乐升半导体
LEVETOP

LT7589

Uart TFT 显示控制芯片

High Performance Uart TFT Display Controller

规 格 书

V1.3

www.levetop.cn

乐升半导体有限公司

版本记录

版本	日期	描述
V1.0	2024/11/22	● LT7589 初版
V1.1	2025/03/20	● 原理图更新 ● VDD33_IO Pin 脚说明更新
V1.2	2025/04/21	● 更新引脚 LCD_XI 的说明 ● 更新表 6-4：外部串行 Flash 信号说明 ● 原理图更新
V1.3	2025/11/28	● 更新表 28-1：QFN-96Pin 尺寸参数 ● 更新图 29-2：LT7589B 参考原理图

版权说明

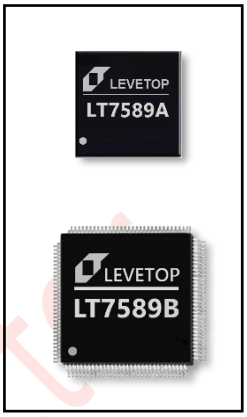
内　容

图 列 表

表 列 表

1. 芯片介绍

1.1. 基本简介



LT7589 是一款高效能 Uart TFT 串口屏控制芯片。内部结合了 32-bits RISC MCU 及具有 TFT LCD 图形显示控制器的 GUI（Graphical User Interface），主要的功能就是提供 Uart 串口通讯，让主控端 MCU 透过简易的串口指令就能轻易的将要显示的信息呈现到 TFT 屏上。LT7589 内部硬件还提供 JPG 图片解码、PIP（Picture-in-Picture）、几何图形绘图等功能，能够提升 TFT 屏显示效率，及降低 MCU 处理图形显示所花费的时间，LT7589 支持的显示分辨率由 480*480 (QVGA) 到 1280*800，用于 16/24-bits 的 RGB 接口显示屏。

LT7589 内部的 MCU 主频最高为 200MHz，含有 2MB Flash、768KB SRAM、16MB 显示内存，并结合 **JPG 解码器**、2D 图形加速显示器、DMA 数据读取、与高速 **QSPI Flash 接口**，用来快速读取储存在外部 QSPI Flash 的图片、动画、字库等信息，具有良好的显示效能。LT7589 可以配合 乐升半导体 开发的 UI 编辑软件（UI_Editor）、模拟软件（UI_Emulator），直接在电脑上将设计好的 UI 素材与显示交互逻辑进行导入与显示界面开发，其所支持的显示功能包括图片显示、动画显示、滑动菜单显示、进度条显示、字符串显示、中英文键盘、数字键盘、模拟时钟、数字时钟、指针显示、二维码生成、多国语言、音频播放、变量控制，及结合触控屏或编码器功能的控制效果等等。除了串口屏 Uart 通讯接口，LT7589 还提供多组的 SCI（Uart）接口可以连接如蓝牙模块、WiFi 模块等元件，也提供 **CanBus**、SD 卡（SPI 模式），模拟输入 AIN、PWM 及 INT 中断等接口，还自带 RTC 时钟。亦可用于 Little VGL 的 GUI 图形开发，具有良好的流畅度与极高的性价比。

由于含有高容量的 Flash 及 SRAM，LT7589 也可以作为一个带 TFT 控制器的主控 MCU 来使用，将主控及 TFT 显示功能由一颗 LT7589 来完成，它的显示功能非常适合用在 1280x800 分辨率以下带 TFT-LCD 屏的电子产品上，或用来取代原单色屏产品，提升产品达到智能化显示信息、增加产品质感、档次。LT7589 强大的显示功能非常适合用在有 TFT-LCD 屏的电子产品上，如各式智慧家电、汽车仪表盘、摩托车面板、多功能事务机、工业控制、电子仪器、医疗美容设备、检测设备、充电设备、逆变器、UPS 等电源设备、音响设备、及带屏智能音箱、机器人等产品。

1.2. 系统应用方块图

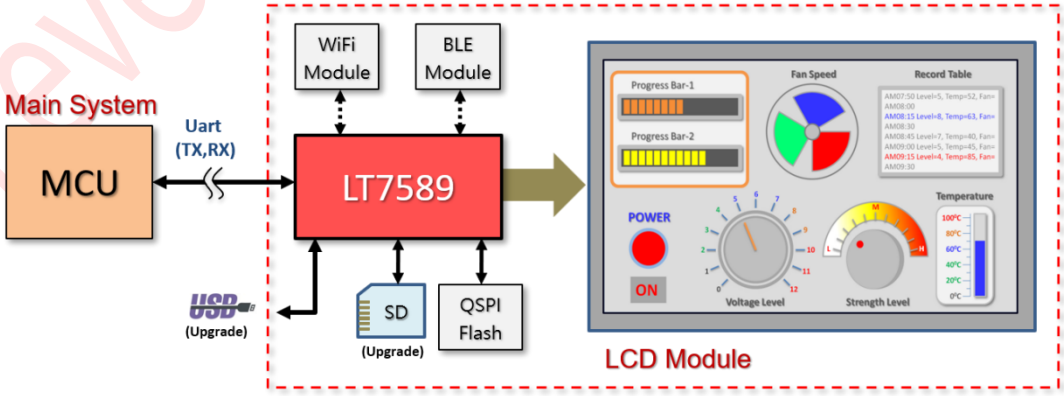


图 1-1：LT7589 设置在模块板上

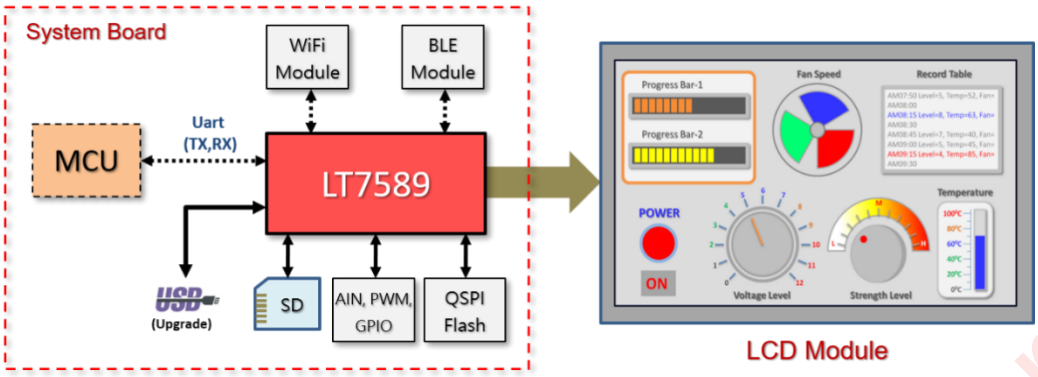


图 1-2：LT7589 设置在系统主板上

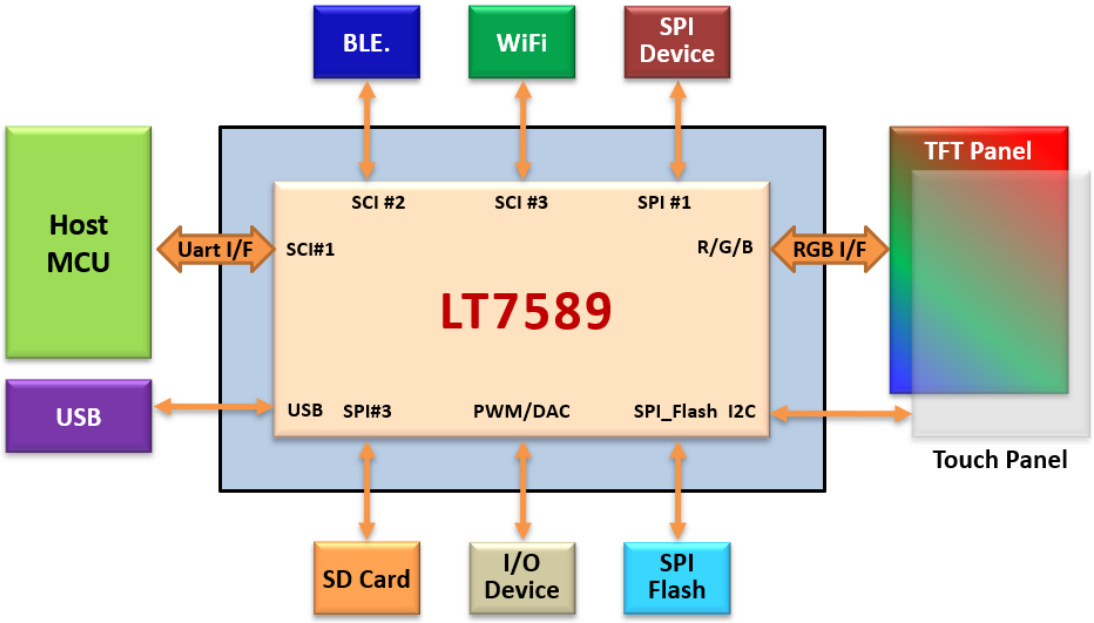


图 1-3：LT7589 应用架构

1.3. 内部方块图

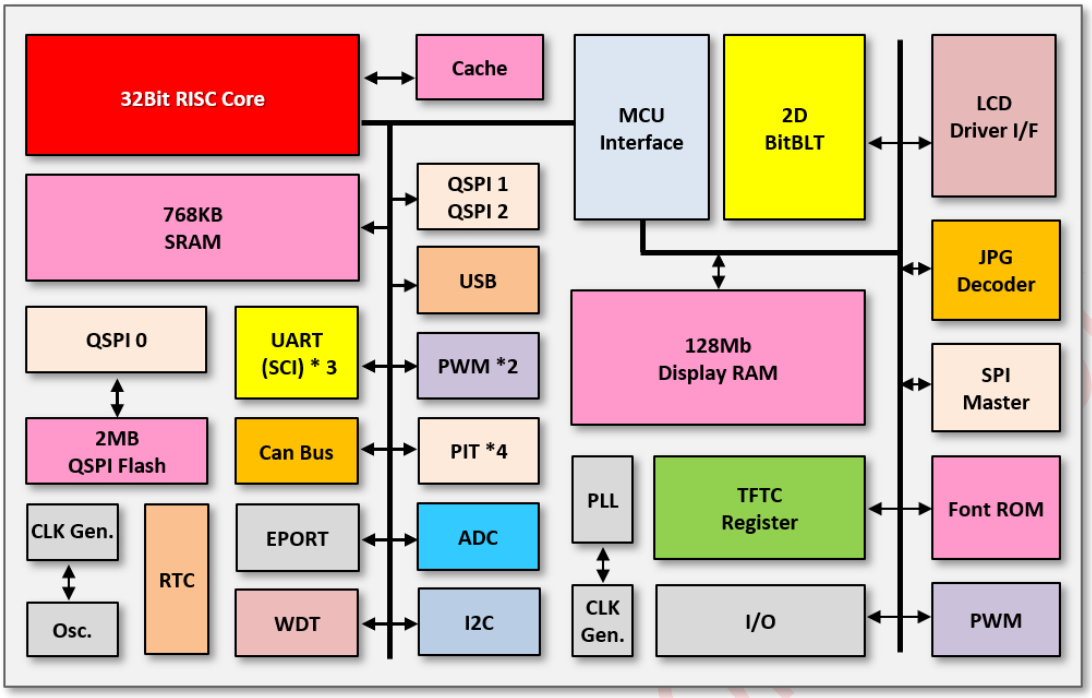


图 1-4：LT7589 内部方块图

表 1-1：型号说明

型 号	封 装	内 建 显 示 内 存	分 辨 率	色 彩
LT7589A	QFN-96	128Mb	1280*800	16.7M 色 αRGB 8:8:8:8
LT7589B	LQFP-128	128Mb	1280*800	16.7M 色 αRGB 8:8:8:8

1.4. 功能简介

主控端 MCU 界面

- 支持 Uart、USB 接口。
- 内建高效能 32-bits MCU、主频为 180MHz，最高可达 200MHz。

USB 界面

- 支持 USB2.0 Full Speed。

SCI (Uart)界面

- 提供 3 組 SCI (Serial Communications Interface)。
- 可连接外部 SCI 接口之元件或是模块。

内存

- MCU 内建 2M bytes Flash。
- MCU 内建 512K+256K bytes SRAM。
- TFT 控制器内建 128Mb 的显示内存（Display RAM）。

显示色彩数据格式

- 16bpp ：彩色 RGB 5:6:5（2bytes/像素）。
- 24bpp ：彩色 RGB 8:8:8（3bytes/像素或是 4bytes/像素）。
  - αRGB 4:4:4:4（4,096 索引色/像素，含透明度属性）
- 32bpp ：彩色 αRGB 8:8:8:8（4bytes/像素）。

面板接口与分辨率

- 支持 16、24-bits RGB 接口面板。
- 采用裸机开发时支持的分辨率:
  - VGA ：640*480 TFT 屏
  - WVGA：800*480 TFT 屏
  - SVGA ：800*600 TFT 屏
  - XGA ：1024*768 TFT 屏
  - SXGA ：1280*1024 TFT 屏
- 串口屏开发时支持最大分辨率：1280*800。

显示功能

- **内建 JPG 硬件解码器**
- 支持使用者可自行定义 4 个 32*32 的图形光标。
- 提供虚拟显示功能: 虚拟显示可显示大于 LCD 面板大小的图像，这样图像可以在任何方向上轻松滚动。
- 提供画中画 (PIP) 显示: 支持两个 PIP 视窗区域: 启用的 PIP 视窗显示在主视窗的上层，而 PIP1 视窗显示在 PIP2 视窗的上层。
- 支持多重显示功能：可以在显示缓冲区之间切换主显示视窗，达到简单的动画显示效果。
- 支持唤醒时迅速显图像功能。
- 支持镜像和垂直翻转显示功能。
- 彩带显示（Color Bar Display）：在没有对内部显示内存写入数据的情况下仍然可以以彩带的方式显示，默认分辨率为 640*480 像素。

区块传输引擎（BitBLT）

- 内建 2D BitBLT 引擎。
- 提供带光栅运算的复制图像功能。
- 提供颜色深度转换。
- 实心填充和图案填充功能:
  - 提供用户定义的 8*8 图像或 16*16 图像。
- 提供两个图像合成一个图像功能:
  - 色度键控功能 (Chroma-Keying)：根据透明度将图像与指定的 RGB 颜色混合。
  - 图形混合透明模式（Window Alpha - Blending）：根据指定区域内的透明度将两个图像混合。
  - 像素混合透明模式（Dot Alpha - Blending）：根据 RGB 格式及透明度将两个图像混合。

几何图形加速器

- 提供画点、线、曲线、椭圆、三角形、矩形、圆角矩形等绘图功能。

显示文字功能

- 内建 ISO/IEC 8859-1/2/4/5 的 8*16、12*24、16*32 字型。
- 支持使用者自定义半型字角与全型字（8*16、12*24、16*32、16*16、32*32）。
- 支持 48*48、72*72 大全型字。
- 提供可程序文字光标。
- 支持垂直与水平放大字型（*1, *2, *3, *4 倍）。
- 支持文字 90 度旋转。

SPI Master 界面

- TFT 图形加速器提供外部串行闪存（Serial Flash）数据复制至图框缓冲区。
- **兼容标准 QSPI 规格 NOR/NAND Flash。**
- **支持 Nand Flash 坏块处理。**
- **支持 MCU 對 SPI Flash 的 by Pass Mode。**
- 提供 16bytes 读取 FIFO 及 16bytes 写入 FIFO。
- 在 Tx FIFO 完全清空并且 SPI Tx/Rx 引擎闲置时会发出中断。
- 提供额外 2 組兼容标准 SPI 接口。

I2C 界面

- MCU 提供 I2C 接口与外部 I2C 装置连接。
- 提供标准传输模式（100kbps）与快速传输模式（400kbps）。

PWM 界面

- MCU 提供 8 个 PWM 接口。
- TFT 控制器内建 2 组 16-bits 计数器，提供 2 个 PWM 输出接口。
- 可程序化的工作周期定。

中断信号界面

- MCU 最多可提供 19 个中断输入接口。
- TFT 控制器提供 1 中断输出接口。

GPIO 界面

- MCU 最多可提供 24 个 GPIO 接口。
- TFT 控制器最多可提供 17 个 GPIO 接口。

模拟输入界面

- MCU 提供 8 个 ADC 的模拟输入接口。

复位方式

- MCU 提供 电源开启复位、外部复位输入、软件复位、看门狗复位和电压侦测复位
- TFT 控制器提供电源启动复位、外部硬件复位和软件命令复位。

省电模式

- 提供 3 种省电模式：待机（Standby）、休眠（Suspend）与睡眠（Sleep）模式。
- 支持使用 MCU 唤醒。

时钟（Clock）

- MCU 与 TFT 控制器独立时钟。
- MCU 内建精准高频时钟
- 内建 RTC、外部 32KHz 晶振电路。
- TFT 控制器内建可程序化 PLL，提供内部时钟、外部 LCD 时钟、内部显示内存时钟。

电源供应

- VDD 电压：3.3V +/- 0.3V。
- 内建 1.2V LDO。

封装型式

- QFN-96Pin（10*10mm^2）封装。
- LQFP-128Pin（14*14mm^2）封装。

工作温度

- -40℃~85℃。(@180MHz)

2. 引脚信号

2.1. 芯片脚位图

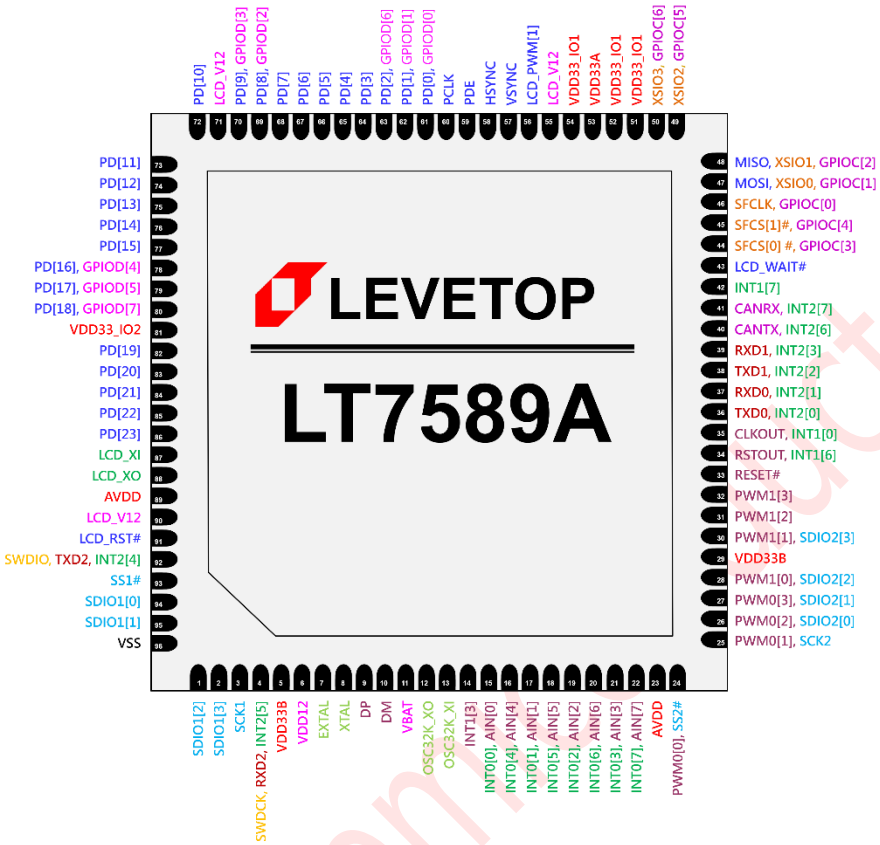


图 2-1：LT7589A 引脚图（QFN-96Pin）

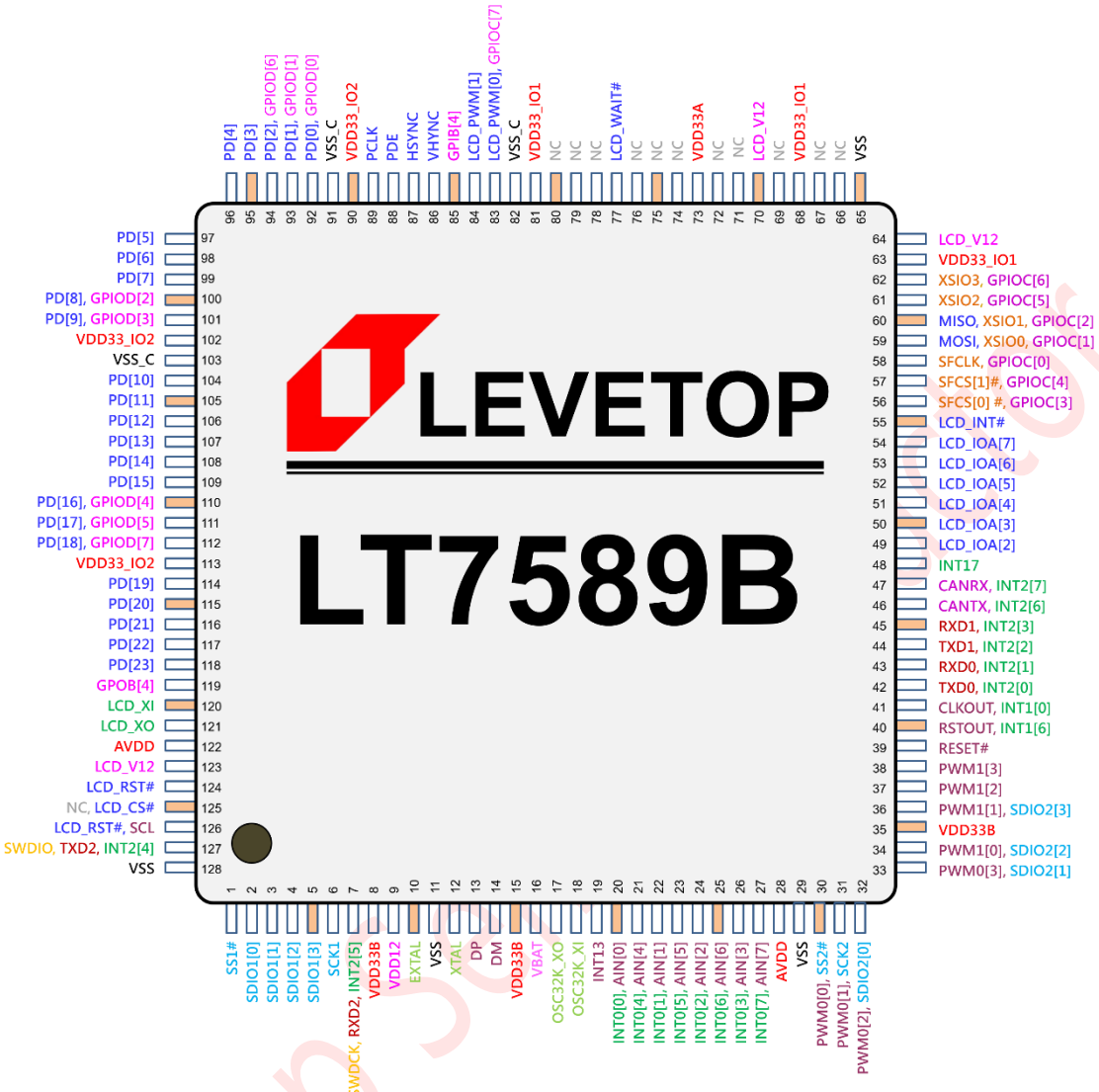


图 2-2：LT7589B 引脚图（LQFP-128Pin）

2.2. 信号说明

2.2.1. SCI（Uart）串口信号

表 2-1：SCI（Uart）串口信号

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
37	43	RXD0	I	**串口通信(Uart) #0 接收数据输入** 此信号用于 SCI #0 接收器数据输入，可用于连接外部 SCI 接口之元件或是模块。也可作为普通的 GPIO 或是中断输入接口 INT2[1]使用。
36	42	TXD0	O	**串口通信(Uart) #0 发送数据输出** 此信号用于 SCI #0 发送器数据输出，可用于连接外部 SCI 接口之元件或是模块。也可作为普通的 GPIO 或是中断输入接口 INT2[0]使用。
39	45	RXD1	I	**串口通信(Uart) #1 接收数据输入** 此信号用于 SCI #1 接收器数据输入，经由内部 MCU 的寄存器设定，也可作为普通的 GPIO 或是中断输入接口 INT2[3]使用。
38	44	TXD1	O	**串口通信(Uart) #1 发送数据输出** 此信号用于 SCI #1 发送器数据输出，经由内部 MCU 的寄存器设定，也可作为普通的 GPIO 或是中断输入接口 INT2[2]使用。
4	7	RXD2	I	**串口通信(Uart) #2 接收数据输入** 此信号用于 SCI #2 接收器数据输入，经由内部 MCU 的寄存器设定，也可作为普通的 GPIO 或是中断输入接口 INT2[5]使用。
92	127	TXD2	O	**串口通信(Uart) #2 发送数据输出** 此信号用于 SCI #2 发送器数据输出，经由内部 MCU 的寄存器设定，也可作为普通的 GPIO 或是中断输入接口 INT2[4]使用。

