = 禁用内部低速 128KHz 振荡器的低功耗模式

ADCEN — 模数转换器时钟使能位

1 = ADC 时钟使能

0 = ADC 时钟禁用

PLLOCKM — 由 PLL 宏生成的 PLL 锁定检测标志。

1 = 当 PLLEN 被设置时，PLL 宏将生成 PLL 锁。

0 = 当 PLLEN 被设置时，PLL 宏不生成 PLL 锁。

ENLOWPOWER — 启用进入低功耗模式状态位

在系统进入待机模式之前，必须确保该位被设置，否则无法成功进入低功耗模式。该位在从低功耗模式恢复后被设置，并在进入低功耗模式时由硬件清除

1 = 启用进入低功耗模式状态位

0 = 低功耗模式不允许状态位

8.5.2.2. 低速振荡器控制和状态寄存器

偏移地址：0x0004

	31	30	29	28	27	26	25	24
R	LOSCCSTEST[1:0]		0	SIRCST[4:0]				
W								
RESET:	0	0	0	1	1	1	1	1

	23	22	21	20	19	18	17	16
R	SXOSCST[15:8]							
W								
RESET:	0	1	0	1	0	0	0	0

	15	14	13	12	11	10	9	8
R	SXOSCST[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	SIRCRDY	SXOSCRDY	WDTCLKCHGDONE	0	0	0	WDTCLKSEL	SXOSCEN
W								
RESET:	1	0	1	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 8-3：低速振荡器控制和状态寄存器（LOSCCSR）

LOSCCSTEST[1:0] — 输入 LOSCCSR 写入访问序列

IOSCCSR 寄存器的可写位不能随意更改，除非按照正确的序列写入。正确的写入顺序是：2‘b01 → 2’b10 → 2‘b11。按照这个顺序写入这两个位后，这两个位的值将变为 2’b11，此时 LOSCCSR 寄存器的可写位就可以随意修改了。只有当写入 2‘b00 时，才能清除这两个位，因为此时寄存器的值等于 2’b11。其他任何写入操作都不会产生效果，只会让寄存器

保持 2'b11 的状态。

LOSCST[4:0] — 内部低速振荡器稳定时间值

内部低速振荡器（SIRC）将等待 SIRCST[4:0] 个周期的 32KHz(SIRC 除以 4）振荡器，然后在打开开关后准备好输出时钟。

SXOSCST[15:0] — 外部低速振荡器稳定时间值

外部低速振荡器（SXOSC）将等待 SXOSCST[15:0] 个周期的 128KHz（SIRC 振荡器的源）振荡器，然后在打开开关后准备好输出时钟。

SXOSCEN — SXOSC 已为 WDT 应用程序启用。

1 = 为 WDT 启用 SXOSC

0 = 禁用 WDT 的 SXOSC 晶体，然后应清除 WDTCLKSEL，否则将丢失 WDT 时钟。

WDTCLKSEL — WDT 时钟选择控制位。

建议您应先清除该位，然后在将 WDT 时钟源从 SXOSC 更改为 SIRC 时关闭 SXOSC。

1 = WDT 时钟源为 SXOSC（32.768KHz）

0 = WDT 时钟源为 SIRC128KHz

SIRCRDY — 内部低速振荡器（SIRC）就绪标志。

1 = 内部低速振荡器（SIRC）就绪

0 = 内部低速振荡器（SIRC）未就绪

SXOSCRDY-外部低速振荡器（SXOSC）已准备好用于 WDT 应用。

1 = 外部低速振荡器（SXOSC）就绪

0 = 外部低速振荡器（SXOSC）未就绪

WDTCLKCHGDONE — WDT 时钟切换完成标志。当 WDTCLKSEL 发生改变时，该位将变为低电平；当 WDTCLK 切换完成时，该位将被置为高电平。

1 = WDT 时钟开关完成，WDT 可以正常工作

0 = WDT 时钟正在切换，WDT 无法正常工作

8.5.2.3. PLL 配置和状态寄存器

偏移地址：0x0008

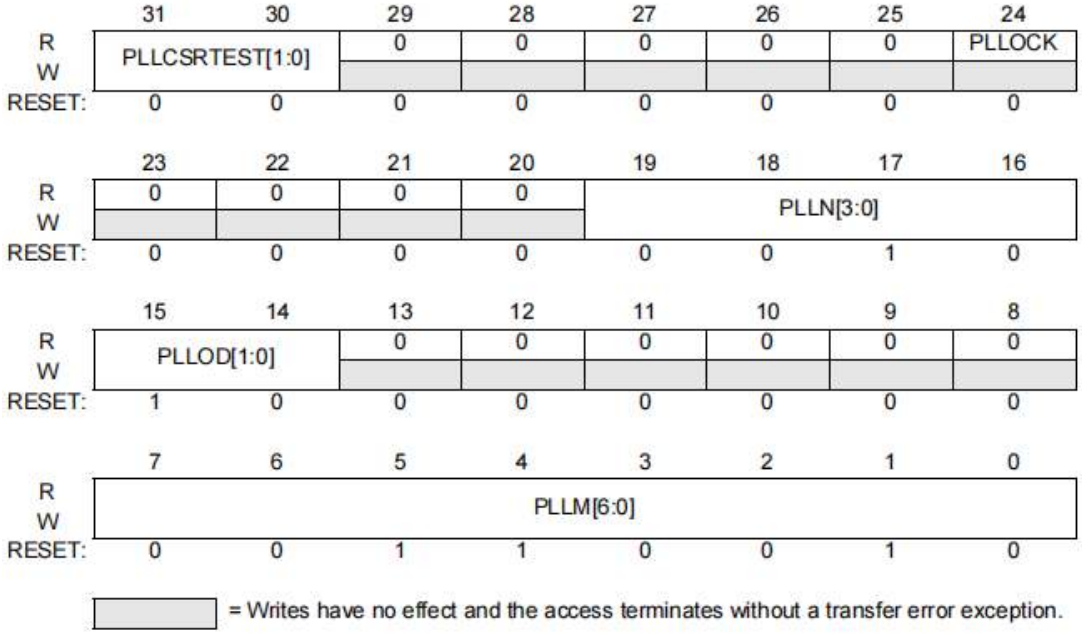


图 8-4：PLL 配置和状态寄存器（PLLCSR）

PLLCSRTEST[1:0] — PLLCSR 写入访问序列输入

PLLCSR 寄存器的可写位不能随意更改，除非按照正确的二进制序列写入。正确的写入顺序是：2'b01 → 2'b10 → 2'b11。按照这个顺序写入这两个位后，这两个位的值将变为 2'b11，此时 PLLCSR 寄存器的可写位就可以随意更改了。只有当写入 2'b00 时，这些位才会被清零，因为此时寄存器的值等于 2'b11。其他任何写入操作都不会产生效果，只会使寄存器保持 2'b11 的状态。

$$\text{PLL 输出频率} = XIN * \frac{M}{N} * \frac{1}{NO}$$

提示：使用时请遵守以下条件：

1. 3.5MHz < = XIN/N < = 35MHz
2. 200MHz < = XIN*M/N < = 400MHz
3. M≥4
4. N≥1

PLLM[6:0] — 反馈 7 位分频器控制位

该字段设置 PLL 反馈分频值，默认值为 7 ‘h32，具体分频值参考表 8-8。

表 8-8：PLL 反馈分频器值

PLLM[7:0]	M
7’ b000_0000	0
7’ b000_0001	1
7’ b000_0010	2
7’ b000_0011	3
7’ b000_0100	4
7’ b000_0101	5
…	…
…	…
7’ b111_1111	127

PLLOD[1:0] — 输出分频器控制位

该字段设置 PLL VCO 时钟的输出分频值，默认值为 2‘b10，分频值详见表 8-9。

表 8-9：PLL VCO 输出时钟分频器

PLLOD[1:0]	NO
00	除以 1
01	除以 2
10	除以 4
11	除以 8

PLLN[3:0] — 输入 4 位分频器控制位

该字段设置 PLL 时钟的输入分频值，默认值为 4‘b0010，具体输入分频值参考表 8-10。

表 8-10：PLL 输入分频器

PLLN[3:0]	N
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

PLLOCK — PLL 锁定标志

在启用 PLL 之后，PLLOCKCR[PLLST] 的周期结束后将设置此标志。

1 = PLL 已锁定

0 = PLL 未锁定

8.5.2.4. 模块停止控制寄存器

模块停止控制寄存器（MSCR）可进行永久性读写。

偏移地址：0x000C

	31	30	29	28	27	26	25	24
R W	MSCRTEST[1:0]		MS[29]	MS[28]	MS[27]	MS[26]	MS[25]	MS[24]
RESET:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R W	MS[23]	MS[22]	MS[21]	MS[20]	MS[19]	MS[18]	MS[17]	MS[16]
RESET:	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R W	MS[15]	MS[14]	MS[13]	MS[12]	MS[11]	MS[10]	MS[9]	MS[8]
RESET:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	MS[7]	MS[6]	MS[5]	MS[4]	MS[3]	MS[2]	MS[1]	MS[0]
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 8-5：模块停止控制寄存器（MSCR）

MSCRTEST[1:0] — 输入 MSCR 写入访问序列

MSCR 寄存器的可写位不能随意更改，除非按照正确的序列写入。正确的写入顺序是：2‘b01 → 2’b10 → 2‘b11。按照这个顺序写入这两个位后，这两个位的值为 2’b11，此时 MSCR 寄存器的可写位就可以随意更改了。只有当写入 2‘b00 时，这些位才会被清零，因为此时寄存器的值等于 2’b11。其他任何写入操作都不会改变状态，只会让寄存器保持 2‘b11 的状态。

MS[29:0] — 模块停止位

MS[25:0] 位禁用顶层模块的时钟（参见表 8-11 MS[29:0] 位对应模块）。

1 = 模块时钟禁用

0 = 时钟模块已启用

表 8-11：MS[29:0] 位对应模块

MS 位	对应模块
0	Reserved
1	COMP0
2	Reserved
3	ADC
4	PIT0
5	PIT1
6	Reserved
7	Reserved
8	RTC
9	DMA
10	PWM0
11	Reserved
12	EPORT0
13	EPORT1
14	XBAR
15	OPTION
16	RESET
17	WDT
18	SCI0
19	SIM
20	I2C
21	SCI1
22	SCI2
23	CAN
24	Reserved
25	QSPI0
26	SPI0
27	SPI1
28	Reserved
29	TOUCH

8.5.2.5. EPT 外部时钟源和 CLKOUT 时钟源控制寄存器

偏移地址：0x0010

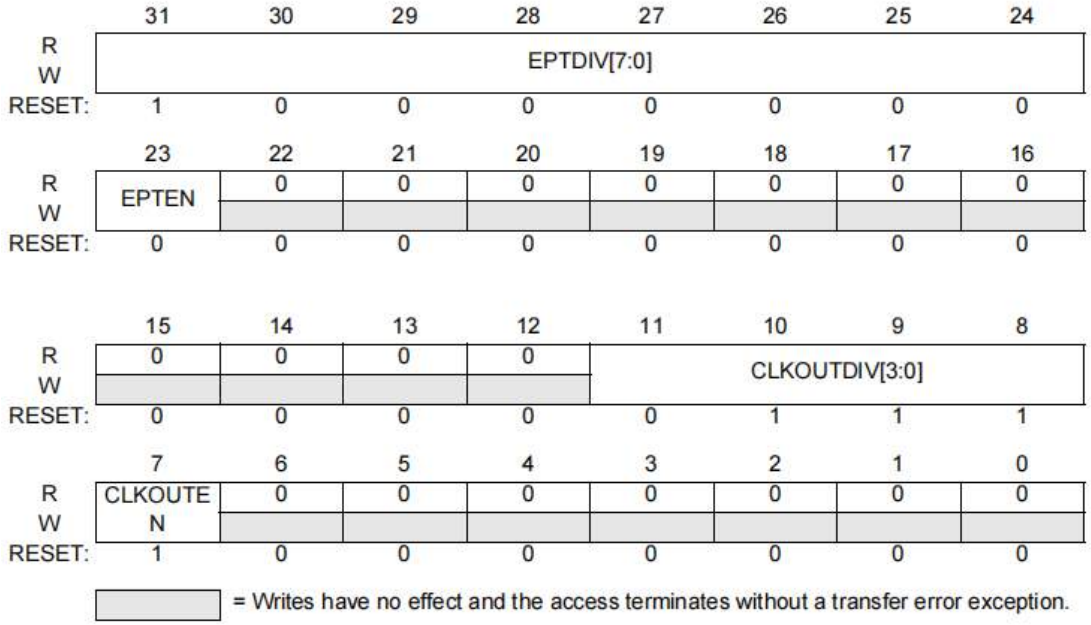


图 8-6：EPT 外部和 CLKOUT 时钟源控制寄存器（ECCSCR）

EPTDIV[5:0] — EPT 时钟分频器

该字段设置 EPT 时钟的分频值。默认值为 8 'h80。其他分频值参考表 8-12

表 8-12：EPT 时钟分频器

EPTDIV[8:0]	除数值
00000000	除以 1
00000001	除以 2
00000010	除以 3
00000011	除以 4
00000100	除以 5
00000101	除以 6
…	…
…	…
…	…
11111110	除以 255
11111111	除以 256

EPTEN — EPT 时钟使能位

如果 EPTEN 位被设置，则 EPT 时钟将从系统时钟进行分频。

1 = EPT 时钟使能

0 = EPT 时钟禁用

CLKOUTDIV[3:0] — 时钟输出分频器

该字段为 CLKOUT 设置系统分频值。默认值为 4'h8。其他分频值参考表 8-13

表 8-13：CLKOUT 分频器

CLKOUTDIV[3:0]	除数值
0000	除以 2
0001	除以 4
0010	除以 6
...	...
1111	除以 32

CLKOUTEN — CLKOUT 使能位

如果设置了 CLKOUTEN 位，则系统时钟分频器将被启用，用于 CLKOUT。

1 = 将为 CLKOUT 启用系统时钟分频器

0 = 时钟输出将禁用系统时钟分频器

8.5.2.6. OSC Bist 测试配置寄存器 1

偏移地址：0x0014

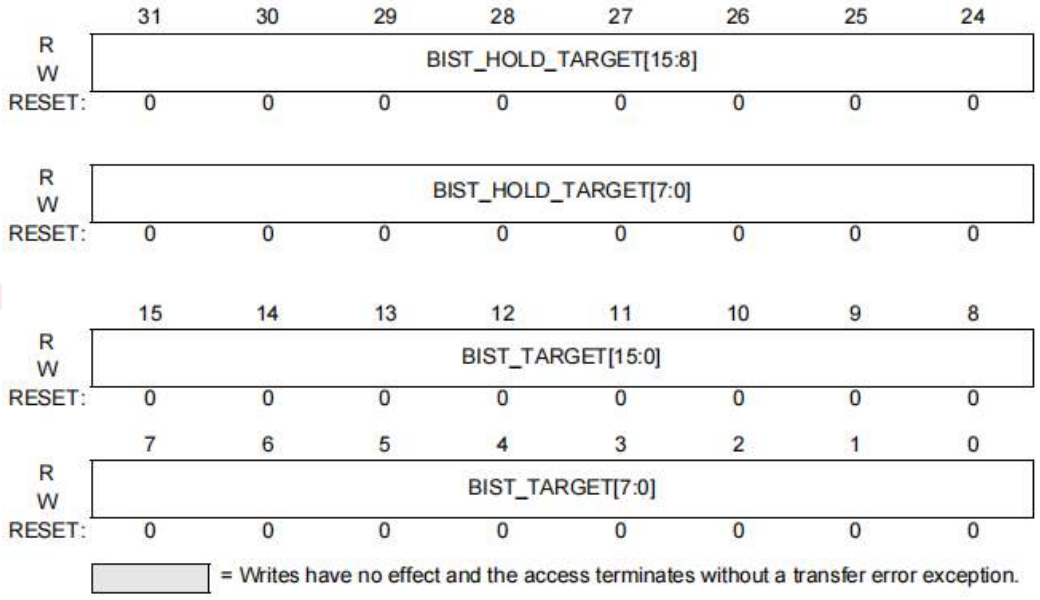


图 8-7：OSC Bist 测试配置寄存器 1（OTCR1）

BISTHoldTarget — BIST 时钟保持计数目标值

等待经过调节值更改后的受试 CLK 稳定

BIST_TARGET — BIST 时钟计数目标值

8.5.2.7. OSC Bist 测试配置寄存器 2

偏移地址：0x0018

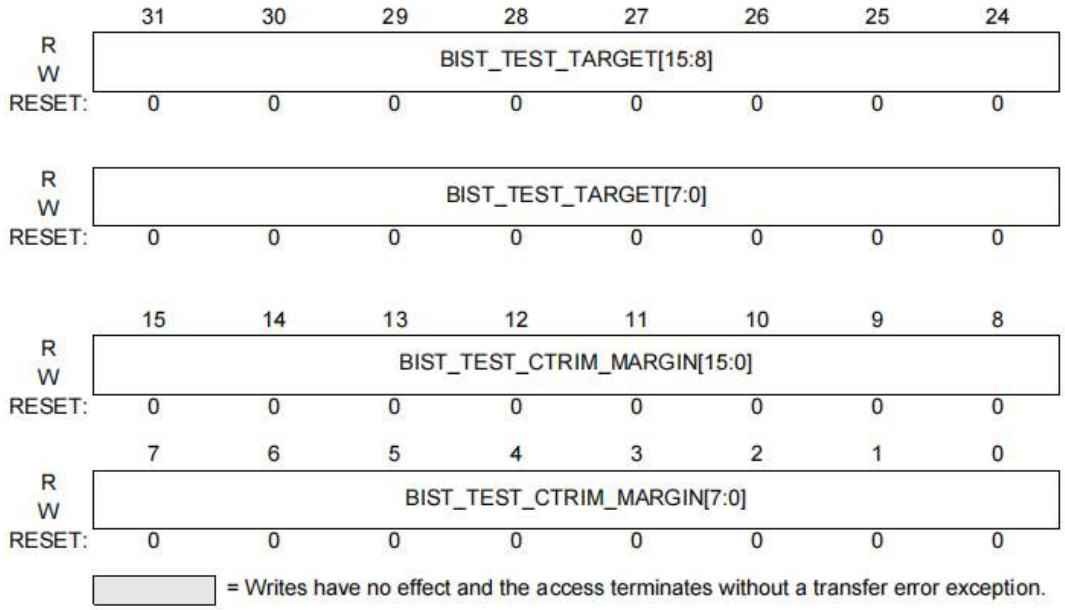


图 8-8：OSC 双测试配置寄存器 2（OTCR2）

BIST_TEST_TARGET — 被测时钟计数目标值

BIST_TEST_CTRIMMargin — 待测设备的粗调电平下的点击计数误差范围

8.5.2.8. OSC Bist 测试控制寄存器

偏移地址：0x001C

	31	30	29	28	27	26	25	24
R W	BIST_TEST_FTRIM_MARGIN[15:8]							
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R W	BIST_TEST_FTRIM_MARGIN[7:0]							
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R W	BIST_START	0	BIST_RESETN	BIST_MODE	BIST_IRC_EN	BIST_IRC_SEL[1:0]		BIST_EN
RESET:	0	0	0	1	0	1	0	0

	7	6	5	4	3	2	1	0
R W	0	0	0	0	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 8-9：OSC Bist 测试控制寄存器（OBTCR）

BIST_TEST_FTRIM_MARGIN — 待测的精细调整时钟计数误差范围

BIST Mode — BIST 模式

1 = 跟踪模式（用于测量 PLL、FIRC 或 128K 时钟的频率）
0 = 剪辑模式

BIST_START — BIST 启动

1 = 开始
0 = 停止

BIST_RESETN — Bist 复位否定信号

1 = 非 Bist 复位
0 = 确认、断开、复位

BIST_IRC_EN — 当 bist_en 被设置时的 FIRC 控制位

1 = FIRC 时钟已启用
0 = FIRC 时钟已禁用

BIST_IRC_SEL — FIRC、PLL 或 128KHz 时钟选择

BIST_IRC_SEL[1:0]	描述
2’ b00	128KHz
2’ b01	128KHz
2’ b10	FIRC150MHz
2’ b11	PLL

BIST_EN — 参考时钟使能

1 = 启用

0 = 关闭

8.5.2.9. OSC BIST 测试计数器寄存器

偏移地址：0x0020

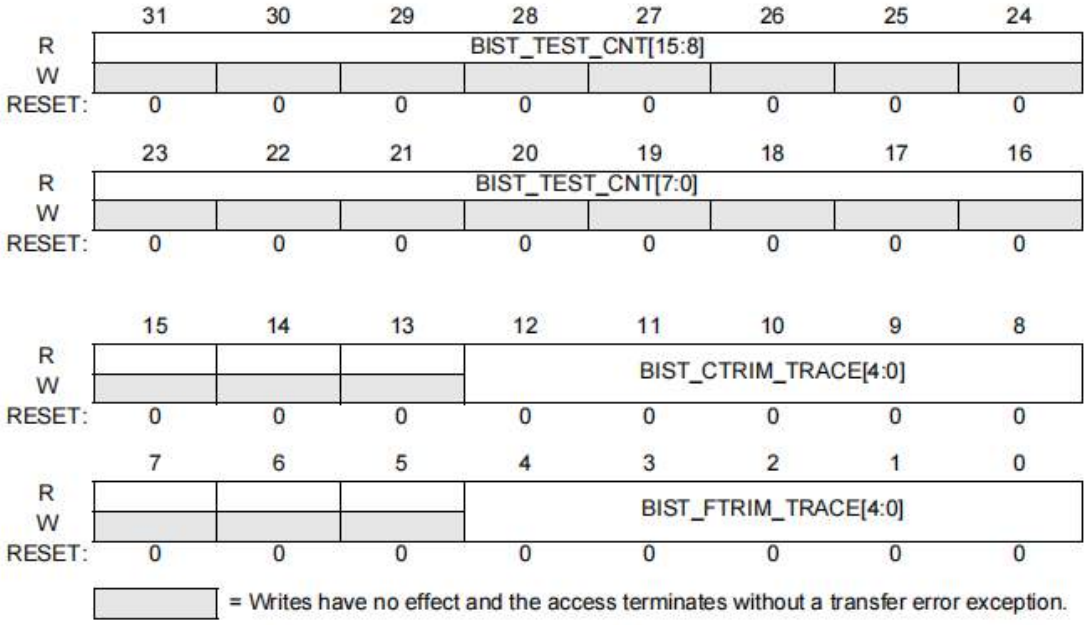


图 8-10：OSC BIST 测试计数器寄存器（OBTCNTR）

BIST_CTRIM_TRACE — BIST Ctrim 跟踪值

BIST_FTRIM_TRACE — BIST Ftrim 跟踪值

BIST_TEST_CNT — BIST 测试计数器值

8.5.2.10. OSC BIST 测试结果寄存器

偏移地址：0x0024

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	BIST_DONE	BIST_PASS
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	BISTRE-SULT[16]
W								
RESET:	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8
R	BISTRESULT[15:8]							
W								
RESET:	1	0	0	0	0	1	1	1

	7	6	5	4	3	2	1	0
R	BISTRESULT[7:0]							
W								
RESET:	0	1	1	1	1	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

图 8-11：OSC BIST 测试结果寄存器（OBTRR）

BIST_DONE — BIST 完成

1 = 完成
0 = 未完成

BIST Pass — BIST 通行证

1 = 通过
0 = 失败

BISTRESULT — BIST 修剪结果有效

当 bistDone = 1 且 bistPass = 1 时

8.6. 功能说明

8.6.1. 打开 PLL

建议执行以下步骤：

1. 等待 SYNCR[ENLOWPOWER] = 1‘b1；
2. 首先将 SYNCR[PLLEN] = 1‘b0 并关闭 PLL。
3. 配置 SYNCR[PLLSRCEN] = 1‘b0；
4. 根据目标 VCO 频率配置 PLLCSR[PLLOD]/PLLCSR[PLLN]/PLLCSR[PLLM]。
5. 配置 SYNCR[PLLEN] 和打开 PLL。
6. 等待 SYNCR[PLLOCK] 状态，系统时钟将被更改为 PLL 时钟。
7. 根据目标系统频率配置 SYNCR[PLLDIV]。

8.6.2. PLL 测量频率

建议执行以下步骤：

1. 将 BIST_RESETN 位设置为 0 以断言复位。
2. 设置 BIST_TARGET 计数器值。
3. 将 BIST_MODE 位设置为 1。
4. 将 IRC_BIST_SEL 位设置为 1，用于 PLL 测量。
5. 将 BIST START 位设置为 1。
6. 将 BIST_RESETN 位设置为 1，以实现否定复位。
7. 等待 BIST_DONE 位。
8. 读取 BIST_TEST_CNT 以测量 PLL 时钟的频率。
9. 根据 BIST_TEST_CNT 和 fxosc 时钟频率计算 PLL 时钟频率。

PLL 时钟频率的计算如下：

fPLL = 2 * ffxosc * BIST_TEST_CNT[15:0] / BIST_TARGET[15:0]

8.6.3. 128KHz 测量的频率

建议执行以下步骤：

1. 将 BIST_RESETN 位设置为 0 以断言复位。
2. 设置 BIST_TARGET 计数器值。
3. 将 BIST_MODE 位设置为 1。
4. 将 IRC_BIST_SEL 位设置为 0，以测量 128KHz 的时钟。
5. 将 BIST START 位设置为 1。
6. 将 BIST_RESETN 位设置为 1，以实现否定复位。
7. 等待 BIST_DONE 位。
8. 阅读 BIST_TEST_CNT 以测量 128KHz 时钟的频率。
9. 根据 BIST_TEST_CNT 和 fxosc 时钟频率计算 128KHz 时钟的频率。

128KHz 的频率计算如下：

$$f_{128KHz} = f_{fxosc} * BIST\_TEST\_CNT[15:0] / BIST\_TARGET[15:0]$$

9. 重置控制器模块（RCM）

9.1. 介绍

重置控制器用于确定重置的原因，向系统断言适当的重置信号，并且随后保存导致重置的原因的历史记录。

9.2. 特性

模块特性包括：

- 重置的四个来源：
  - 通电复位
  - 外部复位引脚
  - 软件重置
  - 看门狗定时器复位
  - 可编程电压检测复位
- 软件可读状态标志，指示上次复位的原因

9.3. 方块图

图 9-1 展示了复位控制器。

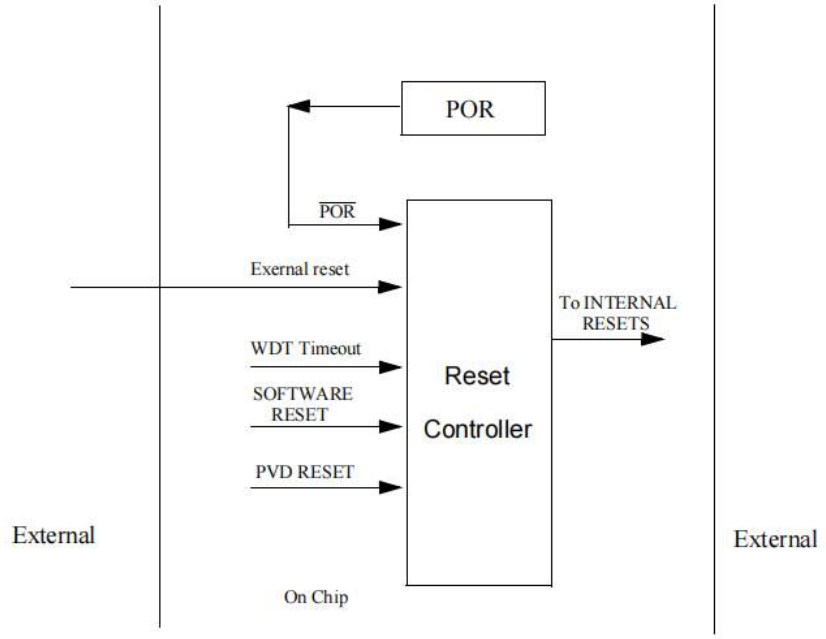


图 9-1：复位控制器方块图

9.4. 内存映射和寄存器

复位控制器编程模型由以下寄存器组成：

- 复位控制寄存器(RCR) — 选择复位控制器功能
- 复位状态寄存器(RSR) — 反映最后一个复位源的状态

地址映射请参见表 9-1，寄存器说明请参见下文。

表 9-1：复位控制器地址映射

地址偏移量	Bit[7:0]	访问权限 [1]
0x0000	已反转	S/U
0x0001	RTR — 复位测试寄存器	S/U
0x0002	RSR — 复位状态寄存器	S/U
0x0003	RCR — 复位控制寄存器	S/U

提示：S/U = 主管或用户模式访问。

9.4.1. 重置测试记录器

重置测试寄存器（RTR）仅用于工厂测试。

偏移地址：0x0001

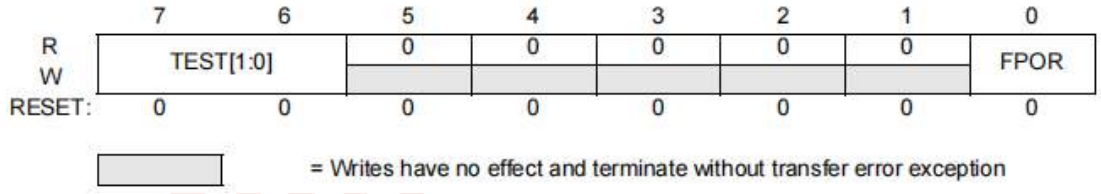


图 9-2：复位测试寄存器（RTR）

测试[1:0] — RTR 写入访问序列已启用

FPOR 寄存器的可写位无法直接修改，除非按照正确的二进制序列写入。正确的写入顺序是：2‘b01 → 2’b10 → 2‘b11。按照这个顺序写入这两个位后，这两个位的值将变为 2’b11，此时 FPOR 寄存器的可写位就可以自由修改了。只有当写入 2‘b00 时，才能清除这两个位，因为此时寄存器的值等于 2’b11。其他任何写入操作都不会改变状态，只会使寄存器保持 2‘b11 的值。

FPOR — 强制上电复位

向 FPOR 位写入 1 将导致系统上电复位。复位将导致芯片再次进行校准。

9.4.2. 重置状态寄存器

复位状态寄存器（RSR）为每个复位源都设有对应的状态位。当系统进入复位状态时，系统会锁存当前复位原因，并将其他未待处理的复位源值设为 0。这些数值随后会被写入 RSR 寄存器。多个状态位可能同时被置位。此外，该寄存器还会记录后续任一复位的原因，覆盖前次复位状态的原有设置。

RSR 可以随时读取。对 RSR 的写入操作不会产生任何影响。

偏移地址：0x0002

	7	6	5	4	3	2	1	0
R	ERST	PVD	SOFT	0	POR	0	0	0
W								
RESET:	0	Reset Dependant						

= Writes have no effect and terminate without transfer error exception

图 9-3：复位状态寄存器（RSR）

ERST — 外部复位

此位表示最后的复位状态是由外部复位引起的。

1 = 上次复位状态是由外部复位引起的

0 = 上次复位状态不是由外部复位引起的

PVD — 可编程电压检测器

此位表示最后的复位状态是由 PVD 复位引起的。

1 = 上次复位状态是由 PVD 复位引起的

0 = 上次复位状态不是由 PVD 复位引起的

SOFT — 软件复位标志

此位表示最后的复位状态是由软件引起的。

1 = 上次重置状态是由软件引起的。

0 = 最后一次重置状态不是由软件引起的。

POR — 上电复位标志

此位表示最后的复位状态是由上电或 WDT 复位引起的

1 = 上次复位状态是由通电复位引起的。

0 = 上次复位状态不是由上电复位引起的。

9.4.3. 重置控制寄存器

复位控制寄存器（RCR）允许软件控制请求复位。

偏移地址：0x0003

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	FRCR-STOUT
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and terminate without transfer error exception

图 9-4：复位控制寄存器（RCR）

FRCRSTOUT — Force RSTOUT 引脚

FRCRSTOUT 位允许软件断言或否定外部 RSTOUT 引脚。

1 = 确认 RSTOUT 引脚

0 = 非 RSTOUT 引脚

9.5. 功能说明

9.5.1. 重置源

表 9-2 定义了复位源和复位控制器驱动的信号。

表 9-2：复位源总结

来源	类型
POR	异步
ERST	异步
Watchdog timer	异步
Software	同步
PVD	异步

为确保数据完整性，同步复位源不会被复位控制逻辑触发，直至当前总线周期结束。系统时钟在周期结束后下一个上升沿才会激活复位信号。当复位控制逻辑需要将复位操作与总线周期结束同步时，内部总线监控功能会自动启用。

异步复位源通常表示灾难性故障。因此，复位控制逻辑不会等待当前总线周期完成。系统会立即被置为复位状态。

9.5.1.1. 通电复位（POR）

在通电时，复位控制器会断言系统复位。系统复位将持续保持，直到 POR 达到最低可接受水平。

9.5.1.2. 看门狗定时器重置

看门狗定时器超时导致定时器重置请求被识别并锁存。

9.5.1.3. 软件重置

如果 RISC Core 嵌入式中断控制器（EIC）中的 SYSRESTEQ 位被设置，则将生成软件复位。复位控制器将在大约 2048 个周期内断言系统复位。然后，部件退出复位并恢复操作。

9.5.1.4. 可编程电压检测复位

当嵌入式闪存模块（EFM）中的 CCR 寄存器的 PVDRE 位被设置时，PVD 将在 VDD 超过 PVD 阈值时生成复位。

9.5.2. 重置控制流

复位逻辑控制流程如图 9-5 所示。给出的所有周期计数均为近似值。

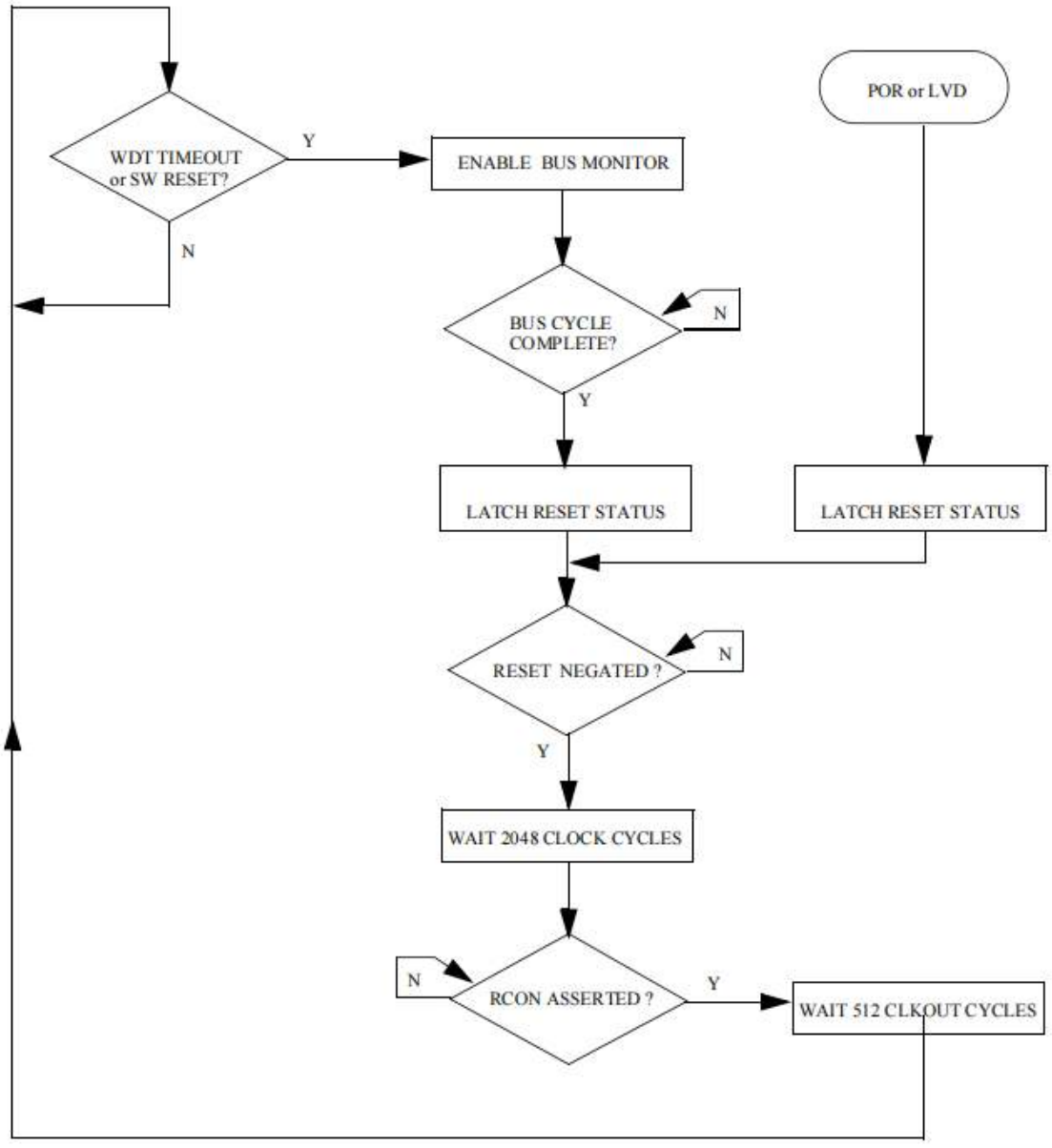


图 9-5：复位控制流程

10. 静态随机存取存储器（SRAM）

10.1. 介绍

静态随机存取存储器（SRAM）的特征包括：

- 芯片上 32KB SRAM
- 固定地址空间
- 字节、半字（16 位）或字（32 位）读/写访问
- 每次访问时钟一次（包括字节、半字和字）
- 监视器或用户模式访问

10.2. 操作模式

对 SRAM 的访问没有任何限制。可以在管理员和用户模式下访问该阵列。

10.3. 低功耗模式

在等待、12 点和停止模式下，SRAM 的时钟被禁用。退出这些模式时不需要恢复时间。

10.4. 重置操作

SRAM 内容在通电复位后立即未定义。系统复位不影响 SRAM 内容。如果在读取或写入存取期间发生同步复位，则存取正常完成，且正在进行的任何流水线存取停止，SRAM 内容未损坏。

10.5. 中断

SRAM 模块不产生中断请求。

11. 缓存模块（CACHEM）

11.1. 介绍

缓存模块为处理器提供与之紧密耦合的处理器本地存储器以及到 EBI 和 QSPI0 存储空间的总线路径。

高速缓存是一种高速存储器，包含地址信息（通常称为标签）和相关数据。其目的是减少内存访问的平均时间。缓存基于两个局部性原则：

- 空间局部性 — 访问一个位置后，很可能接着访问相邻的位置（例如，顺序执行指令或使用数据结构）。
- 时间局部性 — 对内存区域的访问很可能在短时间内重复（例如，执行代码循环）。

为了尽量减少存储的控制信息的数量，空间局部性特性被用来将几个位置组合到同一标签下。这个逻辑块通常被称为缓存行。

当数据被加载到高速缓存中时，可以减少后续加载和存储的访问时间，从而获得整体性能优势。对已经存在于高速缓存中的信息的访问称为高速缓存命中，而其他访问则称为高速缓存未命中。

通常，高速缓存是自我管理的，更新是自动发生的。每当处理器想要访问一个可缓存的位置时，就会检查是否命中高速缓存。如果访问是命中，就立即进行访问。否则，分配一个位置，并从内存中加载高速缓存行。不同的高速缓存拓扑和访问策略是可能的。

但是，它们必须遵循底层体系结构的内存一致性模型。缓存会引入许多潜在问题，主要原因在于：

- 在程序员通常预期的时间之外发生的内存访问；
- 存储数据项的物理位置可以是多个。

缓存控制器的目标是与任何需要缓存功能的基于 32 位 AHB 总线的应用程序一起使用。此缓存功能可通过快速访问最近使用的代码或数据来提高性能。

本地存储控制器支持三种操作模式：

1. Write-through — 使用此缓存模式访问地址空间时，这些地址空间可被缓存。
   - 输入总线上的直写式读取缺失将导致在输出总线上读取一个包含目标地址的 16 字节对齐内存地址。该缺失数据将被加载到缓存中，并标记为有效且未被修改。
   - 对有效缓存位置的直写式读取命中将从缓存中返回数据，而无需访问输出总线。
   - write-through 写入缺失绕过高速缓存，将数据写入输出总线（对于 write-through 模式空间，不采用写入缺失策略进行分配）。
   - write-through 写入命中更新缓存命中数据，并将数据写入输出总线。
2. 写回 — 使用此缓存模式访问地址空间时，可进行缓存。

- 输入总线上的写回读取未命中将导致在输出总线上读取一个包含目标地址的 16 字节对齐内存地址。此未命中数据被加载到缓存中，并标记为有效且未被修改。
- 对有效缓存位置的写回读取命中将返回来自缓存的数据，而无需访问输出总线。
- 写回模式下的写未命中会执行“读取到写入”操作（采用写回模式空间的写未命中策略进行分配）。系统会在输出总线上读取一个包含目标写地址的 16 字节对齐内存地址。该未命中数据将被加载至缓存并标记为有效且已修改，随后写入数据将更新相应的缓存数据位置。

3. 不可缓存 — 使用此缓存模式访问地址空间时不可缓存。
   - 这些访问绕过了缓存并访问输出总线。

高速缓存控制器有两个 AHB 总线接口，一个为主接口，一个为从接口。主接口具有解码逻辑，用于确定有效地址阶段访问的缓存模式。主访问将根据其缓存模式访问或绕过缓存。从接口用于缓存未命中以及绕过缓存的访问。

该缓存控制器采用双路组相联结构。该缓存具有 32 位宽的地址和数据路径，以及 16 字节的行大小。缓存标签和数据存储采用单端口同步 RAM。控制器配备多种读写数据缓冲器以提升性能。缓存未命中和行推送会产生 4 拍 32 位环绕突发访问，优先访问关键字以实现最佳性能。

11.2. 方块图

此缓存模块提供对 RAM 和可缓存地址空间的零等待状态访问。

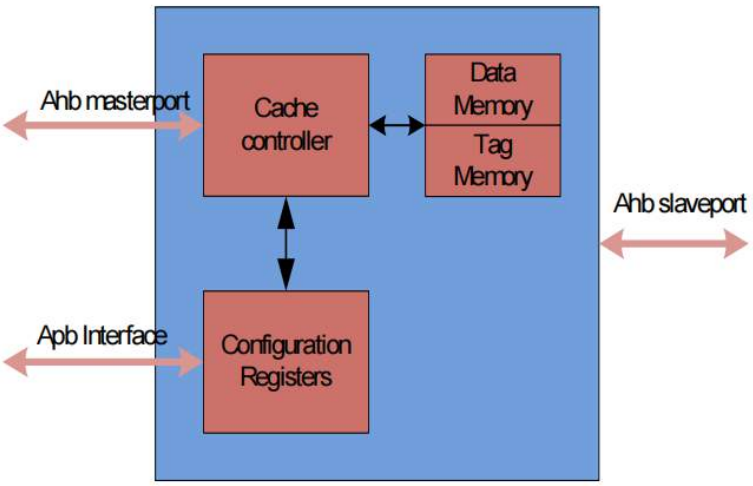


图 11-1：缓存模块方块图

11.3. 内存映射和寄存器

表 11-1：缓存模块内存映射

地址偏移量	Bit[31:24]	Bit[23:16]	Bit[15:8]	Bit[7:0]	访问权限 [1]
0x0000	缓存控制寄存器（LMEM_CCR）				S/U
0x0004	缓存行控制寄存器（LMEM_CLCR）				S/U
0x0008	缓存搜索地址寄存器（LMEM_CSAR）				S/U
0x000C	缓存读写值寄存器（LMEM_CCVR）				S/U
0x0020	缓存访问寄存器（LMEM_ACR）				S/U
0x0180	缓存页失效起始地址				S/U
0x0184	缓存页面无效大小				S/U
0x0188	缓存锁存器门				S/U

提示：

1. S = 仅允许访问 CPU 的监督器模式，U = 仅允许访问 CPU 的用户模式
2. 对仅限于监控程序访问的地址位置进行用户模式访问不会产生任何影响，并且将导致循环终止传输错误。

11.4. 寄存器说明

本小节描述了缓存模块。

11.4.1. 缓存控制寄存器（LMEM_CCR）

偏移地址：0x0000

	31	30	29	28	27	26	25	24
R	GO	0	0	0	PUSHW1	INVW1	PUSHW0	INVW0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	ENCACHE
W								
RESET:	0	1	1	1	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 11-2：缓存控制寄存器（LMEM_CCR）

ENCACHE — 启用缓存。

1 = 启用缓存

0 = 禁用缓存

INVW0 — 使方式 0 失效。如果 PUSHW0 和 INVW0 位被设置，则在设置 GO 位之后，将按位推入方式 0 中所有被修改的行，并使方式 0 中所有行失效（清除方式 0）。

1 = 设置 GO 位时，使方式 0 中的所有线路无效。

0 = 无操作

PUSHW0 — 推送方式 0

1 = 设置 GO 位时，以方式 0 推送所有修改过的线路

0 = 无操作

INVW1 — 使方式 1 失效。如果 PUSHW0 和 INVW0 位被设置，则在设置 GO 位之后，将按方式 1 推送所有修改过的线路，并使方式 1 中的所有线路失效（清除方式 1）。

1 = 设置 GO 位时，使方式 1 中的所有线路无效。

0 = 无操作

PUSHW1 — 推送方式 1

1 = 设置 GO 位时，以方式 1 推送所有修改过的线路

0 = 无操作

GO — 启动缓存命令设置此位将启动由 27-24 位指示的缓存命令。读取此位可指示命令是否处于活动状态。此位将保持为有效状态，直至该命令完成。写入零无任何作用。

1 = 写入：启动由 27-24 位指示的命令。读取：缓存命令处于活动状态。

0 = 写入：无影响。读取：无缓存命令处于活动状态。

11.4.2. 缓存行控制寄存器（LMEM_CLCR）

偏移地址：0x0004

	31	30	29	28	27	26	25	24
R	0	0	0	0	LACC	LADSEL	LCMD[1:0]	
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	LCR[2:0]			0	0	0	TDSEL
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	WSEL	Cache Address[13:8]					
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	Cache Address[7:2]						0	LGO
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 11-3：缓存行控制寄存器（LMEM_CLCR）

LGO — 启动缓存行命令。设置此位将启动由 LMEM_CLCR[27:24] 指示的缓存行命令。读取此位可指示是否激活行命令。此位将保持设置状态，直至命令完成。写入零无影响。此位与 CSAR[LGO] 共享。

1 = 写入：启动由 27-24 位指示的行命令。读取：行命令有效。

0 = 写入：无影响。读取：无行命令有效。

CACHE Address — 缓存地址。

CLCR[13:4] 位用于访问标签阵列；CLCR[13:2] 位用于访问数据阵列。

WSEL — 方式选择。选择线路命令的方式。

1 = 方式 1。

0 = 方式 0。

TDSEL — 标签/数据选择。选择标签或数据以执行搜索以及读取或写入命令。

1 = 天。

0 = 数据。

LCR[2:0] — 线路命令，当前线路状态。

LCMD[1:0] — 线命令。

00 = 搜索和读取或写入。

01 = 销售无效

10 = 推

11 = 清晰

LADSEL — 行地址选择。当使用缓存地址时，还必须在 CLCR[WSEL] 中指定方式。当使用物理地址时，将搜索两种方式，并且仅在命中时执行命令。

0 = 缓存地址

1 = 物理地址

LACC — 线路访问类型。

0 = 读取

1 = 写入

11.4.3. 缓存搜索地址寄存器（LMEM_CSAR）

CSAR 寄存器用于定义 CLCR[LADSEL] 位中指定的行大小指令的显式缓存地址或物理地址。

偏移地址：0x0008

	31	30	29	28	27	26	25	24
R W	Physical Address[31:24]							
RESET:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R W	Physical Address[23:16]							
RESET:	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R W	Physical Address[15:8]							
RESET:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	Physical Address[7:2]						0	LGO
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 11-4：缓存搜索地址寄存器（LMEM_CSAR）

PhysicalAddress[31:2] — 物理地址。它代表系统地址的[31:2] 位。

PhysicalAddress[31:14] 位用于标签比较，PhysicalAddress[13:4] 位用于访问标签阵列，PhysicalAddress[13:2] 位用于访问数据阵列。

LGO — 启动缓存行命令。设置此位将启动由 LMEM_CLCR[27:24] 指示的缓存行命令。读取此位可指示行命令是否处于活动状态。此位将保持设置状态，直至命令完成。写入零无任何影响。

该位与 CLCR 共享[LGO]

- 写入：启动 LMEM_CLCR[27:24] 指示的行命令。读取：行命令有效。
- 写入：无影响。读取：没有行命令处于活动状态。

11.4.4. 缓存读写值寄存器（LMEM_CCVR）

CCVR 寄存器用于存储 CLCR 寄存器中指定的命令的写入数据或读取数据。

偏移地址：0x000C

	31	30	29	28	27	26	25	24
R W	DATA[31:24]							
RESET:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R W	DATA[23:16]							
RESET:	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R W	DATA[15:8]							
RESET:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	DATA[7:0]							
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 11-5：缓存搜索地址寄存器（LMEM_CSAR）

Data — 缓存读写数据

对于标签搜索、读取或写入：

- 数据[31:14] 位用于标签阵列读/写值
- 数据[13:4] 位用于读取时的标签集地址；写入时未使用
- 保留了[3:2] 位的数据

对于数据搜索、读取或写入：

- 数据[31:0]位用于数据阵列读/写值

11.4.5. 缓存访问寄存器（LMEM_ACRG）

ACRG 用于控制 10 个不同内存区域的属性，例如：直写、回写或不可缓存。

偏移地址：0x0020

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	R3_CACHEABLE	R3_WT_WB	R2_CACHEABLE	R2_WT_WB	R1_CACHEABLE	R1_WT_WB	R0_CACHEABLE	R0_WT_WB
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 11-6：缓存访问寄存器（LMEM_ACRG）

Region0: 0x2000_0000 ~ 0x2FFF_FFFF

Region1: 0x6000_0000 ~ 0x6FFF_FFFF

Region2: 0x7000_0000 ~ 0x7FFF_FFFF

Region3: 0x8000_0000 ~ 0x8FFF_FFFF

Rx cachesable — 区域 x 可缓存。

1 = cacheable.

0 = non-cacheable

Rx_WT_WB — 如果区域 x 可缓存，则该区域为写通。

1 = write-back

0 = write-through

11.4.6. 缓存页失效基地址寄存器（LMEM_PAGEINV_BADDR）

偏移地址：0x0180

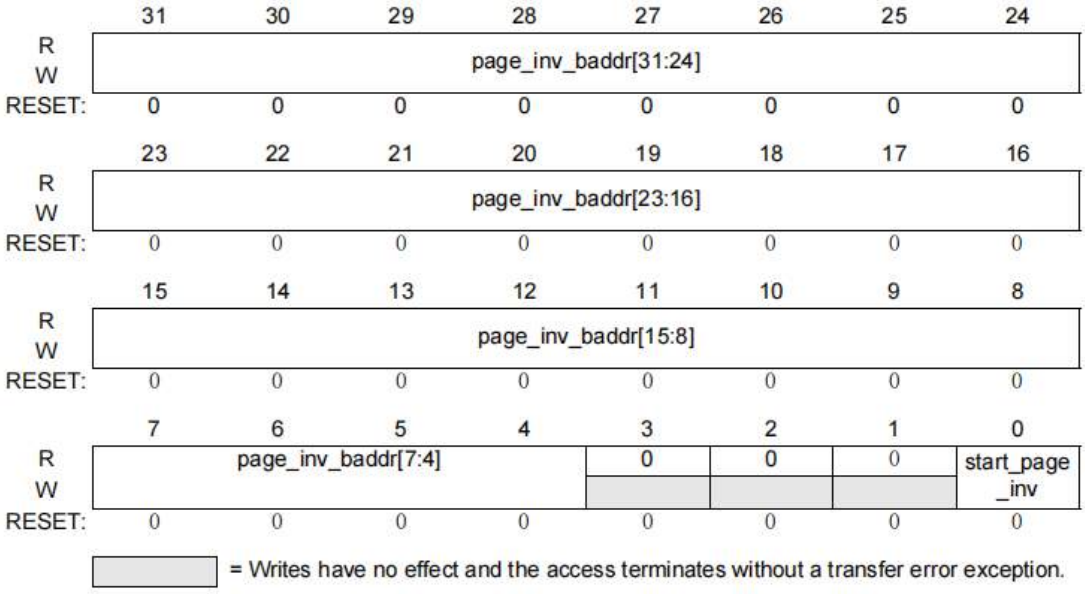


图 11-7：缓存页无效化基地址寄存器（LMEM_PAGE_INV_BADDR）

page_inv_baddr[31:4] — 缓存无效的系统内存起始地址。由于行对齐的原因，下 4 位被忽略。

startPageInv — 开始到页面无效。如果无效操作已完成，则清除。

该位共享在 LMEM_PAGE_INV_SIZERegs 的 0 位中。

11.4.7. 缓存页失效基数寄存器（LMEM_PAGE_INV_SIZE)

偏移地址：0x0184

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	page_inv_size[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	page_inv_size[7:4]				0	0	0	start_page_inv
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 11-8：缓存页无效大小寄存器（LMEM_PAGE_INV_SIZE)

page_inv_size[15:4] — 系统内存缓存无效大小。由于行对齐，下 4 位被忽略。

startPageInv — 开始到页面无效。如果无效操作已完成，则清除。

该位共享在 LMEM_PAGEINV_BADDR_REG 的 0 位中。

11.4.8. 缓存时钟使能寄存器（LMEM_CACHE_CLK_EN）

偏移地址：0x0188

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	cache_clk_enable
W								
RESET:	0	0	0	0	0	0	0	1

= Writes have no effect and the access terminates without a transfer error exception.

图 11-9：缓存时钟使能寄存器（LMEM_CACHE_CLK_EN）

cache_enable — 缓存时钟已启用。如果缓存已启用，则必须为“1”。

11.5. 缓存功能

该设备的高速缓存结构如下：两个高速缓存都采用 2 路组相联结构，总容量为 32KB 。高速缓存具有 32 位地址和数据通路，每行大小为 16 字节。高速缓存标签和数据存储使用外部单端口同步 RAM。

对于这 32KB 的缓存，每个缓存 TAG 功能使用两个 1024×20 位 RAM 阵列，而缓存 DATA 功能使用两个 4096×32 位 RAM 阵列。缓存 TAG 条目存储每个缓存行的高地址 18 位以及修改和有效位。缓存 DATA 条目存储四个字节的代码或数据。

所有常规的缓存访问都使用物理地址。这导致了以下的缓存地址使用：

缓存容量为 32 KB，其计算公式为：（1024 组）×（16 字节每行）×（双路组相联）。

TAG:

- address[31:14] 标签中用于比较（命中）的逻辑
- address[13:4] 用于从选择 1024 组中的一组
- address[3:0] 未使用

DATA:

- address[31:14] 未使用
- address[13:4] 用于选择 1024 组中的其中一组
- address[3:2] 用于在集合中选择四个 32 位字中的一个的
- address[1:0] 用于选择 32 位字中的字节

缓存采用单周期并行 TAG 和 DATA 访问结构。该结构在两个阶段的 AHB 总线流水线中从地址阶段的负边沿到数据阶段的负边沿进行对齐：

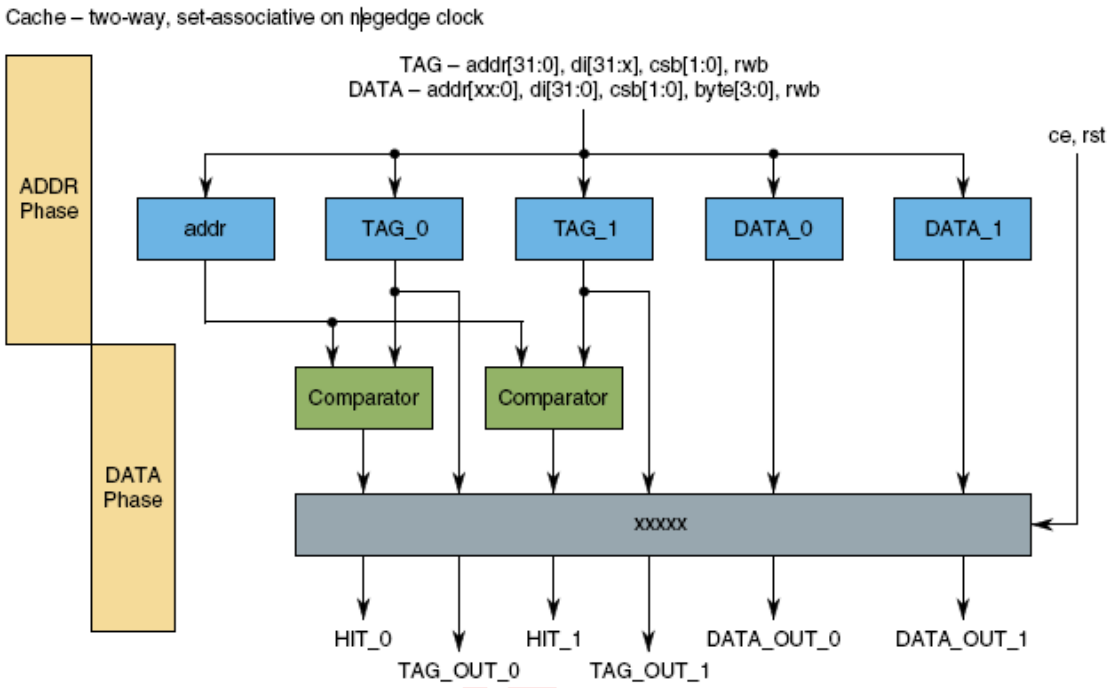


图 11-10：缓存标签和数据访问结构

在常规可缓存操作中，完整地址会在地址阶段的负边沿被缓存控制逻辑注册，其中地址位[12:4] 存入 TAG 阵列，地址位[12:2] 存入 DATA 阵列。从地址阶段负边沿到数据阶段负边沿的完整周期内，TAG 和 DATA 阵列会被访问并进行缓存命中分析。对于命中情况，目标读取数据会在数据阶段后半段返回，写入操作则从数据阶段周期的负边沿开始执行。对于未命中情况，系统会根据需要保持数据阶段状态，同时通过从属总线（CCM、CSM）向交叉开关发送行级数据传输，以访问所需的缓存行。

11.6. 缓存控制

代码缓存和系统缓存在复位时处于禁用状态。缓存标签和数据阵列在复位时不被清除。因此，要启用缓存，必须执行缓存命令以清除和初始化所需的标签阵列位，并配置和启用缓存。

11.6.1. 缓存设置命令

缓存设置命令可以作用于：

- 全程 0，
- 全程 1，或
- 全双向（完全缓存）。

使用 CCR 寄存器的高比特位启动缓存设置命令。缓存设置命令在独立于缓存使能位 CCR[ENCACHE] 的情况下执行其操作。

缓存集命令通过设置 CCR[GO] 位来启动。此位还充当设置命令的忙位。在命令有效期间，该位保持为置位状态，并且在设置命令完成后由硬件清除。

支持的缓存设置命令如图 11-11 所示。设置命令的工作原理如下：

- Invalidate－无条件清除缓存条目中的有效位并修改其位。
- Push－如果缓存条目有效且被修改，则推送该条目并清除修改位。如果条目无效或未被修改，则保持原样。
- 清除－如果缓存条目有效且被修改，则推送该缓存条目，然后清除有效和修改位。如果条目无效或未被修改，则清除有效位。

CCR[27:24]				Command
PUSH W1	INVW1	PUSH W0	INVW0	
0	0	0	0	NOP
0	0	0	1	Invalidate all way 0
0	0	1	0	Push all way 0
0	0	1	1	Clear all way 0
0	1	0	0	Invalidate all way 1
0	1	0	1	Invalidate all way 1; invalidate all way 0 (invalidate cache)
0	1	1	0	Invalidate all way 1; push all way 0
0	1	1	1	Invalidate all way 1; clear all way 0
1	0	0	0	Push all way 1
1	0	0	1	Push all way 1; invalidate all way 0
1	0	1	0	Push all way 1; push all way 0 (push cache)
1	0	1	1	Push all way 1; clear all way 0
1	1	0	0	Clear all way 1
1	1	0	1	Clear all way 1; invalidate all way 0
1	1	1	0	Clear all way 1; push all way 0
1	1	1	1	Clear all way 1; clear all way 0 (clear cache)

图 11-11：缓存组命令

重置后，在使用缓存之前，完成无效缓存命令。

11.6.2. 缓存行命令

缓存行命令一次只对缓存中的单行进行操作。可以使用物理地址或缓存地址执行缓存行命令。

- 缓存地址由集合地址和路选择组成。line 命令作用于指定的缓存行。
- 对于带有物理地址的缓存行指令，首先会以物理地址[13:4] 位所指定的缓存组进行双向搜索。如果命中，则在命中路执行相应操作。

缓存行指令通过 CLCR 寄存器的高位进行指定。这类指令在执行缓存操作时，其操作与缓存使能位（CCR[ENCACHE]）无关。当使用缓存地址时，可通过 CLCR 寄存器完整指定指令；若采用物理地址，则需同时借助 CSAR 寄存器来确定具体物理地址。

行缓存命令的启动是通过设置行缓存命令进行位（CLCR[LGO] 或 CSAR[LGO] ）来实现的。此位还充当行缓存命令的忙位。在命令有效期间，此位保持为置位状态，当命令完成时，硬件会清除此位。

CLCR[27:24] 位选择线路命令如下：

CLCR[27:24]			Command
LACC	LADSEL	LCMD	
0	0	00	Search by cache address and way
0	0	01	Invalidate by cache address and way
0	0	10	Push by cache address and way
0	0	11	Clear by cache address and way
0	1	00	Search by physical address
0	1	01	Invalidate by physical address
0	1	10	Push by physical address
0	1	11	Clear by physical address
1	0	00	Write by cache address and way
1	0	01	Reserved, NOP
1	0	10	Reserved, NOP
1	0	11	Reserved, NOP
1	1	*xx*	Reserved, NOP

图 11-12：缓存行命令

11.6.2.1. 使用缓存地址执行一系列行命令

通过写入 CLCR 就可以执行一系列具有递增缓存地址的行命令。

- 将命令放入 CLCR[27:24]，
- 根据需要设置方式（CLCR[WSEL]）和标记/数据（CLCR[TDSEL]）控件，
- 将缓存地址放入 CLCR[CACHEADDR] 中，以及
- 设置行命令 Go 位（CLCR[LGO]）。

当某行命令完成时，通过以下步骤启动下一个命令：

- 增加缓存地址（在第 2 位对数据进行遍历，或在第 4 位对行进行遍历），以及
- 设置行命令 Go 位（CLCR[LGO]）。

11.6.2.2. 使用物理地址执行一系列行命令

使用以下步骤，通过递增的物理地址执行一系列行命令：

- 将命令放入 CLCR[27:24]
- 设置标签/数据（CLCR[TDSEL]）控制
- 将物理地址放入 CSAR[PHYADDR] 中，并设置行命令 go 位（CSAR[LGO]）。

当某行命令完成时，通过以下步骤启动下一个命令：

- 增加物理地址（在第 2 位进行数据遍历或在第 4 位进行行遍历），以及
- 设置线路命令发送位（CSAR[LGO]）。

行命令 GO 位由 CLCR 和 CSAR 寄存器共享，因此，上述步骤可以通过向 CSAR 寄存器写入一次来完成。

11.6.2.3. 行命令结果

当完成行命令操作时，CLCR 寄存器会记录该指令所指向数据行的初始状态信息。对于使用缓存地址的行命令，在执行指令操作前会先从目标缓存行读取这些信息。若采用物理地址的行命令，则会在命中时读取相关信息，随后执行命中缓存行的操作；若未命中，则会清除初始有效位。通常情况下，如果有效指示符 CLCR[LCIVB] 被清除，说明该行在命令开始时已失效，因此不会执行任何行操作。

CLCR[22:20]			For cache address commands	For physical address commands
LCWAY	LCIMB	LCIVB		
0	0	0	Way 0 line was invalid	No hit
0	0	1	Way 0 valid, not modified	Way 0 valid, not modified
0	1	0	Way 0 line was invalid	No hit
0	1	1	Way 0 valid and modified	Way 0 valid and modified
1	0	0	Way 1 line was invalid	No hit
1	0	1	Way 1 valid, not modified	Way 1 valid, not modified
1	1	0	Way 1 line was invalid	No hit
1	1	1	Way 1 valid and modified	Way 1 valid and modified
4'b1xxx			Bit 23 is available for future use, giving 8 more options if necessary.	

图 11-13：线路指令结果

当执行除写入指令外的其他行指令时，CCVR（缓存读写值寄存器）会存储该指令目标数据或行标签的初始状态信息。对于行指令，CLCR[TDSEL] 寄存器会根据需要选择标记或数据。若行指令使用物理地址且发生未命中，则数据内容无需特别关注。在写入指令场景下，CCVR 寄存器将保存待写入的数据内容。

12. 横条开关（XBAR）

12.1.介绍

12.1.1. 概述

本章详细阐述交叉开关的布局设计、配置方案及编程方法。该设备通过硬件互连矩阵架构，实现总线主控与从属设备间的高效连接。这种创新设计不仅确保所有主控设备能同时访问不同从属设备且互不干扰，还能在主控设备同时访问同一从属设备时自动进行仲裁控制。此外，系统支持按从属设备逐一编程设置多种总线仲裁机制及其参数配置。

XBAR 有三个主端口和八个从端口，图 12-1 是 XBAR 的方块图。

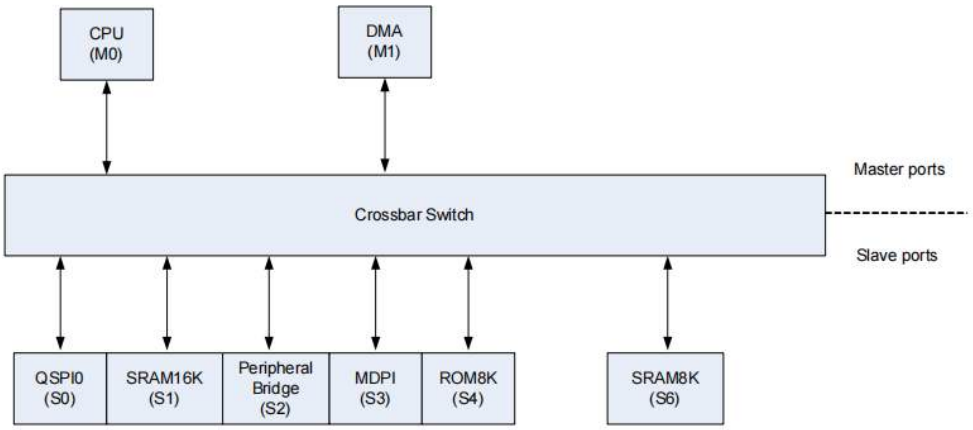


图 12-1：XBAR 模块的方块图

12.1.2. 特性

交叉开关具有以下显著特点：

- 2 个主端口：
  - CPU
  - DMA
- 6 个从属端口
  - ROM 内存控制器
  - 两个内部 SRAM 存储器控制器
  - 一个 QSPI 接口
  - EBI 接口
  - 边缘桥
- 对称交叉开关总线交换机实现

- 不同主服务器对不同从服务器的并发访问
- 可根据从属设备之间的关系配置从属设备仲裁属性

- 32 位地址，32 位数据路径
- 固定优先方案和固定停车策略
- 支持 8、16 和 32 位单传输
- 支持低功耗停车模式

12.2. 内存映射和寄存器

本节提供有关 XBAR 寄存器的信息。

12.2.1. 寄存器摘要

交叉开关的每个从属端口都包含配置寄存器。配置寄存器的读取和写入传输需要两个总线时钟周期。只能在监督模式下对这些寄存器进行读取和写入操作。此外，只能通过 32 位访问对这些寄存器进行读取或写入操作。

如果在交叉开关中访问了未实现的位置，则会返回总线错误响应。

从属寄存器还具有一个位，当设置时，寄存器将无法被写入。寄存器仍可读取，但以后的写入尝试对寄存器没有影响，并且会向发起写入操作的主设备发送总线错误响应而终止。

提示：

本节显示所有八个主从端口的寄存器。如果该设备未使用某个主从端口，则写入其寄存器时会出现异常结果。请参阅芯片配置详情以获取您设备的主从端口具体分配信息。此外，所有交叉开关寄存器的引用均基于物理端口连接，而非逻辑端口号。

XBAR 寄存器的映射图如表 12-1 所示。

表 12-1：XBAR 寄存器配置总结

地址偏移量	寄存器	访问权限 [1]
0x0000	从机端口 0 的主优先级寄存器（MPR0）	S
0x0010	从机端口 0 的通用控制寄存器（SGPCR0）	S
0x0100	从属端口 1 的主优先级寄存器（MPR1）	S
0x0110	从属端口 1 通用控制寄存器（SGPCR1）	S
0x0100	从属端口 2 的主优先级寄存器（MPR2）	S
0x0210	从属端口 2 的通用控制寄存器（SGPCR2）	S
0x0300	从机端口 3 的主优先级寄存器（MPR3）	S
0x0310	从机端口 3 通用控制寄存器（SGPCR3）	S
0x0400	从机端口 4 的主优先级寄存器（MPR4）	S
0x0410	从机端口 4 的通用控制寄存器（SGPCR4）	S
0x0500	从属端口 5 的主优先级寄存器（MPR5）	S
0x0510	从机端口 5 的通用控制寄存器（SGPCR5）	S
0x0600	从机端口 6 的主优先级寄存器（MPR6）	S
0x0610	从机端口 6 的通用控制寄存器（SGPCR6）	S
0x0700	从机端口 7 的主优先级寄存器（MPR7）	S
0x0710	从机端口 7 的通用控制寄存器（SGPCR7）	S

提示：S = 仅允许访问 CPU 监控模式。S/U = CPU 监控或用户模式访问。用户模式访问仅限于监控器的地址，否则将不会产生任何影响，并且会导致周期终止传输错误。

12.2.2. XBAR 寄存器说明

12.2.2.1. 主优先级寄存器（XBAR_MPRn）

主优先级寄存器（MPR）驻留在每个从属端口中，并且按从属端口的基准设置每个主端口的优先级，例如，MPR0 为从属端口 0 设置每个主端口的优先级。

主优先级寄存器只能在具有 32 位访问权限的监督模式下访问。一旦从属通用控制寄存器中的 RO（只读）位被设置，就只能从 MPR 中读取主优先级寄存器，对 MPR 的写入尝试将不会对 MPR 产生影响，并将导致错误响应。

偏移地址：0x0000 + n*0x100

	31	30	29	28	27	26	25	24
Read:	0	0	0	0	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0	0	0	0	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0	0	0	0	0	MSTR2		
Write:								
RESET:	0	0	0	0	0	0	1	0

	7	6	5	4	3	2	1	0
Read:	0	MSTR1				MSTR0		
Write:								
RESET:	0	0	0	1	0	0	0	0

= Writes have no effect and without a transfer error exception.

图 12-2：主优先级寄存器（XBAR_MPRn）

表 12-2：XBAR 主优先级寄存器字段说明

字段	描述
31–11	保留
10–8 MSTR2	第二优先级 这些位为相关联的从属端口为主端口 2（USBC）设置仲裁优先级。 这些位由硬件复位初始化，复位值为 010。 000：当访问从端口时，此主设备具有级别 1（最高）优先级。 001：此主设备在访问从属端口时具有级别 2 优先级。 ... 111：当访问从端口时，此主设备具有级别 3（最低）优先级。
7	保留 此位保留给将来扩展使用，读取时为零，为了向上兼容，写入时也应为零。
6–4 MSTR1	硕士一年级优先 这些位为相关联的从属端口为主端口 1（DMA）设置仲裁优先级。 这些位由硬件复位初始化，复位值为 001。 000：当访问从端口时，此主设备具有级别 1（最高）优先级。 001：此主设备访问从属端口时具有级别 2 优先级。 ... 111：当访问从端口时，此主设备具有级别 3（最低）优先级。
3	保留 此位保留给将来扩展使用，读取时为零，为了向上兼容，写入时也应为零。
2–0 MSTR0	主控台 0 优先级 这些位为相关联的从属端口中的主端口 0（CPU）设置仲裁优先级。 这些位由硬件复位初始化，复位值为 000。 000：当访问从端口时，此主设备具有 1 级（最高）优先权。 001：此主设备访问从属端口时具有级别 2 优先级。 ... 111：当访问从端口时，此主设备具有级别 3（最低）优先级。

提示：不能用相同的优先级对两个可用的主端口进行编程。如果尝试用相同的优先级对两个或多个可用的主端口进行编程，将导致错误响应，并且 MPR 不会更新。

12.2.2.2. 从属通用控制寄存器（XBAR_SGPCRn）

从属通用控制寄存器（SGPCR）负责管理每个从属端口的多项功能。其中只读位（RO）一旦被置为 1，将禁止对该端口相关寄存器进行任何写入操作。用户可自由设置该位的写入次数，但当其被写入 0 后，必须通过复位操作才能再次进行写入。

PCTL 位决定了从端口在没有主设备主动请求时的停放方式。可选方案包括：停靠由 park 位定义的主设备、停靠最后一个使用该端口的主设备，或进入低功耗停放模式 — 这种模式会在无主设备请求访问时强制所有输出端口处于非活动状态。虽然低功耗停放功能能在端口未满载时节省整体功耗，但当端口闲置时，任何主设备尝试访问都会因无法停靠而产生额外时钟延迟。

当没有主设备发出有效请求时，PARK 位决定从设备将停靠哪个主设备。请谨慎地只选择实际存在于设计中的主设备端口。如果用户将 PARK 位编程为当前设计实现中不存在的主设备，将会导致未定义的行为。

提示：SGPCR 只能在具有 32 位访问权限的监督模式下访问。一旦设置了 XBAR_SGPCR[RO]，SGPCR 只能被读取；尝试写入它将不会对 SGPCR 产生影响，并导致错误响应。

偏移地址：0x0010 + n*0x100

	31	30	29	28	27	26	25	24
Read:	RO	0	0	0	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0	0	0	0	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
Read:	0	0	0	0	0	0	ARB	
Write:								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
Read:	0	0	PCTL			PARK		
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and without a transfer error exception.