2.2.2. LCD 屏接口信号

表 2-2：LCD 屏接口信号

<table>
<tr><th>脚号
LT7589A</th><th>脚号
LT7589B</th><th>引脚名称</th><th>I/O</th><th>功 能 说 明</th></tr>
<tr><td>86~82,
80~72,
70~61</td><td>118~114
112~104
101~92</td><td>PD[23:19],
PD[18:10],
PD[9:0]</td><td>O</td><td>LCD 数据总线
输出数据至 TFT-LCD 屏的数据总线，可经由寄存器来设定连接相对应的 RGB 总线。
<table>
<tr><th rowspan="2">Pin
Name</th><th colspan="2">TFT-LCD Interface</th></tr>
<tr><th>16-bits</th><th>24-bits</th></tr>
<tr><td>PD[0]</td><td>GPIOD[0]</td><td>B0</td></tr>
<tr><td>PD[1]</td><td>GPIOD[1]</td><td>B1</td></tr>
<tr><td>PD[2]</td><td>GPIOD[6]</td><td>B2</td></tr>
<tr><td>PD[3]</td><td>B0</td><td>B3</td></tr>
<tr><td>PD[4]</td><td>B1</td><td>B4</td></tr>
<tr><td>PD[5]</td><td>B2</td><td>B5</td></tr>
<tr><td>PD[6]</td><td>B3</td><td>B6</td></tr>
<tr><td>PD[7]</td><td>B4</td><td>B7</td></tr>
<tr><td>PD[8]</td><td>GPIOD[2]</td><td>G0</td></tr>
<tr><td>PD[9]</td><td>GPIOD[3]</td><td>G1</td></tr>
<tr><td>PD[10]</td><td>G0</td><td>G2</td></tr>
<tr><td>PD[11]</td><td>G1</td><td>G3</td></tr>
<tr><td>PD[12]</td><td>G2</td><td>G4</td></tr>
<tr><td>PD[13]</td><td>G3</td><td>G5</td></tr>
<tr><td>PD[14]</td><td>G4</td><td>G6</td></tr>
<tr><td>PD[15]</td><td>G5</td><td>G7</td></tr>
<tr><td>PD[16]</td><td>GPIOD[4]</td><td>R0</td></tr>
<tr><td>PD[17]</td><td>GPIOD[5]</td><td>R1</td></tr>
<tr><td>PD[18]</td><td>GPIOD[7]</td><td>R2</td></tr>
<tr><td>PD[19]</td><td>R0</td><td>R3</td></tr>
<tr><td>PD[20]</td><td>R1</td><td>R4</td></tr>
<tr><td>PD[21]</td><td>R2</td><td>R5</td></tr>
<tr><td>PD[22]</td><td>R3</td><td>R6</td></tr>
<tr><td>PD[23]</td><td>R4</td><td>R7</td></tr>
</table>
当 LCD 设置为 16/18bpp 功能模式时，有些 PD 可被定义为 GPIO 引脚。</td></tr>
</table>

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
60	89	PCLK	O	**LCD 屏幕扫描时钟信号** 屏幕扫描时钟信号连接至通用的 TFT 驱动接口讯号。此信号为内部 PPLL 驱动产生。
57	86	VSYNC	O	**LCD 垂直同步信号** 垂直同步信号 VSYNC 连接至通用的 TFT 驱动接口讯号。
58	87	HSYNC	O	**LCD 水平同步信号** 水平同步讯号 HSYNC 连接至通用的 TFT 驱动接口讯号。
59	88	PDE	O	**LCD 屏幕数据使能** 此信号为连接至通用 TFT 驱动接口的数据有效或数据使能信号。

2.2.3. QSPI 信号

表 2-3：QSPI 信号

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
3	6	SCK1	O	**QSPI #1 串行时钟信号** 此信号为第 1 组 SPI 的时钟信号输出，可用于连接外部 SPI 接口之元件或是模块。
93	1	SS1#	O	**QSPI #1 芯片选择信号** 此信号为第 1 组 SPI 的片选输出。
94	2	SDIO1[0]	IO	**QSPI #1 的数据输出/输入信号** 此信号为第 1 组 QSPI 数据 0 的输出/输入信号。
95	3	SDIO1[1]	IO	**QSPI #1 的数据输出/输入信号** 此信号为第 1 组 QSPI 数据 1 的输出/输入信号。
1	4	SDIO1[2]	IO	**QSPI #1 的数据输出/输入信号** 此信号为第 1 组 QSPI 数据 2 的输出/输入信号。
2	5	SDIO1[3]	IO	**QSPI #1 的数据输出/输入信号** 此信号为第 1 组 QSPI 数据 3 的输出/输入信号。
25	31	SCK2 PWM0[1]	O	**QSPI #2 串行时钟信号** 此信号为第 2 组 SPI 的时钟信号输出，可用于连接外部 SPI 接口之元件或是模块。 此引脚与 PWM0[1] 为共享脚位。
24	30	SS2# PWM0[0]	O	**QSPI #2 芯片选择信号** 此信号为第 2 组 SPI 的片选输出。 此引脚与 PWM0[0] 为共享脚位。

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
26	32	SDIO2[0] PWM0[2]	IO	**QSPI #2 的数据输出/输入信号** 此信号为第 2 组 QSPI 数据 0 的输出/输入信号。 此引脚与 PWM0[2] 为共享脚位。
27	33	SDIO2[1] PWM0[3]	IO	**QSPI #2 的数据输出/输入信号** 此信号为第 2 组 QSPI 数据 1 的输出/输入信号。 此引脚与 PWM0[3] 为共享脚位。
28	34	SDIO2[2] PWM1[0]	IO	**QSPI #2 的数据输出/输入信号** 此信号为第 2 组 QSPI 数据 2 的输出/输入信号。 此引脚与 PWM1[0] 为共享脚位。
30	36	SDIO2[3] PWM1[1]	IO	**QSPI #2 的数据输出/输入信号** 此信号为第 2 组 QSPI 数据 3 的输出/输入信号。 此引脚与 PWM1[1] 为共享脚位。

2.2.4. 外部串行 Flash 信号

表 2-4：外部串行 Flash 信号

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
45	57	SFCS[1]# GPIOC[4]	IO	**外部 Serial Flash #1 芯片选择信号（默认的片选信号）** 此信号为 LT7589 内部的 LCD 控制器所控制，如果串行 QSPI 功能被禁能，则可以将此引脚设成为 GPIOC[4]，默认为输入功能。 **提示：** 在使用乐升半导体的串口屏功能框架下，只允许 SFCS[1]# 做为外部 QSPI Flash 的片选信号，不可以使用 SFCS[0]# 。
44	56	SFCS[0]# GPIOC[3]	IO	**外部 Serial Flash #0 芯片选择信号（备用的片选信号）** 此信号为 LT7589 内部的 LCD 控制器所控制，如果串行 QSPI 功能被禁能，则可以将此引脚设成为 GPIOC[3]，默认为输入功能。 **提示：** 在使用乐升半导体的串口屏功能框架下，不能使用本信号做为外部 QSPI Flash 的片选信号，必须使用 SFCS[1]# 。而 SFCS[0]# 只有在客户自行裸机开发时方可使用。
46	58	SFCLK GPIOC[0]	IO	**外部 Serial Flash 时钟信号** 此引脚是串行时钟信号输出，为 LT7589 内部的 LCD 控制器所控制，连接到外部 Serial Flash 或是 QSPI 元件。 如果串行 QSPI 功能被禁能，则可以将此引脚设成为 GPIOC[0]，默认为输入功能。

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
47	59	XSIO0 GPIOC[1]	IO	**LT7589 的 QSPI 数据输入/输出信号 0** 此信号为 LT7589 内部的 LCD 控制器所控制，此数据线连到外部的 Serial Flash 或是 QSPI 元件。 如果串行 SPI 功能被禁能，则可以将此引脚设成为 GPIOC[1]，默认为输入功能。
48	60	XSIO1 GPIOC[2]	IO	**LT7589 的 QSPI 数据输入/输出信号 1** 此信号为 LT7589 内部的 LCD 控制器所控制，此数据线连到外部的 Serial Flash 或是 QSPI 元件。 如果串行 SPI 功能被禁能，则可以将此引脚设成为 GPIOC[2]，默认为输入功能。
49	61	XSIO2 GPIOC[5]	IO	**LT7589 的 QSPI 数据输入/输出信号 2** 此信号为 LT7589 内部的 LCD 控制器所控制，此数据线连到外部的 Serial Flash 或是 QSPI 元件。 如果串行 SPI 功能被禁能，则可以将此引脚设成为 GPIOC[5]，默认为输入功能。
50	62	XSIO3 GPIOC[6]	IO	**LT7589 的 QSPI 数据输入/输出信号 3** 此信号为 LT7589 内部的 LCD 控制器所控制，此数据线连到外部的 Serial Flash 或是 QSPI 元件。 如果串行 SPI 功能被禁能，则可以将此引脚设成为 GPIOC[6]，默认为输入功能。

2.2.5. PWM 信号

表 2-5：PWM 信号

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
--	83	LCD_PWM[0] GPIOC[7]	IO	**LCD PWM #0 输出信号** 此信号为 LT7589 内部的 LCD 控制器的寄存器所控制，为一个可程序化的 PWM 输出信号，可以用来控制 TFT LCD 屏的背光或是其他元件。LCD_PWM 的输出模式可经由 LCD 控制器的寄存器来设定。 此引脚与 GPIOC[7] 共享。
56	84	LCD_PWM[1]	IO	**LCD PWM #1 输出信号** 此信号为 LT7589 内部的 LCD 控制器的寄存器所控制，为一个可程序化的 PWM 输出信号，可以用来控制 TFT LCD 屏的背光或是其他元件。LCD_PWM[1] 的输出模式可经由 LCD 控制器的寄存器来设定。

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
24	30	PWM0[0] SS2#	IO	**MCU 控制的第一组 PWM0 输出信号 0** 可作为 PWM 输出或是 GPIO 使用，由内部 MCU 寄存器来设定。 此引脚与 SS2#为共享脚位。
25	31	PWM0[1] SCK2	IO	**MCU 控制的 PWM0 输出信号 1** 可作为 PWM 输出或是 GPIO 使用，由内部 MCU 寄存器来设定。 此引脚与 SCK2 为共享脚位。
26	32	PWM0[2] SDIO2[0]	IO	**MCU 控制的 PWM0 输出信号 2** 可作为 PWM 输出或是 GPIO 使用，由内部 MCU 寄存器来设定。 此引脚与 SDIO2[0] 为共享脚位。
27	33	PWM0[3] SDIO2[1]	IO	**MCU 控制的 PWM0 输出信号 3** 可作为 PWM 输出或是 GPIO 使用，由内部 MCU 寄存器来设定。 此引脚与 SDIO2[1] 为共享脚位。
28	34	PWM1[0] SDIO2[2]	IO	**MCU 控制的第二组 PWM1 输出信号 0** 可作为 PWM 输出或是 GPIO 使用。此引脚与 SDIO2[2] 为共享脚位。
30	36	PWM1[1] SDIO2[3]	IO	**MCU 控制的 PWM1 输出信号 1** 可作为 PWM 输出或是 GPIO 使用，由内部 MCU 寄存器来设定。 此引脚与 SDIO2[3] 为共享脚位。
31	37	PWM1[2]	IO	**MCU 控制的 PWM1 输出信号 2** 可作为 PWM 输出或是 GPIO 使用，由内部 MCU 寄存器来设定。
32	38	PWM1[3]	IO	**MCU 控制的 PWM1 输出信号 3** 可作为 PWM 输出或是 GPIO 使用，由内部 MCU 寄存器来设定。

2.2.6. USB 信号

表 2-6：USB 接口信号

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
9	13	DP	IO	**USB 数据端（Positive）** 此为 USB 数据端 DP 的信号。

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
10	14	DM	IO	**USB 数据端（Negative）** 此为 USB 数据端 DM 的信号。

2.2.7. GPIO 与中断信号

表 2-7：中断信号

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
15	20	INT0[0] AIN[0]	I	**中断 INT0 信号** 可作为中断输入使用或是模拟信号输入。
17	22	INT0[1] AIN[1]	I	**中断 INT0 信号** 可作为中断输入使用或是模拟信号输入。
19	24	INT0[2] AIN[2]	I	**中断 INT0 信号** 可作为中断输入使用或是模拟信号输入。
21	26	INT0[3] AIN[3]	I	**中断 INT0 信号** 可作为中断输入使用或是模拟信号输入。
16	21	INT0[4] AIN[4]	I	**中断 INT0 信号** 可作为中断输入使用或是模拟信号输入。
18	23	INT0[5] AIN[5]	I	**中断 INT0 信号** 可作为中断输入使用或是模拟信号输入。
20	25	INT0[6] AIN[6]	I	**中断 INT0 信号** 可作为中断输入使用或是模拟信号输入。
22	27	INT0[7] AIN[7]	I	**中断 INT0 信号** 可作为中断输入使用或是模拟信号输入。
36	42	INT2[0] TXD0	I	**中断 INT2 信号** 可作为中断输入使用。 此引脚与 TXD0 为共享脚位。
37	43	INT2[1] RXD0	IO	**中断 INT2 信号** 可作为中断输入使用。 此引脚与 RXD0 为共享脚位。
38	44	INT2[2] TXD1	I	**中断 INT2 信号** 可作为中断输入使用。 此引脚与 TXD1 为共享脚位。
39	45	INT2[3] RXD1	IO	**中断 INT2 信号** 可作为中断输入使用。 此引脚与 RXD1 为共享脚位。

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
92	127	INT2[4] TXD2	I	**中断 INT2 信号** 可作为中断输入使用。 此引脚与 TXD2 为共享脚位。
4	7	INT2[5] RXD2	IO	**中断 INT2 信号** 可作为中断输入使用。 此引脚与 RXD2 为共享脚位。
40	46	INT2[6] CANTX	I	**中断 INT2 信号** 可作为中断输入使用。 此引脚与 CANTX 为共享脚位。
41	47	INT2[7] CANRX	IO	**中断 INT2 信号** 可作为中断输入使用。 此引脚与 CANRX 为共享脚位。
35	41	INT1[0] CLKOUT	IO	**中断 INT1 信号** 可作为中断输入使用。 此引脚与 CLKOUT 为共享脚位。
14	19	INT1[3]	IO	**中断 INT1 信号** 可作为中断输入使用。
34	40	INT1[6] RSTOUT	IO	**中断 INT1 信号** 可作为中断输入使用。 此引脚与 RSTOUT 为共享脚位。
42	48	INT1[7]	IO	**中断 INT1 信号** 可作为中断输入或是 GPIO 使用。
--	55	LCD_INT#	O	**LCD 中断输出信号** 当 LCD 控制器设定的中断条件发生，此脚变成低电位，用来产生一中断输出告知 MCU。
--, 50, 49, 45, 44, 48, 47, 46	83, 62, 61, 57, 56, 60, 59, 58	GPIOC[7] GPIOC[6:0]	IO	**LCD 控制器的 GPIO 输出/输入信号** 此信号为 LT7589 内部的 LCD 控制器的寄存器所控制，GPIOC[7] 的输出数据与 LCD_PWM[0] 共享引脚。 GPIOC[6:0] 与 { XSIO3, XSIO2, SFCS[1]#, SFCS[0]#, XSIO1, XSIO0, SFCLK } 共享引脚，只有在 LCD_PWM 与 SPI Master 的功能被禁止时才能使用。这些引脚的输出模式可经由 TFT LCD 控制器的寄存器来设定。

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
80, 63, 79, 78, 70, 69, 62, 61	112, 94, 111, 110, 101, 100, 93, 92	GPIOD[7:0]	IO	**LCD 控制器的 GPIO 输出/输入信号** 此信号为 LT7589 内部的 LCD 控制器的寄存器所控制，GPIOD[7:0] 的输出数据与{ PD[18], PD[2], PD[17], PD[16], PD[9], PD[8], PD[1], PD[0] } 共享引脚。 GPIOD[7:0] 只有在 LCD 屏幕数据总线设成 16-bits 时才能使用。这些引脚的输出模式可经由 TFT LCD 控制器的寄存器来设定。
--	54~49	LCD_IOA[7:2]	IO	**LCD 控制器的 GPIO 输出/输入信号** 这些引脚的输出/输入模式可经由 TFT LCD 控制器的寄存器来设定。

2.2.8. ADC 模拟信号

表 2-8：ADC 模拟输入信号

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
22, 20, 18, 16, 21, 19, 17, 15	27, 25, 23, 21, 26, 24, 22, 20	AIN[7:0] INT0[7:0]	IO	**模拟信号输入** 这些模拟信号用作 ADC 模拟输入通道。 当未配置为模拟输入时，这些信号也可用于 INT0[7:0]。

2.2.9. 其他控制信号

表 2-9：其他控制信号

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
43	77	LCD_WAIT#	O	**等待输出信号** 当内部 MCU 对 LCD 控制电路进行读写控制时，如果处于忙碌状态，会将 WAIT#变成低电位，用来告知 MCU 进入等待周期。
--	125	LCD_CS# NC	I	**LCD 控制电路片选信号** LCD_CS# = 0，代表内部 MCU 对 LCD 控制电路进行命令或是数据读写周期。 注意：此引脚内部已经与 MCU 的 EBI_CS#对接，务必保持 NC。

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
91	124	LCD_RST#	I	**LCD 控制电路复位输入信号** 当 RST# = 0 时，并且维持大于 32 个时钟周期长度，LT7586 将产生复位动作。
--	126	SCL LCD_RST#	IO	**I2C 时钟信号** 此信号为 MCU 的 I2C 的时钟信号或是 GPIO 使用。 注意：此引脚应用上与 Pin-124 的 LCD_RST#连接，用来控制 LCD 电路是否进行复位动作。
4	7	SWDCK RXD2, INT2[5]	I	**程序烧录时钟信号** 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。 此引脚与 RXD2, INT2[5] 为共享脚位。
92	127	SWDIO TXD2, INT2[4]	I	**程序烧录数据信号** 此输入信号是用于对内部闪存进行程序烧录时的数据信号。 此引脚与 TXD2, INT2[4] 为共享脚位。
35	41	CLKOUT INT1[0]	O	**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，此信号也可用于 INT1[0]。
34	40	RSTOUT INT1[6]	O	**MCU 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，该信号也可用于 INT1[6]。
33	39	RESET#	I	**MCU 复位输入信号** 当 RESET# = 0 时，将对内部 MCU 产生复位动作。

2.2.10. 电源与时钟信号

表 2-10：电源与时钟信号

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
87	120	LCD_XI	I	**晶振（Crystal）/ 时钟信号输入** 此引脚连接至外部晶振，为内部 TFT LCD 控制器的晶振电路输入信号，当使用有源晶振或是外部时钟信号可以由此脚输入，通常与 Pin 脚的 XTAL 时钟信号接在一起，晶振频率（OSC）建议为 12MHz。
88	121	LCD_XO	O	**晶振（Crystal）输出** 此引脚连接至外部晶振，为内部 TFT 控制器的晶振电路输出信号。
13	18	OSC32K_XI	I	**32.768Khz 晶振输入** RTC 时钟信号，此引脚连接至外部 32.768Khz 晶振。