图 12-3：从属通用控制寄存器（XBAR_SGPCRn）

表 12-3：XBAR 从机通用控制寄存器字段说明

字段	描述
31 RO	只读 该位用于强制将从属端口的所有寄存器设置为只读。一旦写入‘1’，只能通过硬件复位清除。 该位由硬件复位初始化，复位值为 0。 0：所有这些从属端口的寄存器都可以写入。 1：所有这些从属端口的寄存器都是只读的，不能写入（尝试写入没有效果，并导致错误响应）。
30–10	保留 这些位保留给将来扩展使用，读取时为零，写入时也应为零以保持向上兼容性。
9–8 ARB	仲裁模式 这些位用于选择从属端口的仲裁策略。这些位由硬件复位初始化。复位值为 00。 00：固定优先级 01：轮询（循环）优先级 10：保留 11：保留
7–6	保留 这些位保留给将来扩展使用，读取时为零，为了向上兼容，写入时也应为零。
5–4 PCTL	停车控制系统 确定从属端口的驻车控制。低功耗驻车功能可实现 如果从属端口未饱和，则可节省总体功耗；但是，当任何主设备试图访问未使用的从属端口时，这将强制产生额外的时钟延迟，因为该端口没有停驻在任何主设备上。 这些位由硬件复位初始化，复位值为 00。 00：当没有主设备发出请求时，仲裁器将把从端口停放在由定义的主端口上。 PARK 位字段。 01：当没有主设备发出请求时，仲裁器将把从属端口停放在最后一个控制该从属端口的主设备上。 10：当没有主设备发出请求时，仲裁器将把从属端口停放在“无主”位置，并将所有输出驱动到一个恒定的安全状态。 11：保留

字段	描述
3	保留 此位保留给将来扩展使用，读取时为零，为了向上兼容，写入时也应为零。
2–0 PARK	PARK 这些位用于确定当没有主端口主动发出请求时，该从端口停靠在哪个主端口上。 这些位由硬件复位初始化，复位值为 000。 **提示：**仅选择实际存在于设备上的主端口。否则，可能会出现未定义的行为。 000：将车辆停靠在主端口 0（CPU） 001：停靠主端口 1（DMA） 010：将车辆停靠在主端口 2（USBC） 011：保留 100：保留 101：保留 110：保留 111：保留

12.3. 功能说明

本节更详细地描述了 XBAR 的功能。

12.3.1. 一般操作

当主设备向交叉开关发送访问请求时，该请求会立即被接收。若目标从端口处于空闲状态，则请求会立即在该端口呈现。通过交叉开关可实现单时钟周期（零等待状态）的访问。若目标从端口处于忙态或被挂起在其他主端口上，请求方主设备将直接进入等待状态，直至目标端口能够响应其请求。请求处理的延迟取决于各主设备的优先级等级及响应从端口的访问时间。

由于交叉开关似乎只是主设备的另一个从设备，因此主设备并不知道它是否实际拥有其所指向的从设备端口。虽然主设备不控制其所指向的从设备端口，但它只是进入等待状态。

只有在先前访问了其他从属端口之后，系统才会将目标从属端口的控制权交给某个主控器，而不管该主控器在新目标从属端口上的优先级如何。这样可以防止出现以下情况下的死锁：

- 优先级较高的主机会：
- 对一个响应时间较长的从属端口发出的未决请求

- 即将访问另一个从属端口，以及
- 低优先级主设备也向与高优先级主设备的等待访问相同的从属端口发出请求。

当主设备成功接管目标从端口后，将始终保持对该端口的控制权，直至通过执行空闲周期或释放该端口以备后续访问而主动放弃控制权。若其他高优先级主设备向该端口发起请求，主设备也可能失去控制权；但若执行固定长度的突发传输操作，则会持续保持控制权直至传输完成。当主设备接管特定从端口时，交叉开关会将所有来自从端口的响应信息完整返回给请求方主设备。

交叉开关终止所有主设备的空闲传输（与允许由某个从设备总线发出终止操作的情况相反）。此外，当没有主设备请求访问从设备端口时，即使默认主设备可能被授予对从设备端口的访问权，交叉开关也会将空闲传输驱动到从设备总线上。

当交叉开关对从属总线进行空闲处理时，可通过 XBARx_CRSn[PARK] 指令将从属端口挂接到指定的主端口。这种操作旨在节省仲裁延迟的初始时钟周期 — 若主端口需要通过仲裁来获取从属端口控制权，原本会消耗这部分时钟资源。此外，通过 XBARx_CRSn[PCTL] 指令可将从属端口切换至低功耗挂起模式，从而实现节能效果。

12.3.2. 寄存器一致性

由于寄存器的内容对交叉开关的操作具有实时影响，因此必须了解，任何寄存器修改在写入寄存器时即生效。寄存器的值不跟踪与从属端口相关的主访问；相反，它们只跟踪从属访问。

12.3.3. 仲裁

XBAR 支持两种仲裁方案：一种是简单的固定优先级比较算法，另一种是简单的轮询公平算法。每个从属端口的仲裁方案都可以独立编程。

12.3.3.1. 固定优先级操作

在固定优先级模式下运行时，每个主设备都会在 XBARx_PRSn（优先级寄存器）中被分配一个唯一的优先级。如果两个主设备请求访问从设备端口，则在所选优先级寄存器中具有更高优先级的主设备将获得对从设备端口的控制权。

当主设备向从端请求控制权时，从端会检查新请求者的优先级是否高于当前控制该端口的主设备（除非该端口处于停用状态）。从端会在每个时钟边沿执行仲裁检查，以确保当前控制权归属正确的主设备（若存在）。

若新请求主设备的优先级高于当前控制从端口的主设备，则在下一个时钟边沿，新请求主设备将获得从端口控制权。但存在例外情况：当当前控制从端口的主设备正在执行固定长度突发传输或锁定传输时，新请求主设备必须等待该传输完成，才能获得从端口控制权。

如果新请求主节点的优先级低于当前控制从属端口的主节点，则新请求主节点将被迫等待，直到当前主节点运行以下周期之一：

- 空闲周期

• 不进入当前从属端口以外的其他位置的空闲循环

12.3.3.2. 循环优先操作

在采用轮询模式运行时，系统会根据物理主端口号为每个主设备分配相对优先级。该相对优先级将与在从属总线上执行传输的最后一个主设备的ID（即主端口号）进行比较。请求在从属总线上执行传输的主设备具有最高优先级。

横条开关（XBAR）

在下一个转运边界处成为运输巴士的所有者（包括锁定和

固定长度的突发传输）。优先级基于请求主设备的 ID 与最后一个主设备的 ID 之间的差距。

当主设备获得某个从设备端口的访问权限后，可以持续对该端口进行任意数量的数据传输，直到有其他主设备对该端口发起请求为止。下一个排队等待的主设备将在下一个传输周期边界获得该端口的访问权限 — 如果当前主设备没有待处理的访问请求，这个权限甚至可能在下一个时钟周期就立即生效。

作为轮询模式下仲裁的一个示例，假设交叉开关使用主端口 0、1、4 和 5 实现。如果从属端口的最后一个主端口是主端口 1，而主端口 0、4 和

5 发出同时请求，它们按顺序 4、5 和 0 获得服务。

虽然系统仍可能以轮询模式使用驻留状态，但除非被驻留的主设备实际执行数据传输，否则不会影响轮询指针。经过一个仲裁周期后，设备会自动切换到队列中的下一个主设备。当从设备端口进入低功耗驻留模式时，轮询指针会被重置为指向主设备端口 0，使其获得最高优先级。

12.3.4. 优先权分配

需要为每个主端口分配唯一的 3 位优先级。如果尝试在寄存器中用相同优先级对多个主端口进行编程（MPR），则 XBAR 将响应错误，且寄存器将不会更新。

12.4. 初始化/信息

交叉开关不需要初始化，也不需要为交叉开关进行初始化。硬件复位确保交叉开关使用的寄存器位被正确初始化到有效状态。应设置和优先级以实现最大系统性能。

13. 直接内存访问（DMA）

13.1. DMA 专用信息

本节介绍 DMA 特定的参数化、自定义和功能可用性信息，这些信息在本章的其余部分中没有特别提及。

13.1.1. DMA 特定功能

- 6 个可编程通道，支持独立的 8、16 或 32 位单值或块传输
- 支持可变大小队列和循环队列
- 源地址寄存器和目标地址寄存器独立配置，以保持后增量器恒定
- 由外围设备、CPU、周期性计时器中断或 DMA 通道请求发起的每次传输
- 可通过 QSPI、QADC 获取外围 DMA 请求源
- 每个 DMA 通道在完成单值或块传输后，可选择性地向 CPU 发送中断请求。
- DMA 可以在系统内存和所有可访问的内存映射位置之间传输，包括外围设备和寄存器
- DMA 支持以下功能：
  - 散点图
  - 通道链接
  - 内环偏移
  - 仲裁

    固定组，固定频道

    Round Robin 组，固定信道

    轮询组，轮询通道

    固定组，轮询通道
  - 信道优先权
  - 取消频道传输
- 中断 — DMA 为每个已实现的通道提供一个中断请求，并提供一个组合 DMA 错误中断，用于向系统标记传输错误。每个通道 DMA 中断都可以启用或禁用，并提供传输完成的通知。有关这些中断的分配，请参阅参考手册 EIC 章节中的中断向量。

13.1.2. 通道分配

块到 DMA 通道的 DMA 请求之间的分配如表 13-1 所示。

表 13-1：DMA 信道分配

DMA 请求	通道	描述
ADC_ISR[EMPTY]	0	当 FIFO 中的数据编号不为空且位 ADC_CFGR1 [DMAEN] 被设置时，qadc_dmaREQ
QSPI0 DMA 接收请求	1	当接收 FIFO 中的有效数据条目数等于或大于 DMARDLR[DMARDL] + 1 且 DMACR[RDMAE] = 1 时，将生成 QSPI0dma_rxREQ
QSPI0 DMA 发送申请	2	当发送 FIFO 中的有效数据条目数等于或小于 DMATDLR[DMATDL] 时，将生成 QSPI0dma_txREQ 信号，且 DMACR[TDMAE] = 1
SPI0 DMA 接收申请	3	当接收 FIFO 中的有效数据条目数超过 SPIDMATHR[RXDMATH] 或 RX FIFO TimeOut 时，将生成 SPI0 dma_rx_req 信号，且 SPIDMACR[RXDMAE] = 1
SPI0 DMA 发送申请	4	当发送 FIFO 中的有效数据条目数小于 SPIDMATHR[TXDMATH] 或 TX FIFO TimeOut，且 SPIDMACR[TXDMAE] = 1 时，生成 SPI0dma_txREQ 信号
SPI1 DMA 接收申请	5	当接收 FIFO 中的有效数据条目数超过 SPIDMATHR[RXDMATH] 或 RX FIFO TimeOut 时，将生成 SPIDMA_RXREQ 信号，且 SPIDMACR[RXDMAE] = 1
SPI1 DMA 发送申请	6	当发送 FIFO 中的有效数据条目数小于 SPIDMATHR[TXDMATH] 或 TX FIFO TimeOut，并且 SPIDMACR[TXDMAE] = 1 时，生成 SPIDMA Tx_req 信号
PIT1 DMA 请求	7	pit1_dma_req 当 PIT0 计数器达到 0x0000 且 PCSR[PDMAE] = 1 时，生成信号 pit1_dma_req
PIT0 DMA 请求	8	pit0_dma_req 当 PIT0 计数器达到 0x0000 且 PCSR[PDMAE] = 1 时，生成信号 pit0_dma_req
保留	9	保留

DMA 请求	通道	描述
SCI0 TX DMA 请求	10	sci0_tx_req 当发送 FIFO 中的数据字数等于或小于 SCI_water[TXWATER] 所指示的数值时，将生成信号 sci0_tx_req
SCI0 RX DMA 请求	11	sci0_rx_req 当接收缓冲区中的数据字数大于 SCI_WATER[RXWATER] 所指示的数值时，将生成 sci0_rx_req 信号
SCI1 TX DMA 请求	12	sci1_tx_req 当发送 FIFO 中的数据字数等于或小于 SCI_water[TXWATER] 所指示的数值时，将生成信号 sci1_tx_req
SCI1 RX DMA 请求	13	sci1_rx_req 当接收缓冲区中的数据字数大于 SCI_WATER[RXWATER] 所指示的数值时，将生成 sci1_rx_req 信号
保留	14	保留
保留	15	保留

13.2. 介绍

直接内存访问控制器（DMA）能够通过 16 个可编程通道执行复杂的数据传输操作，且几乎无需主机处理器介入。其硬件微架构包含一个 DMA 引擎，该引擎负责执行源地址与目标地址的计算，并完成实际的数据传输操作，同时配备基于 SRAM 的存储器来存储各通道的传输控制描述符（TCD）。这种设计有效缩小了整体数据块的大小。

图 13-1 是 DMA 模块的方块图。

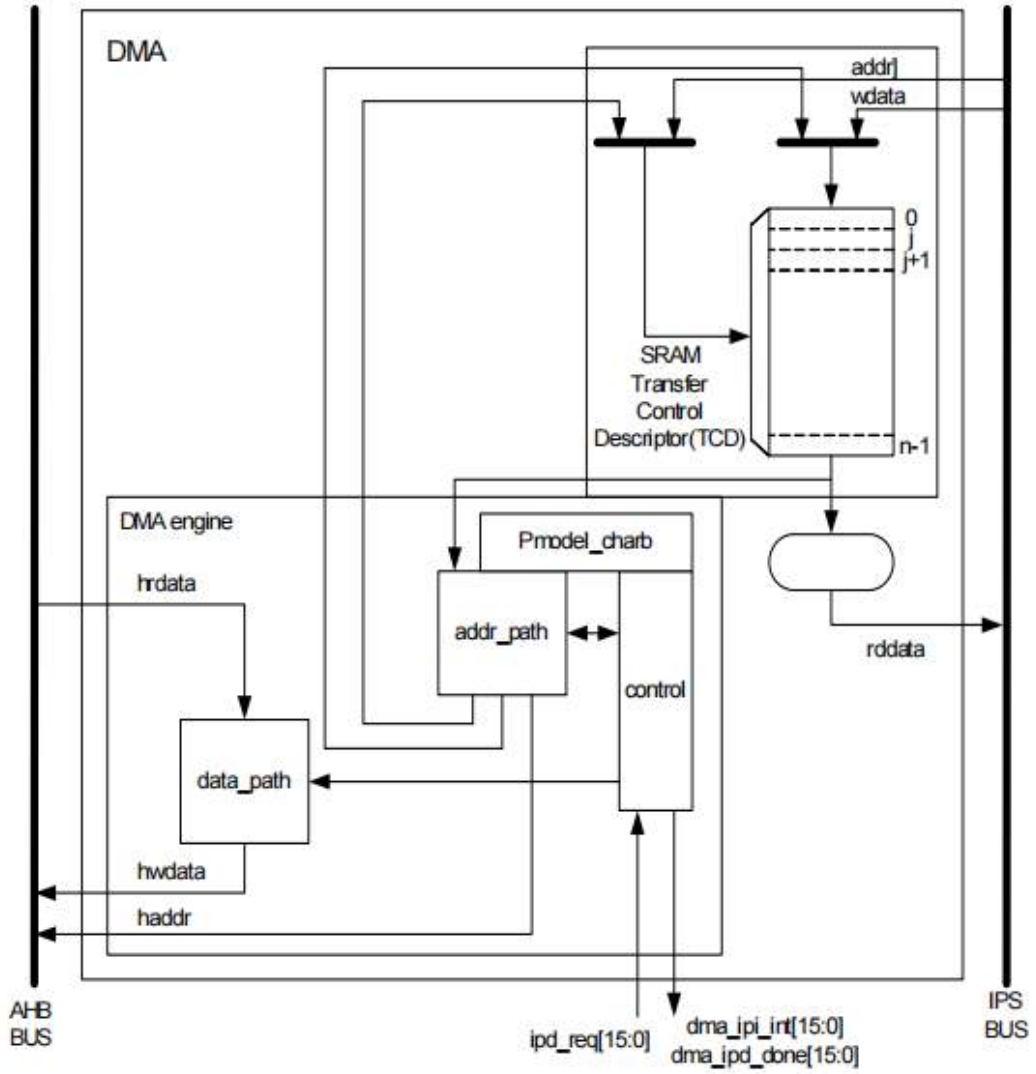


图 13-1：DMA 方块图

13.2.1. 特性

DMA 模块具有以下特性：

- 所有数据移动均通过双地址传输：从源读取，写入目标
  - 可编程源地址、目标地址和传输大小，以及对增强型寻址模式的支持

- 传输控制描述符的组织方式支持两层嵌套的传输操作
  - 由“次要”字节传输计数定义的内部数据传输循环
  - 由“主要”迭代次数定义的外部数据传输循环

- 通过以下三种方法之一进行信道服务请求：
  - 明确的软件启动
  - 通过通道间链接机制启动连续传输
  小环路和/或大环路末端的独立通道连接
  - 外围设备的硬件请求（每个通道一个）
  - 对于所有三种方法，每次执行子循环时都需要一个服务请求。

- 支持固定优先级和轮询式信道仲裁

- 通过可选中断请求报告通道完成
  - 每个通道一个中断，可选地在主要迭代计数完成后断言
  - 可按通道选择性启用错误终止功能，这些错误终止在逻辑上进行汇总，形成少量错误中断输出

- 支持分散/集中 DMA 处理

- 支持复杂数据结构

- 支持通过软件或硬件取消传输

14. 选项字节（OPB）

14.1. 寄存器内存映射

表 14-1：寄存器内存映射

地址偏移量	Bit[31:24]	Bit[23:16]	Bit[15:8]	Bit[7:0]	访问权限 [1,2]
0x001C	CCR		PVDC		S
0x0020	PLLOCKCR		EIOSCST		S
0x0024	PVDFEVR		RFEVR		S
0x002C	Reserved[3]				
0x0030	IOSTC				
0x0034	LDOTCR	VREFTCR	S		S
0x0038	RTCTCR		EOSCTCR		S

提示：

1. S = 仅限于 CPU 监控程序模式访问。
2. 用户模式对仅限于监督程序的地址位置的访问没有影响，并导致循环终止传输错误。
3. 写入保留地址位置没有效果，读取返回 0。

14.1.1. 寄存器说明

14.1.1.1. PVDC — 可编程电压探测器配置寄存器

偏移地址：0x001C

	15	14	13	12	11	10	9	8
Read:	PVDF	PVD-PORRE	PVDTEST[1:0]		PVDOE	PVDRE	PVDIE	PVDE
Write:								
RESET:	0	0	0	0	1	0	0	1

	7	6	5	4	3	2	1	0
Read:	0	0	0	0	0	0	PVDC[1:0]	
Write:								
RESET:	0	0	0	0	0	0	1	0

= Writes have no effect and the access terminates without a transfer error exception.

图 14-1：PVDC-可编程电压探测器配置寄存器

PVDTEST[1:0] — 写入访问启用序列输入

PVDC 寄存器不能随意改变，除非按照正确的序列写入。正确的顺序是：2'b01 → 2'b10 →

2‘b11。按照这个顺序写入这两个位之后，这两个位的值就等于 2’b11，这时 PVDC 寄存器就可以随意改变了。只有写入 2‘b00 才能清除这些位，当值等于 2’b11 时。写入其他值是没有作用的，返回 2‘b11。

PVDF — 可编程电压探测器标志

PVDF 指示 VDD 低于 PVD 阈值。POR 可清除该阈值。

1 = VDD33 低于 PVD 阈值

0 = VDD33 不小于 PVD 阈值

PVDOE — 可编程电压探测器输出使能

1 = PVD 输出使能

0 = PVD 输出禁用

PVDE — 可编程电压探测器启用

如果启用 PVD，则为 PVDE 配置。POR 可清除该配置。

1 = 启用 PVD

0 = 禁用 PVD

PVDPORRE — 程序电压检测器（PVD）上电复位使能

PVDPORRE 显示程序电压探测器上电复位是否启用，当启用时，PVD 复位后 RSR[POR] 标志将被设置。

1 = 当 VDD33 低于 PVD 阈值时，PVD 将产生上电复位

0 = 当 VDD33 低于 PVD 阈值时，PVD 不会产生上电复位

PVDRE — 程序电压探测器（PVD）复位使能

PVDRE 显示程序电压检测器复位是否启用，当启用且 PVDPORRE 处于禁用状态时，在 PVD 复位后 RSR[PVDF] 标志将被设置。

1 = PVD 复位启用

0 = 禁用 PVD 重置

PVDIE — 可编程电压探测器中断使能

如果 VCC 低于 PVD 阈值，PVDRE 配置可生成中断

1 = 当 VCC 低于 PVD 阈值时，将生成中断

0 = VCC 低于 PVD 阈值，不产生中断

PVDC[1:0] — 可编程电压探测器配置值。

表 14-2：PVDC 设置表

PVDC	检测电压
2’ b00	2.16V
2’ b01	2.32V
2’ b10	2.48V
2’ b11	2.64V

14.1.1.2. CCR — 客户配置寄存器

偏移地址：0x001E

	31	30	29	28	27	26	25	24
Read:	0	0	RTC_INTERFACE_EN	0	0	0	RTC_INTERFACE_LVD_ISOEN	0
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	CLKMD	0	0	0	0	0	CCRTEST[1:0]	
Write:								
RESET:	Note1	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

- *Note:1, Determined by the value of CCR[CLKMD]*

图 14-2：CCR — 客户配置寄存器

CCRTEST[1:0] — 写入访问启用序列输入

CCR 寄存器的可写位不能随意更改，除非按照正确的二进制序列写入。正确的写入顺序是：2‘b01 → 2’b10 → 2‘b11。按照这个顺序写入这两个位后，这两个位的值将变为 2’b11，此时 CCR 寄存器的可写位就可以随意更改了。只有当写入 2‘b00 时，这些位才会被清零，因为此时寄存器的值等于 2’b11。其他任何写入操作都不会改变状态，只会使寄存器保持 2‘b11 的值。

RTCINTERFACE_EN — 此位确定当 PRCSR[Dir] 或 PRENR[RTC_EN_Dir] 被设置时 RTC 模拟模块是否可以重新配置。

1 = 可重新配置 RTC 模拟模块。

0 = 无法重新配置 RTC 模拟模块。

RTCINTERFACE_LVD_ISOEN — 此位确定当程序电压检测器事件发生时 RTC 模拟接口是否隔离。

1 = 当发生低电压时，RTC 模拟接口不会被隔离。

0 = 当发生低电压时，RTC 模拟接口将被隔离。

CLKMD — 时钟模式控制位

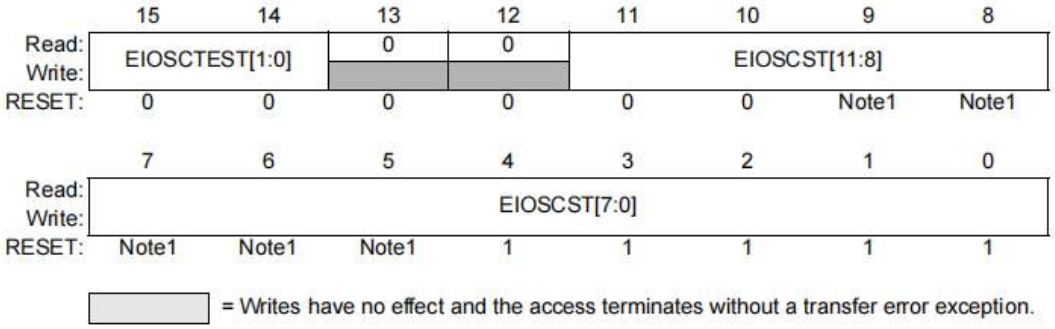
CLKMD 反映芯片的时钟源，由外部 CLKMODE 引脚决定。

表 14-3：时钟模式设置表

CLKMD	Description
1’b1	FIRC150MHz
1’b0	FXOSC

14.1.1.3. EIOSCST — 外部或内部高速振荡器稳定时间配置寄存器

偏移地址：0x0020



Note:1, Determined by the value of CLKMD *bit*

图 14-3：EIOSCST — 外部或内部高速振荡器稳定时间配置寄存器

EIOSCTEST[1:0] — 写入访问启用序列输入

EIOSCST 寄存器的可写位不能随意改变，除非按照正确的写入顺序进行。正确的写入顺序是：2‘b01 → 2’b10 → 2‘b11。按照这个顺序写入这两个位后，这两个位的值为 2’b11，此时 EIOSCST 寄存器的可写位就可以随意改变了。只有当写入 2‘b00 时，这些位才会被清零，因为此时寄存器的值等于 2’b11。其他值的写入不会产生任何影响，寄存器的值仍然会保持 2‘b11。

EIOSCST[11:0] — 外部或内部高速振荡器稳定时间值。

在外部或内部振荡器被启用后，它将等待 EIOSCST[11:0] 个周期的 128KHz 振荡器，然后时钟门电路将打开。当 CLKMD 被设置时，其默认值为 0x1F，否则为 0x3FF。

14.1.1.4. PLLOCKCR — PLL 锁定时间配置寄存器

偏移地址：0x0022

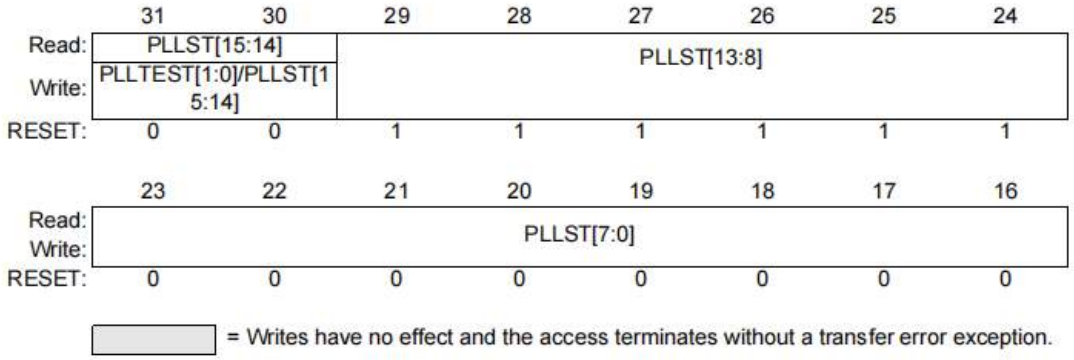


图 14-4：PLLOCKCR — PLL 锁定时间配置寄存器

PLLTEST[1:0] — 写入访问使能序列输入

PLLST 寄存器不能随意改变，除非按照正确的写入顺序进行。正确的写入顺序是：2‘b01 → 2’b10 → 2‘b11。按照这个顺序写入这两个位之后，这两个位的值为 = = 2’b11，此时就可以随意改变 PLLST 寄存器了。任何写入操作都可以清除这两个位，只要它们的值等于 2‘b11 即可。

PLLST[15:0] — PLL 锁定时间值。

启用后，PLL 将等待 PLLST[15:0] 个周期的外部高速振荡器（FXOSC）信号，然后时钟门极将被打开。

提示：只有当 PLLTEST[1:0] 的值等于 2‘b11 时，PLLST[15:14] 才可写入。

14.1.1.5. RFEVR — 重置引脚滤波器启用和值寄存器

偏移地址：0x0024

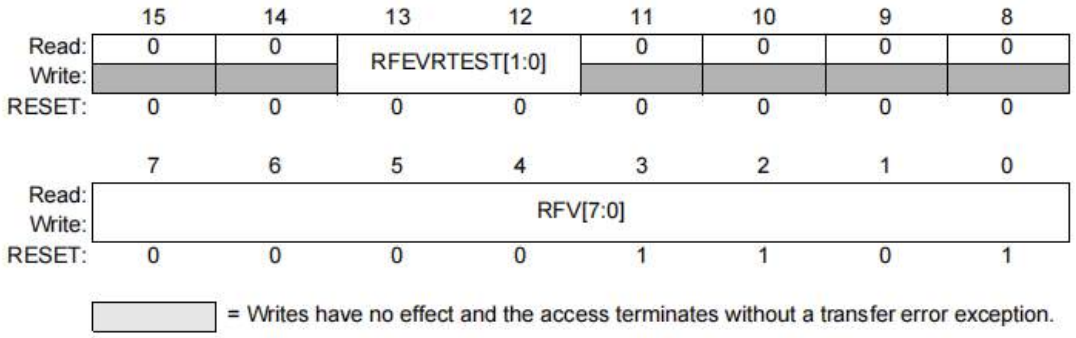


图 14-5：RFEVR-RESET 引脚滤波器使能和值寄存器

PVDFTEST[1:0] — 写入访问启用序列输入

RFEVR 寄存器的可写位不能随意更改，除非按照正确的序列写入。正确的写入顺序是：2‘b01 → 2’b10 → 2‘b11。按照这个顺序写入这两个位后，这两个位的值将变为 = = 2’b11，此时 RFEVR 寄存器的可写位就可以自由修改了。只有当写入 2‘b00 时，才能清除这两个位，因为此时寄存器的值等于 2’b11。其他任何写入操作都不会有效，只会让寄存器保持 2‘b11 的状态。

RFV[7:0] — RESET 引脚滤波器值

这些位决定了外部 RESET 引脚的最小脉冲宽度。

14.1.1.6. PVDFEVR — 可编程电压探测器滤波器启用和数值寄存器

偏移地址：0x0026

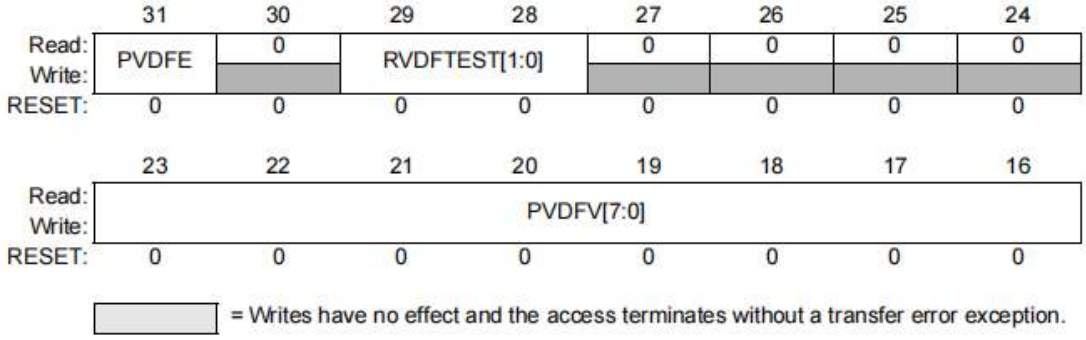


图 14-6：PVDFEVR-可编程电压探测器滤波器启用和数值寄存器

PVDFTEST[1:0] — 写入访问启用序列输入

PVDFEVR 寄存器的可写位不能随意更改，除非按照正确的序列写入。正确的写入顺序是：2'b01 → 2'b10 → 2'b11。按照这个顺序写入这两个位后，这两个位的值为 = = 2'b11，此时 PVDFEVR 寄存器的可写位就可以随意更改了。只有当写入 2'b00 时，这些位才会被清零，因为此时寄存器的值等于 2'b11。其他值的写入不会产生任何影响，返回的仍然是 2'b11。

PVDFE — 可编程电压探测器滤波器启用。

PVDFE 显示是否启用了程序电压探测器过滤器。

1 = 启用 PVD 过滤器

0 = 未启用 PVD 过滤器

PVDFV[7:0] — 可编程电压探测器滤波器值

这些比特确定内部可编程电压检测器的最小脉冲宽度。

14.1.1.7. IOSCTC — 内部高速振荡器微调配置寄存器

偏移地址：0x0030

	31	30	29	28	27	26	25	24
Read:	IOSCTCTEST[1:0]		0	0	0	0		
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	0	0	0	0	0	0	0	CTRIM_LV [1]
Write:								
RESET:	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8
Read:	CTRIM_LV [0]	CTRIM_TD_LV[6:0]						
Write:								
RESET:	1	0	0	0	0	1	1	1

	7	6	5	4	3	2	1	0
Read:	FTRIM_LV[4:0]					TEMPTRIM_LV[2:0]		
Write:								
RESET:	0	1	1	1	1	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

图 14-7：IOSCTC — 内部高速振荡器微调配置寄存器

IOSCTCTEST[1:0] — IOSCTC 写入访问序列输入

IOSCTC 寄存器的可写位不能随意更改，除非按照正确的序列写入。正确的写入顺序是：2‘b01 → 2’b10 → 2‘b11。按照这个顺序写入这两个位后，这两个位的值为 2’b11，此时 IOSCTC 寄存器的可写位就可以随意更改了。只有当写入 2‘b00 时，才能清除这两个位，因为此时寄存器的值等于 2’b11。其他值的写入不会产生任何影响，只会返回 2‘b11。

CTRIM_LV[1:0] — 内部高速振荡器的粗调值

CTRIM_TD_LV[4:0] — 内部高速振荡器的粗调值

FTRIM_LV[6:0] — 内部高速振荡器的精细调整值

TEMPTRIM_LV[2:0] — 内部高速振荡器的温度校正。

14.1.1.8. VREFTCR — VREF1P2 微调配置寄存器

偏移地址：0x0036

	23	22	21	20	19	18	17	16
Read: Write:	VREFTCTEST[1:0]		VREFPD	VREFTRIM[4:0]				
RESET:	0	0	1	1	1	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

- *Note:1, Determined by the value of FUSE Area*

图 14-8：VREFTCR-VREF 微调配置寄存器

VREFTCTEST[1:0] — WSFTCR 写入访问序列输入

VREFTCTEST 寄存器的可写位无法直接修改，除非按照正确的二进制序列写入。正确的写入顺序是：2‘b01 → 2’b10 → 2‘b11。按照此顺序写入这两个位后，其值将变为 2’b11，此时 VREFTCTEST 寄存器的可写位即可自由修改。只有当写入 2‘b00 时，这些位才会被清除，此时寄存器值等于 2’b11。其他写入值
无效，返回 2‘b11。

VREFPD — 内部 VREF1P2 模块断电。设置时间是 10us 当内部 VREF1P2 模块已打开后。

1 = 内部 VREF1P2 模块断电

0 = 内部 VREF1P2 模块通电

VREFTRIM[4:0] — VREF1P2 模块校正值

14.1.1.9. LDO1P2TC-LDO1P2 调整配置寄存器

偏移地址：0x0037

	31	30	29	28	27	26	25	24
Read:	LDO1P5TC[7:6]		LDO1P5TC[5:0]					
Write:	LDO1P2TEST[1:0]/ LDO1P5TC[7:6]							
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

- *Note:1, Determined by the value of FUSE Area*

图 14-9：LDO1P5TC-LDO1P2 微调配置寄存器

LDO1P2TEST[1:0] — LDO1P2TC 写入访问序列

LDO1P2TC 寄存器的可写位不能随意改变，除非按照正确的写入顺序进行。正确的写入顺序是：2‘b01 → 2’b10 → 2‘b11。按照这个顺序写入这两个位之后，这两个位的值为 2’b11，此时 LDO1P2TC 寄存器的可写位就可以随意改变了。任何写入操作都可以清除这两个位，只要它们的值等于 2‘b11 即可。

LDO1P2TC[5:0] — LDO1P2 校正值

LDO1P2TC[6] — LDO1P2 电源关闭控制位

1 = LDO1P2 将被断电。

0 = LDO1P2 将通电。

LDO1P2TC[7] — LDO 过电流限值功能控制位

1 = LDO 过电流限制功能将被关闭。

0 = LDO 过电流限制功能将被打开。

提示：只有当 LDO1P2TEST[1:0] 的值等于 2‘b11 时，才能对 LDO1P2TC[7:6] 进行写入。

14.1.1.10. RTCTCR — RTC 调整配置寄存器

偏移地址：0x003A

	31	30	29	28	27	26	25	24
Read:	RTCTCTEST[1:0]		0	0	0	0	RTC_TEST_MODE	Trim_load
Write:								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
Read:	RTC_BIAS_TRIM[3:0]				RTC_CAP_TRIM[3:0]			
Write:								
RESET:	0	1	1	1	1	1	0	0

= Writes have no effect and the access terminates without a transfer error exception.

- *Note:1, Determined by the value of FUSE Area*

图 14-10：RTCTCR-RTC 微调配置寄存器

RTCTCTEST[1:0] — RTCTC 写入访问序列输入

RTCTC 寄存器的可写入位不能随意更改，除非按照正确的序列写入。正确的写入顺序是：2‘b01 → 2’b10 → 2‘b11。按照这个顺序写入这两个位后，这两个位的值将变为 2’b11，此时 RTCTC 寄存器的可写入位就可以自由修改了。只有当写入 2‘b00 时，才能清除这两个位，因为此时寄存器的值等于 2’b11。其他任何写入操作都不会产生效果，只会使寄存器保持 2‘b11 的状态。

RTC_BIAS_TRIM[3:0] — RTC 振荡器偏置调整配置位。

RTC_CAP_TRIM[3:0] — RTC 振荡器负载电容 C1 和 C2 的微调配置位。

TrimLoad — 从低到高边将 RTC_BIAS_TRIM 和 RTC_CAP_TRIM 值锁存到 RTC 模拟模块中。

RTC_TEST_MODE — RTC 测试模式启用。

1 = RTC 测试模式已启用。

0 = 禁用 RTC 测试模式。

14.1.1.11. EOSCTCR — 外部振荡器边缘配置寄存器

偏移地址：0x0039

	15	14	13	12	11	10	9	8
Read: Write:	EOSCTCTEST[1:0]					DS2	DS1	DS0
RESET:	0	0	0	0	0	1	0	0

= Writes have no effect and the access terminates without a transfer error exception.

- *Note:1, Determined by the value of FUSE Area*

图 14-11：EOSCTCR — 外部振荡器微调配置寄存器

EOSCTCTEST[1:0] — EOSCTCR 写入访问序列输入

EOSCTCR 寄存器的可写位不能随意更改，除非按照正确的序列写入。正确的写入顺序是：2'b01 → 2'b10 → 2'b11。按照这个顺序写入这两个位后，这两个位的值为 2'b11，此时 EOSCTCR 寄存器的可写位就可以随意更改了。只有当写入 2'b00 时，才能清除这两个位，因为此时寄存器的值等于 2'b11。其他值的写入不会产生任何影响，只会返回 2'b11。

EOSCTRIM[2:0] — 外部振荡器微调值

表 14-4：不同驱动强度下振荡器单元的 GM 值

Driving Select			GM (mA/V)
DS2	DS1	DS0	
0	0	0	3.6
0	0	1	6.9
0	1	0	10.3
0	1	1	13.2
1	0	0	16.1
1	0	1	18.9
1	1	0	21.5
1	1	1	24.3

15. 外部总线接口（EBI）

15.1. 介绍

LT165A 具有一个嵌入式外部总线接口（EBI），用于控制内部总线和外部 8 位 MCU 显示面板之间的信息传输。有一个异步芯片选择通道可用。

15.2. 特性

MCU 显示面板接口的特性包括：

- 系统复杂性降低
  - 如果使用芯片选择器，则典型系统不需要外部胶合逻辑。
- 一个可编程异步低电平有效芯片选择器。
- 可编程芯片选择等待周期
  - 最多可对 64 个芯片选择权置周期进行编程，以便与各种面板连接。
- 可编程读写断言周期
- 可编程读/写否定循环

15.3. 信号描述

表 15-1 提供了下文所述信号的概述。

表 15-1：信号特性

姓名	功能	上拉
EBI_D[7:0]	数据总线	—
EBI_WR#	写入使能和低活动	Active
EBI_RD#	读取使能和低活动	Active
EBI_RS	命令或数据指示	Active
EBI_CS#	芯片选择和低活动	Active

15.4. 模块内存映射

表 15-2 显示了 EBI 模块的寄存器内存映射：。

表 15-2：寄存器内存映射

地址偏移量	Bit[31:16]	Bit[15:0]	访问权限 [1,2]
0x0000	EBICR0 — EBI 控制寄存器 0		S
0x0004	EBICR1 — EBI 控制寄存器 1		S

提示：

1. S = 仅限于 CPU 监控程序模式访问。
2. 用户模式下对仅限监督程序访问的地址位置的访问没有影响，将导致循环终止传输错误。

15.5. 寄存器说明

15.5.1. EBI 控制寄存器 0（EBICR0）

偏移地址：0x0000

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	WS[7:0]							
W								
RESET:	0	0	0	0	0	1	1	1

	7	6	5	4	3	2	1	0
R	PS[1:0]		0	0	0	0	EBI_LIT_ENDIAN	CSEN
W								
RESET:	0	1	0	0	0	0	1	0

= Writes have no effect and the access terminates without a transfer error exception.

图 15-1：EBI 控制寄存器 0（EBICR0）

PS[1:0] — 端口大小位

PS 位定义芯片选择所支持的外部数据端口的宽度为 8 位或 16 位。此字段仅在此设备中

读取。

00 = 不支持，将导致不可预知的错误

10 = 16 位端口

01 = 8 位端口

11 = 不支持，将导致不可预测的错误

EBI_LITEndian — EBI 端口采用小端字序

1 = EBI 端口是小字节序的。

0 = EBI 端口是大字节序的。

WS[7:0] — 等待状态字段

WS 字段决定了芯片选择逻辑在触发内部周期终止信号前需要插入的等待状态数量。每个等待状态对应一个系统时钟周期。若将 WS 设置为零等待状态，则在周期访问开始后的下一个时钟周期触发内部周期终止信号，实现单时钟周期传输。当 WS 配置为一个等待状态时，该信号会在周期访问启动后两个时钟周期才被触发。

由于内部周期终止信号是在经过预定数量的等待状态后由内部发出的，因此软件可以调整总线定时，以适应外部设备的访问速度。最多有 256 种可能的等待状态，即使速度较慢的设备也可以与 MCU 连接。

表 15-3：EBI_CS#芯片选择等待状态编码

WS[7:0]	EBI_CS#循环次数	
	读取权限	写入访问权限
00000000	1	1
00000001	2	2
00000010	3	3
00000011	4	4
00000100	5	5
00000101	6	6
00000110	7	7
00000111	8	8
00001000	9	9
00001001	10	10
00001010	11	11
00001011	12	12
00001100	13	13
00001101	14	14
..	...	...
11111111	256	256

CSEN-芯片选择使能位

CSEN 位启用芯片选择逻辑。当芯片选择功能被禁用时，EBI_CS 信号为高电平否定。

1 = 芯片选择功能启用。

0 = 禁用芯片选择功能。

15.5.2. EBI 控制寄存器 1（EBICR1）

偏移地址：0x0004

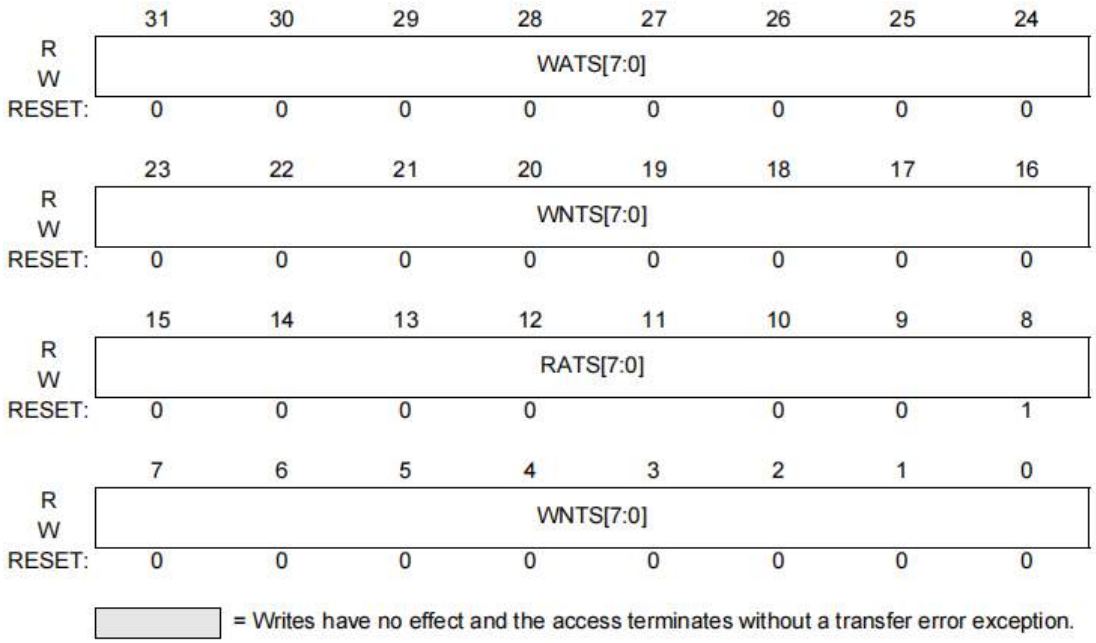


图 15-2：EBI 控制寄存器 1（EBICR1）

WATS[7:0] — 写入信号断言定时选择

这些位选择写入操作时 EBI_WR#的断言时序。有关详细信息，请参阅功能描述。

WNTS[7:0] — 写入信号否定定时选择

这些位选择写入操作时 EBI_WR#的否定时序。有关详细信息，请参阅功能描述。

RATS[7:0] — 读取断言时序选择

这些位选择在读取操作时 EBI_RD#的断言时序。有关详细信息，请参阅功能描述。

RNTS[7:0] — 读取断言时序选择

这些位选择在读取操作时 EBI_RD#的否定时序。有关详细信息，请参阅功能描述。

15.6.功能说明

15.6.1. EBI_WR#、EBI_RD# 信号时序

对于写入和读取操作，可以配置 EBI_WR# 和 EBI_RD# 的动作时序。

表 15-4：EBI_WR# 和 EBI_RD# 的动作时序（单位：系统周期）

RATS[7:0]/WATS[7:0]	Assert Time (T_{ass})	RNTS[7:0]/WNTS[7:0]	Negate Time (T_{neg})
00000000	1/2	00000000	1/2
00000001	2/2	00000001	2/2
00000010	3/2	00000010	3/2
00000011	4/2	00000011	4/2
00000100	5/2	00000100	5/2
...	...	...	...
11111110	2555/2	11111110	255/2
11111111	256/2	11111111	256/2

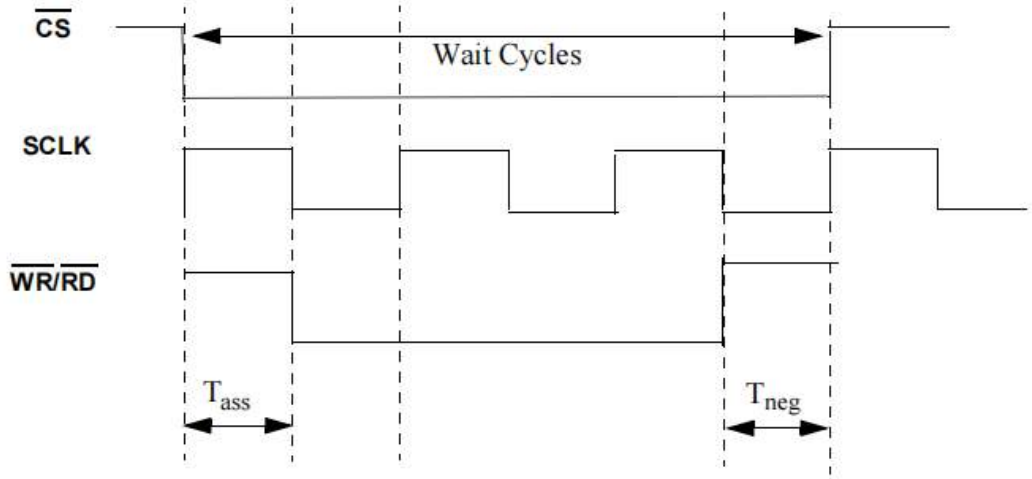


图 15-3：EBI_WR# 和 EBI_RD# 的动作时序

15.6.2. EBI 功能描述

15.6.2.1. 芯片选择

芯片选择器可为 MCU 面板提供芯片使能信号，并断言内部总线周期终止信号。

在 EBICR 中设置 CSEN 位，使芯片选择能够提供面板芯片使能信号。

主动芯片选择（CSEN）、等待状态（WS）字段和端口大小（PS）字段的配置用于访问。

表 15-5：芯片选择地址范围编码

芯片选择	区块大小	正常模式地址范围
EBI_CS#	1MB	0x2000_0000 - 0x2FFF_FFFF

15.6.2.2. operands 传输

内部 AHB 总线的可能操作访问为：

- 字节
- 上半字对齐
- 对齐的下半字
- 对齐的单词

系统不支持非对齐传输。EBI 模块负责控制 AHB 总线与 8 位或 16 位端口之间的字节、半字或字操作数传输。此处的“端口”特指外部设备在数据传输时使用的数据路径宽度，每个端口对应数据总线上的特定引脚位：对于每个芯片选择信号，D[31:24] 引脚对应 8 位端口，而 D[31:16] 引脚则对应 16 位端口。

表 15-6 至表 15-7 展示了 EBI_CS#的各可能传输尺寸、对齐方式及端口宽度。表中所示的数据字节代表外部数据引脚，这些数据通过多路复用方式驱动至外部数据总线（如图所示）。标有破折号的数据字节并非必需：在读取传输时总线会忽略这些字节，在写入传输时则会用未定义数据进行驱动。

表 15-6：EBI_CS#的 8 位端口数据传输（大端）

Transfer Size	Port Width	内部地址		外部引脚	数据总线传输			
		位 1	位 0	EBI_RS				
Byte	8	0	0	0	D[15:8]	-	-	-
		0	1	1	-	D[15:8]	-	-
		1	0	0	-	-	D[15:8]	-
		1	1	1	-	-	-	D[15:8]
Half-word	8	0	0	0	D[15:8]	-	-	-
				1	-	D[15:8]	-	-
		1	0	0	-	-	D[15:8]	-
				1	-	-	-	D[15:8]
Word	8	0	0	0	D[15:8]	-	-	-
				1	-	D[15:8]	-	-
				0	-	-	D[15:8]	-
				1	-	-	-	D[15:8]

表 15-7：EBI_CS#的 16 位端口数据传输（大端）

Transfer Size	Port Width	内部地址		外部引脚	数据总线传输			
		位 1	位 0	EBI_RS				
Byte	16	0	0	0	D[15:8]	-	-	-
		0	1	0	-	D[7:0]	-	-
		1	0	1	-	-	D[15:8]	-
		1	1	1	-	--	-	D[7:0]
Half-word	16	0	0	0	D[15:8]	D[7:0]	-	-
		1	0	1	-	-	D[15:8]	D[7:0]
Word	16[1]	0	0	0	D[15:8]	D[7:0]	-	-
				1	-	-	D[15:8]	D[7:0]

提示：

1. EBI 对 16 位端口的单词访问运行两个周期。该表显示了两个总线周期的数据放置情况。

16. 可编程中断计时器模块（PIT）

16.1. 介绍

可编程中断计时器（PIT）是一个 16 位计时器，它提供精确的中断，以固定的间隔进行，并且对处理器的干预最小。该计时器可以基于模锁存器中写入的值进行倒计数，也可以作为一个自由运行的递减计数器。

16.2. 方块图

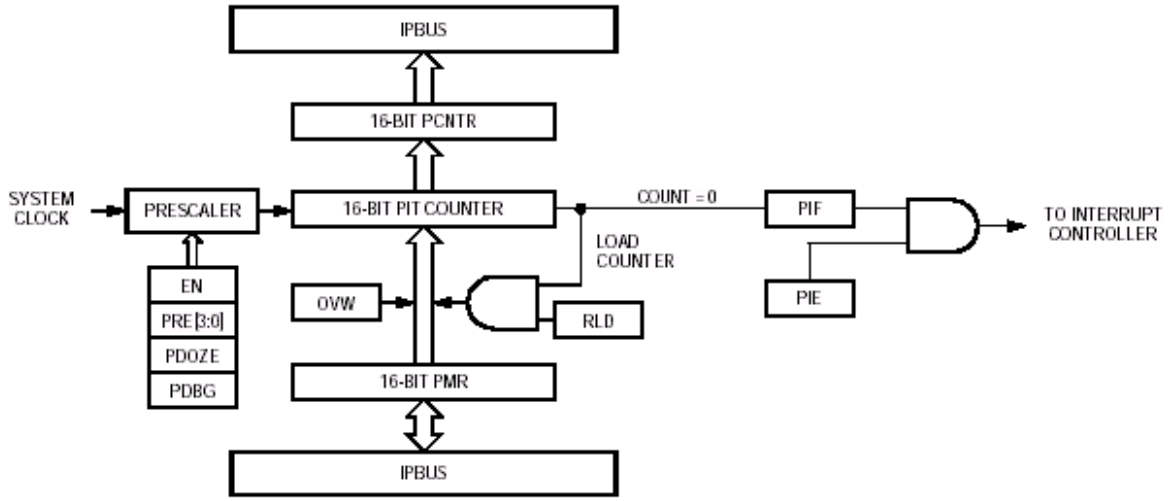


图 16-1：PIT 方块图

16.3. 操作模式

本小节描述了三种低功耗模式和调试模式。

16.3.1. 等待模式

在等待模式下，PIT 模块继续正常运行，并且可以配置为通过生成中断请求退出低功耗模式。

16.3.2. Doze 模式

当 PIT 控制和状态寄存器（PCSR）中的 PDOZE 位被设置为 12 时，PIT 模块停止运行。若 PDOZE 位被清除，则 Doze 模式不会影响 PIT 的正常工作。退出 Doze 模式后，PIT 将从进入该模式前的状态继续运行。

16.3.3. 停止模式

在停止模式下，系统时钟不存在，PIT 模块停止运行。

16.3.4. 调试模式

在调试模式下，如果 PCSR 中设置了 PDBG 位，则 PIT 模块操作停止。在调试模式下，如果 PDBG 位清除，则调试模式不会影响 PIT 操作。退出调试模式时，PIT 操作从进入调试模式前的状态继续运行，但调试模式中所做的任何更新仍然有效。

16.4. 信号

PIT 模块没有片外信号。

16.5. 内存映射和寄存器

本小节描述了 PIT 的内存映射和寄存器结构。

16.5.1. 内存映射

关于内存映射的描述，请参见表 16-1。该设备具有四个可编程中断计时器。

表 16-1：可编程中断计时器模块内存映射

PIT1 偏移地址	Bit[15:8]	Bit[7:0]	访问权限 [1]
0x0000	PIT 模数寄存器（PMR）		S
0x0002	PIT 控制和状态寄存器（PCSR）		S
0x0004	未实现(2)		—
0x0006	PIT 计数寄存器（PCNTR）		S/U

提示：

1. S = 仅允许访问 CPU 监控模式。S/U = CPU 监控或用户模式访问。用户模式访问仅限于监控器的地址，否则将不会产生任何影响，并导致周期终止传输错误。
2. 访问未实现的地址位置没有效果，并导致循环终止传输错误。

16.5.2. 寄存器

PIT 编程模型由以下寄存器组成：

- PIT 控制和状态寄存器（PCSR）配置计时器的操作。请参见 16.5.2.1 PIT 模数寄存器。
- PIT 模数寄存器（PMR）确定计时器模数重置值。请参阅 16.5.2.3 PIT 计数寄存器。
- PIT 计数寄存器（PCNTR）提供对计数器值的可见性。请参阅 16.5.2.3 PIT 计数寄存器。

16.5.2.1. PIT 模数寄存器

16 位读写 PIT 模数寄存器（PMR）包含计时器模数值，当计数达到 0x0000 且 RLD 位被设置时，该值将被加载到 PIT 计数器中。

当 OVW 位被设置时，PMR 是透明的，写入 PMR 的值会立即加载到 PIT 计数器中。预分频器计数器会在新值加载到 PIT 计数器时以及复位期间重置。读取 PMR 返回模数锁存器中写入的值。复位将 PMR 初始化为 0xFFFF。

偏移地址：0x0000

	15	14	13	12	11	10	9	8
R W	PM15	PM14	PM13	PM12	PM11	PM10	PM9	PM8
RESET:	1	1	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
R W	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
RESET:	1	1	1	1	1	1	1	1

图 16-2：PIT 模量寄存器（PMR）

16.5.2.2. PIT 控制和状态寄存器

偏移地址：0x0002

	15	14	13	12	11	10	9	8
R W	0	0	0	0	PRE3	PRE2	PRE1	PRE0
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R W	PDMAE	PDOZE	PDBG	OVW	PIE	PIF	RLD	EN
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 16-3：PIT 控制和状态寄存器（PCSR）

PRE[3:0] — 预分频器位

如表 16-2 所示，通过读写 PRE[3:0] 位选择用于生成 PIT 时钟的系统时钟分频器。

要准确预测下一次计数的时机，仅在使能位（EN）清除时修改 PRE[3:0] 位。修改 PRE[3:0] 会重置预分频计数器。系统复位和向计数器加载新值时也会重置预分频计数器。在同一个写入周期内，可以同时设置 EN 位并写入 PRE[3:0]。清除 EN 位会停止预分频计数器的运行。

表 16-2：预分频器选择编码

PRE[3:0]	IPS 时钟除数
0000	1
0001	2
0010	4
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1,024
1011	2,048
1100	4,096
1101	8,192
1110	16,384
1111	32,768

PDMAE — DMA 使能控制位

读/写 PDMAE 位控制 PIT 计数器达到 0x0000 时是否生成 dma 请求。

1 = 当 PIT 计数器达到 0x0000 时，将生成 Dma 请求。

0 = 当 PIT 计数器达到 0x0000 时，将不会生成 Dma 请求。

PDOZE — Doze 模式比特

PDOZE 读/写位控制 PIT 在十二进制模式下的功能。复位清除 PDOZE。

1 = PIT 功能在 Doze 模式下停止

0 = 在 Doze 模式下 PIT 功能不受影响

退出休眠模式时，计时器操作从进入休眠模式前的状态继续运行。

PDBG — 调试模式位

读写 PDBG 位控制调试模式下 PIT 的功能。复位清除 PDBG。

1 = PIT 功能在调试模式下停止

0 = 在调试模式下，PIT 功能不受影响

在调试模式下，寄存器读写访问正常工作。退出调试模式后，计时器操作从进入调试模式前的状态继续运行，但调试模式下所做的任何更新仍然有效。

提示：在调试模式下，将 PDBG 位从 1 更改为 0 会启动 PIT 计时器。同样，在调试模式下，将 PDBG 位从 0 更改为 1 会停止 PIT 计时器。

OVW — 重写位

读/写 OVW 位允许向 PMR 写入数据，以便立即覆盖 PIT 计数器中的值。

1 = 写入 PMR 会立即替换 PIT 计数器中的值。

0 = 当计数达到 0x0000 时，PMR 中的值将取代 PIT 计数器中的值。

PIE-PIT 中断使能位

读/写 PIE 位使 PIF 标志能够生成中断请求。

1 = PIF 中断请求已启用

0 = 禁用 PIF 中断请求

PIF-PIT 中断标志

当 PIT 计数器达到 0x0000 时，读写 PIF 标志被设置。通过向其写入 1、写入 PMR 或通过 dma 请求确认来清除 PIF。写入

0 无影响。复位清除 PIF。

1 = PIT 计数达到 0x0000。

0 = PIT 计数未达到 0x0000。

RLD — 重载位

读写 RLD 位使计数器数值达到 0x0000 时，能够将 PMR 值加载到 PIT 计数器中。

1 = 从 PMR 重新加载计数器，计数值为 0x0000

0 = 计数为 0x0000 时，翻转至 0xFFFF

EN — PIT 启用位

读写 EN 位启用 PIT 操作。当 PIT 被禁用时，计数器和预分频器保持在停止状态。

1 = 已启用 PIT

0 = 禁用 PIT

16.5.2.3. PIT 计数寄存器

16 位只读 PIT 控制寄存器（PCNTR）包含计数器值。用两次 8 位读取来读取 16 位计数器不能保证一致。对 PCNTR 的写入没有影响，写入周期将正常终止。

偏移地址：0x0006

	15	14	13	12	11	10	9	8
R	PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8
W								
RESET:	1	1	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
R	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
W								
RESET:	1	1	1	1	1	1	1	1

= Writes have no effect and the access terminates without a transfer error exception.

图 16-4：PIT 计数寄存器（PCNTR）

16.6. 功能说明

本小节描述了 PIT 的功能操作。

16.6.1. 设置并忘记计时器操作

当 PCSR 寄存器中的 RLD 位被置为 1 时，系统将启用该操作模式。当 PIT 计数器达到 0x0000 时，PCSR 寄存器中的 PIF 标志位会被激活。此时，模数锁存器的数值将被加载至计数器，计数器随即开始向 0x0000 方向递减。若 PCSR 寄存器中的 PIE 位处于激活状态，PIF 标志位会向 CPU 发送中断请求。

当 PCSR 中的 OVW 位被设置时，可以通过写入 PMR 直接初始化计数器，而无需等待计数达到 0x0000。

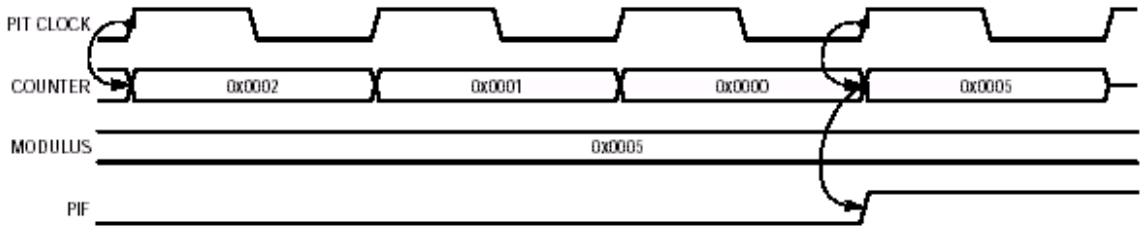


图 16-5：从模块锁存器进行的逆向重装

16.6.2. 自由运行计时器操作

当 PCSR 中的 RLD 位清除时，选择这种操作模式。在此模式下，计数器从 0x0000 滚动到 0xFFFF，而不从模数锁存器中重新加载，并继续递减。

当计数器达到 0x0000 时，PCSR 中设置 PIF 标志。如果 PCSR 中设置 PIE 位，则 PIF 标志向 CPU 发出中断请求。

当 PCSR 中的 OVW 位被设置时，可以通过写入 PMR 直接初始化计数器，而无需等待计数达到 0x0000。

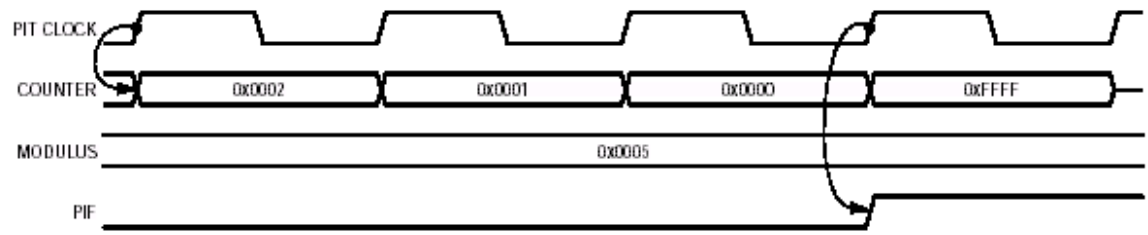


图 16-6：自由运行模式下的计数器

16.6.3. 超时规格

16 位 PIT 计数器和预分频器支持不同的超时周期。预分频器将系统时钟除以 PCSR 中 PRE[3:0] 位所选的值。PMR 中的 PM[15:0] 位选择超时周期。

$$\text{timeout period} = \text{PRE}[3:0] \times (\text{PM}[15:0] + 1) \text{ clocks}$$

16.7. 中断操作

表 16-3 列出了由 PIT 产生的中断请求。

表 16-3：PIT 中断请求

Interrupt Request	Flag	Enable Bit
Timeout	PIF	PIE

当 PIT 计数器达到 0x0000 时，PIF 标志被设置。PIE 位使 PIF 标志能够生成中断请求。通过向其写入 1 或向 PMR 写入来清除 PIF。

17. 看门狗定时器模块（WDT）

17.1. 介绍

看门狗计时器是一个 16 位计时器，用于帮助软件从失控代码中恢复，或在操作运行时间超过预期时发出中断。看门狗计时器具有一个自由运行的递减计数器（看门狗计数器），当发生下溢时会生成复位或中断。为了防止复位，软件必须定期通过服务看门狗来重新启动计数器。

17.2. 操作模式

本小节描述了看门狗定时器在低功耗模式和调试模式下的操作。

17.2.1. 等待模式

在看门狗控制寄存器（WCR）的 wait 位设置为“1”的等待模式下，看门狗定时器停止工作。在看门狗控制寄存器的 wait 位设置为“0”的等待模式下，看门狗定时器继续正常工作。

17.2.2. Doze 模式

当 WCR 寄存器中的 doze 位设置为 12 时，看门狗定时器模块停止工作。若 doze 位保持为 0，则看门狗定时器将继续正常运行。

17.2.3. 停止模式

在 stop 位设置于 WCR 中的停止模式下，看门狗操作在停止模式中停止。退出停止模式后，看门狗操作继续从进入停止模式前的状态运行。在 stop 位清除的停止模式下，看门狗定时器继续正常运行。

17.2.4. 调试模式

在调试模式下，如果 WCR 中设置了 DBG 位，则看门狗定时器模块停止运行。在调试模式下，如果 DBG 位未设置，则看门狗定时器继续正常运行。退出调试模式后，看门狗定时器从进入调试模式前的状态继续运行，但调试模式中所做的任何更新仍然有效。

17.3.17.3 模块示意图

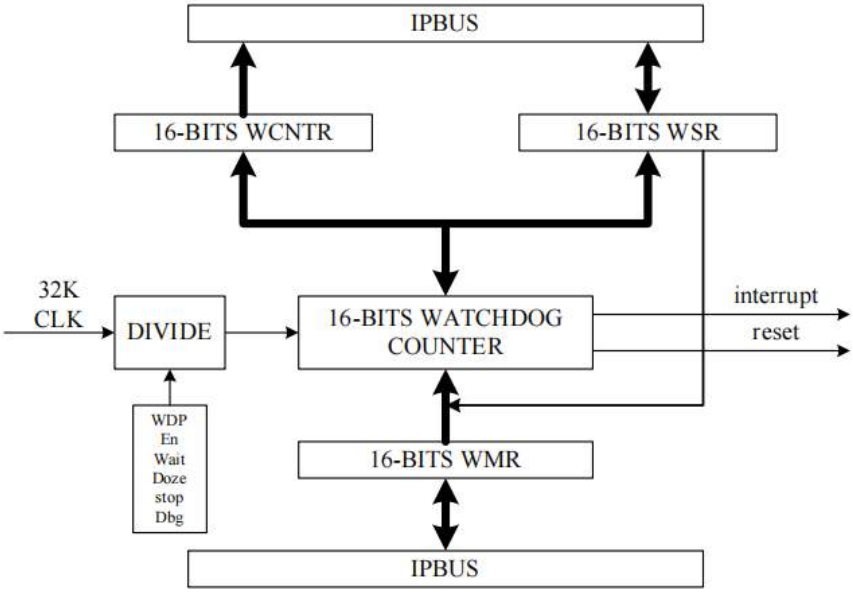


图 17-1：看门狗定时器方块图

17.4.信号

看门狗定时器模块没有片外信号。

17.5.内存映射和寄存器

本小节描述了看门狗计时器的内存映射和寄存器。

17.5.1. 内存映射

有关看门狗内存映射的概述，请参见表 17-1。

表 17-1：看门狗定时器模块内存映射

地址偏移量	Bit[15:8]	Bit[7:0]	访问权限 [1]
0x0000	看门狗模数寄存器（WMR）		S
0x0002	看门狗控制寄存器（WCR）		S
0x0004	监视器服务寄存器（WSR）		S/U
0x0006	看门狗计数器（WCNTR）		S/U

提示：S = 仅允许访问 CPU 监督模式。S/U = CPU 监督或用户模式访问。用户模式访问监督模式专用地址无效，并导致周期终止传输错误。

17.5.2. 寄存器

看门狗定时器编程模型由以下寄存器组成：

- 看门狗控制寄存器（WCR）配置看门狗计时器操作。请参阅 17.5.2.2 看门狗控制寄存器。
- 看门狗模数寄存器（WMR）确定计时器模数重置值。请参阅 17.5.2.4 看门狗计数寄存器。
- 看门狗计数寄存器（WCNTR）提供对看门狗计数器值的可见性。请参见 17.5.2.4 看门狗计数寄存器。
- 看门狗服务寄存器（WSR）需要通过特定服务序列来防止复位。需注意：只读 WC[15:0] 字段反映的是看门狗计数器的当前值。采用两次 8 位读取方式读取 16 位 WCNTR 寄存器时，无法保证返回有效数值。对 WCNTR 寄存器进行写入操作不会产生实际影响，写入周期仍会正常终止。该寄存器专用于看门狗工作域，因此读取值可能尚未稳定，请持续多次读取以确认其稳定性。

17.5.2.1. 监视器模块寄存器

偏移地址：0x0000

	15	14	13	12	11	10	9	8
R W	WM15	WM14	WM13	WM12	WM11	WM10	WM9	WM8
RESET:	0	0	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
R W	WM7	WM6	WM5	WM4	WM3	WM2	WM1	WM0
RESET:	1	1	1	1	1	1	1	1

图 17-2：看门狗模块寄存器（WMR）

WM[15:0] — 看门狗模数字段

WM[15:0] 字段包含一个模数，该模数通过服务序列重新加载到看门狗计数器中。写入 WMR 即会将新的模数值加载到看门狗计数器中。新值也会在下一次及所有后续的重新加载中使用。

读取 WMR 返回模数寄存器中的值。重置将把 WM[15:0] 字段初始化为 0x3FFF。

17.5.2.2. 看门狗控制寄存器

16 位读写看门狗控制寄存器（WCR）配置看门狗计时器操作。

偏移地址：0x0002

	15	14	13	12	11	10	9	8
R	0	0	0	0	WAIT	DOZE	STOP	DBG
W								
RESET:	0	0	0	0	1	1	1	1

	7	6	5	4	3	2	1	0
R	IS	WDP[2:0]			IF	IE	0	EN
W							CU	
RESET:	0	1	1	1	0	1	0	1

= Writes have no effect and the access terminates without a transfer error exception.

图 17-3：看门狗控制寄存器（WCR）

等待 — 等待模式位

WAIT 位控制等待模式下看门狗定时器的功能。重置设置为 WAIT。

1 = 看门狗定时器停止在等待模式

0 = 等待模式下，看门狗定时器不受影响

DOZE — 时尚比特

DOZE 控制 DOZE 模式下看门狗定时器的功能。重置设置 DOZE。

1 = 看门狗定时器在 Doze 模式下停止

0 = 在十二分之一模式下，看门狗定时器不受影响

STOP — STOP 模式位

STOP 位控制看门狗定时器在停止模式下的功能。重置设置 STOP。

1 = 看门狗定时器在停止模式下停止

0 = 在停止模式下，看门狗定时器不受影响

DBG — 调试模式位

DBG 位控制调试模式下看门狗定时器的功能。在调试模式下，可以正常写入和读取看门狗定时器寄存器。退出调试模式后，定时器操作从进入调试模式前的状态继续运行，但调试模式下所做的任何更新仍然有效。

1 = 在调试模式下，看门狗计时器停止

0 = 在调试模式下，看门狗定时器不受影响

提示：在调试模式下，将 DBG 位从 1 更改为 0 会启动看门狗定时器。

EN — 看门狗启用位

EN 位启用看门狗定时器。

1 = 看门狗定时器已启用

0 = 看门狗定时器已禁用

CU — Watchdog 变更更新位

向 CU 写入 1，将 WDP[2:0] 和 WMR 更新到工作锁存器。

IE — 看门狗中断使能位

IE 位启用看门狗定时器中断模式。一旦生成中断且 EN 位为 1，该位将自动清除。

1 = 看门狗定时器中断模式已启用

0 = 看门狗定时器中断模式已禁用

IF — 看门狗中断标志位

写一个 1 到这个位将清除标志。

IS — 看门狗时钟域中断状态位

该位为只读，如果该位为 1‘b1，则看门狗时钟域的状态不会被清除，因此如果 CPU 想要进入睡眠或停止，将会再次唤醒，所以，在 CPU 想要进入睡眠或停止之前，请先检查该位。

WDP[2:0] — 看门狗定时器预分频器

WDP[2:0] 位确定看门狗定时器运行时的预分频值。不同的预分频值及其对应的超时周期如表 17-2 “看门狗定时器预分频器” 所示。

表 17-2：看门狗定时器预分频器

WDP[2:0]	Prescaler
000	128KHz/2048
001	128KHz/1024
010	128KHz/512
011	128KHz/256
100	128KHz/128
101	128KHz/64
110	128KHz/32
111	128KHz/16

17.5.2.3. 监视器服务寄存器

启用看门狗定时器后，在看门狗计数器超时之前，向看门狗服务寄存器（WSR）写入 0x5555 和 0xAAAA 可防止复位。如果在超时之前没有对 WSR 进行服务，看门狗定时器将向复位控制器或中断控制器模块发送信号，并断言系统复位或中断。

两个写入必须按照之前列出的顺序在超时之前发生，但是在两个写入之间可以执行任意数量的指令。但是，向 WSR 写入任何非 0x5555 或 0xAAAA 的值都会重置服务序列，因此需要将这两个值都写入才能防止看门狗计时器导致重置。

偏移地址：0x0004

	15	14	13	12	11	10	9	8
R W	WS15	WS14	WS13	WS12	WS11	WS10	WS9	WS8
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R W	WS7	WS6	WS5	WS4	WS3	WS2	WS1	WS0
RESET:	0	0	0	0	0	0	0	0

图 17-4：看门狗服务寄存器（WSR）

17.5.2.4. 监视器计数器

偏移地址：0x0006

	15	14	13	12	11	10	9	8
R	WC15	WC14	WC13	WC12	WC11	WC10	WC9	WC8
W								
RESET:	0	0	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
R	WC7	WC6	WC5	WC4	WC3	WC2	WC1	WC0
W								
RESET:	1	1	1	1	1	1	1	1

= Writes have no effect and the access terminates without a transfer error exception.

图 17-5：看门狗计数寄存器（WCNTR）

WC[15:0] — 监视器计数字段

只读寄存器 WC[15:0] 字段反映看门狗计数器的当前值。用两次 8 位读取来读取 16 位的 WCNTR 不能保证返回一个连贯的值。对 WCNTR 写入没有任何影响，写入周期会正常终止。该寄存器用于看门狗工作域，所以读取的值可能没有稳定，建议连续多次读取查看。

18. 实时控制器（RTC）

18.1. 介绍

该模块是控制一个具有 RTC 功能的综合 PMU 的控制器，可对 RTC 计数器进行重新配置，设置闹钟以及产生时间/闹钟中断。

18.2. 特性

该模块的主要特性：

– 重新配置 RTC 计数器并读取秒、分钟、小时和天数计数器

– 支持闹钟设置

– 中断源：秒、分钟、小时、日中断，可编程闹钟中断，1KHz/32KHz 周期性中断。

18.3. 测试模式

在测试模式下，我们可以使用 CPU 或芯片引脚配置 RTC，我们可以直接通过芯片引脚检查 RTC 状态和 CLK 信号。

18.4. 方块图

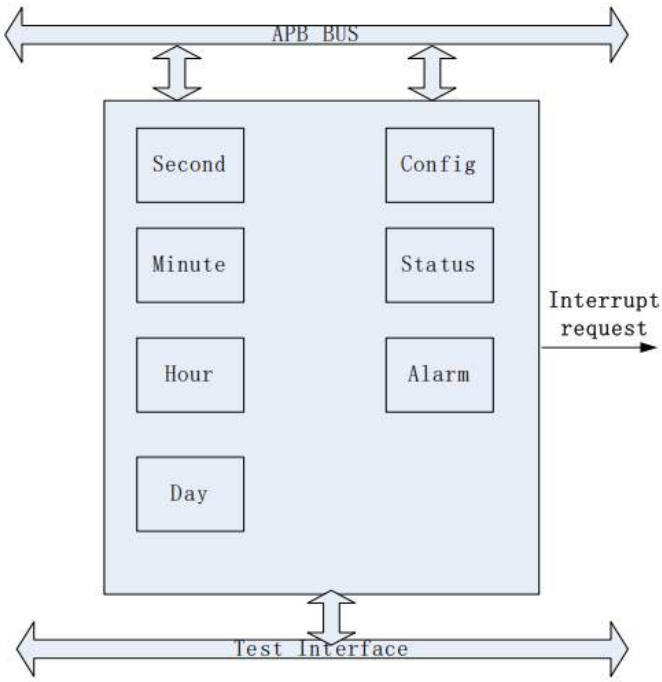


图 18-1：RTC 方块图

19. 边缘端口模块（EPORT）

19.1. 介绍

边沿端口模块（EPORT）具有八个外部中断引脚。每个引脚均可单独配置为低电平敏感中断引脚、边沿检测中断引脚（上升沿、下降沿或同时为两者）、通用输入/输出（I/O）引脚。请参见图 21-1。

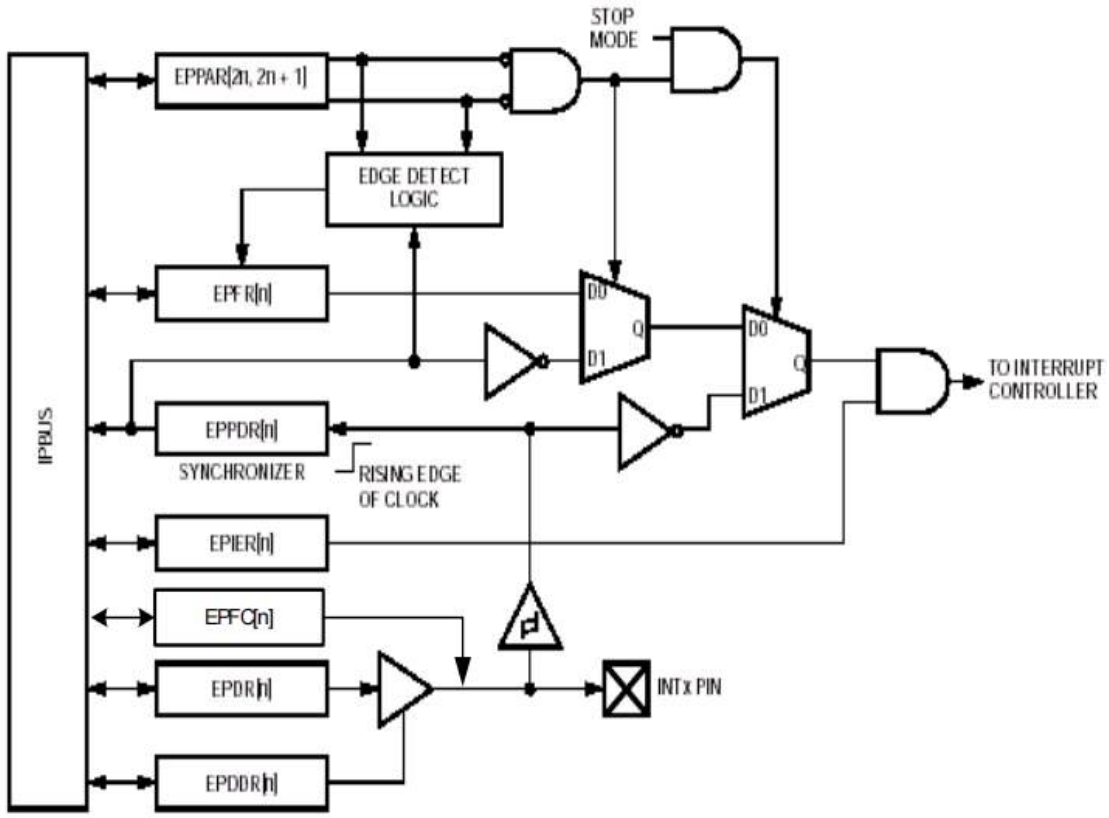


图 19-1：EPORT 方块图

19.2. 低功耗模式操作

本小节描述了 EPORT 模块在低功耗模式下的操作。

19.2.1. 等待和打盹模式

在等待和打盹模式下，EPORT 模块继续正常运行，并且可通过配置使其通过在外部引脚上选定的边沿或低电平生成中断请求来退出低功耗模式。

19.2.2. 停止模式

在停止模式下，没有可用的时钟来执行边沿检测功能。只有电平检测逻辑处于活动状态（如果已配置），以允许外部中断引脚上的任何低电平生成中断（如果已启用），从而退出停止模式。

提示：由于没有可用时钟，电平检测逻辑的输入引脚同步器被绕过。

19.3. 中断/通用 I/O 引脚说明

所有引脚在复位时默认为通用输入引脚。当从 EPORT 引脚数据寄存器（EPPDR）读取时，引脚值会与 CLKOUT 的上升沿保持同步。边缘/电平检测逻辑中使用的数值同样与 CLKOUT 的上升沿保持同步。这些引脚采用施密特触发输入缓冲器，其内置滞后功能可有效降低对上升/下降时间较长的输入信号产生误触发中断的概率。

19.4. 存储器映射和寄存器

本小节描述内存映射和寄存器结构。

19.4.1. 内存映射

有关 EPORT 内存映射的描述，请参见表 21-1。

表 19-1：模块内存映射

地址偏移量	Bit[15:8]	Bit[7:0]	访问权限 [1]
0x0000	EPORT 数据方向寄存器（EPDDR）	EPORT 中断使能寄存器（EPIER）	S
0x0002	EPORT 引脚分配寄存器（EPPAR）		S
0x0004	EPORT 标志寄存器（EPFR）	EPORT 引脚上拉使能寄存器	S/U
0x0006	EPORT 数据寄存器（EPDR）	EPORT 引脚数据寄存器（EPPDR）	S/U
0x0008	EPORT 数字滤波器控制	EPORT 位组寄存器（EPBSR）	S
0x000A	EPORT 电平极性寄存器（EPLPR）	EPORT 开放引流登记表（EPODE）	S
0x000C	已反转		S
0x000E	EPORT Bit Clear Register		S

提示：S = 仅允许访问 CPU 监督模式。S/U = CPU 监督或用户模式访问。用户模式访问监督模式专用地址无效，并导致周期终止传输错误。

19.4.2. 寄存器

EPORT 编程模型包含以下寄存器：

- EPORT 引脚分配寄存器（EPPAR）单独控制每个引脚的功能。
- EPORT 数据方向寄存器（EPDDR）控制每个引脚的方向。
- EPORT 中断使能寄存器（EPIER）可单独为每个引脚启用中断请求。
- EPORT 数据寄存器（EPDR）保存要驱动到引脚的数据。
- EPORT 引脚数据寄存器（EPPDR）反映引脚的当前状态。
- EPORT 标志寄存器（EPFR）单独锁存 EPORT 边沿事件。
- EPORT 引脚上拉使能寄存器（EPPUE）控制每个引脚的上拉。
- EPORT 电平极性寄存器（EPLPR）控制每个电平敏感引脚的电平极性。
- EPORT 开放漏极使能寄存器（EPODE）控制每个引脚的开放漏极，以单独输出。
- EPORT 数字滤波器控制寄存器（EPFC）可启用滤波器并控制输入脉冲的滤波宽度。

19.4.2.1. 边缘端口中断启用寄存器

偏移地址：0x0000

	7	6	5	4	3	2	1	0
R W	EPIE7	EPIE6	EPIE5	EPIE4	EPIE3	EPIE2	EPIE1	EPIE0
RESET:	0	0	0	0	0	0	0	0

图 19-2：EPORT 端口中断使能寄存器（EPIER）

EPIE[7:0] — 边缘端口中断使能位

读写 EPIE[7:0] 位启用 EPORT 中断请求。如果 EPIER 中的某一位被设置，则当出现以下情况时，EPORT 将生成中断请求：

- EPORT 标志寄存器（EPFR）中的对应位被设置或随后被设置，或者
- 相应引脚电平为低电平，且该引脚配置为电平敏感操作

在 EPIER 中清除一点，就拒绝了来自相应设备的任何中断请求
EPORT 引脚。复位清除了 EPIE[7:0]。

1 = 启用了来自相应 EPORT 引脚的中断请求
0 = 禁用来自相应 EPORT 引脚的中断请求

19.4.2.2. EPORT 数据方向寄存器

偏移地址：0x0001

	7	6	5	4	3	2	1	0
R W	EPDD7	EPDD6	EPDD5	EPDD4	EPDD3	EPDD2	EPDD1	EPDD0
RESET:	0	0	0	0	0	0	0	0

图 19-3：EPORT 数据方向寄存器（EPDDR）

EPDD[7:0] — 边缘端口数据方向位

在 EPDDR 中设置任意位可将对应的引脚配置为输出。清除 EPDDR 中的任意位可将对应的引脚配置为输入。引脚方向与电平/边沿检测配置无关。复位清除了 EPDD[7:0]。

若要将 EPORT 引脚用作外部中断请求源，则 EPDDR 中的对应位必须清除。当 EPDDR 选择输出时，软件可通过编程 EPORT 数据寄存器生成中断请求。

1 = 将 EPORT 引脚配置为输出

0 = 与之对应的 EPORT 引脚被配置为输入

19.4.2.3. EPORT 引脚分配寄存器

偏移地址：0x0002 ~ 0x0003

	15	14	13	12	11	10	9	8
R W	EPPA7		EPPA6		EPPA5		EPPA4	
RESET:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	EPPA3		EPPA2		EPPA1		EPPA0	
RESET:	0	0	0	0	0	0	0	0

图 19-4：EPORT 引脚分配寄存器（EPPAR）

EPPA[7:0] — EPORT 引脚分配选择字段

如图所示，通过读写 EPPAx 字段配置 EPORT 引脚以进行电平检测和上升沿和/或下降沿检测。

配置为电平敏感的引脚会被反相处理，使得外部引脚上的逻辑 0 或逻辑 1 都可视为有效

的中断请求。这类电平敏感型中断输入不会被锁存。为确保电平敏感型中断请求得到确认，中断源必须持续保持信号有效状态，直至软件完成确认操作。选择电平敏感模式时，需通过 INTx 中断使设备退出停止模式。

配置为边沿触发的引脚会被锁存，不需要保持有效状态即可生成中断。配置为边沿检测的引脚将被持续监控，无论其配置为输入还是输出。

表 19-2：EPPAx 现场设置

艾普阿克	引脚配置
00	Pin INTx 水平敏感
01	引脚 INTx 的上升沿被触发
10	引脚 INTx 由下降沿触发
11	将 INTx 设置为同时触发下降沿和上升沿

中断控制器模块可以屏蔽在 EPORT 模块中产生的中断请求。EPPAR 功能独立于所选引脚方向。

重置将清除 EPPAx 字段。

19.4.2.4. EPORT 引脚上拉使能寄存器

偏移地址：0x0004

	7	6	5	4	3	2	1	0
R W	EPPUE7	EPPUE6	EPPUE5	EPPUE4	EPPUE3	EPPUE2	EPPUE1	EPPUE0
RESET:	1	1	1	1	1	1	1	1

图 19-5：EPORT 引脚上拉使能寄存器（EPPUE）

提示：EPPUE 在此 EPORT 模块中无效

EPPUE[7:0] — 边缘端口引脚上拉使能位

设置 EPPUE 中的任意一位，将配置对应的引脚启用上拉功能。清除 EPPUE 中的任意一位，将配置对应的引脚禁用上拉功能。复位操作将设置 EPPUE[7:0]。

1 = 配置为启用上拉的对应 EPORT 引脚

0 = 配置为禁用上拉的对应 EPORT 引脚

19.4.2.5. 边缘端口标志寄存器

偏移地址：0x0005

	7	6	5	4	3	2	1	0
R	EPFR7	EPFR6	EPFR5	EPFR4	EPFR3	EPFR2	EPFR1	EPFR0
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

图 19-6：EPORT 端口标志寄存器（EPFR）

EPF[7:0] — 边缘端口标志位

当 EPORT 引脚配置为边沿触发时，EPFR 中对应的读/写位会指示已检测到选定的边沿。复位操作将清除 EPF[7:0]。

1 = 已检测到 INTx 引脚的选定边。

0 = 未检测到 INTx 引脚的所选边。

当在对应的引脚上检测到选定的边沿时，该寄存器中的位将被设置。该位将保持设置状态，直到通过向其写入 1 来清除。写入 0 不会产生任何影响。如果引脚被配置为电平敏感（EPPARx = 00），则引脚的转换不会影响该寄存器。

19.4.2.6. 边缘端口引脚数据寄存器

偏移地址：0x0006

	7	6	5	4	3	2	1	0
R	EPPD7	EPPD6	EPPD5	EPPD4	EPPD3	EPPD2	EPPD1	EPPD0
W								
RESET:	P	P	P	P	P	P	P	P

= Writes have no effect and the access terminates without a transfer error excepiton.

P = Current pin state

图 19-7：EPORT 端口引脚数据寄存器（EPPDR）

EPPD[7:0] — 边缘端口引脚数据位

只读 EPPDR 反映 EPORT 引脚的当前状态。对 EPPDR 写入没有影响，写入周期正常终止。复位不影响 EPPDR。

19.4.2.7. 边缘端口数据寄存器

偏移地址：0x0007

	7	6	5	4	3	2	1	0
R W	EPD7	EPD6	EPD5	EPD4	EPD3	EPD2	EPD1	EPD0
RESET:	1	1	1	1	1	1	1	1

图 19-8：EPORT 端口数据寄存器（EPDR）

EPD[7:0] — 边缘端口数据位

写入 EPDR 的数据存储在内部寄存器中；如果端口的任何引脚被配置为输出，则存储在该引脚上的位会被驱动到引脚上。读取 EDPR 返回寄存器中存储的数据。复位设置 EPD[7:0]。

19.4.2.8. EPORT 端口位集寄存器

偏移地址：0x0008

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	EPBSR7	EPBSR6	EPBSR5	EPBSR4	EPBSR3	EPBSR2	EPBSR1	EPBSR0
RESET:	0	0	0	0	0	0	0	0

图 19-9：EPORT 端口位组寄存器（EPBSR）

EPBSR[7:0] — EPORT 端口位组寄存器

1 = 将设置 EPDR 的对应位；

0 = EPDR 的对应位不会受到影响；

19.4.2.9. EPORT 数字滤波器控制寄存器

偏移地址：0x0009

	7	6	5	4	3	2	1	0
R W	Filter_Width[6:0]							Filter_En
RESET:	0	0	0	0	0	0	0	0

图 19-10：EPORT 数字滤波器控制寄存器（EPFC）

Filter_En — EPORT 数字滤波器启用位

1 = EPORT 数字滤波器启用；
0 = EPORT 数字滤波器禁用；

Filter[j0:0] — EPORT 数字滤波器脉冲宽度选择位

Filter Width[6:0] 确定输入脉冲的宽度。如果输入脉冲宽度小于（Filter Width[6:0]+2），则将对脉冲进行滤波。

19.4.2.10. EPORT Open Drain 启用寄存器

偏移地址：0x000A

	7	6	5	4	3	2	1	0
R W	EPODE7	EPODE6	EPODE5	EPODE4	EPODE3	EPODE2	EPODE1	EPODE0
RESET:	0	0	0	0	0	0	0	0

图 19-11：EPORT Open Drain 启用寄存器（EPODE）

提示：EPODE 在此设备中无效

EPODE[7:0] — 边缘端口 Open Drain(开漏极)启用位

如果将 EPORT 配置为输出，则设置 EPODE 中的任意一位可将对应引脚配置为开漏输出。清除 EPODE 中的任意一位可将对应引脚配置为 CMOS 输出。复位操作会清除 EPODE[7:0]。

1 = 配置为打开 Open Drain 输出的对应 EPORT 引脚

0 = 配置为 CMOS 输出的对应 EPORT 引脚

19.4.2.11. EPORT 电平极性寄存器

偏移地址：0x000B

	7	6	5	4	3	2	1	0
R W	EPLPR7	EPLPR6	EPLPR5	EPLPR4	EPLPR3	EPLPR2	EPLPR1	EPLPR0
RESET:	0	0	0	0	0	0	0	0

图 19-12：EPORT 电平极性寄存器（EPLPR）

EPLPR[7:0] — 边缘端口电平极性位

如果将 EPORT 配置为电平敏感，则设置 EPLPR 中的任何位可将对应引脚配置为高电平敏感。清除 EPLPR 中的任何位可将对应引脚配置为低电平敏感。复位操作会清除 EPLPR[7:0]。

1 = 将 EPORT 引脚配置为高电平敏感

0 = 与之对应的 EPORT 引脚被配置为低电平敏感

19.4.2.12. EPORT 端口位清除寄存器

偏移地址：0x000F

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	EPBCR7	EPBCR6	EPBCR5	EPBCR4	EPBCR3	EPBCR2	EPBCR1	EPBCR0
RESET:	0	0	0	0	0	0	0	0

图 19-13：EPORT 端口位清除寄存器（EPBCR）

EPBCR[7:0] — EPORT 端口位清除寄存器

1 = EPDR 的对应位将为 0；

0 = EPDR 的对应位不会受到影响；

20. CANBus 控制器（CANBC)

20.1.介绍

CAN 总线模块是根据 CAN 2.0B 协议规范实现 CAN 协议的通信控制器。图 20-1 展示了该模块的通用方块图，其中包含两个嵌入式存储器：一个用于存储信息缓冲区（MB），另一个用于存储接收个体掩码寄存器。该模块最多可支持 64 个信息缓冲区。各子模块的具体功能将在后续章节中详细说明。

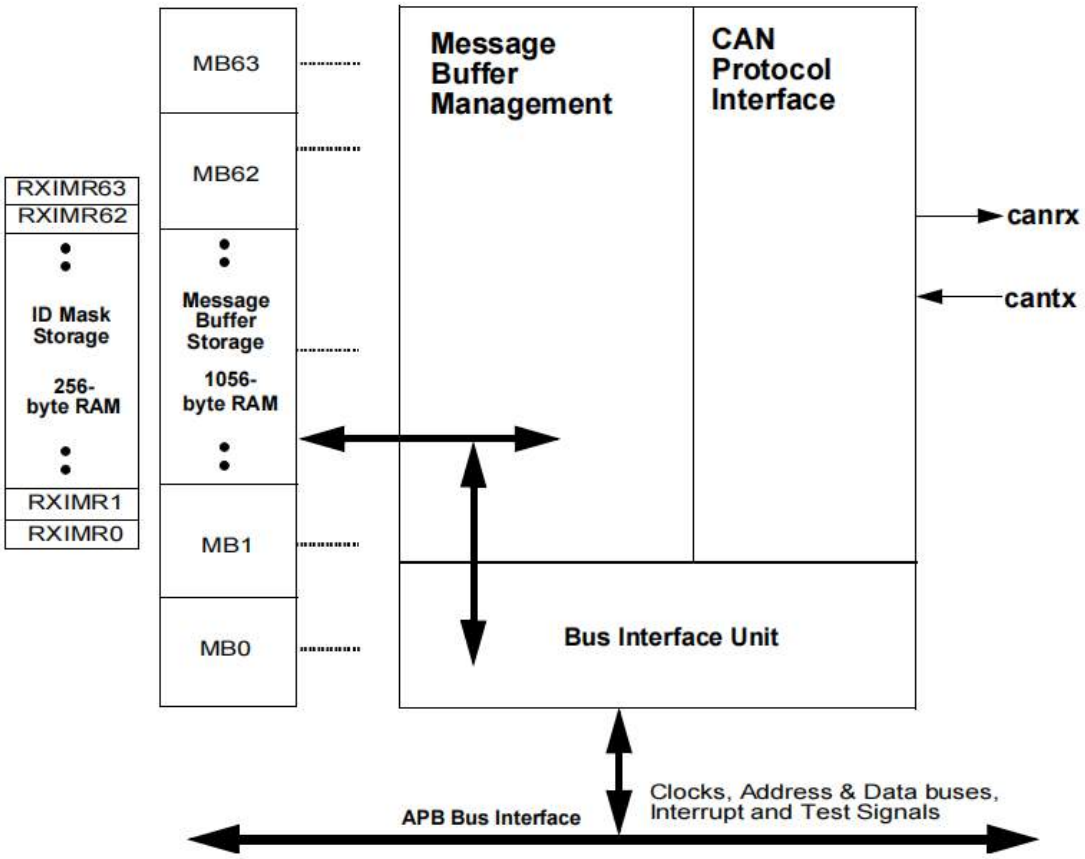


图 20-1：CAN 总线方块图

20.1.1. 概述

CAN 协议的设计初衷主要是作为车载串行数据总线使用，但其适用范围并不局限于此。该协议特别针对实时处理、车辆电磁干扰环境下的稳定运行、成本效益以及所需带宽等特定需求而设计。CAN 总线模块完整实现了 CAN 协议规范 2.0 B 版本，能够同时支持标准信息帧和扩展信息帧两种格式。

支持 16 个信息缓冲区。信息缓冲区存储在专用于 CANBus 模块的嵌入式 SRAM 中。

CAN 协议接口（CPI）子模块负责管理 CAN 总线上的串行通信，通过请求 SRAM 访问来接收和发送信息帧，同时验证接收到的信息并执行错误处理。信息缓冲管理（MBM）子模块负责信息缓冲区的选中操作，处理仲裁机制与 ID 匹配算法。总线接口单元（BIU）子模块控制内部接口总线的访问权限，实现与 CPU 及其他模块的连接。时钟总线、地址总线、数据总线、中断输出及测试信号均通过总线接口单元进行访问。

20.1.2. CAN 总线模块功能

CAN 总线模块具有以下显著特点：

- 全面实施 CAN 协议规范，版本 2.0B
  - 标准数据和远程帧
  - 扩展数据和远程帧
  - 0-8 字节数据长度
  - 可编程比特率最高可达 1 Mbit/s
  - 内容相关寻址
- 16 个信息缓冲区，数据长度为 0 到 8 字节
- 每个 MB 可配置为 Rx 或 Tx，均支持标准和扩展信息
- 每个信息缓冲区中的单个 Rx 掩码寄存器
- 包括用于 MB 存储的 288 字节(16 MB）SRAM
- 包括用于单个 Rx 掩码寄存器的 64 字节(16 MB）SRAM
- 全功能 Rx FIFO，具有存储 6 帧的容量和内部指针处理
- 强大的 Rx FIFO ID 过滤，能够将输入的 ID 与 8 个扩展、16 个标准或 32 个部分（8 位）ID 进行匹配，并具有单独掩码功能
- 可编程的 CAN 协议接口时钟源，可以是总线时钟或晶体振荡器
- 未使用的 MB 和 Rx 隔离寄存器空间可用作通用 SRAM 空间
- 只能收听模式功能
- 支持自检操作的可编程回环模式
- 可编程传输优先级方案：最低 ID、最低缓冲区号或最高优先级
- 基于 16 位自由运行计时器的时间戳
- 通过特定信息同步的全球网络时间
- 可屏蔽中断
- 与传输介质无关（假设为外部收发器）

- 由于采用高优先级信息的仲裁方案，延迟时间较短
- 低功耗模式
- 发送信息缓冲区的硬件取消

20.1.3. 操作模式

CAN 总线模块有四种工作模式：正常模式（用户和监控器）、冻结模式、只听模式和环回模式，以及低功耗模式：禁用模式。

- 正常模式（用户或主管）

  在正常模式下，模块运行，接收和/或发送信息帧，正常处理错误，并且启用所有 CAN 协议功能。用户和监督模式的区别在于对某些受限控制寄存器的访问。

- 冷冻模式

  当 MCR 中的 FRZ 位被置为有效时，该功能即被启用。启用后，当 MCR 中的 HALT 位被设置或 MCU 层级请求进入调试模式时，系统将进入冻结模式。在此模式下，不会进行任何帧的发送或接收操作，并且会丢失与 CAN 总线的同步性。更多详细信息请参阅 20.4.10.1 的《冻结模式》文档。

- 仅听模式

  当控制寄存器中的 LOM 位被激活时，模块将进入该工作模式。在此模式下，传输功能被禁用，所有错误计数器均被冻结，模块将以 CAN 错误被动模式运行。此时仅能接收其他 CAN 站点已确认的信息。若 CAN 总线检测到未被确认的信息，系统会标记 BIT0 错误（但不改变 REC 状态），如同正在尝试确认该信息一般。

- 循环模式

  当控制寄存器中的 LPB 位被激活时，模块即进入该工作模式。在此模式下，can 总线会执行内部环回操作，可用于自检功能。发射端的比特流输出会被内部反馈至接收端输入端，此时接收端的 can 输入引脚将被忽略，发送端的 can 输出则进入隐性状态（逻辑‘1’）。此时 can 总线仍保持正常传输模式，将自身发送的信息视为来自远程节点的数据。为确保正确接收自身信息，系统会忽略 can 帧确认字段中 ACK 时隙发送的比特。该模式下同时触发发送和接收中断。

- 模块禁用模式

  当 MCR 中的 MDIS 位被置为有效时，系统将进入低功耗模式。禁用时，模块将关闭 CAN 协议接口和信息缓冲管理子模块的时钟。要退出此模式，只需将 MCR 中的 MDIS 位置为无效即可。有关详细信息，请参阅 20.4.10.2，“模块禁用模式”。

20.2. 外部信号说明

20.2.1. 概述

CAN 总线模块有两个 I/O 信号连接到外部 MCU 引脚，这些信号汇总在表 20-1 中，并在下一小节中详细描述。

表 20-1：CAN 总线信号

信号名称	方向	描述
CANRX	Input	CAN 接收引脚
CANTX	Output	CAN 传输引脚

20.2.2. 信号描述

20.2.2.1. CANRX

此引脚是来自 CAN 总线收发器的接收引脚。主导状态用逻辑电平‘0’表示，从属状态用逻辑电平‘1’表示。

20.2.2.2. CANTX

此引脚是 CAN 总线收发器的发送引脚。逻辑电平“0”表示为主状态，逻辑电平“1”表示为从状态。

21. 串行通信接口模块（SCI）

21.1. 介绍

串行通信接口模块（SCI）支持基本的 UART，允许与外围设备和其他微控制器单元（MCU）进行异步串行通信。该模块还支持 LIN 从机操作。

21.2. 特性

每个 SCI 模块的特性包括：

- 全双工标准非归零（NRZ）格式
- 可编程波特率（13 位模数分频器），支持从 4 倍到 256 倍的可配置过采样率
- 中断，轮询操作：
  - 传输数据寄存器为空且传输完成
  - 接收数据寄存器满
  - 接收溢出、奇偶校验错误、帧错误和噪声错误
  - 闲置转速检测器
  - 接收引脚上的有效边沿
  - 检测支持 LIN 的探测器
  - 收到数据匹配
- 硬件奇偶校验生成和检查
- 可编程的 8 位、9 位或 10 位字符长度
- 可编程的 1 位或 2 位停止位
- 三种接收器唤醒方式：
  - 怠速线唤醒
  - 禁止标记唤醒
  - 收到数据匹配
- 自动地址匹配以减少 ISR 开销：
  - 地址标记匹配
  - 空闲线路地址匹配
  - 地址匹配开始，地址匹配结束
- 可选的 13 位中断字符生成/11 位中断字符检测
- 可配置的空闲长度检测，支持 1、2、4、8、16、32、64 或 128 个空闲字符
- 可选择发射机输出和接收机输入极性
- 可选择的 IrDA 1.4 返回到零反转（RZI）格式，具有可编程脉冲宽度

- 用于发送和接收的独立 FIFO 结构
  - 支持为接收和发送请求配置可分离的可定义水印
  - 如果接收 FIFO 为空，则接收器可选择在配置的空闲字符数后断言请求

21.3. 操作模式

- 停止模式
- 等待模式

21.4. 方块图

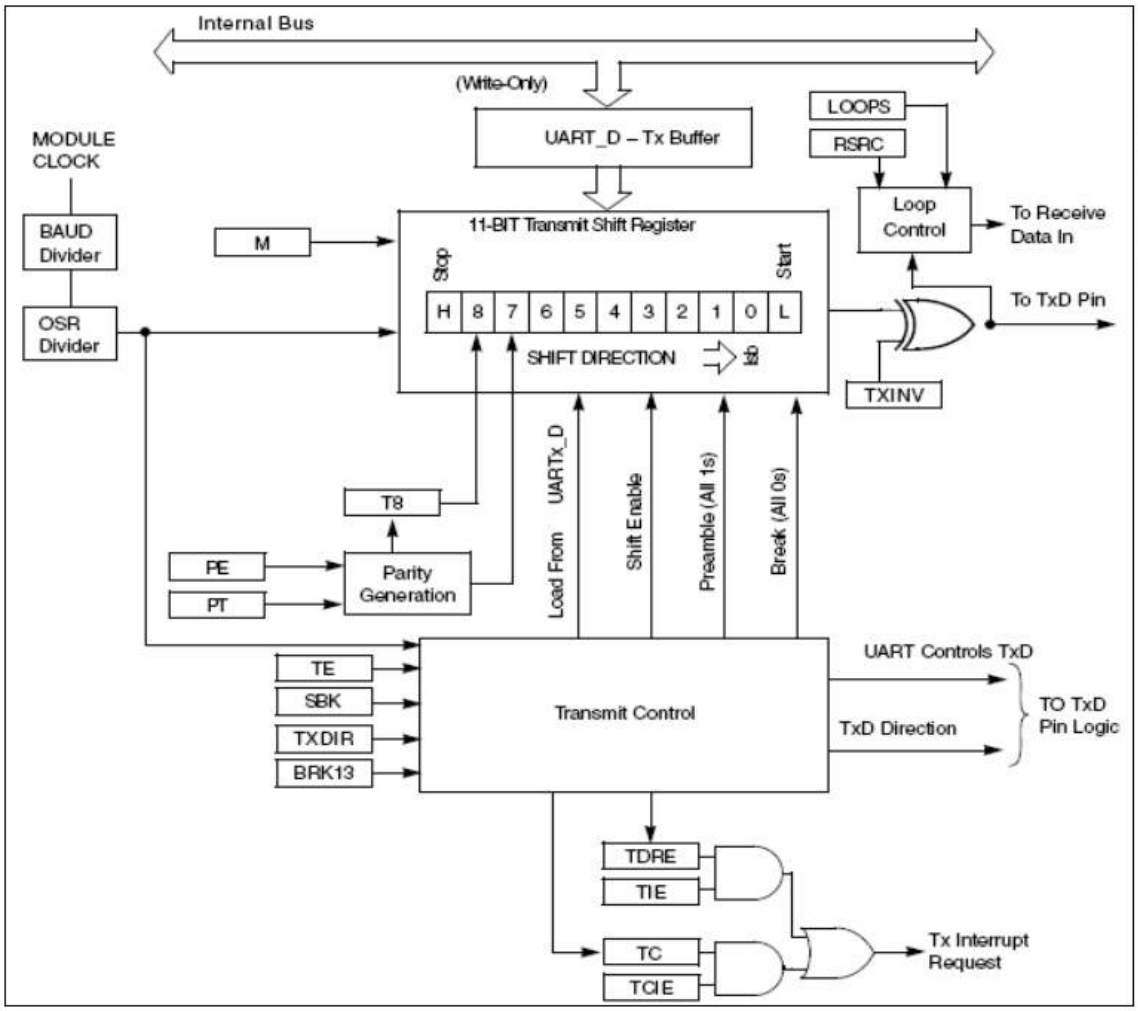


图 21-1：SCI 发送器方块图

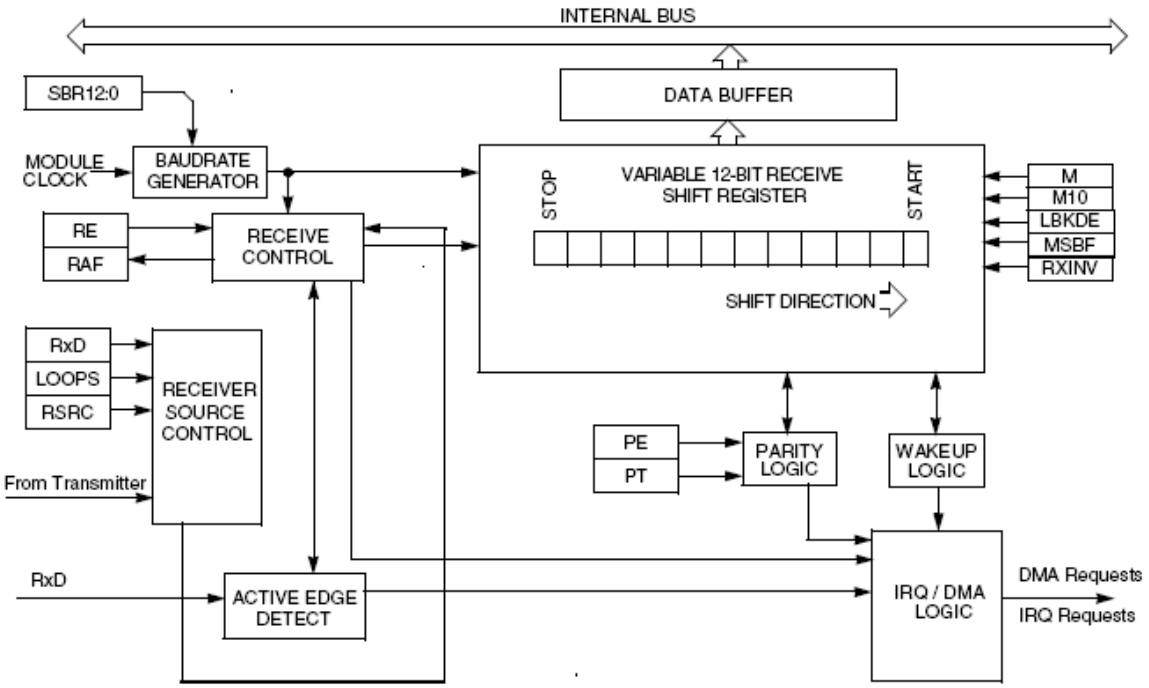


图 21-2：SCI 接收器方块图

21.5. 操作模式

21.5.1. 停止模式

SCI 在停止模式下将无法正常工作。

21.5.2. 等待模式

当 DOZEEN 位被设置时，SCI 可配置为“停止等待”模式。此时，发送器和接收器将完成当前字的发送/接收。

21.6. 信号描述

表 21-1 给出了此处描述的信号概述。

表 21-1：信号特性

信号	描述	I/O
TXD	传输数据。此引脚通常为输出，但在发射器被禁用或发射方向配置为接收数据时，该引脚在单线模式下为输入（三态）。	I/O
RXD	接收数据	I
SCI_DE	RS-485 收发器的控制信号	O

21.7. 内存映射和寄存器

表 21-2 显示了 SCI 内存映射。

表 21-2：SCI 模块内存映射 [1]

地址偏移量	寄存器
0x0000	SCI 版本 ID（SCIВерсий）
0x0004	SCI 参数（SCI_PARAM）
0x0008	SCI 重置（SCI_RESET）
0x000C	SCI 引脚（SCI_PIN）
0x0010	SCI 波特率寄存器（SCI_BAUD）
0x0014	SCI 状态登记（SCIStat）
0x0018	SCI 控制寄存器（SCI_CTRL）
0x001C	SCI 数据登记（SCI_DATA）
0x0020	SCI Match 地址寄存器（SCI_MATCH）
0x0024	SCI 调制解调器 IrDA 寄存器（SCI_MODIR）
0x0028	SCI FIFO 寄存器（SCI_FIFO）
0x002C	SCI 水印寄存器（SCI_WATER）
0x0030	SCI 过采样比率寄存器（SCI_OSR）

提示：每个模块分配了 16KB 的地址空间，所有这些地址空间可能无法被解码。访问指定模块内存映射之外的地址将产生总线错误异常。

21.7.1. SCI 版本 ID 寄存器

偏移地址：0x0000

	31	30	29	28	27	26	25	24
R	VERID							
W								
RESET	0	0	0	0	0	1	0	0
	23	22	21	20	19	18	17	16
R	VERID							
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	VERID							
W								
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	VERID							
W								
RESET:	0	0	0	0	0	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

图 21-3：SCI 版本 ID 寄存器（SCIВерсийID）

VERSION_ID[31:0] — SCI 版本 ID

21.7.2. SCI 参数寄存器

偏移地址：0x0004

	31	30	29	28	27	26	25	24
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	Reserved				RXFIFO_SZ			
W								
RESET	0	0	0	0	0	0	1	1
	7	6	5	4	3	2	1	0
R	Reserved				TXFIFO_SZ			
W								
RESET:	0	0	0	0	0	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

图 21-4：SCI 参数寄存器（SCI_param）

RXFIFO_SZ — 接收缓冲区/FIFO 大小

此只读控制位指示接收缓冲器/FIFO 大小。

TXFIFO_SZ — 发送缓冲区/FIFO 大小

此只读控制位指示发送缓冲器/FIFO 大小。

21.7.3. SCI 复位寄存器

偏移地址：0x0008

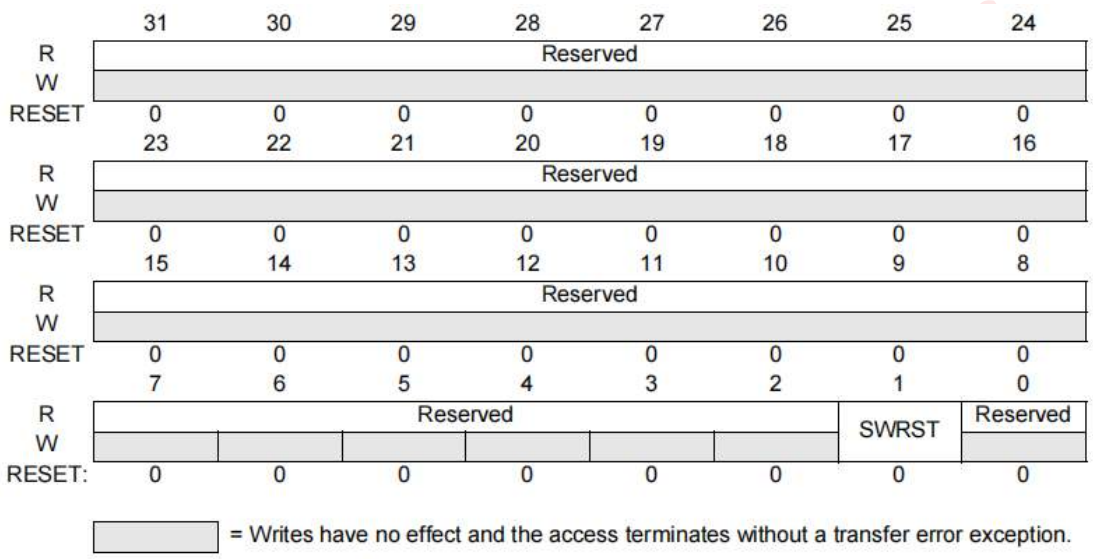


图 21-5：SCI 复位寄存器（SCI_RESET）

SWRST — 软件复位

该读/写位允许软件重置 SCI IP。

1 = 软件复位被断言

0 = 未执行软件复位

21.7.4. SCI 引脚寄存器

偏移地址：0x000C

	31	30	29	28	27	26	25	24
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	Reserved						PINCFG	
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 21-6：SCI 引脚寄存器（SCI_PIN）

PINCFG-本模块中无用

21.7.5. SCI Baud 率寄存器

偏移地址：0x0010

	31	30	29	28	27	26	25	24
R	MAEN1	MAEN2	M10	Reserved				
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	TDMAE	Reserved	RDMAE	Reserved	MATCFG		BOTHEDGE	RESYNC-DIS
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	LBKDIE	RXEDGIE	SBNS	SBR				
W								
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	SBR							
W								
RESET:	0	0	0	0	0	1	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 21-7：SCI 波特率寄存器（SCI_BAUD）

MAEN1 — 匹配地址模式启用 1

1 = 为 MATCH[MA1] 启用自动地址匹配或数据匹配模式

0 = 正常运行

MAEN2 — 匹配地址模式启用 1

1 = 为 MATCH[MA2] 启用自动地址匹配或数据匹配模式

0 = 正常运行

M10 — 10 位模式选择

M10 位使第 10 位成为串行传输的一部分。只有在发送器和接收器都处于禁用状态时，才应更改此位。

1 = 接收器和发射器使用 10 位数据字符

0 = 接收器和发送器使用 8 位或 9 位数据字符

TDMAE — 发射器 DMA 使能

TDMAE 将发送数据寄存器空置标志 LPUART_STAT[TDRE] 配置为生成 DMA 请求。

1 = DMA 请求已启用

0 = 禁用 DMA 请求

RDMAE — 接收器全 DMA 使能

RDMAE 将接收器数据寄存器满标志 LPUART_STAT[RDRF] 配置为生成 DMA 请求。

1 = DMA 请求已启用

0 = 禁用 DMA 请求

MATCFG — 匹配配置

配置所使用的匹配寻址模式：

00--地址匹配唤醒；

01--空闲匹配唤醒；

10-比赛开始和比赛结束；

11--启用 RWU 对发射器 CTS 输入的数据匹配和匹配开/关功能

BOTHEDGE — 双边缘采样

允许在波特率时钟的两个边沿对接收数据进行采样，从而在给定过采样比率下使接收器对输入数据的采样次数有效翻倍。对于 x4 到 x7 之间的过采样比率，必须设置此位；对于更高比率的过采样，则为可选配置。该位仅应在接收器处于禁用状态时进行修改。

1 = 接收器使用波特率时钟的上升沿和下降沿来采样输入数据。

0 = 从设备使用波特率时钟的上升沿来接收输入数据。

RESYNCDIS — 重同步禁用

当设置时，禁用检测到数据 1 后跟数据 0 的转换时接收的数据字的重新同步。此位仅应在禁用接收器时更改。

1 = 接收数据字时的重新同步被禁用

0 = 支持在接收数据字期间进行重新同步

LBKDIE-LIN 中断检测和中断使能

LBKDIE 使 LIN 坏道检测标志 LBKDIF 能够生成中断请求。

1 = 当 SCI_STAT[LBKDIF] 标志为 1 时，请求硬件中断

0 = SCI_stat[LBKDIF] 的硬件中断已禁用（使用轮询）

RXEDGIE — RX 输入主动边缘中断使能

启用接收输入有效边沿 RXEDGIF 以生成中断请求。当 RXEDGIE 被设置时，更改 CTRL[LOOP] 或 CTRL[RSRC] 可导致 RXEDGIF 被设置。

1 = 当 SCI_STAT[RXEDGIF] 标志为 1 时，请求硬件中断

0 = SCI_stat[RXEDGIF] 的硬件中断已禁用（使用轮询）

SBNS — 停止位号选择

SBNS 确定数据字符是一个还是两个停止位。只有在发送器和接收器都处于禁用状态时，才应更改此位。

1 = 两个停止位

0 = 单站位

SBR — 码率模数分频器

SBR[12:0] 中的 13 位用于设置波特率发生器的模数分频比。当 SBR 为 1 至 8191 时，波特率等于“波特时钟/（（OSR+1）×SBR）”。必须在发送器和接收器均处于禁用状态(SCI_CTRL[RE] 和 SCI_CTRL[TE] 均为 0）时，才能更新 13 位波特率设置[SBR12：SBR0]。

提示：在 LT165A 可工作温度范围内，使用外部晶振（XTAL, EXTAL 引脚接 12Mhz 晶振）则波特率实际误差在 1%内。

21.7.6. SCI 状态登记

偏移地址：0x0014

	31	30	29	28	27	26	25	24
R	LBKDIF	RXEDGIF	MSBF	RXINV	RWUID	BRK13	LBKDE	RAF
W	w1c	w1c						
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W				w1c	w1c	w1c	w1c	w1c
RESET	1	1	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	MA1F	MA2F	Reserved					
W	w1c	w1c						
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	Reserved							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 21-8：SCI 状态寄存器（SCI_STAT）

LBKDIF — LIN 中断检测中断标志

当 LIN 断路检测电路被启用并且检测到 LIN 断路字符时，将设置 LBKDIF。通过向其写入 1 可清除 LBKDIF。

1 = 检测到 LIN 断行符

0 = 未检测到 LIN 断行符

RXEDGIF — RXD 引脚有效边中断标志

当 RXD 引脚上出现有效边沿时，RXEDGIF 会被设置。如果 RXINV = 0，则为下降沿；如果 RXINV = 1，则为上升沿。通过向 RXEDGIF 写入 1 即可清除该信号。

1 = 接收引脚上出现有效边沿

0 = 接收引脚上没有出现有效边沿

MSBF — MSB First

设置此位可反转在导线上发送和接收的位的顺序。此位不影响位的极性、奇偶校验位的位置或起始或停止位的位置。只有在发送器和接收器都处于禁用状态时，才能更改此位。

1 表示 MSB（第 9、8、7 或 6 位），根据 CTRL[M]、CTRL[PE] 和 BAUD[M10] 的设置，它是起始位之后传输的第一个比特。此外，根据 CTRL[M] 和 CTRL[PE] 的设置，起始位之后接收到的第一个比特被识别为第 9、8、7 或 6 位。

0 = LSB（bit0）是起始位之后传输的第一个比特。此外，起始位之后接收到的第一

个比特被标识为 bit0。

RXINV-接收数据反转

设置此位可反转所接收数据输入的极性。设置 RXINV 可在所有情况下反转 RXD 输入：数据位、起始和停止位、中断和空闲。

1 = 接收数据反转

0 = 接收未反转的数据

RWUID — 接收唤醒空闲检测

对于 RWU 空闲字符，RWUID 控制唤醒接收器的空闲字符是否设置 idle 位。对于地址匹配唤醒，RWUID 控制地址不匹配时是否设置 idle 位。此位仅应在接收器被禁用时更改。

1 = 在接收待机状态期间（RWU = 1），检测到空闲字符时，将设置 IDLE 位。在地址匹配唤醒期间，如果地址不匹配，则会设置 IDLE 位。

0 = 在接收待机状态期间（RWU = 1），检测到空闲字符时，IDLE 位不会被设置。在地址匹配唤醒期间，如果地址不匹配，则 IDLE 位也不会被设置。

BRK13 — 断字符生成长度

BRK13 选择较长的传输中断字符长度。检测到的帧错误不受该位的状态影响。只有在禁用发送器时才应更改该位。

1 = 剪切字符的传输长度为 13 个比特时间（如果 M = 0，SBNS = 0)或 14 个比特时间（如果 M = 1，SBNS = 0 或 M = 0，SBNS = 1)或 15 个比特时间（如果 M = 1，SBNS = 1 或 M = 0，SNBS = 0)或 16 个比特时间（如果 M = 1，SNBS = 1)。

0 = 分割符的传输长度为 10 比特时间（如果 M = 0，SBNS = 0)或 11（如果 M = 1，SBNS = 0 或 M = 0，SBNS = 1)或 12（如果 M = 1，SBNS = 1 或 M = 1，SNBS = 0)或 13（如果 M = 1，SNBS = 1)

LBKDE — LIN 中断检测启用

LBKDE 选择较长的中断字符检测长度。在设置 LBKDE 时，接收数据不会存储在接收数据缓冲区中。

1 = 在长度为 11 个比特时间（如果 M = 0，SBNS = 0)或 12 个比特时间（如果 M = 1，SBNS = 0 或 M = 0，SBNS = 1)或 14 个比特时间（如果 M = 1，SBNS = 1 或 M = 10，SNBS = 0)或 15 个比特时间（如果 M = 10，SNBS = 1)处检测到分隔字符

0 = 在长度为 10 比特时间（如果 M = 0，SBNS = 0)或 11（如果 M = 1，SBNS = 0 或 M = 0，SBNS = 1)或 12（如果 M = 1，SBNS = 1 或 M = 1，SNBS = 0)或 13（如果 M = 1，SNBS = 1)处检测到中断字符

RAF — 接收器活动标志

当接收器检测到有效起始位的开始时，将设置 RAF，当接收器检测到空闲线路时，RAF 将自动清除。

1 = SCI 接收器处于活动状态(RXD 输入未处于空闲状态)

0 = SCI 接收器空闲，等待起始位

TDRE — 发送数据寄存器空标志

当发送 FIFO 处于启用状态时，TDRE 会在发送 FIFO 中的数据字数（SCI_DATA）等于或小于 SCI_WATER[TXWATER] 指示的数值时置位。要清除 TDRE，需持续向 SCI 数据寄存器（SCI_DATA）写入数据，直至发送 FIFO 中的字数超过 SCI_WATER[TXWATER] 指示的数值。当发送 FIFO 处于禁用状态时，TDRE 会在发送数据寄存器（SCI_DATA）为空时置位。要清除 TDRE，只需向 SCI 数据寄存器（SCI_DATA）写入数据即可。

TDRE 不受正在传输的字符的影响，它在每个传输的字符开始时更新。

1 = 传输数据缓冲区为空

0 = 传输数据缓冲区已满

TC — 传输完成标志

当传输正在进行或前导码/中断字符被加载时，传输控制（TC）会被清除。当发送缓冲区为空且没有数据、前导码或中断字符正在传输时，TC 会被置位。TC 置位后，发送数据输出信号将变为空闲状态（逻辑 1）。要清除 TC，需执行以下操作：向 SCI_DATA 写入数据以传输新数据，若需发送前导码则先清空再置位 SCI_CTRL[TE]，若需发送中断字符则向 SCI_CTRL[SBK] 写入 1。

1 = 发射机处于空闲状态（发射活动已完成）

0 = 发射机处于活动状态（发送数据、前导码或中断）

RDRF — 接收数据寄存器满标志

当接收 FIFO 处于启用状态时，若接收缓冲区中的数据字数超过 SCI_WATER[RXWATER] 指示的数值，则会触发 RDRF 寄存器的置位。要清除 RDRF，需持续读取 SCI_DATA 寄存器，直至接收数据缓冲区中的数据字数等于或小于 SCI_WATER[RXWATER] 指示的数值。当接收 FIFO 处于禁用状态时，若接收缓冲区（SCI_DATA）已满，则会触发 RDRF 寄存器的置位。要清除 RDRF，只需读取 SCI_DATA 寄存器即可。

正在接收的字符不会导致 RDRF 发生改变，直到整个字符被接收为止。即使设置了 RDRF，也应继续接收该字符，直到整个字符被接收后发生溢出条件为止。

1 = 接收数据缓冲区已满

0 = 接收到空的数据缓冲区

IDLE — 空闲线路标志

当 SCI 接收线路在活动期结束后进入空闲状态满字符周期时，IDLE 信号即被置位。清除 ILT 信号后，接收器开始计算起始比特之后的空闲比特时长。若接收字符全为 1，则这些比特时长与停止位时长将计入逻辑高电平所需的满字符周期（10 至 13 个比特时长），这是接收器检测空闲线路所需的时间。当 ILT 信号被置位时，接收器不会在停止位之后开始计算空闲比特时长。停止位及前一个字符末尾的任何逻辑高电平比特时长，均不计入接收器检测空闲线路所需的逻辑高电平满字符周期。

若要清除 IDLE，请将逻辑 1 写入 IDLE 标志。清除 IDLE 后，在接收缓冲区中存储新字符或 LIN 中断字符设置 LBKDIF 标志之前，该标志不能再次被设置。即使接收线路长时间处于空闲状态，IDLE 也只被设置一次。

1 = 检测到空闲线路

0 = 未检测到空闲线路

OR — 接收器溢出标志

当软件未能阻止接收数据寄存器溢出时，系统会触发或门（OR）被置位。在缓冲区溢出的数据字完全接收停止位后，或门立即被激活，同时所有其他错误标志（FE、NF 和 PF）均被禁止触发。此时移位寄存器中的数据会丢失，但 SCI 数据寄存器中已存储的数据不受影响。若启用了低电平中断功能（LBKDE），且检测到线路中断信号时，若在接收下一个数据字符前未清除 LBKDIF 标志，则或门字段将被激活。

在 OR 标志设置时，即使有足够的空间，也不会在数据缓冲区中存储其他数据。若要清除 OR，请将逻辑 1 写入 OR 标志。

1 = 接收超限（丢失新的 SCI 数据）

0 = 无超限

NF — 噪声标志

接收器采用的高级采样技术会在每个接收到的比特中进行三次采样。若在帧内的任意比特时间内，任一采样值与其余采样值存在偏差，则会检测到该字符存在噪声干扰。当从 SCI_DATA 接口读取的下一个字符被检测到含噪时，系统将自动触发噪声清除（NF）机制。要清除 NF 状态，需向 NF 寄存器写入逻辑 1。

1 = 在 SCI_DATA 中接收到的字符内检测到噪声

0 = 未检测到噪声

FE — 框架错误标志

当从 SCI_DATA 接收的下一个字符在预期停止位处检测到逻辑 0 时，FE 被设置。要清除 FE，请向 FE 写入逻辑 1。

1 = 框架错误

0 = 没有检测到框架错误。这并不能保证框架是正确的

PF — 均等性错误标志

当启用奇偶校验（PE = 1）时，如果从 SCI_DATA 接收的下一个字符的奇偶校验位与预期值不一致，则设置 PF。要清除 PF，请向 PF 写入逻辑 1。

1 = 奇偶校验错误

0 = 无奇偶校验错误

MA1F — 第 1 轮比赛旗

当从 SCI_DATA 中读取的下一个字符与 MA1 匹配时，将设置 MA1F。要清除 MA1F，请向 MA1F 写入逻辑 1。

1 = 接收数据等于 MA1

0 = 接收到的数据不等于 MA1

MA2F — 双旗对抗赛

当从 SCI_DATA 中读取的下一个字符与 MA2 匹配时，将设置 MA2F。要清除 MA2F，请向 MA2F 写入逻辑 1。

1 = 接收的数据等于 MA2

0 = 接收到的数据不等于 MA2

21.7.7. SCI 控制寄存器

偏移地址：0x0018

	31	30	29	28	27	26	25	24
R W	R8T9	R9T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R W	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R W	MA1IE	MA2IE	Reserved			IDLECFG		
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	LOOPS	DOZEEN	RSRC	M	WAKE	ILT	PE	PT
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 21-9：SCI 控制寄存器（SCI_CTRL）

R8T9 — 接收第 8 位/发送第 9 位

当 SCI 被配置为 9 位或 10 位数据格式时，R8 是接收的第九个数据位。在读取 9 位或 10 位数据时，请先读取 R8，然后再读取 SCI_DATA。

当 SCI 被配置为 10 位数据格式时，T9 是接收的第十个数据位。在写入 10 位数据时，在写入 SCI_DATA 之前写入 T9。如果 T9 不需要改变其先前值，例如当其用于生成地址标记或奇偶校验时，则不需要每次写入 SCI_DATA 时都写入 T9。

R9T8 — 接收第 9 位/发送第 8 位

当 SCI 配置为 10 位数据格式时，R9 是接收的第 10 个数据位。在读取 10 位数据时，请先读取 R9，然后再读取 SCI_DATA

当 SCI 被配置为 9 位或 10 位数据格式时，T8 是接收的第九个数据位。在写入 9 位或 10 位数据时，应在写入 SCI_DATA 之前写入 T8。如果 T8 不需要改变其与前一次不同的值，例如当其被用来生成地址标记或奇偶校验时，则不必每次写入 SCI_DATA 时都写入 T8。

TXDIR — 单线模式下的 TXD 引脚方向

当 SCI 配置为单线半双工操作（LOOPS = RSRC = 1）时，此位确定 TXD 引脚上的数据方向。清除 TXDIR 时，发送器将在接收器开始从 TXD 引脚接收数据之前完成对当前字符（如果有）的接收。

1 = TXD 引脚是单线模式下的输出

0 = TXD 引脚是单线模式下的输入

TXINV — 传输数据反转

设置此位可反转传输数据输出的极性。设置 TXINV 可反转所有情况下的 TXD 输出：数据位、起始和停止位、中断和空闲。

1 = 传输数据反转

0 = 传输未反转的数据

ORIE — 中断溢出使能

此位使溢出标志（OR）能够生成硬件中断请求。

1 = 当 OR 被设置时，请求硬件中断

0 = 禁用中断，使用轮询

NEIE — 噪声错误中断启用

此位使噪声标志（NF）能够生成硬件中断请求。

1 = 设置 NF 时请求硬件中断

0 = 禁用 NF 中断；使用轮询

FEIE — 帧错误中断使能

此位使帧错误标志（FE）能够生成硬件中断请求。

1 = 当 FE 被设置时，请求硬件中断

0 = FE 中断被禁用；使用轮询

PEIE — Parity 错误中断启用

此位使奇偶校验错误标志（PF）能够生成硬件中断请求。

1 = 当 PF 被设置时，请求硬件中断

0 = 禁用 PF 中断；使用轮询）

TIE — 发送中断使能

使 STAT[TDRE] 能够生成中断请求。

1 = 当 TDRE 标志为 1 时请求硬件中断

0 = 禁用来自 TDRE 的硬件中断；使用轮询

TCIE — 传输完成中断使能，用于

TCIE 使传输完成标志 TC 能够生成中断请求。

1 = 当 TC 标志为 1 时，请求硬件中断

0 = TC 的硬件中断已禁用；使用轮询

RIE — 接收器中断使能

使 STAT[RDRF] 能够生成中断请求。

1 = 当 RDRF 标志为 1 时，请求硬件中断

0 = 禁用来自 RDRF 的硬件中断；使用轮询

ILIE — 空闲线路中断使能

ILIE 使空闲线路标志 STAT[IDLE] 能够生成中断请求。

1 = 当 IDLE 标志为 1 时请求硬件中断

0 = 禁用来自 IDLE 的硬件中断；使用轮询

TE — 发射机启用

启用 SCI 发送器。TE 也可用于通过清除并设置 TE 来排队空闲前导码。当 TE 被清除时，此寄存器位将读取为 1，直到发送器完成当前字符且 TXD 引脚处于三态。

1 = 发射器启用

0 = 发射器禁用

RE — 接收器启用

启用 SCI 接收器。当将 RE 写入 0 时，此寄存器位将读为 1，直到接收器完成对当前字符（如果有）的接收为止。

1 = 接收器启用

0 = 适配器被禁用

RWU — 接收器唤醒控制

该字段可设置为将 SCI 接收器置于待机状态。当 RWU 事件发生时，RWU 自动清除，即当 CTRL[WAKE] 清除时的空闲事件或当 CTRL[WAKE] 设置且 STAT[RWUID] 清除时的地址匹配。

若信道当前处于非空闲状态，则必须将 CTRL[WAKE] 设置为 0（空闲唤醒）才能触发 RWU。该状态可通过 STAT[RAF] 寄存器确定。若将标志位设置为唤醒空闲事件而信道已处于空闲状态，SCI 可能会丢弃数据。这是因为系统需要先检测到数据接收或 LIN 中断，才能重新确认空闲状态。

1 = SCI 接收器处于待机状态，等待唤醒

0 = 正常接收器操作

SBK — 发送中断

当 SBK 位被写入 1 后转为 0 时，系统会在传输数据流中插入一个中断字符。若配置了 SCI_STATBRK13] 参数，则会插入 10 到 13 或 13 到 16 的额外中断字符。只要 SBK 位保持激活状态，逻辑 0 的比特时间就会持续被排队处理。根据 SBK 位的设置与清除时机与当前传输信息的时间关系，软件可能在清空 SBK 位前就已排队了第二个中断字符。

1 = 将要发送的队列中断字符（秒）

0 = 正常发射机操作

MA1IE — 匹配 1 中断启用

1 = MA1F 中断已启用

0 = MA1F 中断被禁用

MA2IE — Match 2 中断启用

1 = MA2F 中断已启用

0 = MA2F 中断已禁用

IDLECFG — 空闲配置

配置在设置 idle 标志之前必须接收的空闲字符数。

000 -- 1 个空闲字符；

001-2 个空闲字符；

010 -- 4 个空闲字符；

011-8 个空闲字符；

100-16 个空闲字符；

101-32 空闲字符；

110-64 空闲字符；

111-128 个空闲字符

LOOPS — LOOP 模式选择

当设置 LOOPS 时，RXD 引脚与 SCI 断开连接，发送器输出在内部连接到接收器输入。必须启用发送器和接收器才能使用环路功能。

1 = 环路模式或单线模式，其中发射器输出与接收器输入内部连接（参见 RSRC 位）

0 = 正常操作- RXD 和 TXD 使用单独的引脚

DOZEEN — Doze Enable

1 = 在 Doze 模式下禁用 SCI

0 = 在 Doze 模式下启用 SCI

RSRC — 接收器源选择

除非设置了 LOOPS 字段，否则此字段没有意义或效果。当设置了 LOOPS 时，RSRC 字段确定接收器移位寄存器输入的源。

1 = 单线 SCI 模式，其中 TXD 引脚连接到发送器输出和接收器输入

0 = 已设置 LOOPS，RSRC 被清除，选择内部环回模式且 SCI 不使用 RXD 引脚

M-9bit 或 8bit 模式选择

1 = 接收器和发射器使用 9 位数据字符

0 = 接收器和发送器使用 8 位数据字符

WAKE — 接收器唤醒方法选择

当 RWU = 1 时，确定唤醒 SCI 的条件：接收数据字符最高有效位中的地址标记，或接收引脚输入信号的空闲状态。

1 = 配置 RWU 以使用 address-mark 唤醒

0 = 配置 RWU 以进行空闲线路唤醒

ILT — 沉寂线路类型选择

设定接收器将逻辑 1 计数为空闲字符位的起始时机。计数可能在有效起始位之后或停止位之后开始。若计数始于起始位之后，那么停止位前的逻辑 1 串可能导致误判为空闲字符。将计数起始点设在停止位之后虽能避免误判，但需要确保传输同步。当 SCI 编程设置 ILT = 1 时，接收停止位后会自动移除逻辑 0，从而重置空闲字符计数。

1 = 停止位之后开始的空闲字符数

0 = 在起始位之后开始闲置字符计数

PE — 等效性启用

启用奇偶校验生成和检查功能。启用奇偶校验时，将停止位之前的位视为奇偶校验位。

1 = 同步功能已启用

0 = 无硬件奇偶校验生成或检查

PT 平衡类型

如果提供了启用奇偶校验的选项（PE = 1），则此位选择偶校验或奇校验。奇校验意味着数据字符中包括奇偶校验位在内的 1 的总数为奇数。偶校验意味着数据字符中包括奇偶校验位在内的 1 的总数为偶数。

1 = 奇数

0 = 偶数平衡

21.7.8. SCI 数据登记

偏移地址：0x001C

	31	30	29	28	27	26	25	24
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	NOISY	PARITYE	FRETSC	RXEMPT	IDLINE	Reserved	R9T9	R8T8
W								
RESET	0	0	0	1	0	0	0	0
	7	6	5	4	3	2	1	0
R	R7T7	R6T6	R5T5	R4T4	R3T3	R2T2	R1T1	R0T0
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 21-10：SCI 数据寄存器（SCI_DATA）

噪声 — 当前在 DATA[R9:R0] 中包含的接收数据字含有噪声

1 = 接收数据时存在噪声

0 = 接收数据字时无噪声

PARITYE — 当前在 DATA[R9:R0] 中包含的接收数据字存在奇偶校验错误

1 = 接收到的数据字存在奇偶校验错误

0 = 接收到的数据字没有出现奇偶校验错误

FRETSC — 帧错误/发送特殊字符

对于读取操作，表示当前在 DATA[R9:R0] 中接收的数据字块存在帧错误。对于写入操作，则表示需要传输中断字符或空闲字符，而非 DATA[T9:T0] 中的内容。其中 T9 位用于指示：当为 0 时为中断字符，当为 1 时为空闲字符。

DATA[T8:T0] 应为零。

1 = 接收到的数据字存在帧错误，在发送时发送空闲或中断字符

0 = 在读取时，数据字未出现帧错误，但在写入时传输了一个正常字符

RXEMPT — 接收缓冲区为空

在接收缓冲区中没有数据时发出通知。此字段不考虑接收移位寄存器中的数据。

1 = 接收缓冲区为空，读取返回的数据无效

0 = 该接收缓冲区包含有效数据

IDLINE — 空闲线路

表示在接收 DATA[9:0] 中的字符之前，接收器线路处于空闲状态。与 idle 标志不同，此位可以设置为接收器首次启用时接收到的第一个字符。

1 = 在接收此字符之前，接收器处于空闲状态

0 = 接收器在接收此字符之前未处于空闲状态

R9T9 — 读取接收数据缓冲区 9 或写入发送数据缓冲区 9

R8T8 — 读取接收数据缓冲区 8 或写入发送数据缓冲区 8

R7T7 — 读取接收数据缓冲区 7 或写入发送数据缓冲区 7

R6T6 — 读取接收数据缓冲区 6 或写入发送数据缓冲区 6

R5T5 — 读取接收数据缓冲区 5 或写入发送数据缓冲区 5

R4T4 — 读取接收数据缓冲区 4 或写入发送数据缓冲区 4

R3T3 — 读取接收数据缓冲区 3 或写入发送数据缓冲区 3

R2T2 — 读取接收数据缓冲区 2 或写入发送数据缓冲区 2

R1T1 — 读取接收数据缓冲区 1 或写入发送数据缓冲区 1

R0T0 — 读取接收数据缓冲区 0 或写入发送数据缓冲区 0

21.7.9. SCI 匹配地址寄存器

偏移地址：0x0020

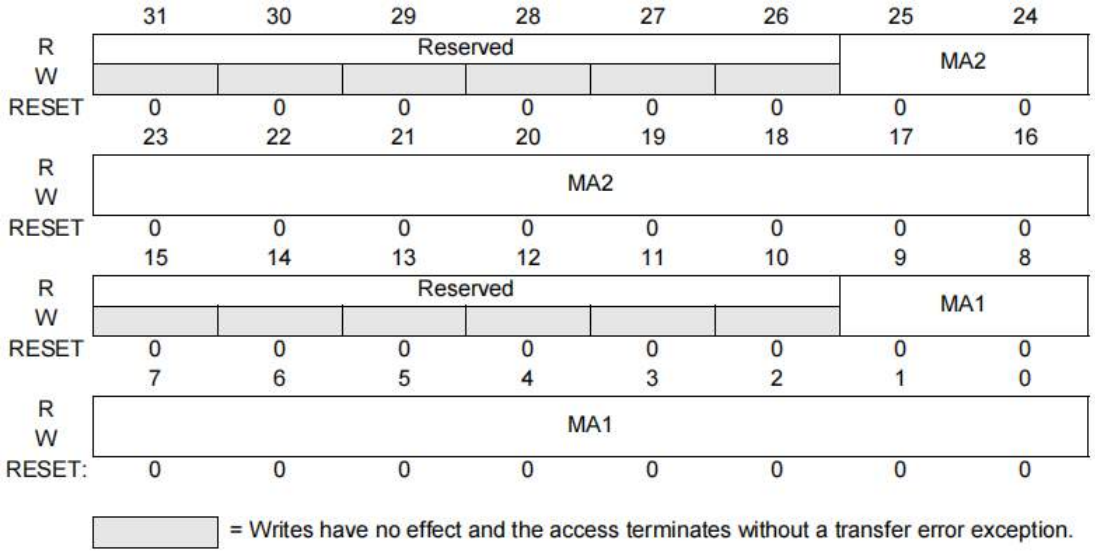


图 21-11：SCI 匹配地址寄存器（SCI_MATCH）

MA2 — 匹配地址 2

当最高有效位被置位且对应的 BAUD[MAEN] 位被置位时，系统会将 MA1 和 MA2 寄存器与输入数据地址进行比较。若匹配成功，则将后续数据传输至数据寄存器；若未匹配，则丢弃后续数据。软件仅应在对应 BAUD[MAEN] 位清零时写入 MA 寄存器。

MA1 — 匹配地址 1

当最高有效位被置位且对应的 BAUD[MAEN] 位被置位时，MA1 和 MA2 寄存器将与输入数据地址进行比较。若匹配成功，则将后续数据传输至数据寄存器；若匹配失败，则丢弃后续数据。软件仅应在对应的 BAUD[MAEN] 位被清除时写入 MA 寄存器。

21.7.10. SCI 调制解调器 IrDA 寄存器

偏移地址：0x0024

	31	30	29	28	27	26	25	24
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	Reserved					IREN	TNP	
W								
RESET	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	RTSWATER							
W								
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	Reserved		TXCTSS-RC	TXCTSC	RXRTSE	TXRT-SPOL	TXRTSE	TXCTSE
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 21-12：SCI 调制解调器 IrDA 寄存器（SCI_MODIR）

IREN — 启用红外功能

启用/禁用红外调制/解调。

1 = 使 IR 启用

0 = 禁用 IR

TNP — 发射器窄脉冲

允许 SCI 是否发送 1/OSR、2/OSR、3/OSR 或 4/OSR 窄脉冲。

00 -- 1/OSR;

01 -- 2/OSR;

10 -- 3/OSR;

11 -- 4/OSR

RTSWATER — 接收 RTS 配置

根据接收 FIFO 中可存储的附加字符数来配置 RX RTS 输出取消的点。当配置为 0 时，将在检测到将导致 FIFO 满的字符的起始位时取消 RTS。

1 = 当接收 FIFO 小于或等于 RXWATER 配置时，RTS 断言；当接收 FIFO 大于 RXWATER 配置时，RTS 否定。

0 = RTS 在接收器 FIFO 充满或接收导致 FIFO 变为满的字符时发出断言。

TXCTSSRC — 发送 CTS 源

配置 CTS 输入的源。

1 = CTS 输入是接收机匹配结果的反相

0 = CTS 输入是 SCI_CTS 引脚

TXCTSC — 发送 CTS 配置

配置 CTS 状态是否在每个字符开始时检查，还是仅在发送器处于空闲状态时检查。

1 = 发射器处于空闲状态时，对 CTS 输入进行采样

0 = 在每个字符开始时对 CTS 输入进行采样

RXRTSE — 接收器请求发送功能启用

允许 RTS 输出控制发送设备的 CTS 输入，以防止接收器溢出。请勿同时设置 RXRTSE 和 TXRTSE。

1：若接收方数据寄存器已满或检测到可能导致其满载的起始位，则会取消 RTS 信号。当接收方数据寄存器未满且未检测到可能使其满载的起始位时，RTS 信号将被激活。RTSWATER 字段用于配置 RTS 信号的激活状态。

0 = 接收器对 RTS 没有影响

TXRTSPOL — 发射机请求发送极性

控制发送器 RTS 的极性。TXRTSPOL 不影响接收器 RTS 的极性。除非设置了 TXRTSE，否则 RTS 将保持在低电平有效状态。

1 = 发射器 RTS 处于高电平有效状态

0 = 发射器 RTS 处于低电平有效状态

TXRTSE — 发射机请求发送启用

在传输前后控制 RTS。

1 = 当字符被放入空的发送器数据缓冲区时，RTS 会在开始位传输前一个比特时间内激活。RTS 在发送器数据缓冲区和移位寄存器中的所有字符（包括最后一个停止位）完全发送后一个比特时间内解除激活。

0 = 发射器对 RTS 没有影响

TXCTSE — 发送器允许发送

TXCTSE 控制发射器的操作。TXCTSE 可以独立于 TXRTSE 和 RXRTSE 的状态进行设置。

1 = 启用发送许可操作。每次准备发送字符时，发送器都会检查 CTS 状态。若 CTS 被激活，则发送该字符；若 CTS 未激活，TXD 信号将保持标记状态，传输会延迟至 CTS 被激活。字符发送过程中 CTS 的状态变化不会影响其传输。

0 = CTS 对发送器没有影响

21.7.11. SCI FIFO 寄存器

偏移地址：0x0028

	31	30	29	28	27	26	25	24
R	Reserved							
W								
RESET	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	TXEMPT	RXEMPT	Reserved				TXOF	RXUF
W							w1c	w1c
RESET	1	1	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	0	0	Reserved	RXIDEN			TXOFE	RXUFE
W	TXFLUSH	RXFLUSH						
RESET	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	TXFE	TXFIFOSIZE			RXFE	RXFIFOSIZE		
W								
RESET:	0	0	1	0	0	0	1	0

= Writes have no effect and the access terminates without a transfer error exception.