脚号 LT7589A	脚号 LT7589B	引脚名称	I/O	功 能 说 明
12	17	OSC32K_XO	O	**32.768Khz 晶振输出** RTC 时钟信号，此引脚连接至外部 32.768Khz 晶振。
8	12	XTAL	I	USB 时钟信号，此引脚连接至外部 12Mhz 晶振。
7	10	EXTAL	O	USB 时钟信号，此引脚连接至外部 12Mhz 晶振。
11	16	VBAT	PWR	**3.3V~3.6V 电池电源输入（RTC）** 须外接滤波电容到确保供电稳定。
53	73	VDD33A	PWR	**3.3V 电源输入（LCD）** 靠近引脚端必须外接一个 10uF 和一个 0.1uF 滤波电容到地。 **提示：**此电源输入应独立供电，勿与 VDD33_IO 或 AVDD 直接连在一起。
5, 29	8, 15, 35	VDD33B	PWR	**3.3V 电源输入（MCU）** 此引脚须外接一个 1uF 和 0.1uF 滤波电容到地，并确保供电稳定。
51, 52, 54	63, 68, 81	VDD33_IO1	PWR	**3.3V 电源输入（I/O）** 这些引脚须外接一个 2.2uF 和 0.1uF 滤波电容到地，并确保供电稳定。
81	90, 102, 113	VDD33_IO2	PWR	**3.3V 电源输入（I/O）** 这些引脚须外接一个 2.2uF 和 0.1uF 滤波电容到地，并与 VDD33_IO1 分开路径确保供电稳定，参考原理图。
23, 89	28, 122	AVDD	PWR	**内部仿真电路的电源输入（Analog）** 提供 3.3V 电压，靠近引脚端必须外接一个 1uF 和一个 0.1uF 滤波电容到地。
6	9	VDD12	PWR	**1.2V 内核电源输出（MCU）** 靠近引脚端必须外接一个 1uF 和一个 0.1uF 滤波电容到地。
55, 71, 90	64, 70, 123	LCD_V12	PWR	**1.2V 内核电源输出（LCD）** 靠近引脚端只须外接一个 0.01uF 滤波电容到地即可。 **提示：**外接电容勿超过 0.01uF，并不可与 VDD12 接连在一起。
96	11, 29, 65, 128	VSS	PWR	**GND 接地**
--	82, 91, 103	VSS_C	PWR	**内核 GND 接地**
0	--	Thermal Pad	-	**散热焊盘** LT7589B 的封装背部散热焊盘，必须直接接地。 **提示：**为了达到更好的焊接效果，在 PCB Layout 时建议参考后面图 28-3 的说明。

2.2.11. 使用不同 TFT 屏的 IO 口资源

表 2-11：LT7589A IO 口资源数量表

芯片 / 基本功能	LT7589A (QFN-96)	
TFT 屏接口	RGB 565 接口	RGB 888 接口
最大分辨率	1280x800 (串口协议模式)	1280x800 (串口协议模式)
Flash	2MB	2MB
SRAM	256KB+512KB	256KB+512KB
Display RAM	16MB	16MB
USB 2.0	V (DP/DM)	V (DP/DM)
RTC	V	V
串口通讯	V (RXD1, TXD1)	V (RXD1, TXD1)
Ext. SPI Flash	V (758_SF)	V (758_SF)
背光控制	V (758_PWM1)	V (758_PWM1)
SWD 烧录口	V (SWD)	V (SWD)

TP 类型 / 可用 IO 口	xTP (1)	CTP	RTP (2)	xTP	CTP	RTP
IO 口总数量	38	34	33	30	26	25
IO 口种类说明	GPIO (2), GINT, PWM, SCI, QSPI, ADCIN	GPIO (2), GINT, PWM, SCI, QSPI, ADCIN	GPIO (2), GINT, PWM, SCI, QSPI, ADCIN	GINT, PWM, SCI, QSPI, ADCIN	GINT, PWM, SCI, QSPI, ADCIN	GINT, PWM, SCI, QSPI, ADCIN

表 2-12：LT7589B IO 口资源数量表

芯片 / 基本功能	LT7589B (LQFP-128)	
TFT 屏接口	RGB 565 接口	RGB 888 接口
最大分辨率	1280x800 (串口协议模式)	1280x800 (串口协议模式)
Flash	2MB	2MB
SRAM	256KB+512KB	256KB+512KB
Display RAM	16MB	16MB
USB 2.0	V (DP/DM)	V (DP/DM)
RTC	V	V
串口通讯	V (RXD1, TXD1)	V (RXD1, TXD1)
Ext. QSPI Flash	V (758_SF)	V (758_SF)
背光控制	V (758_PWM1)	V (758_PWM1)
SWD 烧录口	V (SWD)	V (SWD)

TP 类型 / 可用 IO 口	xTP (1)	CTP	RTP (2)	xTP	CTP	RTP
IO 口总数量	46	42	41	38	34	33
IO 口种类说明	GPIO (2), GINT, PWM, SCI, QSPI, ADCIN	GPIO (2), GINT, PWM, SCI, QSPI, ADCIN	GPIO (2), GINT, PWM, SCI, QSPI, ADCIN	GPIO, GINT, PWM, SCI, QSPI, ADCIN	GPIO, GINT, PWM, SCI, QSPI, ADCIN	GPIO, GINT, PWM, SCI, QSPI, ADCIN

提示：

(1) : xTP 表示不带 TP

(2) : GPIO 增加 8 个

(3) : RTP 需增加一 RTP 控制芯片 (如 TSC2046)

3. 串口屏功能说明

LT7589 内部结合了高效能的 32-bits RISC MCU 及 TFT LCD 图形显示控制器（以下简称 TFT LCD 控制器），此 MCU 的主体架构与乐升半导体的 LT32U05 相同，具有 2Mbytes Flash 程序空间及 768Kbytes SRAM，其功能可以直接参考后面第 4~25 章有关的 MCU 内核及各个控制模块的规格。另外 TFT LCD 图形显示控制器是采用乐升半导体的 LT7586 硬件架构，具有 128Mbytes 的显存及 **JPG 解码器**，其与 32-bits MCU 的连接方式如下图所示：

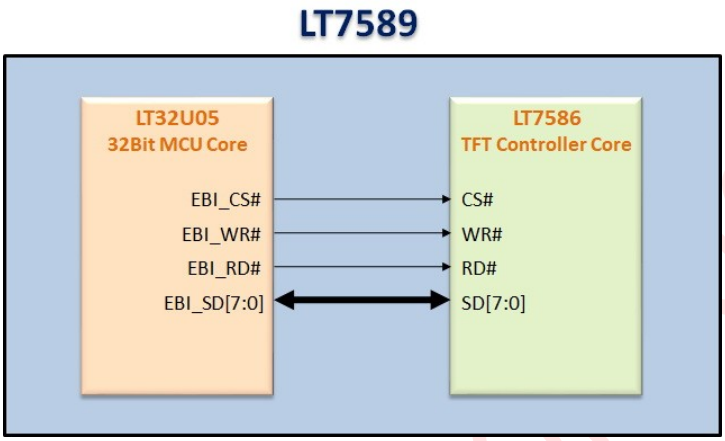


图 3-1：LT7589 内部 MCU 与 TFT 图形加速器的连接图

3.1. 串口通讯接口

当使用 LT7589 作为 TFT 串口屏控制器时，它与主控端 MCU 通信的模式就是透过 Uart 接口，LT7589 有三组 SCI（Uart）串口，默认是采用第二组 TXD1、RXD1 与主控端 MCU 通信，至于 LT7589 内部 MCU 程序的开发相关设置可以参考 LT7589 串口屏应用手册，乐升半导体也提供完整的开发环境或烧录工具。

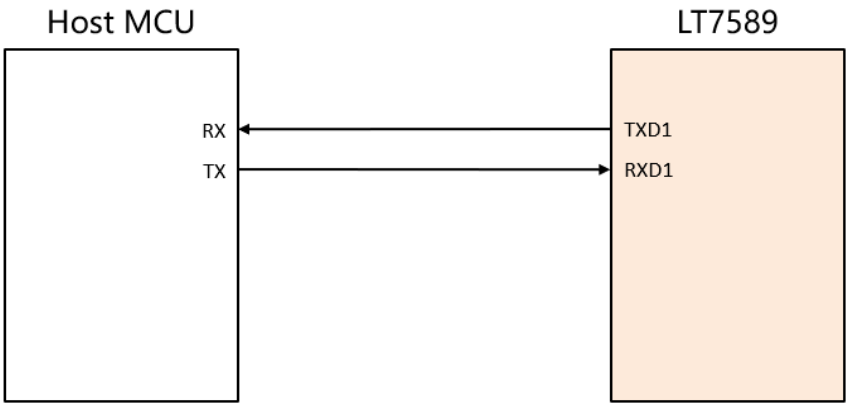


图 3-2：与主控端 MCU 的通信模式

当作为串口屏芯片开发时，LT7589 内部 MCU 程序会提供串口协议程序，用户可以用乐升半导体提供的串口屏开发软件 UI-Editor，将已经设计好的图片、动画等 UI 结构流程导入，完成 TFT 显示的开发，基本上不需要修改 LT7589 内部 MCU 程序，也不需要了解 LT7589 内部寄存器及控制方式，主控端 MCU 程序只需要依据产品应用负责发送串口协议的指令格式，及接收、解读 LT7589 发出的反馈信息，因此在 TFT 屏显示开发工作上节

省许多时间。

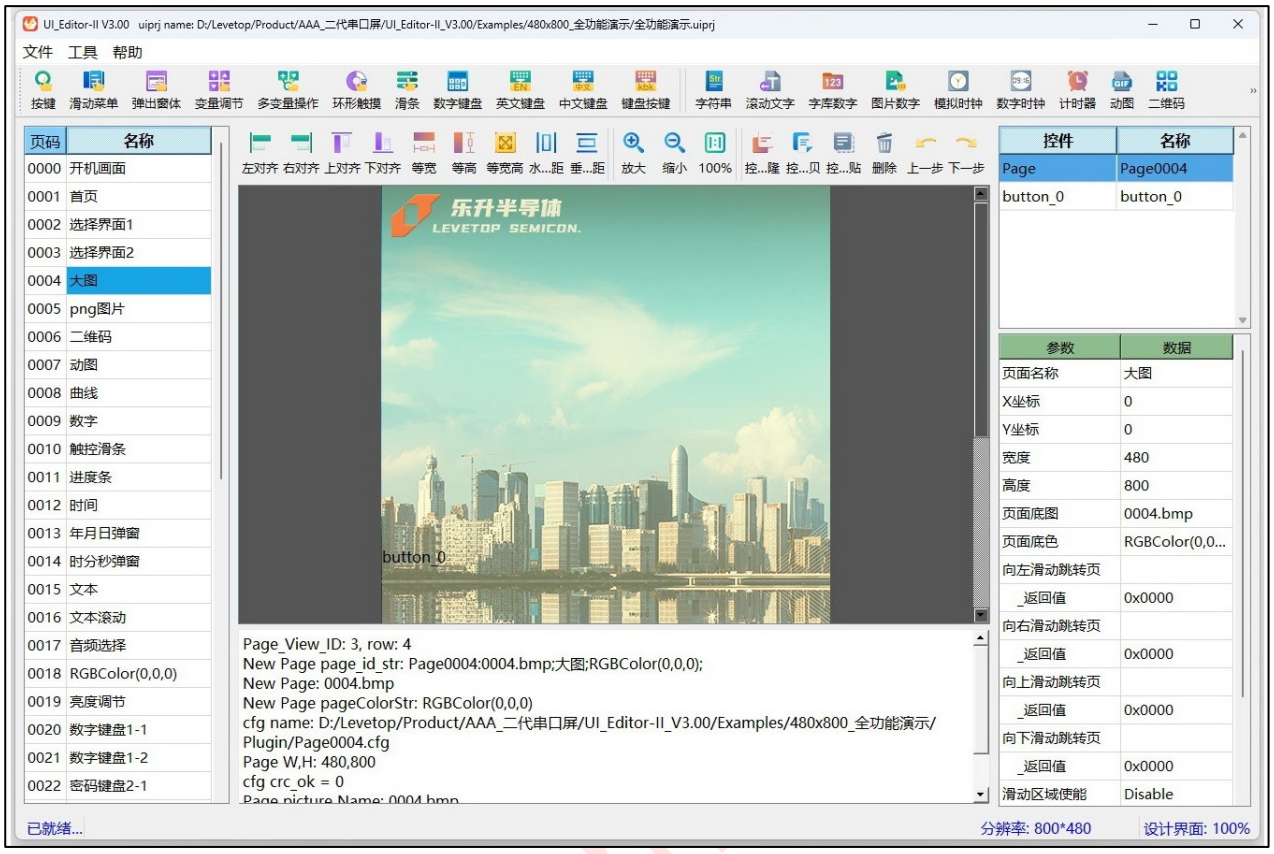


图 3-3：UI-Editor 工具画面

提示：LT7589 串口屏芯片开发时，内部 2Mbytes Flash 用来储存引导程序及串口协议程序，在样品开发阶段，乐升半导体 提供的样片会包含引导程序，方便用户更新串口协议程序及自行开发或修改的代码，量产阶段则提供的芯片不包含任何程序（Flash 为空片），用户必须在生产时自行烧录，有关代码的更新烧录请参考 LT7589 烧录说明书。

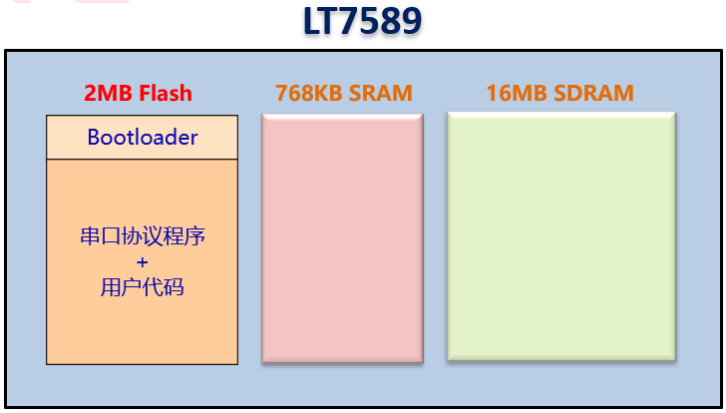


图 3-4：TFT 串口屏模块

3.2. UI_Editor-III 串口屏协议表

LT7589 支持乐升半导体开发的第三代 UI_Editor-III 串口屏开发软件，UI_Editor-III 是采用变量控制模式，串口通信与指令如下表：

表 3-1：UI_Editor-III 串口通信与指令差异

模式	UI_Editor-III
串口通信	由“**帧头+长度+读/写指令码+变量数据+CRC**”构成，帧头可修改，支持 Modbus，I2C。
串口指令	串口指令只进行变量读写操作，串口与功能不一一对应，单个功能数目不受限制。

下表为使用 UI_Editor-III 模式下 LT7589 支持的串口屏协议表：

表 3-2：UI_Editor-III 串口通信指令格式

指令模式	帧头 (2 Bytes)	长度 (1Byte)	指令码 (1Byte)	变量地址 (2 Bytes)	数据 1 (2*n Bytes)	数据 2	CRC (2 Bytes)
写入变量指令 (串口发送数据)	0x5AA5	0xXX	0x10 (写指令)	0x0000~0x5FFF	写入数据 (2*n Bytes)	NULL	0xXXXX
读取变量指令	0x5AA5	0xXX	0x03 (读指令)	0x0000~0x5FFF	读出的 Word 数量 (2 Bytes)	NULL	0xXXXX
返回指令格式	0x5AA5	0xXX	0x03 (读指令)	0x0000~0x5FFF	读出的 Word 数量 (2 Bytes)	数据 (2*n Bytes)	0xXXXX
触摸反馈指令	0x5AA5	0xXX	0x41	变量地址 (寄存器地址)	返回的 KeyValue (2 Bytes)	NULL	0xXXXX

有关 UI_Editor-III 串口通信模式请参考 UI_Editor-III 串口屏使用说明书。

3.3. 二次开发

LT7589 由于含有高容量的 Flash 及 SRAM，也可以作为一个带 TFT 控制器的主控 MCU 来使用，将主控及 TFT 显示功能由一颗 LT7589 来完成，或者是当客户使用 LT7589 时需要用到内部的 MCU 资源，或做些程序修改进行二次开发，此时需要参考第 4 章起的内部 32-bits MCU 规格说明，或是和乐升半导体的 FAE 联系。

3.4. 芯片接口应用

前面章提到主控端 MCU 通信的模式就是透过 Uart（0）接口，而其他的接口将在本章节描述。

3.4.1. TFT LCD 屏的接口

LT7589 与标准的 RGB 屏连接使用到数据线及 4 个控制信号：HSYNC、VSYNC、PDE、PCLK（如下图所示），如果采用 RGB:888 则用到 24 条数据线，如果采用 RGB:565 则用到 16 条数据线，没用到的数据线可以当作 IO 口使用，如下表 3-3 所示。实际应用原理图可以参考第 29 章。

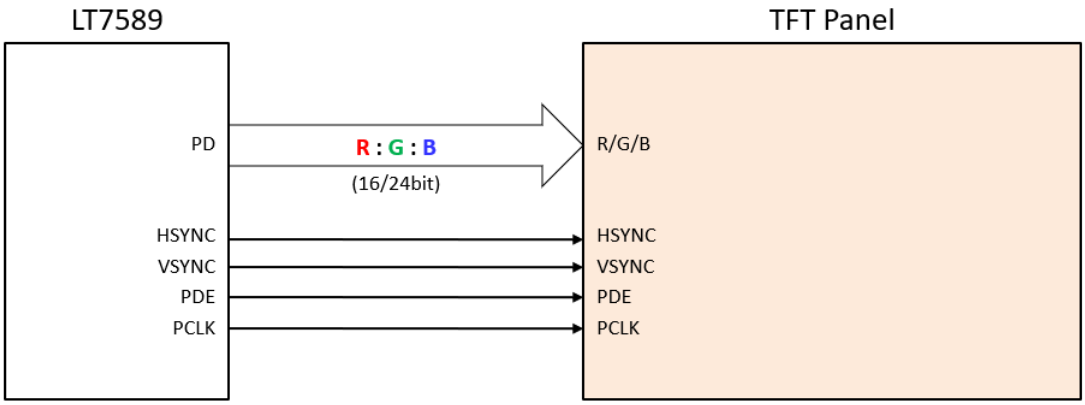


图 3-5：LT7589 与 TFT LCD 屏的连接

表 3-3：RGB 数据与颜色深度

RGB 接口数	LCD 数据线	色彩深度
R:G:B = 8:8:8	PD[23~0]	16.7M 色
R:G:B = 5:6:5	PD[23~19], PD[15~10], PD[7~3]	65K 色

3.4.2. SSI（Synchronous Serial Interface）串行接口

3.4.2.1. TFT LCD 控制器端的 SSI 接口

LT7589 作为串口屏芯片，其 TFT LCD 控制器端的 SSI 接口是专门连接到外部的 QSPI Flash，大部分的 LCD 显示素材是存放在这个接口的外部 Flash 内，LT7589 采用高速的 DMA 传输模式，透过快速的 QSPI 接口读取 UI 素材数据，然后写入到内部 SDRAM 显示内，再透过内部扫描电路将 SDRAM 数据送到 TFT LCD 屏。LT7589 TFT 控制器可以同时连接 2 颗的外部 Flash，支持 NOR 或是 NAND Type Flash，如果是使用 NAND Flash，LT7589 还支持坏块处理。在使用乐升半导体的串口屏功能框架下，只允许 SFCS[1]# 做为外部 QSPI Flash 的片选信号，而 SFCS[0]#只保留给客户自行裸机开发时方可使用，如下图所示。

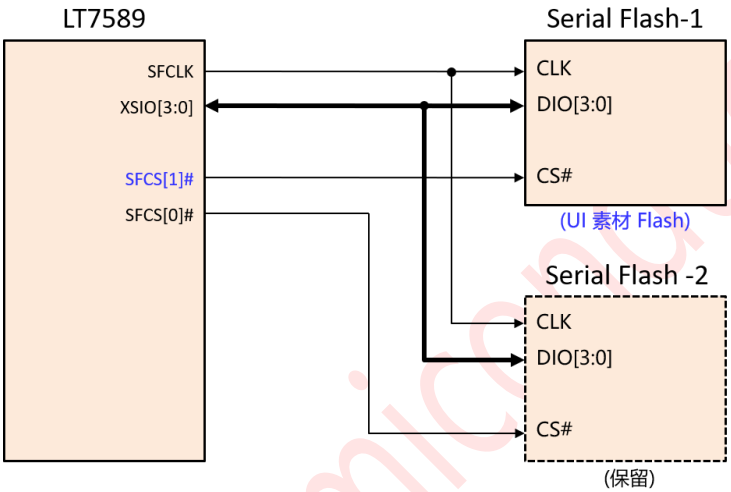


图 3-6：LT7589 存放素材的 QSPI Flash 连接示意图

3.4.2.2. MCU 端的 SSI 接口

LT7589 除了 TFT LCD 控制器提供 QSPI 接口外，在 MCU 端也提供了 2 组完全独立的 SSI 同步串行接口，可以连接 SPI Flash 或是其他 SPI 接口的元器件，如下图所示。其中第 2 组的 QSPI 接口与 PWM0 及 PWM1 部分引脚共享，请参考第 2.2.3、2.2.5 节。客户如果想二次开发控制这些 SSI 信号，请参考第 21 章说明。同时这些 SSI 信号也可以设置成 GPIO 使用，请参考第 6 章的芯片配置模块（CCM）说明。

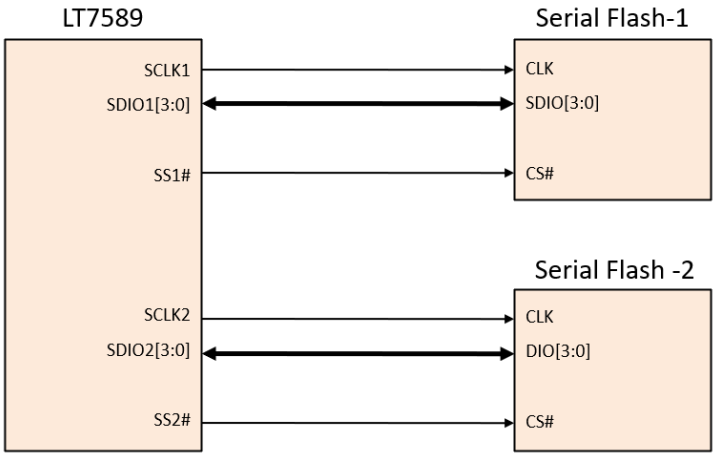


图 3-7：LT7589 MCU 端的 QSPI Flash 连接示意图

3.4.3. SCI（Serial Communication Interface）串行接口

LT7589 提供 3 组 SCI 串口，前面提到第二组 TXD1、RXD1 默认是与主控端 MCU 通信用，其他 2 组可以用来连接蓝牙模块、WiFi 模块，或其他串口元件，如下图所示。TXD0、RXD0、TXD2、RXD2 接口也可以透过寄存器设置成 INT 中断输入或是 GPIO 口，请参考第 2.2.1、2.2.7 节相关的引脚信号说明。使用者如果想二次开发控制这些 SCI 输入信号，请参考第 20 章说明。