图 21-13：SCI FIFO 寄存器（SCI_FIFO）

TXEMPT — 发送缓冲区/FIFO 为空

在 Transmit FIFO/缓冲器中没有数据时，该字段会声明。此字段不考虑发送移位寄存器中的数据。

1 = 传输缓冲区为空

0 = 传输缓冲区不为空

RXEMPT — 接收缓冲区/FIFO 空

当接收 FIFO/Buffer 中没有数据时，该字段将被声明。此字段不考虑接收移位寄存器中的数据。

1 = 接收缓冲区为空

0 = 未清空接收缓冲区

TXOF — 发射器缓冲区溢出标志

表示已向发送缓冲区写入的数据量超过了其容量。无论 TXOFE 的值为何，此字段都将断言。但是，只有当 TXOFE 被设置时才会向主机发出中断。通过写入 1 可清除此标志。

1 = 自上次清除标志以来，至少发生过一次发送缓冲区溢出

0 = 自上次清除标志以来，未发生传输缓冲区溢出

RXUF — 接收器缓冲区下溢标志

表示从接收缓冲区读取的数据量大于其当前容量。无论 RXUFE 的值如何，此字段都将置

为真。但是，只有当 RXUFE 被设置时才会向主机发出中断。通过写入 1 可清除此标志。

1 = 自上次清除标志以来，至少发生一次接收缓冲区下溢

0 = 自上次清除标志以来，未发生接收缓冲区下溢

TXFLUSH — 传输 FIFO/缓冲区刷新

向该字段写入数据将导致发送 FIFO/缓冲器中存储的所有数据被清除。这不会影响发送移位寄存器中的数据。

1 = 发送 FIFO/缓冲器中的所有数据被清除

0 = 不发生冲洗操作

RXFLUSH — 接收 FIFO/缓冲区刷新

向该字段写入数据将导致接收 FIFO/缓冲器中存储的所有数据被清除。这不会影响接收移位寄存器中的数据。

1 = 接收 FIFO/缓冲器中的所有数据被清除

0 = 不发生冲洗操作

RXIDEN — 接收器空闲空闲启用

设置时，当接收器处于空闲状态且 FIFO 未满时，启用对 RDRF 的断言。

000 = 接收器处于空闲状态时，由于 FIFO 部分填满，禁用 RDRF 断言；

001 = 当接收器空闲 1 个字符时，由于 FIFO 部分填充，启用 RDRF 断言；

010 = 当接收器空闲 2 个字符时，由于 FIFO 部分填充，启用 RDRF 断言；

011 = 当接收器空闲 4 个字符时，由于 FIFO 部分填充，启用 RDRF 断言；

100 = 当接收器空闲 8 个字符时，由于 FIFO 部分填充，启用 RDRF 断言；

101 = 当接收器空闲 16 个字符时，由于 FIFO 部分填满，启用 RDRF 断言；

110 = 当接收器空闲 32 个字符时，由于 FIFO 部分填满，启用 RDRF 断言；

111 =当接收器空闲 64 个字符时，由于 FIFO 部分填满，启用 RDRF 断言。

TXOFE — 传输 FIFO 溢出中断使能

设置此字段时，TXOF 标志将向主机生成中断。

1 = TXOF 标志向主机生成中断

0 = TXOF 标志不会向主机生成中断

RXUFE — 接收 FIFO 下溢中断使能

设置此字段时，RXUF 标志将向主机生成中断。

1 = TRXUF 标志向主机生成中断

0 = RXUF 标志不会向主机生成中断

TXFE — 发送 FIFO 启用

当该字段被设置时，将启用发送缓冲器的内置 FIFO 结构。FIFO 结构的大小由 TXFIFOSIZE 指示。如果未设置该字段，则发送缓冲器将作为一个深度为 1 的数据字 FIFO 运行，无论 TXFIFOSIZE 的值为何。在更改该字段之前，必须清除 CTRL[TE] 和 CTRL[RE]。

1 = 发送 FIFO 已启用。缓冲区深度由 TXFIFOSIZE 指示。

0 = 发送 FIFO 未启用。缓冲区深度为 1。（旧式支持）

TXFIFOSIZE — 传输 FIFO/缓冲深度

发送缓冲器中可以存储的最大发送数据字数。此字段只读。

000 =传输 FIFO/缓冲区深度 = 1 个数据字;

001 = 传输 FIFO/Buffer 深度 = 4 个数据字;

010 = 传送 FIFO/缓冲区深度 = 8 个数据字;

011 = 传送 FIFO/缓冲区深度 = 16 个数据字;

100 = 传送 FIFO/缓冲深度 = 32 个数据字;

101 = 传送 FIFO/缓冲区深度 = 64 个数据字;

110 = 传输 FIFO/缓冲区深度 = 128 个数据字;

111 = 传输 FIFO/缓冲深度 = 256 个数据字。

RXFE — 接收 FIFO 使能

当设置此字段时，接收缓冲器的内置 FIFO 结构将被启用。FIFO 结构的大小由 RXFIFOSIZE 字段指示。如果未设置此字段，接收缓冲器将作为深度为一个数据字的 FIFO 运行，无论 RXFIFOSIZE 字段的值如何。在更改此字段之前，必须清除 CTRL[TE] 和 CTRL[RE]。

1 = 接收 FIFO 被启用。缓冲器的深度由 RXFIFOSIZE 指示。

0 = 未启用接收 FIFO。缓冲区深度为 1。（旧式支持）

RXFIFOSIZE — 接收 FIFO/缓冲深度

接收缓冲区中可以存储的最大发送数据字数。此字段只读。

000 = 接收 FIFO/缓冲区深度 = 1 个数据字;

001 = 接收 FIFO/Buffer 深度 = 4 个数据字;

010 = 接收 FIFO/缓冲区深度 = 8 个数据字;

011 = 接收 FIFO/缓冲区深度 = 16 个数据字;

100 = 接收 FIFO/缓冲区深度 = 32 个数据字;

101 = 接收 FIFO/缓冲区深度 = 64 个数据字;

110 = 接收 FIFO/缓冲区深度 = 128 个数据字;

111 = 接收 FIFO/缓冲深度 = 256 个数据字。

21.7.12. SCI 水印（Watermark）寄存器

偏移地址：0x002C

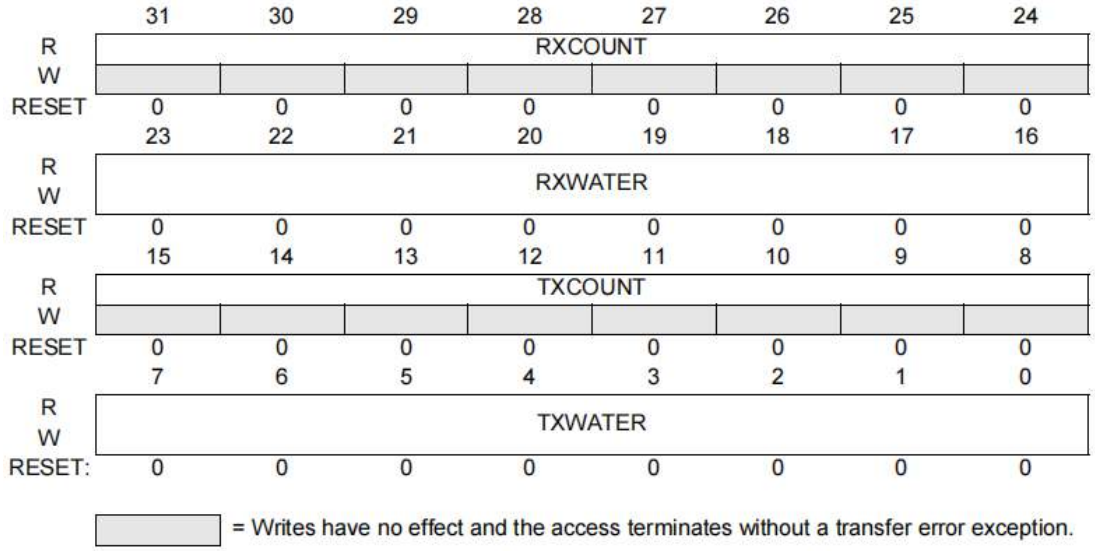


图 21-14：SCI Watermark 寄存器（SCI_WATER）

RXCOUNT — 接收计数器

此寄存器的值表示接收 FIFO/缓冲器中的数据字数。如果正在接收一个数据字，即在接收移位寄存器中，则不包括在计数值内。此值可与 FIFO[RXFIFOSIZE] 一起使用，以计算接收 FIFO/缓冲器中剩余的空间。

RXWATER — 接收水印

当接收 FIFO/缓冲器中的数据字数大于该寄存器字段中的值时，将生成中断。为确保正常运行，RXWATER 中的值必须设置为小于由 FIFO[RXFIFOSIZE] 和 FIFO[RXFE] 指示的接收 FIFO/缓冲器大小，并且必须大于 0。

TXCOUNT — 发送计数器

此寄存器的值表示发送 FIFO/缓冲器中的数据字数。如果正在发送数据字，则表示该数据字在发送移位寄存器中，因此不包括在计数中。此值可与 FIFO[TXFIFOSIZE] 一起使用，以计算发送 FIFO/缓冲器中剩余的空间。

TXWATER — 传输水印

当发送 FIFO/缓冲器中的数据字数等于或小于该寄存器字段的值时，将生成中断。为确保正常运行，TXWATER 的值必须设置为小于发送缓冲器/FIFO 的大小，如 FIFO[TXFIFOSIZE] 和 FIFO[TXFE] 所示。

21.7.13. SCI 过采样比率寄存器

偏移地址：0x0030

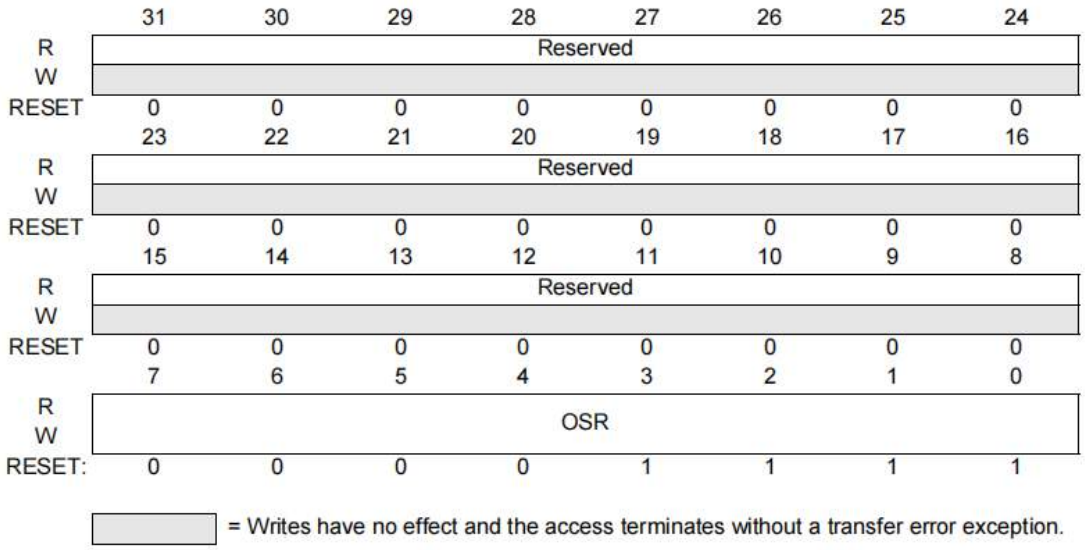


图 21-15：SCI 过采样比率寄存器（SCI_OSR）

OSR — 过采样比率

该字段用于配置接收端的过采样率，范围在 4 倍（00000011）到 256 倍（11111111）之间。若输入无效的过采样率（例如超出 4 倍至 256 倍范围的数值），系统将默认设置为 16 倍（00001111）。

提示：仅当发射端和接收端均处于禁用状态时，方可修改 OSR 字段。特别说明：过采样率等于 OSR + 1。

21.8. 功能说明

SCI 支持全双工、异步、NRZ 串行通信，包含波特率发生器、发送器和接收器模块。发送器和接收器独立工作，但它们使用相同的波特率发生器。以下介绍了 SCI 的每个模块。

21.9. 波特率生成

波特率生成器中的 13 位模数计数器负责为收发双方计算波特率。写入 SBR[12:0] 寄存器的 1 至 8191 数值，决定了异步 SCI 波特时钟的分频比，这些 SBR 寄存器位于 SCI 波特率寄存器 BDH 和 BDL 中，接收端由波特时钟驱动，而发送端则通过过采样比率分频后的时钟进行控制。根据不同的过采样比率设置，接收端的采样速率可达每比特时间 4 到 256 次采样。

波特率生成受以下两个误差来源的影响：

- 通过异步 SCI 波特时钟进行的整数除法可能无法给出精确的目标频率。
- 与异步 SCI 波特时钟同步可能导致相位偏移

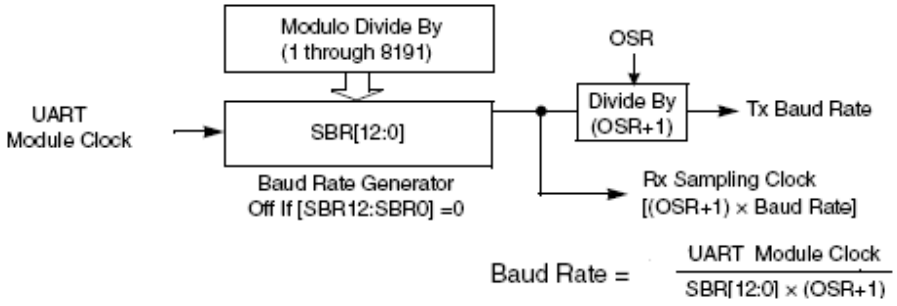


图 21-16：SCI 波特率生成

21.10. 变送器功能描述

本节描述了 SCI 发送器的整体方块图，以及发送中断和空闲字符的专用功能。

发射器输出（TXD）的空闲状态默认为高电平，CTRL[TXINV] 寄存器在复位后会被清零。通过设置 CTRL[TXINV] 可对发射器输出进行反相操作。启用发射器需要配置 CTRL[TE] 位，该操作会将前导码序列（相当于空闲状态下的完整字符帧）存入队列。发射器将保持空闲状态，直到发送数据缓冲区有可用数据时才会重新激活。程序通过向 SCI 数据寄存器写入数据来完成向发送数据缓冲区的存取操作。

SCI 发送器的核心元件是发送移位寄存器，其长度为 10 到 13 位，具体取决于 CTRL[M] 、BAUD[M10] 和 BAUD[SBNS] 控制位。在本节后续内容中，假设 CTRL[M]、BAUD[M10] 和 BAUD[SBNS] 均处于清除状态，此时系统将选择常规的 8 位数据模式。在 8 位数据模式下，移位寄存器会存储起始位、八个数据位和停止位。当发送移位寄存器准备好接收新字符时，发送数据寄存器中的待处理值将被传输至该寄存器，并与波特率时钟保持同步。同时，发送数据寄存器的空闲状态（STAT[TDRE]）标志会被置为有效，这表明 SCI_DATA 接口处的发送缓冲区可继续接收新字符写入。

如果在停止位移出 TXD 引脚后，发送数据缓冲区中没有新的字符等待，则发射器设置发送完成标志并进入空闲模式，TXD 高电平，等待更多字符传输。

在写入 0 到 CTRL[TE] 时，发送器不会立即停止工作。必须先完成当前的发送活动（可能包括数据字符、空闲字符或中断字符），尽管发送器不会开始发送另一个字符。

21.10.1. 发送中断和排队空闲

SCI_CTRL[SBK] 位发送的中断字符最初用于吸引旧式电传打字机接收器的注意。中断字符是一个完整的逻辑 0 字符时间，包含起始位、停止位以及 10 到 12 比特的时间。通过设置 SCI_STAT[BRK13] 可启用更长的 13 比特中断时间。通常情况下，程序会等待 SCI_STAT[TDRE] 被置位以表明信息末个字符已移至发送移位器，写入 1 后向 SCI_CTRL[SBK] 位写入 0。此操作会将中断字符排队待发，一旦移位器就绪即可发送。若当排队的中断

字符进入移位器时（同步于波特率时钟），SCI_CTRL[SBK] 仍保持 1，则会额外排队一个中断字符。若接收设备是另一款飞思卡尔半导体公司的 SCI 设备，中断字符会在所有数据位中被接收为 0，并触发帧错误（SCI_STAT[FE] = 1）。

中断字符也可以通过向 SCI_DATA 寄存器写入数据来发送，同时将第 13 位设置为 1，并清除数据位。这样就可以将中断字符作为正常数据流的一部分进行发送。

当启用空闲线唤醒功能时，需要在信息之间预留完整的空闲字符周期（逻辑 1 状态）来唤醒处于休眠状态的接收器。通常情况下，程序会等待 SCI_STAT[TDRE] 置位以确认信息末字符已移至发送移位寄存器，随后依次向 SCI_CTRL[TE] 位写入 0 和 1。这一操作会在移位寄存器就绪时立即排队发送空闲字符。只要 SCI_CTRL[TE] 位保持清除状态，且移位寄存器中的字符未完成传输，SCI 发送器就永远不会真正释放 TXD 引脚的控制权。

若要发送空闲字符，可将 SCI_DATA 寄存器写入 13 位设置且数据位也设置的状态。这样支持将空闲字符作为正常数据流的一部分进行发送。

如下面所示，中断字符的长度受 SCI_STAT[BRK13]、SCI_CTRL[M]、SCI_BAUD[M10] 和 SCI_BAUD[SNBS] 位的影响。

表 21-3：断字符长度

BRK13	M	M10	SBNS	Break Character Length
0	0	0	0	10 比特时间
0	0	0	1	11 位时间
0	1	0	0	11 位时间
0	1	0	1	16 位时间
0	X	1	0	16 位时间
0	X	1	1	13 位时间
1	0	0	0	13 位时间
1	0	0	1	13 位时间
1	1	0	0	16 位时间
1	1	0	1	14 位时间
1	X	1	0	15 比特时间
1	X	1	1	15 比特时间

21.11. 接收器功能描述

本节中，接收机方块图是整个接收机功能描述的指南。接下来，更详细地描述了用于重建接收机数据的数据采样技术。最后，解释了接收机唤醒功能的不同变体。

通过设置 SCI_STAT[RXINV] 寄存器可实现接收端输入信号的反转功能。启用接收端时需配置 SCI_CTRL[RE] 位。字符帧由逻辑 0 的起始位、8 至 10 个数据位（最高有效位或最低有效位优先）以及 1 个或 2 个逻辑 1 的

停止位组成。关于 9 位或 10 位数据模式的具体说明，请参阅 8 位、9 位和 10 位数据模式的相关内容。在后续讨论中，我们将默认 SCI 配置为标准 8 位数据模式。

当接收移位器接收到停止位后，若接收数据寄存器尚未满载，则会将数据字符传输至寄存器并触发满载状态（SCI_STAT[RDRF]）标志。若此时 SCI_STAT[RDRF] 已处于满载状态（即缓冲区已满），系统将触发溢出（OR）状态标志，导致新数据丢失。由于 SCI 接收器采用双缓冲设计，在 SCI_STAT[RDRF] 被置位后，程序需等待一个字符传输周期才能读取缓冲区数据，从而避免接收器发生溢出。

当程序检测到接收数据寄存器已满（SCI_STAT[RDRF] = 1）时，通过读取 SCI_DATA 从接收数据寄存器获取数据。有关清除标志的详细信息，请参阅中断和状态标志。

21.11.1. 数据采样技术

SCI 接收器支持将波特率时钟的采样速率配置为 4×至 256×。接收器启动时，会以过采样速率乘以波特率进行逻辑电平采样，通过 RXD 串行数据输入引脚检测下降沿。所谓下降沿，是指在连续三个逻辑 1 样本后出现的逻辑 0 样本。过采样波特率时钟将比特时间划分为 4 到 256 个分段（OSR 为配置的过采样比率）。当检测到下降沿后，系统会在 OSR/2、OSR/2+1 和 OSR/2+2 位置各采集三个样本，以确认该信号为有效起始位而非噪声干扰。若这三个样本中至少有两个为 0，则接收器判定已与接收到的字符同步。若在接收器认为同步前再次检测到下降沿，系统将从首个采样分段重新开始采样。

接收端会对每个比特时间（包括起始位和结束位）进行采样，采样点分别位于（OSR/2）、（OSR/2）+1 和（OSR/2）+2 处，以此确定该比特的逻辑电平。逻辑电平的判定依据是该比特时间内多数采样点的电平值。若字符帧中任何比特时间（包括起始位和结束位）的采样点与逻辑电平不一致，则当接收端将字符传输至数据缓冲区时，会触发噪声标志（SCI_STAT[NF]）的设置。

当 SCI 接收器配置为在波特率时钟的两个边沿进行采样时，每个接收比特中的数据段数量实际上会翻倍（从 1 增加到 OSR×2）。此时起始位和数据位将分别以 OSR、OSR+1 和 OSR+2 的采样率进行采样。对于 4×至 7×的过采样率，必须启用时钟双边沿采样功能；而对于更高倍数的过采样率，则可选择性启用该功能。

下降沿检测逻辑会持续监测信号的下降沿。一旦检测到信号边沿，系统就会将采样时钟重新同步至比特时间（除非已禁用该功能）。这种机制能有效提升接收端在噪声干扰或波特率失配情况下的可靠性。但需注意的是，由于某些字符在帧内可能完全不包含额外的下降沿信号，因此该机制无法改善最坏情况分析的准确性。

在出现帧错误的情况下，只要接收到的字符不是中断字符，搜索下降沿的采样逻辑就会被三个逻辑 1 的样本填满，以便几乎可以立即检测到新的起始位。

21.11.2. 接收器唤醒操作

接收器唤醒和接收器地址匹配是允许 SCI 接收器忽略发送给其他接收器的信息中的字符的硬件机制。

在接收器唤醒阶段，所有接收器都会先解析每条信息的首字符（或多个字符）。一旦确认该信息是发往其他接收器的，就会立即向接收器唤醒控制位（SCI_CTRL[RWU]）写入逻辑 1。当 RWU 位和 SCI_S2[RWUID] 位被

置位时，除空闲位 idle 外，与接收器相关的状态标志将被禁止设置，从而省去处理无关信息字符的软件开销。每当信息结束或新信息开始时，所有接收器会自动将 SCI_CTRL[RWU] 清零，确保所有接收器都能及时唤醒并处理下一条信息的首字符（或多个字符）。

在接收器地址匹配过程中，硬件中执行地址匹配，SCI 接收器将忽略所有不符合地址匹配要求的字符。

如下面所示，中断字符的长度受 SCI_STAT[BRK13]、SCI_CTRL[M]、SCI_BAUD[M10] 和 SCI_BAUD[SNBS] 位的影响。

表 21-4：接收器唤醒选项

罗致恒富	MA1 | MA2	MAT 配置文	沃克：鲁伊德	接收器唤醒
0	0	X	X	正常运行
1	0	00	00	接收器在空闲线路上唤醒，idle 标志未设置
1	0	00	01	接收器在空闲线路上唤醒，空闲标志被设置
1	0	00	10	接收器在地址标记处唤醒
1	1	11	0	数据匹配时接收器唤醒
0	1	00	X0	地址标记地址匹配，丢弃时未设置 IDLE 标志 字符
0	1	00	X1	地址标记地址匹配，IDLE 标志设置用于丢弃字符
0	1	01	X0	空闲行地址匹配
0	1	10	X0	地址匹配打开和关闭，IDLE 标志未设置为已丢弃的字符
0	1	10	X1	地址匹配打开和关闭，IDLE 标志用于丢弃字符

21.11.2.1. 怠速线性唤醒

当空闲线唤醒功能被激活时，接收器将切换至空闲线唤醒模式。在此模式下，当检测到空闲线处于满字符时间状态时，SCI_CTRL[RWU] 寄存器会自动清零。SCI_CTRL[M] 和 SCI_BAUD[M10] 控制位用于选择 8 位至 10 位的数据传输模式，而 SCI_BAUD[SBNS] 位则用于设置 1 位或 2 位的停止位数量 — 由于起始和停止位的存在，总需要 10 到 13 个比特时间来构成一个完整的字符周期。

当 SCI_CTRL[RWU] 为 1 且 SCI_STAT[RWUID] 为 0 时，唤醒接收器的空闲状态不会设置 SCI_STAT[IDLE] 标志。此时接收器会立即响应并等待下一条信息的第一个数据字符，该字符将触发 SCI_STAT[RDRF] 标志的设置，若该标志处于启用状态则会触发中断。当 SCI_STAT[RWUID] 为 1 时，无论 SCI_CTRL[RWU] 是 0 还是 1，任何空闲状态都会设置 SCI_STAT[IDLE] 标志，并在启用中断功能的情况下触发中断。

空闲行控制位（SCI_CTRL[ILT] ）采用两种检测方式中的一种。当 SCI_CTRL[ILT] 被清除时，空闲位计数器会在起始位之后开始计数，因此停止位和字符末尾的逻辑 1 都会计入空闲时间。当 SCI_CTRL[ILT] 被置位时，空闲位计数器需等到停止位之后才开始计数，这样空闲检测就不会受到前一条信息末尾数据的影响。

21.11.2.2. 地址标记-唤醒

当设置 SCI_CTRL[WAKE] 时，接收器被配置为地址标记唤醒模式。在此模式下，当接收器检测到接收字符的最高有效位为逻辑 1 时，SCI_CTRL[RWU] 将自动清除。

address_must wakeup 允许信息包含空闲字符，但要求最高有效位必须保留用于地址帧。地址帧最高有效位中的逻辑 1 会在停止位接收前清除 SCI_CTRL[RWU] 位，并设置 SCI_STAT[RDRF] 标志。在这种情况下，即使接收器在此字符传输期间大部分时间处于休眠状态，仍会接收到最高有效位被设置的字符。

21.11.2.3. 数据匹配唤醒

当 SCI_CTRL[RWU] 被设置且 SCI_BAUD[MATCFG] 等于 11 时，接收器配置为数据匹配唤醒。在此模式下，当 BAUD[MAEN1] 被设置时，如果接收器检测到与 MATCH[MA1] 字段匹配的字符，则 SCI_CTRL[RWU] 将自动清除；当 BAUD[MAEN2] 被设置时，如果检测到与 MATCH[MA2] 匹配的字符，则 SCI_CTRL[RWU] 将自动清除。

21.11.2.4. 地址匹配操作

地址匹配操作在 SCI_BAUD[MAEN1] 或 SCI_BAUD[MAEN2] 位被置位且 SCI_BAUD[MATCFG] 等于 00 时启用。该功能中，若 RXD 引脚接收到的字符在停止位前一个比特位置为逻辑 1，则视为地址并与其对应的 MATCH[MA1] 或 MATCH[MA2] 字段进行比对。若匹配成功，该字符将被传输至接收缓冲区，并设置 SCI_STAT[RDRF] 标志。对于停止位前一个比特位置为逻辑 0 的后续字符，系统将其视为与该地址相关的数据并传输至接收数据缓冲区。若未检测到匹配地址，则不进行传输操作，同时所有后续停止位前一个比特位置为逻辑零的字符也会被丢弃。当 SCI_BAUD[MAEN1] 和 SCI_BAUD[MAEN2] 两位均被置为非时，接收器正常工作，所有接收到的数据都会传输至接收数据缓冲区。

对于 MATCH[MA1] 和 MATCH[MA2] 字段，Address match 操作的功能相同。

- 如果 SCI_BAUD[MAEN1] 和 SCI_BAUD[MAEN2] 中仅有一个被激活，则只将标记地址与关联的匹配寄存器进行比较，只有在匹配时才将数据传输到接收数据缓冲区。
- 如果 SCI_BAUD[MAEN1] 和 SCI_BAUD[MAEN2] 被置为有效，则将标记地址与两个匹配寄存器进行比较，仅当与任一寄存器匹配时才传输数据。

21.11.2.5. 空闲匹配操作

当 SCI_BAUD[MAEN1] 或 SCI_BAUD[MAEN2] 位被置位且 SCI_BAUD[MATCFG] 等于 01 时，空闲匹配操作将被激活。在此功能中，RXD 引脚在空闲线路状态后接收到的第一个字符将被视为地址，并与对应的 MA1 或 MA2 寄存器进行比对。若匹配成功，该字符将被传输至接收缓冲区，并触发 SCI_STAT[RDRF] 状态。所有后续字符均视为与该地址相关联的数据，会持续传输至接收数据缓冲区直至检测到下一次空闲线路状态。若未发生地址匹配，则不会向接收数据缓冲区传输任何数据，且后续所有帧数据也将被丢弃。当 SCI_BAUD[MAEN1] 和 SCI_BAUD[MAEN2] 两位同时被置为无效时，接收器将正常工作，所有接收到的数据都会传输至接收数据缓冲区。

空闲匹配操作功能对 MA1 和 MA2 寄存器都具有相同的作用。

- 如果 SCI_BAUD[MAEN1] 和 SCI_BAUD[MAEN2] 中仅有一个被激活，则空闲行之后的第一个字符将只与关联的匹配寄存器进行比较，只有在匹配时才将数据传输到接收数据缓冲区。
- 如果 SCI_BAUD[MAEN1] 和 SCI_BAUD[MAEN2] 被置为有效，则空闲行之后的第一个字符将与两个匹配寄存器进行比较，只有当与任一寄存器匹配时才传输数据。

21.11.2.6. 匹配与解除操作

当 SCI_BAUD[MAEN1] 和 SCI_BAUD[MAEN2] 两位均被置位且 SCI_BAUD[MATCFG] 等于 10 时，匹配开启/关闭功能即被激活。在此功能中，RXD 引脚接收到的符合 MATCH[MA1] 寄存器的字符会被接收并传输至接收缓冲区，同时 SCI_STAT[RDRF] 寄存器被置位。所有后续字符均被视为有效数据并传输至接收数据缓冲区，直至接收到符合 MATCH[MA2] 寄存器的字符为止。此时系统会丢弃符合 MATCH[MA2] 的字符及其后续所有字符，这一过程将持续执行，直到再次接收到符合 MATCH[MA1] 的字符为止。若 SCI_BAUD[MAEN1] 和 SCI_BAUD[MAEN2] 两位均被置为非，则接收器正常工作，所有接收到的数据都会传输至接收数据缓冲区。

21.11.3. 红外解码器

该红外解码器将从 IrDA 格式接收的字符转换为接收器使用的 NRZ 格式。它还具有 OSR 过采样波特率时钟计数器，用于过滤噪声并指示何时接收到 1。

21.11.3.1. 启动比特检测

当 STAT[RXINV] 信号被清除时，接收到的字符首个下降沿即对应起始位。红外解码器随即重置其计数器，此时接收端也启动起始位检测流程。完成起始位检测后，接收端会将自身比特时间与该起始位时间进行同步。在后续字符接收过程中，红外解码器的计数器与接收端的比特时间计数器将各自独立运行。

21.11.3.2. 噪声滤波

在红外解码器计数器的前半段检测到的任何上升沿都会被解码器忽略。任何小于一个过采样波特时钟周期的脉冲，无论出现在计数器的前半段还是后半段，都可能被解码器漏检。

21.11.3.3. 低比特检测

在解码器计数的后半部分，将上升沿解码为 0，并将其发送给接收器。同时，解码器计数器也被重置。

21.11.3.4. 高比特检测

在 OSR 过采样波特率时钟之后的前一个上升沿，如果未看到上升沿，则解码器向接收器发送 1。

如果下一个比特为 0，而它到达较晚，则根据低比特检测，检测到低比特。发送给接收机的值从 1 变为 0。然后，如果噪声脉冲发生在接收机比特时间采样周期之外，则 0 的延迟不会被记录为噪声。

21.12. 其他 SCI 功能

以下部分描述了其他 SCI 函数。

21.12.1. 8bit、9bit 和 10bit 数据模式

SCI 收发器可通过配置 SCI_CTRL[M] 寄存器设置为 9 位数据模式，或通过设置 SCI_CTRL[M10] 寄存器切换至 10 位数据模式。在 9 位模式下包含第九个数据位，在 10 位模式下则增加第十个数据位。对于发送端数据缓冲区，这些数据位存储在 SCI_CTRL[T8] 和 SCI_CTRL[T9] 寄存器中；接收端则保存于 SCI_CTRL[R8] 和 SCI_CTRL[R9] 寄存器。此外，这些数据位也可通过 16 位或 32 位访问 SCI_DATA 寄存器进行读取。

对于传输数据缓冲区的连续 8 位写入操作，请先将数据写入 SCI_CTRL[T8] 和 SCI_CTRL[T9] ，然后再写入 SCI_DATA[7:0] 。对于向 SCI_DATA 寄存器的 16 位和 32 位写入操作，所有 10 个发送位将同时写入传输数据缓冲区。

若新字符的第九和第十位传输比特值与前一个字符相同，则无需再次写入 SCI_CTRL[T8] 和 SCI_CTRL[T9] 。当数据从发送缓冲区传输至移位器时，SCI_CTRL[T8] 和 SCI_CTRL[T9] 的值会与数据从 SCI_DATA[7:0] 传输到移位器的过程同步进行。

9 比特数据模式通常与奇偶校验配合使用，使得第八位数据加上第九位奇偶校验形成完整数据；或采用地址标记唤醒机制时，第九位数据可作为唤醒信号位。10 比特数据模式则通常结合奇偶校验和地址标记唤醒，此时第九位数据充当唤醒信号位，第十位则作为奇偶校验位。在定制协议中，第九位和/或第十位还可作为软件控制的标记位使用。

21.12.2. 空闲长度

空闲字符是指起始位、所有数据位和停止位均位于标记位置的字符。可配置 CTRL[ILT] 寄存器，以从上一个起始位（所有数据位和停止位都计入空闲字符检测）或从上一个停止位开始检测空闲字符。

也可以使用 CTRL[IDLECFG] 字段配置检测空闲行条件之前必须接收的空闲字符数。此字段配置在设置 STAT[IDLE] 标志、清除 STAT[RAF] 标志以及用下一个接收到的字符设置 DATA[IDLINE] 标志之前必须接收的

空闲字符数。

空闲线唤醒和空闲匹配功能同样受 CTRL[IDLECFG] 字段影响。当启用地址匹配或匹配开关功能时，设置 STAT[RWUID] 位会将所有被丢弃的字符视为空闲字符处理，该机制独立于外部系统连接状态，有助于系统故障排查。在此模式下，发送器输出端与接收器输入端形成内部直连，此时 SCI 模块不会占用 RXD 引脚。

21.12.3. 单线操作

当 SCI 控制器的[循环模式] 寄存器（SCI_CTRL[LOOPS]）被设置时，同一寄存器中的[资源分配] 位（SCI_ctrl[RSRC]）会切换工作模式：当该位为 0 时启用循环模式，为 1 时则切换至单线模式。循环模式常用于独立于外部系统连接的软件测试，有助于快速定位系统故障。在此模式下，发送端输出端与接收端输入端形成内部直连，此时 SCI 不会占用 RXD 引脚。

21.12.4. 循环模式

当 SCI_CTRL[LOOPS] 被设置时，同一寄存器中的 RSRC 位会在环路模式（SCI_CTRL[RSRC] = 0）和单线模式（SCI_CTRL[RSRC] = 1）之间进行选择。单线模式实现半双工串行连接，接收器内部连接至发送器输出端口和 TXD 引脚（RXD 引脚未被使用）。

在单线模式下，SCI_CTRL[TXDIR] 位控制 TXD 引脚上的串行数据传输方向。当 SCI_CTRL[TXDIR] 被清除时，TXD 引脚作为接收端的输入端口，此时发送器会暂时断开与 TXD 引脚的连接，以便外部设备向接收器发送串行数据。当 SCI_CTRL[TXDIR] 被置位时，TXD 引脚则由发送器驱动作为输出端口，此时内部回环连接被禁用，导致接收器无法接收发送器发出的数据字符。

21.13. 红外接口

SCI 接口具备向红外 LED 发送窄脉冲信号、接收窄脉冲并将其转换为串行比特的能力，这些数据随后会被传输至 SCI。IrDA 物理层规范定义了用于数据交换的半双工红外通信链路，完整标准支持最高 16Mbit/s 的数据传输速率，而本设计仅涵盖 2.4kbit/s 至 115.2kbit/s 之间的数据传输速率范围。

SCI 协议配备红外发射编码器和接收解码器。在发送数据时，SCI 通过红外子模块将串行数据位编码为窄脉冲传输，每个零位对应一个窄脉冲，而每个比特位则不产生脉冲。接收端采用红外光电二极管检测红外脉冲信号，经外部红外接收解码器转换为 CMOS 电平信号。随后，该解码器会将窄脉冲信号拉伸为串行比特流，供 UART 模块接收。通过反转发送脉冲与预期接收脉冲的极性，可实现与采用高电平有效脉冲的外部红外数据收发器模块直接对接。

红外子模块从 SCI 获取时钟源。在传输过程中，红外子模块选择其中的一个时钟来生成 1/OSR、2/OSR、3/OSR 或 4/OSR 窄脉冲。

21.13.1. 红外发射编码器

红外传输编码器将发射移位寄存器中的串行数据位转换为 TXD 信号。对于零位会发送窄脉冲，而一位则不发送脉冲。窄脉冲会在比特开始时发送，持续时间占比特时长的 1/OSR、2/OSR、3/OSR 或 4/OSR。当 SCI_CTRL[TXINV] 被清除时，零位会发送低电平窄脉冲；而当 SCI_CTRL[TXINV] 被置位时，零位则会发送高电平窄脉冲。

21.13.2. 红外接收解码器

红外接收模块将 RXD 信号转换为接收移位寄存器数据。每个接收到的零位应产生窄脉冲，而每个接收到的 1 位则不产生脉冲。当 SCI_STAT[RXINV] 被清除时，零位对应窄低脉冲；当 SCI_STAT[RXINV] 被置位时，零位则对应窄高脉冲。该接收解码器符合 IrDA 红外物理层规范中定义的边沿抖动要求。

21.14. 中断和状态标志

SCI 发送器包含两个状态标志位，可选择触发硬件中断请求。发送数据寄存器空闲（SCI_STAT[TDRE] ）表示发送数据缓冲区有空间可向 SCI_DATA 写入新字符。若发送中断使能（SCI_CTRL[TIE]）位被置位，则当 SCI_STAT[TDRE] 被触发时会请求硬件中断。发送完成（SCI_STAT[TC]）表示发送器已完成所有数据、前导码和停止字符的传输，处于 TXD 非激活状态的空闲模式。该标志位常用于调制解调器系统中，用于判断何时可安全关闭调制解调器。若发送完成中断使能（SCI_CTRL[TCIE]）位被置位，则当 SCI_STAT[TC] 被触发时会请求硬件中断。若对应的 SCI_CTRL[TIE] 或 SCI_CTRL[TCIE] 本地中断屏蔽位已被清除，也可通过软件轮询方式监控 SCI_STAT[TDRE] 和 SCI_STAT[TC] 状态标志位。

当程序检测到接收数据寄存器已满（SCI_STAT[RDRF] = 1）时，通过读取 SCI_DATA 从接收数据寄存器获取数据。通过读取 SCI_DATA 清除 SCI_STAT[RDRF] 标志。

IDLE 状态标志包含防止在 RXD 线路长时间处于空闲状态时重复设置该标志的逻辑。通过向 SCI_STAT[IDLE] 标志写入 1 可清除 IDLE。清除 SCI_STAT[IDLE] 后，该标志将保持清除状态，直至接收器至少收到一个新字符并设置了 SCI_STAT[RDRF]。

如果在接收到的字符中检测到相关错误，导致设置了 SCI_STAT[RDRF]，则错误标志 — 噪声标志（SCI_STAT[NF]）、帧错误标志（SCI_STAT[FE]）和奇偶校验错误标志（SCI_STAT[PF]） — 将与 SCI_STAT[RDRF] 同时被设置。这些标志不会在超频情况下被设置。

如果在准备将新字符从接收移位器传输到接收数据缓冲区时，SCI_STAT[RDRF] 已被设置，则会设置溢出（SCI_STAT[OR]）标志，而不会设置数据以及任何相关的 NF、FE 或 PF 条件。

如果接收到的字符与 MATCH[MA1] 和/或 MATCH[MA2] 的内容相匹配，则 SCI_STAT[MA1F] 和/或 SCI_STAT[MA2F] 标志将同时被设置，同时 SCI_STAT[RDRF] 也被设置。

RXD 串行数据输入引脚上的有效边沿可随时触发 SCI_STAT[RXEDGIF] 标志的设置。通过向该标志写入 1 可清除 SCI_STAT[RXEDGIF] 标志。此功能取决于接收器是否处于启用状态（SCI_CTRL[RE] = 1）。

22. 同步串行接口（SSI）

22.1. 介绍

SSI 是一种可编程的同步串行接口（SSI）外围设备。这是一个符合 AMBA 2.0 标准的 AHB（高级高性能总线）组件。主机处理器通过 AHB 接口访问 SSI 上的数据、控制和状态信息。SSI 还可以通过一组可选的 DMA 信号与 DMA 控制器进行交互，这些信号可以在配置时进行选择。

22.2. 特性

SSI 模块特性包括：

- 串行主操作。
- DMA 控制器接口-使 SSI 能够通过总线与 DMA 控制器进行交互，使用握手接口来传输请求。
- 增强的 SPI 传输中的时钟拉伸支持。
- 数据项大小（4 到 32 位） — 每个数据传输项的大小由程序员控制。
- FIFO 深度 — 发送和接收 FIFO 缓冲器的深度为 8 个字，FIFO 宽度固定为 32 位。
- 改进的 SPI 支持。
- 支持就地执行（XIP）模式。

22.3. 操作模式

SSI 在以下三种模式下运行：

1. 运行模式 — 运行模式是正常工作模式。
2. Doze 模式 — Doze 模式是一种可配置的低功耗模式。
3. 停止模式-在停止模式下，SSI 处于非活动状态。

22.4.方块图

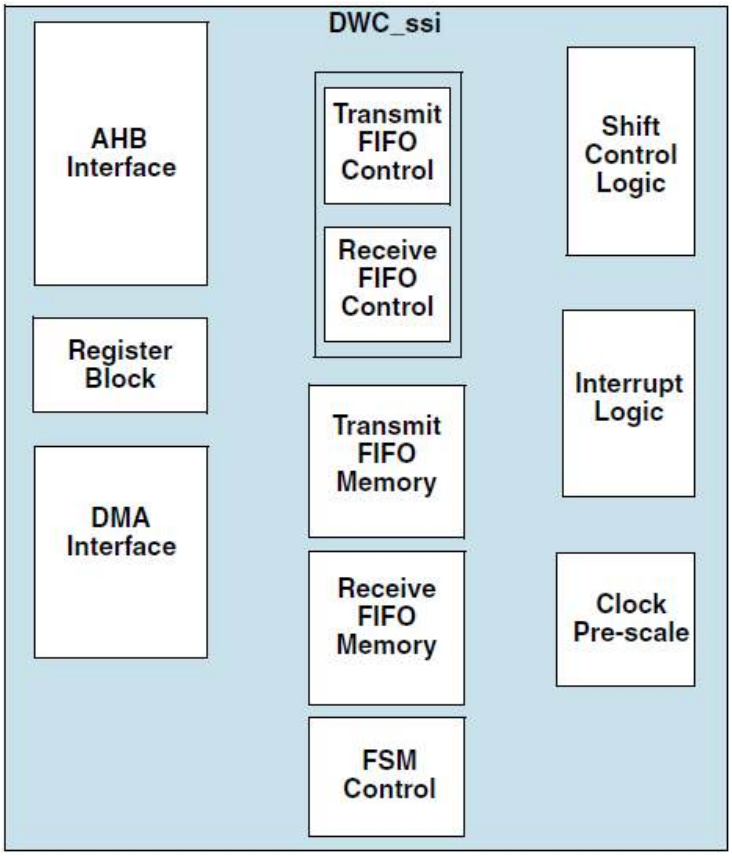


图 22-1：SSI 方块图

22.5. 模块内存映射

SSI 寄存器内存映射如下所示。

表 22-1：SSI 内存映射

地址偏移量	Bit[31:0]	访问权限
0x0000	控制寄存器 0（CTRLR0）	S/U
0x0004	控制寄存器 1（CTRLR1）	S/U
0x0008	SSI 启用寄存器（SSIENR）	S/U
0x0010	从属选择寄存器（SER）	S/U
0x000C	微线控制寄存器（MWCR）	S/U
0x0014	波特率选择寄存器（BAUDR）	S/U
0x0018	发送信道阈值寄存器（TXFTLR）	S/U
0x001C	接收 FIFO 阈值寄存器（RXFTLR）	S/U
0x0020	传输信标信号电平寄存器（TXFLR）	S/U
0x0024	接收 FIFO 级寄存器（RXFLR）	S/U
0x0028	状态寄存器（SR）	S/U
0x002C	中断模式寄存器（IMR）	S/U
0x0030	中断状态寄存器（ISR）	S/U
0x0034	原始中断状态寄存器（RISR）	S/U
0x0038	发送 FIFO 溢出中断清除寄存器（TXOICR）	S/U
0x003C	接收 FIFO 溢出中断清除寄存器（RXOICR）	S/U
0x0040	接收 FIFO 下溢中断清除寄存器（RXUICR）	S/U
0x0048	中断清除寄存器（ICR）	S/U
0x004C	DMA 控制寄存器（DMACR）	S/U
0x0050	DMA 传输数据电平寄存器（DMATDLR）	S/U
0x0054	DMA 接收数据级别寄存器（DMARDLR）	S/U
0x0058	识别寄存器（IDR）	S/U
0x005C	版本 ID 寄存器（VIDR）	S/U
0x0060+i*0x4	SSI 数据登记（DRx）	S/U
0x00F0	RX 样本延迟寄存器（RXSDR）	S/U
0x00F4	SPI 控制寄存器 0（SPICTRLR0）	S/U
000FC	XIP 模式位（XIPMBR）	S/U
0x0100	XIP Incr Inst Register（XIPIIR）	S/U

地址偏移量	Bit[31:0]	访问权限
0x0104	XIP Wrap Inst Register（XIPWIR）	S/U
0x0108	XIP 控制寄存器（XIPCR）	S/U
0x010C	XIP 从机使能寄存器（XIPSER）	S/U
0x0110	XIP 接收 FIFO 溢出中断清除寄存器（XRXIOCR）	S/U
0x0114	XIP 继续传输超时寄存器（XIPCTTOR）	S/U

22.6. 寄存器说明

22.6.1. 控制寄存器 0（CTRLR0）

偏移地址：0x0000

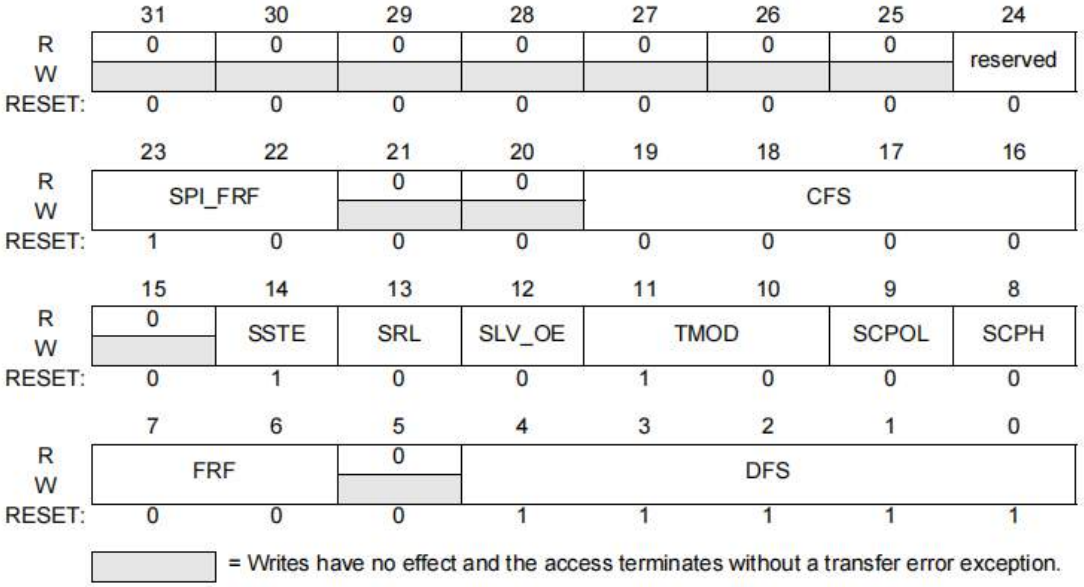


图 22-2：控制寄存器 0（CTRLR0）

此寄存器控制串行数据传输。当 SSI 被启用时，无法向该寄存器写入数据。通过向 SSIENR 写入数据可启用和禁用 SSI。

SPI_FRF — SPI 帧格式

选择用于发送/接收数据的数据帧格式。仅当 SSIC_SPI_MODE 设置为“Dual”或“Quad”或“Octal”模式时有效。

0x0（SPIStandard）：标准 SPI 格式

0x1（SPI_DUAL）：双 SPI 格式

0x2（SPI_QUAD）：四通道 SPI 格式

0x3（SPI_OCTAL）：八位 SPI 格式

CFS — 控制帧大小

选择微线帧格式的控制字长度。

0x0（SIZE_01_BIT）：01 位控制 Word

0x1（SIZE_02_BIT）：02 位控制 Word

0x2（SIZE_03_BIT）：03 位控制 Word

0x3（SIZE_04_BIT）：04 位控制 Word

0x4（SIZE_05_BIT）：05 位控制 Word

0x5（SIZE_06_BIT）：06 位控制 Word

0x6（SIZE_07_BIT）：07 位控制 Word

0x7（SIZE_08_BIT）：08 位控制 Word

0x8（SIZE_09_BIT）：09 位控制 Word

0x9（SIZE_10_BIT）：10 位控制 Word

0xA（SIZE_11_BIT）：11 位控制 Word

0xB（SIZE_12_BIT）：12 位控制 Word

0xC（SIZE_13_BIT）：13 位控制 Word

0xD（SIZE_14_BIT）：14 位控制 Word

0xE（SIZE_15_BIT）：15 位控制 Word

0xF（SIZE_16_BIT）：16 位控制 Word

SSTE — 从属选择切换启用。

在 SPI 模式下运行且时钟相位（SCPH）设置为 0 时，此寄存器控制数据帧之间的从设备选择线（ss_*_n）的行为。

1 = （TOGGLE_EN）： ss_*_n 线将在连续数据帧之间切换，同时保持串行时钟（SCLK）为默认值，且 ss_*_n 为高电平

0 = （TOGGLE_DISABLE）：ss_*_n 将保持低电平，且 SCLK 将在传输期间持续运行

SRL — 移位寄存器环路。

本模块仅供测试用途。当处于内部激活状态时，可将发送移位寄存器输出端与接收移位寄存器输入端连接。支持串行从机和串行主控两种工作模式。若将 SSI 配置为环回模式下的从机，则 ss_in_n 和 ssi_clk 信号必须由外部源提供。在此模式下，由于缺乏需要环回的信号源，从机会无法生成这些信号。

1 = （TESTING_MODE）：测试模式操作

0 = （NORMAL_MODE）：正常模式运行

SLV_OE — 从属输出使能。

仅当 SSI 被配置为串行从设备时才有意义。如果将其配置为串行主设备，则此位字段不起作用。

1 = （禁用 D）：从属输出被禁用

0 = （已启用）：从属输出已启用

TMOD — 传输模式。

选择串行通信的传输模式。此字段不影响传输重复性，仅指示接收或发送的数据是否有效。

0x0（TX_AND_RX）：发送和接收；不适用于增强型 SPI 工作模式

0x1（TX_ONLY）：仅发送模式；或在增强型 SPI 操作模式下写入

0x2（RX_ONLY）：仅接收模式；或在增强型 SPI 操作模式下读取

0x3（EEPROM_READ）：EEPROM 读取模式；不适用于增强型 SPI 工作模式

SCOPL — 串行时钟极性。

当帧格式（FRF）设置为 Motorola SPI 时有效。用于选择非活动串行时钟的极性，当 SSI 主机没有在串行总线上主动传输数据时，该时钟将保持非活动状态。

1 = （INACTIVE_LOW）：串行时钟处于非活动状态，高电平

0 = （INACTIVE_HIGH）：串行时钟处于低电平的非活动状态

SCPH — 串行时钟相位。

当帧格式（FRF）设置为 Motorola SPI 时有效。串行时钟相位选择串行时钟与从属选择信号之间的关系。

当 SCPH = 0 时，数据在串行时钟的第一个边沿被捕获。当 SCPH = 1 时，串行时钟在从选线激活后一个周期开始切换，数据在串行时钟的第二个边沿被捕获。

1 = （START_BIT）：第一个比特开始时，串行时钟切换

0 = （中间位）：第一个比特位中间的串行时钟切换

FRF — 帧格式。

选择要传输数据的串行协议。

0x0（SPI）：摩托罗拉 SPI 帧格式

0x1（SSP）：德州仪器 SSP 帧格式

0x2（微线）：National Semiconductors 微线框架格式

0x3（保留）：保留

DFS — 数据帧大小。

选择数据帧长度。当数据帧大小被设定为小于 32 位时，接收逻辑会自动对收据数据进行右对齐，并且收据 FIFO 的上位用零填充。

在写入发送 FIFO 之前，必须正确地对发送数据进行加权。发送逻辑在发送数据时忽略上层未使用的位。

0x0（DFS_01_BIT）：保留
0x1（DFS_02_BIT）：保留
0x2（DFS_03_BIT）：保留
0x3（DFS_04_BIT）：04 位串行数据传输
0x4（DFS_05_BIT）：05 位串行数据传输
0x5（DFS_06_BIT）：06 位串行数据传输
0x6（DFS_07_BIT）：07 位串行数据传输
0x7（DFS_08_BIT）：08 位串行数据传输
0x8（DFS_09_BIT）：09 位串行数据传输
0x9（DFS_10_BIT）：10 位串行数据传输
0xA（DFS_11_BIT）：11 位串行数据传输
0xB（DFS_12_BIT）：12 位串行数据传输
0xC（DFS_13_BIT）：13 位串行数据传输
0xD（DFS_14_BIT）：14 位串行数据传输
0xE（DFS_15_BIT）：15 位串行数据传输
0xF（DFS_16_BIT）：16 位串行数据传输
0x10（DFS_17_BIT）：17 位串行数据传输
0x11（DFS_18_BIT）：18 位串行数据传输
0x12（DFS_19_BIT）：19 位串行数据传输
0x13（DFS_20_BIT）：20 位串行数据传输
0x14（DFS_21_BIT）：21 位串行数据传输
0x15（DFS_22_BIT）：22 位串行数据传输
0x16（DFS_23_BIT）：23 位串行数据传输
0x17（DFS_24_BIT）：24 位串行数据传输
0x18（DFS_25_BIT）：25 位串行数据传输
0x19（DFS_26_BIT）：26 位串行数据传输
0x1A（DFS_27_BIT）：27 位串行数据传输
0x1B（DFS_28_BIT）：28 位串行数据传输
0x1C（DFS_29_BIT）：29 位串行数据传输

0x1D（DFS_30_BIT）：30 位串行数据传输

0x1E（DFS_31_BIT）：31 位串行数据传输

0x1F（DFS_32_BIT）：32 位串行数据传输

22.6.2. 控制寄存器 1（CTRLR1）

该寄存器仅在 SSI 被配置为主设备时存在。当 SSI 被配置为串行从设备时，对该位置进行写入操作将无效；读取该位置返回 0。控制寄存器 1 在 RX-ONLY 模式和 TX-ONLY 模式下控制串行传输的结束。当 SSI 处于启用状态时，无法对该寄存器进行写入操作。通过向 SSIENR 寄存器写入数据即可实现 SSI 的启用与禁用功能。

偏移地址：0x0004

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	NDF							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	NDF							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-3：控制寄存器 1（CTRLR1）

NDF — 数据帧数。

当 TMOD = 01 或 TMOD = 10 或 TMOD = 11 时，此寄存器字段设置 SSI 连续接收或发送的数据帧数。SSI 会持续接收或发送串行数据，直到数据帧数等于此寄存器值加 1。

当 SSI 被配置为串行从机时，该寄存器将不起作用且不存在。当 TMOD = 01 时，必须设置 SPI_CTRLR0 中的 CLK_STRETCH_EN。

22.6.3. SSI 启用寄存器（SSIENR）

偏移地址：0x0008

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	SSIC_EN
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-4：SSI 使能寄存器（SSIENR）

SSIC_EN — SSI 启用。

启用和禁用所有 SSI 操作。当禁用时，所有串行传输将立即停止。设备禁用时，发送和接收 FIFO 缓冲区将被清空。启用状态下无法对部分 SSI 控制寄存器进行编程。禁用时，系统会延迟设置 SSI 休眠输出，以通知可安全移除 ssiClock 信号，从而节省系统功耗。

1 = （已启用）：启用 SSI

0 = （禁用）：禁用 SSI

22.6.4. 微线控制寄存器（MWCR）

偏移地址：0x000C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	MHS	MDD	MWMOD
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-5：微导线控制寄存器（MWCR）

该寄存器控制半双工微线串行协议的数据字方向。当 SSI 被启用时，无法对该寄存器进行写入操作。通过向 SSIENR 寄存器写入数据可启用和禁用 SSI。

MHS — 微导丝手柄。

仅当 SSI 被配置为串行主设备时才相关。当配置为串行从设备时，此位字段没有功能。用于启用和禁用 Microwire 协议的忙/就绪握手接口。当启用时，SSI 会在传输最后一个数据/控制位后，从目标从设备检查就绪状态，然后清除 SR 寄存器中的 BUSY 状态。

1 = （启用）：握手接口已启用

0 = （禁用）：握手接口被禁用

MDD — 微导丝控制。

定义使用微线串行协议时数据字的方向。当此位设置为 0 时，SSI 宏单元从外部串行设备接收数据字。当此位设置为 1 时，SSI 宏单元将数据字发送到外部串行设备。

1 = （传输）：SSI 传输数据

0 = （接收）：SSI 接收数据

MWMOD — 微导丝传输模式。

定义微线传输是顺序还是非顺序。当使用顺序模式时，只需要一个控制字就可以发送或接收一组数据字。当使用非顺序模式时，必须为每个发送或接收的数据字提供一个控制字。

1 = （顺序）：顺序转移

0 = （非顺序）：非顺序传输

22.6.5. 从属启用寄存器（SER）

偏移地址：0x0010

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	SER
W								
RESET:	0	0	0	0	0	0	0	1

= Writes have no effect and the access terminates without a transfer error exception.

图 22-6：从属使能寄存器（SER）

该寄存器仅在 SSI 被配置为主设备时有效。当 SSI 被配置为串行从设备时，对该位置写入数据无效；读取该位置返回0。该寄存器用于启用SSI主设备的各个从设备选择输出引脚。SSI主设备最多可提供16个从设备选择输出引脚。当SSI处于忙状态或SSSIC_EN = 1时，无法对该寄存器进行写入操作。

SER - 从属设备选择启用标志。

1 = 已选中

0 = 未选择

22.6.6. 波特率选择（BAUDR）

偏移地址：0x0014

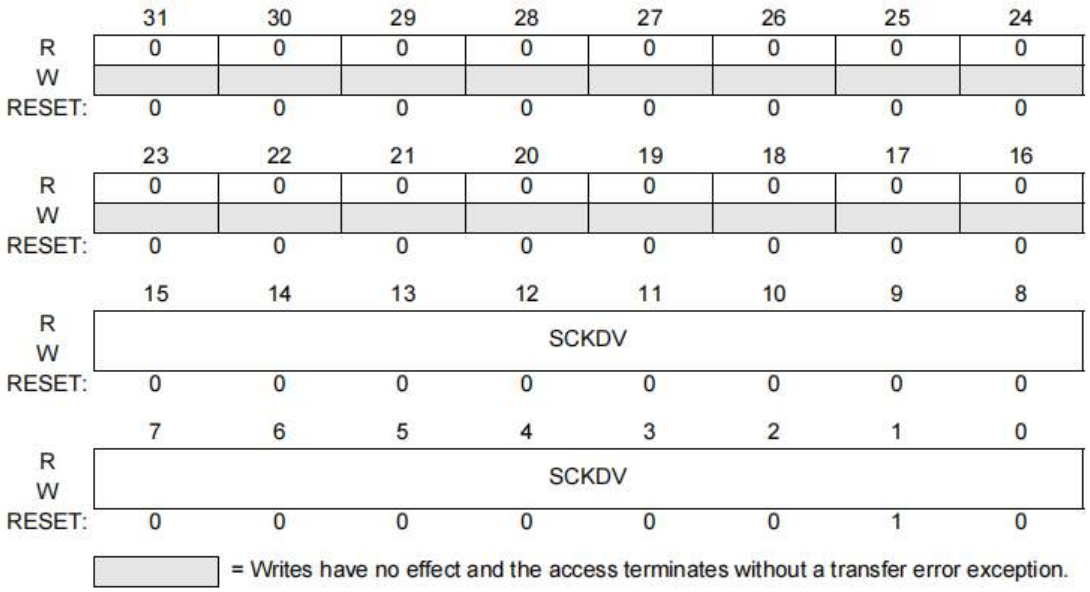


图 22-7：波特率选择（BAUDR）

SCKDV-SSI 时钟分频器。

该字段的 LSB 始终被设置为 0，并且不受写入操作的影响，这确保了寄存器中保持偶数值。如果值为 0，则串行输出时钟（sclk_out）将被禁用。sclk_out 的频率由以下方程推导得出：

$$Fsclk\_out = FssiClock/SCKDV$$

其中，SCKDV 是 2 到 65534 之间的任意偶数值。例如：对于 FssiClock = 3.6864MHz 和 SCKDV = 2，FsclkOut = 3.6864/2 = 1.8432MHz

22.6.7. 发送信道阈值（TXFTLR）

偏移地址：0x0018

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	TXFTHR				
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	TFT				
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-8：发送 FIFO 阈值（TXFTLR）

该寄存器控制发送 FIFO 存储器的阈值。通过向 SSIENR 寄存器写入数据，可启用或禁用 SSI。

TXFTHR — 传输起始 FIFO 电平。

用于控制发送 FIFO 中的条目水平，超过该水平后将在串行线上开始传输。此寄存器可用于确保在对串行线进行写入操作之前，发送 FIFO 中存在足够的数据。这些字段仅在主控模式下有效。

TFT — 传输 FIFO 阈值。

该寄存器用于控制发送 FIFO 控制器触发中断的条目数量（或更低阈值）。FIFO 深度可在 8 至 256 之间配置，其容量由访问 FIFO 所需的地址位数决定。若尝试设置的数值大于或等于 FIFO 实际深度，则该字段将保持当前值且不进行写入。当发送 FIFO 条目数量小于或等于此阈值时，系统会触发发送 FIFO 空闲中断。

22.6.8. 接收 FIFO 阈值（RXFTLR）

偏移地址：0x001C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	RFT				
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-9：接收 FIFO 阈值（RXFTLR）

该寄存器控制接收 FIFO 存储器的阈值。通过向 SSIENR 寄存器写入数据，可启用和禁用 SSI。

RFT — 接收 FIFO 阈值。

该寄存器用于控制接收 FIFO 控制器触发中断的条目数量（或更高阈值）。寄存器容量根据访问 FIFO 所需的地址位数确定。若尝试设置的数值超过 FIFO 深度，该字段将保持当前值且不进行写入。当接收 FIFO 条目数量达到或超过此阈值+1 时，系统会触发接收 FIFO 满载中断。

22.6.9. 传输信标信号电平寄存器（TXFLR）

偏移地址：0x0020

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	TXTFL					
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-10：发送 FIFO 电平寄存器（TXFLR）

TXFLR — 传输 FIFO 水平。

包含发送 FIFO 中的有效数据条目数。

22.6.10. 接收 FIFO 级寄存器（RXFLR）

偏移地址：0x0024

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	RXTFL					
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-11：接收 FIFO 电平寄存器（RXFLR）

RXTFL — 接收 FIFO 级别。

包含接收 FIFO 中的有效数据条目数。

22.6.11. 状态寄存器（SR）

偏移地址：0x0028

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	reserved	TXE	RFF	RFNE	TFE	TFNF	BUSY
W								
RESET:	0	0	0	0	0	1	1	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-12：状态寄存器（SR）

TXE — 传输错误。

设置为传输开始时发送 FIFO 为空。此位只能在 SSI 被配置为从设备时设置。在 TXD 线上重新发送来自上一次传输的数据。读取时清除此位。

1 = （TX_ERROR）：传输错误

0 = （NO_ERROR）：没有错误

RFF — 接收 FIFO 满。

当接收 FIFO 完全满时，该位被设置。当接收 FIFO 包含一个或多个空位置时，该位被清除。

1 = （FULL）：接收 FIFO 已满

0 = （未满）：接收 FIFO 未满

RFNE-接收 FIFO 未空。

当接收 FIFO 包含一个或多个条目时设置，接收 FIFO 为空时清除。软件可通过轮询此位来完全清空接收 FIFO。

1 = （NOT_EMPTY）：接收 FIFO 不为空

0 = （空）：接收 FIFO 为空

TFE — 传输 FIFO 为空。

当发送 FIFO 完全为空时，该位被设置。如果发送 FIFO 包含一个或多个有效条目，则清除该位。此位字段不请求中断。

1 = （EMPTY）：传输 FIFO 为空

0 = （NOT_EMPTY）：发送 FIFO 不为空

TFNF — 传输 FIFO 未满。

当发送 FIFO 包含一个或多个空位置时设置，当 FIFO 满时清除。

1 = （未满）：Tx FIFO 未满

0 = Tx FIFO 已满

BUSY — SSI 工作标志。

设置时，表示串行传输正在进行；清除时，表示 SSI 处于空闲或禁用状态。

1 = （激活）：SSI 正在主动传输数据

0 = （非活动）：SSI 处于闲置或禁用状态

22.6.12. 中断模式寄存器（IMR）

偏移地址：0x002C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	XRXOIM	reserved	RXFIM	RXOIM	RXUIM	TXOIM	TXEIM
W								
RESET:	0	1	1	1	1	1	1	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-13：中断掩码寄存器（IMR）

XRXOIM — XIP 接收 FIFO 溢出中断掩码

1 = （未掩蔽）：未掩蔽 ssi_xrxo Intr 中断

0 = （掩蔽）：ssi_xrxo Intr 中断被掩蔽

RXFIM — 接收 FIFO 满中断掩码

1 = （未掩蔽）：未掩蔽 ssi_rxf_intr 中断

0 = （掩蔽）：ssi_rxf_intr 中断被掩蔽

RXOIM — 接收 FIFO 溢出中断掩码

1 = （未掩蔽）：未掩蔽 ssi_rxo_intr 中断

0 = （掩蔽）：ssi_rxo_intr 中断被掩蔽

RXUIM — 接收 FIFO 下溢中断掩码

1 = （未屏蔽）：未屏蔽 ssi_rxu_intr 中断

0 = （掩蔽）：ssi_rxu_intr 中断被掩蔽

TXOIM — 发送 FIFO 溢出中断掩码

1 = （未屏蔽）：未屏蔽 ssi_txo_intr 中断

0 = （掩蔽）：ssi_txo_intr 中断被掩蔽

TXEIM — 发送 FIFO 空中断掩码

1 = （未掩蔽）：未掩蔽 ssi_txe_intr 中断

0 = （掩蔽）：sii_txe_intr 中断被掩蔽

22.6.13. 中断状态寄存器（ISR）

偏移地址：0x0030

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	XRXOIS	reserved	RXFIS	RXOIS	RXUIS	TXOIS	TXEIS
W								
RESET:	0	0	0	0	0	0	0	1

= Writes have no effect and the access terminates without a transfer error exception.

图 22-14：中断状态寄存器（ISR）

此寄存器报告被屏蔽后的 SSI 中断的状态。

XRXOIS-XIP 接收 FIFO 溢出中断状态

1 = （激活）：掩蔽后 ssi_xrxo_intr 中断处于激活状态

0 = （非活动）：掩蔽后 ssi_xrxo_intr 中断未激活

RXFIS — 接收 FIFO 满中断状态

1 = （激活）：掩码后 ssi_rxf_intr 中断处于激活状态

0 = （非活动）：掩蔽后 ssi_rxf_intr 中断未激活

RXOIS — 接收 FIFO 溢出中断状态

1 = （激活）：掩码后 ssi_rxo_intr 中断处于激活状态

0 = （非活动）：掩蔽后 ssi_rxo_intr 中断未激活

RXUIS — 接收 FIFO 下溢中断状态

1 = （激活）：掩码后 ssi_rxu_intr 中断处于激活状态

0 = （非活动）：掩蔽后 ssi_rxu_intr 中断未激活

TXOIS — 传输 FIFO 溢出中断状态

1 = （激活）：掩码后 ssi_txo Intr 中断处于激活状态

0 = （非活动）：掩码后 ssi_txo_intr 中断未激活

TXEIS — 传输 FIFO 空闲中断状态

1 = （激活）：掩蔽后 ssi_txe_intr 中断处于激活状态

0 = （非活动）：掩蔽后 ssi_txe_intr 中断未激活

22.6.14. 原始中断状态寄存器（RISR）

偏移地址：0x0034

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	XRXOIR	reserved	RXFIR	RXOIR	RXUIR	TXOIR	TXEIR
W								
RESET:	0	0	0	0	0	0	0	1

= Writes have no effect and the access terminates without a transfer error exception.