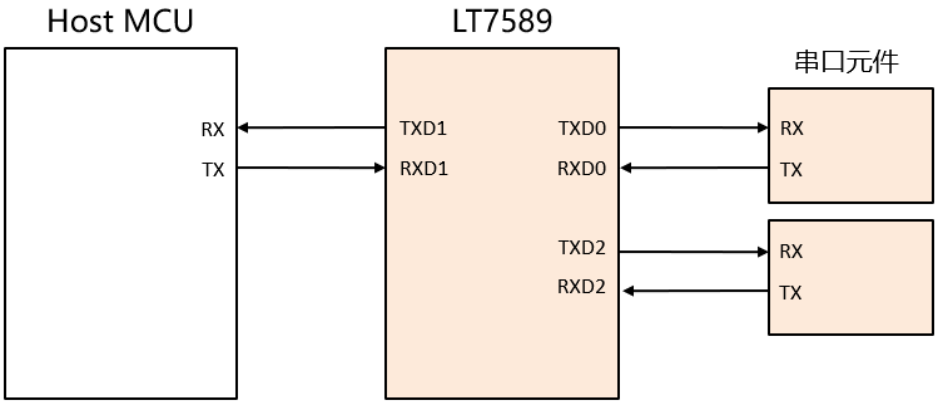


图 3-8：LT7589 提供 3 组 SCI 串口

3.4.4. ADC 模拟输入接口

LT7589 提供了 8 个模拟输入接口，这 8 个模拟输入透过寄存器设置也可以各自单独变成中断输入接口，如下图所示。AIN 接口与 INT0 引脚共享，请参考第 2.2.8 节。使用者如果想二次开发控制这些 ADC 输入信号，请参考第 25 章说明。LT7589 内部还自带比较器（Comparator），可以透过 ADC 模拟输入接口来进行，详细请参考第 23 章说明。

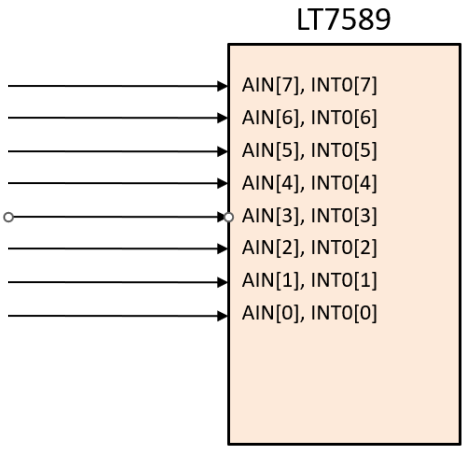


图 3-9：LT7589 提供 8 个模拟输入接口

3.4.5. GPIO 与中断输入接口

3.4.5.1. MCU 端的 GPIO 与中断输入

在前面第 2.2.7 节引脚信号说明提到 LT7589 在 MCU 端有三组中断信号：INT0, INT1, INT2，其中 INT0 与 AIN 共享，INT2 与 SCI 和 Canbus 串口信号共享，如下图所示。LT7589 的 MCU 端有一个嵌入式中断控制器（EIC），使用者如果想透过二次开发使用这些中断输入信号，请参考第 6 章芯片配置模块（CCM）及第 7 章嵌入式中断控制器（EIC）的说明。

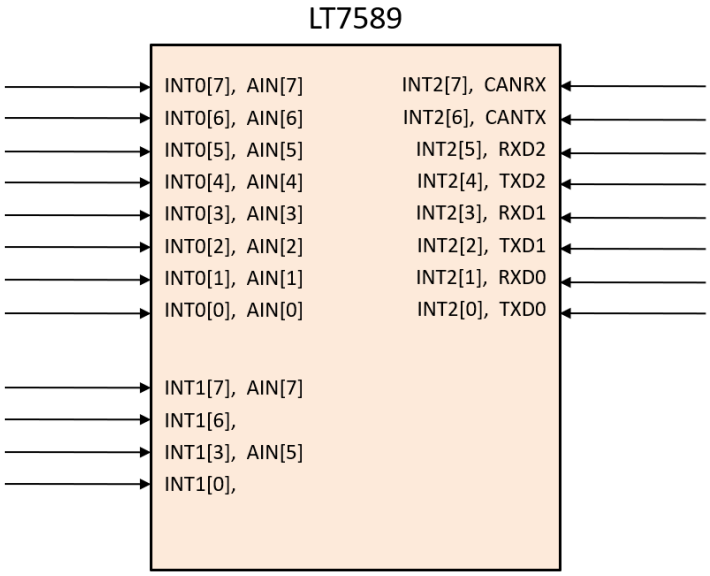


图 3-10：LT7589 MCU 端中断输入

3.4.5.2. TFT LCD 控制器的 GPIO

在 TFT LCD 控制器内也提供一些 GPIO 接口，可以作为 MCU 接口的延伸，其中 GPIOD[7:0] 接口是与 LCD 屏幕数据线共享脚位（如下表所示），LT7589B 还多了一组 LCD_IOA[7:2] ，这些 GPIO 接口是设定为输出或是输入，其输出数据或是读取输入数据的 TFT LCD 控制器寄存器为 REG[F5-F6h]，请参考后面第 27.2 节有关 TFT LCD 控制器的 GPIO 寄存器说明。

表 3-4：GPIO 接口与其他控制信号共享脚位

TFT LCD 控制器供的 GPIO 接口	共享信号
GPIOD[7:0]	PD[18, 2, 17, 16, 9, 8, 1, 0] 共享引脚，只有在 LCD 屏幕数据总线设成 16-bits 时才能使用。这些引脚的输出模式可经由寄存器来设定。
GPIOC[7]	LCD_PWM[0] 共享引脚
LCD_IOA[7:2]	None（LT7589B 才有）

3.4.6. PWM 输出接口

3.4.6.1. MCU 端的 PWM 输出

LT7589 在 MCU 端也提供了 2 组 PWM 输出，给组有 4 个通道，如下图所示。其中 PWM0[3:0]、PWM1[1:0]接口与第 2 组的 QSPI 引脚共享，请参考第 2.2.5 节。客户如果想二次开发控制这些 PWM 输出信号，请参考第 22 章说明。

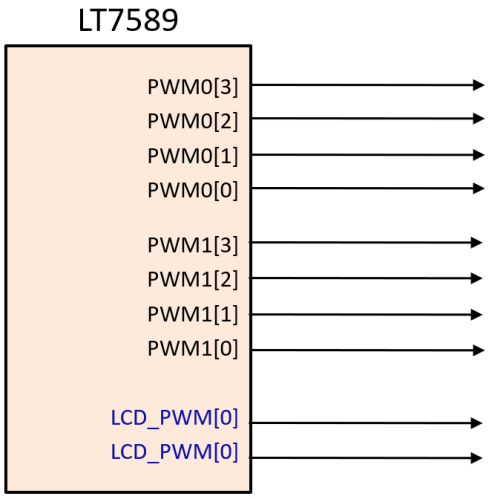


图 3-11：LT7589 MCU 端的 PWM 输出

3.4.6.2. TFT LCD 控制器端的 PWM 输出

LT7589 在 TFT LCD 控制器内还提供 2 个 PWM 输出：LCD_PWM[0]、LCD_PWM[1]（如上图）。TFT 控制器内部含有 2 个 16-bits 的计数器 Timer-0 和 Timer-1，其动作关系着 LCD_PWM 的输出状态。以 LCD_PWM[0]为例，在使用前必须先设置 Timer-0 计数寄存器（REG[8Ah-8Bh]，TCNTB0）及 Timer-0 计数比较寄存器（REG[88h-89h]，TCMPB0]），启动 PWM 功能后，Timer-0 计数器会先加载 TCNTB0 的值，并依照 PWM Clock 的设定频率开始下数，当 Timer-0 计数器下数的值等于 TCMPB0 寄存器的值时 PWM 就会动作，也就是 LCD_PWM[0] 如果原本为 0 就转态为 1，而 Timer-0 计数器依然会继续下数，当 Timer-0 继续下数等于 0 时会产生中断，LCD_PWM[0] 回到原本的准位 0，同时会自动加载寄存器 TCNTB0 的值，这就是代表一个完整的 PWM 周期。

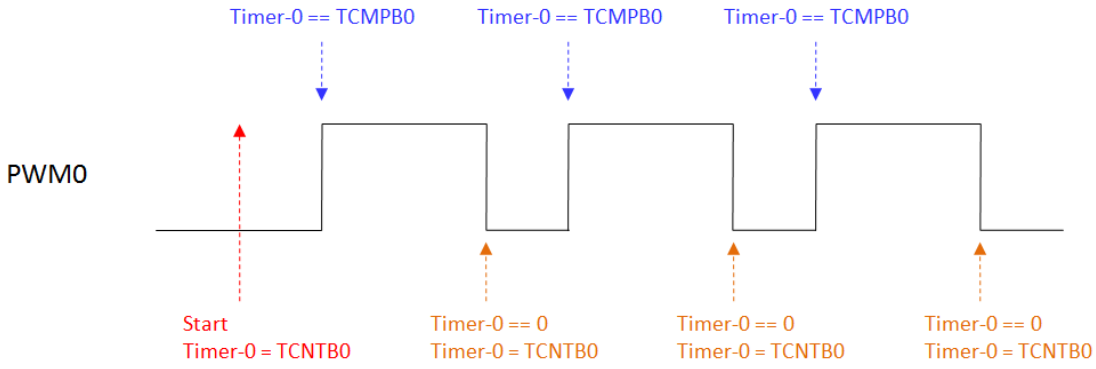


图 3-12：LCD_PWM 波形图

由以上动作可以了解，LCD_PWM[0] 的 Duty 决定于比较寄存器（REG[88h-89h], TCMPB0]），例如想要藉由 LCD_PWM[0] 产生一个近似 DC 准位的电压，当 LCD_PWM[0] 初始设定为 0，想要产生的等效电压高，那么 TCMPB0 值就要设大一点；当 LCD_PWM[0] 初始设定为 1，想要产生的等效电压高，那么 TCMPB0 值就要设小一点。

要注意的是 REG[86h]（PCFGR）的自动加载功能必须开启，Timer-0 等于 0 才会自动重载寄存器 TCNTB0 的值，因此如果在 Timer-0 等于 0 之前 MCU 变更 TCNTB0 或是 TCMPB0 的值，就可以产生不同 Duty 的动态 PWM 波型。

■ LCD_PWM 时序来源

LCD_PWM 的计数器时序来自 CCLK、Timer-0 和 Timer-1 的时序基频由寄存器 PSCLR（REG[84h]）决定：

时序基频 = CCLK / （Prescaler + 1）

而送到 Timer 的 Clock 再由各自的除频寄存器（REG[85h]）决定，每个计数器的除频器可以产生 4 种不同的除频选择：1、1/2、1/4、1/8。例如除频寄存器 REG[85h] bit[5:4] = 10b，则 Timer-0 的计数 Clock = 时序基频 / 4，请参考后面第 26 章有关 TFT LCD 控制器的寄存器 REG[84h] 与 REG[85h] 说明。

■ LCD_PWM 输出信号

LCD_PWM 除了脉波之外也可以设定固定的高电平或低电平，如果是 LCD_PWM[0]，首先要关闭自动重载功能，寄存器 REG[86h]（PCFGR）的 bit1 = 0，及停止 Timer-0 计数，寄存器 REG[86h]（PCFGR）的 bit0 = 0，如果 Timer-0 < TCMP0 则输出值为高电平，如果 Timer-0 > TCMP0，则输出值为低电平（假设反相位被关闭）。LCD_PWM[0] 的输出可以经由 PCFG bit2 设定输出值反相。

此外，LCD_PWM[0] 和 LCD_PWM[1] 是共享的输出脚位，它可以做为其他用途使用，请参考寄存器 REG[85h]（PMUXR）bit[3:0] 的说明。

表 3-5：寄存器 REG[85h] 说明

Bit	说　明
3-2	**LCD_PWM[1] 功能设定（LCD_PWM[1] Function Control）** 0xb：LCD_PWM[1] 输出系统错误旗标（Scan FIFO pop 错误或是内存存取超过范围）。 10b：LCD_PWM[1] 输出 PWM 计数器 1 的波形或是 PWM 计数器 0 的反相波形（dead zone 使能）。 11b：LCD_PWM[1] 输出 Oscillator 晶振频率（OSC）。

1-0	**LCD_PWM[0] 功能设定（LCD_PWM[0] Function Control）** 0xb：LCD_PWM[0] 为 GPIO-C[7]。 10b：LCD_PWM[0] 输出 PWM 计数器 0。 11b：LCD_PWM[0] 输出系统频率。

LCD_PWM[0] 和 LCD_PWM[1] 也可以设成互补输出，此情况下 LCD_PWM[1] 输出是跟随着 LCD_PWM[0] 的设定与控制，只是它是 LCD_PWM[0] 的反向输出状态：

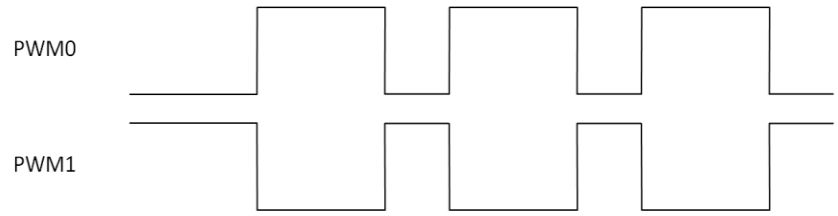


图 3-13：LCD_PWM[0] 和 LCD_PWM[1] 互补输出

而在某些应用上为了避免 LCD_PWM[0] 和 LCD_PWM[1] 同时转态，造成电流过大引起的干扰，LT7589 提供了盲区时序控制，错开 LCD_PWM[0] 和 LCD_PWM[1] 同时转态时间，盲区间格由寄存器 REG[87h]（DZ_LENGTH）来设定：

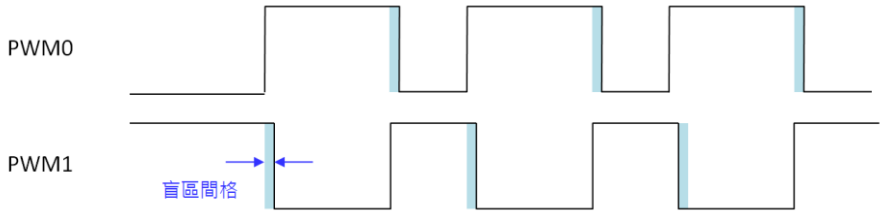


图 3-14：LCD_PWM[0] 和 LCD_PWM[1] 互补输出的盲区时序

提示：使用者可以参考后面第 27.1 节有关 TFT LCD 控制器的 PWM 寄存器说明。

3.4.7. USB 接口

LT7589 提供一组 USB 接口，具有 USB 从机功能。此 USB 接口可用于连接用于更新 LT7589 内部 MCU 程序的 PC 和外部 SPI 闪存的数据。有关更多详细信息，请参阅 UI_Editor-III 用户手册或第 24 章说明。

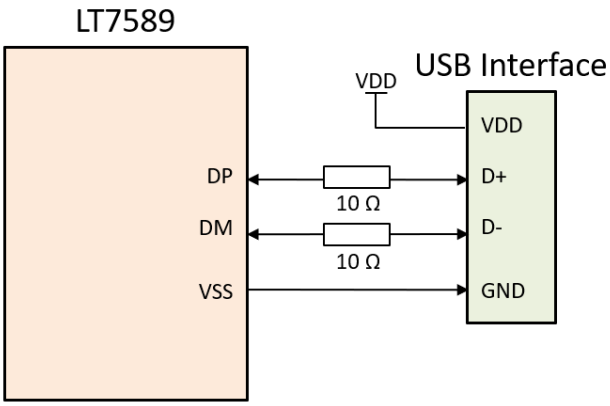


图 3-15：LT7589 USB 接口

3.4.8. Canbus 接口

LT7589 提供 Can Bus 总线接口 CANTX 和 CANRX，在实际应用上需要外接一 Canbus 驱动芯片，参考如下的原理图，更多的 Can Bus 应用信息，请参考第 19 章说明。

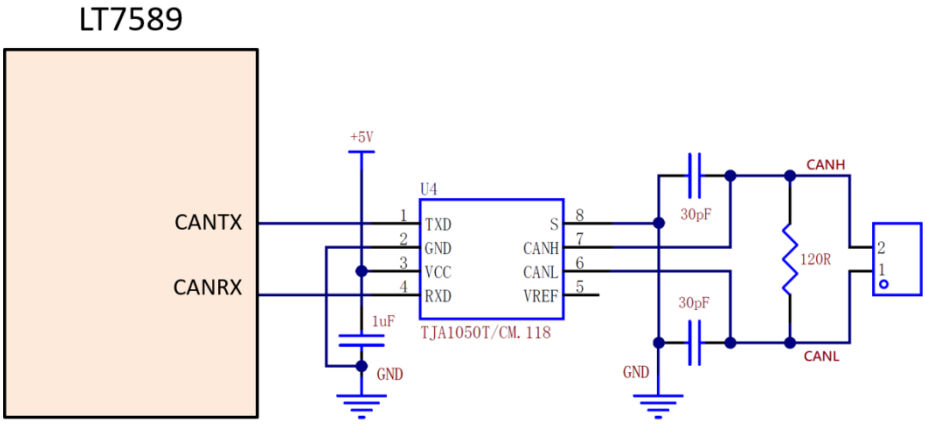


图 3-16：Canbus 接口应用范例

3.4.9. 时钟信号

3.4.9.1. MCU 时钟信号

LT7589 内部的 Clock 时钟信号如下图所示，MCU 部分包括一外部 12MHz 晶振电路，其内部 PLL 电路产生时钟信号供给 CPU、SPI Flash、Uart 等电路使用。

LT7589 的 MCU 的时钟由时钟和电源控制模块（CLKPWRM）来设置，细请参考第 9 章说明。

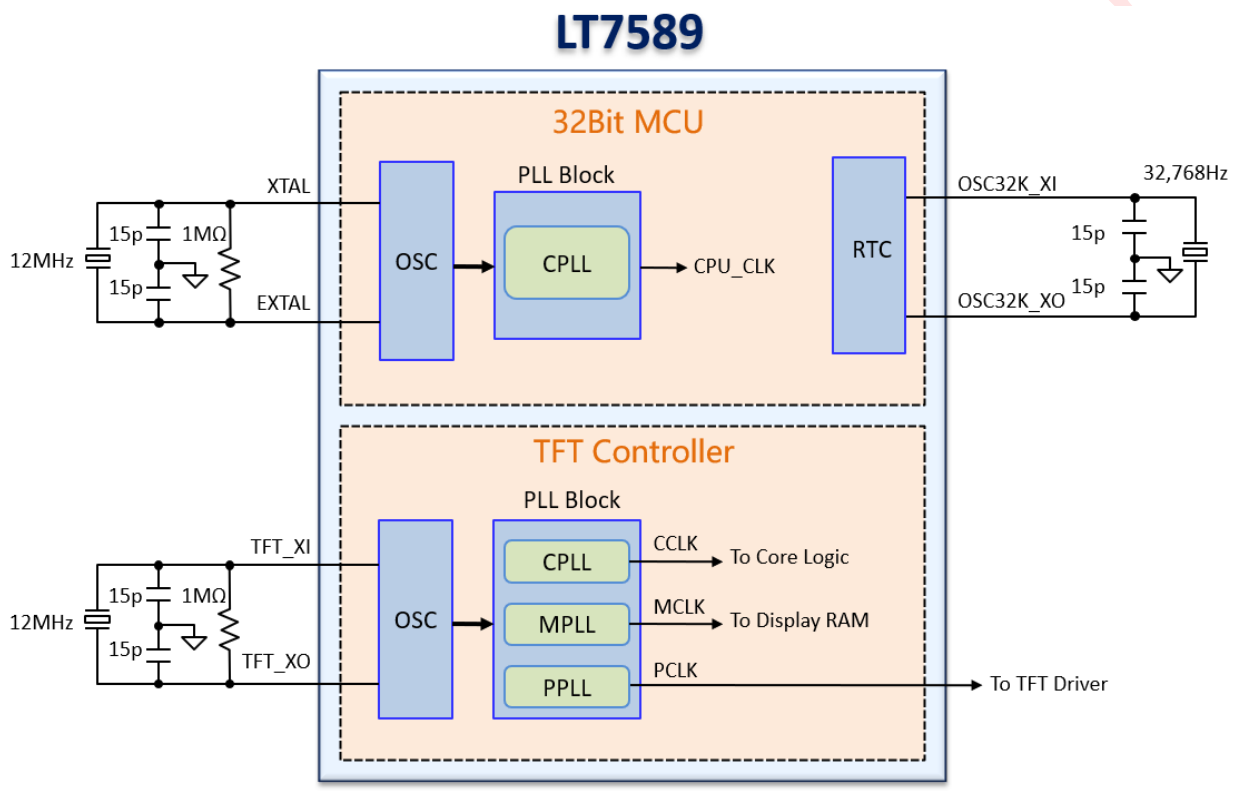


图 3-17：LT7589 时钟信号结构图

3.4.9.2. TFT LCD 控制器时钟信号

TFT LCD 控制器也包括一外部 12MHz 晶振电路，其内部有 3 个 PLL 电路，用来产生时钟信号供给 TFT LCD 图形显示控制器的 GUI 内部逻辑电路、SDRAM 及外部 LCD Driver 电路使用。

提示： TFT LCD 控制器时钟信号源可以藉由 MCU 的一组 PWM 输出产生，或是使用引用 MCU 的 12MHz 晶振信号（如下图所示），以省下一颗晶振成本。

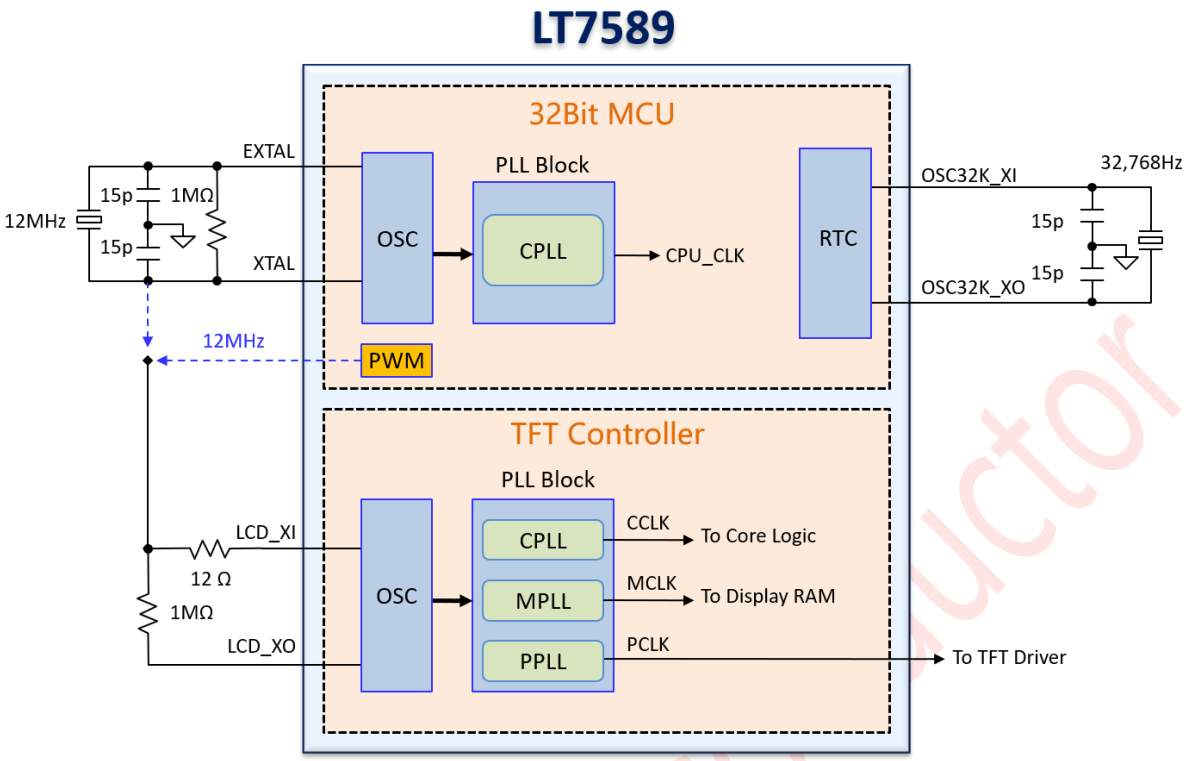


图 3-18：TFT 控制器使用外部 12MHz 晶振电路

3.4.9.3. RTC 时钟信号

LT7589 内部有一个 RTC 模块，具有独立供电（VBAT）与低功耗的 32.768KHz 时钟晶振（如上图所示）及 RTC 电路，外部电源切断时可以在电池运作下继续进行计时工作。请参考第 17 章时钟控制器（RTC）说明及第 29 章原理图。

3.4.10. 复位信号

LT7589 的 MCU 与 TFT LCD 控制器有各自独立的复位信号（如下图 3-19 所示），MCU 的复位信号可以用简单的 RC 电路产生，若是担心电源开关频繁引发的不稳定，则建议采用一个复位芯片来产生复位信号（如下图 3-20 所示）。

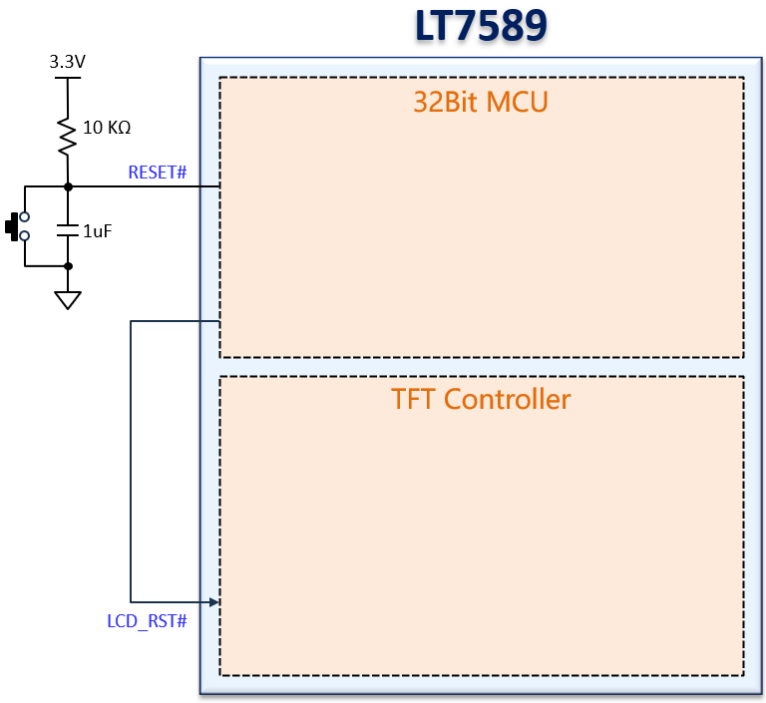


图 3-19：LT7589 复位信号结构图

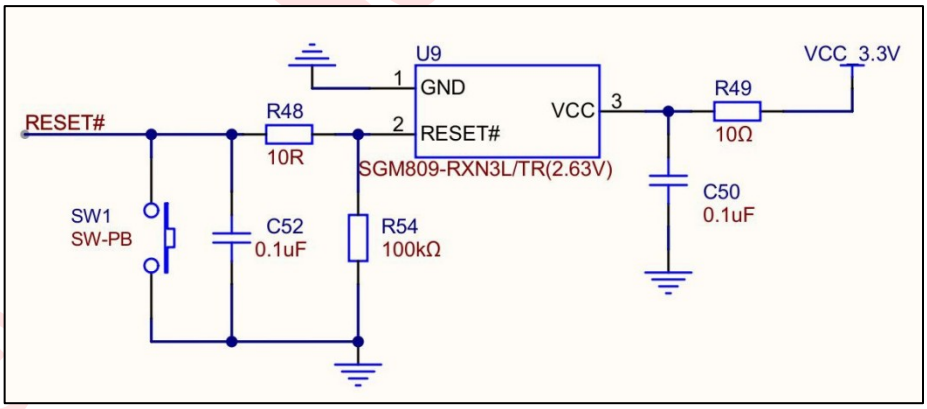


图 3-20：由复位芯片产生复位信号

TFT LCD 控制器的复位信号可用 MCU 的一根 IO 口来产生，请参考第 29 章的应用电路。

LT7589 的 MCU 带有一个复位控制模块（RCM），用来确定复位的原因，向系统确认适当的复位信号，然后保持复位原因的历史记录，也可用来控制复位输出信号 RSTOUT，详细请参考第 10 章复位控制模块（RCM）说明。

4. 32 位 RISC 微处理器介绍

本章描述了 LT7589 内部 32 位 RISC 微处理器的功能，它基于乐升半导体的 32 位 MCU – LT32U05 为主体的 M*Core 指令集/架构，为低功耗和成本敏感的嵌入式控制应用程序而设计。对于较小的尺寸和功耗，32 位 RISC 是建立在一个新的 3 级管道冯-诺伊曼(Von-Neumann)架构上。

32 位的 RISC 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。在 32 位的 RISC 设计中，有更多可配置的选项。通过利用这些可配置的选项，在性能、功能和成本之间的权衡将更加灵活。32 位 RISC 的门计数从 12K 到 20K 不等。

4.1. 功能特点

32 位 RISC 的主要特点如下：

- 32 位负载/存储减少指令集计算机（RISC）体系结构，具有固定的 16 位指令长度
- 16 条目 32 位通用寄存器文件
- 高效的 3 阶段执行管道，隐藏在应用软件中
- 单周期指令执行多个指令，分支执行三个周期
- 支持字节/半字段的内存访问
- 嵌入式中断控制器，支持嵌套的向量中断和低功耗模式唤醒
- 单周期 32 位 x32 位硬件整数乘法器数列
- 3~13 循环硬件整数分法器阵列
- AHB-lite 外部总线

4.2. 微体系结构总结

32 位的 RISC 利用一个 3 阶段的管道来执行指令。指令获取、指令解码/寄存器文件读取和执行/回写阶段以重叠的方式运行，允许对大多数指令执行单个时钟指令。

此 RISC 为源操作数和指令结果提供了 16 个通用寄存器。寄存器 R15 被用作链接寄存器，以保持子程序调用的返回地址，并且寄存器 R0 按约定与当前堆栈指针值相关联。提供了一个双入口 32 位指令缓冲区，以允许指令预取从每个时钟周期的存储器中获得两个指令，从而减少或消除与数据存储器访问的总线资源冲突。统一的总线结构足以维持指令和数据带宽的需求，而不诉诸于昂贵的双总线结构。

此 RISC 为字节、半字和字（32 位）数据提供了内存加载和存储操作，字节和半字加载数据。这些指令可以流水线化，以允许短序列有效的单周期吞吐量。与数据相关的操作可以在两个时钟周期内完成。加载和存储多个寄存器指令允许低开销的上下文保存和恢复操作；这些指令可以在（N 个 + 1）时钟周期中执行，其中 N 是要传输的寄存器数。提供了一个单一的条件代码/携带(C)位，用于条件测试和用于实现大于 32 位的算术和逻辑操作。通常，C 位仅通过显式的测试/比较操作来设置，而不是作为正常指令操作的副作用。对于需要将条件设置与实际计算结合起来的专门操作，此规则会出现例外。

4.3. 程序设计模型

32 位 RISC 编程模型分别为两种特权模式定义：管理员和用户。HPROT[1]位用于表示特权模式。程序根据所指示的模式访问寄存器。用户程序只能访问特定于用户模式的寄存器；在管理模式下执行的系统软件可以访问所有寄存器，并使用控制寄存器来执行管理功能。因此，可以限制用户程序访问特权信息，并且操作系统通过协调用户程序的活动来为用户程序执行管理和服务任务。

所有的指令都可以在任何一种模式下执行。用户程序还可以执行停止、睡觉或等待指令。陷阱#n 指令为用户程序提供对操作系统服务的受控访问。为了防止用户程序进入监督模式，除非以受控的方式，可以改变程序状态寄存器（PSR）中的 S-位的指令是有特权的。

当设置了 PSR 中的 S 位时，处理器将在监督模式下执行指令。与指令相关联的总线周期根据模式指示主管或用户访问。处理器在正常用户模式处理时利用用户编程模型。在异常处理期间，处理器将从用户模式更为监控模式。异常处理保存 PSR 的当前值以堆栈内存，然后在 PSR 中设置 S 位，迫使处理器进入监督模式。要返回到以前的操作模式，系统例程可能会执行 RTE（从异常返回）指令，导致指令管道从适当的地址空间被刷新和重新填充。

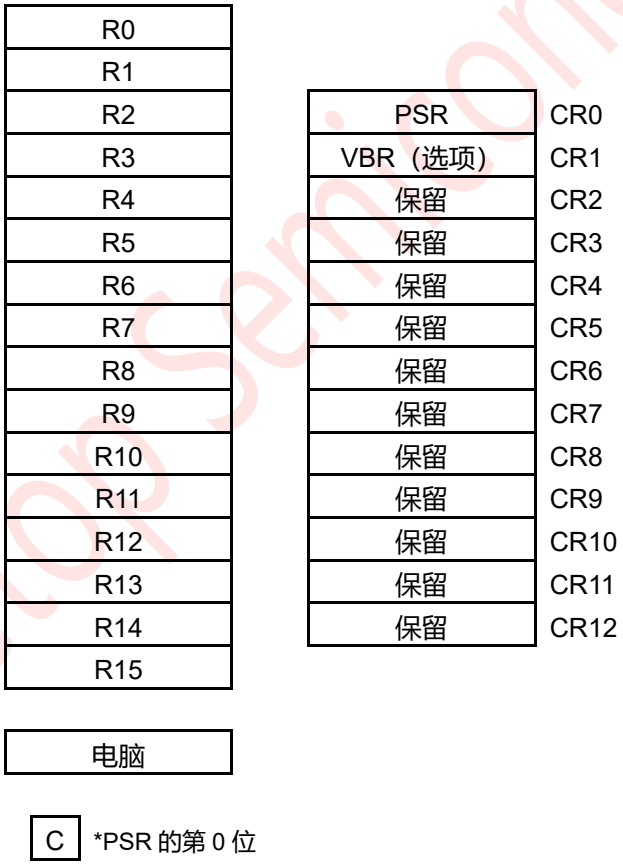


图 4-1：编程模型

编程模型中描述的寄存器（参考图 4-1)提供了操作数存储和控制。用户编程模型由 16 个通用的 32 位寄存器、32 位程序计数器（PC）和条件/携带(C)位组成。C 位被实现为 PSR 的第 0 位。按照惯例，寄存器 R15 被用作子例程调用的链接寄存器，而寄存器 R0 通常被用作当前的堆栈指针。

4.4. 数据格式汇总

整数单位支持的操作数数据格式是标准的 2 的补充数据格式。每个指令的操作数大小可以用指令显式编码（加载/存储指令），也可以由指令操作（索引操作、字节提取）隐式定义。通常，指令对源操作数的所有 32 位都进行操作，并生成 32 位的结果。

根据处理器配置，可以从大或小端字节顺序角度查看内存（参考图 4-2）。在 Big Endian 模式（默认操作模式）中，字符 0 的最重要字节（字节 0）位于地址 0 处。对于小环境模式，字 0 的最重要的再见位于地址 3。在寄存器中，位在以位开始的字中编号（参考图 4-3）。按照惯例，寄存器的字节 0 如何都是最重要的字节。这只是在执行 XTRB[0-3]指令时出现的一个问题。

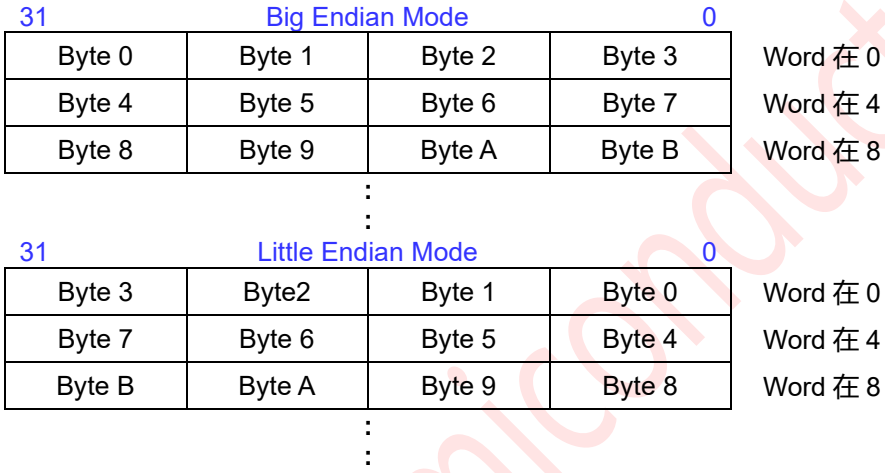


图 4-2：内存中的数据结构

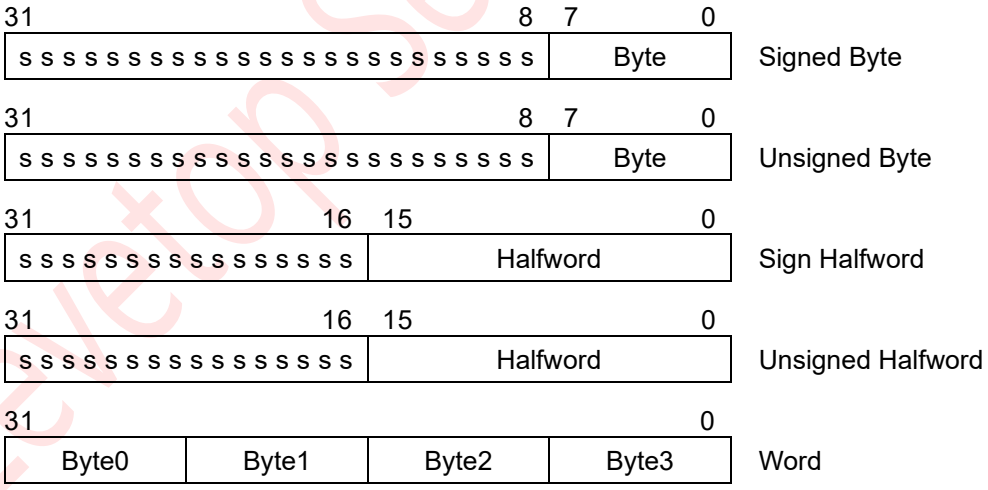


图 4-3：寄存器中的数据结构

4.5. 操作和寻址能力

32 位的 RISC MCU 通过加载和存储指令访问所有的内存操作数，在通用寄存器（GPRs）和内存之间传输数据。寄存器 + 4 位缩放位移寻址模式用于加载和存储指令来地址字节、半字或字（32 位）数据。

加载和存储多个指令允许 16 个 GPR 的一个子集传输到寄存器 R0（约定的默认堆栈指针）指向的基本地址。

加载和存储寄存器象限指令使用寄存器间接寻址传输一个寄存器象限到或从存储器。

4.6. 指令集概述

该指令集是为支持高级语言而定制的，并针对那些最常执行的指令进行了优化。提供了一套标准的算术和逻辑指令，以及对位操作、字节提取、数据移动、控制流修改的指令支持，以及一小组有条件执行的指令，可用于消除短的条件分支。

下表提供了 32 位 RISC 指令集的字母顺序列表。

表 4-1：32 位 RISC 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
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
BTSTI	Bit Test Immediate
CLRF	Clear Register on Condition False
CLRT	Clear Register on Condition True
CMPHS	Compare Higher or Same
CMPLT	Compare Less-Than
CMPLTI	Compare Less-Than Immediate
CMPNE	Compare Not Equal
CMPNEI	Compare Not Equal Immediate
DECF	Decrement on Condition False
DECGT	Decrement Register and Set Condition if Result Greater-than Zero
DECLT	Decrement Register and Set Condition if Result Less-than Zero
DECNE	Decrement Register and Set Condition if Result Not Equal to Zero
DECT	Decrement On Condition True
DIVS (1)	Divide Signed Integers
DIVU (1)	Divide Unsigned Integers
DOZE	Doze
FF1 (1)	Find First One
INCF	Increment on Condition False
INCT	Increment On Condition True
IXH	Index Halfword
IXW	Index Word
JAVASW	Java interpreter switch
JMP	Jump
JMPI	Jump Indirect
JSR	Jump to Subroutine
JSRI	Jump to Subroutine Indirect
LD.[BHW]	Load
LDM	Load Multiple Registers
LDQ	Load Register Quadrant
LRW	Load Relative Word
LSL, LSR	Logical Shift Left and Right
LSLC, LSRC	Logical Shift Left and Right, update C bit
LSLI, LSRI	Logical Shift Left and Right by Immediate
MFCR	Move from Control Register
MOV	Move
MOVI	Move Immediate
MOVF	Move on Condition False
MOVT	Move on Condition True
MTCR	Move to Control Register

助记符号	描 述
MULSH	Multiply signed Halfwords
MULT	Multiply
MVC	Move C bit to Register
MVCV	Move Inverted C bit to Register
NOT	Logical Complement
OR	Logical Inclusive-OR
ROTLI	Rotate Left by Immediate
RSUB	Reverse Subtract
RSUBI	Reverse Subtract Immediate
RTE	Return from Exception
RFI	Return from Interrupt
SEXTB	Sign-extend Byte
SEXTH	Sign-extend Halfword
ST.[BHW]	Store
STM	Store Multiple Registers
STQ	Store Register Quadrant
STOP	Stop
SUBC	Subtract with C bit
SUBU	Subtract
SUBI	Subtract Immediate
SYNC	Synchronize
TRAP	Trap
TST	Test Operands
TSTNBZ	Test for No Byte Equal Zero
WAIT	Wait
XOR	Exclusive OR
XSR	Extended Shift Right
XTRB0	Extract Byte 0
XTRB1	Extract Byte 1
XTRB2	Extract Byte 2
XTRB3	Extract Byte 3
ZEXTB	Zero-extend Byte
ZEXTH	Zero-extend Halfword

提示(1)： 在当前版本中未实现。

5. 内存与寄存器配置

5.1. 简要说明

LT7589 中 MCU 的内建/外部内存、寄存器包括：

- 内建 2M Bytes QSPI Flash 闪存，作为存放串口屏协议代码及客户二次开发所增加或修改的代码
- 内建 8K Bytes 内部 Boot ROM
- 内建 768K Bytes 内部静态 SRAM
  - 系统 RAM：前 256K Bytes，地址由 0x00800000 开始
  - 显存 RAM：后 512K Bytes，地址由 0x00840000 开始
- 最高 128M Bytes 的外部 QSPI Flash 闪存空间
- 各模块内部寄存器

5.2. 地址配置图

图 5-1：地址配置图

表 5-1：硬件模块与寄存器地址的配置图

配置地址	最大区块	硬 件 模 块	参考章节
0x4000_0000	64Kbyte	Direct Memory Access Controller (DMAC)	13
0x4001_0000	64Kbyte	Chip Configuration Module (CCM)	6
0x4002_0000	64Kbyte	Reset Control Module (RCM)	10
0x4003_0000	64Kbyte	Clock and Power Control Module (CLKPWRM)	9
0x4004_0000	64Kbyte	Programmable Interrupt Timer 0 (PIT0)	15
0x4005_0000	64Kbyte	Programmable Interrupt Timer 1 (PIT1)	15
0x4006_0000	64Kbyte	Programmable Interrupt Timer 2 (PIT2)	15
0x4007_0000	64Kbyte	Programmable Interrupt Timer 3 (PIT3)	15
0x4008_0000	64Kbyte	Serial Communication Interface 1 (SCI1)	20
0x4009_0000	64Kbyte	Serial Communication Interface 0 (SCI0)	20
0x400A_0000	64Kbyte	Analog Comparator 0 (COMP0)	23
0x400B_0000	64Kbyte	Analog Comparator 1 (COMP1)	23
0x400C_0000	64Kbyte	Serial Communication Interface 2 (SCI2)	20
0x400D_0000	64Kbyte	Pulse Width Modulator 0 (PWM0)	22
0x400E_0000	64Kbyte	Pulse Width Modulator 1 (PWM1)	22
0x400F_0000	64Kbyte	Edge Port Module 0 (EPORT0)	18
0x4010_0000	64Kbyte	Edge Port Module 1 (EPORT1)	18
0x4011_0000	64Kbyte	Analog-to-Digital Convertor (ADC)	25
0x4012_0000	64Kbyte	Option Byte (OPB)	14
0x4013_0000	64Kbyte	WatchDog Timer (WDT)	16
0x4014_0000	64Kbyte	Real Time Controller (RTC)	17
0x4015_0000	64Kbyte	Reserved.	--
0x4016_0000	64Kbyte	USB2.0 Full-Speed Device Controller (USBC)	24
0x4017_0000	64Kbyte	Reserved.	--
0x4018_0000	64Kbyte	Reserved.	--
0x4019_0000	64Kbyte	CACHE Module (CACHEM)	12
0x401A_0000	64Kbyte	Reserved.	--
0x401B_0000	64Kbyte	Reserved.	--
0x401C_0000	64Kbyte	CANBus Controller (CANBC)	19
0x401D_0000	64Kbyte	Edge Port Module 2 (EPORT2)	18
0x6000_0000	64Kbyte	Synchronous Serial Interface 0 (SSI0) - QSPI0	21
0x7000_0000	64Kbyte	Synchronous Serial Interface 1 (SSI1) - QSPI1	21
0x8000_0000	64Kbyte	Synchronous Serial Interface 2 (SSI2) - QSPI2	21
0xE000_0000	4Kbyte	Embedded Interrupt Controller (EIC)	7
0xE000_1000	4Kbyte	Embedded Programmable Timer (EPT)	8