图 22-15：原始中断状态寄存器（RISR）

XRXOIR-XIP 接收 FIFO 溢出原始中断状态

1 = （激活）：掩蔽前 ssi_xrxo_intr 中断处于激活状态

0 = （非活动）：掩蔽前 ssi_xrxo_intr 中断未激活

RXFIR — 接收 FIFO 满原始中断状态

1 = （激活）：掩蔽前 ssi_rxf_intr 中断处于激活状态

0 = （非活动）：掩蔽前 ssi_rxf_intr 中断未激活

RXOIR — 接收 FIFO 溢出原始中断状态

1 = （激活）：掩蔽前 ssi_rxo_intr 中断处于激活状态

0 = （非活动）：掩蔽前 ssi_rxo_intr 中断未激活

RXUIR — 接收 FIFO 下溢原始中断状态

1 = （激活）：掩蔽前 ssi_rxu_intr 中断处于激活状态

0 = （非活动）：掩蔽前 ssi_rxu_intr 中断未激活

TXOIS — 传输 FIFO 溢出原始中断状态

1 = （激活）：掩蔽前 ssi_txo Intr 中断处于激活状态

0 = （非活动）：掩蔽前 ssi_txo_intr 中断未激活

TXEIS — 传输 FIFO 空闲原始中断状态

1 = （激活）：掩蔽前 ssi_txe_intr 中断处于激活状态

0 = （非活动）：掩蔽前 ssi_txe_intr 中断未激活

22.6.15. 传输 FIFO 溢出中断清除寄存器（TXOICR）

偏移地址：0x0038

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	TXOICR
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-16：发送 FIFO 溢出中断清除寄存器（TXOICR）

TXOICR — 清除发送 FIFO 溢出中断。

此寄存器反映中断的状态。从该寄存器读取会清除 ssi_txo_intr 中断，写入则没有影响。

22.6.16. 接收 FIFO 溢出中断清除寄存器（RXOICR）

偏移地址：0x003C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	RXOICR
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-17：接收 FIFO 溢出中断清除寄存器（RXOICR）

RXOICR — 清除接收 FIFO 溢出中断。

此寄存器反映中断的状态。从该寄存器读取会清除 ssi_rxo_intr 中断，写入则没有影响。

22.6.17. 接收 FIFO 下溢中断清除寄存器（RXUICR）

偏移地址：0x0040

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	RXUICR
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-18：接收 FIFO 下溢中断清除寄存器（RXUICR）

RXUICR-清除接收 FIFO 下溢中断。

此寄存器反映中断的状态。从该寄存器读取会清除 ssi_rxu_intr 中断；写入则没有效果。

22.6.18. 中断清除寄存器（ICR）

偏移地址：0x0048

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	ICR
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-19：中断清除寄存器（ICR）

ICR — 清除中断。

如果以下任一中断处于激活状态，则设置此寄存器。读取清除了 ssi_txo_intr、ssi_rxu_intr、ssi_rxo_intr 和 ssi_mst_intr 中断。对这个寄存器写入没有任何效果。

22.6.19. DMA 控制寄存器（DMACR）

偏移地址：0x004C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	TDMAE	RDMAE
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-20：DMA 控制寄存器（DMACR）

该寄存器仅在 SSI 配置为使用一组 DMA 控制器接口信号时有效（SSIC_HAS_DMA = 1）。当 SSI 未配置为 DMA 操作时，该寄存器将不存在，对寄存器地址的写入将没有效果；从该寄存器地址读取将返回零。该寄存器用于启用 DMA 控制器接口操作。

TDMAE — 传输 DMA 使能

此位启用/禁用发送 FIFO DMA 通道。

1 = （已启用）：传输 DMA 已启用

0 = （禁用）：传输 DMA 被禁用

RDMAE — 接收 DMA 使能

此位启用/禁用接收 FIFO DMA 通道。

1 = （已启用）：接收 DMA 已启用

0 = （禁用）：接收 DMA 被禁用

22.6.20. DMA 传输数据电平（DMATDLR）

偏移地址：0x0050

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	DMATDL				
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-21：DMA 发送数据电平（DMATDLR）

该寄存器仅在为 SSI 配置一组 DMA 接口信号时有效（SSIC_HAS_DMA = 1）。如果未将 SSI 配置为 DMA 操作，则此寄存器将不存在，对地址的写入操作将不会产生任何影响；从该地址读取将返回零。

DMATDL — 传输数据层。

此位字段控制发送逻辑发出 DMA 请求的级别。它等于水印级别，即当发送 FIFO 中的有效数据条目数等于或小于此字段值且 TDMAE = 1 时，dma_tx_req 信号将被生成。

22.6.21. DMA 接收数据级别（DMARDLR）

偏移地址：0x0054

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	DMARDL				
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-22：DMA 接收数据电平（DMARDLR）

该寄存器仅在为 SSI 配置一组 DMA 接口信号时有效（SSICHASDMA = 1）。如果未将 SSI 配置为 DMA 操作，则此寄存器将不存在，对地址的写入操作将不会产生任何影响；从该地址读取将返回零。

DMARDL — 接收数据级别。

此位字段控制接收逻辑发出 DMA 请求的级别。水印级别 = DMARDL+1；也就是说，当接收 FIFO 中的有效数据条目数等于或大于此字段值+1 且 RDMAE = 1 时，将生成 dma_rxREQ。

22.6.22. 识别寄存器（IDR）

偏移地址：0x0058

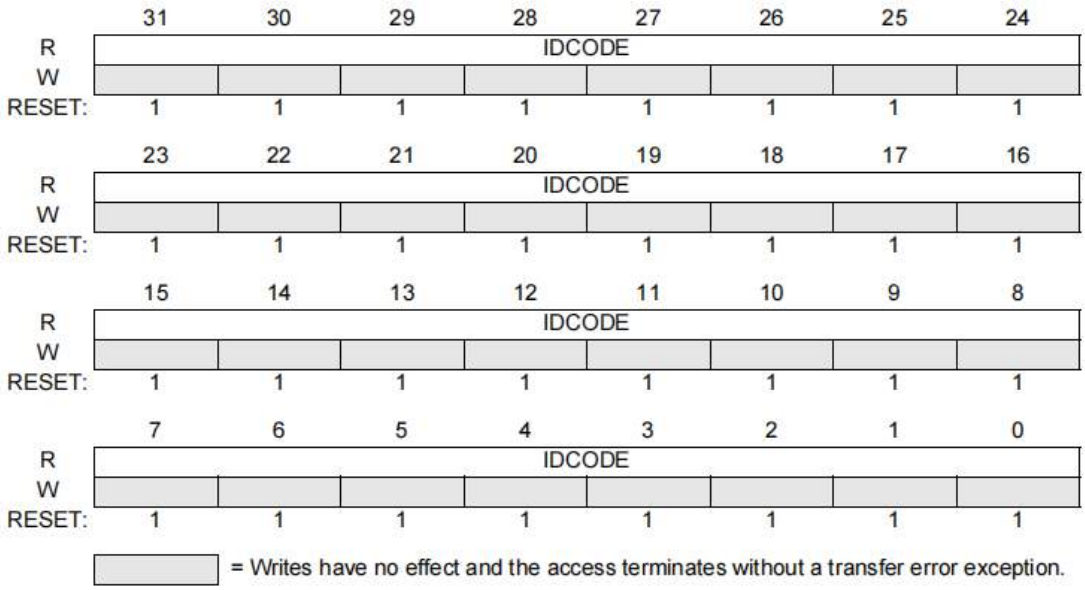


图 22-23：识别寄存器（IDR）

IDCODE — 识别代码。

寄存器包含外围设备的识别码，该识别码在配置时使用 Core Consultant 写入寄存器。

22.6.23. 版本 ID 寄存器（VIDR）

偏移地址：0x005C

	31	30	29	28	27	26	25	24
R	SSIC_COMP_VERSION							
W								
RESET:	0	0	1	1	0	0	0	1

	23	22	21	20	19	18	17	16
R	SSIC_COMP_VERSION							
W								
RESET:	0	0	1	1	0	0	0	0

	15	14	13	12	11	10	9	8
R	SSIC_COMP_VERSION							
W								
RESET:	0	0	1	1	0	0	0	1

	7	6	5	4	3	2	1	0
R	SSIC_COMP_VERSION							
W								
RESET:	0	0	1	0	1	0	1	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-24：版本 ID 寄存器（VIDR）

SSICCOMP_VERSION

包含 Synopsys 组件版本的十六进制表示形式。由版本中每个数字的 ASCII 值和*组成。例如，32_30_31_2A 表示版本 2.01*。

22.6.24. SSI 数据登记（DRx）

偏移地址：0x0060

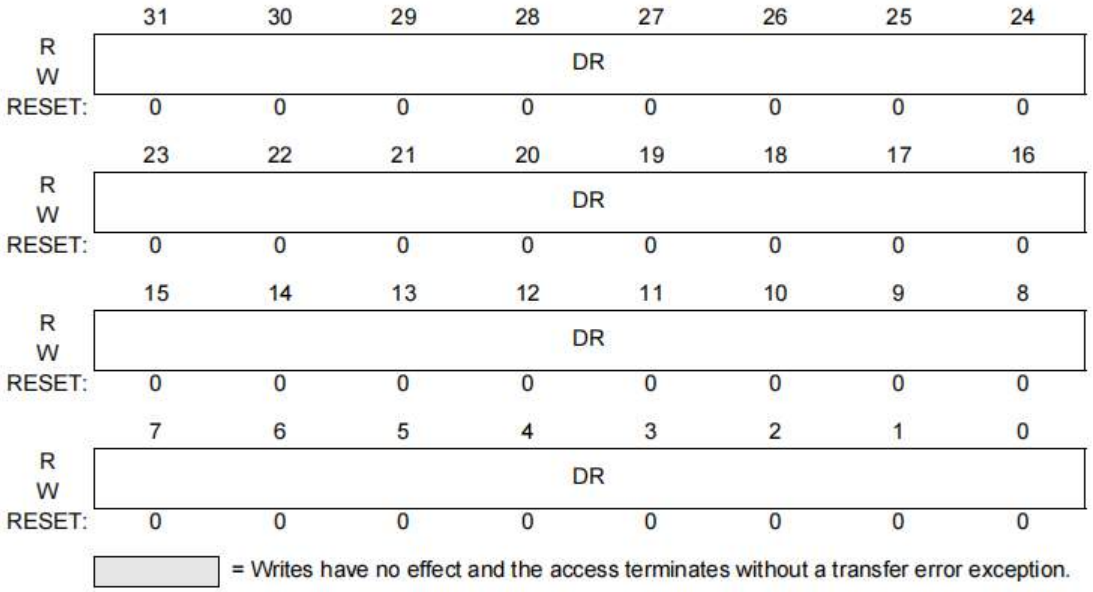


图 22-25：SSI 数据寄存器（DRx）

SSI 数据寄存器是用于发送/接收 FIFO 的 32 位读写缓冲器。当寄存器被读取时，会访问接收 FIFO 缓冲器中的数据。当寄存器被写入时，数据会被移入发送 FIFO 缓冲器；只有当 SSIC_EN = 1 时才能进行写入操作。当 SSIC_EN = 0 时，FIFO 将被复位。

DR — 数据寄存器。

向该寄存器写入数据时，必须对数据进行右对齐。读取的数据自动右对齐。

Read = 接收 FIFO 缓冲器

写入 = 传输 FIFO 缓冲器。

22.6.25. RX 样本延迟寄存器（RXSDR）

偏移地址：0x00F0

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	SE
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	RSD							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-26：RX 样本延迟寄存器（RXSDR）

SE-接收数据（RXD）采样边沿。

1 = ssiClock 的负边将用于对输入数据进行采样

0 = ssiClock 的起始边沿将用于对输入数据进行采样

RSD — 接收数据（RXD）采样延迟。

该寄存器用于延迟 RXD 输入端口的采样。每个值代表 RXD 采样上的单个 ssiClock 延迟。

22.6.26. SPI 控制寄存器 0（SPICTRLR0）

偏移地址：0x00F4

	31	30	29	28	27	26	25	24
R	0	CLK_STR ETCH_EN	reserved	0	reserved		reserved	reserved
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	reserved	reserved	reserved	reserved	reserved	reserved
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	WAIT_CYCLES					0	INST_L	
W								
RESET:	0	0	0	0	0	0	1	0

	7	6	5	4	3	2	1	0
R	reserved	0	ADDR_L				TRANS_TYPE	
W								
RESET:	0	0	0	1	1	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-27：SPI 控制寄存器 0（SPICTRLR0）

CLKSTRETCH_EN

在 SPI 传输中启用时钟拉伸功能。

在写入时，如果 FIFO 变为空，则 SSI 将延长时钟，直到 FIFO 有足够的数据继续传输。在读取时，如果接收 FIFO 已满，则 SSI 将停止时钟，直到从 FIFO 中读取数据为止。

提示：建议始终设置此位

1 = CLK_STRETCH_ENABLE

0 = CLK_STRETCH DISABLE

等待周期 — 双通道/四通道/八通道模式下，控制帧发送与数据接收之间的等待周期。以 SPI 时钟周期数表示。

INST_L — 双模式/四模式/八模式指令长度，单位为比特。

0x0（INST_L0）：无指令

0x1（INST_L4）：4 位指令长度

0x2（INST_L8）：8 位指令长度

0x3（INST_L16）：16 位指令长度

ADDR_L — 要传输的地址长度

0x0（ADDR_L0）：无地址

0x1（ADDR_L4）：4 位地址长度

0x2（ADDR_L8）：8 位地址长度

0x3（ADDR_L12）：12 位地址长度

0x4（ADDR_L16）：16 位地址长度

0x5（ADDR_L20）：20 位地址长度

0x6（ADDR_L24）：24 位地址长度

0x7（ADDR_L28）：28 位地址长度

0x8（ADDR_L32）：32 位地址长度

0x9（ADDR_L36）：36 位地址长度

0xA（ADDR_L40）：40 位地址长度

0xB（ADDR_L44）：44 位地址长度

0xC（ADDR_L48）：48 位地址长度

0xD（ADDR_L52）：52 位地址长度

0xE（ADDR_L56）：56 位地址长度

0xF（ADDR_L60）：60 位地址长度

TRANS_TYPE — 地址和指令传输格式。

选择 SSI 是在标准 SPI 模式还是在 CTRLR0 中选定的 SPI 模式下发送指令/地址。SPI_FRF 字段。

0x0（TT0）：指令和地址将通过标准 SPI 模式发送。

0x1（TT1）：指令将通过标准 SPI 模式发送，地址将通过 CTRLR0 指定的模式发送。SPI_FRF。

0x2（TT2）：指令和地址将按照 SPI_FRF 指定的模式发送。

0x3（TT3）：保留。

22.6.27. XIP 模式位（XIPMBR）

偏移地址：0x00FC

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	XIP_MD_BITS							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	XIP_MD_BITS							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-28：XIP 模式位（XIPMBR）

该寄存器承载在地址阶段之后通过 XIP 操作模式发送的模式位。这是一个 8 位寄存器，只有当 SSIENR 寄存器设置为 0 时才能写入。

XIP MDI

XIP 传输地址阶段后要发送的模式位。

22.6.28. XIP Incr Inst Register（XIPIIR）

偏移地址：0x0100

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	INCR_INST							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	INCR_INST							
W								
RESET:	0	1	1	0	1	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

图 22-29：XIP Incr Inst Register（XIPIIR）

该寄存器仅在 SSIC_XIP_INST_EN 等于 1 时有效。该寄存器用于存储当在 AHB 接口上请求相同操作时，要在 INCR 事务中使用的指令操作码。当 SSI 被启用（SSIC_EN = 1）时，无法向该寄存器写入数据。通过向 SSIENR 寄存器写入数据，可启用或禁用 SSI。

INCRINST — XIP INCR 传输操作码。

当 SPI_CTRLR0.XIP_INST_EN 位设置为 1 时，SSI 会为所有 XIP 传输发送指令。此寄存器字段存储当在 AHB 总线上请求 INCR 类型传输时要发送的指令操作码。指令阶段要发送的位数由 SPI_CTRL0.INST_L 字段决定。

22.6.29. XIP Wrap Inst Register（XIPWIR）

偏移地址：0x0104

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	WRAP_INST							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	WRAP_INST							
W								
RESET:	0	1	1	0	1	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

图 22-30：XIP Wrap Inst 寄存器（XIPWIR）

该寄存器仅在 SSIC_XIP_INST_EN 等于 1 时有效。该寄存器用于存储当 AHB 接口请求相同操作时 WRAP 事务中使用的指令操作码。当 SSI 被启用（SSIC_EN = 1）时，无法对该寄存器进行写入。通过向 SSIENR 寄存器写入数据可启用或禁用 SSI。

WRAP_INST — XIP WRAP 转移操作码。

当 SPI_CTRLR0.XIP_INST_EN 位设置为 1 时，SSI 会为所有 XIP 传输发送指令。此寄存器字段存储在 AHB 总线请求 WRAP 类型传输时要发送的指令操作码。指令阶段要发送的位数由 SPI_CTRL0.INST_L 字段决定。

22.6.30. XIP 控制寄存器（XIPCR）

偏移地址：0x0108

	31	30	29	28	27	26	25	24
R	0	0	XIP_PREFETCH_EN	0	XIP_MBL		reserved	reserved
W								
RESET:	0	0	1	0	1	0	0	0

	23	22	21	20	19	18	17	16
R	CONT_XFER_EN	INST_EN	reserved	reserved	reserved	reserved	WAIT_CYCLES	
W								
RESET:	1	1	0	0	0	0	0	1

	15	14	13	12	11	10	9	8
R	WAIT_CYCLES			MD_BITS_EN	0	INST_L		0
W								
RESET:	0	0	0	0	0	1	0	0

	7	6	5	4	3	2	1	0
R	ADDR_L				TRANS_TYPE		FRF	
W								
RESET:	0	1	1	0	0	0	1	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-31：XIP 控制寄存器（XIPCR）

此寄存器仅在 SSIC_CONCURRENT_XIP_EN 等于 1 时有效。此寄存器用于存储将在并发模式下使用的 XIP 传输的控制信息。

XIP_PREFETCH_EN

1 = 在 SSI 中启用 XIP 预取功能。

0 = 禁用 SSI 中的 XIP 预取功能。

XIP_MBL — XIP 模式位长度。

设置 XIP 模式下运行的模式位长度。这些位仅在 XIP_CTRL.XIP_MD_BIT_EN 被设置为 1 时有效。

0x0（MBL_2）：模式位长度等于 2

0x1（MBL_4）：模式位长度等于 4

0x2（MBL_8）：模式位长度等于 8

0x3（MBL_16）：模式位长度等于 16

CONT_XFER_EN

1 = 启用 XIP 模式下的连续传输。

0 = 禁用 XIP 模式中的连续传输。

INST_EN

1 = XIP 传输将具有指令阶段。

0 = XIP 传输将不具有指令阶段。

等待周期 — 双通道/四通道/八通道模式下，控制帧发送与数据接收之间的等待周期。以 SPI 时钟周期数表示。

0x0：0 个等待周期

0x1：1 个等待周期

0x2：2 个等待周期

....

....

0x1F：31 个等待周期

MD Bits_EN — XIP 模式中的模式位启用。

1 = 在地址阶段后插入模式位。

0 = 地址阶段之后没有模式位。

INST_L

双/四/八模式指令长度，单位为比特。

0x0（INST_L0）：无指令

0x1（INST_L4）：4 位指令长度

0x2（INST_L8）：8 位指令长度

0x3（INST_L16）：16 位指令长度

ADDR_L

该位定义要传输的地址长度。只有将这么多位编程到 FIFO 之后，才能开始传输。

0x0（ADDR_L0）：无地址

0x1（ADDR_L4）：4 位地址长度

0x2（ADDR_L8）：8 位地址长度

0x3（ADDR_L12）：12 位地址长度

0x4（ADDR_L16）：16 位地址长度

0x5（ADDR_L20）：20 位地址长度

0x6（ADDR_L24）：24 位地址长度

0x7（ADDR_L28）：28 位地址长度

0x8（ADDR_L32）：32 位地址长度

0x9（ADDR_L36）：36 位地址长度

0xA（ADDR_L40）：40 位地址长度

0xB（ADDR_L44）：44 位地址长度

0xC（ADDR_L48）：48 位地址长度

0xD（ADDR_L52）：52 位地址长度

0xE（ADDR_L56）：56 位地址长度

0xF（ADDR_L60）：60 位地址长度

TRANS_TYPE

地址和指令传输格式。

选择 SSI 是在标准 SPI 模式还是在 CTRLR0 中选择的 SPI 模式下发送指令/地址。SPI_FRF 字段。

0x0（TT0）：指令和地址将通过标准 SPI 模式发送。

0x1（TT1）：指令将通过标准 SPI 模式发送，地址将通过 XIP_ctrl 指定的模式发送。SPI_FRF。

0x2（TT2）：指令和地址将按照 XIP_CTRL.FRF 指定的模式发送。

0x3（TT3）：保留。

FRF — SPI 帧格式

选择用于发送/接收数据的数据帧格式。

0x0（RSVD）：保留

0x1（SPI_DUAL）：双 SPI 格式

0x2（SPI_QUAD）：四通道 SPI 格式

0x3（SPI_OCTAL）：八位 SPI 格式

22.6.31. XIP 从机使能寄存器（XIPSER）

偏移地址：0x010C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	SER
W								
RESET:	0	0	0	0	0	0	0	1

= Writes have no effect and the access terminates without a transfer error exception.

图 22-32：XIP 从机使能寄存器（XIPSER）

该寄存器仅在 SSIC_CONCURRENT_XIP_EN 等于 1 时有效。该寄存器启用 SSI 主控器的各个从选输出线，用于 XIP 操作模式。SSI 主设备最多有 16 个从设备选择输出引脚可用。当 SSI 处于忙状态或 SSIC_EN = 1 时，无法对这个寄存器进行写入操作。

SER — 从属设备选择启用标志。

该寄存器中的每个比特位对应 SSI 主设备的从设备选择线（ss_x_n）。当寄存器中的某个比特位被置位(1)时，主设备对应的从设备选择线会在 XIP 传输开始时激活。需要注意的是，在 XIP 传输启动前，对寄存器中比特位的设置或清除操作不会影响对应的从设备选择输出。在传输开始前，应先启用与主设备需要通信的从设备相对应的寄存器比特位。当设备未处于广播模式时，该字段中仅需设置一个比特位。

1 = 选择

0 = 未选择

22.6.32. XIP 接收 FIFO 溢出中断清除寄存器（XRXIOCR）

偏移地址：0x0110

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	XRXOICR
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-33：XIP 接收 FIFO 溢出中断清除寄存器（XRXIOCR）

XRXOICR — 清除 XIP 接收 FIFO 溢出中断。

此寄存器反映中断的状态。从该寄存器读取将清除 ssi_xrxo_intr（_n）中断，写入则没有影响。

22.6.33. XIP 继续传输超时寄存器（XIPCTTOR）

偏移地址：0x0114

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	XTOC							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 22-34：XIP 持续传输超时寄存器（XIPCTTOR）

连续模式下的 XIP 计数器寄存器。该计数器用于在连续传输模式下取消选择从设备。当 SSI 被启用（SSIC_EN = 1）时，无法对该寄存器进行写入操作。通过向 SSIENR 寄存器写入数据可启用或禁用 SSI 功能。

XTOC — 以 hclk 为单位的 XIP 超时值。

一旦在连续 XIP 模式中选择了从属设备，如果计数器中指定的时间没有请求，则该计数器将用于取消选择从属设备。

22.7.功能说明

22.7.1. 主模式

该模式支持与串行从属外围设备进行串行通信。当配置为串行主设备时，SSI 将启动并控制所有串行传输。图 31-35 显示了 SSI 配置为串行主设备的示例，串行总线上的所有其他设备均配置为串行从属设备。

由 SSI 生成和控制的串行比特率时钟通过 sclk_out 线输出。当 SSI 被禁用（SSIC_EN = 0）时，无法进行串行传输，并且 sclk_out 保持在“非活动”状态，如其运行的串行协议所定义。

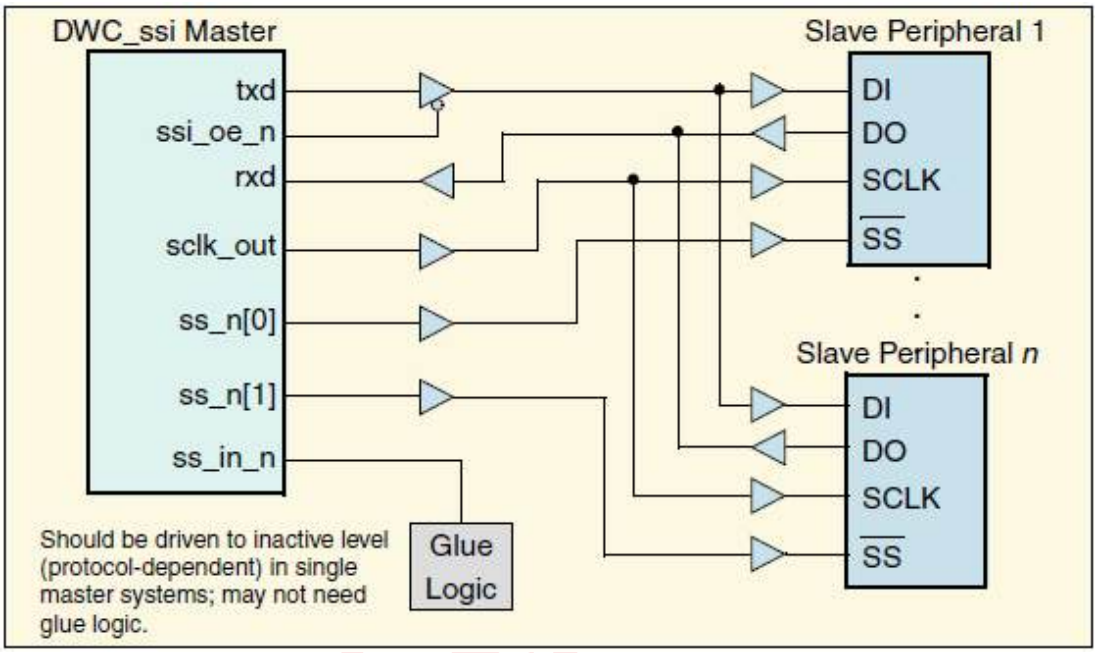


图 22-35：SSI 配置为主设备

22.7.2. 时钟比率

SSI 采用过采样架构。在主模式下，外围时钟（sclk_out）周期是内部核心时钟（ssi_clk）的倍数。

当 SSI 宏单元被配置为主设备时，比特率时钟（sclk_out）的最大频率是 ssiClock 频率的一半。这是为了允许移位控制逻辑在 sclk_out 的一个时钟边沿捕获数据，并在相反的边沿传播数据。

sclk_out 的频率可通过以下方程式推导。

$$F_{sclk\_out} = \frac{F_{ssi\_clk}}{SCKDV}$$

SCKDV 是一个可编程寄存器，保存 0~65534 范围内的任何偶数值。如果 SCKDV = 0，则禁用 sclk_out。

22.7.3. 接收和发送 FIFO 缓冲器

SSI 系统采用的 FIFO 缓冲器是内部 D 型触发器。根据串行传输规范要求，发送和接收 FIFO 缓冲器的宽度均固定为 32 位，该规范规定数据帧长度可在 4 至 32 位之间变化。当数据帧长度不足 32 位时，必须在写入发送 FIFO 缓冲器时进行右对齐处理。接收 FIFO 缓冲器中的数据则由移位控制逻辑自动完成右对齐操作。

22.7.3.1. 传输 FIFO

发送 FIFO 通过 AHB 写入指令将数据加载到 SSI 数据寄存器（DR）中。移位控制逻辑会从发送 FIFO 中弹出（移除）数据至发送移位寄存器。当 FIFO 中的条目数量小于或等于阈值时，发送 FIFO 会触发 FIFO 空闲中断请求（ssi_txe_intr）。该阈值由可编程寄存器 TXFTLR 设定，用于确定触发中断的 FIFO 条目数量上限。该阈值可提前向处理器发出信号，提示发送 FIFO 即将满载。若尝试向已满载的发送 FIFO 写入数据，则会触发发送 FIFO 溢出中断（ssi_txo_intr）。

22.7.3.2. 接收 FIFO

数据通过 AHB 读取指令从接收 FIFO 中弹出并存入 SSI 数据寄存器（DR）。移位控制逻辑从接收移位寄存器中加载接收 FIFO。当 FIFO 中的条目数量达到或超过 FIFO 阈值加 1 时，接收 FIFO 会触发 FIFO 满中断请求（ssi_rxf_intr）。该阈值由可编程寄存器 RXFTLR 设定，用于确定触发中断的 FIFO 条目数量阈值。

阈值设置能及时向处理器发出接收 FIFO 快满的预警信号。当接收移位逻辑试图将数据存入已满的接收 FIFO 时，系统会触发接收 FIFO 溢出中断（ssi_rxo_intr），但此时新接收的数据将丢失。若尝试从空置的接收 FIFO 中读取数据，则会触发接收 FIFO 下溢中断（ssi_rxu_intr），这相当于给处理器发出了“读取无效数据”的警报。

22.7.4. DMA 操作

SSI 芯片内置可选的 DMA 功能模块，可在配置阶段进行选择。该模块通过与 DMA 控制器建立握手接口，实现数据传输请求与控制。数据传输通过 AHB 总线完成。虽然 SSI 的 DMA 操作采用通用设计以适配各类 DMA 控制器，但其核心设计旨在实现无缝衔接与最佳性能表现。要启用 SSI 的 DMA 控制器接口，需对 DMA 控制寄存器（DMACR）进行写入操作：在 DMACR 寄存器的 TDMAE 位字段写入 1，即可启用发送端握手接口；在 RDMAE 位字段写入 1，则可启用接收端握手接口。

22.7.5. SSI 中断

SSI 中断的描述如下：

发送 FIFO 空中断（ssi_txe_intr） — 当发送 FIFO 队列容量等于或低于阈值时触发此中断，需及时处理以防止发生欠载。该阈值通过软件可编程寄存器设定，用于确定触发中断的 FIFO 队列条目数量。当数据写入发送 FIFO 缓冲区使队列容量超过阈值时，硬件会自动清除此中断。

发送 FIFO 溢出中断（ssi_txo_intr） — 当 AHB 访问尝试在发送 FIFO 已完全填满后写入其中时，

将设置此中断。设置后，从 AHB 写入的数据将被丢弃。此中断将持续保持设置状态，直到您读取发送 FIFO 溢出中断清除寄存器（TXOICR）为止。

接收 FIFO 满中断（ssi_rxf_intr） — 当接收 FIFO 队列长度等于或超过阈值加 1 时触发，此时需要处理以防止溢出。该阈值通过软件可编程寄存器设置，用于确定触发中断的接收 FIFO 队列条目数量。当数据从接收 FIFO 缓冲区读取时，硬件会清除该中断，使队列长度低于阈值水平。

接收 FIFO 溢出中断（ssi_rxo_intr） — 当接收逻辑尝试将数据放入已完全填满的接收 FIFO 时触发。设置后，新接收的数据将被丢弃。此中断将持续保持，直到您读取接收 FIFO 溢出中断清除寄存器（RXOICR）。

接收 FIFO 下溢中断（ssi_rxu_intr） — 当 AHB 访问尝试从空的接收 FIFO 中读取时触发。设置后，将从接收 FIFO 中读取零值。此中断将持续保持，直到您读取接收 FIFO 下溢中断清除寄存器（RXUICR）。

组合中断请求（ssi_intr） — 屏蔽所有上述中断请求后进行或运算的结果。要屏蔽此中断信号，必须屏蔽所有其他 SSI 中断请求。

22.7.6. 增强的 SPI 模式

SSI 通过 SSIC_SPI_MODE 配置参数支持 SPI 的双通道、四通道和八通道工作模式。该参数的可能取值包括标准模式、双通道 SPI、四通道 SPI 和八通道 SPI 模式。当选择双通道、四通道或八通道模式时，TXD、RXD 和 ssi_oe_n 信号的宽度将分别调整为 2、4 或 8。这意味着数据会通过多条线路进行传输输出/输入，从而提升整体吞吐量。双通道 SPI、四通道 SPI 和八通道 SPI 模式的工作原理基本相同，主要区别在于 TXD、RXD 和 ssi_oe_n 信号的宽度设置。操作模式（写入/读取）可通过 CTRLR0.TMOD 字段进行选择。

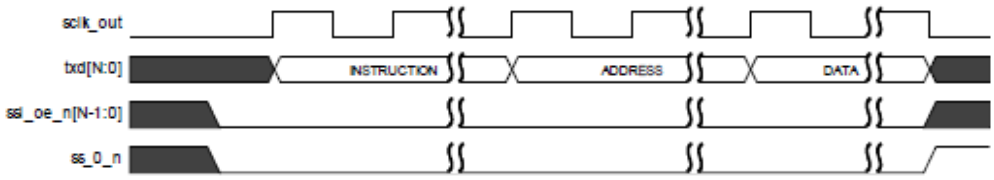


图 22-36：典型写入操作双/四/八 SPI 模式

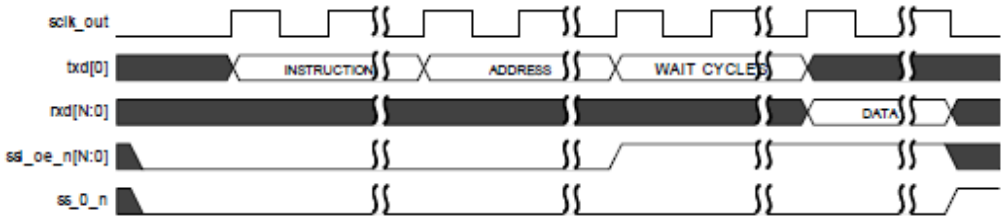


图 22-37：典型读取操作双/四/八 SPI 模式

22.7.7. 执行就地（XIP）模式

SSI 提供了一种直接从 AHB 事务中执行内存读取操作的功能，这种模式被称为就地执行模式。在此模式下，SSI 作为 SPI 内存的内存映射接口工作。通过选择配置参数 SSIC_XIP_EN，可在 SSI 中启用 XIP 模式。该模式会在 AHB 接口上生成额外的侧边带信号 xip_en，其电平高低决定了 AHB 传输是寄存器读写还是 XIP 读取。在 XIP 操作期间仅支持 AHB 读取。当 xip_en 信号被置为 1 时，SSI 会预期在 AHB 接口上发出读取请求。该请求会被转换为串行接口上的 SPI 读取操作。数据接收后，会立即通过同一事务返回至 AHB 接口。haddr 参数用于推导要发送到 SPI 接口的地址。某些设备要求在 XIP 传输期间存在指令阶段。SSI 支持在 XIP 操作模式下包含某些固定指令集。

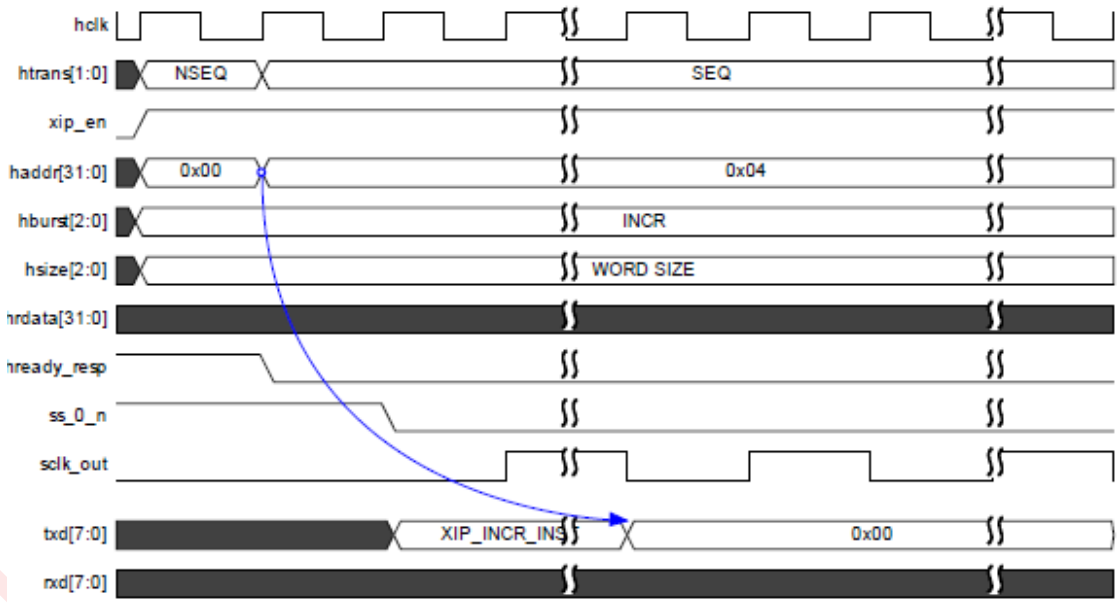


图 22-38：带指令阶段的 XIP 传输

22.7.8. XIP 中的连续传输模式

当 SSI 收到 XIP 请求时，来自 AHB 接口的地址会直接通过 SPI 接口传输。AHB 接口上的每个新传输(XIP 读取）都以相同的方式处理。因此，对于每个请求，必须向设备发送新的地址，从而导致系统延迟。

若存储设备支持在 XIP 读取传输间隙拉伸从属选择信号，即可通过编程设置连续 XIP 模式来提升性能。在此模式下，主机将两个或多个 AHB 突发请求整合为单一 SPI 命令，既确保了命令与地址不会重复传输，又省去了等待这些突发操作间空闲周期的冗余步骤。

启用此功能后，当首次接收到 XIP 命令时，SSI 会立即以连续 XIP 模式运行。在首次 XIP 传输中，地址信息通过 SPI 接口发送。接收到请求数据后，SSI 会持续保持从设备选中状态，同时时钟信号（sclk_out）仍保持默认工作模式。后续通过 AHB 接口进行的 XIP 传输时，SSI 将重新启动时钟信号（sclk_out），此时既不通过 SPI 接口发送命令或地址，也不会立即从设备读取数据（无需等待空闲周期）。

- 不支持在连续读取模式下使用未定义的 INCR（hburst = 001）突发。

在持续传输过程中，由于始终选择从设备，因此从设备上耗散了大量功率。为了避免这种情况，SSI 提供了一个配置选项，以启用看门狗计时器，在计数器运行完毕后取消选择从设备。

在以下情况下，SSI 可取消选择从属设备：

- 在 XIP 接口上接收到非 XIP 命令（实际上，任何由 xip_en 驱动为 0 的 AHB 事务）。
- 当 AHB 事务要发送到非连续地址时，将移除从属选择，然后 SSI 发起新的 XIP 请求。
- 在 XIP_CNT_TIME_OUT 寄存器指定的时间段内，SSI 未在 AHB 接口上检测到任何 XIP 传输。

22.7.9. XIP 操作中的数据预取

通过 SSI 的数据预取功能，控制器能在当前 XIP 事务处理期间预先获取连续地址突发数据。当后续事务请求指向连续地址时，系统可直接从接收端 FIFO 读取数据，无需等待新地址和数据传输完成。这种设计显著提升了系统的整体性能。

预取数据量应等于最后一个 AHB 请求的突发长度或 FIFO 深度（以较低者为准）。例如，若 AHB 定义从地址 0x00 开始的突发长度为 16 个节拍，则 SSI 会预取 16 个节拍完成当前传输，随后再次预取并存入数据寄存器的 16 个数据帧。当 AHB 总线再次请求从末地址开始的最后一批数据时，剩余数据将直接从接收 FIFO 发送至设备。与此同时，SSI 会启动 XIP 传输向设备预取下一数据块。若 AHB 主控未定义连续地址，则当前数据将被清空出 FIFO，控制器将重新发起事务处理。

当 SSI 完成当前突发并开始预取下一突发的数据时，可能会发出新的 XIP 请求。这种情况下，SSI 可以根据地址终止当前传输或增加要获取的数据节拍数。

•如果启用了 XIP 预取，则不允许使用 AHB 请求进行未定义长度的增量传输（hburst = 3‘b001）。

23. 串行外围接口模块（SPI）

23.1. 介绍

串行外围接口（SPI）模块允许微控制器单元（MCU）与外围设备之间进行全双工同步串行通信。软件可以轮询 SPI 状态标志，也可以通过中断驱动 SPI 操作。

23.2. 特性

其特点包括：

- 主模式和从属模式
- 有线-OR 模式
- 选择性输出
- 与中央处理器（CPU）中断功能相关的故障错误标志模式
- 在待机模式下控制 SPI 操作
- 降低驱动控制以降低功耗
- 可编程接口操作，适用于飞思卡尔 SPI 或德州仪器同步串行接口
- 分离的发送和接收 FIFO，每个 8 位宽和 8 位深
- 可编程数据帧大小为 4 至 16 位
- 用于诊断/检测测试的内部环回测试模式
- 基于 FIFO 的标准中断和传输结束中断
- 通过 DMA 接口实现高效传输
- 可以查看 TX 和 RX FIFO，便于调试
- 高速模式，用于传输时间调整

23.3. 操作模式

SPI 在以下三种模式下工作：

1. 运行模式 — 运行模式是正常工作模式。
2. Doze 模式 — Doze 模式是一种可配置的低功耗模式。
3. 停止模式 — SPI 在停止模式下处于非活动状态。

23.4. 方块图

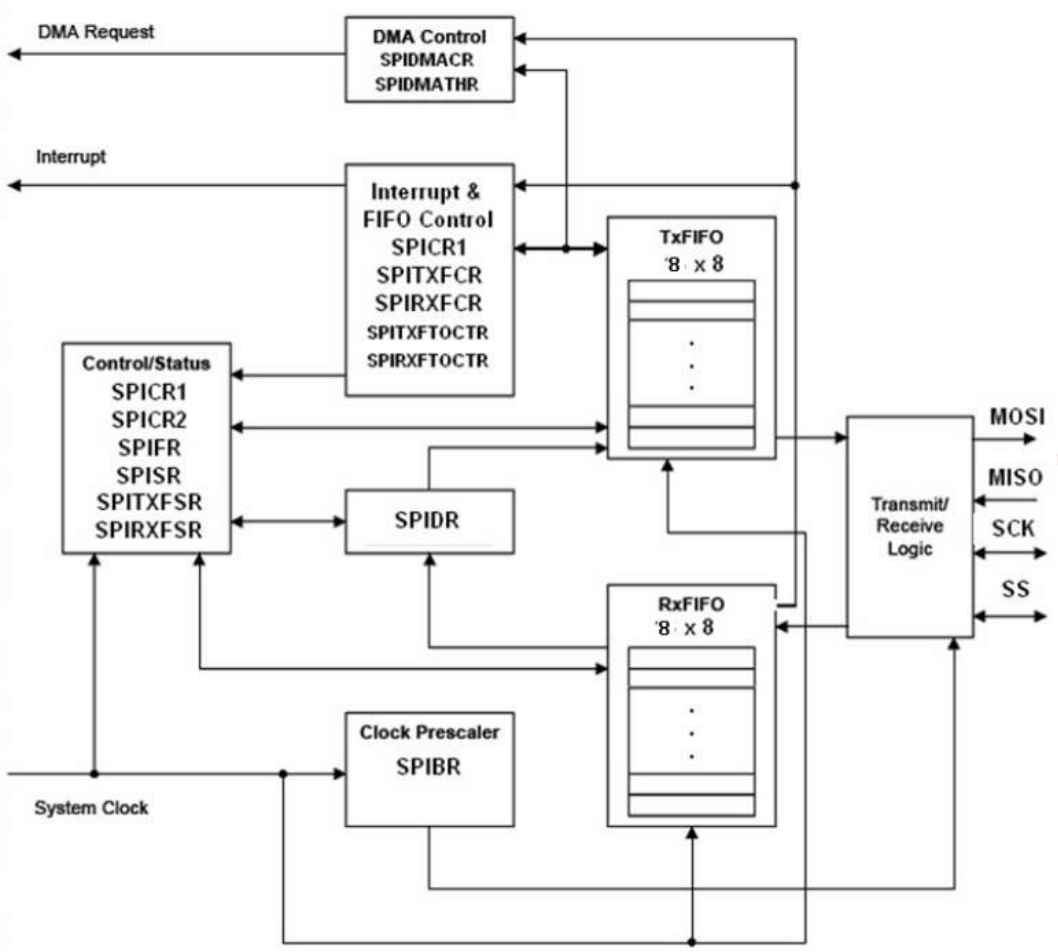


图 23-1：SPI 方块图

23.5. 信号描述

信号概述见表 23-1。

表 23-1：信号特性

Name	Port	Function[1]	Reset State
MISO	SPIPORT[0]	Master Data In / Slave Data Out	0
MOSI	SPIPORT[1]	Master Data Out / Slave Data In	0
SPI_SCK	SPIPORT[2]	Serial Clock	0
SPI_CS#	SPIPORT[3]	Slave Select	0

提示：当 SPI 被禁用（SPE = 0）时，特定 SPI 端口（MISO、MOSI、SPI_SCK、SPI_CS#）是 GP I/O 端口。

23.5.1. MISO（主输入/从输出）

MISO 是 SPI 数据引脚中的两个引脚之一。

- 在主模式下，MISO 是数据输入。
- 在从属模式下，MISO 是数据输出，并且在主设备将 SPI_CS#输入引脚驱动为低电平时处于三种状态。
- 在双向模式下，从属 MISO 引脚是 SISO 引脚（从属输入/从属输出）。
- 在多主系统中，所有 MISO 引脚都连接在一起。

23.5.2. MOSI（主输出/从输入）

MOSI 是 SPI 数据引脚中的两个引脚之一。

- 在主模式下，MOSI 是数据输出。
- 在从属模式下，MOSI 是数据输入。
- 在双向模式下，MOSI 主控引脚是 MOMI 引脚（主输出/主输入）。
- 在多主系统中，所有 MOSI 引脚都连接在一起。

23.5.3. SPI_SCK（串行时钟）

SCK 引脚是用于同步主设备和从设备之间传输的串行时钟引脚。

- 在主模式下，SPI_SCK 是一个输出。
- 在从属模式下，SPI_SCK 是输入。
- 在多主系统中，所有 SPI_SCK 引脚都连接在一起。

23.5.4. SPI_CS#（从属选择）

在主模式下，SPI_CS#引脚可以：

- 模式故障输入
- 通用输入
- 通用输出
- 从属选择输出

在从属模式下，SPI_CS#引脚始终是从属选择输入。

23.6. 内存映射和寄存器

SPI 内存映射如表 23-2 所示。

表 23-2：SPI 内存映射

地址偏移量	Bit[7:0]	访问权限
0x0000	SPI 控制寄存器 1（SPICR1）	S/U
0x0001	SPI 控制寄存器 2（SPICR2）	S/U
0x0002	SPI 波特率寄存器（SPIBR）	S/U
0x0003	SPI 帧寄存器（SPIFR）	S/U
0x0004	SPI RXFIFO 控制寄存器（SPIRXFCR）	S/U
0x0005	SPI TXFIFO 控制寄存器（SPITXFCR）	S/U
0x0006	SPI RX FIFO 超时计数器寄存器（SPIRXFTOCTR）	S/U
0x0007	SPI TX FIFO 超时计数器寄存器（SPITXFTOCTR）	S/U
0x0008	SPI 端口数据方向寄存器（SPIDDR）	S/U
0x0009	SPI Pullup 和 Reduced Drive Register（SPIPURD）	S/U
0x000A	SPI SCK 延迟寄存器（SPIASCDR）	S/U
0x000B	SPI 在 SCK 延迟寄存器之前（SPIBSCDR）	S/U
0x000C	SPI 端口数据寄存器（SPIPORT）	S/U
0x000D ~ 0x000F	SPI 发送计数器寄存器（SPITCNT）	S/U
0x0010	SPI 数据寄存器（SPIDR）	S/U
0x0014	SPI 状态寄存器（SPISR）	S/U
0x0016	SPI RX FIFO 状态寄存器（SPIRXFSR）	S/U
0x0017	SPI TX FIFO 状态寄存器（SPITXFSR）	S/U
0x0018	SPI DMA 控制寄存器（SPIDMACR）	S/U
0x0019	SPI DMA 阈值寄存器（SPIDMATHR）	S/U
0x001A	SPI FIFO 调试控制寄存器（SPIFDCR）	S/U
0x001B	SPI 中断控制寄存器（SPIICR）	S/U
0x001C	SPI RX FIFO 调试寄存器（SPIRXFDBGR）	S/U
0x001E	SPI TX FIFO 调试寄存器（SPITXFDBGR）	S/U

23.6.1. SPI 控制寄存器

偏移地址：0x0000

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SPIE	SPE	SWOM	MSTR	CPOL	CPHA	SSOE	LSBFE
RESET:	0	0	0	0	0	1	0	0

图 23-2：SPI 控制寄存器 1（SPICR1）

Read：任何时候

Write：任何时候

SPIE — SPI 中断使能位

SPIE 位使 EOTF 标志能够生成中断请求。复位清除 SPIE。

1 = EOTF 中断请求已启用

0 = 禁用 EOTF 中断请求

SPE — SPI 系统使能位

SPE 位启用 SPI，并将 SPI 端口引脚[3:0] 专门用于 SPI 功能。当 SPE 被清除时，SPI 系统处于初始化状态但处于低功耗禁用状态。复位清除了 SPE。

1 = 启用 SPI

0 = SPI 处于禁用状态

SWOM — SPI 有线或模式比特

SWOM 位将 SPI 端口引脚[3:0] 的输出缓冲器配置为开漏输出。SWOM 控制 SPI 端口引脚[3:0] 是 SPI 输出还是通用输出。复位清除了 SWOM。

1 = SPI 端口引脚[3:0] 的输出缓冲器采用开漏模式

0 = SPI 端口引脚[3:0] 的输出缓冲器 CMOS 驱动

提示：SWOM 位在此部分中不起作用。

MSTR — 主比特

MSTR 位选择 SPI 主模式或 SPI 从模式操作。复位清除 MSTR。

1 = 主模式

0 = 从机模式

CPOL — 时钟极性位

CPOL 位选择反相或非反相的 SPI 时钟。要在 SPI 模块之间传输数据，SPI 模块必须具有相同的 CPOL 值。复位清除 CPOL。

1 = 低电平有效时钟；SPI_SCK 空闲时高电平

0 = 高电平有效时钟；SPI_SCK 低电平空闲

CPHA — 时钟相位比特

CPHA 位延迟 SPI_SCK 时钟的第一个边沿。复位设置 CPHA。

1 = 传输开始时的第一个 SPI_SCK 边沿

0 = 传输开始后第一个 SPI_SCK 边沿的半个周期

提示：在传输期间（SPI_CS#为低值时），CPOL 或 CPHA 的任何数值变化都可能导致错误结果。请在启动传输之前（SPI_CS#为高值时）更改 CPOL 或 CPHA 值。

SSOE — 从属选择输出使能位

SSOE 位和 DDRSP3 位将 SPI_CS#引脚配置为通用输入或从属选择输出。复位将清除 SSOE。

表 23-3：SPI_CS# Pin I/O 配置

DDRSP3	SSOE	Master Mode	Slave Mode
0	0	Mode-Fault Input	Slave-Select Input
0	1	General-Purpose Input	Slave-Select Input
1	0	General-Purpose Output	Slave-Select Input
1	1	Slave-Select Output	Slave-Select Input

提示：设置 SSOE 位禁用模式故障检测功能。

LSBFE — LSB-第一个使能位

LSBFE 使数据能够以 LSB 为先进行传输。复位清除了 LSBFE。

1 = 先发送数据的 LSB。

0 = 以 MSB 为先传输数据

23.6.2. SPI 控制寄存器 2

偏移地址：0x0001

	Bit7	6	5	4	3	2	1	Bit0
Read: Write:	GT[5:0]						SPISDOZ	SPC0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 23-3：SPI 控制寄存器 2（SPICR2）

Read：任何时候

Write：任何时候

GT — 保护时间位

GT = (GT[5:3]+1)*2(GT[2:0]+1)

SPISDOZ — 十二位 SPI 接口

当 CPU 处于休眠模式时，SPIDOZ 位会停止 SPI 时钟。

1 = 在 Doze 模式下 SPI 处于非活动状态

0 = 在 Doze 模式下激活的 SPI

SPC0 — 串行引脚控制位 0

SPC0 位启用表 23-4 所示的双向引脚配置。

表 23-4：双向引脚配置

	Pin Mode	SPC0	MSTR	MISO Pin[1]	MOSI Pin[2]	SPI_SCK Pin[3]	SPI_CS# Pin[4]
A	Normal	0	0	Slave Data Output	Slave Data Input	SCK Input	Slave-Select Input
B			1	Master Data Input	Master Data Output	SCK Output	MODF/GP Input (DDRSP3 = 0) Or GP Output (DDRSP3 = 1)
C	Bidirectional	1	0	Slave Data I/O	GP[5] I/O	SCK Input	Slave-Select Input
D			1	Gp I/O	Master Data I/O	SCK Output	MODF/GP Input (DDRSP3 = 0) Or GP Output (DDRSP3 = 1)

提示：

1. 如果 SPIDDR 位 0 = 1、SPI_CS# = 0 且 MSTR = 0 (A，C），则启用从机输出。
2. 如果 SPIDDR 位 1 = 1 且 MSTR = 1 (B，D），则主输出被启用。
3. 如果 SPIDDR 位 2 = 1 且 MSTR = 1 (B，D），则启用 SCK 输出。
4. 若 SPI DDR 第 3 位为 1、SPICR1 第 1 位（SSOE）为 1 且 MSTR = 1 (B、D），则启用 SPI_CS#输出。若 SPI DDR 第 3 位为 0 且 SSOE 为 0，则启用 MODF 输入。若 SPI DDR 第 3 位为 0 且 SSOE 为 1，则启用 GP 输入。
5. GP = General-purpose 通用型

23.6.3. SPI 波特率寄存器

偏移地址：0x0002

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	SPPR6	SPPR5	SPPR4	0	SPR2	SPR1	SPR0
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 23-4：SPI 波特率寄存器（SPIBR）

Read：任何时候

Write：任何时候；对未实现的位进行写入没有影响

SPPR[6:4] — SPI 波特率预选位

SPPR[6:4] 和 SPR[2:0] 位选择 SPI 时钟分频，如表 23-5 所示。复位清除 SPPR[6:4] 和 SPR[2:0]，选择 SPI 时钟分频为 2。

SPR[2:0] — SPI 波特率位

SPPR[6:4] 和 SPR[2:0] 位选择 SPI 时钟分频，如表 23-5 所示。复位清除 SPPR[6:4] 和 SPR[2:0]，选择 SPI 时钟分频为 2。

提示：在传输过程中向 SPIBR 写入数据可能会导致错误结果。

表 23-5：SPI 波特率选择(10-MHz 模块时钟)

SPPR[6:4]	SPR[2:0]	SPI Clock Divisor	Baud Rate	SPPR[6:4]	SPR[2:0]	SPI Clock Divisor	Baud Rate
000	000	2	5 MHz	100	000	10	1 MHz
000	001	4	2.5 MHz	100	001	20	0.5 MHz
000	010	8	1.25 MHz	100	010	40	0.25 MHz
000	011	16	0.625 MHz	100	011	80	125 kHz
000	100	32	0.31 MHz	100	100	160	62.5 kHz
000	101	64	156.25 kHz	100	101	320	31.25 kHz
000	110	128	78.125 kHz	100	110	640	15.625 kHz
000	111	256	39.06 kHz	100	111	1280	7.81 kHz
001	000	4	2.5 MHz	101	000	12	833.33 kHz

表 23-6：SPI 波特率选择(10-MHz 模块时钟)

SPPR[6:4]	SPR[2:0]	SPI Clock Divisor	Baud Rate	SPPR[6:4]	SPR[2:0]	SPI Clock Divisor	Baud Rate
001	001	8	1.25 MHz	101	001	24	416.67 kHz
001	010	16	0.625 MHz	101	010	48	208.33 kHz
001	011	32	0.31 MHz	101	011	96	104.17 kHz
001	100	64	156.25 kHz	101	100	192	52.08 kHz
001	101	128	78.125 kHz	101	101	384	26.04 kHz
001	110	256	39.06 kHz	101	110	768	13.02 kHz
001	111	512	19.53 kHz	101	111	1536	6.51 kHz
010	000	6	1.67 MHz	110	000	14	714.29 kHz
010	001	12	0.83 MHz	110	001	28	357.14 kHz
010	010	24	0.42 MHz	110	010	56	178.57 kHz
010	011	48	208.33 kHz	110	011	112	89.29 kHz
010	100	96	104.17 kHz	110	100	224	44.64 kHz
010	101	192	52.08 kHz	110	101	448	22.32 kHz
010	110	384	26.04 kHz	110	110	896	11.16 kHz

SPPR[6:4]	SPR[2:0]	SPI Clock Divisor	Baud Rate	SPPR[6:4]	SPR[2:0]	SPI Clock Divisor	Baud Rate
010	111	768	13.02 kHz	110	111	1792	5.58 kHz
011	000	8	1.25 MHz	111	000	16	0.625 MHz
011	001	16	0.625 MHz	111	001	32	0.31 MHz
011	010	32	0.31 MHz	111	010	64	156.25 kHz
011	011	64	156.25 kHz	111	011	128	78.125 kHz
011	100	128	78.125 kHz	111	100	256	39.06 kHz
011	101	256	39.06 kHz	111	101	512	19.53 kHz
011	110	512	19.53 kHz	111	110	1024	9.77 kHz
011	111	1024	9.77 kHz	111	111	2048	4.88 kHz

23.6.4. SPI 帧寄存器

偏移地址：0x0003

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	CONT	GTE	LBM	FFSEL	FMSZ3	FMSZ2	FMSZ1	FMSZ0
RESET:	0	1	0	0	0	1	1	1

图 23-5：SPI 帧寄存器（SPIFR）

Read：任何时候

Write：任何时候

CONT — 持续外围芯片选择使能

1 = 在传输之间保持外围芯片选择信号低，直到 EOTF 被设置

0 = 在传输之间将外围芯片选择信号置为高电平

GTE — 保护时间启用

1 = 已启用保护时间

0 = 禁用保护时间

LBM — 循环模式

1 = 循环模式

0 = 正常模式

FFSEL — 帧格式选择

1 = 选择 TI 帧格式

0 = 选择 FREESCALE 帧格式

FMSZ[3:0] — 帧大小

FMSZ[3:0] 位控制帧数据长度，范围为 4 到 16 位。其中 0x3 设置帧长为 4 位，0xF 设置为 16 位。同时，0x0~0x2 案例会自动将帧长设置为 4 位。

23.6.5. SPI RX FIFO 控制寄存器

偏移地址：0x0004

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RXFCLR	RXFOVIE	RXFUDIE	RXFSTHIE	0	RXFSTH2	RXFSTH1	RXFSTH0
RESET:	0	0	0	0	0	0	0	0

图 23-6：SPI RX FIFO 控制寄存器（SPIRXFCR）

Read：任何时候

Write：任何时候

RXFCLR — RX FIFO 清除

将 1 写入该位将重置 RX FIFO。

RXFOVIE — RX FIFO 溢出中断使能

1 = RX FIFO 溢出中断已启用

0 = RX FIFO 溢出中断已禁用

RXFUDIE-RX FIFO 下溢中断使能

1 = RX FIFO 下溢中断已启用

0 = RX FIFO Underflow Interrupt Disabled

RXFSTHIE — RX FIFO 服务阈值中断启用

1 = RX FIFO 服务阈值中断已启用

0 = RX FIFO 服务阈值中断已禁用

RXFSTH[2:0] — RX FIFO 服务阈值

一旦 RX FIFO 中的有效数据数高于或等于指定阈值，RX FIFO 服务标志将被断言。

数据编号 = RXFSTH[2:0] + 1

23.6.6. SPI TX FIFO 控制寄存器

偏移地址：0x0005

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TXFCLR	TXFOVIE	TXFUDIE	TXFSTHIE	0	TXFSTH2	TXFSTH1	TXFSTH0
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 23-7：SPI TX FIFO 控制寄存器（SPITXFCR）

Read：任何时候

Write：任何时候

TXFCLR — TX FIFO 清除

将 1 写入该位将重置 TXFIFO

TXFOVIE — TX FIFO 溢出中断使能

1 = TX FIFO 溢出中断已启用

0 = TX FIFO 溢出中断已禁用

TXFUDIE — TX FIFO 下溢中断使能

1 = TX FIFO 下溢中断已启用

0 = TX FIFO 下溢中断已禁用

TXFSTHIE — TX FIFO 服务阈值中断启用

1 = TX FIFO 服务阈值中断已启用

0 = TX FIFO 服务阈值中断已禁用

TXFSTH[2:0] — TX FIFO 服务阈值

一旦 TX FIFO 中的有效数据数低于或等于指定阈值，TX FIFO 服务标志将被断言。

数据编号 = TXFSTH[2:0]

23.6.7. SPI RX FIFO 超时计数器寄存器

偏移地址：0x0006

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RXFTOIE	RXFTOE	5	4	3	2	1	0
RESET:	0	0	1	0	0	0	0	0

图 23-8：SPI RX FIFO 超时计数器寄存器（SPIRXFTOCTR）

Read：任何时候；

Write：任何时候；

SPIRXFTOCTR[5:0] 设置 SPI RX FIFO 超时计数器编号。一旦 RX FIFO 不为空，计数器就打开。如果在计数器倒计时至 0 之前没有 RX FIFO 操作，则 SPISR 中的 RXF_TIMEOUT 标志将被设置。

RXFTOIE — RX FIFO 超时中断使能

1 = RX FIFO 超时中断已启用

0 = RX FIFO 超时中断已禁用

RXFTOE — RX FIFO 超时功能启用

1 = RX FIFO 超时功能已启用

0 = RX FIFO 超时功能已禁用

23.6.8. SPI TX FIFO 超时计数器寄存器

偏移地址：0x0007

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TXFTOIE	TXFTOE	5	4	3	2	1	0
RESET:	0	0	1	0	0	0	0	0

图 23-9：SPI TX FIFO 超时计数器寄存器（SPITXFTOCTR）

Read：任何时候；

Write：任何时候；

SPITXFTOCTR[5:0] 设置 SPI TX FIFO 超时计数器数值。当 TX FIFO 未清空时，计数器处于开启状态。如果 TX FIFO 在计数器倒计时至 0 之前没有操作，SPISR 中的 TXF_TIMEOUT 标志将被设置。

TXFTOIE — TX FIFO 超时中断使能

1 = TX FIFO 超时中断已启用

0 = TX FIFO 超时中断已禁用

TXFTOE — TX FIFO 超时功能启用

1 = TX FIFO 超时功能已启用

0 = TX FIFO 超时功能已禁用

23.6.9. SPI 端口数据方向寄存器

偏移地址：0x0008

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RSVD7	RSVD6	RSVD5	RSVD4	DDRSP3	DDRSP2	DDRSP1	DDRSP0
RESET:	0	0	0	0	0	0	0	0
Pin function:					SS	SCK	MOSI/ MOMI	MISO/ SISO

图 23-10：SPI 端口数据方向寄存器（SPIDDR）

Read：任何时候

Write：任何时候

RSVD[7:4] — 保留

向这些读/写位写入数据会更新其值，但不会影响功能。

DDRSP[3:0] — 数据方向位

DDRSP[3:0] 位控制 SPIPORT 引脚的数据方向。复位清除了 DDRSP[3:0]。

1 = 配置为输出的对应引脚

0 = 配置为输入的对应引脚

在从属模式下，DDRSP3 没有意义或效果。在主控模式下，DDRSP3 和 SSOE 位确定 SPI 端口引脚 3 是模式故障输入、通用输入、通用输出还是从属选择输出。

提示：当 SPI 被启用（SPE = 1）时，MISO、MOSI 和 SPI_SCK 引脚：

- 输入是输入函数，无论其 DDRSP 位的状态如何。
- 如果 SPI 功能的输出函数仅当 DDRSP 位被设置时才输出

23.6.10. SPI 上拉和驱动寄存器降低

偏移地址：0x0009

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	HS	PSW	RESERVED		MSPD1	MSPD0	PDPSP	PUPSP
RESET:	0	0	0	0	0	0	0	1

图 23-11：SPI 上拉和降低驱动寄存器（SPIPURD）

Read：任何时候；

Write：任何时候；写入未实现的位没有效果

HS-高速模式启用位

在主模式下将 HS 设置为 1，使 MSPD[1:0] 生效。

1 = 启用高速模式

0 = 高速模式已禁用

PSW — 引脚切换位

将 MOSI 切换为 MISO，同时将 MISO 切换为 MOSI。

1 = 开启切换

0 = 开关已关闭

MSPD[1:0] — SPI 主采样点延迟

指定延迟的用于 SPI_SCK 采样边沿的系统时钟周期数

0x0 = 无延迟;

0x1 = 1 周期延迟;

0x2 = 2 周期延迟;

0x3 = 2 周期延迟;

PDPSP — SPI 端口下拉使能位

1 = 为 SPIPORT 位[3:0] 启用下拉器件

0 = 对 SPIPORT 位[3:0] 禁用下拉器件

PUPSP — SPI 端口上拉使能位

1 = 启用 SPIPORT 位[3:0] 的上拉器件

0 = 禁用 SPIPORT 位[3:0] 的上拉器件

23.6.11. SPI 在 SCK 延迟之后的寄存器

偏移地址：0x000A

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	ASCDE	PASCD6	PASCD5	PASCD4	0	ASCD2	ASCD1	ASCD0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 23-12：SPI SCK 延迟寄存器（SPIASCDR）

Read：任何时候

Write：任何时候

ASCDE — SPI_SCK 延迟使能后

1 = 启用 SPI_SCK 延迟后

0 = 禁用 SPI_SCK 延迟

PASCD[6:4] — SPI 在 SCK 延迟后预选位

PASCD[6:4] 和 ASCD[2:0] 位选择 SCK 延迟除数后的 SPI。

ASCD[2:0] — SPI SCK 延迟位

ASCD = (PASCD[6:4]+1) *2 (ASCD[2:0]+1) ; t = 0.5*SCK+ASCD

23.6.12. SPI 在 SCK 延迟寄存器之前

偏移地址：0x000B

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BSCDE	PBSCD6	PBSCD5	PBSCD4	0	BSCD2	BSCD1	BSCD0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 23-13：SPI 在 SCK 延迟寄存器之前（SPIBSCDR）

Write：任何时候

BSCDE — 在 SPI_SCK 延迟使能之前

1 = 在启用 SPI_SCK 延迟之前

0 = 禁用 SPI_SCK 延迟

PBSCD[6:4] — SPI 在 SCK 延迟前的预选位

PBSCD[6:4] 和 BSCD[2:0] 位选择 SCK 延迟分频器之前的 SPI。

BSCD[2:0] — SPI 在 SCK 延迟位之前

BSCD = (PBSCD[6:4]+1) *2 (BSCD[2:0]+1) ; tL = 0.5*SCK+BSCD

23.6.13. SPI 端口数据寄存器

偏移地址：0x000C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RSVD7	RSVD6	RSVD5	RSVD4	PORTSP3	PORTSP2	PORTSP1	PORTSP0
RESET:	0	0	0	0	0	0	0	0
Pin function:					SS	SCK	MOSI/ MOMI	MISO/ SISO

图 23-14：SPI 端口数据寄存器（SPIPORT）

Read：任何时候

Write：任何时候

RSVD[7:4] — 保留

向这些读/写位写入数据会更新其值，但不会影响功能。

PORTSP[3:0] — SPI 端口数据位

仅当引脚被配置为通用输出时，数据才会写入 SPIPORT 驱动器。

读取输入(DDRSP 位清除）返回引脚电平；读取输出(DDRSP 位设置）返回引脚驱动器输入电平。

当引脚配置为 SPI 输出时，向任何 PORTSP[3:0] 引脚写入数据都不会改变引脚状态。

SPIPORT I/O 功能取决于 SPICR1 中的 SPE 位的状态以及 SPIDDR 中的 DDRSP 位的状态。

23.6.14. SPI 传输计数器寄存器

偏移地址：0x000D ~ 0x000F

	Bit 31	30	29	28	27	26	25	Bit 24
Read: Write:								
RESET:	0	0	0	0	0	0	0	0
	Bit 23	22	21	20	19	18	17	Bit 16
Read: Write:								
RESET:	0	0	0	0	0	0	0	0
	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:								
RESET:	0	0	0	0	0	0	0	0

图 23-15：SPI 发送计数器寄存器（SPITCNT）

Read：任何时候

Write：任何时候

SPITCNT[31:8] — SPI 发送计数器寄存器

当 SPIFR 中的 CONT 位被置为有效时，以主模式发送帧会使计数器减少 1。一旦计数器被减至 0 并发送了另一个帧，引脚 SPI_CS#将被设置为高电平，计数器值将重新加载。

23.6.15. SPI 数据寄存器

偏移地址：0x0010

	Bit 15	14	13	12	11	10	9	Bit 8
Read: Write:	15	14	13	12	11	10	9	8
RESET:	0	0	0	0	0	0	0	0
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	7	6	5	4	3	2	1	0
RESET:	0	0	0	0	0	0	0	0

图 23-16：SPI 数据寄存器（SPIDR）

Read：任何时候；

写作：任何时候；

SPIDR 是 SPI 数据的输入和输出寄存器。向 SPIDR 写入数据将填满 TX FIFO，而从 SPIDR 读取数据将清空 RX FIFO。

23.6.16. SPI 状态寄存器

偏移地址：0x0014

	Bit 15	14	13	12	11	10	9	Bit 8
Read:	TXFTO	TXFOVF	TXFUDF	TXFSER	RXFTO	RXFOVF	RXFUDF	RXFSER
Write:	w1c	w1c	w1c		w1c	w1c	w1c	
RESET:	0	0	0	0	0	0	0	0
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SPIF	FLOST	EOTF	MODF	TXFFULL	TXFEMP	RXFFULL	RXFEMP
Write:		w1c	w1c					
RESET:	0	0	0	0	0	1	0	1

= Writes have no effect and the access terminates without a transfer error exception.