6. 芯片配置模块（CCM）

6.1. 基本介绍

配置模块（CCM）控制着 LT7589 内 MCU 的配置。

6.2. 功能特点

CCM 可执行以下操作:

- 配置唤醒功能
- 配置 LDO 模式
- 配置 IO 功能

6.3. 内存映射和寄存器

本章节提供了对内存映射和寄存器的描述。CCM 基地址为 0x4001_0000。下表显示了 CCM 寄存器的偏移量地址。

6.3.1. 内存映射

表 6-1：CCM 内存映射

偏移地址	Bits[31:16]	Bits[15:0]	访问权限 [1]
0x0000	WKUPC — Wakeup Configuration Register		S
0x0004	保留		S
0x0008	保留		S
0x000C	CPPDC — Chip Pin Pull Down Configuration Register		S
0x0010	保留		S
0x0014	QSPIXIPCR — QSPI XIP Mode Configuration Register		S
0x0018	保留		S
0x001C	QSPIKEYR — QSPI 32-Bit Key Register		S
0x0020	QSPIGPIOCR — QSPI GPIO Configuration Register		S
0x0024	MCURAMPRIOCR — MCU Access RAM Priority Configuration Register		S
0x0028	EPORT2FCR — EPORT2 Function Configuration Register		S

提示:

S = 仅主管访问。在用户模式下，只访问主管的地址位置没有影响，并导致周期终止传输错误。

6.3.2. 寄存器描述

6.3.2.1. 唤醒配置寄存器（WKUPC）

地址：CCM_BASEADDR + 0x0000_0000

	31	30	29	28	27	26	25	24
读: 写:	WKUPFILTEREN	WKUPSEN [30:24]						
复位:	0	1	1	1	1	1	1	1

	23	22	21	20	19	18	17	16
读: 写:	WKUPSEN [23:16]							
复位:	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8
读: 写:	WKUPSEN [15:8]							
复位:	1	1	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
读: 写:	WKUPSEN [7:0]							
复位:	1	1	1	1	1	1	1	1

图 6-1：唤醒配置寄存器（WKUPC）

WKUPFILTEREN — Wakeup Source Filter Enable

如果设置了 WKUPFILTEREN，唤醒源将通过过滤器去除 Glith，然后将芯片从待机模式唤醒。

1 = 唤醒源过滤器开启

0 = 唤醒源过滤器被禁用

WKUPSEN[30:0] — Wakeup Source Enable

此字段控制是否将相应的来源用作将芯片从待机模式唤醒的来源。如果设置，则将相应的来源用作唤醒的来源。。

表 6-2：WKUPSEN 和相应的唤醒源

WKUPSEN	Wakeup Source
WKUPSEN[30]	INT2[3]
WKUPSEN[29]	INT2[2]
WKUPSEN[28]	INT2[1]
WKUPSEN[27]	RTC Interrupt
WKUPSEN[26]	PVD Interrupt
WKUPSEN[25]	USB Resume
WKUPSEN[24]	COMP1 Interrupt

WKUPSEN	Wakeup Source
WKUPSEN[23]	COMP0 Interrupt
WKUPSEN[22]	INT2[6]
WKUPSEN[21]	INT2[5]
WKUPSEN[20]	I2C
WKUPSEN[19]	WDT0 Interrupt
WKUPSEN[18]	JTAG POWER ON REQUEST
WKUPSEN[17]	RESET# Pin
WKUPSEN[16]	WDT0 Reset
WKUPSEN[15]	INT1[7]
WKUPSEN[14]	INT1[6]
WKUPSEN[13]	INT1[5]
WKUPSEN[12]	INT1[4]
WKUPSEN[11]	INT1[3]
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]

6.3.2.2. 芯片引脚下拉配置寄存器（CPPDC）

地址：CCM_BASEADDR + 0x0000_000C

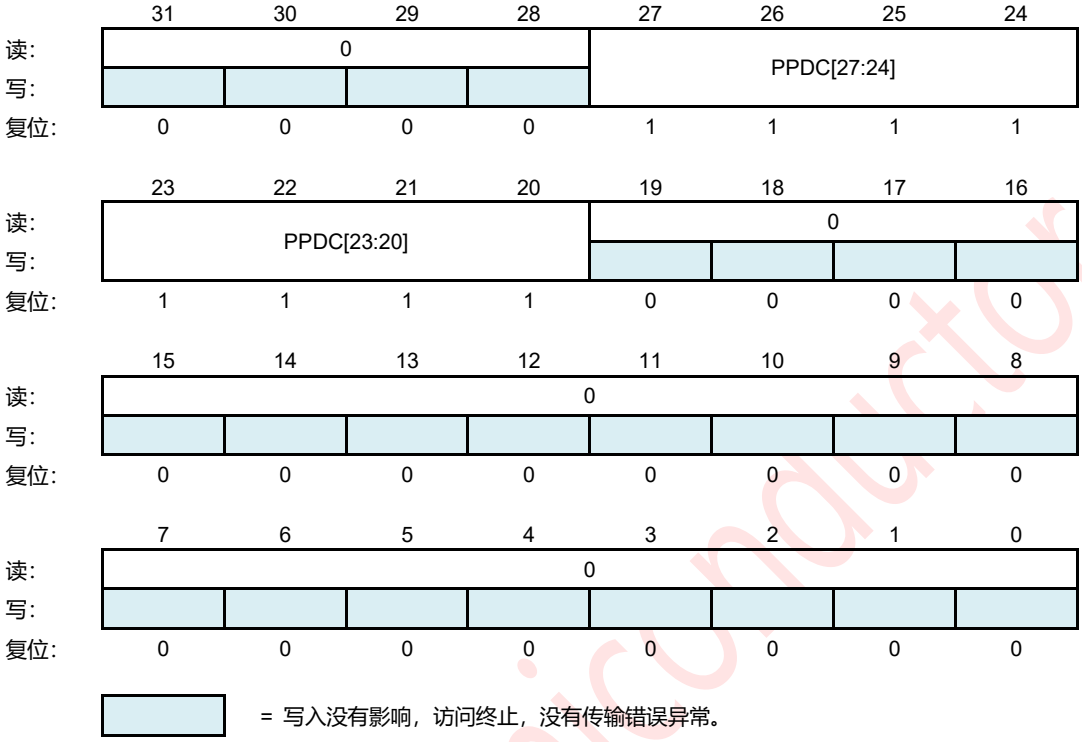


图 6-2：芯片引脚下拉配置寄存器（CPPDC）

PPDC[27:20] — Pin Pull Down Configuration Field

此读/写字段控制相应引脚的下拉功能，如下表所示。

1 = 启用相应引脚的下拉电阻功能

0 = 相应引脚的下拉电阻功能被禁用

表 6-3：芯片引脚下拉配置

引脚名	下拉配置位	下拉功能
PWM0[3]	PPDC[27]	0：禁用，1：启用
PWM0[2]	PPDC[26]	0：禁用，1：启用
PWM0[1]	PPDC[25]	0：禁用，1：启用
PWM0[0]	PPDC[24]	0：禁用，1：启用
PWM1[3]	PPDC[23]	0：禁用，1：启用
PWM1[2]	PPDC[22]	0：禁用，1：启用
PWM1[1]	PPDC[21]	0：禁用，1：启用
PWM1[0]	PPDC[20]	0：禁用，1：启用

6.3.2.3. QSPI XIP 模式配置寄存器（QSPI）

QSPIXIPMCFR 寄存器是一个可读写寄存器。

地址：CCM_BASEADDR + 0x0000_0014

	31	30	29	28	27	26	25	24
读:	0						QSPI2_DATA_ENCR_EN	QSPI2_XIPEN
写:								
复位:	0	0	0	0	0	0	0	0

	23	17	16	20	19	18	17	16
读:	0						QSPI1_DATA_ENCR_EN	QSPI1_XIPEN
写:								
复位:	0	0	0	0	0	0	0	0

	15	9	8	12	11	10	9	8
读:	0						QSPI0_DATA_ENCR_EN	QSPI0_XIPEN
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 6-3：QSPI XIP 模式配置寄存器（QSPI）

QSPIx_DATA_ENCR_EN — QSPIx XIP Transfer Data Encrypt function Enable Control bit

1 = QSPIx XIP 传输数据加密功能被启用

0 = QSPIx XIP 传输数据加密功能被禁用

QSPIx_XIPEN — QSPIx XIP Mode Enable Control bit

1 = QSPIx XIP 模式启用

0 = QSPIx XIP 模式禁用

6.3.2.4. QSPI 32 位密钥寄存器（QSPI QSPI）

QSPPIKEYR 寄存器是一个可写的寄存器，在读取时总是返回 0's 。

地址：CCM_BASEADDR + 0x0000_001C

	31	30	29	28	27	26	25	24
读:	0							
写:	QSPI_KEY[31:24]							
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0	0	0	0	0	0	0	0
写:	QSPI_KEY[23:16]							
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0	0	0	0	0	0	0	0
写:	QSPI_KEY[15:8]							
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0	0	0	0	0	0	0	0
写:	QSPI_KEY[7:0]							
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 6-4：QSPI 32 位密钥寄存器（QSPPIKEYR）

6.3.2.5. QSPI GPIO 配置寄存器（QSPI）

QSPIGPIOCR 寄存器是一个可读可写的寄存器。

地址：CCM_BASEADDR + 0x0000_0020

	31	30	29	28	27	26	25	24
读:	QSPI2_SS_GPIOEN	QSPI1_SS_GPIOEN	QSPI0_SS_GPIOEN	0		QSPI2_SS_PUE	QSPI1_SS_PUE	QSPI0_SS_PUE
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0					qspi2_ss_obe	QSPI1_SS_OBE	QSPI0_SS_OBE
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0					QSPI2_SS_DO	QSPI1_SS_DO	QSPI0_SS_DO
写:								
复位:	0	0	0	0	0	1	1	1

	7	6	5	4	3	2	1	0
读:	0					QSPI2_SS_DI	QSPI1_SS_DI	QSPI0_SS_DI
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 6-5：QSPI GPIO 配置寄存器（QSPIGPIOCR）

QSPIx_SS_GPIOEN — QSPIx CS# Pin GPIO Mode Enable Control bit

1 = QSPIx 的 CS# Pin GPIO 模式已启用

0 = QSPIx 的 CS# Pin GPIO 已禁用

QSPIx_SS_PUE — QSPIx CS# Pin Pullup Enable Control bit in GPIO Input Mode.

提示：QSPI2_SS_PUE 对 LT7589 无效。

1 = QSPIx CS# Pin GPIOG 上拉电阻已启用

0 = QSPIx CS# Pin GPIO 上拉电阻被禁用

QSPIx_SS_OBE — QSPIx CS# Pin Output Enable Control bit in GPIO Mode

1 = QSPIx 的 CS# Pin 当作 GPIO 输出已启用

0 = QSPIx 的 CS# Pin 当作 GPIO 输出已禁用

QSPIx_SS_DO — QSPIx CS# Pin Output Data in GPIO Output Mode

1 = QSPIx CS# 将在 GPIO 输出模式下输出高电位

0 = QSPIx CS# 将在 GPIO 输出模式下输出低电位

QSPIx_SS_DI — QSPIx CS# Pin value in GPIO Mode

1 = 指示 QSPIx CS# Pin 在 GPIO 输入模式下是高电位

0 = 指示 QSPIx CS# Pin 在 GPIO 输入模式下是低电位

6.3.2.6. MCU 访问 RAM 优先级配置寄存器（MCURAMPRIOCR）

该寄存器是一个可读可写的寄存器。

地址：CCM_BASEADDR + 0x0000_0024

	31	30	29	28	27	26	25	24
读: 写:	RAMPRIOTEST[1:0]		CACHERAMPD	DISPLAYRAMPD	SYSRAM3PD	SYSRAM2PD	SYSRAM1PD	SYSRAM0PD
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0		CACHERAMISOEN	DISPLAYRAMISOEN	SYSRAM3ISOEN	SYSRAM2ISOEN	SYSRAM1ISOEN	SYSRAM0ISOEN
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0						RAM256KP	RAM512KP
写:								
复位:	0	0	0	0	0	0	1	1

= 写入没有影响，访问终止，没有传输错误异常。

图 6-6：MCU 访问 RAM 优先级配置寄存器（MCURAMPRIOCR）

RAMPRIOTEST[1:0] — MCURAMPRIOCR Write Access Sequence In

除非写入了正确的序列，否则不能更改 MCURAMPRIOCR 寄存器的可写位。正确的序列是：2'b01 → 2'b10 → 2'b11。在这两个位被这个序列写入后，两位的值== 2'b11，然后可以随意改变内存寄存器的可写位。当值等于 2'b11 时，写入 2'b00 可以清除这两个位。写入其他值没有效果，并返回 2'b11。

CACHERAMPD — The power supply of Cache RAM will be shut-down when writing 1'b to it.

DISPLAYRAMPD — Please refer to CACHERAMPD bit description.

SYSRAM3PD — Please refer to CACHERAMPD bit description.

SYSRAM2PD — Please refer to CACHERAMPD bit description.

SYSRAM1PD — Please refer to CACHERAMPD bit description.

SYSRAM0PD — Please refer to CACHERAMPD bit description.

CACHERAMISOEN — Cache RAM output value will be isolated when writing 1'b1 to it.

提示：在关闭缓存 RAM 的电源之前，缓存 RAM 的输出应被隔离，以避免产生漏电流。

DISPLAYRAMISOEN — Please refer to CACHERAMISOEN bit description.

SYSRAM3ISOEN — Please refer to CACHERAMISOEN bit description.

SYSRAM2ISOEN — Please refer to CACHERAMISOEN bit description.

SYSRAM1ISOEN — Please refer to CACHERAMISOEN bit description.

SYSRAM0ISOEN — Please refer to CACHERAMISOEN bit description.

RAM256KP — MCU Accesses System RAM256K Priority Configuration Register

这位确定当 MCU 和 Blender 同时访问系统 RAM256K 时，是否具有更高的优先级。

1 = 当 MCU 和 Blender 同时访问系统 RAM256K 时，MCU 具有更高的优先级。

0 = MCU 和 Blender 同时访问系统 RAM256K 时，MCU 的优先级较低。

RAM512KP — MCU Accesses Display RAM512K Priority Configuration Register

该位确定当 MCU 和 Blender 同时访问显示 RAM512K 时，MCU 是否具有更高的优先级。

1 = MCU 和 Blender 同时访问显示 RAM512K 时，MCU 具有更高的优先级。

0 = MCU 和 Blender 同时访问显示 RAM512K 时，MCU 的优先级较低。

6.3.2.7. EPORT2 功能配置寄存器（EPORT2FCR）

EPORT2FCR 寄存器是一个可读写寄存器。

地址：CCM_BASEADDR + 0x0000_0028

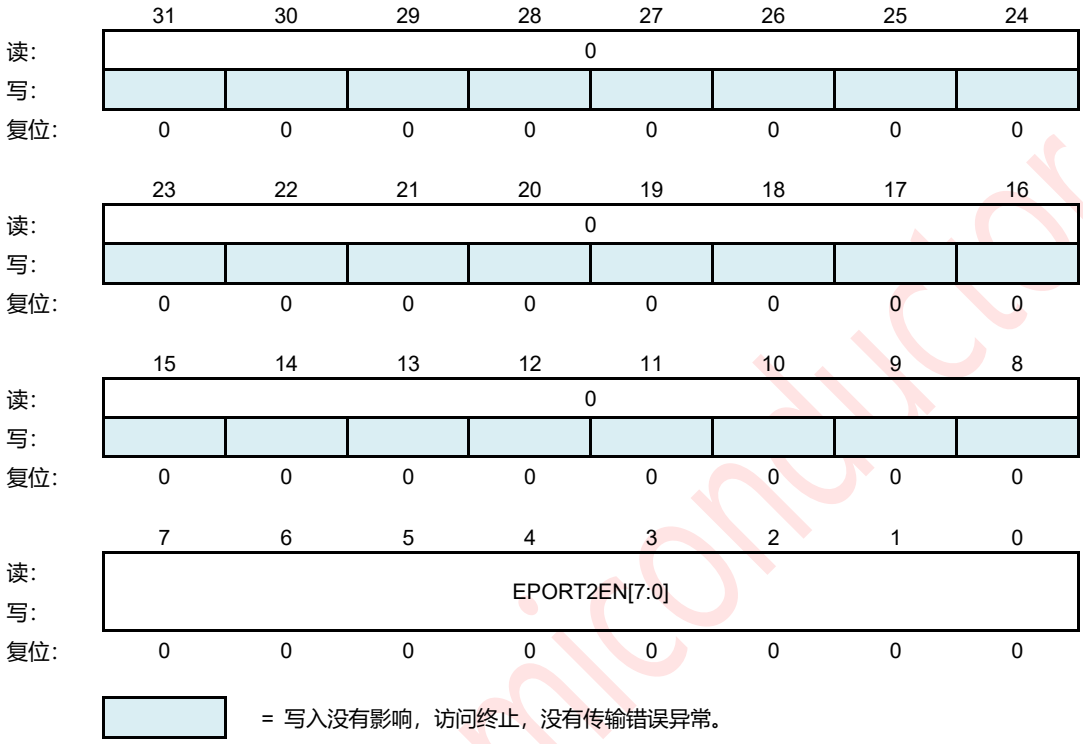


图 6-7：EPORT2 功能配置寄存器（EPORT2FCR）

EPORT2EN[7:0] — EPORT2 Function Enable control bit.

表 6-4：EPORT2 功能控制位

EPORT2 Enable Bit	1’b1	1’b0
EPORT2EN[7]	INT2[7]	CANRX
EPORT2EN[6]	INT2[6]	CANTX
EPORT2EN[5]	INT2[5]	RXD2
EPORT2EN[4]	INT2[4]	TXD2
EPORT2EN[3]	INT2[3]	RXD1
EPORT2EN[2]	INT2[2]	TXD1
EPORT2EN[1]	INT2[1]	RXD0
EPORT2EN[0]	INT2[0]	TXD0

7. 嵌入式中断控制器（EIC）

本章节介绍了 32 位 RISC 的嵌入式中断控制器。

7.1. 基本介绍

LT7589 具有一个中断控制器，它从多个中断源收集请求，并为 CPU 中断逻辑提供一个接口。

7.2. 功能特点

中断控制器的功能包括：

- 可配置的中断源，最多为 32 个
- 每个中断源都有 32 个独特的可编程优先级级别
- 基于优先级独立启用/禁用待起中断
- 每个中断源的一个固定的向量数
- 同时支持电平敏感中断和脉冲中断
- 支持 PendTrap 功能
- 支持软件复位

7.3. 内存映射和寄存器

本章节介绍了内存映射图（请参考下表)和寄存器。嵌入式中断控制器的基本地址为 0xE000_0000。

7.3.1. 内存映射

EIC 模块基地址（EIC_BASEADDR）定义为 32 位 RISC 内部参数定义。默认值为 0xE0000000。EIC 寄存器的实际地址是 EIC_BASEADDR 加上每个 EIC 寄存器的偏移量地址。核心内部模块占 64K 地址区。系统应避免将其他寄存器映射到从 EIC_BASEADDR 到 EIC_BASEADDR + 0x0000_FFFF 的区域。

表 7-1：中断控制器模块内存映射

偏移地址	Bits[31-24]	Bits[23-16]	Bits[15-8]	Bits[7-0]	访问权限(1)
0x0000_0000	Interrupt control status register (ICSR)				S/U
0x0000_0004	保留				S/U
0x0000_0008	保留				S/U
0x0000_000C	保留				S/U
0x0000_0010	Interrupt Enable Register (IER)				S/U
0x0000_0014	保留				S/U
0x0000_0018	Interrupt Pending Set Register (IPSR)				S/U
0x0000_001C	Interrupt Pending Clear Register (IPCR)				S/U

偏移地址	Bits[31-24]	Bits[23-16]	Bits[15-8]	Bits[7-0]	访问权限(1)
0x0000_0020 | 0x0000_003C	未使用(2)				-
Priority level select registers (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	System Priority level select register(SYSPLSR)				S/U
0x0000_0064 | 0x0000_007C	未使用(2)				-

提示：

(1) 在 32 位 RISC，该寄存器在任何情况下都可以被访问。

(2) 访问未实现的地址位置没有任何影响，并将导致一个周期终止传输错误。

7.3.2. 寄存器描述

本章节包含了对中断控制器模块寄存器的描述。嵌入式中断控制器的基本地址（EIC_BASEADDR）为 0xE000_0000。

7.3.2.1. 中断控制状态寄存器（ICSR）

32 位中断控制寄存器（ICSR）反映了中断控制器的状态

地址：EIC_BASEADDR + 0x0000_0000

	31	30	29	28	27	26	25	24
读:	SRST	0		SetPTrap	ClrPTrap	0		
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0	VEC[6:0]						
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 7-1：中断控制状态寄存器（ICSR）

SRST — Software Reset Bit

此只写位用于创建一个软件复位请求。设置这个位将在系统重新序列信号上产生一个脉冲。读取总是返回 0。

SetPTrap — Set PendTrap Bit

此读/写位用于创建一个等待处理的软件中断。该操作类似于执行“陷阱”指令。但是，待所有更高优先级的异常/中断退出之前，将不会进入待定的软件中断。当软件中断时，比特将自动清除。复位也会清除这一点。

读内容:

1 = 软件中断正在等待处理

0 = 软件中断没有等待处理

写数据:

1 = 将软件中断设置为等待处理

0 = 无影响

ClrPTrap — Clear PendTrap Bit

读/写 ClrDSI 位用于取消等待处理的软件中断（PendTrap）。

复位清除此位。

读内容:

1 = 软件中断正在等待处理

0 = 软件中断没有等待处理

写数据:

1 = 取消了等待处理的软件中断

0 = 无影响

VEC[6:0] — Interrupt Vector Number Field

只读 VEC[6:0]字段包含 7 位的中断向量数。复位清除 VEC[6:0]。

7.3.2.2. 中断启用寄存器（IER）

这个可读/写的 32 位中断启用寄存器（IER）分别启用任何当前等待处理的中断，这些中断是作为一个被分配给每个优先级级别的正常中断源。启用具有发送请求的中断源会导致该请求等待处理，如果还未完成，则会对 CPU 发送请求。