图 23-17：SPI 状态寄存器（SPISR）

Read：任何时候

Write：w1c 表示将 1 写入该位将清除相应的标志

SPIF — SPI 完成标志

每次传输完成后都会设置 SPIF 标志。通过读取 SPIF 清除该标志，然后访问 SPIDR。

1 = 单次传输已完成

0 = 单个传输尚未完成或没有传输

EOTF — 传输结束标志

当 TX FIFO 中的所有数据传输完毕时，EOTF 标志被设置。

1 = 传输结束

0 = 传输未结束或未向位写入 1，或向 SPIDR 写入 1 将清除此标志

FLOST — 框架丢失

当 SPI 处于从机模式且 TX FIFO 中没有有效数据时，如果此时主设备启动传输，则 SPI 将把最后接收的数据返回给主设备并设置 FLOST。如果 FLOSTIE 位也设置，FLOST 将生成中断请求。

1 = 发生帧丢失

0 = 无帧丢失

向位写入 1 将清除此标志

MODF — 模式故障标志

当主 SPI 的 SPI_CS#引脚被置为低电平且该引脚配置为模式错误输入时，MODF 标志会被设置。若同时设置了 SPIE 位，MODF 会触发中断请求。模式错误将清除 SPE、MSTR 和 DDRSP[2:0] 位。通过读取 MODF 置位的 SPISR 寄存器并写入 SPICR1 寄存器可清除 MODF。复位操作也能清除 MODF。

1 = 模式故障

0 = 无模式故障

TXFFULL — TX FIFO 满标志

1 = TX FIFO 满

0 = TX FIFO 未满

TXFEMP — TX FIFO 空标志

1 = TX FIFO 空

0 = TX FIFO 未满

RXFFULL — RX FIFO 满标志

1 = TX FIFO 满

0 = TX FIFO 未满

RXFEMP — RX FIFO 空标志

1 = RX FIFO 空

0 = RX FIFO 未空

TXFTO — TX FIFO 超时

1 = TX FIFO 超时已发生

0 = TX FIFO TimeOut 未发生

向该位写入 1 或向 SPIDR 写入将清除此标志

TXFOVF — TX FIFO 溢出标志

1 = TX FIFO 溢出已发生

0 = 未发生 TX FIFO 溢出

TXFUDF — TX FIFO 下溢标志

1 = TX FIFO 发生下溢

0 = TX FIFO 未发生下溢

TXFSER — TX FIFO 服务标志

1 = TX FIFO 数据编号低于或等于 TXFSTH

0 = TX FIFO 数据编号高于 TXFSTH

RXFTO — RX FIFO 超时

1 = RX FIFO 超时已发生

0 = RX FIFO TimeOut 未发生

向位写入 1 或从 SPIDR 读取将清除此标志

RXFOVF — RX FIFO 溢出标志

1 = RX FIFO 溢出发生

0 = 未发生 RX FIFO 溢出

RXFUDF — RX FIFO 下溢标志

1 = RX FIFO 发生下溢

0 = RX FIFO 下溢未发生

RXFSER — RX FIFO 服务标志

1 = RX FIFO 数据编号高于 RXFSTH

0 = RX FIFO 数据编号低于或等于 RXFST

23.6.17. SPI RX FIFO 状态寄存器

偏移地址：0x0016

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	RXNXTP2	RXNXTP1	RXNXTP0	RXFCTR3	RXFCTR2	RXFCTR1	RXFCTR0
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 23-18：SPI RX FIFO 状态寄存器（SPIRXFSR）

Read：任何时候

Write：无意义或无影响

RXNXTP[2:0] — RX 下一个指针

指示 RX FIFO 指针，以指向要读取的数据。

RXFCTR[3:0] — RX FIFO 计数器

表示 RX FIFO 数据计数器。

23.6.18. SPI TX FIFO 状态寄存器

偏移地址：0x0017

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	TXNXTP2	TXNXTP1	TXNXTP0	TXFCTR3	TXFCTR2	TXFCTR1	TXFCTR0
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 23-19：SPI TX FIFO 状态寄存器（SPITXFSR）

Read：任何时候

Write：无意义或无影响

TXNXTP[2:0] — TX 下指针

指示 TX FIFO 指针，以指向要发送的数据。

TXFCTR[3:0] — TX FIFO 计数器

表示 TX FIFO 数据计数器。

23.6.19. SPI DMA 控制寄存器

偏移地址：0x0018

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	TXDMAE	RXDMAE
Write:								
RESET:	0	0	0	0	0	0	0	0

图 23-20：SPI DMA 控制寄存器（SPIDMACR）

Read：任何时候；

Write：任何时候；

TXDMAE — TX FIFO DMA 请求启用

1 = TX DMA 请求已启用

0 = TX DMA 请求已禁用

RXDMAE — RX FIFO DMA 请求使能

1 = RX DMA 请求已启用

0 = 禁用 RX DMA 请求

23.6.20. SPI DMA 阈值寄存器

偏移地址：0x0019

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	TXDMATH	TXDMATH	TXDMATH	0	RXDMATH	RXDMATH	RXDMATH
Write:		2	1	0		2	1	0
RESET:	0	0	0	0	0	0	0	0

图 23-21：SPI DMA 阈值寄存器（SPIDMATHR）

Read：任何时候；

Write：任何时候；

TXDMATH[2:0] — TX DMA 阈值

指定 TX FIFO 的数据数阈值，一旦 TX FIFO 中的数据数小于该阈值并且 SPIDMACR 中设置了位 TXDMAE，SPI 将向 DMA 发送 TX DMA 请求。

数据编号 = TXFDMATH[2:0]

RXDMATH[2:0] — RX DMA 阈值

指定 RX FIFO 的数据数阈值，一旦 RX FIFO 中的数据数超过该阈值并且 SPIDMACR 中设置了位 RXDMAE，SPI 就会向 DMA 发送 RX DMA 请求。

数据编号 = RXFDMATH[2:0]+1

23.6.21. SPI FIFO 调试控制寄存器

偏移地址：0x001A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	TXFIDX2	TXFIDX1	TXFIDX0	0	RXFIDX2	RXFIDX1	RXFIDX0
Write:								
RESET:	0	0	0	0	0	0	0	0

图 23-22：SPI FIFO 调试控制寄存器（SPIFDCR）

Read：任何时候；

Write：任何时候；

TXFIDX[2:0] — TX FIFO 索引

指定要从 TX FIFO 读取到 SPITXFDBGR 的数据索引。

RXFIDX[2:0] — RX FIFO 索引

指定要从 RX FIFO 读取到 SPIRXFDBGR 的数据索引。

23.6.22. SPI 中断控制寄存器

偏移地址：0x001B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	FLOSTIE	0	MODFIE	0	0	0	0
Write:								
RESET:	0	0	0	0	0	0	0	0

图 23-23：SPI 中断控制寄存器（SPIICR）

Read：任何时候；

Write：任何时候；

MODF — MODF 中断使能。

1 = MODF 中断启用

0 = MODF 中断已禁用

FLOSTIE-FLOST 中断使能。

1 = FLOST 中断启用

0 = FLOST 中断关闭

23.6.23. SPI RX FIFO 调试寄存器

偏移地址：0x001C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	7	6	5	4	3	2	1	0
Write:								
RESET:	0	0	0	0	0	0	0	0

图 23-24：SPI RX FIFO 调试寄存器（SPIRXFDBGR）

Read：任何时候；

Write：无影响；

SPIRXFDBGR 提供对 RX FIFO 的可见性，以便进行调试。该寄存器是只读的，不能修改。读取 SPITXRDBGR 不会改变 RX FIFO 的状态。

23.6.24. SPI TX FIFO 调试寄存器

偏移地址：0x001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	7	6	5	4	3	2	1	0
Write:								
RESET:	0	0	0	0	0	0	0	0

图 23-25：SPI TX FIFO 调试寄存器（SPITXFDBGR）

Read：任何时候；

Write：无影响；

SPITXFDBGR 为调试目的提供了对 TX FIFO 的可见性。该寄存器是只读的，不能修改。读取 SPITXFDBGR 不会改变 TX FIFO 的状态。

23.7.功能说明

SPI 模块允许 MCU 和外围设备之间进行全双工、同步、串行通信，软件可以轮询 SPI 状态标志，也可以通过中断驱动 SPI 操作。

在 SPICR1 中设置 SPE 位，即可启用 SPI，并将四个 SPI 端口引脚专门用于 SPI 功能：

- 选线（SPI_CS#）
- 串行时钟（SPI_SCK）
- 掌握/成为（MOSI）的下属
- 主/从机输出（MISO）

当 SPE 位为清零时，SPI_CS#、SPI_SCK、MOSI 和 MISO 引脚是通用 I/O 引脚，由 SPIDDR 控制。

主 SPI 设备中的移位寄存器通过 MOSI 和 MISO 引脚与从设备的移位寄存器相连，这两个寄存器共同构成分布式寄存器。在 SPI 传输过程中，主设备的 SPI_SCK 时钟信号负责数据传输，主从双方进行数据交换。写入主设备 SPIDR 寄存器的数据即作为输出数据发送至从设备，而交换完成后，从设备读取的 SPIDR 寄存器数据则作为输入数据返回。

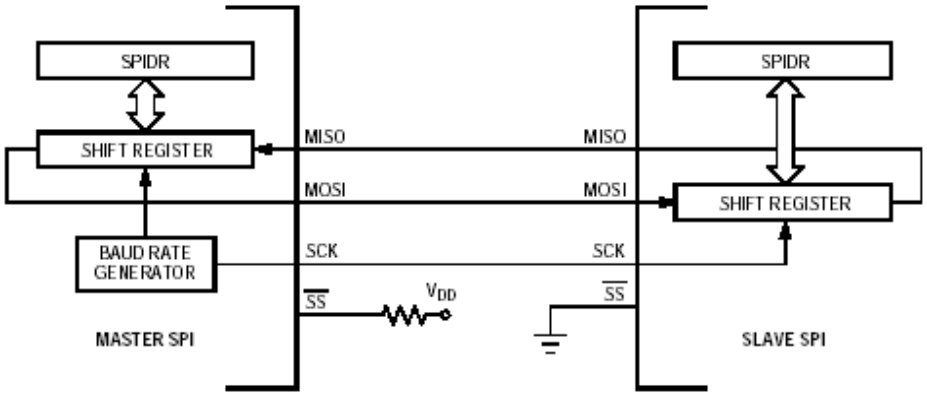


图 23-26：全双工操作

23.7.1. 主模式

在 SPICR1 寄存器中设置 MSTR 位后，SPI 将进入主控模式。只有处于主控模式的 SPI 才能发起传输操作。向主控 SPI 寄存器写入数据即可启动传输过程。当移位寄存器为空时，系统会将字节传输至该寄存器，并在主控 SPI 时钟 SPI_SCK 的控制下，通过 MOSI 引脚开始移位输出。需要特别注意的是，SPI_SCK 时钟会在完成 SPI 寄存器写入操作后半个时钟周期自动启动。

SPIBR 中的 SPR[2:0] 和 SPPR[6:4] 位控制波特率发生器并确定移位寄存器的速率，SPI_SCK 引脚是 SPI 时钟输出，通过 SPI_SCK 引脚，主设备的波特率发生器控制从设备的移位寄存器。

SPICR1 中的 MSTR 位和 SPICR2 中的 SPC0 位控制数据引脚 MOSI 和 MISO 的功能。

SPI_CS#引脚通常是一个保持高电平非活动状态的输入端。在 SPIDDR 中设置 DDRSP3 位可将 SPI_CS#配置为输出引脚。通过 SPICR1 中的 DDRSP3 位和 SSOE 位，可将 SPI_CS#配置为通用 I/O、模式故障检测或从属选择功能。详见表 23-3。

在每次传输过程中，SPI_CS#信号输出端会处于低电平状态，而当 SPI 处于空闲状态时则保持高电平。若主设备将 SPI_CS#输入端置为低电平，则会在 SPISR 寄存器中设置 MODF 标志位，表明存在模式错误。当多个主设备同时尝试驱动 MOSI 和 SPI_SCK 线路时，系统会触发模式错误。此时，MISO、MOSI（或 MOMI)以及 SPI_SCK 引脚的数据方向位会被清零并设为输入状态，同时 SPICR1 寄存器中的 SPE 和 MSTR 位也会被清除。若 SPIE 位同时被置位，MODF 标志位将触发中断请求。

23.7.2. 从属模式

清除 SPICR1 中的 MSTR 位将 SPI 置于从机模式。SPI_SCK 引脚是 SPI 时钟输入端，来自主设备，SPI_CS#引脚是从机选择输入端。要进行传输，必须将 SPI_CS#引脚驱动为低电平并保持低电平直至传输完成。

SPICR2 中的 MSTR 位和 SPC0 位控制数据引脚 MOSI 和 MISO 的功能。SPI_CS#输入也控制 MISO 引脚。如果 SPI_CS#为低电平，移位寄存器的最高有效位将通过 MISO 引脚输出。如果 SPI_CS#为高电平，MISO 引脚处于高阻抗状态，从设备将忽略 SPI_SCK 输入。

提示：当使用具有全双工功能的外围设备时，请勿同时启用两个驱动相同 MISO 输出线的接收器。

只要只有一个从子驱动主输入线，多个从子就可以同时接收相同的传输。

如果 SPICR1 中的 CPHA 位为清除状态，则 SPI_SCK 的奇数边沿会锁存 MOSI 引脚上的数据。偶数边沿则会将数据移入 SPI 移位寄存器的 LSB 位置，并将 MSB 移出至 MISO 引脚。

如果 CPHA 位被设置，SPI_SCK 的偶数边沿会锁存 MOSI 引脚上的数据。奇数边沿则会将数据移入 SPI 移位寄存器的最低有效位，并将最高有效位移出至 MISO 引脚。

在第八次轮换后，传输完成。接收的数据传输到 SPIDR，并在 SPISR 中设置 SPIF 标志。

23.7.3. FIFO 操作

当数据大小小于 16 或 8 位时，用户必须对写入发送 FIFO 的数据进行右对齐。发送逻辑将忽略未使用的比特。接收缓冲器中小于 16 或 8 位的接收数据会自动进行右对齐。

23.7.3.1. 传输 FIFO

通用传输 FIFO 是一个 8 位宽、8 位深的先进先出存储器缓冲区，CPU 通过 SPI 数据（SPIDR）寄存器将数据写入 FIFO，数据在 FIFO 中存储，直到由传输逻辑读取。

当 SPI 总线配置为主设备或从设备时，数据会先以并行方式存入发送 FIFO，随后通过 SPI 发送引脚进行串行转换传输。在从设备模式下，每当主设备发起事务时，SPI 就会立即发送数据。若发送 FIFO 为空且主设备发起事务，从设备会以最后发送的方式传输接收到的数据。需注意确保 FIFO 中存有有效数据。当 FIFO 数据量低于预设阈值时，SPI 可配置为触发中断或 DMA 请求。

23.7.3.2. 接收 FIFO

公共接收 FIFO 是一个 8 位宽、8 位深的先进先出存储缓冲器。从串行接口接收的数据存储在该缓冲器中，直到 CPU 读取为止，CPU 通过读取 SPIDR 寄存器访问读取 FIFO。

当配置为主设备或从设备时，通过 SPI Rx 引脚接收的串行数据在并行加载到连接的从设备或主设备接收 FIFO 之前被注册。

23.7.4. 传输格式

SPICR1 中的 CPHA 和 CPOL 位选择四个串行时钟相位和极性组合中的一个。主 SPI 设备和通信从设备的时钟相位和极性必须相同。

23.7.4.1. CPHA = 1 时的传输格式

某些外设需要先出现第一个 SPI_SCK 时钟边沿，才能使从设备的 MISO 引脚获得最高有效位（MSB）。当 CPHA 位被置位后，主设备 SPI 会等待半个 SPI_SCK 时钟周期的同步延迟。随后在传输开始时发出首个 SPI_SCK 时钟边沿，该边沿促使从设备将 MSB 传输至主设备的 MISO 引脚。第二个边沿及后续偶数编号的边沿用于锁存数据，第三个边沿及后续奇数编号的边沿则将锁存的数据移入主设备移位寄存器，并通过 MOSI 引脚输出主设备数据。

第 16 次和最后一次 SPI_SCK 边缘之后：

- 主 SPIDR 寄存器中的数据位于从 SPIDR 中。
从属 SPIDR 寄存器中的数据位于主 SPIDR 中。

- SPI_SCK 时钟停止，SPISR 中的 SPIF 标志被设置，表示传输完成。如果 SPCR1 中的 SPIE 位被

设置，SPIF 将生成中断请求。

图 23-27 显示了 CPHA 位被设置时的传输时间。主设备的 SPI_CS#引脚必须是高电平或配置为不影响 SPI 的通用输出。

当 CPHA = 1 时，从属 SPI_CS#线可以在字节之间保持低电平。这种格式适用于只有一个主设备和一个从设备驱动 MISO 数据线的系统。

SPIF 中断请求出现在每个传输结束时。

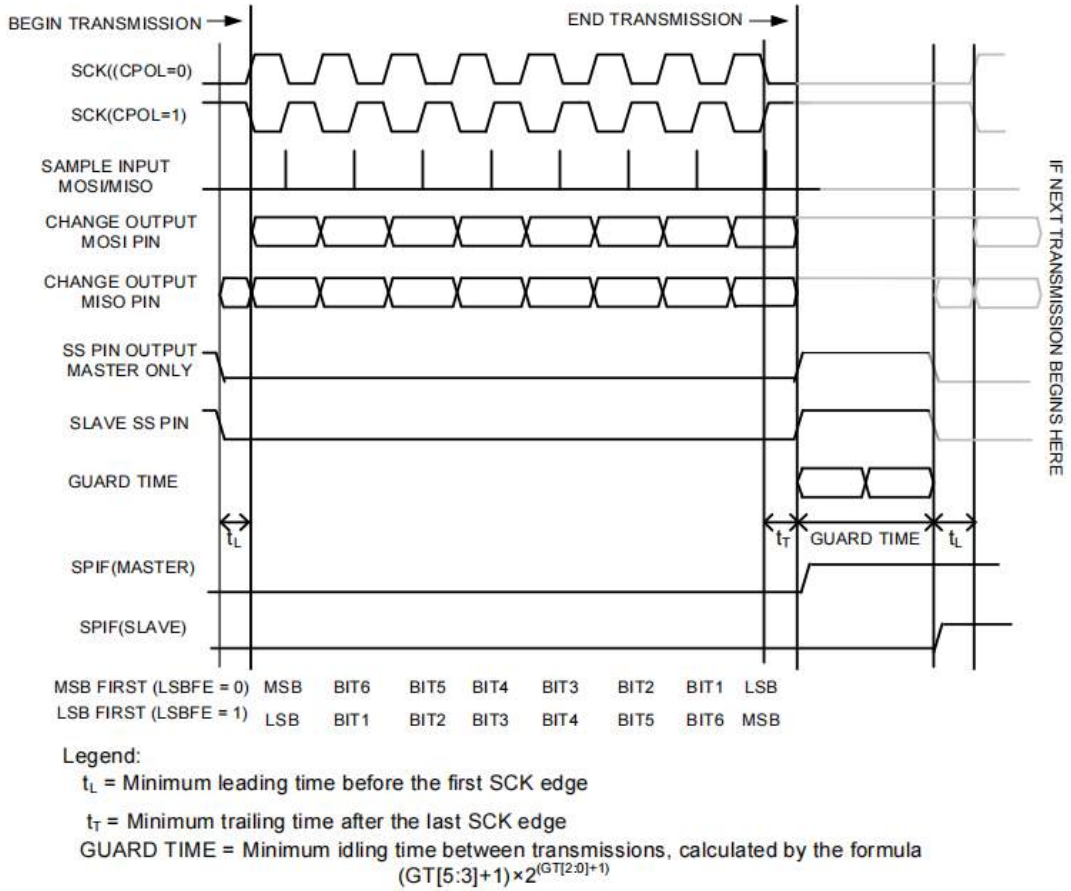


图 23-27：SPI 时钟格式 1（CPHA = 1）

23.7.4.2. CPHA = 0 时的传输格式

在某些外设中，从设备的最高有效位（MSB）会在被选中后立即通过其 MISO 引脚输出。当 CPHA 位清零时，主设备 SPI 会在传输开始后延迟半个 SPI_SCK 周期才触发首个 SPI_SCK 边沿。首个边沿及后续所有奇数边沿会锁存从设备数据，而偶数边沿则将从设备数据移入主设备移位寄存器，并通过主设备 MOSI 引脚输出主设备数据。

第 16 次和最后一次 SPI_SCK 边缘之后：

- 主 SPIDR 中的数据在从 SPIDR 中，从 SPIDR 中的数据在主 SPIDR 中。
- SPI_SCK 时钟停止，SPISR 中的 SPIF 标志被设置，表示传输完成。如果 SPCR1 中的 SPIE 位被设置，SPIF 将生成中断请求。

图 23-28 显示了 CPHA 位为清零时的传输时间。主设备的 SPI_CS#引脚必须为高电平或配置为不影响 SPI 的通用输出。

当 CPHA = 0 时，从属 SPI_CS#引脚必须在字节之间被否定和重新断言。

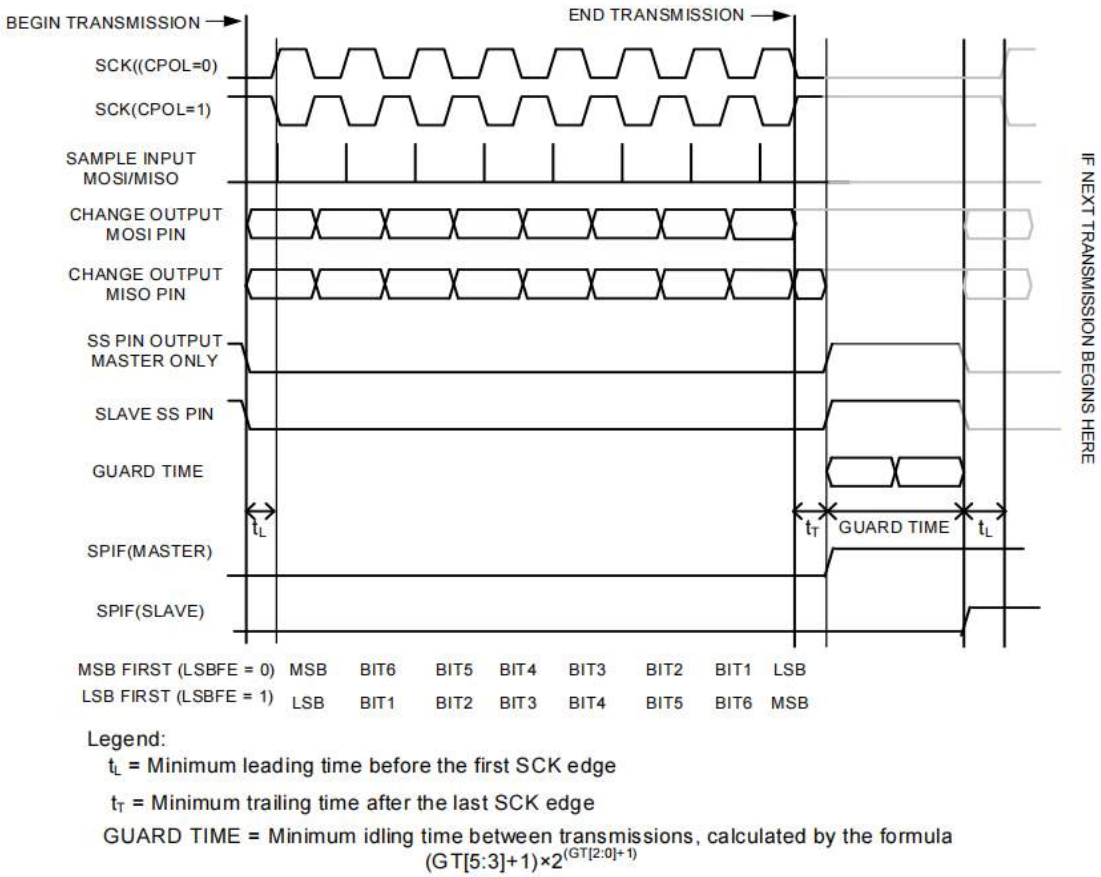


图 23-28：SPI 时钟格式 0（CPHA = 0）

提示：当出现以下情况时，主从时钟之间的时间偏差会导致数据丢失：

- CPHA = 0，且，
- 波特率是 SPI 时钟除以二，以及
- 主 SPI_SCK 频率是从 SPI 时钟频率的一半，且
- 软件在同步 SPI_CS#信号变低之前写入从机 SPIDR。

同步 SPI_CS#信号与 SPI 时钟保持同步。图 23-29 展示了一个典型示例：此时从设备的同步 SPI_CS#信号几乎滞后完整 SPI 时钟周期。当从设备的同步 SPI_CS#信号处于高电平时，即使 SPI_CS#引脚已处于低电平状态，仍可进行写入操作。这种写入操作可能在主设备采样 MISO 线时改变 MISO 引脚的状态。由于主设备采样时传输的第一位数据可能尚未稳定，因此发送给主设备的字节可能会出现损坏。

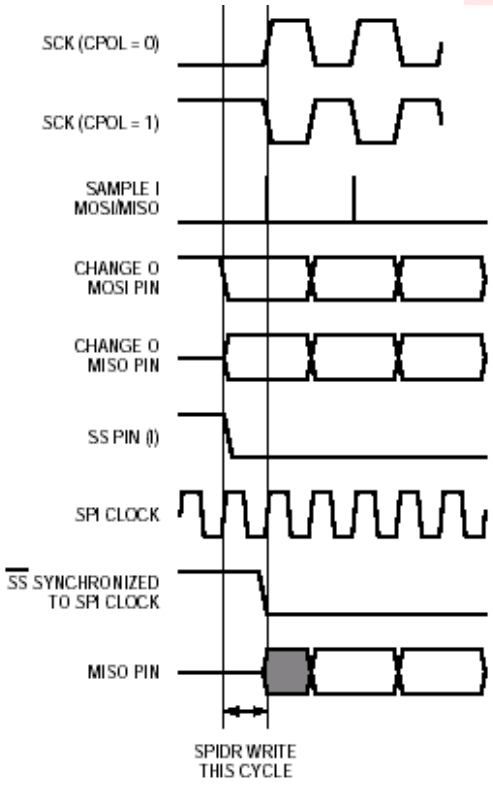


图 23-29：主时钟/从时钟偏移导致的传输错误

此外，如果从设备产生一个延迟写入，则其状态机可能没有时间重置，导致它错误地从主设备接收一个字节。

当 SPI_SCK 频率为从设备 SPI 时钟频率的一半时，最可能出现此错误。在其他波特率下，SPI_SCK 偏移量不会超过一个 SPI 时钟周期，同步 SPI_CS#信号与首个 SPI_SCK 边沿之间的时间间隔也会相应增加。例如当 SPI_SCK 频率为从设备 SPI 时钟频率的四分之一时，在 SPI_CS#信号下降沿与 SPI_SCK 边沿之间会存在两个

SPI 时钟周期的间隔。

只要没有发生另一个 SPIDR 写入操作，从属设备之间的下列字节传输都是正确的。

23.7.4.3. 德州仪器同步串行帧格式

在此工作模式下，SPI_SCK 和 SPI_CS#信号被强制置低电平，当 SPI 处于空闲状态时，发送数据线 SPI Tx 始终处于三态。当发送 FIFO 的底部条目存入数据后，SPI_CS#信号会在一个 SPI_SCK 周期内被置高电平。待传输的数据值也会从发送 FIFO 传送至发送逻辑的串行移位寄存器。在 SSIClk 信号的下一个上升沿，4 到 16 位数据帧的最高有效位会通过 SSITx 引脚输出。同样地，接收端数据的最高有效位也会由片外串行从设备传送至 SSIRx 引脚。

SPI 和片外串行从设备随后在 SPI_SCK 的每个下降沿将每个数据位时钟输入到各自的串行移位器中。在锁存最低有效位之后，接收的数据在 SPI_SCK 的第一个上升沿从串行移位器传输到接收 FIFO 中。

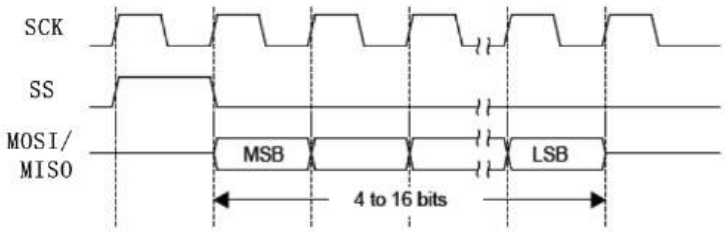


图 23-30：TI 单次转移

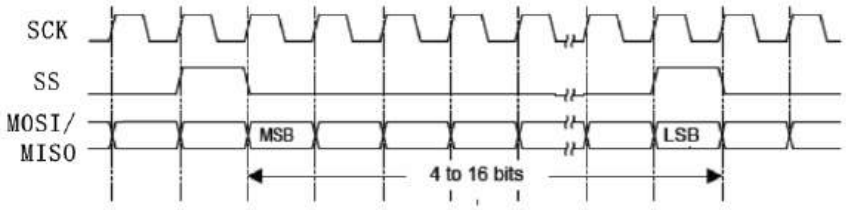


图 23-31：TI 连续转移

23.7.5. SPI 波特率生成

波特率发生器将 SPI 时钟进行分频，从而产生 SPI 波特率时钟。SPIBR 中的 SPPR[6:4] 和 SPR[2:0] 位选择 SPI 时钟分频比：

$$\text{SPI 时钟分频系数} = (SPPR + 1) \times 2^{(SPR + 1)}$$

此时

SPPR = 写入 SPPR[6:4] 位的数值

SPR = 写入到位 SPR[2:0] 的值

波特率发生器仅在 SPI 处于主模式和发送时激活。否则，分频器将处于非活动状态以减少 IDD 电流。

23.7.6. 从属选择输出

在传输期间，slave-select 输出功能自动将 SPI_CS#引脚驱动为低电平以选择外部设备，并在空闲期间将其驱动为高电平以取消选择外部设备。当 SPI_CS#输出被选择时，SPI_CS#输出引脚连接到外部设备的 CS#输入引脚。

仅在主模式下，将 SSOE 位设置为 SPICR1 中的位，并将 DDRSP[3] 位设置为 SPIDDR 中的位，可将 SPI_CS#引脚配置为从属选择输出。将 SSOE 位设置为 0 可禁用模式故障功能。

提示：在多主控系统中使用从属选择输出功能时要小心。模式故障功能不可用于检测主控之间的系统错误。

23.7.7. 双向模式

在 SPICR1 寄存器中设置 SPC0 位，即可启用双向传输模式（参见表 23-6)。该芯片仅通过单个数据引脚与外部设备通信，具体使用哪个引脚由 MSTR 位决定。当处于主控模式时，MOSI 引脚作为主控输出/主控输入引脚（MOSI）；切换为从属模式后，MISO 引脚则作为从属输出/从属输入引脚（MISO）。特别说明的是，主控模式下的 MISO 引脚和从属模式下的 MOSI 引脚，都是通用型输入输出引脚。

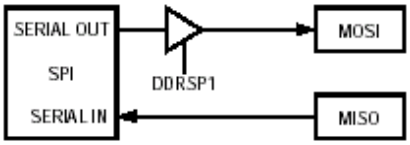
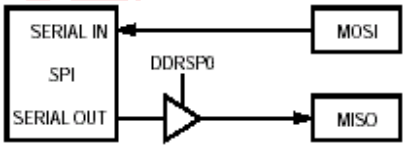
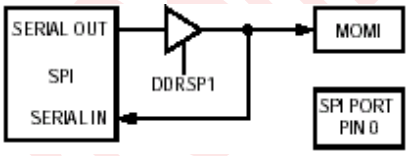
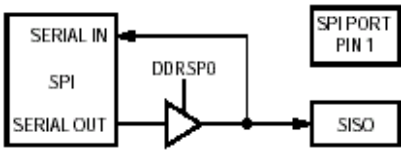
每个数据 I/O 引脚的方向取决于其数据方向寄存器位。配置为输出的引脚是移位寄存器的输出，配置为输入的引脚是移位寄存器的输入，并且移位寄存器输出的数据被丢弃。

SPI_SCK 引脚在主模式下是输出，在从属模式下是输入。

在主模式下，SPI_CS#引脚可以是输入或输出，在从模式下，它始终是输入。

在双向模式下，模式故障不会清除 DDRSP0，即 SISO 引脚的数据方向位。

表 23-7：正常模式和双向模式

SPE = 1	主模式，MSTR = 1	从机模式，MSTR = 0
Normal Mode SPC0 = 0	 SWOM 支持开放式排水输出。	 SWOM 支持开放式排水输出。
Bidirectional Mode SPC0 = 1	 SWOM 支持开放式排水输出。SPI 端口引脚 0 是通用 I/O。	 SWOM 支持开放式排水输出。SPI 端口引脚 1 是通用 I/O。

23.7.8. DMA 操作

SPI 外设提供与 DMA 控制器的接口，设有独立的发送和接收通道。通过 SPI DMA 控制寄存器（SPIDMACR）可启用 SPI 的 DMA 功能。当 DMA 功能启用时，若对应的 FIFO 满足传输需求，SPI 会在接收或发送通道触发 DMA 请求。对于接收通道，当接收 FIFO 的数据量达到或超过 SPIDMACR 设定的阈值，或 RXF 超时计数器（由 SPIRXFTOCTR 设置）归零时，系统会触发 DMA 请求。对于发送通道，当发送 FIFO 的数据量低于或等于 SPIDMACR 设定的阈值，或 TXF 超时计数器（由 SPITXFTOCTR 设置）归零时，系统会触发单次传输请求。DMA 控制器将根据通道配置自动处理这些请求。要启用接收通道的 DMA 功能，需设置 SPIDMACR 寄存器中的 RXDMAE 位；若要启用发送通道的 DMA 功能，则需设置 SPIDMACR 寄存器中的 TXDMAE 位。

23.7.9. 高速模式

在高速模式下，主样本延迟和从样本在 SPI_SCK 周期中的输出时间都比正常 SPI 模式更早，以允许设备引脚和板迹的延迟。随着波特率的增加，SPI_SCK 周期缩短，这些延迟在 SPI_SCK 周期中所占的比例会变得更为显著。

对于主设备，当配置为主设备时，SPIURD 寄存器中的 HS 位启用高速模式，并且 SPIPURD 寄存器中的 MSPD[1:0] 设置采样点延迟。

对于从机，当被配置为从机时，SPIURD 寄存器中的 HS 位启用高速模式。如果设置了 HS，则 SPI 从机会在移入数据边沿时输出数据，而移入数据的定时与正常模式相同。

要使高速模式正常运行，必须彻底分析 SPI 链路的时序预算。

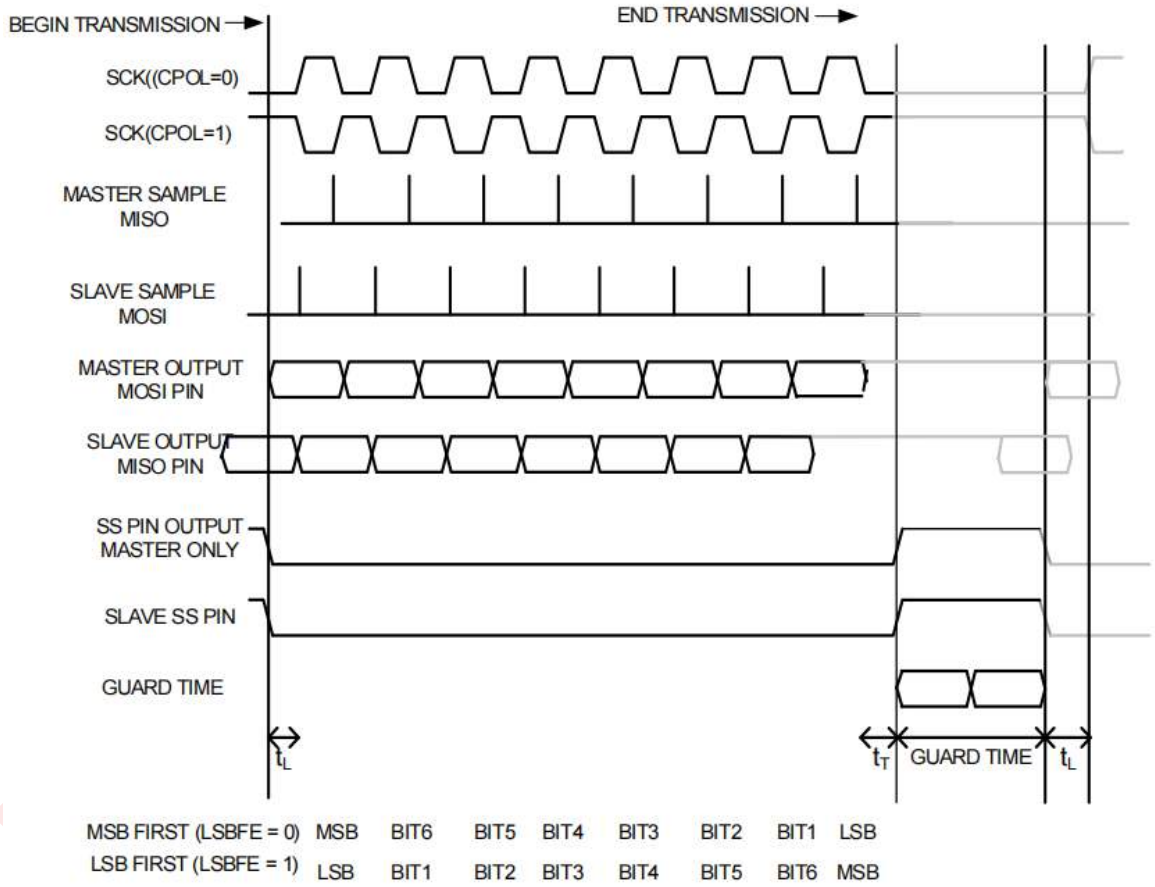


图 23-32：高速模式（CPHA = 0）

23.7.10. 低功耗模式选项

本小节描述低功耗模式选项。

23.7.10.1. 运行模式

清除 SPICR1 中的 SPE 位会使 SPI 进入禁用、低功耗状态。此时 SPI 寄存器可访问，但 SPI 时钟被禁用。

23.7.10.2. Doze 模式

十二进制模式下的 SPI 操作取决于 SPICR2 中 SPISDOZ 位的状态。

- 如果 SPISDOZ 为“清楚”，则 SPI 在 doze 模式下正常运行。
- 如果设置了 SPISDOZ，则 SPI 时钟停止，SPI 进入低功耗模式的 doze 模式。
  - 正在进行的任何主传输都会在进入 Doze 模式时停止，并在退出 Doze 模式时恢复。
  - 如果主设备持续驱动从设备的 SPI_SCK 引脚，则正在进行的任何从设备传输都会继续。从设备将保持与主设备 SPI_SCK 时钟同步。

提示：从属移位寄存器可以接收 MOSI 数据，但不能将数据传送到 SPIDR 或在 doze 或 stop 模式下设置 SPIF 标志。如果从属设备在空闲状态下进入 doze 模式并在空闲状态下退出 doze 模式，则 SPIF 保持为清除状态且不会向 SPIDR 传输数据。

23.7.10.3. 停止模式

在停止模式下，SPI 操作与 doze 模式下的操作相同，但 SPISDOZ 位被设置。

23.8. SPI 重置

重置将 SPI 寄存器初始化为 23.6 存储器映射和寄存器中描述的已知启动状态。从重置之后到写入 SPIDR 寄存器之前，从设备的传输是不确定的，或者是在重置之前从主设备接收的最后一个字节。重置之后读取 SPIDR 返回 0。

23.9. SPI 中断

表 23-8：SPI 中断请求源

中断请求	旗帜	启用位
Mode Fault	MODF	MODFIE
Transmission Complete	EOTF	SPIE
Frame Lost	FLOST	FLOSTIE
TXFIFO Timeout	TXFTO	TXFTOIE
TXFIFO Overflow	TXFOVF	TXFOVIE
TXFIFO Underflow	TXFUDF	TXFUDIE
TXFIFO Service	TXFSER	TXFSTHIE
RXFIFO Timeout	RXFTO	RXFTOIE
RXFIFO Overflow	RXFOVF	RXFOVIE
RXFIFO Underflow	RXFUDF	RXFUDIE
RXFIFO Service	RXFSER	RXFSTHIE

23.9.1. 模式故障（MODF）标志

当主 SPI 的 SPI_CS#引脚被置为低电平且该引脚被配置为模式错误输入时，MODF 将被置位。如果 MODFIE 位也被置位，MODF 将生成中断请求。模式错误会清除 SPE、MSTR 和 DDRSP[2:0] 位。清除 MODF 的方法是：先读取被 MODF 置位的 SPISR 寄存器，然后写入 SPICR1 寄存器。

23.9.2. EOT 中断标志（EOTF）

当 TX FIFO 中的所有数据传输完毕时，EOTF 被设置。如果 SPIE 位也被设置，EOTF 将生成中断请求。通过向该位写入 1 来清除 EOTF，或者通过 CPU 或 DMA 填充 TX FIFO。复位将清除 EOTF。

23.9.3. 帧丢失中断标志（FLOST）

当 SPI 处于从机模式且 TX FIFO 中没有有效数据时，如果此时主设备启动传输，则 SPI 将把最后接收的数据返回给主设备并设置 FLOST。如果 FLOSTIE 位也设置，则 FLOST 生成中断请求。通过向该位写入 1 来清除 FLOST。复位将清除 EOTF。

23.9.4. TXFIFO 超时中断标志（TXFTO）

当 TX FIFO 不为空时，计数器处于开启状态。如果在计数器倒计时至 0 之前 TX FIFO 中没有操作，则会设置 TXFTO。计数器的计数周期为 SPI_SCK 周期。通过向该位写入 1 来清除 TXFTO。

23.9.5. TXFIFO 溢出中断标志（TXFOVF）

当试图进行数据写入操作，导致 TX FIFO 的数据数量大于 8 时，TXFOVF 被设置。通过向该位写入 1 来清除 TXFOVF。

23.9.6. TXFIFO 下溢中断标志（TXFUDF）

当试图进行数据写入操作，导致 TX FIFO 的数据编号小于 0 时，TXFUDF 被设置。通过向该位写入 1 来清除 TXFUDF。

23.9.7. TXFIFO 服务中断标志（TXFSER）

当 TX FIFO 中的数据数量低于或等于 SPITXFCR 中指定的阈值时，将设置 TXFER。

23.9.8. RXFIFO 超时中断标志（RXFTO）

当 RX FIFO 不为空时，计数器处于开启状态。如果在计数器倒计数至 0 之前 RX FIFO 没有操作，则会设置 RXFTO。计数器以 SPI_SCK 周期进行计数。通过向该位写入 1 来清除 RXFTO。

23.9.9. RXFIFO 溢出中断标志（RXFOVF）

当试图进行数据写入操作，导致 RX FIFO 的数据数量大于 8 时，RXFOVF 被设置。通过向该位写入 1 来清除 RXFOVF。

23.9.10. RXFIFO 下溢中断标志（RXFUDF）

当试图进行数据写入操作，导致 RX FIFO 的数据编号小于 0 时，RXFUDF 被设置。通过向该位写入 1 来清除 RXFUDF。

23.9.11. RXFIFO 服务中断标志（RXFSER）

当 RX FIFO 中的数据数量高于或等于 SPIRXFCR 中指定的阈值时，将设置 RXFSER。

24. 集成电路间（I2C）

24.1. 介绍

I2C 是一种双线、双向串行总线，提供了一种简单、高效的数据交换方法，最大限度地减少了设备之间的互连。该总线适用于需要在多个设备之间进行短距离偶尔通信的应用。灵活的 I2C 允许将更多设备连接到总线上，以实现扩展和系统开发。

两条总线线路在标准模式下提供高达 100Kbit/s 的数据传输速率，在快速模式下提供高达 400Kbit/s 的数据传输速率，在高速模式下提供高达 3.4Mbit/s 的数据传输速率。

I2C 系统是一个真正的多主总线，包括仲裁和碰撞检测，如果多个设备试图同时控制总线，可以防止数据损坏。此功能支持具有多处理器控制的复杂应用，并且可以通过与装配线计算机的外部连接用于快速测试和终端产品的校准。

24.2. 特性

- 支持 7 位寻址。
- 支持标准模式、快速模式和高速模式
- 选择高速模式和标准/快速模式的软件选项
- 与 I2C 总线 2.1 版标准的正常模式和快速模式兼容。
- 多主操作。
- 可编程为 64 种不同的串行时钟频率之一。
- 软件可选择的确认位。
- 通过中断驱动逐字节传输数据。
- 仲裁中断，自动切换主设备为从设备。
- 传输完成和读取配置中断。
- 启动和停止信号的生成/检测。
- 重复生成 START 信号。
- 确认比特生成/检测。
- 巴士繁忙检测。
- 系统时钟停止模式下，选择性启用从地址接收功能
- 支持 SCL 或 SDA 线路的 GPIO 功能

24.3. 系统和方块图

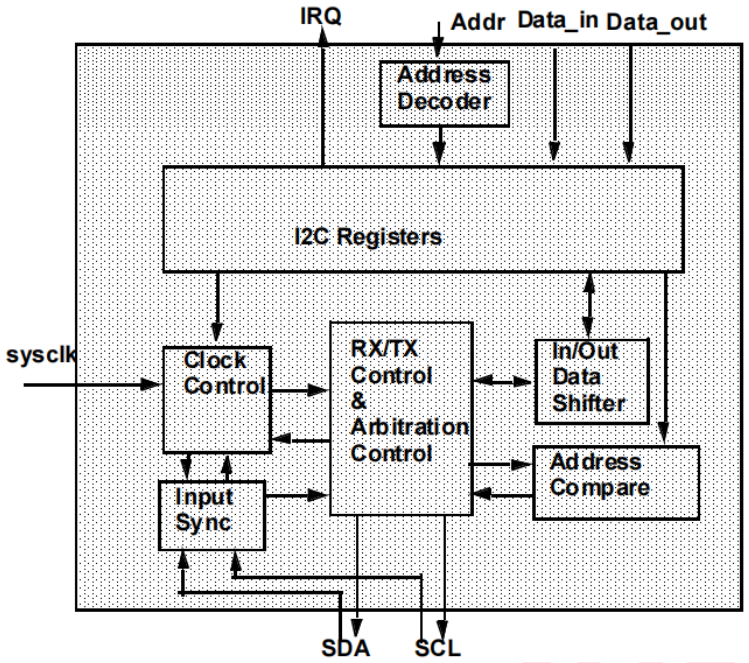


图 24-1：I2C 方块图

24.4. 内存映射和寄存器

I2C 内存映射中包含五个寄存器，如表 24-1 所示。

表 24-1：I2C 存储器映射

地址偏移量	Bit[7:0]
0x0000	I2C 状态寄存器（I2CS）
0x0001	I2C 时钟分频器寄存器（I2CP）
0x0002	I2C 控制寄存器（I2CC）
0x0003	I2C 从地址寄存器（I2CSA）
0x0004	I2C 端口控制寄存器（I2CPCR）
0x0005	I2C 从机高速指示寄存器
0x0006	I2C 从站 SDA 保持时间寄存器
0x0007	I2C 数据寄存器（I2CD）
0x0008	已反转
0x0009	已反转
0x000A	I2C 端口方向寄存器（I2CDDR）
0x000B	I2C 端口数据寄存器（I2CPDR）

24.4.1. I2C 状态寄存器（I2CS）

偏移地址：0x0000

	Bit7	6	5	4	3	2	1	Bit0
Read:	AACK	DACK	RXTX	ARBL	BBUSY	AASLV	RC	TF
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 24-2：I2C 状态寄存器（I2CS）

I2CS 寄存器显示 I2C 模块的状态。

TF — 传输完成标志

该信号用于指示数据传输或接收状态。对于接收端，其触发条件为接收到的数据字节（非地址字节）第九个时钟沿的下降沿，无论是否检测到确认位。对于主发射端，其触发条件为发送的数据字节或地址字节第九个时钟沿的下降沿，无论是否检测到确认位。对于从发射端，其触发条件为传输的地址字节或数据字节第九个时钟沿的下降沿，且必须检测到确认位。若 I2CC 中的 IEN 位同时被置位，则会触发中断。清除 TF 信号时，需先读取处于 TF 置位状态的 I2CS 模块，再通过访问 I2CD 或主发射模式写入 I2CC。

0 = 无意义，

1 = 数据或地址传输完成。

RC — 已收到完整文件

该信号指示接收器配置时机已到。对于主接收器，配置时间由数据或地址字节传输时第九个时钟信号的下降沿决定，无论是否检测到确认位。对于从接收器，配置时间由接收到地址或数据字节时第九个时钟信号的下降沿确定，并且必须同时收到确认位。若 I2CC 中的 IEN 位也被置位，则会触发中断。清除 RC 需先将 RC 置位后读取 I2CS，再写入 I2CC。

0 = 无意义，

1 = 表示是时候配置接收器了。

AASLV — 作为从属设备进行寻址

如果将 I2C 模块作为从设备，并且其自身的从设备地址与 SDL 上接收到的调用地址相匹配，则该时间由第八个时钟的下降沿设定。通过检测 START 或 STOP 位可明确这一点。

0 = 不是奴隶。

1 = 以从属方式寻址。

BBUSY — I2C 总线忙

显示总线状态。

0 = 公交车处于空闲状态。已清除 STOP 位。

1 = 总线忙。它由 START 位设置。

ARBL — 仲裁失败

显示总线的仲裁状态。在 SCL 高电平期间，将在以下情况下设置：

1. 当主驱动器在启动条件、地址周期、数据发送周期或停止条件下处于高电平时，SDA 的采样值较低。
2. 当数据接收周期的确认位期间主驱动器为高电平时，SDA 的采样值较低。
3. 总线忙时将尝试启动周期。

必须通过读取 I2CS 寄存器，由软件清除 ARBL。

0 = 未丢失的仲裁。

1 = 仲裁失败。

RXTX — 接收或发送。

指示 I2C 模块作为接收器或发送器的功能。其有效时间是第八个时钟的下降沿。

0 = 接收，接收数据。

1 = 发送，发送数据。

DACK — 数据确认接收完成。

指示是否在地址或数据传输期间检测到确认位。其有效时序为第九个时钟上升沿。

0 = 未收到确认位。

1 = 已收到确认位。

AACK — 地址确认错误。

指示主控器在地址阶段是否检测到确认位。该位由地址阶段的第九个时钟上升沿置为有效，并通过将 MSMOD 从 1 更改为 0 或重复启动条件清除。

0 = 没有地址确认错误。

1 = 地址确认错误。

24.4.2. I2C 时钟分频器寄存器（I2CP）

偏移地址：0x0001

	Bit7	6	5	4	3	2	1	Bit0
Read:	0	TEST	PRE5	PRE4	PRE3	PRE2	PRE1	PRE0
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 24-3：I2C 时钟分频器寄存器（I2CP）

I2CP 是一个预分频器，用于为数据收发器生成比特率时钟。由于 SCL 和 SDA 的上升沿和下降沿时间可能较慢，总线信号以预分频器频率进行采样。串行比特时钟频率等于 OPB 时钟除以分频器。

PRE[5:0] — 预分频器分频值

TEST — 时钟测试启用

它支持 I2C 时钟的测试模式。在正常模式下，其频率为 fipgclk/（396×（PRE[5:0]+1）+1），在测试模式下，其频率为 fipgclk/（8×（PRE[5:0]+1）+1）。

1 = 时钟测试模式启用

0 = 正常模式

24.4.3. I2C 控制寄存器（I2CC）

偏移地址：0x0002

	Bit7	6	5	4	3	2	1	Bit0
Read:	SLV_HSIE	HMS_EN	AMIE	REPSTA	ACKEN	MSMOD	IEN	EN
Write:								
RESET:	0	0	0	0	1	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 24-4：I2C 控制寄存器（I2CC）

I2CC 用于控制 I2C 模块的工作。EN - I2C 启用/禁用控制。

1 = 模块已启用。

0 = 模块已禁用。

该功能可启用/禁用模块，也可控制整个 I2C 模块的软件复位。设置该位将生成对模块的内部复位，在设置该位后经过 2 个时钟周期后置为有效，并保持有效状态 3 个时钟周期。因此，EN 位在设置后经过 5 个时钟周期后被取消为有效。

若在传输过程中启用了该模块，从设备模式会忽略当前 I2C 总线传输，并在检测到下一次启动条件时开始工作。主设备模式则无法感知总线是否处于忙状态，因此发起启动周期可能会干扰当前总线传输，最终导致当前主设备或 I2C 模块失去仲裁权，此时总线操作将恢复正常。

IEN — I2C 中断使能控制。

当设置时，它将启用 I2C 中断。

1 = I2C 中断已启用。

0 = I2C 中断已禁用。

MSMOD — I2C 主/从模式选择控制。

将 MSMOD 从 1 更改为 0 会在总线上产生 STOP 并选择从机模式。将 MSMOD 从 0 更改为 1 会在总线上产生 START 并选择主机模式。

1 = 主模式。

0 = 模拟模式。

ACKEN — 确认启用控制。

指定在主从接收器的确认周期中加载到 SDA 上的值。请注意，ACKEN 位仅在 I2C 总线接收到数据字节时才适用。在接收地址期间，如果接收到的地址与从地址匹配，则将发送 ACKEN 位，而不考虑 ACKEN 位。

1 = 在接收一个字节数据后，向 SDA 发送一个确认信号，时间是第九个时钟位。

0 = 在接收到一个字节数据后，向 SDA 发送无确认信号，持续第九个时钟位。

REPSTA — 重复启动

在此类情况下：1）主接收器发送从设备地址或数据字节后未收到确认；2）主发送器已发送地址或数据字节，无论是否收到确认，主设备可重复发送起始信号，并随后发送新的从设备地址。当配置从设备地址（通过写入 I2CD)后，将在 REPSTA 位设置时发送重复起始位。

1 = 生成重复的起始。

0 = 不重复启动。

AMIE — 地址匹配中断启用

选择如果 I2C 中断使能控制设置为 1，则从设备地址匹配是否将生成中断请求。在进入停止模式之前，应设置 AMIE，以便在从设备地址匹配时唤醒系统，并且在正常工作模式下必须清除 AMIE。

1 = 启用地址匹配中断请求。

0 = 禁用地址匹配中断请求。HSM_EN -高速模式启用

选择模块在主模式下的高速模式或快速/标准模式操作。应将 HSM_EN 位设置为选择高速模式操作。在 F/S 模式下发送主码且 HS_I2C 未丢失仲裁权后，应通过软件设置

HSM_EN，如果准备清除 MSMOD 以传输 STOP，则应在 MSMOD 后通过软件清除 HSM_EN。

0 = 快速/标准模式操作（默认）

1 = 高速模式运行

SLV_HSIE — 从机高速模式中断使能

选择如果 2C 中断使能控制设置为 1，则从机高速模式状态是否将生成中断请求。

1 = 启用从机高速模式状态请求。

0 = 暂停从机高速模式状态请求。

24.4.4. I2C 从地址寄存器（I2CSA）

偏移地址：0x0003

	Bit7	6	5	4	3	2	1	Bit0
Read:	ADDR7	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	0
Write:								
RESET:	0	1	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 24-5：：I2C 从地址寄存器（I2CSA）

当 I2C 作为从机工作时，I2CSA 存储地址并响应由主设备发送的地址。

ADDR[7:1] — 模块从地址。

当 I2C 模块处于从属模式时的从属地址。（默认情况下，I2C 为从属）。

提示：从机地址不可为 0x4、0x5、0x6、0x7，其他 124 个从机地址是允许的。

24.4.5. I2C 端口控制寄存器（I2CPCR）

偏移地址：0x0004

	Bit7	6	5	4	3	2	1	Bit0
Read: Write:	SDAPA	SCLPA	WOMI2C[1:0]		PDI2C[1:0]		PUI2C[1:0]	
RESET:	0	0	0	0	0	0	1	1

= Writes have no effect and the access terminates without a transfer error exception.

图 24-6：I2C 端口控制寄存器（I2CPCR）

SDAPA — SDA 端口分配位。

读/写位选择 SDA 的功能模式。

1 = 配置为 GPIO 的引脚。

0 = 配置为主要功能的 Pin。

SCLPA — SCL 端口分配位。

读/写位选择 SCL 的功能模式。

1 = 配置为 GPIO 的引脚。

0 = 配置为主要功能的 Pin。

WOMI2C[1:0] — 有线或模式位

读写位将对应的 I2C 引脚设置为开漏驱动模式。WOMI2C[1] 用于 SDA 引脚，WOMI2C[0] 用于 SCL 引脚。这些位仅在 GPIO 模式下可用。

1 = 输出时打开排水口。

0 = 输出时的 CMOS 驱动器。

PDI2C[1:0] — 下拉使能位

读写位用于控制对应的 I2C 引脚的下拉功能。其中，PDI2C[1] 位对应 SDA 引脚，PDI2C[0] 位对应 SCL 引脚。这些控制位在 GPIO 模式和主功能模式下均可使用。

1 = 启用 Pullup。

0 = 禁用上拉。

PUI2C[1:0] — 上拉使能位

读写位用于启用对应 I2C 引脚的上拉功能。PUI2C[1] 用于 SDA 引脚，PUI2C[0] 用于 SCL 引脚。这些位在 GPIO 和主功能模式下均可用。

1 = 启用 Pullup。

0 = 禁用上拉。

24.4.6. I2C 从机高速模式指示寄存器（I2CSHIR）

偏移地址：0x0005

	Bit7	6	5	4	3	2	1	Bit0
Read:	0	0	0	0	0	0	0	SLV_HS
Write:								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 24-7：I2C 从机高速模式指示寄存器（I2CSHIR）

SLV_HS — 从机高速模式。

当 I2C 被选为从机时，该位指示模块是选择高速模式还是快速/标准模式数据传输。该位在主代码和未确认位被接收的第九个 SCL 上升沿被设置。必须通过写入 1 来清除该位，否则 SCL 线将被强制置为低电平状态。

0 = 选择 I2C 进行快速/标准数据传输（默认）。

1 = 选择 I2C 进行高速模式数据传输。

24.4.7. I2C 从机 SDA 保持时间寄存器（I2CSHT）

偏移地址：0x0006

	Bit7	6	5	4	3	2	1	Bit0
Read: Write:	SCL_FILTER_EN	SDA_FILTER_EN	SLVHT					
RESET:	0	0	0	0	1	0	0	1

= Writes have no effect and the access terminates without a transfer error exception.

图 24-8：I2C 从机 SDA 保持时间寄存器（I2CSHT）

SCL_FILTER_EN — SCL 过滤器启用。

如果启用了 SCL 滤波器，在 HS 模式传输过程中，将过滤掉 SCL 线上出现的 10ns 脉冲。如果正在进行 F/S 模式传输，
SCL 线上出现的 50ns 脉冲将被过滤。

SDA_FILTER_EN — SDA 过滤器启用。

如果启用了 SDA 滤波器，在 HS 模式传输过程中，将过滤掉 SDA 线上出现的 10ns 脉冲。如果正在进行 F/S 模式传输，则
SDA 线上出现的 50ns 脉冲将被过滤。

SLVHT — 从属 SDA 线路保持时间配置。

当 I2C 以从机输出模式工作时，数据将在 SCL 下降沿之后改变，且内部 SDA 保持寄存器的值等于 SLVHT。不允许写入数值 0。

24.4.8. I2C 数据寄存器（I2CD）

偏移地址：0x0007

	Bit7	6	5	4	3	2	1	Bit0
Read:	R7	R6	R5	R4	R3	R2	R1	R0
Write:	T7	T6	T5	T4	T3	T2	T1	T0
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 24-9：I2C 数据寄存器（I2CD）

I2CD 寄存器用于存储待传输数据（下一字节）或接收数据。在主控模式下，它还保存从设备地址和传输方向信息。其中[7:1] 位构成从设备地址，[0] 位表示传输方向（读/写）。在主控-接收模式下，读取 I2CD 寄存器可触发数据读取操作，并启动下一字节的接收流程。在主控-发送模式下，写入 I2CD 寄存器会存储待传输的下一个字节。当处于从设备模式时，该寄存器在被寻址后同样具备相同功能。

R[7:0] — 接收比特[7:0]

T[7:0] — 发送比特[7:0]

24.4.9. I2C 端口方向寄存器（I2CDDR）

偏移地址：0x000A

	Bit7	6	5	4	3	2	1	Bit0
Read:	0	0	0	0	0		DDRI2C[1:0]	
Write:								
RESET:	0	0	0	0	0		0	0

图 24-10：I2C 端口方向寄存器（I2CDDR）

Read：任何时候

Write：任何时间

DDRI2C[1:0] — I2C 数据方向位

读/写位控制 I2C 引脚的数据方向。这些位仅在 GPIO 模式下可用

1 = 配置为输出的对应引脚

0 = 配置为输入的对应引脚

24.4.10. I2C 端口数据寄存器（I2CPDR）

偏移地址：0x000B

	Bit7	6	5	4	3	2	1	Bit0
Read:	0	0	0	0	0	0	PORTI2C[1:0]	
Write:								
RESET:	0	0	0	0	0	0	0	0

图 24-11：I2C 端口数据寄存器（I2CPDR）

Read：任何时候

Write：任何时候

PORTI2C[1:0] — I2C 端口数据位

写入这些位将设置对应 I2C 引脚的输出数据，这些引脚被配置为 GPIO。读取这些位将返回 I2C 引脚的电平。

24.5. 功能说明

I2C 模块是一种双线双向串行总线，能以简单高效的方式实现数据交换，最大限度减少设备间的连接。软件既可通过轮询方式检测 I2C 状态标志，也可采用中断驱动模式操作。当接收到或需要发送字节时，I2C 总线的 TF 位会被置位；若同时触发 I2CC 的 IEN 位，系统将立即生成中断信号。

如果在传输过程中丢失了仲裁，将设置 ARBL，如果 I2CC 中的 IEN 位也设置，则会生成中断。如果没有从属设备的地址确认(AACK 设置），也会生成中断。

如果模块不想接收下一个字节，它可以清除 ACKEN。

24.5.1. 主模式

当 I2C 模块以主设备模式工作时，若总线空闲（即 I2CS 的 BBUSY 位为清零状态），则可能发起传输操作。将 MSMOD 位从 0 切换为 1 会在总线上触发起始信号并切换为主设备模式。主设备通过控制 R/W 位来决定传输方向：发送时首字节为从设备地址，后续字节为数据。若在每个时钟周期的第九个周期后未收到应答信号，则将 MSMOD 位从 1 切换为 0 以在总线上生成停止信号。I2CP 的 PRE[5:0] 位用于控制 I2C 总线的比特率时钟。

通过在设置 REPSTA 位之后配置从地址，主设备可以重复发送 START 信号，而不是发送 STOP 信号。

24.5.2. 从属模式

如果 MSMOD 位被清除，该模块就是从属模块，可以被其他主设备寻址。当获胜的主设备试图寻址它时，它将释放 SDA 线并立即切换到从属模式。

提示：I2C 不能同时在从属模式和主模式下工作。

24.5.3. 通信协议

I2C 通信协议由六个部分组成：START、数据源/接收器、数据方向、从机确认、数据、数据确认和 STOP，如图 24-2 所示，下文将对这些部分进行描述。

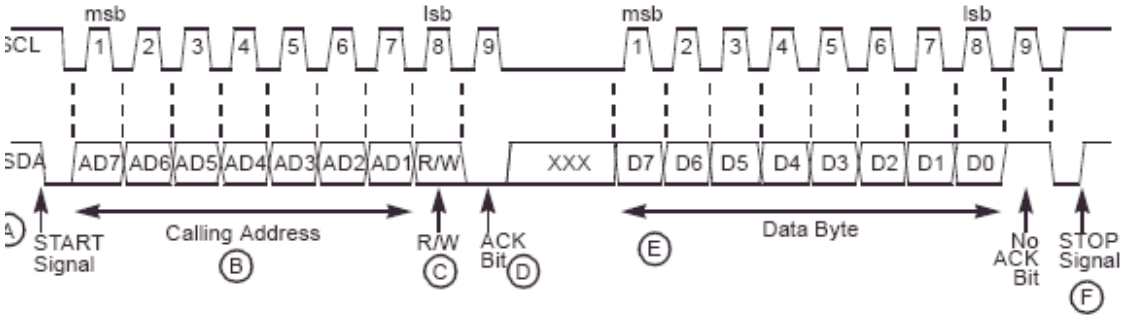


图 24-12：I2C 通信协议

1. START 信号 — 当没有其他设备处于总线主控状态（SCL 和 SDA 线均保持高电平）时，设备可通过发送 START 信号发起通信（参见图 24-2 中的 A）。该信号定义为在 SCL 保持高电平时，SDA 线从高电平跳变为低电平的信号变化。此信号标志着数据传输的开始（每次传输可能包含多个字节），并唤醒所有从属设备。

2. 从站地址传输 — 主站发送从站地址在 START 信号(B)后的第一个字节，发送完 7 位呼叫地址后，发送 R/W 位(C)，该位告诉从站数据传输方向。

每个从设备必须具有唯一地址。I2C 主设备不能发送与其自身从设备地址相同的地址；不能同时作为主设备和从设备。与主设备发送的地址相匹配的从设备将在第 9 个时钟(D)将 SDA 拉低，以返回一个确认位。

3.数据传输 — 当成功实现从机寻址后，就可以按照主设备发送的 R/W 位指定的方向以字节为单位进行数据传输(E)。

如图 24-2 所示，数据只能在 SCL 低电平时改变，且必须在 SCL 高电平时保持稳定。每个数据位 SCL 脉冲一次，最高有效位最先发送。接收设备必须在第九个时钟脉冲时将 SDA 拉低来确认每个字节，因此一个数据字节的传输需要九个时钟脉冲。

如果从机不承认主设备，则必须将 SDA 置高。主设备随后可以生成 STOP 信号来中止数据传输，或者生成 START 信号（重复的启动，如图 24-3 所示）来开始新的呼叫序列。

如果主接收器在发送一个字节后没有确认从机发射器，则意味着从机数据结束。从机释放 SDA，以便主机生成 STOP 或 START 信号。

4. 停止信号 — 主设备可通过发送停止信号来终止通信，从而释放总线。停止信号的定义是当从设备处于逻辑高电平(F)时，SDA 信号从低电平跳变至高电平。需要注意的是，即使从设备已经发出确认响应，主设备仍可发送停止信号，此时从设备必须立即释放总线控制权。

5. 主设备可重复发送 START 信号后接呼叫指令（图 24-3 中的 A），而非发送 STOP 信号。当主设备在未先发出 STOP 信号结束通信的情况下生成 START 信号时，即触发重复 START 机制。这种机制允许主设备在不释放总线的情况下，与另一从设备或处于不同工作模式（收发模式）的同一从设备进行通信。

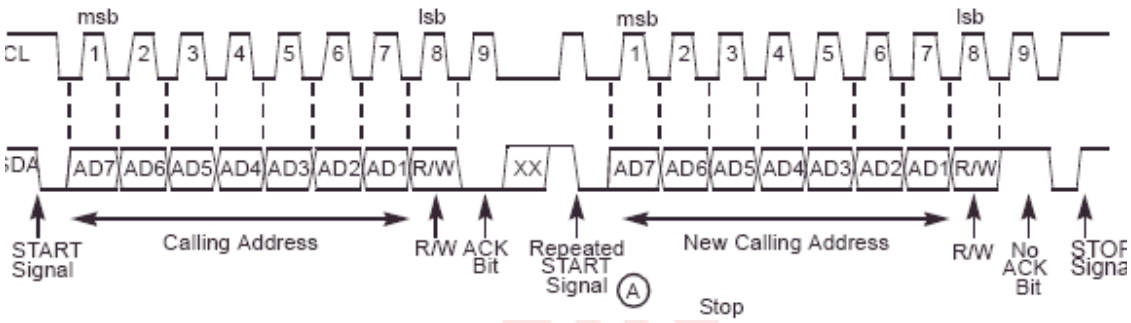


图 24-13：I2C 协议的重复启动

24.5.4. 仲裁程序

当多个设备同时请求总线时，总线时钟的确定需通过同步机制：低电平周期取各设备中最长的时钟低电平周期，高电平周期则取最短值。数据仲裁机制负责判定竞争设备的相对优先级。若某设备在其他设备发送逻辑高电平时仍发送逻辑高电平，该设备将失去仲裁权，随即切换至从属接收模式并停止驱动 SDA 信号。

失去仲裁权的主设备可以生成时钟脉冲，直到它失去仲裁权的那个字节结束。获胜的主设备可能正在试图寻址它。因此，失去仲裁权的主设备必须立即切换到从属模式。

在此情况下，从主模式到从模式的转换不会产生 STOP 条件。同时，硬件将 I2CSR 的 ARBL 位设置为表示仲裁丢失。

24.5.5. 时钟同步

由于采用了线与逻辑，SCL 信号从高电平跳变至低电平时，会触发总线所有连接设备的响应。当主设备将 SCL 拉低时，各设备随即开始计算自身的低电平周期。当某个设备时钟进入低电平状态后，它会持续保持 SCL 处于低电平直至时钟恢复高电平。但需要注意的是，若其他设备时钟仍处于低电平阶段，此时该设备时钟的低电平到高电平转换可能不会改变 SCL 的状态。

因此，低电平持续时间最长的器件将保持同步时钟 SCL 处于低电平，低电平持续时间较短的器件在此期间进入高等待状态（见图 24-4），当所有相关器件都用尽其低电平时，同步时钟 SCL 被释放并被拉高。

此时，设备时钟和 SCL 的状态之间没有区别，因此所有设备都开始计算它们的高电平周期。第一个完成其高电平周期的设备将再次把 SCL 拉低。

24.5.6. 握手机制

时钟同步机制可用作数据传输中的握手。从设备在完成一个字节传输（9 位）后可保持 SCL 低电平。在这种情况下，时钟机制将停止总线时钟，并强制主设备时钟进入等待状态，直到从设备释放 SCL。

24.5.7. 时钟延展

从设备可以使用时钟同步机制来减慢传输的比特率。在主设备将 SCL 拉低之后，从设备可以在需要的时间段内将 SCL 拉低，然后释放它。如果从设备的 SCL 低电平时间段长于主设备的 SCL 低电平时间段，则所产生的 SCL 总线信号低电平时间段将会被拉长。

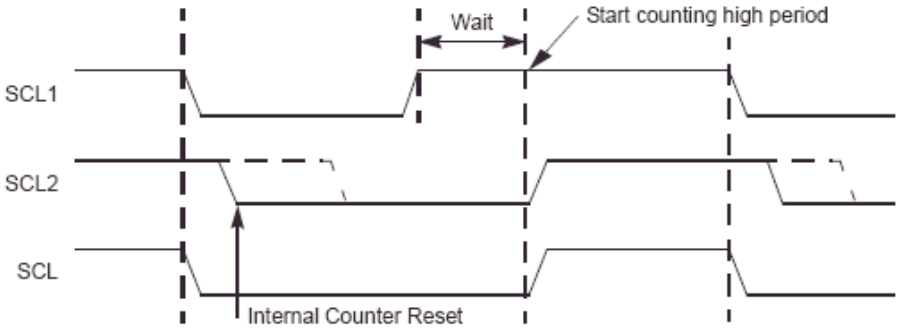


图 24-14：SCL 同步

24.5.8. 高速模式操作

I2C 模块在高速模式下可实现最高 3.4 兆比特/秒的传输速率，同时仍与快速模式或标准模式（F/S 模式）设备保持完全向下兼容，支持混合速度总线系统中的双向通信。除高速传输期间不执行仲裁和时钟同步操作外，该模块采用的串行总线协议和数据格式与 F/S 模式系统完全一致。

HS 模式下的串行数据传输格式符合标准模式 I2C 总线规范。只有满足以下条件时，HS 模式才能启动（所有这些条件均在 F/S 模式下发生）：

1. 起动条件(S)
2. 8 位主代码（00001XXX）
3. 不确认位(A)

（参见 HS 模式下的图 24-5 数据传输格式）和（参见 HS 模式下的图 24-6 完整的传输）更详细地说明了这一点。HS 主代码有两个主要功能：

1. 它允许在 F/S 模式速度下进行仲裁和竞争主控之间的同步，从而产生一个获胜的主控。
2. 表示 HS 模式传输的开始。

HS 模式主码是预留的 8 位代码，不用于从设备寻址或其他用途。此外，由于每个主控器都有其独特的主码，单个 I2C 总线系统最多可容纳八个 HS 模式主控器（但主码 0000 1000 应保留用于测试和诊断）。I2C 模块的主码可通过软件编程设置。仲裁和时钟同步仅在传输主码时进行，而非确认位(A)之后，此时获胜的主控器将保持激活状态。主码向其他设备发出信号，表明 HS 模式传输即将开始，且连接设备必须符合 HS 模式规范。由于不允许任何设备对主码进行确认，主码后会跟随一个非确认位(A)。当非确认位(A)出现且 SCL 信号线被拉高至高电平时，当前激活的主控器将切换至 HS 模式，并启用电流源上拉电路（时间 tH，参见图 24-6《完整 HS 模式传输》）来提升 SCL 信号电平。由于其他设备可能通过延长 SCL 信号的低电平周期来延迟 tH 之前的串行传输，当前激活的主控器将在所有设备释放 SCL 线且 SC 信号达到高电平时启用其电流源上拉电路，从而加快 SCL 信号上升沿的末段速度。然后，主动主设备发送重复的 START 条件（Sr），随后发送一个带有 R/W 比特地址的 7 位从设备地址，并从所选的从设备接收确认比特(A)。

在经历重复的起始信号（START）条件后，每当接收到确认位(A)或非确认位(A)时，当前活动主设备都会禁用其电流源上拉电路。这一操作使得其他设备能够通过延长 SCL 信号的低电平周期来延迟串行传输。当所有设备释放信号且 SCL 信号达到高电平时，活动主设备会重新启用电流源上拉电路，从而加快 SCL 信号上升沿的末段速度。

在下一次重复的 START（Sr）之后，数据传输继续以 Hs 模式进行，只有在 STOP 条件(P)之后才会切换回 F/S 模式。为了减少主控码的开销，主控器可以将多个 HS 模式传输连接起来，这些传输之间由重复的 START 条件（Sr）分隔。

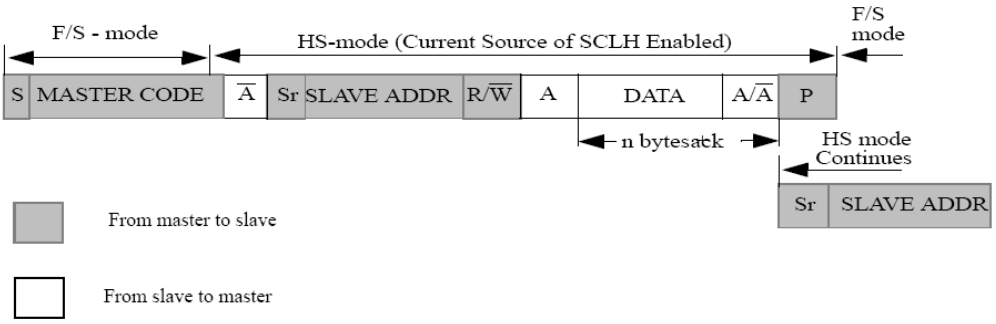


图 24-15：HS 模式下的数据传输格式

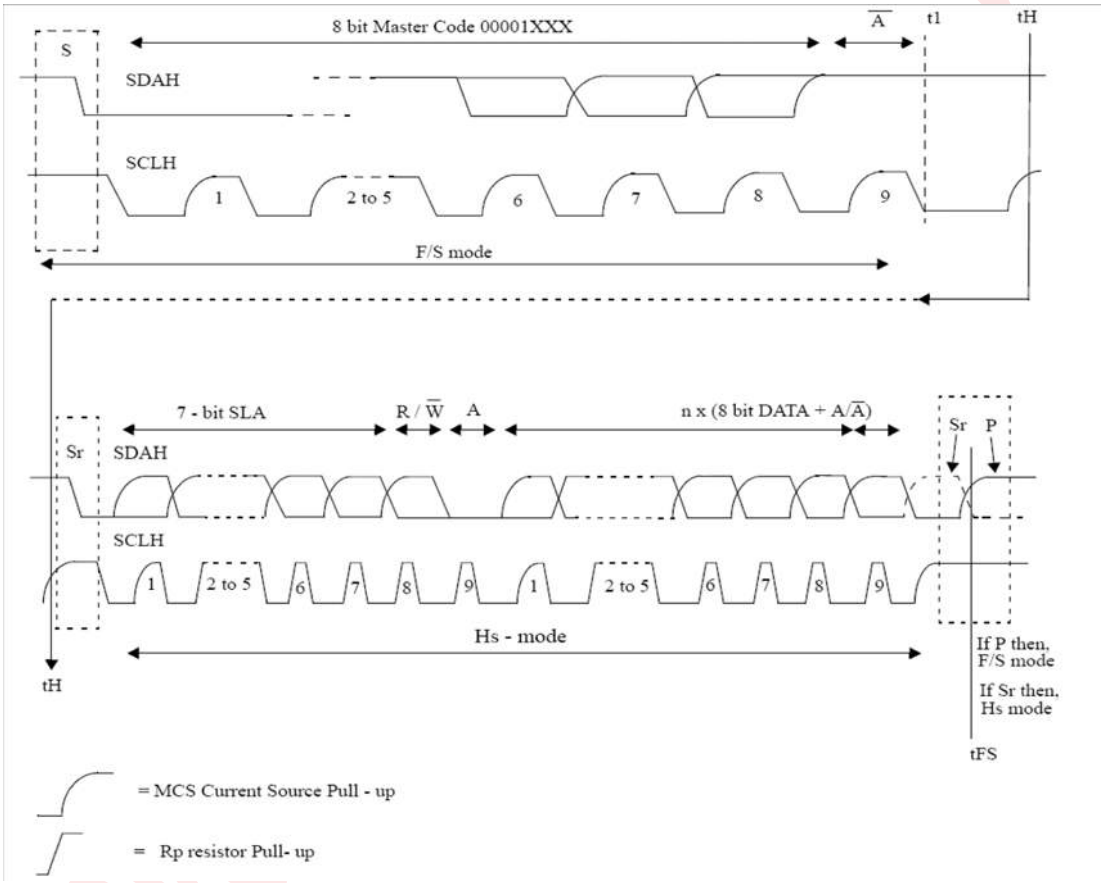


图 24-16：完整的 HS 模式切换

24.5.9. 软件处理流程图

1. 初始化

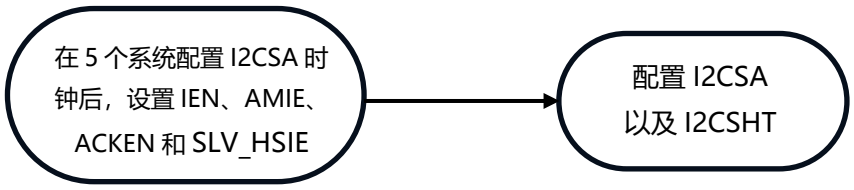


图 24-17：从机模式初始化

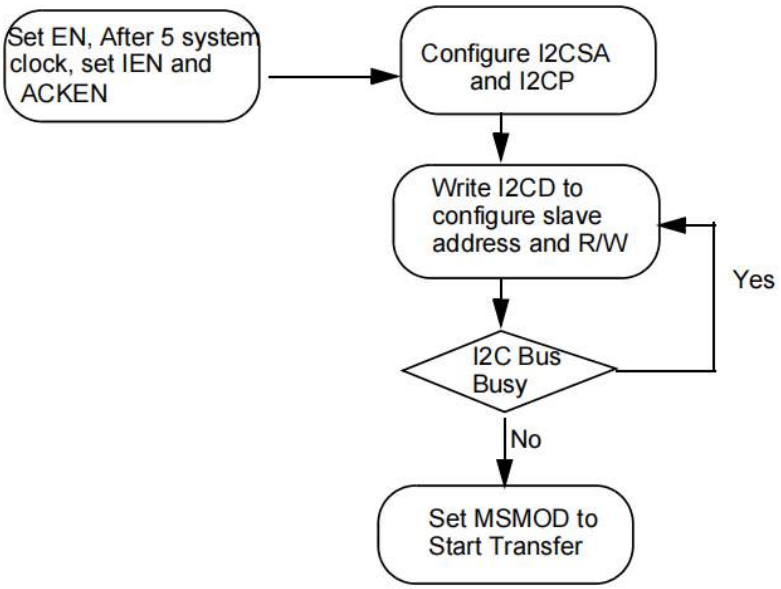


图 24-18：主模式初始化

2、中断事务

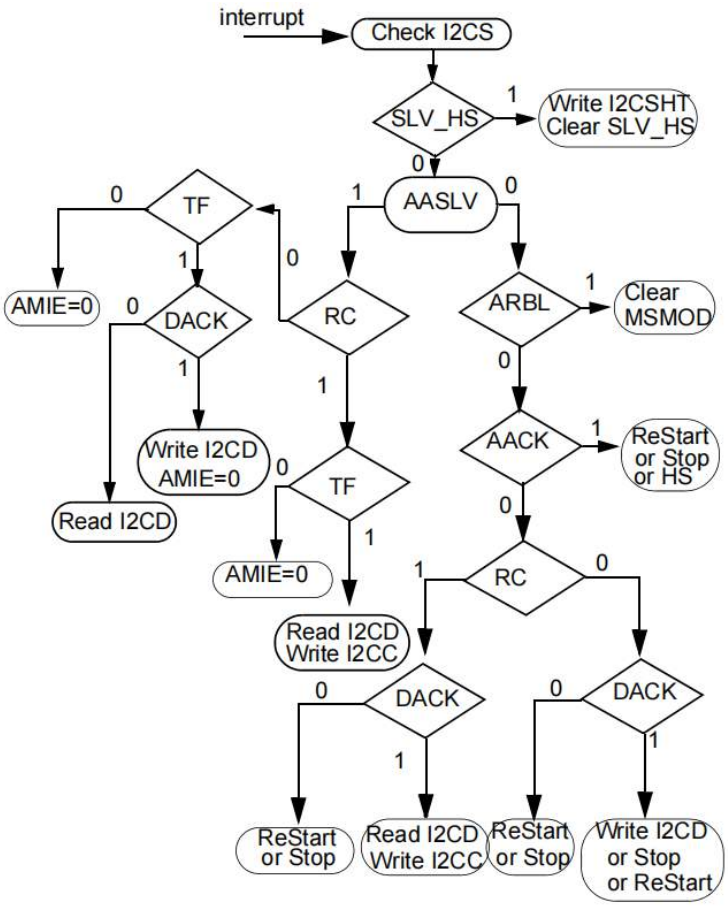


图 24-19：中断事务

如果出现来自 I2C 的中断，首先检查 I2CS，以确认 I2C 是否处于主模式或从属模式。

首先检查 SLV_HS 状态，如果 SLV_HS 被设置，则写入 I2CSHT 以配置高速计时，并将 I2CSHIR 写入 1 以清除 SLV_HS，否则：

在从设备模式下，若已设置 RC，则表示 I2C 处于从设备接收模式。此时需检查 TF 状态：若 TF 也已设置，则说明已成功接收数据（而非地址），此时应读取 I2CD 并写入 I2CC 以清除 RC 和 TF，以便接收下一个字节数据。若 TF 未被设置，则表示仅接收到从设备地址，此时应写入 I2CC 以清除 RC 和 AMIE，随后继续接收下一个字节数据。

在从机模式下，如果 RC 未被设置，则检查 TF；如果 TF 已被设置，则检查 DACK；如果 DACK 也被设置，则说明 I2C 处于从机发送模式，此时写入 I2CD 以清除 TF，写入 I2CC 以清除 AMIE，并传输下一个字节数据。如果 DACK 未被设置，则读取 I2CD 以清除 TF。

在从属模式下，如果 TF 和 RC 均未设置，则写入 I2CC 以清除 AMIE。

在主模式下，如果 ARBL 已被设置，则表示仲裁丢失，然后写入 I2CC 以清除 MSMOD。如果 ARBL 未被设置，

则检查 AACK，如果 AACK 已被设置，则表示地址确认错误，然后写入 I2CC 以重复启动或停止。

在主模式下，如果 ARBL 和 AACK 未被设置，则检查 RC，如果 RC 被设置，则表示 I2C 处于主-接收器模式，然后检查 DACK，如果 DACK 也被设置，则读取 I2CD 以清除 RC，并写入 I2CC 选择是否确认数据。如果 DACK 未被设置，则写入 I2CC 以重复开始或停止。

在主模式下，如果 ARBL 和 AACK 未被设置，则检查 RC，如果 RC 未被设置，则表示 I2C 处于主发送模式，然后检查 DACK，如果 DACK 被设置，则写入 I2CD 以清除 TF 并发送下一个字节数据或写入 I2CC 以重复开始或停止。如果 DACK 未被设置，则写入 I2CC 以重复开始或停止。

25. 脉冲宽度调制器（PWM）

25.1. 介绍

内置四个 PWM 定时器，每个定时器包含 2 个预设分频、2 个时钟分频器、4 个时钟选择器、4 个 16 位计数器、4 个 16 位比较器和 2 个死区发生器，每个定时器均可独立用作定时器并触发中断。

两个 PWM 定时器共享同一预分频器。时钟分频器为每个定时器提供 5 个时钟源（1、1/2、1/4、1/8、1/16)。每个定时器中的 16 位计数器接收来自时钟选择器的时钟信号，可处理一个 PWM 周期。16 位比较器通过将计数器数值与预先加载的阈值寄存器数值进行比对，生成 PWM 占空比。时钟分频器输出的时钟信号称为 PWM 时钟。死区发生器以 PWM 时钟作为时钟源，一旦启用，两个 PWM 定时器的输出将被阻断。两个输出引脚均作为死区发生器的输出信号，用于控制片外功率器件。比较器的数值用于脉冲宽度调制。当倒计数器数值与比较寄存器值匹配时，计数器控制逻辑会改变输出电平。

每个 PWM 定时器都包含一个捕获通道。捕获通道 0 与 PWM 通道 0 共享 PWM 通道 0 内置的计时器，捕获通道 1 与 PWM 通道 1 共享另一个计时器，依此类推。因此，用户必须在启用捕获功能前完成 PWM 定时器的配置。启用捕获功能后，当输入通道出现上升沿时，系统会将当前 PWM 计数器值锁存至 CRLR 寄存器；当出现下降沿时，则锁存至 CFLR 寄存器。通过设置 CCR0[1]（上升沿锁存中断使能）和 CCR0[2]（下降沿锁存中断使能），可编程捕获通道 0 的中断触发条件。捕获通道 1 至 3 具有相同功能特性。每当捕获通道触发 0/1/2/3 号中断时，对应的 PWM 计数器 0/1/2/3 将立即重置。最大捕获频率应由中断处理时间决定：若中断处理时间为 T0，则捕获通道输入信号在 T0 期间不得改变，此时最大捕获频率为 1/T0。

PWM 到中断控制器（INTC）只有四个中断通道，PWM 0 和 Capture 0 共用同一个中断通道，PWM1 和 Capture 1 共用同一个中断通道，以此类推，所以同一通道内的 PWM 功能和 Capture 功能不能同时使用。

25.2. 特性

脉冲宽度调制器具有以下显著特点：

- 可编程周期
- 可编程工作周期
- ·一个“死亡区”发生器
- 捕获功能
- 引脚可配置为通用 I/O

25.3. 方块图

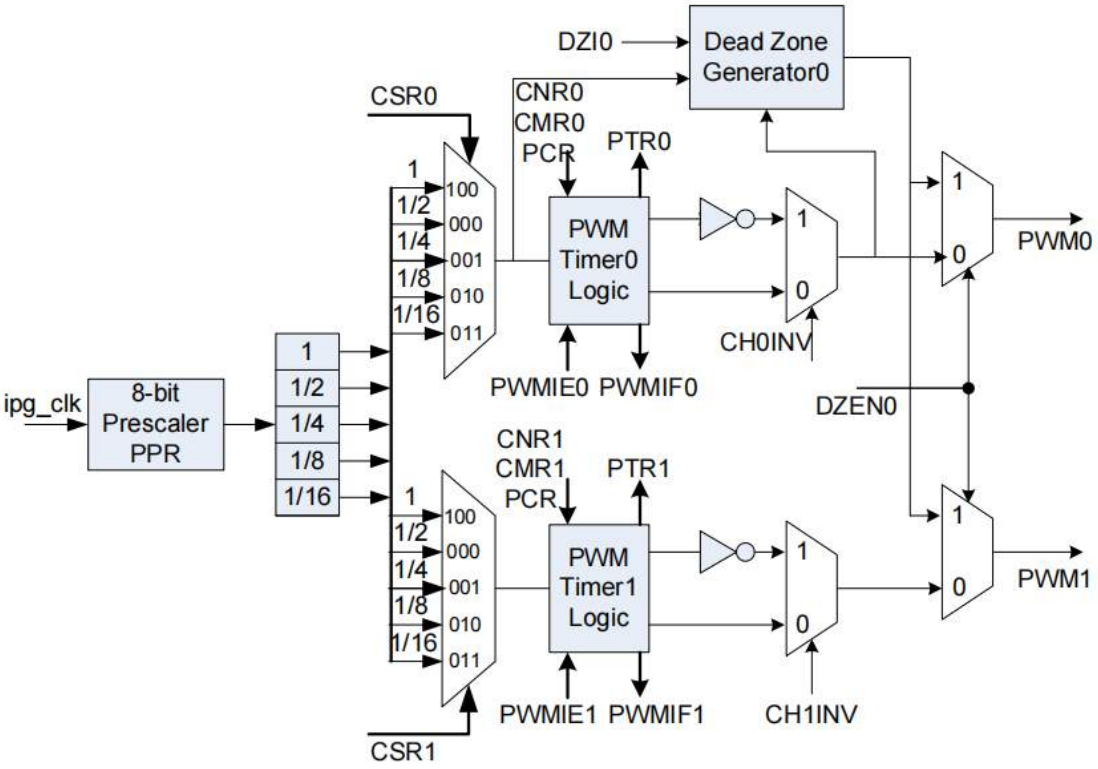


图 25-1：PWM 方块图

25.4. 信号描述

表 25-1：PWM 信号描述

名称	I/O	宽度	重置状态	描述
PWM0	I/O	1	0	PWM0 pin
PWM1	I/O	1	0	PWM1 pin
PWM2	I/O	1	0	PWM2 pin
PWM3	I/O	1	0	PWM3 pin

PWMx 用作通用输入/输出，也可用作 PWM 发送输出或捕获输入

默认状态下，它用作通用输入端口。

25.5. 内存映射和寄存器

本小节描述内存映射和寄存器结构。

25.5.1. 内存映射

表 25-2：模块内存映射

地址偏移量	Bit[15:8]	Bit[7:0]	访问权限 [1]
0x0000	PWM 预缩放寄存器（PPR）		S/U
0x0004	PWM 时钟选择寄存器（PCSR）		S/U
0x0008	PWM 控制寄存器（PCR）		S/U
0x000C	PWM 计数器寄存器 0（PCNR0）		S/U
0x0010	PWM 比较器寄存器 0（PCMR0）		S/U
0x0014	PWM 计时器寄存器 0（PTR0）		S/U
0x0018	PWM 计数器寄存器 1（PCNR1）		S/U
0x001C	PWM 比较器寄存器 1（PCMR1）		S/U
0x0020	PWM 计时器寄存器 1（PTR1）		S/U
0x0024	PWM 计数器寄存器 2（PCNR2）		S/U
0x0028	PWM 比较器寄存器 2（PCMR2）		S/U
0x002C	PWM 计时器寄存器 2（PTR2）		S/U
0x0030	PWM 计数器寄存器 3（PCNR3）		S/U
0x0034	PWM 比较器寄存器 3（PCMR3）		S/U
0x0038	PWM 定时器寄存器 3（PTR3）		S/U
0x003C	PWM 中断使能寄存器（PIER）		S/U
0x0040	PWM 中断标志寄存器（PIFR）		S/U
0x0044	PWM 捕获控制寄存器 0（PCCR0）		S/U
0x0048	PWM 捕获控制寄存器 1（PCCR1）		S/U
0x004C	PWM 捕获上升锁存寄存器 0（PCRLR0）		S/U
0x0050	PWM 捕获锁存寄存器 0（PCFLR0）		S/U
0x0054	PWM 捕获上升锁存寄存器 1（PCRLR1）		S/U
0x0058	PWM 捕获锁存寄存器 1（PCFLR1）		S/U
0x005C	PWM 捕获上升锁存寄存器 2（PCRLR2）		S/U
0x0060	PWM 捕获锁存寄存器 2（PCFLR2）		S/U
0x0064	PWM 捕获上升锁存寄存器 3（PCRLR3）		S/U
0x0068	PWM 捕获锁存寄存器 3（PCFLR3）		S/U
0x006C	PWM 端口控制寄存器（PPCR）		S/U

提示： S/U = CPU 监控程序或用户模式访问。

25.5.2. 寄存器

25.5.2.1. PWM 预缩放寄存器（PPR)

寄存器（PPR）用于设置预分频器和死区长度。

偏移地址：0x0000 和 0x0003

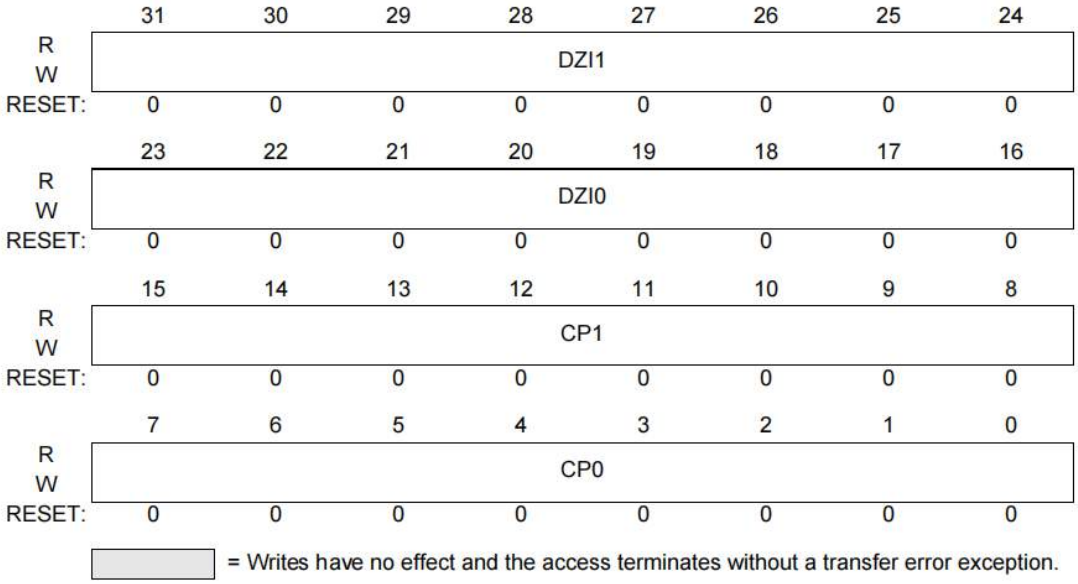


图 25-2：PWM 预缩放寄存器（PPR)

DZI1[7:0] — PWM2 和 PWM3 的死区间隔寄存器 1

这 8 位确定死区长度。死区长度的 1 个单位时间由时钟选择器 1 接收。

DZI0[7:0] — PWM0 和 PWM1 的死区间隔寄存器 0

这 8 位确定死区长度。死区长度的 1 个单位时间由时钟选择器 0 接收。

CP1[7:0] — PWM 计时器 2 和 3 的时钟预设值 1

时钟输入在被送入 PWM 计时器 2 和 3 的计数器之前被除以（CP1 + 1）。

CP0[7:0] — PWM 计时器 0 和 1 的时钟预设值 0

时钟输入被除以（CP0 + 1），然后输入到 PWM 计时器 0 和 1 的计数器中。

25.5.2.2. PWM 时钟选择寄存器（PCSR）

分频器为每个定时器提供 5 个时钟源（1、1/2、1/4、1/8、1/16)。每个定时器从分频器接收其自身的时钟信号，而分频器的时钟信号则来自 8 位预分频器。

偏移地址：0x0004 和 0x0007

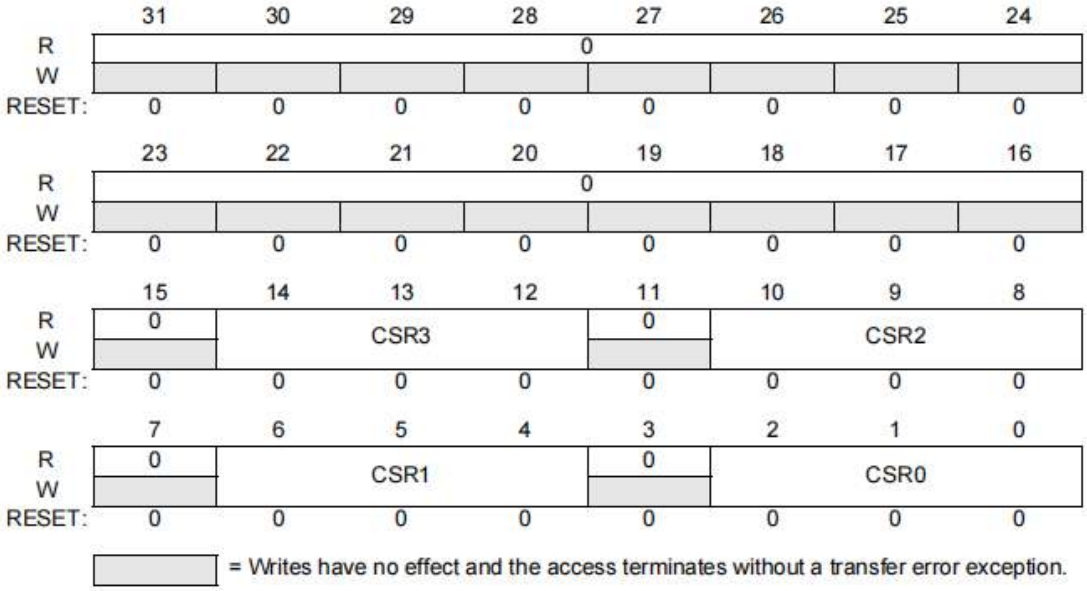


图 25-3：PWM 时钟选择寄存器（PCSR）

CSR3[2:0] — 定时器 3 时钟源选择

为计时器 3 选择时钟输入。

表 25-3：PWM 时钟分频器设置

CSR3[2:0]	输入时钟除以
100~111	1
011	16
010	8
001	4
000	2

CSR2[2:0] — 定时器 2 时钟源选择

为计时器 2 选择时钟输入。与 CSR3 相同。

CSR1[2:0] — 定时器 1 时钟源选择

选择计时器 1 的时钟输入。与 CSR3 相同

CSR0[2:0] — 定时器 0 时钟源选择

选择计时器 0 的时钟输入。与 CSR3 相同

25.5.2.3. PWM 控制寄存器（PCR）

此寄存器是 PWM 控制寄存器。

偏移地址：0x0008 和 0x000B

	31	30	29	28	27	26	25	24
R	0				CH3MOD	CH3INV	0	CH3EN
W								
RESET:	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R	0				CH2MOD	CH2INV	0	CH2EN
W								
RESET:	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R	0				CH1MOD	CH1INV	0	CH1EN
W								
RESET:	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	0		DZEN1	DZEN0	CH0MOD	CH0INV	0	CH0EN
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 25-4：PWM 控制寄存器（PCR）

CH3MOD — 定时器 3 自动加载/单次模式

1 = 自动加载模式

0 = 单次模式

提示：如果在此位有上升或下降的转换，将导致 CNR3 和 CMR3 为清零。

CH3INV — 计时器 3 的反相器打开/关闭

1 = 反相器打开

0 = 反相器关闭

CH3EN — 定时器 3 启用/禁用

1 = 启用

0 = 关闭

CH2MOD — 定时器 2 自动加载/单次模式

1 = 自动加载模式

0 = 一次性模式

提示：如果在此位有上升或下降的转换，将导致 CNR2 和 CMR2 为清零。

CH2INV — 计时器 2 的反相器打开/关闭

1 = 反相器打开

0 = 反相器关闭

CH2EN — 定时器 2 启用/禁用

1 = 启用

0 = 关闭

CH1MOD — 定时器 1 自动加载/单次模式

1 = 自动加载模式

0 = 单次模式

提示：如果在此位有上升或下降的转换，将导致 CNR1 和 CMR1 为清零。

CH1INV — 计时器 1，反相器 ON/OFF

1 = 反相器打开

0 = 反相器关闭

CH1EN — 定时器 1 启用/禁用

1 = 启用

0 = 关闭

CH0MOD-定时器 0 自动加载/单次模式

1 = 自动加载模式

0 = 一次性模式

提示：如果在此位有上升或下降的转换，将导致 CNR0 和 CMR0 为清零。

CH0INV — 计时器 0 反相器 ON/OFF

1 = 反相器打开

0 = 反相器关闭

CH0EN — 计时器 0 启用/禁用

1 = 启用

0 = 关闭

DZEN1 — Dead-Zone 1 发生器启用/禁用

1 = 启用

0 = 关闭

提示：当启用 DZEN1 时，CH3EN 应设置为禁用。因为通道 3 和通道 2 的输出都由通道 2 决定。

DZEN0 — Dead-Zone 0 发生器启用/禁用

1 = 启用

0 = 关闭

提示：当启用 DZENO 时，应将 CH1 EN 设置为禁用。因为通道 1 和通道 0 的输出都由通道 0 决定。

25.5.2.4. PWM 计数器寄存器（PCNR0/1/2/3）

这些寄存器通过定义周期内的 count Pulse 数来控制 PWM 的周期。

偏移地址：0x000C 和 0x000F

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CNR0[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CNR0[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x0018 和 0x001B

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CNR1[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CNR1[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x0024 和 0x0027

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CNR2[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CNR2[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x0030 和 0x0033

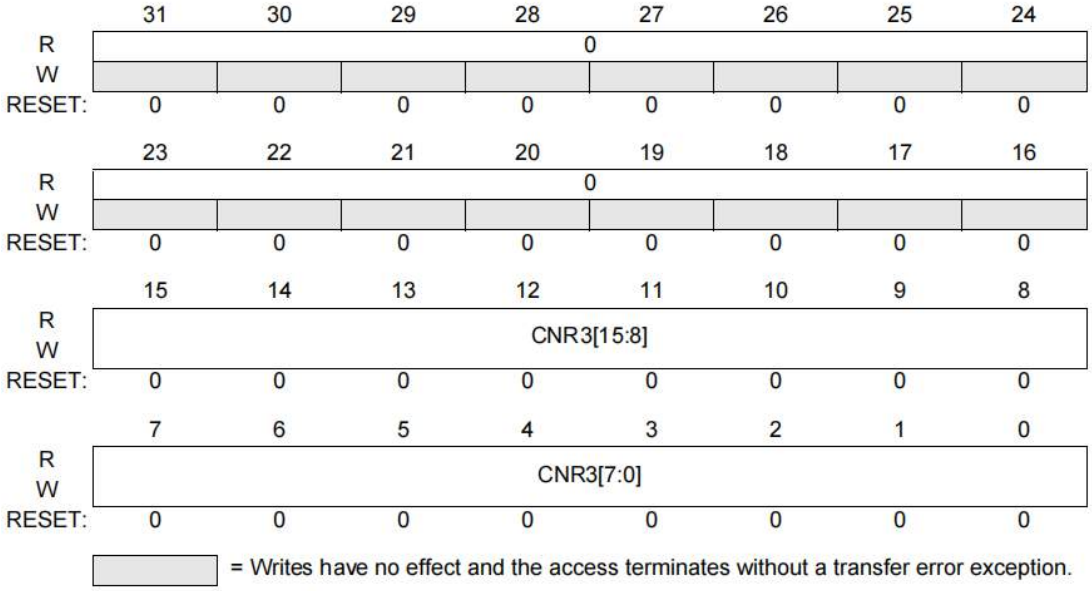


图 25-5：PWM 计数器寄存器（PCNR）

CNR[15:0] — PWM 计数器/定时器加载值

插入的数据范围：65535~0（单位：1 PWM 时钟周期）

提示：

1. 一个 PWM 周期宽度 = CNR + 1。如果 CNR 等于零，PWM 计数器/定时器将停止。
2. 程序员可以随时将数据写入 CNR，且将在下一个 PWM 计数器周期生效。

25.5.2.5. PWM 比较器寄存器（PCMR0/1/2/3）

这些寄存器定义脉冲的宽度。当计数器与该寄存器中的值相匹配时，在周期的持续时间内，输出将被重置。

偏移地址：0x0010 和 0x0013

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CMR0[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CMR0[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x001C 和 0x001F

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CMR1[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CMR1[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x0028 和 0x002B

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CMR2[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CMR2[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x0034 和 0x0037

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CMR3[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CMR3[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 25-6：PWM 比较器寄存器（PCMR）

CMR[15:0] — PWM 比较器寄存器

插入的数据范围：65535~0（单位：1 PWM 时钟周期）

CMR 用于确定 PWM 输出占空比。

假设：PWM 输出初始值为高电平

CMR > = CNR：PWM 输出始终为高电平

CMR < CNR： PWM 输出高电平 = （CMR + 1）单位

CMR = 0：PWM 输出高电平 = 1 单位

提示：

1. PWM 占空比 = CMR + 1。如果 CMR 等于零，则 PWM 占空比 = 1
2. 编程器可以随时将数据写入 CMR，且将在下一个 PWM 计数器周期生效。

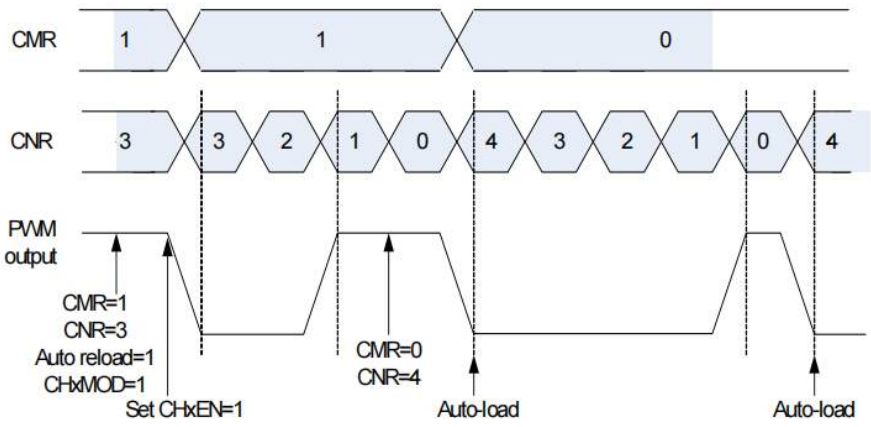


图 25-7：PWM 比较器寄存器设置时序

调节 PWM 控制器输出占空比（CNR = 150）

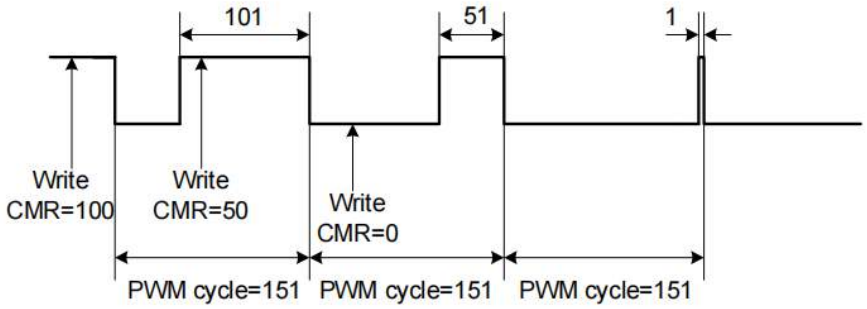


图 25-8：PWM 占空比

25.5.2.6. PWM 计时器寄存器（PTR0/1/2/3）

只读 PWM 计时器寄存器保存当前计数值，可随时读取，而不会干扰计数器。

偏移地址：0x0014 和 0x0017

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	PTR0[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	PTR0[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x0020 和 0x0023

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

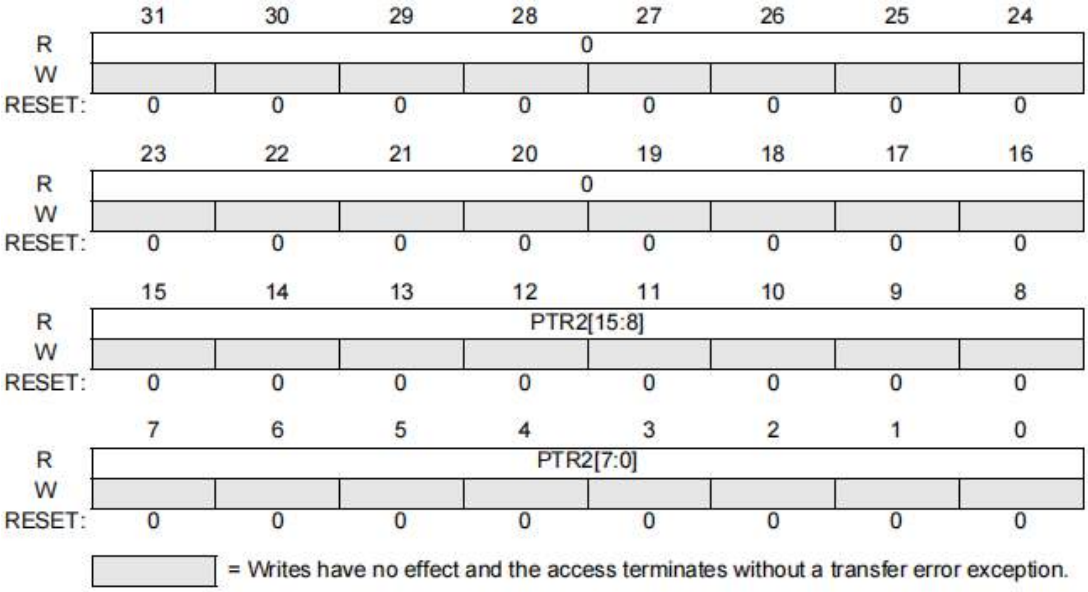
	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	PTR1[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	PTR1[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x002C 和 0x002F



偏移地址：0x0038 和 0x003B

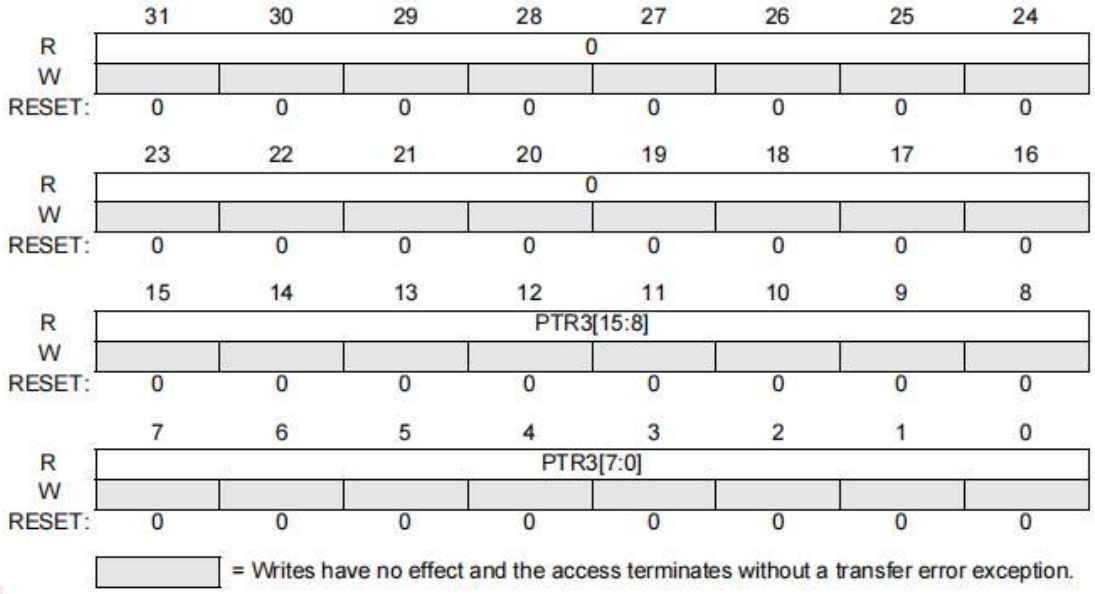


图 25-9：PWM 计时器寄存器（PTR）

PTR[15:0] — PWM 计时器值

只读 PTR 位保存当前计数值，用户可通过监测 PTR 了解 16 位向下计数器的当前值。

25.5.2.7. PWM 中断使能寄存器（PIER）

该寄存器用于启用 PWM 计时器中断。

偏移地址：0x003C 和 0x003F

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0				PIER3	PIER2	PIER1	PIER0
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 25-10：PWM 中断使能寄存器（PIER）

PIER3 — PWM 定时器 3 中断使能

1 = 启用

0 = 关闭

PIER2 — PWM 定时器 2 中断使能

1 = 启用

0 = 关闭

PIER1 — PWM 计时器 1 中断使能

1 = 启用

0 = 关闭

PIER0 — PWM 计时器 0 中断使能

1 = 启用

0 = 关闭

25.5.2.8. PWM 中断标志寄存器（PIFR）

该寄存器用于指示 PWM 计时器中断标志。

偏移地址：0x0040 和 0x0043

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0				PIFR3	PIFR2	PIFR1	PIFR0
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 25-11：PWM 中断标志寄存器（PIFR）

PIFR3 — PWM 计时器 3 中断标志。

当 PWM 计时器 3 的计数为 0 且 PIER3 = 1 时，PIFR3 将被设置为 1。将 1 写入该位将清除 PIFR3。

1 = 开关标志开启

0 = 中断标志关闭

PIFR2 — PWM 计时器 2 中断标志。

当 PWM 计时器 2 的计数为 0 且 PIER2 = 1 时，PIFR2 将被设置为 1。将 1 写入该位将清除 PIFR2。

1 = 中断标志开启

0 = 中断标志关闭

PIFR1 — PWM 计时器 1 中断标志。

当 PWM 计时器 1 的计数为 0 且 PIER1 = 1 时，PIFR1 将被设置为 1。将 1 写入该位将清除 PIFR1。

1 = 中断标志开启

0 = 中断标志关闭

PIFR0 — PWM 计时器 0 中断标志。

当 PWM 计时器 0 计数为 0 且 PIER0 = 1 时，PIFR0 将被设置为 1。将 1 写入该位将清除 PIFR0。

1 = 开关标志开启

0 = 中断标志关闭

25.5.2.9. PWM 捕获控制寄存器（PCCR0/1）

这些寄存器用于控制捕获功能。

偏移地址：0x0044 和 0x0047

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	CFLRD1	CRLRD1	0	CAPIF1	CAPCH1EN	FL_IE1	RL_IE1	INV1
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CFLRD0	CRLRD0	0	CAPIF0	CAPCH0EN	FL_IE0	RL_IE0	INV0
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x0048 和 0x004B

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	CFLRD3	CRLRD3	0	CAPIF3	CAPCH3EN	FL_IE3	RL_IE3	INV3
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CFLRD2	CRLRD2	0	CAPIF2	CAPCH2EN	FL_IE2	RL_IE2	INV2
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 25-12：PWM 捕获控制寄存器（PCCR0/1）

CFLRDx — 捕获下落锁存寄存器加载标志

1 = 当输入通道 x 发生下降沿转换时，CFLRx 被更新，并且该位为“1”。

0 = 当输入通道 x 没有下降沿时，写入 1 以清除该位。

CRLRDx — 捕获上升锁存寄存器加载标志

1 = 当输入通道 x 具有上升跳变时，CRLRx 被更新并且该比特为“1”。

0 = 当输入通道 x 没有上升沿时，写入 1 以清除该位。

CAPIFx — 通道 x 中断捕获标志

1 = 当输入通道 x 有下降沿且 FL&IEx 位启用时，设置此中断标志。当输入通道 x 有上升沿且 RL&IEx 位启用时，同样设置此中断标志。

0 = 未设置通道 x 中断标志。写入 1 以清除该位。

CAPCHxEN — 捕获通道 x 启用 / 禁用

1 = 启用

0 = 关闭

启用捕获功能时，会锁存 PMW 计数器并将其保存到 CRLR（上升沿锁存）和 CFLR（下降沿锁存）。禁用捕获功能时，不会更新 CRLR 和 CFLR，并且禁用通道 x 中断。

FL_IEx — 通道 x 掉线中断使能 ON/OFF

1 = 启用

0 = 关闭

启用时，如果 Capture 检测到通道 x 具有下降转换，Capture 发出中断。

RL_IEx — 通道 x 上升中断使能 ON/OFF

1 = 启用

0 = 关闭

启用时，如果 Capture 检测到通道 x 具有上升转换，则 Capture 发出中断。

INVx — 通道 x 变频器开/关

1 = 反相器打开

0 = 反相器关闭

25.5.2.10. PWM 捕获上升锁存寄存器（PCRLR0/1/2/3）

这些寄存器用于在捕获上升沿时锁存 PWM 计数器。

偏移地址：0x004C 和 0x004F

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CRLR0[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CRLR0[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x0054 和 0x0057

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CRLR1[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CRLR1[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x005C 和 0x005F

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CRLR2[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CRLR2[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x0064 和 0x0067

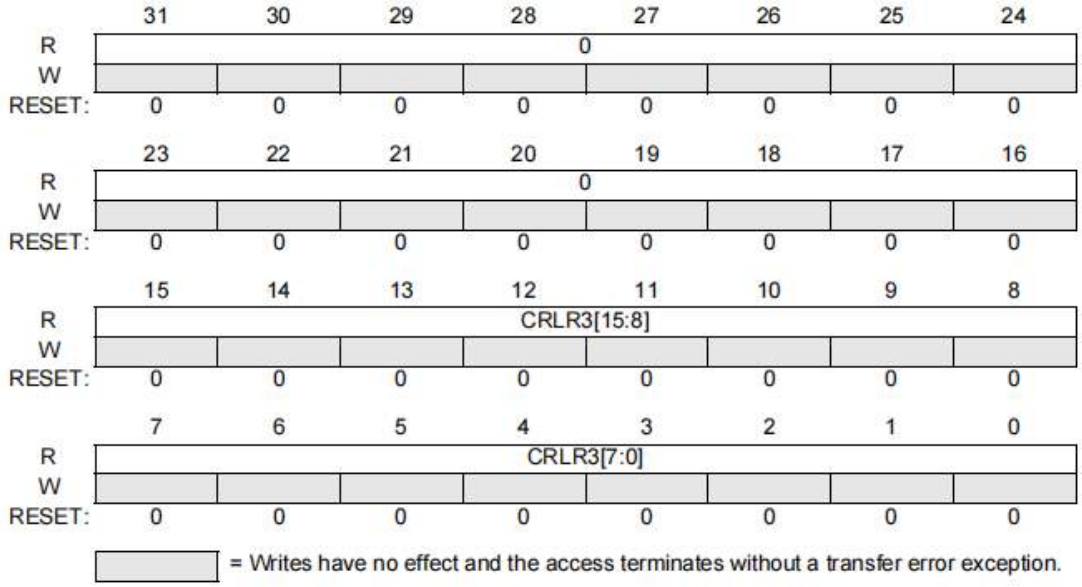


图 25-13：PWM 捕获上升锁存寄存器（PCRLR0/1/2/3）

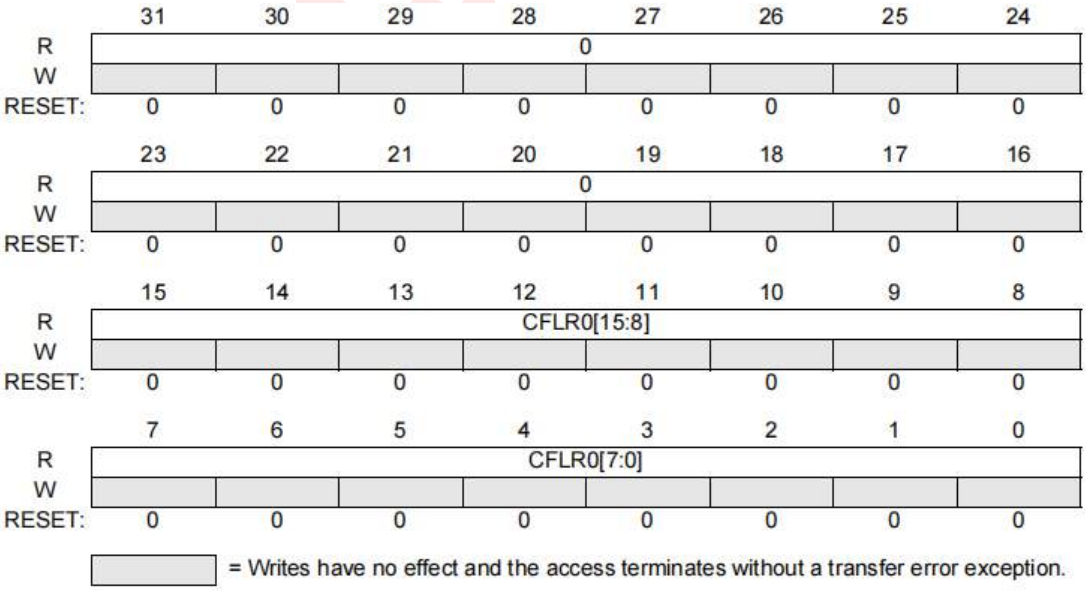
CRLRx[15:0] — 捕获上升锁存寄存器 x

当通道 x 出现上升跳变时，锁定 PWM 计数器。

25.5.2.11. PWM 捕获锁存寄存器（PCFLR0/1/2/3）

这些寄存器用于在捕获下降沿时锁存 PWM 计数器。

偏移地址：0x0050 和 0x0053



偏移地址：0x0058 和 0x005B

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CFLR1[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CFLR1[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x0060 和 0x0063

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	CFLR2[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	CFLR2[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

偏移地址：0x0068 和 0x006B

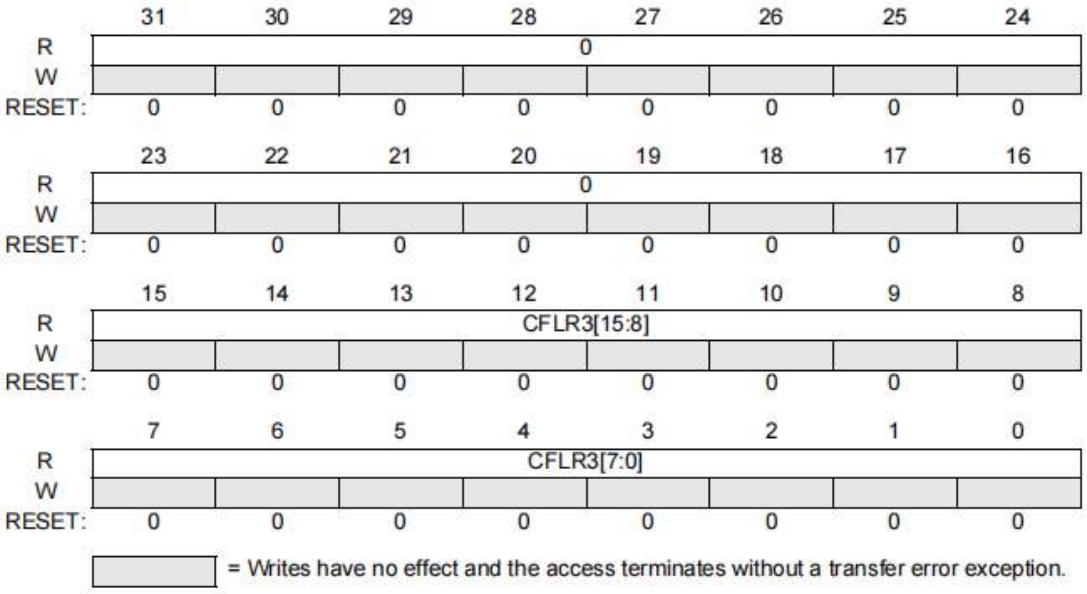


图 25-14：PWM 捕获下降锁存寄存器（PCFLR0/1/2/3）

CFLRx[15:0] — 捕获下降锁存寄存器 x

当通道 x 出现下降沿时，锁定 PWM 计数器。

25.5.2.12. PWM 端口控制寄存器（PPCR）

寄存器（PPCR）用于控制 PWMx 引脚方向和引脚状态。

偏移地址：0x006C 和 0x006F

	31	30	29	28	27	26	25	24
R	0							
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0				PDDR[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0				PULLUP_EN[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0				PDR[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 25-15：PWM 端口控制寄存器（PPCR）

提示：对于该 PWM 模块，PULLUP_EN 无意义。

PDDR[3:0] — 端口数据方向寄存器

PDDR[3:0] 位控制 PWM 引脚的方向。复位将清除 PDDR[3:0]。

1 = 配置为输出的对应引脚

0 = 配置为输入的对应引脚

PULLUP_EN[3:0] — 端口上拉使能（这些位在该芯片中未实现）

PULLUP_EN[3:0] 位控制 PWM 引脚的上拉特性。

1 = 启用上拉

0 = 禁用上拉功能

PDR[3:0] — 端口数据寄存器

数据仅在引脚被配置为通用输出时才会写入 PDR[3:0] 驱动引脚。读取输入(PDDR 位清除）将返回引脚电平。

25.6. 功能说明

本小节描述了 PWM 功能操作。

25.6.1. PWM 双缓冲和自动重载

PWM 计时器具备双缓冲功能，可在不中断当前计时器运行的情况下重新加载新计时值。即使设置了新的计时器数值，当前计时器仍能正常工作。计数器数值可写入 CNR0~3 寄存器，而当前计数值则可通过 PTR0~3 寄存器读取。自动重载功能会在下计数器归零时将数值从 CNR0~3 复制到下计数器。若 CNR0~3 寄存器被置为零，则计数器将在归零时停止运行；若自动重载位被置为零，则计数器会立即停止工作。

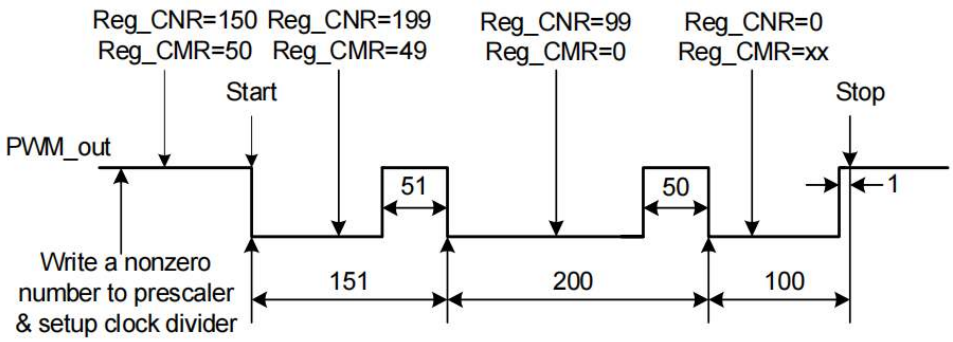


图 25-16：PWM 双缓冲器示意图

25.6.2. 调制占空比

双缓冲功能允许在当前周期的任何时间点写入 CMR。加载的值将从下一个周期开始生效。

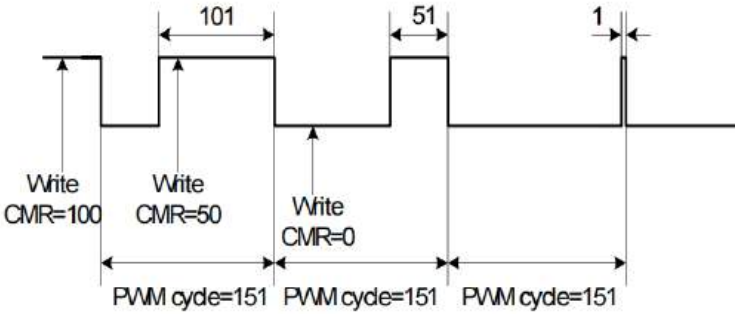


图 25-17：PWM 控制器输出占空比

25.6.3. 死亡区发生器

采用死区发生器实现 PWM，用于对功率器件进行保护。该功能可在 PWM 输出波形上升沿产生可编程的时间间隙，用户可对 PPR [31:24] 和 PPR [23:16] 进行编程，分别确定两个死区间隔。

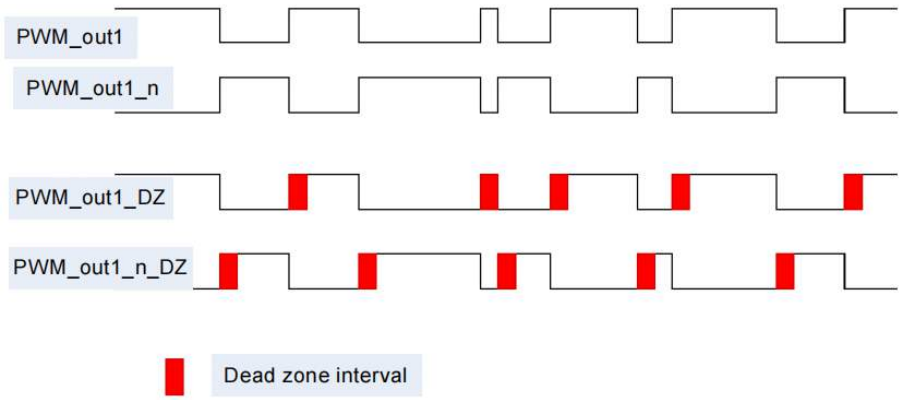


图 25-18：死区生成操作

25.6.4. PWM 计时器启动程序

1. 设置时钟选择器（CSR）
2. 设置预分频器和死区间隔（PPR）
3. 设置反相器开关、死区发生器开关、切换模式/单次模式和 PWM 计时器关闭（PCR）
4. 设置比较器寄存器（CMR）
5. 设置计数器寄存器（CNR）
6. 设置中断使能寄存器（PIER）
7. 将 PWMx 设置为输出引脚（PPCR）
8. 启用 PWM 计时器（PCR）

25.6.5. PWM 计时器停止程序

方法 1：

将 16 位下降计数器（CNR）设置为 0，并监测 PTR。当 PTR 达到 0 时，禁用 PWM 计时器（PCR）。（推荐）

方法 2：

将 16 位下降计数器（CNR）设置为 0。当发生中断请求时，禁用 PWM 计时器（PCR）。（推荐）

方法三：

直接禁用 PWM 计时器（PCR）。（不推荐）

25.6.6. 捕获启动程序

1. 设置时钟选择器（CSR）
2. 预缩率设置（PPR）
3. 设置反相器开关、死区发生器开关、自动加载模式/单次模式和 PWM 计时器关闭。（PCR）
4. 设置计数器寄存器（CNR）
5. 设置捕获寄存器（CCR）
6. 将 PWMx 设置为输入引脚（PPCR）
7. 启用 PWM 计时器（PCR）

25.6.7. 捕获基本计时器操作

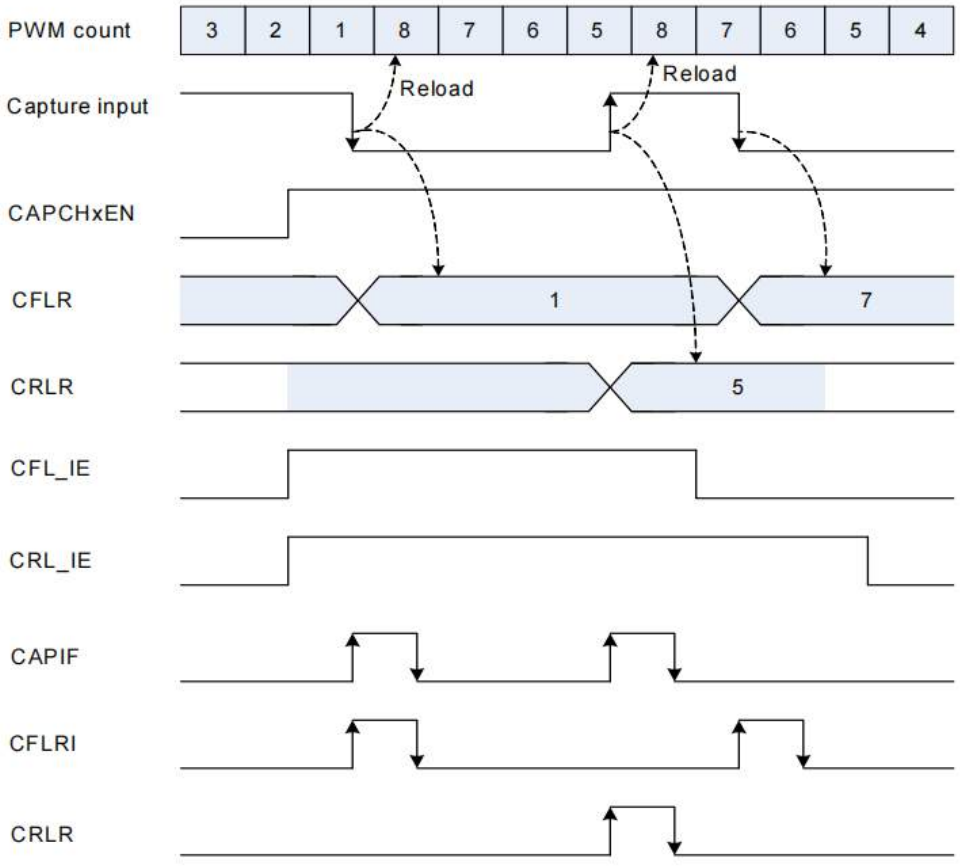


图 25-19：捕获基本定时器操作

在此情况下，CNR 为 8：

1.当 CAPIFx 设置为 1 时，将重新加载 PWM 计数器 CNRx。

2.信道低脉冲宽度为（CNR+1-CRLR）。

3.信道高脉冲宽度为（CNR+1-CFLR）

26. 比较器模块（COMP）

26.1. 介绍

该比较器提供可编程响应时间、滞后功能、模拟输入多路复用器，以及两个可选的端口引脚输出：同步“滤波”输出（CP）和异步“原始”输出（CPA）。即使在系统时钟未激活时，异步 CPA 信号仍可正常传输。这种设计使得比较器能够在设备处于 STOP 模式时持续运行并生成输出信号。

26.2. 方块图

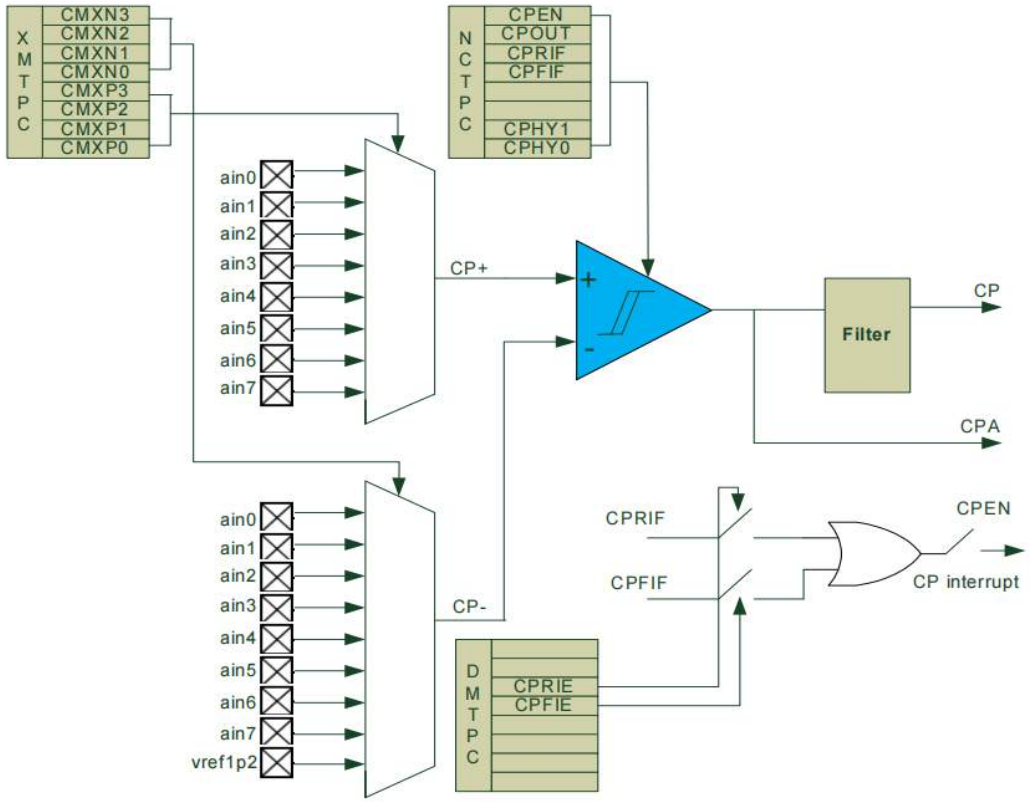


图 26-1：比较器方块图

26.3. 操作模式

本小节描述了三种低功耗模式。

26.3.1. 等待模式

在等待模式中，通过设置 CPEN 位，比较器模块可以继续正常运行，并且可以配置为通过生成中断请求来退出低功耗模式。

26.3.2. Doze 模式

在等待模式中，通过设置 CPEN 位，比较器模块可以继续正常运行，并且可以配置为通过生成中断请求来退出低功耗模式。

26.3.3. 停止模式

在停止模式下，系统时钟不存在，通过设置 CPEN 位，比较器模块可以继续正常运行，并且可以通过生成异步“原始”输出（CPA）唤醒信号来配置为退出低功耗模式。

26.4. 内存映射和寄存器

本小节描述了比较器的内存映射和寄存器结构。

26.4.1. 内存映射

关于内存映射的描述，请参见表 26-1。该设备有两个比较器模块。

表 26-1：比较器模块内存映射

地址偏移量	Bit[7:0]	访问权限 [1]
0x0003	比较器控制寄存器（CPTCN）	S/U
0x0002	比较器模式选择寄存器（CPTMD）	S/U
0x0001	比较器 MUX 选择寄存器（CPTMX）	S/U
0x0000	比较器输出滤波器选择寄存器（CPTFS）	S/U

提示：S = 仅允许访问 CPU 监控程序模式。S/U = 允许访问 CPU 监控程序或用户模式。用户模式对仅限于监控程序的地址的访问无效，并导致周期终止传输错误。

26.4.2. 寄存器

Comparator 编程模型包含以下寄存器：

- CPTCN：比较器控制寄存器。
- CPTMD：比较器模式选择寄存器。
- CPTMX：比较器 MUX 选择寄存器。
- CPTFLS：比较器输出滤波器选择寄存器。

26.4.2.1. 比较器控制寄存器

偏移地址：0x0003

	7	6	5	4	3	2	1	0
R W	CPEN	CPOUT	CPRIF	CPFIF	0	0	CPHY[1:0]	
RESET:	0	0	0	0	0	0	0	1

图 26-2：比较器控制寄存器（CPTCN）

CPEN — 比较器使能位

读写 CPEN 位启用比较器操作。当比较器被禁用时，CPOUT 处于低电平状态。

1 = 启用比较器

0 = 比对器已禁用

CPOUT — 比较器输出状态标志。

1 = CP 上的电压+>CP-。

0 = CP+上的电压<CP–。

CPRIF — 比较器上升沿标志。必须通过向该位写入数据的软件清除。

1 = 发生比较器上升沿。

0 = 自上次清除此标志后，未出现比较器上升边缘。

CPFIF — 比较器下降沿标志。必须通过软件写入该位来清除。

1 = 比较器下沿已出现。

0 = 无比较器自上次清除此标志后未发生下降边缘。

CPHY[1:0] — 比较器滞回控制位。

00：禁用滞后。

01：滞后 = 8 mV。

10：滞后 = 12 mV。

11：滞后 = 15.4 mV。

26.4.2.2. 比较器模式选择寄存器

偏移地址：0x0002

	7	6	5	4	3	2	1	0
R W	CPOEN	0	CPRIE	CPFIE	0	WKUP-LEVEL	0	CPMD
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 26-3：比较器模式选择寄存器（CPTMD）

CPOEN — 比较器输出与垫片控制位的比较。

1 = 在 COMP0 的 pwm0[0] 和 COMP1 的 pwm0[1] 中可以观察到比较器输出（COMP1 未在此设备中实现）。

0 = 禁用比较器输出。

CPRIE — 比较器上升沿中断使能。

1 = 比较器上升沿中断已启用。

0 = 比对器中断禁用。

CPFIE — 比较器下降沿中断使能。

1 = 比较器跌落边沿中断启用。

0 = 比对器后缘中断已禁用。

WKUPLEVEL — 比较器唤醒电平控制位。

1 = 在停止模式下，CP+上的电压高于 CP–时将产生唤醒请求。

0 = CP+上的电压低于 CP–将在停止模式期间产生唤醒请求。

CPMD — 比较器模式选择。这些位选择比较器的响应时间。

表 26-2：比较器模式选择

模式	CPMD	响应时间	耗电量
低速	0	500 纳秒	3.3 微安
高速	1	80 纳秒	22 微安

26.4.2.3. 比较器 MUX 选择寄存器

偏移地址：0x0001

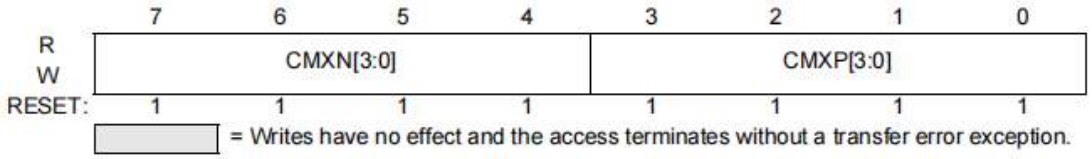


图 26-4：比较器 MUX 选择寄存器（CPTMX）

CMXN[3:0] — 比较器负输入多路复用器选择。这些位选择哪个端口引脚用作比较器负输入。

表 26-3：比较器负输入 MUX 选择

CMXN[3]	CMXN[2]	CMXN[1]	CMXN[0]	Negative Input
0	0	0	0	Ain0
0	0	0	1	Ain1
0	0	1	0	Ain2
0	0	1	1	Ain3
0	1	0	0	Ain4
0	1	0	1	Ain5
0	1	1	0	Ain6
0	1	1	1	Ain7
1	0	0	0	Vref1p2
Other value				None

CMXP[3:0] — 比较器正输入多路复用器选择。这些位选择哪个端口引脚用作比较器的正输入。

表 26-4：比较器正输入 MUX 选择

CMXN[3]	CMXN[2]	CMXN[1]	CMXN[0]	Positive Input
0	0	0	0	Ain0
0	0	0	1	Ain1
0	0	1	0	Ain2
0	0	1	1	Ain3
0	1	0	0	Ain4
0	1	0	1	Ain5
0	1	1	0	Ain6
0	1	1	1	Ain7
1	x	x	x	None

26.4.2.4. 比较器输出滤波器选择寄存器

偏移地址：0x0000

	7	6	5	4	3	2	1	0
R W	CMFLEN	CMFL_WIDTH[6:0]						
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error exception.