地址：EIC_BASEADDR + 0x0000_0010

	31	30	29	28	27	26	25	24
读: 写:	IE[31:24]							
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读: 写:	IE[23:16]							
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读: 写:	IE[15:8]							
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读: 写:	IE[7:0]							
复位:	0	0	0	0	0	0	0	0

图 7-2：中断启用寄存器（IER）

IE[31:0] — Interrupt Enable Field

此读/写 IE[31:0]字段允许来自相应优先级的来源中断请求作为中断请求。复位将清除 IE[31:0]。

1 = 允许中断请求

0 = 禁止中断请求

7.3.2.3. 中断等待集寄存器（IPSR）

地址：EIC_BASEADDR + 0x0000_0018

	31	30	29	28	27	26	25	24
读: 写:	SetPend[31:24]							
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读: 写:	SetPend[23:16]							
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读: 写:	SetPend[15:8]							
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读: 写:	SetPend[7:0]							
复位:	0	0	0	0	0	0	0	0

图 7-3：中断等待集寄存器（IPSR）

SetPend[31:0] — Interrupt Pending Set Field

此读/写 SetPend[31:0]字段设置为等待关联的中断，并指示关联的中断是否等待处理。复位可以清除 SetPend。

读内容：

1 = 相关的中断正在等待处理

0 = 关联的中断未等待处理

写数据：

1 = 将关联中断的状态更改为等待处理

0 = 无影响

7.3.2.4. 中断待定清除寄存器（IPCR）

地址：EIC_BASEADDR + 0x0000_001C

	31	30	29	28	27	26	25	24
读: 写:	ClrPend[31:24]							
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读: 写:	ClrPend [23:16]							
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读: 写:	ClrPend [15:8]							
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读: 写:	ClrPend [7:0]							
复位:	0	0	0	0	0	0	0	0

图 7-4：中断待定清除寄存器（IPCR）

ClrPend[31:0] — Interrupt Pending Clear Field

此读/写 ClrPend[31:0]字段清除，正在等待相关的中断，并指示相关的中断是否正在等待。复位清除 ClrPend[31:0]。

读内容：

1 = 相关的中断正在等待处理

0 = 关联的中断未等待处理

写数据：

1 = 将关联中断的状态更改为不等待处理

0 = 无影响

7.3.2.5. 优先级级别选择寄存器（PLSR）

此读/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 分配了一个优先级级别。

地址：从 EIC_BASEADDR + 0x0000_0040 到 EIC_BASEADDR + 0x0000_005C

	7	6	5	4	3	2	1	0
读:	PLSRx[7:6]		0	0	0	0	0	0
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 7-5：优先级选择寄存器（PLSR0-PLSR31）

PLSRx[7:6] — Priority Level Select Field

IRQ0~31 的默认优先级值为 0~31。该值越低，优先级就越高。这意味着 IRQ0 优先级> IRQ1 > ..... > IRQ31 为默认值。但是，用户可以设置 PLSRx[7:6]来调整中断的优先级。优先级级别的实际值是默认值加上 PLSRx[7:6] *64。例如，如果 PLSR1[7:6] = 2 时，IRQ1 的优先级值为 1 + 2*64 = 129，则 IRQ1 的优先级低于任何优先级值较低的 IRQ。

表 7-2：优先值调整

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

7.3.2.6. 系统优先级别选择寄存器（SYSPLSR）

地址：EIC_BASEADDR + 0x0000_0060

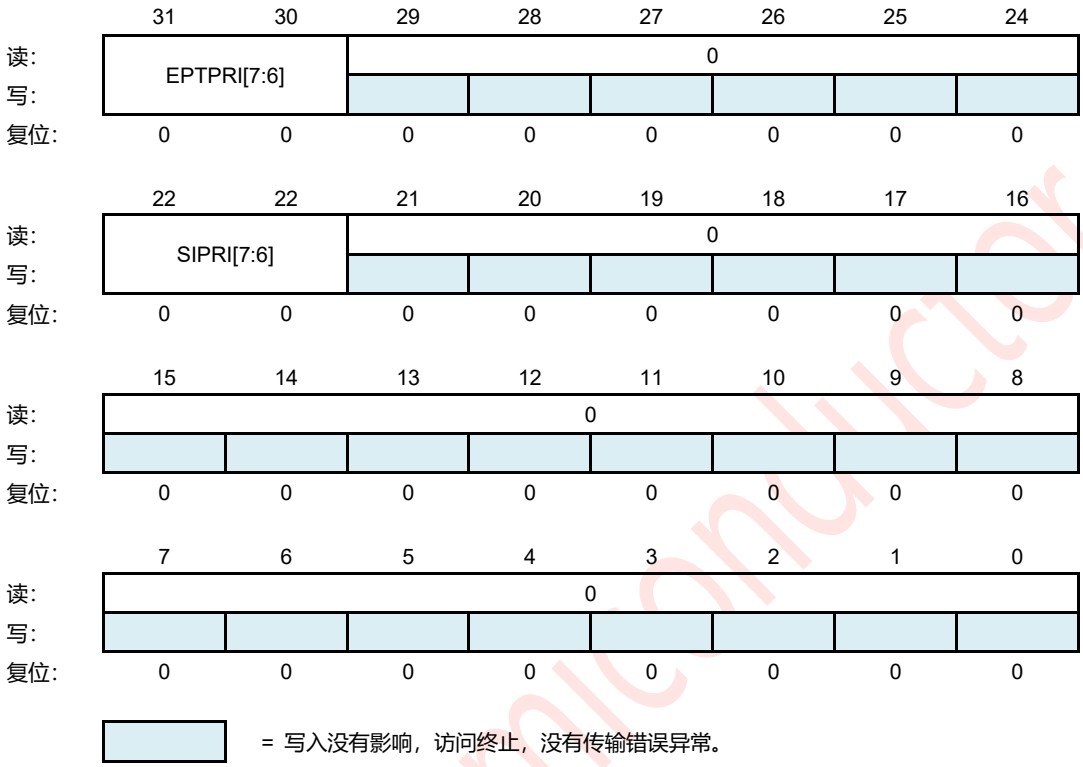


图 7-6：系统优先级级别选择寄存器（SYSPLSR）

EPTPRI[7:6] — EPT Priority Level Select Field

EPT 中断的默认优先级为-2。这意味着 EPT 中断优先级高于其他正常 irq 和系统软件中断（PendTrap）。值越低，优先级越高。但是，用户可以设置 EPTPRI[7:6]来调整 EPT 中断的优先级。优先级级别的实际值是默认值加上 PRI[7:6] *64。例如，如果 PRI[7:6] = 2，则 EPT 中断优先级值为 -2 + 2*64 = 126。

表 7-3：优先值调整

EPTPRI[7:6]	要增加的优先级值
00	0
01	64
10	128
11	192

SIPRI[7:6] — Software Interrupt Priority Level Select Field

软件中断（PendTrap）的默认优先级为-1。这意味着 EPT 中断优先级高于其他正常 irq。值越低，优先级越高。但是，用户可以设置 SIPRI[7:6]来调整软件中断的优先级。优先级级别的实际值是默认值加上 SIPRI[7:6] *64。例如，如果为 SIPRI[7:6] = 2，则软件中断优先级值为 -1 + 2*64 = 127。

表 7-4：优先值调整

SIPRI[7:6]	要增加的优先级值
00	0
01	64
10	128
11	192

7.4. 功能描述

EIC 同时支持电平敏感中断和脉冲中断。中断源号是从 1 到 32。

中断将出现以下原因之一：

- EIC 检测到中断信号是活动的，而相应的中断不是活动的
- EIC 检测到中断信号上的上升边缘

等待处理的中断仍保持等待处理，直到出现以下情况之一：

- 处理器进入 ISR 以进行中断。这将将中断的状态从等待处理状态更改为活动状态。然后
  - 对于电级敏感的中断，当处理器从 ISR 返回时，EIC 采样中断信号。如果信号起作用，中断状态变为待定，这可能导致处理器立即重新进入 ISR。否则，中断的状态将变为非活动状态。
  - 对于脉冲中断，EIC 将连续监控中断信号。如果中断信号是脉冲，中断的状态变为待起和激活。在这种情况下，当处理器从 ISR 返回时，中断的状态会变为等待处理，这可能会导致处理器立即重新进入 ISR。如果当处理器在 ISR 中时中断信号没有被脉冲，则当处理器从 ISR 返回时，中断的状态将变为非活动状态。
- 软件写入相应的中断，等待清除寄存器位。

7.4.1. 无冲突的中断处理

如果中断是脉冲，中断的状态将变为待起。没有冲突，中断会导致处理器立即进入 ISR。当处理器从 ISR 返回时，中断的状态将变为非活动状态。

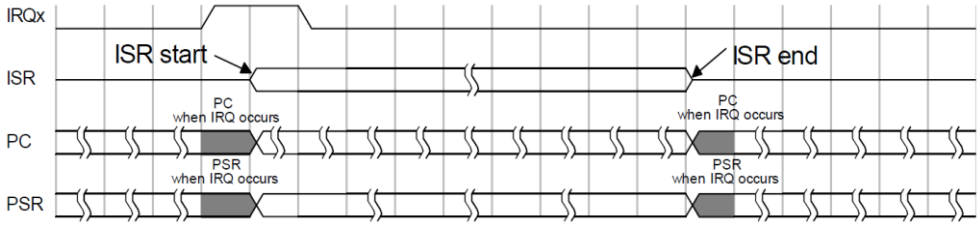


图 7-7：One Pulse Interrupt without Confliction

对于电平敏感的中断，如果信号起作用，则中断的状态将变为待决。没有冲突，中断会导致处理器立即进入 ISR。当处理器从 ISR 返回时，EIC 继续采样中断信号。如果该信号未被清除，则处理器将重新进入 ISR。否则，中断的状态将变为非活动状态。

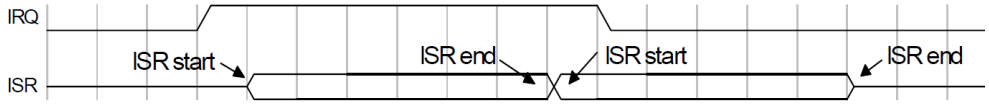


图 7-8：Level-sensitive Interrupt without Confliction

7.4.2. 有冲突的中断

当同时断两个中断信号时，中断仲裁器将判断哪一个具有更大的优先级。例如，如果 IRQx 的优先级大于 IRQy，处理器将输入 ISRx，IRQy 变为待起。当处理器从 ISRx 返回后，处理器将立即进入 ISRy。

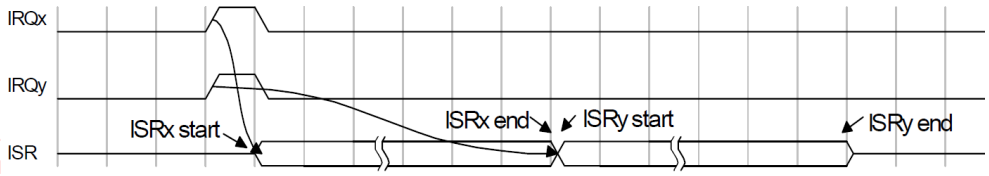


图 7-9：同时发生两次中断

如果在另一个中断处理过程中产生了一个中断信号，则会有两种情况：

1. 产生的中断优先级低于处理中断优先级。在这种情况下，产生的中断状态将等待处理，直到处理中断结束。
2. 产生的中断优先级高于处理中断优先级。在这种情况下，较高优先级的中断处理将嵌套在较低优先级的中断例程中。

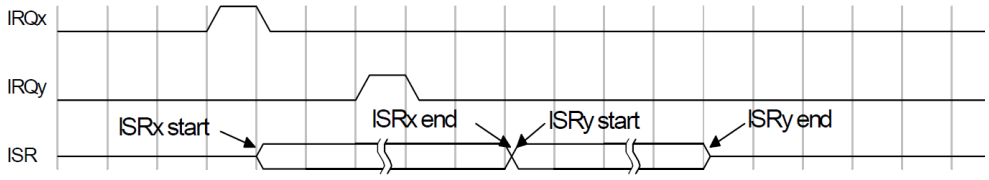


图 7-10：具有冲突的较低优先级中断

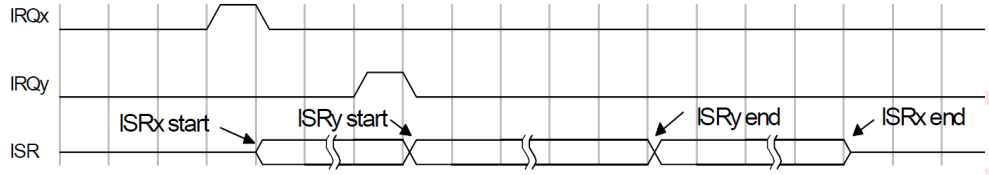


图 7-11：具有冲突的更高优先级中断

7.4.3. 待定陷阱功能

ICSR 中的 SetPTrap/ClrPTrap 位用于创建/取消一个“待定”的软件中断请求。一旦处理器从更高优先级的中断返回，“等待处理”的软件中断将被处理器接受。

通常，等待陷阱功能用于操作系统任务被动切换。

7.5. 中断来源

中断控制器为每个中断源分配一个数字，如下表所示：

表 7-5：中断来源分配

根源	模块	标志	来源描述	标志清除机制
0	ADC			
1	QSPI0	XRXOIS	XIP Receive FIFO Overflow Interrupt	
		RXFIS	Receive FIFO Full Interrupt	
		RXOIS	Receive FIFO Overflow Interrupt	
		RXUIS	Receive FIFO Underflow Interrupt	
		TXOIS	Transmit FIFO Overflow Interrupt	
		TXEIS	Transmit FIFO Empty Interrupt	
2	SCI0	TDRE	Transmit Data Register Empty Flag	
		TC	Transmission Complete	
		TXOF	Transmitter Buffer Overflow Flag	
		LBKDIF	LIN Break Detect Interrupt Flag	
		IDLE	Idle Line Flag	
		RXEDGIF	RXD0 Pin Active Edge Interrupt Flag	
		RDRF	Receive Data Register Full Flag	
		MA1F	Match 1 Flag	
		MA2F	Match 2 Flag	
		OR	Receiver Overrun Flag	
		NF	Noise Flag	
		FE	Framing Error Flag	
		PF	Parity Error Flag	
		RXUF	Receiver Buffer Underflow Flag	
3	COMP0	CPRIF		
		CPFIF		
4	COMP1	CPRIF		
		CPFIF		
5	DMAC	DONE[0]		Write DONE[0] = 1

根源	模块	标志	来源描述	标志清除机制
		DONE[1]		Write DONE[1] = 1
		DONE[14]		Write DONE[14] = 1
		DONE[15]		Write DONE[15] = 1
		DMA_ESR[GPE]	Group Priority Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[CPE]	Channel Priority Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[SAE]	Source Address Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[SOE]	Source Offset Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[DAE]	Destination Address Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[DOE]	Destination Offset Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[NCE]	Nbytes/Citer Configuration Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[SGE]	Scatter/Gather Configuration Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[SBE]	Source Bus Error	Write channel number to CERR[6:0] to clear error status
		DMA_ESR[DBE]	Destination Bus Error	Write channel number to CERR[6:0] to clear error status
6	WDT0	IF		
7	PWM0	PIFR[0]		
		PIFR[1]		
		PIFR[2]		
		PIFR[3]		
8	PWM1	PIFR[0]		
		PIFR[1]		
		PIFR[2]		
		PIFR[3]		
9	PIT0	PIF	PIT Flag	Writing a 1 to it or writing to PMR
10	PIT1	PIF	PIT Flag	Writing a 1 to it or writing to PMR
11	PIT2	PIF	PIT Flag	Writing a 1 to it or writing to PMR
12	PIT3	PIF	PIT Flag	Writing a 1 to it or writing to PMR
13	RTC	Day_intf	Day pulse flag	
		Hou_intf	Hour pulse flag	
		Min_intf	Minute pulse flag	
		Sec_intf	Second pulse flag	
		Ala_intf	Alarm flag	

根源	模块	标志	来源描述	标志清除机制
		1KHz_intf	1KHz pulse flag	
		32KHz_intf	32KHz pulse flag	
14	USB_DEV	USB_DEV Flag	USB_DEV Flag	
15	I2C	I2C Flag	I2C Flag	
16	EPORT2	EPF0	Edge port 2 flag 0	Write EPF0 = 1
		EPF1	Edge port 2 flag 1	Write EPF1 = 1
		EPF2	Edge port 2 flag 2	Write EPF2 = 1
		EPF3	Edge port 2 flag 3	Write EPF3 = 1
		EPF4	Edge port 2 flag 4	Write EPF4 = 1
		EPF5	Edge port 2 flag 5	Write EPF5 = 1
		EPF6	Edge port 2 flag 6	Write EPF6 = 1
		EPF7	Edge port 2 flag 7	Write EPF7 = 1
17	PVD	PVDO	PVD Flag	
18	CANBUS	CAN_IFRH[BUF63:BUF0]	CAN buffers 63–0 interrupts	This bit is cleared by writing it to ‘1’
19	CANBUS	CAN_ESR[BOFF_INT]	CANBus off interrupt	This bit is cleared by writing it to ‘1’
20	CANBUS	CAN_ESR[ERR_INT]	CAN error interrupt	This bit is cleared by writing it to ‘1’
21	CANBUS	CAN_ESR[TWRN_INT]	CAN transmit warning interrupt	This bit is cleared by writing it to ‘1’
22	CANBUS	CAN_ESR[RWRN_INT]	CAN receive warning interrupt	This bit is cleared by writing it to ‘1’
23	CANBUS	CAN_ESR[WKUP_INT]	Wake Up Interrupt	This bit is cleared by writing it to ‘1’
24	BLENDE	INT_END	Transfer Complete	
		INT_OQ	One quarter of the blending operation of the currently data source is completed	
		INT_HALF	Halt of the blending operation of the currently data source is completed	
25	RGBC	INTV	The completion flag of one frame transmission	
		INTN	Transmission completion flag for the specified number of frames	
26	QSPI1	XRXOIS	XIP Receive FIFO Overflow Interrupt	

根源	模块	标志	来源描述	标志清除机制
		RXFIS	Receive FIFO Full Interrupt	
		RXOIS	Receive FIFO Overflow Interrupt	
		RXUIS	Receive FIFO Underflow Interrupt	
		TXOIS	Transmit FIFO Overflow Interrupt	
		TXEIS	Transmit FIFO Empty Interrupt	
27	QSPI2	XRXOIS	XIP Receive FIFO Overflow Interrupt	
		RXFIS	Receive FIFO Full Interrupt	
		RXOIS	Receive FIFO Overflow Interrupt	
		RXUIS	Receive FIFO Underflow Interrupt	
		TXOIS	Transmit FIFO Overflow Interrupt	
		TXEIS	Transmit FIFO Empty Interrupt	
28	SCI1	TDRE	Transmit Data Register Empty Flag	
		TC	Transmission Complete	
		TXOF	Transmitter Buffer Overflow Flag	
		LBKDIF	LIN Break Detect Interrupt Flag	
		IDLE	Idle Line Flag	
		RXEDGIF	RXD1 Pin Active Edge Interrupt Flag	
		RDRF	Receive Data Register Full Flag	
		MA1F	Match 1 Flag	
		MA2F	Match 2 Flag	
		OR	Receiver Overrun Flag	
		NF	Noise Flag	
		FE	Framing Error Flag	
		PF	Parity Error Flag	
		RXUF	Receiver Buffer Underflow Flag	
29	SCI2	TDRE	Transmit Data Register Empty Flag	

根源	模块	标志	来源描述	标志清除机制
		TC	Transmission Complete	
		TXOF	Transmitter Buffer Overflow Flag	
		LBKDIF	LIN Break Detect Interrupt Flag	
		IDLE	Idle Line Flag	
		RXEDGIF	RXD2 Pin Active Edge	
			Interrupt Flag	
		RDRF	Receive Data Register Full Flag	
		MA1F	Match 1 Flag	
		MA2F	Match 2 Flag	
		OR	Receiver Overrun Flag	
		NF	Noise Flag	
		FE	Framing Error Flag	
		PF	Parity Error Flag	
		RXUF	Receiver Buffer Underflow Flag	
30	EPORT0	EPF0	Edge port 0 Flag 0	Write EPF0 = 1
		EPF1	Edge port 0 Flag 1	Write EPF1 = 1
		EPF2	Edge port 0 Flag 2	Write EPF2 = 1
		EPF3	Edge port 0 Flag 3	Write EPF3 = 1
		EPF4	Edge port 0 Flag 4	Write EPF4 = 1
		EPF5	Edge port 0 Flag 5	Write EPF5 = 1
		EPF6	Edge port 0 Flag 6	Write EPF6 = 1
		EPF7	Edge port 0 Flag 7	Write EPF7 = 1
31	EPORT1	EPF0	Edge port 1 Flag 0	Write EPF0 = 1
		EPF1	Edge port 1 Flag 1	Write EPF1 = 1
		EPF2	Edge port 1 Flag 2	Write EPF2 = 1
		EPF3	Edge port 1 Flag 3	Write EPF3 = 1
		EPF4	Edge port 1 Flag 4	Write EPF4 = 1
		EPF5	Edge port 1 Flag 5	Write EPF5 = 1
		EPF6	Edge port 1 Flag 6	Write EPF6 = 1
		EPF7	Edge port 1 Flag 7	Write EPF7 = 1

8. 嵌入式可编程定时器（EPT）

8.1. 基本介绍

嵌入式可编程定时器（EPT）是一个 24 位定时器，提供精确的中断在定期间隔和最小的处理器干预。计时器可以从重新加载值开始倒数，也可以是一个自由运行的倒数计数器。

EPT 中断可以触发一个异常（向量号 = 24）。

EPT 模块可以通过清除参数“EPT”到 0，以减少核心门数。

8.2. 内存映射和寄存器

8.2.1. 内存映射

EPT 基地址定义为 EIC_BASEADDR + 0x1000。默认的基于地址（EPT_BASEADDR）是 0xE000_1000。下表显示了 EPT 寄存器的偏移量地址。EPT 模块占用 4K 地址区。

表 8-1：可编程定时器模块内存配置

偏移地址	Bits[31:0]	访问权限
0x0000_0000	EPT Control and Status Register (EPTCSR)	S/U
0x0000_0004	EPT Reload Register (EPTRLD)	S/U
0x0000_0008	EPT Count Register (EPTCNT)	S/U
0x0000_000 C	保留	S/U