图 26-5：比较器输出滤波器选择寄存器（CPTFLS）

CMFLEN — 比较器输出数字滤波器启用

1 = 比较器输出数字滤波器启用，CPMRIF 和 CPMFIF 由滤波器输出生成；

0 = 比较器输出数字滤波器禁用，CPMRIF 和 CPMFIF 由原始输出生成；

CMFL_WIDTH[6:0] — 比较器输出数字滤波器脉冲宽度选择。

CMFL_WIDTH[6:0] 确定输入脉冲的宽度将被滤波。如果输入脉冲宽度小于（CMFL_width[6:0]+2）*fips 的周期，则该脉冲将被滤波。

26.5. 功能说明

比较器的输入端口在 CPTMX 寄存器中进行配置。CMXP3 ~ CMXP0 位用于选择比较器的正输入端口，CMXN3-CMXN0 位则用于选择比较器的负输入端口。关于比较器输入端口的重要说明：被选作比较器输入的端口引脚需在对应的端口配置寄存器中设置为模拟输入模式，并且必须禁用输入、输出、上拉和下拉控制功能。

比较器输出信号可通过软件轮询、作为中断源使用，或传输至端口引脚。当信号传输至端口引脚时，其输出可与系统时钟异步或同步；即使处于 STOP 模式（无系统时钟激活）下，异步输出仍保持可用。当禁用时，比较器输出默认处于低电平状态，其供电电流将降至 100 纳安以下。

比较器迟滞可通过其比较器控制寄存器 CPTCN 进行软件编程。用户可对迟滞电压的量进行编程。比较器迟滞的编程使用比较器控制寄存器 CPTCN 中的 Bits1 ~ 0 完成。

比较器中断可触发于输出信号的上升沿或下降沿。当比较器出现下降沿时，CPFIF 标志位会被置为逻辑 1；当出现上升沿时，CPRIF 标志位则被置为逻辑 1。这些标志位一旦被置位，将持续保持状态直至被软件清除。要启用比较器上升沿中断屏蔽功能，需将 CPRIE 设置为逻辑 1；若需启用比较器下降沿中断屏蔽功能，则需将 CPFIE 设置为逻辑 1。

通过读取 CPOUT 位，可以随时获得比较器的输出状态。比较器的启用是通过将 CPEN 位设置为逻辑 1 来实现的，而禁用则是通过将该位清零来实现的。

需要注意的是，当比较器首次通电或对迟滞控制位进行修改时，可能会检测到虚假上升沿和下降沿。因此建议在比较器启用后或其模式位变更后的短时间内，将上升沿和下降沿标志位显式清零至逻辑 0。这种情况下，比较器的最小启动时间可缩短至 5 微秒以内。

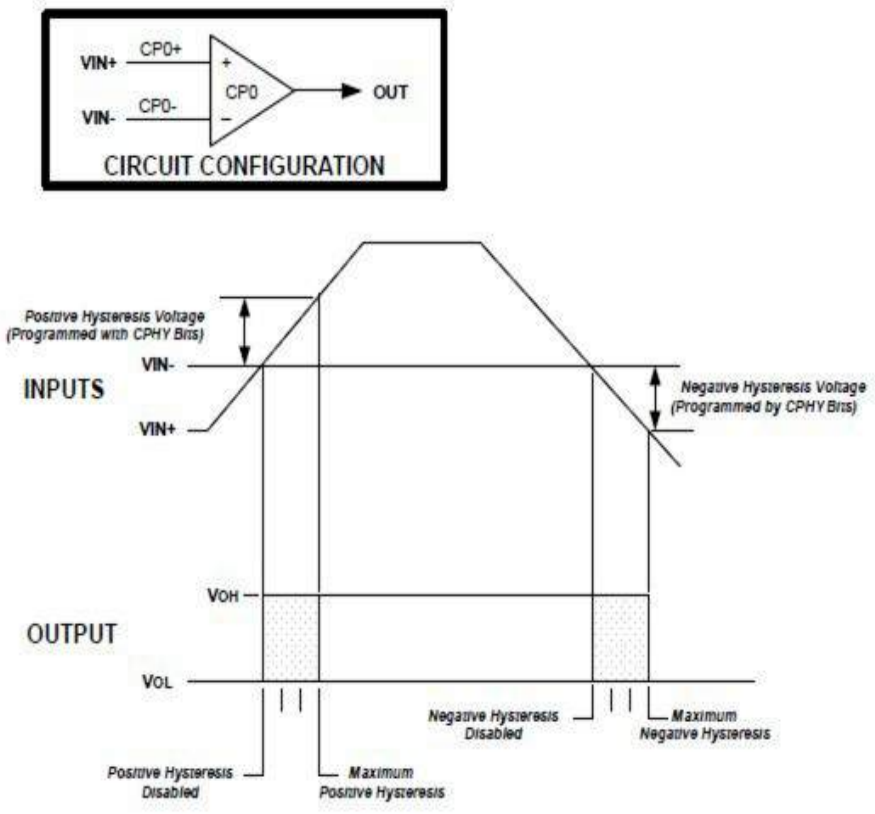


图 26-6：比较器滞回曲线

27. 模数转换器（ADC）

27.1. 介绍

这款 12 位模数转换器采用逐次逼近式工作原理，配备多达 4 个通道，可同时采集 3 个外部信号源和 1 个内部信号源的数据。各通道支持单次连续扫描或非连续扫描等多种模式进行模数转换。转换结果存储于 12 位×8 深度的先进先出（FIFO）存储器中，数据格式支持左对齐或右对齐两种配置。

模拟监督功能允许应用程序检测输入电压是否超出用户定义的较高或较低阈值。

实现了高效的低功耗模式，以允许在低频率下极低的消耗。

27.2. ADC 主要特性

- 高性能
  - 可配置的分辨率：12 位、10 位、8 位或 6 位
  - ADC 转换时间：1.0 μs 用于 12 位分辨率（1 MHz），0.88 μs 转换时间用于 10 位分辨率，通过降低分辨率可以获得更快的转换时间。
  - 可编程采样时间
  - 通过内置数据一致性实现数据对齐
  - DMA 支持
- 低功耗
  - 应用程序可降低 PLCK 频率以进行低功耗操作，同时仍保持最佳 ADC 性能。例如，无论 PCLK 的频率如何，都保持 1.0 μs 的转换时间。
  - 等待模式：在 PLCK 频率较低的应用中防止 ADC 溢出
  - 自动关闭模式：除在活动转换阶段外，ADC 自动关闭。这大大降低了 ADC 的功耗。
- 模拟输入通道
  - 8 个外部模拟输入
  - 1 个通道用于内部参考电压
  - 1 个通道用于内部温度传感器
- 可启动转换开始：
  - 通过软件
  - 采用极性可配置的硬件触发器
- 转换模式
  - -可以转换单个通道，也可以扫描一系列通道。

– 单模式下，每个触发器会转换一次选定的输入

– 持续模式可持续转换所选输入

– 不连续模式

- 在采样结束、转换结束、序列转换结束以及模拟看门狗或超频事件发生时中断生成。
- 电压监测器
- 单端和差分输入配置
- 转换器使用内部参考或外部参考

27.3. ADC 功能描述

图 27-1 显示了 ADC 方块图。

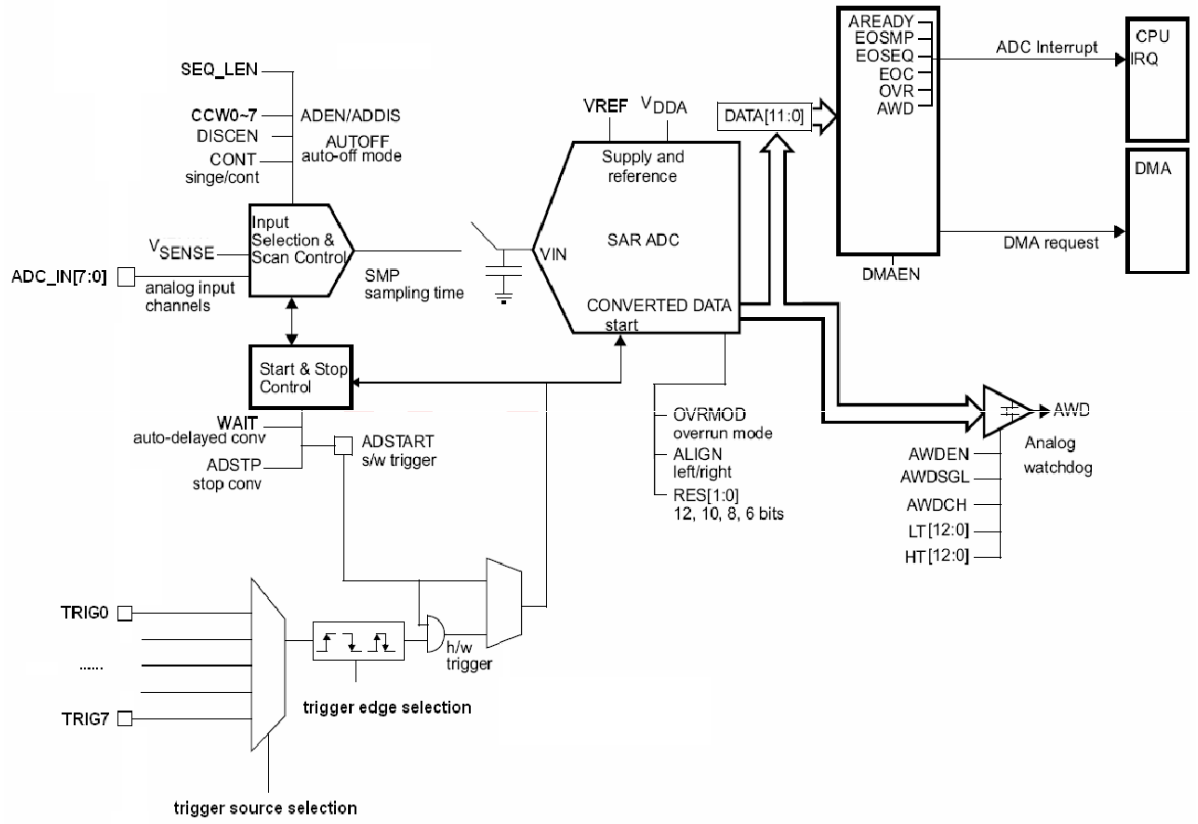


图 27-1：ADC 方块图

27.3.1. ADC 开关控制（ADEN、ADDIS、ADRDY）

MCU 上电时，ADC 处于禁用状态并进入断电模式（ADEN = 0），如图 27-2 所示，ADC 需要 tSTAB（~2.0 μs）的稳定时间才能开始精确转换。

两个控制位用于启用或禁用 ADC：

- 设置 ADEN = 1 以启用 ADC。一旦 ADC 就绪，ADRDY 标志即被设置。
- 将 ADDIS = 1 禁用 ADC 并将其置于断电模式。随后，ADEN 和 ADDIS 位将由硬件自动清除，一旦 ADC 完全禁用。

转换可通过设置 ADSTART = 1 开始，或在启用触发器时通过外部触发事件开始。

按照以下步骤启用 ADC：

- 在 ADC_CR 寄存器中设置 ADEN = 1。
- 直到 ADC_ISR 寄存器中的 ADRDY = 1（ADRDY 是在 ADC 启动时间后设置的）。如果通过在 ADC_IER 寄存器中设置 ADRDYIE 位来启用中断，则可以处理这种情况。

按照以下步骤禁用 ADC：

- 检查 ADC_CR 寄存器中的 ADSTART = 0，以确保没有正在进行的转换。如果需要，通过向 ADC_CR 寄存器中的 ADSTP 位写入 1 并等待该位读取为 0 来停止任何正在进行的转换。
- 在 ADC_CR 寄存器中设置 ADDIS = 1。
- 如果应用程序要求，则等待，直到 ADC_CR 寄存器中的 ADEN = 0，这表示 ADC 已完全禁用（一旦 ADEN = 0，ADDIS 将自动复位）。

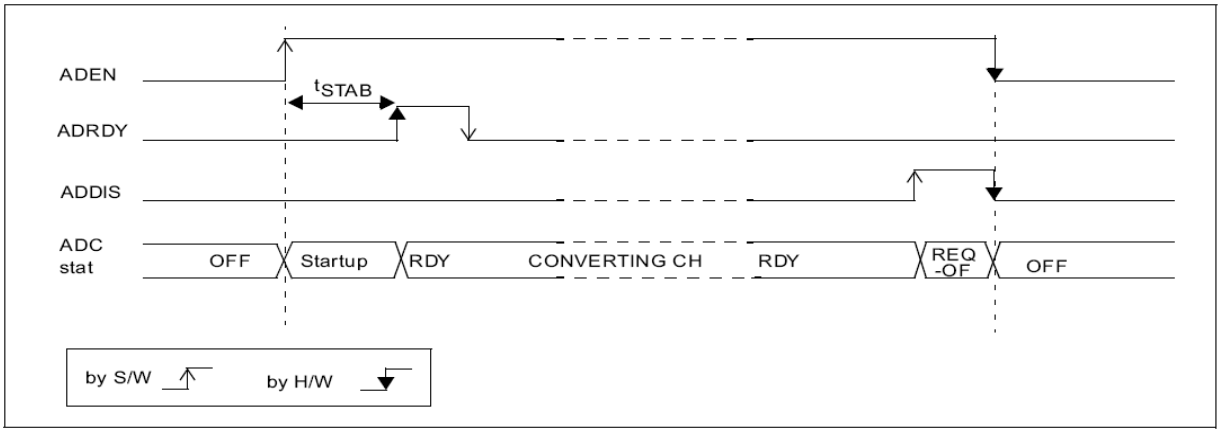


图 27-2：启用 / 禁用 ADC

提示：在自动关闭模式（AUTOFF = 1）下，通过硬件自动执行通电/断电阶段，且 ADRDY 标志未设置。

27.3.2. ADC 时钟

如图 27-3 所示，ADC 具有双时钟域架构，其优势在于，无论选择哪种 IPG 时钟方案，都能达到 ADC 的最大时钟频率。

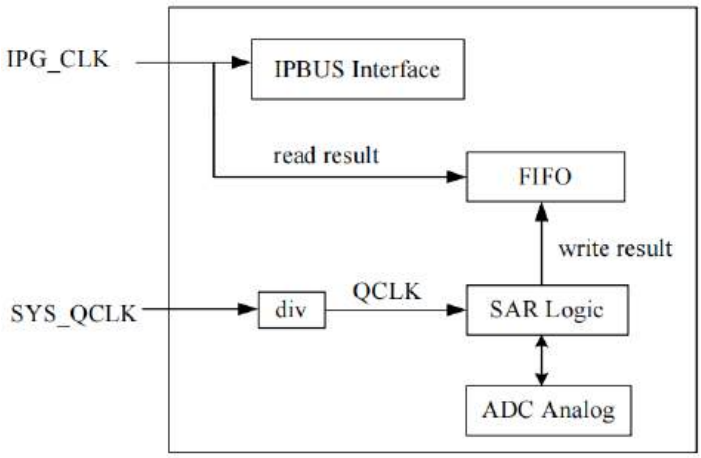


图 27-3：ADC 时钟方案

27.3.3. 配置 ADC

如果 ADC 处于禁用状态(ADEN 必须为 0），软件必须写入 ADC_CR 寄存器中的 ADEN 位。

只有在 ADC 被启用且没有待处理的禁用 ADC 请求（ADEN = 1 且 ADDIS = 0）时，软件才可将数据写入 ADC_CR 寄存器中的 ADSTART 和 ADDIS 位。

对于 ADC_IER、ADC_CFGRi、ADC_SMPR、ADC_TR、ADC_CHSELRi 和 ADC_WDG 寄存器中的所有其他控制位，软件必须仅在没有正在进行的转换时（ADSTART = 0）写入配置控制位。

如果 ADC 已启用（可能正在转换），并且没有待处理的禁用 ADC 请求（ADSTART = 1 且 ADDIS = 0），则软件只能将数据写入 ADC_CR 寄存器中的 ADSTP 位。

提示：没有硬件保护措施可以防止软件执行上述规则禁止的写入操作。如果发生此类禁止的写入访问，ADC 可能进入未定义状态。要恢复这种情况下的正确操作，必须禁用 ADC（ADDIS = 1）。

27.3.4. 信道选择（CCWi）

最多有 10 个多路复用通道：

- 从引脚（ADC_IN0...ADC_IN7）有 8 个模拟输入
- 2 个内部模拟输入(VREFINT 和温度传感器)

可以转换单个通道，也可以自动扫描一系列通道。

待转换通道的序列按 CCW[0]、CCW[1]，...，CCW[7] 的顺序排列。CCWi 参数已预设在 ADC_CHSELRi 模块中，其序列长度由 ADC_CFGR2 寄存器的 SEQ_LEN[2:0] 位进行编程设置。例如当序列长度设为 3 时，通道排列顺序将依次为 CCW[0]、CCW[1]、CCW[2]。

信道解码如表 27-1 所示。

表 27-1：信道编码

CCWi[3:0]	通道选择
4'b0000	ADC_IN0
4'b0001	ADC_IN1
4'b0010	ADC_IN2
4'b0011	ADC_IN3
4'b0100	ADC_IN4
4'b0101	ADC_IN5
4'b0110	ADC_IN6
4'b0111	ADC_IN7
4'b1110	VREFINT
4'b1111	Temperature Sensor

27.3.5. 可编程采样时间（SMP）

在开始转换之前，ADC 需要建立被测电压源和 ADC 的嵌入式采样电容之间的直接连接，这个采样时间必须足够让输入电压源给采样保持电容充电到输入电压电平。

采用可编程采样时间，可以根据输入电压源的输入电阻来调整转换速度。ADC 会根据 ADC_SMPR 寄存器中的 SMP[3:0] 位对输入电压进行采样，该采样周期数可以进行修改。所有通道都具有相同的可编程采样时间。

ADC 通过设置 EOSMP 标志来指示采样阶段的结束。

27.3.6. 单转换模式（CONT = 0）

在单通道转换模式下，ADC 对序列进行一次转换。当 ADC_CFGR1 寄存器中的 CONT = 0 时，系统选择该模式。转换可通过以下任一方式启动：

- 在 ADC_CR 寄存器中设置 ADSTART 位
- 硬件触发事件

在序列中，每次转换完成后：

- 转换后的数据存储在 FIFO 中

- EOC（转换结束）标志被设置
- 若 EOCIE 位被设置，则生成中断

完成转换序列后：

- EOSEQ（序列结束）标志被设置
- 若 EOSEQIE 位被设置，则会生成中断

然后，ADC 停止，直到发生新的外部触发事件或 ADSTART 位再次被设置。

提示：要转换单通道一次，用长度为 1 的序列进行编程。

27.3.7. 连续转换模式（CONT = 1）

在连续转换模式下，当发生软件或硬件触发事件时，ADC 执行一系列转换，转换一次后自动重新启动并持续执行相同的转换序列。当 ADC_CFGR1 寄存器中的 CONT = 1 时，选择此模式。转换可通过以下任一方式启动：

- 在 ADC_CR 寄存器中设置 ADSTART 位
- 硬件触发事件

在序列中，每次转换完成后：

- 转换后的数据存储在 FIFO 中
- EOC（转换结束）标志被设置
- 若 EOCIE 位被设置，则生成中断

完成转换序列后：

- EOSEQ（序列结束）标志被设置
- 若 EOSEQIE 位被设置，则会生成中断

然后，新的序列立即重新开始，ADC 持续重复转换序列。

提示：无法同时启用间断模式和连续模式：

禁止同时设置 DISCEN = 1 和 CONT = 1 位。

27.3.8. 启动转换（ADSTART）

软件通过设置 ADSTART = 1 开始 ADC 转换。当 ADSTART 被设置时，转换：

- TRIGMODE = 0x0（软件触发）时立即启动
- 在选定硬件触发器的下一个有效边沿，如果 TRIGMODE = 0x0

ADSTART 位还用于指示当前是否正在进行 ADC 操作。当 ADSTART = 0 时，可以重新配置 ADC，表示 ADC 处于空闲状态。

通过硬件清除 ADSTART 位：

- 单模式，带软件触发
  – 在转换序列的任何末端（EOSEQ = 1）
- 在软件触发的间断模式下
  – 在转换的任何末端
- 有情况
  – 执行软件调用的 ADSTP 过程之后

提示：

- 在连续模式（CONT = 1）下，当 EOSEQ 标志被设置时，ADSTART 位不会被硬件清除，因为序列会自动重新启动。
- 在单模式下选择硬件触发器时，当 EOSEQ 标志被设置时，ADSTART 不会被硬件清除。这避免了软件必须重新设置 ADSTART 位的需要，并确保不会错过下一个触发事件。

27.3.9. 时序

转换开始和转换结束之间的经过时间是配置的采样时间加上根据数据分辨率而定的逐次逼近时间之和：

tADC = tSMPL + tSAR = [4|min + 12|12bit] x tQCLK = 1µs |min (for fQCLK = 16MHz)

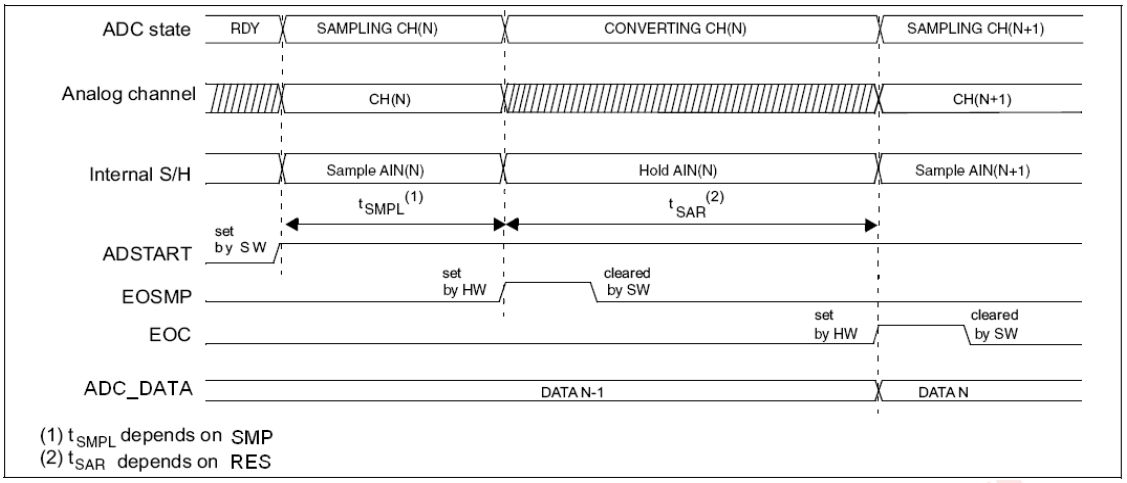


图 27-4：模数转换时间

27.3.10. 停止正在进行的转换（ADSTP）

软件可通过在 ADC_CR 寄存器中设置 ADSTP = 1 来决定是否停止任何正在进行的转换。

这将重置 ADC 操作，ADC 将处于空闲状态，准备执行新操作。

当软件设置 ADSTP 位时，任何正在进行的转换都会中止，并且结果会被丢弃(FIFO 不会用当前转换更新）。扫描序列也会被中止和重置（这意味着重新启动 ADC 将重新开始一个新序列）。

完成该过程后，ADSTP 和 ADSTART 位将通过硬件清除，软件必须等待 ADSTART = 0 后才能开始新的转换。

提示：QADC_ISR 中的标志不会被 STOP 命令清除，FIFO 中的数据也不会丢失。

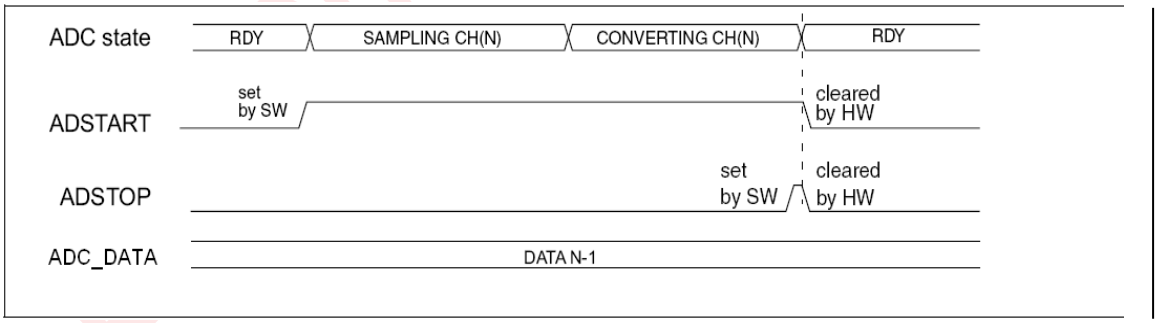


图 27-5：停止正在进行的转换

27.4. 外部触发器和触发极性的转换（TRIGMODE，TRIGSCR）

转换或一系列转换可以由软件或外部事件触发。如果 TRIGMODE 控制位不等于“0”，则外部事件能够触发具有选定极性的转换。一旦软件设置了 ADSTART = 1 位，触发选择即生效。

转换进行期间发生的任何硬件触发器都将被忽略。如果位 ADSTART = 0，则将忽略发生的任何硬件触发器。

表 27-2 提供了 TRIGMODE 值与触发极性之间的对应关系。

表 27-2：配置触发极性

TRIGMODE[2:0]	来源
3’b000	Trigger detection disabled, software trigger
3’b001	Detection on rising edge
3’b010	Detection on falling edge
3’b011	Detection on both rising and falling edges
3’b100	Detection on high level voltage
3’b101	Detection on low level voltage
3’b110	Detection on PIT 0
3’b111	Detection on PWM 0

提示：外部触发器的极性只能在 ADC 未转换时改变（ADSTART = 0）。

TRIGSCR 控制位用于选择 8 种可能的事件中哪一种可以触发转换。表 27-3 给出了常规转换的可能外部触发器。软件源触发事件可通过设置 ADC_CR 寄存器中的 ADSTART 位来生成。

表 27-3：外部触发器

TRIGSCR[2:0]	Name	来源
3’b000	TRG0	EPORT1[0]
3’b001	TRG1	EPORT1[1]
3’b010	TRG2	EPORT1[2]
3’b011	TRG3	EPORT1[3]
3’b100	TRG4	EPORT1[4]
3’b101	TRG5	EPORT1[5]
3’b110	TRG6	EPORT1[6]
3’b111	TRG7	EPORT1[7]

提示：仅当 ADC 未转换时（ADSTART = 0）才能更改触发器选择。

27.4.1. 间断模式（DISCEN）

通过在 ADC_CFGR1 寄存器中设置 DISCEN 位，可启用此模式。

在此模式下（DISCEN = 1），需要硬件或软件触发事件来启动序列中定义的每个转换。相反，如果 DISCEN = 0，则通过单个硬件或软件触发事件连续启动序列中定义的所有转换。示例：

- DISCEN = 1，待转换的通道 = 0、3、7、10
  - 第 1 次触发：通道 0 被转换，并且生成 EOC 事件
  - 第二触发：通道 3 被转换，并且生成 EOC 事件
  - 第 3 个触发器：通道 7 被转换，并且生成 EOC 事件
  - 第 4 个触发器：通道 10 被转换，并且 EOC 和 EOSEQ 事件均被生成。
  - 第 5 个触发器：通道 0 被转换并生成 EOC 事件
  - 第 6 个触发器：通道 3 被转换并生成 EOC 事件
  - ...
- DISCEN = 0，待转换的通道 = 0、3、7、10
  - 第 1 次触发：完整序列被转换：通道 0，然后是 3、7 和 10。每次转换都会产生一个 EOC 事件，最后一个转换还会产生一个 EOSEQ 事件。
  - 任何后续的触发事件都将重新启动整个序列。

27.4.2. 可编程分辨率（RES）-快速转换模式

通过降低 ADC 分辨率，可以实现更快的转换时间（tSAR）。该分辨率可通过编程 ADC_CFGR1 寄存器中的 RES[1:0] 位设置为 12、10、8 或 6 位。在不需要高数据精度的应用场景中，较低的分辨率可显著缩短转换时间。

提示：仅当 ADEN 位复位时，才必须更改 RES[1:0] 位。

转换结果的宽度始终为 13 位，且任何未使用的 LSB 位均被读取为零。

较低的分辨率可减少连续近似步骤所需的转换时间。

27.4.3. 转换结束，采样阶段结束(EOC、EOSMP 标志)

ADC 指示每个转换（EOC）事件的结束。

一旦获得新的转换数据结果，ADC 就会在 ADC_ISR 寄存器中设置 EOC 标志。如果 ADC_IER 寄存器中的 EOCIE 位被设置，则会生成中断。软件可以通过向该标志写入 1 或读取 FIFO 来清除 EOC 标志。

ADC 还通过在 ADC_ISR 寄存器中设置 EOSMP 标志来指示采样阶段的结束。软件可通过向 EOSMP 标志写入 1 来清除该标志。如果 ADC_IER 寄存器中的 EOSMPIE 位被设置，则可以生成中断。

27.4.4. 转换序列结束(EOSEQ 标志)

ADC 将通知应用程序每个序列结束（EOSEQ）事件。

当 FIFO 中出现转换序列的最后一个数据结果时，ADC 会在 ADC_ISR 寄存器中设置 EOSEQ 标志。如果 ADC_IER 寄存器中的 EOSEQIE 位被设置，则会生成中断。软件通过向 EOSEQ 标志写入 1 来清除该标志。

27.4.5. 示例时序图（单/连续模式硬件 / 软件触发器）

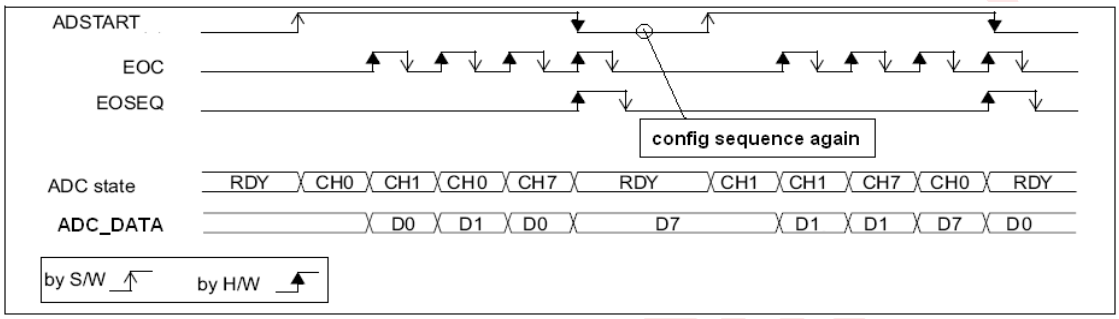


图 27-6：序列的单次转换，软件触发

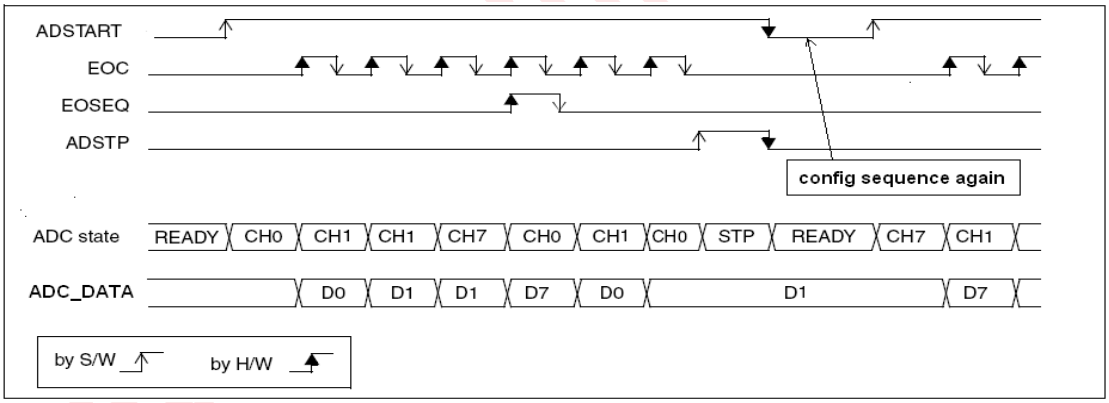


图 27-7：序列的连续转换，软件触发

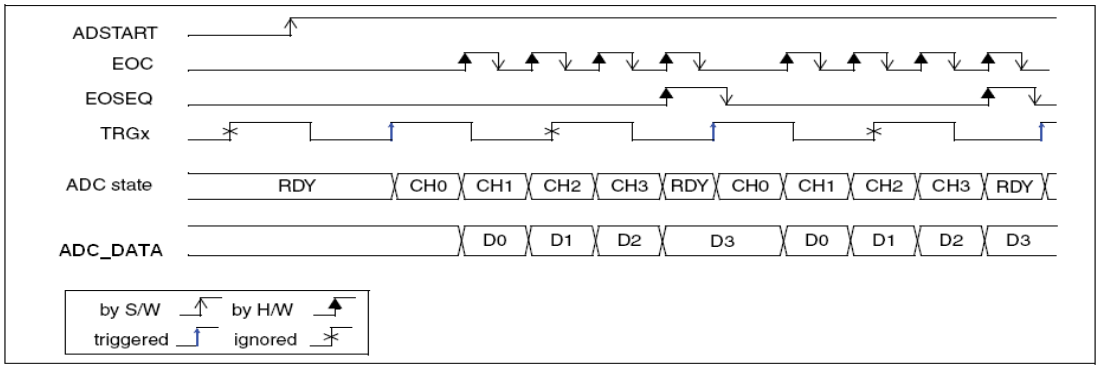


图 27-8：序列的单次转换，硬件触发

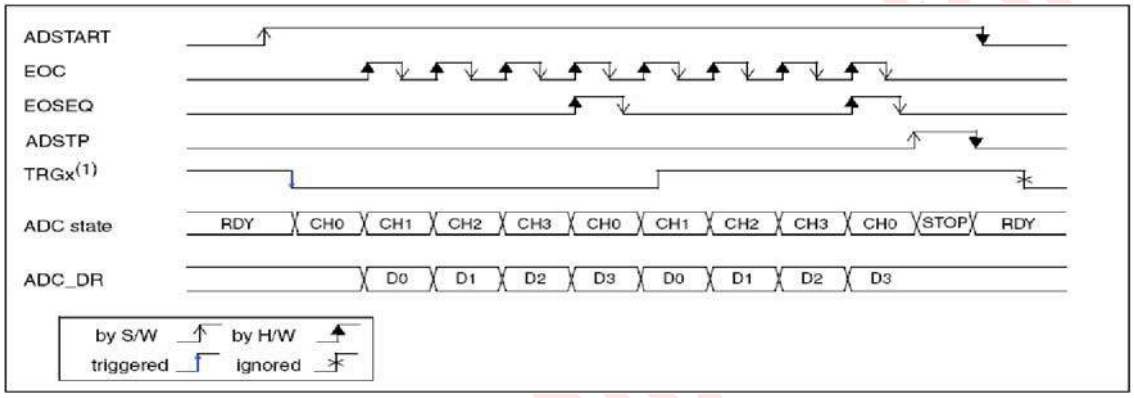


图 27-9：序列的连续转换，硬件触发

27.5.数据管理

27.5.1. 数据 FIFO 和数据对齐（ADC_FIFO，ALIGN）

每次转换结束时（当 EOC 事件发生时），将转换数据的结果存储在 ADC_FIFO 中，其宽度为 13 位 x 深度为 8。

读出数据的格式取决于配置的数据对齐和分辨率。

ADC_CFGR1 寄存器中的 ALIGN 位选择转换后存储数据的对齐方式。如图 27-10 所示，数据可以右对齐（ALIGN = 0）或左对齐（ALIGN = 1）。

对齐	返回[1:0]	31	30	…	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0x0			…					数据[11:0]											
	0x1			…							数据[9:0]									
	0x2			…									数据[7:0]							
	0x3			…											数据[5:0]					
1	0x0			…	数据[11:0]															
	0x1			…	数据[9:0]															
	0x2			…									数据[7:0]							
	0x3			…									数据[5:0]							

图 27-10：数据对齐和分辨率

FIFO 支持字节、半字和字的读取，但地址偏移量应始终为 0x004C。

对于不同的数据对齐和分辨率，用户应注意：

- 如果按字进行读取，则数据格式为 data[31:0]，如图 27-10 所示
- 若按半字读取，则数据格式如图 27-10 所示，即 data[15:0]
- 当数据长度超过 8 位时，若按字节读取，则首先读出高字节，然后再次读取低字节。
- 当数据长度不超过 8 位时，若按字节读取，则数据格式为 data[7:0]，如图 27-10 所示

27.5.2. ADC 溢出（OVR、OVRMOD）

溢出标志（OVR）表示数据溢出事件，当 CPU 或 DMA 在 FIFO 满之前未及时读取转换后的数据时。

如果在新转换完成时 FULL 标志仍为“1”，则 ADC_ISR 寄存器中将设置 OVR 标志。如果 ADC_IER 寄存器中设置了 OVRIE 位，则可以生成中断。

当发生溢出条件时，ADC 保持运行并可继续转换，除非软件决定停止并通过在 ADC_CR 寄存器中设置 ADSTP 位来重置序列。

软件通过向 OVR 标志写入 1 来清除该标志。

通过编程 ADC_CFGR1 寄存器中的 OVRMOD 位，可以配置发生溢出事件时是否保留或覆盖数据：

- OVRMOD = 0
  - 超频事件可防止数据寄存器被覆盖：旧数据得以保留，而新的转换结果将被丢弃。如果 OVR 保持为 1，则可以执行进一步的转换，但产生的数据将被丢弃。
- OVRMOD = 1
  - 数据寄存器被最后的转换结果覆盖。如果 OVR 保持为 1，则可以执行进一步的转换，并且 FIFO 始终包含来自最新转换的数据。

27.5.3. 管理不使用 DMA 转换的数据序列

如果转换速度足够慢，转换序列可以由软件处理。在这种情况下，软件可以使用 EOC 标志及其相关中断来处理每个数据结果。每次转换完成后，ADC_ISR 寄存器中的 EOC 位将被设置，并且可以读取 FIFO 寄存器。

软件也可以使用 FIFO EMPTY 标志来处理每个数据结果。如果 EMPTY 不是“0”，则表示 FIFO 有新数据。

应将 ADC_CFGR1 寄存器中的 OVRMOD 位配置为 0，以便将超频事件管理为错误。

27.5.4. 在不使用 DMA 的情况下管理已损坏数据而不发生溢出

在不需要每次转换后读取数据的情况下，让 ADC 转换一个或多个通道可能会很有用。在这种情况下，必须将 OVRMOD 位配置为 1，并且软件应忽略 OVR 标志。当 OVRMOD = 1 时，过载事件不会阻止 ADC 继续转换，FIFO 中始终包含最新的转换数据。

27.5.5. 使用 DMA 管理已转换数据

一旦 FIFO 中的数据编号不为空且位 DMAEN 被设置，QADC 将向 DMA 发送请求。这允许将转换后的数据从 FIFO 传输到软件所选的目标位置。

但是，如果由于 DMA 不能及时处理 DMA 传输请求而发生溢出（OVR = 1），则 ADC 停止生成 DMA 请求，且 DMA 不传输与新转换对应的数据，这意味着传输到 RAM 的所有数据都可以视为有效。

根据 OVRMOD 位的配置，数据要么被保留，要么被覆盖。

DMA 传输请求被阻塞，直到软件清除 OVR 位。

27.6. 低功率功能

27.6.1. 等待模式转换

等待模式转换可用于简化软件并优化低频运行的应用程序的性能，其中可能会发生 ADC 溢出的风险。当 ADC_CFGR1 寄存器中的 Wait 位设置为 1 时，只有在 FIFO 未满时才能开始新的转换。

这是一种自动调整 ADC 速度以适应读取数据的系统速度的方法。

提示：转换进行期间或读取访问之前等待时间内发生的任何硬件触发器均被忽略。

27.6.2. 自动关闭模式（AUTOFF）

ADC 具有自动电源管理功能，称为自动关闭模式，通过在 ADC_CFGR1 寄存器中设置 AUTOFF = 1 启用。

当 AUTOFF = 1 时，ADC 在不进行转换时始终处于断电状态，并且在开始转换时（由软件或硬件触发）自动唤醒。在启动转换的触发事件和 ADC 的采样时间之间自动插入一个启动时间。一旦转换序列完成，ADC 将自动禁用。

自动关闭模式可使需要相对较少转换的应用程序的功耗大幅降低，或者当转换请求间隔足够长（例如使用低频硬件触发器）时，可以证明用于打开和关闭 ADC 的额外功率和时间是合理的。

对于低频运行的应用程序，自动关机模式可与等待模式转换（WAIT = 1）结合使用。如果 ADC 在等待阶段自动关机，并且应用程序读取 FIFO 后立即重新启动，则这种组合可显著节省功耗。

27.7. 模拟窗口看门狗

通过在 ADC_CFGR1 寄存器中设置 AWDEN 位，可启用 AWD 模拟看门狗功能。该功能用于监测所选的一个通道或所有已启用的通道是否保持在如图 27-11 所示的配置电压范围内（窗口）。

如果 ADC 转换的模拟电压低于一个较低阈值或高于一个较高阈值，则会设置 AWD 模拟看门狗状态位。这些阈值在 ADC_TR 寄存器中进行编程。通过设置 ADC_IER 寄存器中的 AWDIE 位可以启用中断。

软件通过向 AWD 标志写入 1 来清除该标志。

当转换分辨率小于 12bit 的数据时（根据位 RES[1:0]），必须保持编程阈值的 LSB 清除，因为内部比较始终是在完整的 12bit 原始转换数据上执行的（左对齐）。

表 27-4 显示了如何在 ADC_CFGR1 寄存器中配置 AWDSGL 和 AWDEN 位，以在一个或多个通道上启用模拟看门狗。

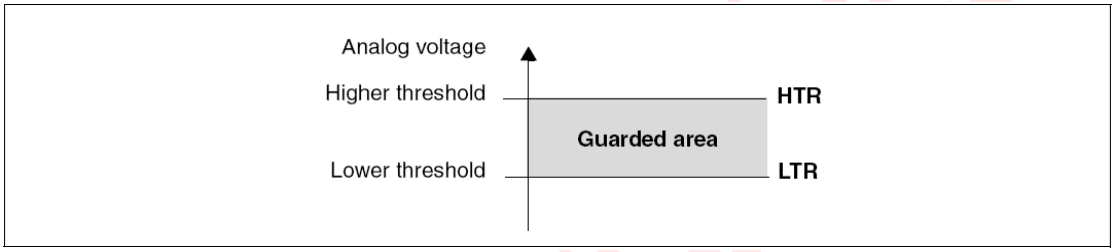


图 27-11：模拟看门狗保护区域

表 27-4：模拟看门狗通道选择

由模拟看门狗保护的通道	AWDSGL 位	AWDEN 位
无	x	0
所有通道	0	1
信号(1)通道	1	1

(1) 由 AWDCH 选择

27.8. 温度传感器

温度传感器与内部的 ADC1_IN15 输入通道相连，该输入通道用于将传感器的输出电压转换为数字值。

27.9. ADC 中断

以下任何事件都可能产生中断：

- ADC 上电，此时 ADC 已就绪(ADRDY 标志)
- 任何转换结束(EOC 标志)
- 转换序列结束(EOSEQ 标志)
- 当发生模拟看门狗检测时(AWD 标志)
- 采样阶段结束时(EOSMP 标志)
- 当发生数据溢出时(OVR 标志），可使用单独的中断使能位以提高灵活性。

表 27-5：ADC 中断

中断事件	事件标记	启用控制位
ADC Ready	ADRDY	ADRDYIE
End of Conversion	EOC	EOCIE
End of Sequence of Conversions	EOSEQ	EOSEQIE
Analog Watchdog Bit is Set	AWD	AWDIE
End of Sampling Phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE

27.10. 内存映射和寄存器

本小节描述内存映射和寄存器结构。

27.10.1. 内存映射

有关 QADC 内存映射的描述，请参见表 27-6。

表 27-6：QADC 内存映射

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_ISR																									AWD	Empty	FULL	OVR	EOSE	EOC	EOSM	ADRD
0x04	ADC_IER																									AWDIE			OVRIE	EOSEQIE	EOCIE	EOSMPI	ADRDYIE
0x08	ADC_CR																													ADSTP	ADSTAR	ADDIS	ADEN
0x0C	ADC_CFGR1	DIFF	OVRMOD				SEQ_LEN [2:0]			DISCEN	AUTOFF	WAIT	CONT		TRIGSCR [2:0]					TRIGMODE		ALIGN	RES [1:0]									DMAEN
0x10	ADC_CFGR2																					QPR [3:0]				STCNT [7:0]							
0x14	ADC_SMPR																									SMP [7:0]							
0x18	ADC_WDG																									AWDEN	AWDSG			AWDCH [3:0]			
0x1C	ADC_TR					HT [11:0]																LT [11:0]											
0x2C	ADC_CHSELR1					CCW3 [3:0]								CCW2 [3:0]								CCW1 [3:0]								CCW0 [3:0]			
0x30	ADC_CHSELR2					CCW7								CCW6								CCW5 [3:0]								CCW4 [3:0]			
0x4C	ADC_FIFO																	DATA [15:0]															

提示：

1. 所有寄存器均可通过 CPU 监控程序或用户模式访问。
2. 黑色部分为保留位，必须保持为复位值。

27.10.2. 寄存器

27.10.2.1. ADC 中断和状态寄存器（ADC_ISR）

偏移地址：0x0000

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	AWD	EMPTY	FULL	OVR	EOSEQ	EOC	EOSMP	ADRDY
W	w1c			w1c	w1c	w1c	w1c	w1c
RESET:	0	1	0	0	0	0	0	0

	= Writes have no effect and the access terminates without a transfer error excepiton.
w1c	= .Write 1 to the bit will clear it

图 27-12：ADC 中断和状态寄存器（ADC_ISR）

Read：任何时候

Write：任何时候

AWD — 模拟式看门狗标志

当转换电压超过 ADC_TR 寄存器中设定的数值时，硬件将设置该位。软件将其清零时，需向该位写入 1。

1 = 发生模拟看门狗事件

0 = 未发生类似监视器事件（或软件已确认并清除标志事件）

EMPTY — FIFO 空状态

当 FIFO 为空时，硬件会设置该位；当 FIFO 不为空时，硬件会清除该位。

1 = FIFO 为空

0 = FIFO 不为空

FULL — FIFO 满状态

当 FIFO 满时，硬件设置此位；当 FIFO 未满时，硬件清除此位。

1 = FIFO 已满

0 = FIFO 未满

OVR — ADC 超频

当发生溢出时，硬件会设置此位，这意味着新的转换在 FULL 标志已经设置的情况下完成。软件通过写入 1 来清除它。

1 = 已发生超限

0 = 未发生溢出（或软件已确认并清除标志事件）

EOSEQ — 序列结束标志

该位由硬件在序列转换结束时设置，软件写入 1 后清除。

1 = 转换序列完成

0 = 转换序列未完成（或软件已确认并清除标志事件）

EOC — 转换结束标志

该位由硬件在每个通道转换结束时设置，此时 ADC FIFO 寄存器中出现新的数据结果。软件通过向其写入 1 或读取 ADC FIFO 寄存器来清除该位。

1 = 信道转换完成

0 = 通道转换未完成（或标志事件已被软件确认并清除）

EOSMP — 采样结束标志

该位由硬件在转换期间、在采样阶段结束时设置。

1 = 达到采样阶段结束

0 = 未在采样阶段结束时（或软件已确认并清除标志事件）

ADRDY — ADC 就绪

该位由硬件在 ADC 被启用（ADEN = 1）后，以及当 ADC 达到可接受转换请求的状态时设置。软件通过向其写入 1 来清除该位。

1 = ADC 已准备好开始转换

0 = ADC 尚未准备好开始转换（或软件已经确认并清除标志事件）

27.10.2.2. ADC 中断使能寄存器（ADC_IER）

偏移地址：0x0004

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	AWDIE	0	0	OVRIE	EOSEQIE	EOCIE	EOSMPIE	ADRDYIE
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

图 27-13：ADC 中断使能寄存器（ADC_IER）

Read：任何时候

Write：在 ADC 启动之前

AWDIE — 模拟看门狗中断启用

通过软件设置和清除此位，以启用/禁用模拟看门狗中断。

1 = 启用模拟看门狗中断

0 = 禁用模拟看门狗中断

OVRIE — 过载中断使能

通过软件设置和清除此位，以启用/禁用过载中断。

1 = 超频中断已启用。当 OVR 位被设置时，将生成中断。

0 = 模拟中断已禁用

EOSEQIE — 转换序列结束中断使能

该位由软件设置和清除，以启用/禁用转换序列结束中断。

1 = EOSEQ 中断被启用。当 EOSEQ 位被设置时，将生成一个中断。

0 = 禁用 EOSEQ 中断

EOCIE — 转换结束中断使能

该位由软件设置和清除，以启用/禁用转换结束中断。

1 = EOC 中断被启用。当 EOC 位被设置时，将生成一个中断。

0 = 禁用 EOC 中断

EOSMPIE — 采样结束标志中断启用

通过软件设置和清除此位，以启用/禁用采样阶段结束中断。

1 = EOSMP 中断已启用。当 EOSMP 位被设置时，将生成中断。

0 = 禁用 EOSMP 中断。

ADRDYIE — ADC 就绪中断使能

该位由软件设置和清除，以启用/禁用 ADC 就绪中断。

1 = ADRDY 中断被启用。当 ADRDY 位被设置时，将生成一个中断。

0 = ADRDY 中断已禁用。

27.10.2.3. ADC 控制寄存器（ADC_CR）

偏移地址：0x0008

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	ADSTP	ADSTART	ADDIS	ADEN
W					rs	rs	rs	rs
RESET:	0	0	0	0	0	0	0	0

	= Writes have no effect and the access terminates without a transfer error excepiton.
rs	= Software can read and set, write 0 has no effect

图 27-14：ADC 控制寄存器（ADC_CR）

Read：任何时候

Write：参见每个比特描述

ADSTP — ADC 停止转换命令

该位由软件设置，用于停止并丢弃正在进行的转换(ADSTP 命令）。当转换实际上被丢弃且 ADC 准备接受新的开始转换命令时，硬件将清除该位。

1 = 写入 1 以停止 ADC。读取 1 表示 ADSTP 命令正在进行中。

0 = 无 ADC 停止转换命令正在进行

提示：仅当 ADSTART = 1 且 ADDIS = 0 时，软件才允许设置 ADSTP（ADC 已启用且可能正在转换，并且没有待处理的禁用 ADC 请求）

ADSTART — ADC 启动转换命令

软件设置此位以启动 ADC 转换。根据 TRIGMODE 配置位，转换要么立即启动（软件触发器配置），要么在硬件触发事件发生后启动（硬件触发器配置）。

通过硬件清除：

– 在单通道转换模式下，当选择软件触发器时：在转换序列结束（EOSEQ）标志被断言时。

– 在已停用的转换模式下，当选择软件触发器时：在断言转换结束（EOC）标志时。

– 在所有情况下：执行 ADSTP 命令后，硬件同时清除 ADSTP 位。

1 = 写入 1 以启动 ADC。读取 1 表示 ADC 正在运行并且可能正在转换。

0 = 无 ADC 转换正在进行。

提示：仅当 ADEN = 1 且 ADDIS = 0 时(ADC 已启用且没有待处理的禁用 ADC 请求），才允许软件设置 ADSTART

ADDIS — ADC 禁用命令

该位由软件设置，以禁用 ADC (ADDIS 命令）并使其进入断电状态（关闭状态）。一旦 ADC 被有效禁用，该位就会由硬件清除(ADEN 也在此时被硬件清除）。

1 = 写入 1 以禁用 ADC。读取 1 表示正在执行 ADDIS 命令。

0 = 没有正在进行的 ADDIS 命令

提示：仅当 ADEN = 1 且 ADSTART = 0 时，软件才允许设置 ADDIS（这可确保没有正在进行的转换）

ADEN — ADC 启用命令

该位由软件设置，以启用 ADC。一旦 ADRDY 标志被设置，ADC 将有效就绪以进行操作。在执行 ADDIS 命令后，硬件将清除 ADC 的该标志，从而禁用 ADC。

1 = 写入 1 以启用 ADC。

0 = ADC 被禁用（关闭状态）。

提示：只有当 ADC_CR 寄存器的所有位均为 0 时（ADSTP = 0、ADSTART = 0、ADDIS = 0 和 ADEN = 0），才允许软件设置 ADEN

27.10.2.4. ADC 配置寄存器 1（ADC_CFGR1）

偏移地址：0x000C

	31	30	29	28	27	26	25	24
R	DIFF	OVRMOD	0	0	0	SEQ_LEN[2:0]		
W								
RESET:	0	0	0	0	0	1	1	1

	23	22	21	20	19	18	17	16
R	DISCEN	AUTOFF	WAIT	CONT	0	TRIGSCR[2:0]		
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	TRIGMODE[2:0]			ALIGN	RES[1:0]	
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	DMAEN
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

rs

图 27-15：ADC 配置寄存器 1（ADC_CFGR1）

Read：任何时候

Write：在 ADC 启动之前

DIFF — 选择差分输入

此位确定输入是单端还是差分输入。

1 = 模拟输入采用差分采样。

0 = 模拟输入为单采样

OVRMOD — 覆盖管理模式

该位由软件设置和清除，并配置数据溢出的管理方式。

1 = 当检测到溢出时，ADC_DR 寄存器被用最后一次转换结果覆盖。

0 = ADC_DR 寄存器在发生溢出时保留旧数据

对...检波

SEQLEN[2:0] — 序列长度

这些位定义了序列的长度，序列长度 = SEQ_LEN+1，例如：SEQ_LEN = 7 表示序列长度为 8，SEQ_LEN = 0 表示序列长度为 1。

DISCEN — 不连续模式

该位由软件设置和清除，以启用/禁用间断模式。

1 = 开启不连续模式

0 = 禁用不连续模式

AUTOFF — 自动关闭模式

该位由软件设置和清除，以启用/禁用自动关闭模式。

1 = 启用自动关闭模式

0 = 关闭自动关闭模式

等待 — 等待转换模式

该位由软件设置和清除，以启用/禁用等待转换模式。

1 = 启用转换模式

0 = 等待转换模式关闭

CONT — 单/连续转换模式

该位由软件设置和清除。如果设置，则转换将持续进行，直到清除为止。

1 = 连续转换模式

0 = 单转换模式

TRIGSCR[2:0] — 外部触发源

这些比特用于选择 8 种可能的事件中哪一种可以触发转换。参见表 27-3。

TRIGMOD[2:0] — 触发模式选择

这些位用于选择软件触发模式或外部触发极性，参见表 27-2。

ALIGN — 数据对齐

该位由软件设置和清除，以选择右对齐或左对齐。请参见图 27-10。

0 = 右对齐

1 = 左对齐

RES[1:0] — 数据分辨率

这些位由软件写入，以选择转换的分辨率。请参见图 27-10。

DMAEN — 直接内存访问启用

该位由软件设置和清除，以启用 DMA 请求的生成。这允许使用 DMA 控制器自动管理转换后的数据。

1 = DMA 已启用

0 = 禁用 DMA

27.10.2.5. ADC 配置寄存器 2（ADC_CFGR2）

偏移地址：0x0010

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	QPR[3:0]			
W								
RESET:	0	0	0	0	0	0	1	0

	7	6	5	4	3	2	1	0
R	STCNT[7:0]							
W								
RESET:	0	0	1	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

rs

图 27-16：ADC 配置寄存器 2（ADC_CFGR2）

Read：任何时候

Write：在 ADC 启用之前

QPR[3:0] — 预分频器时钟分频比位

这些比特选择系统时钟除数以生成 QADC 时钟，如下所示：

FQCLK = Fsys_QCLK / (QPR[3:0] + 1)

在哪里

0 < QPR[3:0] < = 15，且值 1 不允许。

STCNT[7:0] — ADC 启动计数器位

ADC 需要经过 tSTAB（~2us）的稳定时间才能开始精确转换。该时间通过计算 QCLK 周期直至内部计数器达到STCNT[7:0] 来确定，因此用户应在ADC使能前设置这些位。例如，若 QCLK = 16MHz，则应将 STCNT[7:0] 设置为 2000/（1000/16） = 32。

27.10.2.6. ADC 采样时间寄存器（ADC_SMPR)

偏移地址：0x0014

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	SMP[7:0]							
W								
RESET:	0	0	0	0	0	0	1	0

= Writes have no effect and the access terminates without a transfer error excepiton.

图 27-17：ADC 采样时间寄存器（ADC_SMPR)

Read：任何时候

Write：在 ADC 启动之前

SMP[7:0] — 采样时间选择

软件通过这些位选择适用于所有通道的采样时间。

采样时间的计算公式为（SMP[7:0]+2）QCLKs

示例：SMP[7:0] = 0x02 表示采样时间为 4 QCLKs

27.10.2.7. ADC 看门狗寄存器（ADC_WDG）

偏移地址：0x0018

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	AWDEN	AWDSGL	0	0	AWDCH[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

图 27-18：ADC 看门狗寄存器（ADC_WDG）

Read：任何时候

Write：在 ADC 启动之前

AWDEN — 模拟监测器启用

该位由软件设置和清除。

1 = 启用模拟看门狗

0 = 禁用模拟监测器

AWDSGL-在单个通道或所有通道上启用看门狗

通过软件设置和清除该位，以启用 AWDCH[4:0] 位所标识通道上的模拟看门狗或所有通道上的模拟看门狗。

1 = 在单个通道上启用了模拟看门狗

0 = 所有频道均启用模拟监测器

AWDCH[3:0] — 模拟看门狗通道选择

这些位由软件设置和清除，它们选择由模拟看门狗控制的输入通道。

- 0000：由 AWD 监测的 ADC 模拟输入通道 0
- 0001：由 AWD 监测的 ADC 模拟输入通道 1
- .....
- 0111：由 AWD 监测的 ADC 模拟输入通道 7

- 1111：由 AWD 监测的温度传感器
- 其他值：保留，不得使用

27.10.2.8. ADC 看门狗阈值寄存器（ADC_TR）

偏移地址：0x001C

	31	30	29	28	27	26	25	24
R	0	0	0	0HT[12:8]				
W								
RESET:	0	0	0	1	1	1	1	1

	23	22	21	20	19	18	17	16
R	HT[7:0]							
W								
RESET:	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8
R	0	0	0	0LT[12:8]				
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	LT[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

图 27-19：ADC 看门狗阈值寄存器（ADC_TR）

Read：任何时候

Write：在 ADC 启动之前

HT[12:0] — 模拟看门狗高阈值

软件将这些位写入，以定义模拟看门狗的较高阈值。

LT[12:0] — 模拟看门狗下阈值

软件写入这些位以定义模拟看门狗的下限阈值。

27.10.2.9. ADC 通道选择寄存器 1（ADC_CHSELR1，ADC_CHSELR2）

偏移地址：0x002C

	31	30	29	28	27	26	25	24
R	0	0	0	0	CCW3[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	CCW2[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	CCW1[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	CCW0[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

图 27-20：ADC 通道选择寄存器 1（ADC_CHSELR1）

偏移地址：0x0030

	31	30	29	28	27	26	25	24
R	0	0	0	0	CCW7[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	CCW6[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	0	0	0	0	CCW5[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	0	0	0	0	CCW4[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

图 27-21：ADC 通道选择寄存器 2（ADC_CHSELR2）

Read：任何时候

Write：在 ADC 启动之前

CCWi[3:0] — 数字 i 转换选择通道，参见表 27-1。

27.10.2.10. ADC FIFO 访问寄存器（ADC_FIFO）

偏移地址：0x004C

	31	30	29	28	27	26	25	24
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
R	DATA[15:8]							
W								
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R	DATA[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

= Writes have no effect and the access terminates without a transfer error excepiton.

图 27-22：ADC FIFO 访问寄存器（ADC_FIFO）

Read：任何时候

Write：从不

DATA[15:0]：转换后的数据，参见图 27-10。

28. 电气特性

本章提供了 LT165A 的电气特性参数和限值。

28.1. 极限参数

表 28-1：电气极限参数表

符号	参数描述	参数范围	单位
V_{DD33}	电源电压	-0.5 ~ 4.6	V
V_{IN}	逻辑输入电压	-0.5 ~ VDD33+0.5	V
V_{OUT}	逻辑输出电压	-0.5 ~ VDD33+0.5	V
P_D	最大功耗	≦300	mW
T_{OPR}	工作温度范围	-40 ~ 105	℃
T_{JT}	工作结温范围	-40 ~ 125	℃
T_{ST}	储存温度范围	-55 ~ 150	℃
T_{SOL}	最高焊接温度	260	℃

提示：最大极限值是指超出该工作范围时，芯片有可能损坏。推荐工作范围是指在该范围内，器件功能正常，但并不完全保证满足个别性能指针。电气参数定义了器件在工作范围内并且在保证特定性能指针的测试条件下的直流和交流电参数规范。对于未给定上下限值的参数，本规范不予保证其精度，但其典型值合理反映了器件性能。

28.2. DC 电气参数

表 28-2：IO 电气参数表(3.3V)

项目	符号	最小值	典型值	最大值	单位
工作电压	VDD33	VOP_L(*1)	3.3	3.63	V
输入高电位	V_{IH}	2.0	-	VDD33+0.3	V
输入低电位	V_{IL}	-0.3	-	0.8	V
输出高电位	V_{OH}	VOP_L*0.8	-	VDD33	V
输出低电位	V_{OL}	0	-	0.4	V
输入漏电流	I_{IN}	-	-	1	uA
上拉电阻	RPU	33	41	62	KΩ
下拉电阻	RPD	33	42	68	KΩ

提示*1：参考下表 28-4。

表 28-3：电源特性

项目	符号	最小值	典型值	最大值	单位
芯片电源	VDD33	VOP_L	3.3	3.63	V
ADC 工作电压	AVDD	VOP_L	3.3	3.63	V
内核工作电压 (LDO O/P)	VDD12	1.1	1.2	1.3	V
RTC 工作电压	VBAT	1.7	3.3	3.6	V

表 28-4：PVDC 设置表对应 VOPL

PVDC	检测电压	VOP_L
2’ b00	2.16V	2.31V
2’ b01	2.32V	2.47V
2’ b10	2.48V	2.63V
2’ b11	2.64V	2.79V

提示：

1. PVDC 为可编程电压探测器配置寄存器，请参考详细规规格书第 14.1.1 节说明。
2. 因 LT165A 代码配置在外挂的 SPI Flash 内，因此 SPI Flash 的最低工作电压需低于 VOP_L 的设置电压。

28.3. ESD 保护规格

表 28-5：ESD 保护规格

ESD 项目	符号	最大值	单位	参考标准
Human Body Model	HBM	±4,000	V	ANSI/ESDA/JEDEC JS-001-2017
Machine Model	MM	200	V	JEDEC JESD22-A115C-2010
Charged Device Model	CDM	±1000	V	ANSI/ESDA/JEDEC JS-002-2022
Latch Up (常温)	LU	±200	mA	JEDEC JESD78F.01-2022, @105℃

提示：在进行人工焊接时建议人员与设备要做防静电处理，如适当的温湿度环境、焊接设备接地、防静电工作台、及焊接人员戴防静电手腕带等等。

28.4. VDD 上电时序

LT165A 使用时必须注意 VDD 的上电（Power Up）要求，在上电时 VDD33 必须在低电压（V_L）维持至少 400ms 以上的等待时间（T_{WAIT}），同时 VDD33 由 V_L 到正常工作电压的上升时间（T_R）也不能太长，必须在 500ms 内达正常的工作电压范围，否则容易导致 LT165A 内部的 MCU 无法正常启动。

表 28-6：VDD 上电（Power Up）特性

参数	符号	条件与说明	Min.	Nom.	Max.	单位
Rise Time	T_R	输入电压由 V_L 到正常工作电压的上升时间	-	-	500	ms
Wait Time	T_{WAIT}	Power On 之前 VL 的保留时间	400	-	-	ms
VDD 输入电压	V_L	at T = T_1 on pin VDD33 (Power On 前的输入电压)	-	-	200	mV
VDD 输入电压	V_H	正常工作电压	VOP_L(*1)	3.3	3.63	V

提示*1：参考上表 28-4。

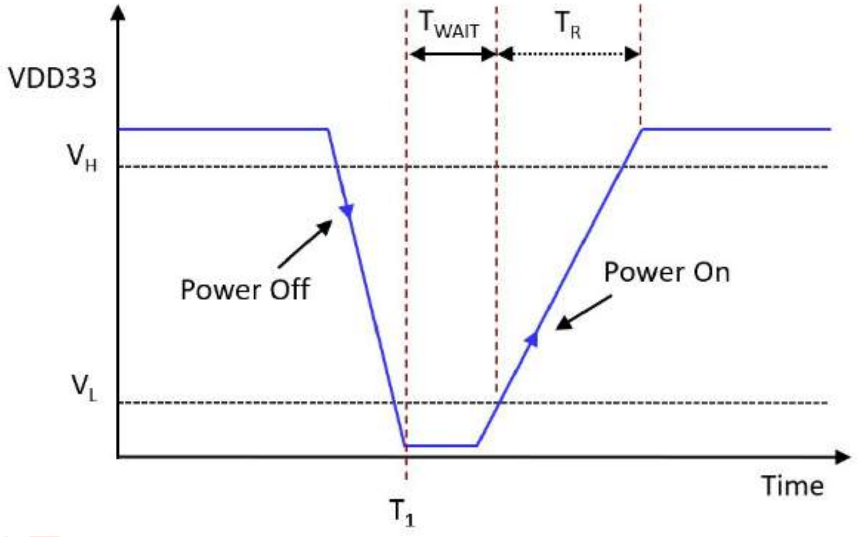


图 28-1：VDD 上电要求时序图

29. 参考原理图

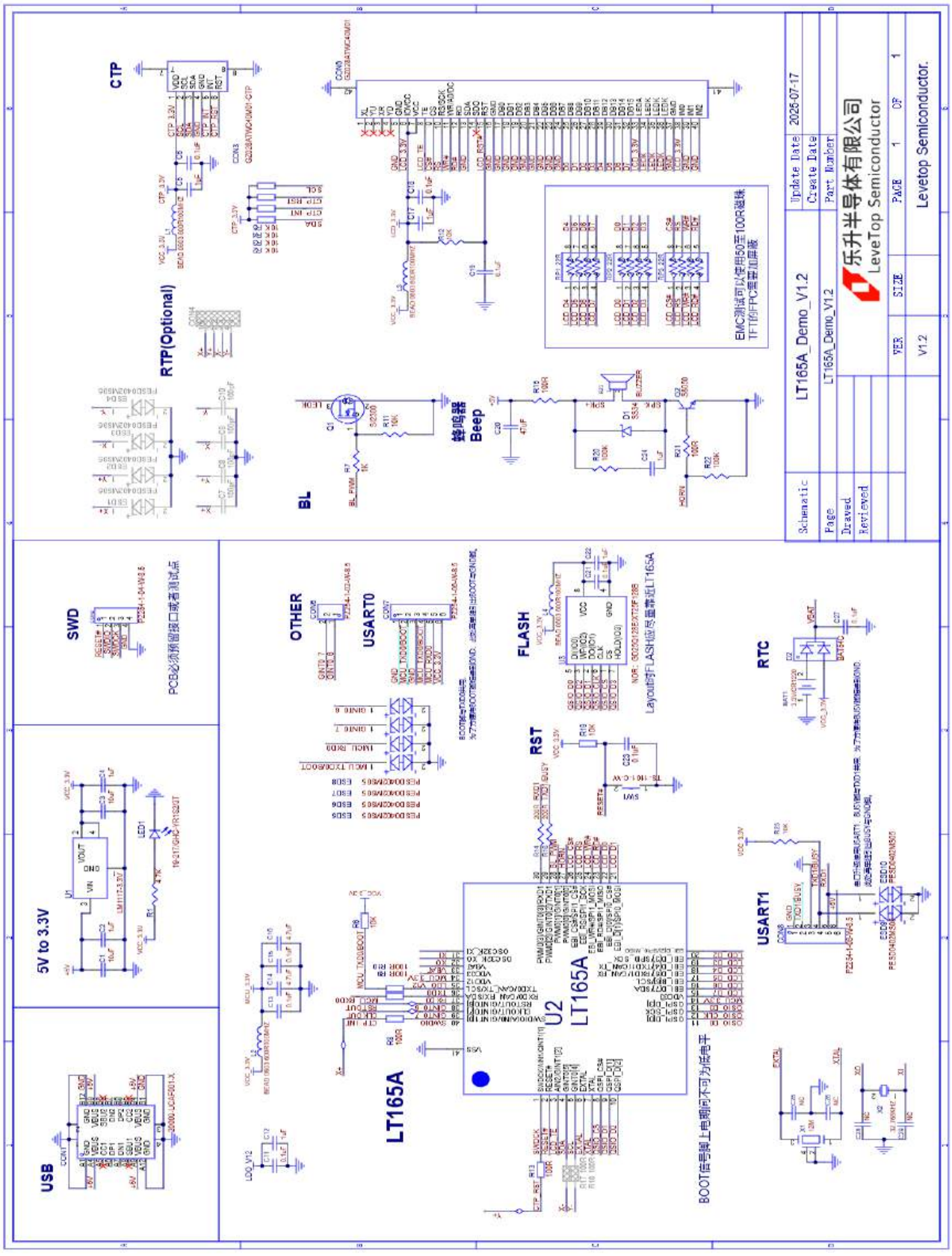


图 29-1：LT165A 接 8bit 8080 MCU 屏的参考原理图

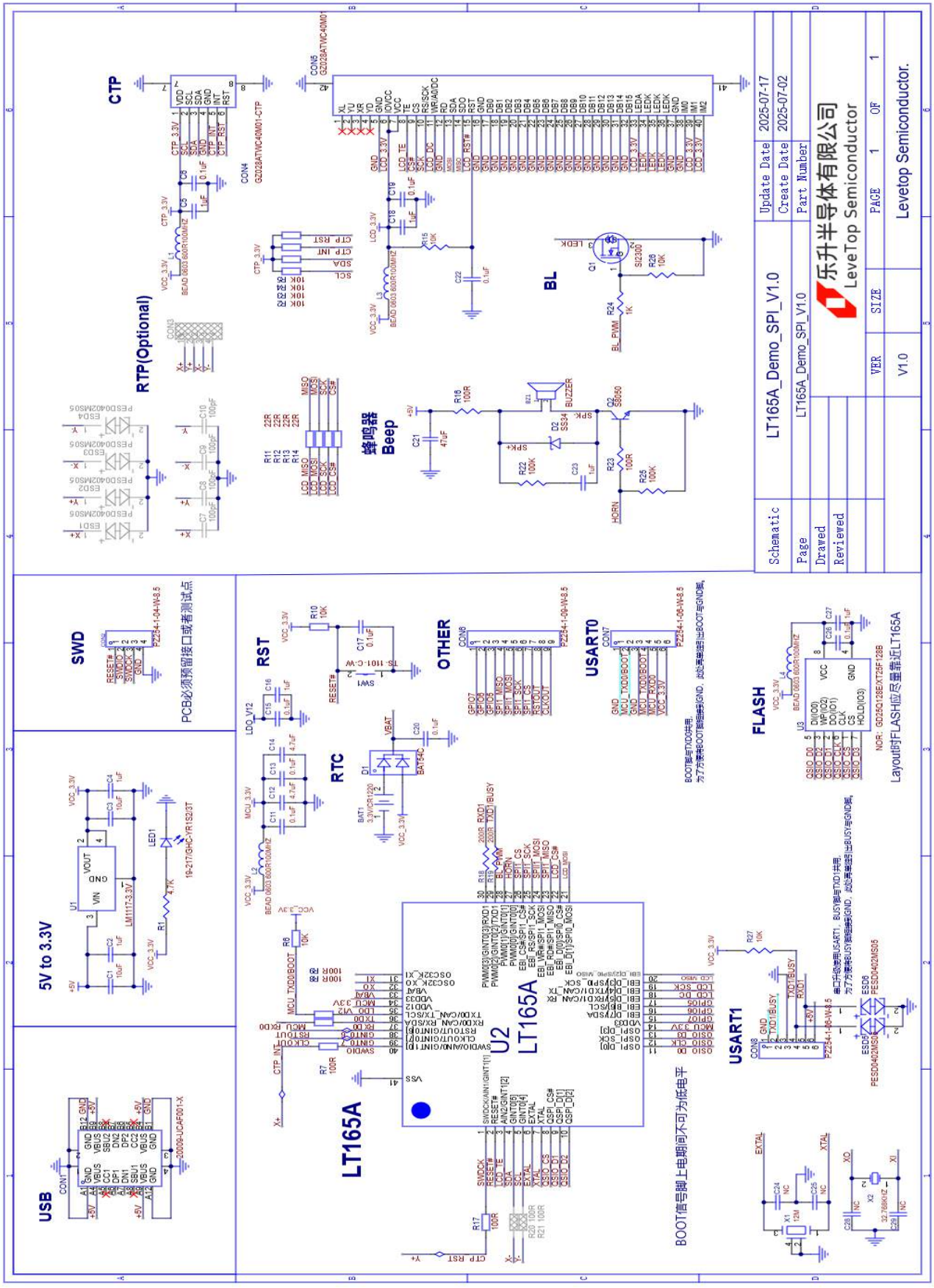


图 29-2：LT165A 接 SPI MCU 屏的参考原理图

30. 封装讯息

30.1. LT165A（QFN-40pin）

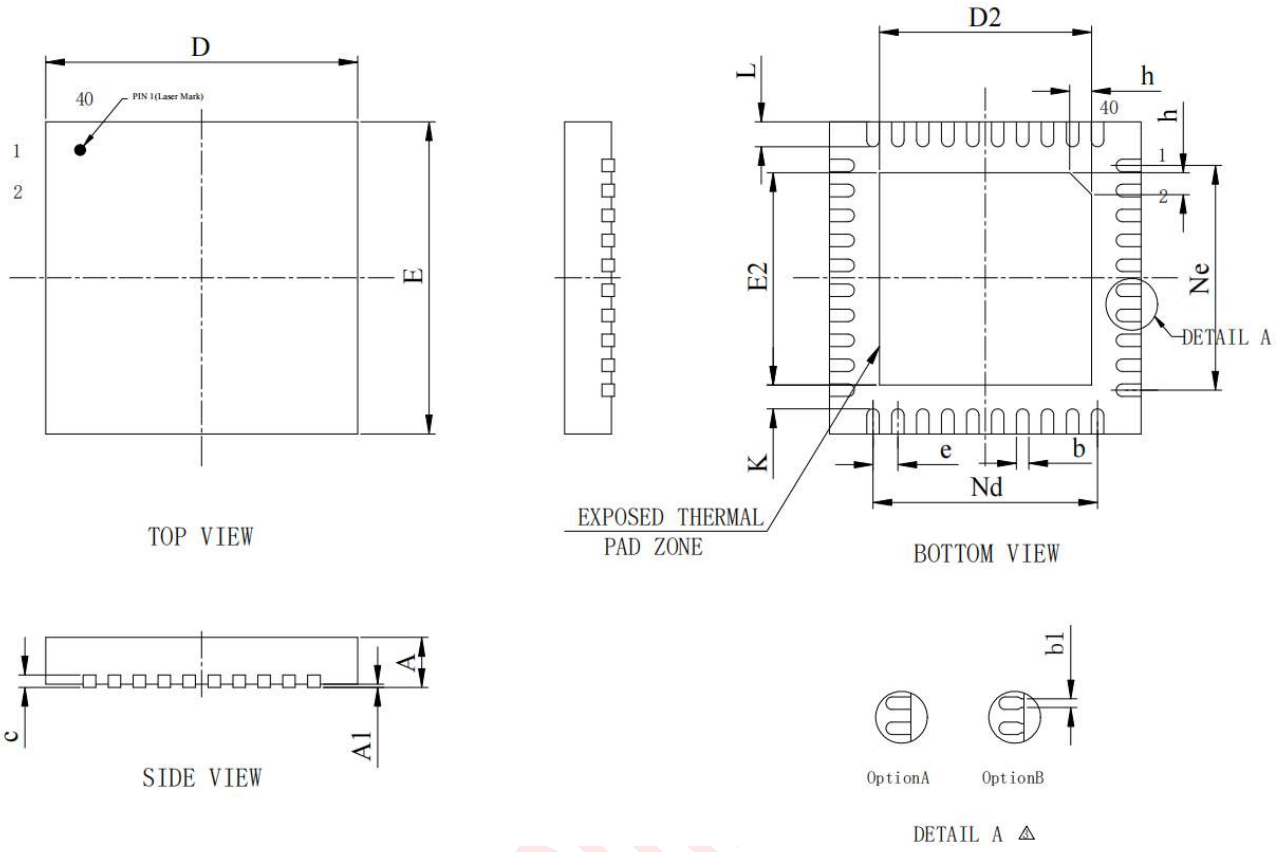


图 30-1：LT165A 外观尺寸图

提示：PCB 布局时，LT165A 背部的散热焊盘（Thermal Pad Zone）必须直接接地。焊盘的 PCB 布线请参考 30.3 节的说明。

表 30-1：LT165A 封装尺寸参数

Symbol	Millimeter			Symbol	Millimeter		
	Min.	Nom.	Max		Min.	Nom.	Max
A	0.70	0.75	0.80	Nd	3.6BSC		
A1	--	0.02	0.05	E	4.90	5.00	5.10
b	0.15	0.20	0.25	E2	3.30	3.40	3.50
b1	0.14REF			Ne	3.60BSC		
c	0.18	0.25	0.30	L	0.35	0.40	0.45
D	4.90	5.00	5.10	K	0.20	--	--
D2	3.30	3.40	3.50	h	0.30	0.35	0.40
e	0.40BSC						

30.2. 芯片接地焊盘的 PCB 设计

LT165A 采用 QFN 封装，芯片背部为接地（GND）的散热焊盘，为了达到更好的散热与降低焊接风险，在 PCB Layout 时建议把 LT165A 底部焊盘的 PCB 铜箔面分割为四个或是多个小的焊接面（方形或是圆形），并且各焊接面之间的间隔设置在~0.8mm，避免 PCB 使用相同甚至大于 LT165A 焊盘大小的完整焊接面而造成焊接不全，或是在焊接冷却后 PCB 与芯片焊盘拉扯导致芯片变形及接触不良。正确的 PCB 焊盘布局如下图 LT165A 范例，中间浅黄色区是 LT165A 底部的接地焊盘，灰色区是 PCB 接地小焊盘（焊接面），每个焊盘过孔接地 1~2 个既可。

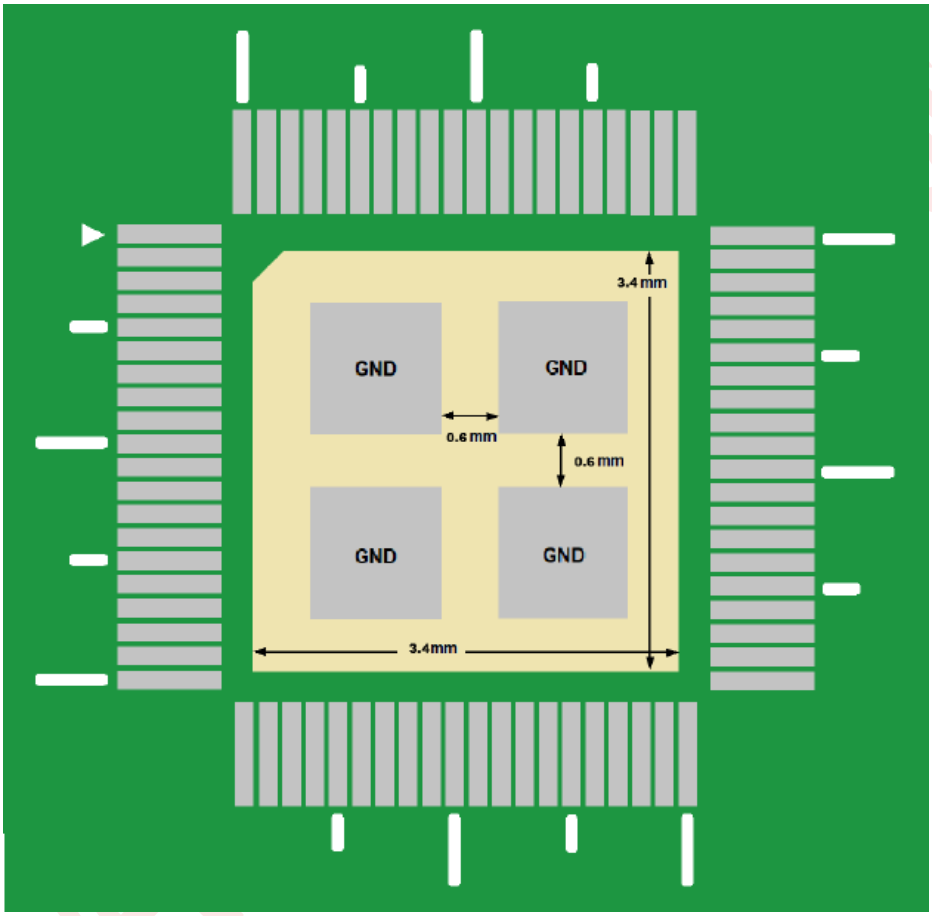


图 30-2：LT165A 底部焊盘 PCB 的设计建议