8.2.2. 寄存器描述

本章节包含了对 EPT 模块寄存器的描述。

8.2.2.1. EPT 控制状态寄存器（EPTCSR）

地址：E PT_BASEADDR + 0x0000_0000

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							CNTFLAG
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0					CLKSRC	INTEN	CNTEN
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 8-1：EPT 控制状态寄存器（EPTCSR）

CNTFLAG — Count Down to 0 flag

只读位表示已计数的计时器为 0。它将被 RESET#复位。

1 = 计时器倒数到 0。

0 = 计时器仍在倒计时。

CLKSRC — Count clock source select

读写位用于选择计数时钟源。它将被 RESET#复位。

1 = 核心时钟。

0 = 外部参考时钟。

INTEN — EPT Interrupt Request Enable

读/写位用于启用 EPT 的中断，当计时器倒数到 0，它将通过复位复位。

1 = 当计时器倒数到 0 时，会发生 = EPT 异常请求。

0 = 当计时器计数到 0 时，不会发生 = EPT 异常请求。

CNTEN — Counter Enable

读/写位用于启用 EPT 的计数器。它将通过 RESET#来进行复位。

1 = 计数器已启用

0 = 计数器已启用

8.2.2.2. EPT 重新加载寄存器（EPTRLD）

地址：EPT _BASEADDR + 0x0000_0004

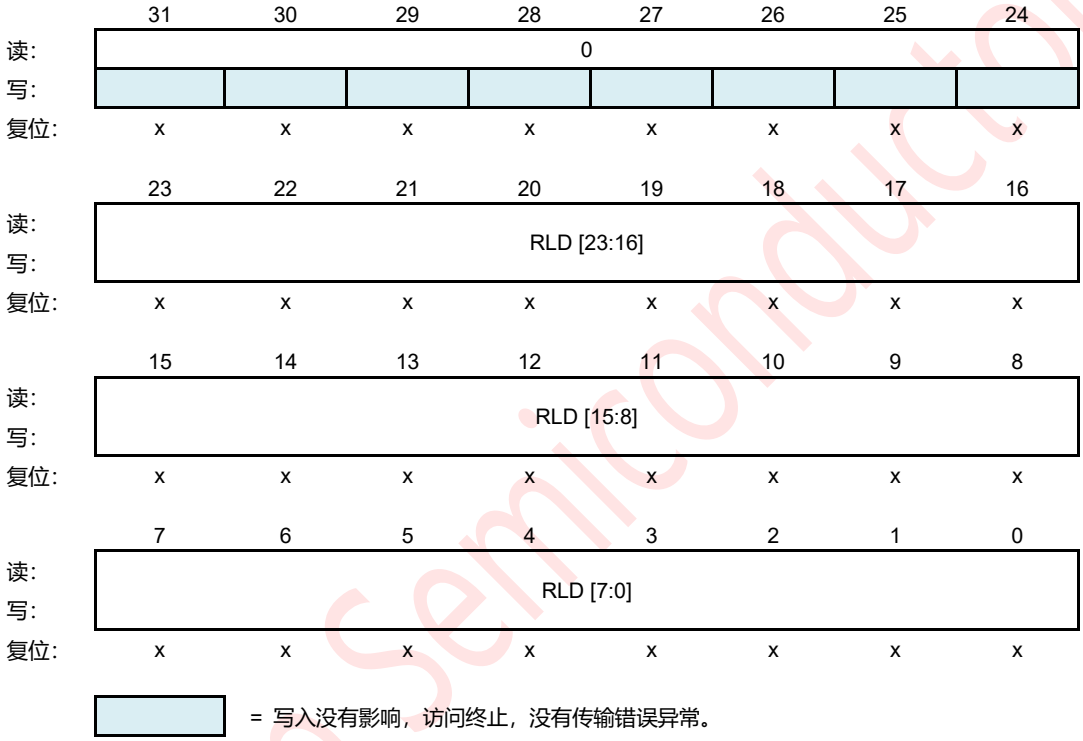


图 8-2：EPT 重新加载寄存器（EPTRLD）

RLD[23:0] — Reload Value

读/写 RLD 字段指定当计时器倒数到 0 时的重新加载值。该寄存器没有复位值。RLD 值可以是 0x000000001~0x00 FFFFFF 范围内的任何值。值 0 没有任何影响。要生成具有 N 个时钟周期的周期定时器，请将 RLD 设置为 N-1。

8.2.2.3. EPT 计数寄存器（EPTCNT）

地址：EPT _BASEADDR + 0x0000_0008

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	x	x	x	x	x	x	x	x

	23	22	21	20	19	18	17	16
读:	CNT[23:16]							
写:								
复位:	x	x	x	x	x	x	x	x

	15	14	13	12	11	10	9	8
读:	CNT[15:8]							
写:								
复位:	x	x	x	x	x	x	x	x

	7	6	5	4	3	2	1	0
读:	CNT[7:0]							
写:								
复位:	x	x	x	x	x	x	x	x

= 写入没有影响，访问终止，没有传输错误异常。

图 8-3：EPT 计数器寄存器（EPTCNT）

CNT[23:0] — EPT Counter Value

只读寄存器表示 EPT 计时器的当前计数值。该寄存器没有复位值。

读取计数器将返回 EPT 计数器的当前值。写入此寄存器的任何值将将计数器值清除为 0，并将 CNTFLAG 清除为 0。

8.3. 功能描述

启用后，EPT 计数从 RLD 设置的值降为零，并在下一个时钟周期中重新加载 RLD 中的值，然后通过随后的时钟周期降低计数，将零写入 RLD 在下一次封装时禁用计数器。当 EPT 计数 s 为 0 时，CLFAG 位将被设置为 1，然后如果启用了 INTEN，则 EPT 将触发 EPT 中断。

读取 CSR 将 CFLAG 位清除为 0。将任何值写入 CNT 也会将 CFLAG 位清除为 0。

8.3.1. 计数时序

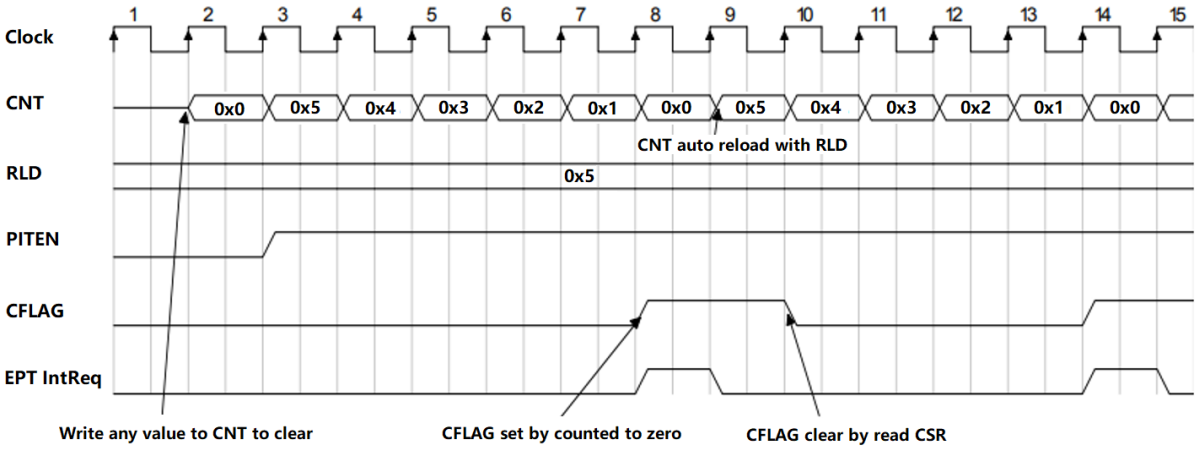


图 8-4：EPT 计数时序

9. 时钟和电源控制模块（CLKPWRM）

9.1. 基本概述

时钟模块包含：

- PLL：内部的 VCO PLL
- FXOSC：外部快速速度晶体振荡器（12MHz）
- SIRC：内部低速 128Khz 振荡器
- SXOSC：外部低速晶体振荡器（32,768Hz）
- 状态和控制寄存器
- 时钟和电源控制逻辑

9.2. 功能特点

该时钟模块的功能包括：

- 两个系统时钟源
  - 内部 PLL 时钟
  - 外部快速速度晶体振荡器（FXOSC）
- IPS、系统和 ADC 时钟的独立时钟分频器
- 支持低功耗模式
- 模块可以通过设置 MSCR 来单独停止

9.3. 时钟结构

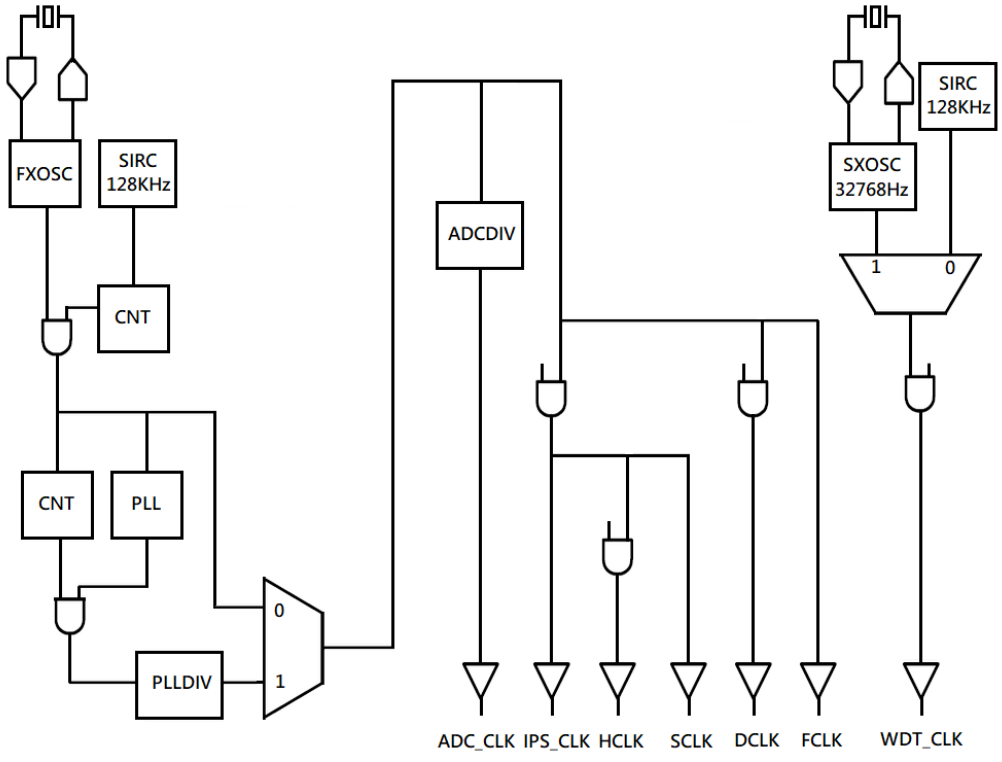


图 9-1：时钟结构

9.4. 时钟源选择

系统时钟源可以是内部 PLL 时钟或外部高速晶体振荡器(FXOSC)。时钟源选择是基于 SYNCR 寄存器的 PLLEN 位。如果设置了 PLLEN 位，则内部 PLL 是系统时钟源，否则系统时钟源是外部高速晶体振荡器（FXOSC）。

9.4.1. 低功耗选项

9.4.1.1. 等待和多兹模式

在等待和休眠模式下，系统到外设的时钟被启用，到 CPU、ROM、SRAM 的时钟被停止。每个模块都可以在模块级或通过设置 MSCR 来在本地禁用模块时钟。

9.4.1.2. 停止模式

在停止模式下，将禁用所有系统时钟。
提示：在停止模式下，不要编写程序或删除 EFLASH。

9.5. 内存映射和寄存器

时钟编程模块由以下寄存器组成：

- 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 外部时钟源启用控制寄存器（ECSECR）
- 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（OBTCTLR）
- OSC Bist 测试计数器寄存器（OBTCNTR）
- OSC Bist 测试结果寄存器（OBTRR）

本章节介绍了时钟模块的内存映射和寄存器。时钟模块的基本地址为 0x4003_0000。下表 9-1 显示了时钟寄存器的偏移量地址。

9.5.1. 内存映射

表 9-1：时钟内存映射

地址	Bits[31:0]	访问权限 [1]
0x0000	Synthesizer Control Register (SYNCR)	S
0x0004	Low Speed Oscillator Control Register (LOSCCR)	S
0x0008	PLL Configuration and Status Register(PLLCSR)	S
0x000C	Module Stop Control Register (MSCR)	S
0x0010	EPT External Clock Source Enable Control Register(ECSECR)	S
0x0014	OSC Bist Test Configuration Register1(OBTCR1)	S
0x0018	OSC Bist Test Configuration Register2(OBTCR2)	S
0x001C	OSC Bist Test Control Register(OBTCTLR)	S
0x0020	OSC BIST Test Counter Register(OBTCNTR)	S
0x0024	OSC BIST Test Result Register(OBTRR)	S

提示：

S = 仅主管访问。在用户模式下，访问主管只有地址位置没有影响，并导致一个周期终止传输错误。

9.5.2. 寄存器说明

本章节提供了对时钟模块寄存器的描述。

9.5.2.1. 合成器控制寄存器（SYNCR）

合成器控制寄存器（SYNCR）始终为读/写。

地址：CLOCK_BASEADDR + 0x0000_0000

	31	30	29	28	27	26	25	24
读:	SYNCTEST[1:0]		SYSRAM3 LPEN	SYSRAM2 LPEN	SYSRAM1 LPEN	SYSRAM0 LPEN	PLLOCKM	ENLOWPOWER
写:								
复位:	0	0	0	0	0	0	0	1

	23	22	21	20	19	18	17	16
读:	PLLDIV[5:0]						0	0
写:								
复位:	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8
读:	ADCDIV[3:0]				LOSCEN	PLLSRCEN	PLLEN	SLEEP
写:								
复位:	0	0	1	0	1	1	0	0

	7	6	5	4	3	2	1	0
读:	DISPLAYRAMLPEN	CLKOUTSEL	STBYMD[1:0]		CACHERAMLPEN	ADCEN	0	LOSCLPEN
写:								
复位:	1	0	1	1	1	1	0	1

= 写入没有影响，访问终止，没有传输错误异常。

图 9-2：合成器控制寄存器（SYNCR）

SYNCTEST[1:0] — SYNCR Write Access Sequence In

不能更改 SYNCR 寄存器的可写位，除非写入正确的序列。正确的序列是：2'b01 → 2'b10 → 2'b11。在这两个位被这个序列写入之后，两位的值== 2'b11，那么 SYNCR 寄存器的可写位可以随意更改。当值等于 2'b11 时，写入 2'b00 可以清除这两个位。写入其他值没有效果，并返回 2'b11。→→

SYSRAM3LPEN — 系统 RAM3（地址范围从 0x0083_0000 到 0x83_FFFF）芯片进入低功耗模式时的低功耗启用位。

系统 RAM3 的低功耗模式是通过关闭 RAM 的电源来实现的，RAM 的内容将丢失。

1 = 当芯片进入低功耗模式时，将启用 SYSRAM3 的低功耗模式。

0 = SYSRAM3 在芯片进入低功耗模式时失效。

SYSRAM2LPEN — 系统 RAM2（地址范围从 0x0082_0000 到 0x82_FFFF）芯片进入低功耗模式时的低功耗启用位。

系统 RAM2 的低功耗模式是通过关闭 RAM 的电源来实现的，RAM 的内容将丢失。

1 = 当芯片进入低功率模式时，SYSRAM2 将启用低功率模式。

0 = 当芯片进入低功耗模式时，将禁用 SYSRAM2 的低功耗模式。

SYSRAM1LPEN — 系统 RAM11（地址范围从 0x0081_0000 到 0x81_FFFF）芯片进入低功耗模式时的低功耗启用位。

系统 RAM1 的低功耗模式是通过关闭 RAM 的电源来实现的，RAM 的内容将丢失。

1 = 当芯片进入低功率模式时，SYSRAM1 将启用低功率模式。

0 = 当芯片进入低功耗模式时，将禁用 SYSRAM1 的低功耗模式。

SYSRAM0LPEN — 系统 RAM3（地址范围从 0x0080_0000 到 0x80_FFFF）芯片进入低功耗模式时的低功耗启用位。

系统 RAM0 的低功耗模式是通过关闭 RAM 的电源来实现的，RAM 的内容将丢失。

1 = 当芯片进入低功率模式时，将启用 SYSRAM0 的低功率模式。

0 = SYSRAM0 在芯片进入低功耗模式时失效。

PLLOCKM — PLL Lock detection flag which is generated by PLL macro.

1 = PLL Lock 是在设置 PLLEN 时由 PLL 宏生成的。

0 = 当设置了 PLLEN 时，PLL 宏不会生成 PLL 锁。

ENLOWPOWER — Enable Enter Low power mode status Bit

在系统进入待机模式之前，请确保设置了此位，否则将无法成功进入低功耗模式。此位在从低功耗模式恢复后设置，并在进入低功耗模式时由硬件清除

1 = 启用输入低功耗模式状态位

0 = 低功耗模式

PLLDIV[5:0] — PLL Clock Divider

此字段设置 PLL 时钟的除法值。默认值为 6'b000001（除以 2）。参考下表对于其他的除法值。

表 9-2：PLL 时钟分频器

PLLDIV[5:0]	Divider Value
000000	Divide-by-2
000001	Divide-by-2
000010	Divide-by-4
000011	Divide-by-6
000100	Divide-by-8
000101	Divide-by-10
.....	.....
.....	.....
111111	Divide-by-126

提示：系统时钟的频率不应大于 300MHz。

ADCDIV[3:0] — ADC Clock Divider

此字段设置 ADC 时钟的除法值。默认值为 4'b0000（除以 1）。参考下表。

表 9-3：ADC 时钟分频器

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

LOSCEN — Internal Low Speed 128KHz Oscillator Enable Bit

1 = 内部低速 128KHz 振荡器已启用
0 = 内部低速 128KHz 振荡器被禁用

PLLSRCEN — This bit determines whether system clock will be stopped or not when PLLEN is changed from 0 to 1.

1 = 系统时钟将不会停止，系统时钟源来自 FXOSC，直到 PLL 被锁定
0 = 系统时钟将停止，直到 PLL 被锁定

PLLEN — PLL Enabled control bit.

1 = PLL 已启用，系统时钟源为 PLL
0 = PLL 被禁用，且系统时钟源为 FXOSC

SLEEP — Chip Sleep Mode Control Bit

设置休眠位后，芯片将进入 STBYMD 指示的待机模式[1:0]。

该操作与“停止”指令相同。

提示：睡眠仅在 STBYMD[1] = 1'b1 时有效。这与停止指令的情况不同，因为停止指令总是有效的。

DISPLAYRAMLPEN — 当芯片进入低功耗模式时，显示 RAM（地址范围从 0x0084_0000 到 0x8B_FFFF）低功耗启用位。

显示 RAM 的低功耗模式是通过关闭 RAM 的电源来实现的，RAM 的内容将丢失。

1 = 当芯片进入低功耗模式时，将启用显示闸板的低功耗模式。

0 = 当芯片进入低功耗模式时，禁用显示器的低功耗模式。

CLKOUTSEL — Clock Out Select Bit

表 9-4：CLKOUTSEL 模式

CLKOUTSEL	CLKOUT
0	系统时钟
1	128KHz 时钟

STBYMD[1:0] — Sleep Operation Control Bits

STBYMD[1:0]在睡眠模式下控制时钟源、系统时钟操作和 LDO 状态，如下表所示 。

表 9-5：睡眠模块中的睡眠操作控制位

STBYMD	ADC Clock	System Clocks	Clock Source	LDO
00	Enable	Disabled	Enable	Normal
01	Disabled	Disabled	Enable	Normal
10	Disabled	Disabled	Disabled	Normal
11	Disabled	Disabled	Disabled	Standby

CACHERAMLPEN — CACHE RAM Low Power Enable Bit.

当芯片进入低功耗模式时，此位启用。CACHE 快取 RAM 的低功耗模式是通过关闭 RAM 的电源来实现的，RAM 的内容将丢失。

1 = 当芯片进入低功耗模式时，将启用快取缓存 RAM 的低功耗模式。

0 = 当芯片进入低功耗模式时，将禁用快取缓存 RAM 的低功耗模式。

ADCEN — Analog-to-digital converter Clock Enable Bit

1 = ADC 时钟已启用

0 = ADC 时钟被禁用

LOSCLPEN — Internal Low Speed 128KHz Oscillator Low Power Enable

如果设置了故障处理，内部低速 128KHz 振荡器将在待机模式下停止。

1 = 内部低功率模式 128KHz 振荡器启用

0 = 内部低速 128KHz 振荡器的低功率模式被禁用

9.5.2.2. 低速振荡器控制和状态寄存器（LOSCCSR）

地址：CLOCK_BASEADDR + 0x0000_0004

	31	30	29	28	27	26	25	24
读:	LOSCCSTEST[1:0]		0	SIRCST[4:0]				
写:								
复位:	0	0	0	1	1	1	1	1

	23	22	21	20	19	18	17	16
读:	SXOSCST[15:8]							
写:								
复位:	0	1	0	1	0	0	0	0

	15	14	13	12	11	10	9	8
读:	SXOSCST[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	SIRCRDY	SXOSCRDY	WDTCLKCHGDONE	0			WDTCLKSEL	SXOSCEN
写:								
复位:	1	1	1	0	0	0	0	1

= 写入没有影响，访问终止，没有传输错误异常。

图 9-3：低速振荡器控制和状态寄存器（LOSCCSR）

LOSCCSTEST[1:0] — LOSCCSR Write Access Sequence In

不能更改 IOSCCSR 寄存器的可写位，除非写入了正确的序列。正确的序列是：2'b01 → 2'b10 → 2'b11。在通过序列写入这两位之后，这两位的值 == 2'b11，然后可以随意更改 LOSCCSR 寄存器的可写位。当值等于 2'b11 时，写入 2'b00 可以清除这两个位。写入其他值 s 没有效果，并返回 2'b11。

SIRCST[4:0] — Internal Low Speed Oscillator Stable Time Value

内部低速振荡器（SIRC）将等待 32KHz（SIRC 除以 4）振荡器的 SIRCST[4:0]周期，然后在打开后准备好进行输出时钟。

SXOSCST[15:0] — External Low Speed Oscillator Stable Time Value

外部低速振荡器（SXOSC）将等待 128KHz（SIRC 源）振荡器的[15:0]周期，然后在接通后准备好进行输出时钟。

SIRCRDY — Internal low speed oscillator (SIRC) ready flag.

1 = 内置低速振荡器（SIRC）已准备就绪

0 = 内部低速振荡器（SIRC）未准备好

SXOSCRDY — External low speed oscillator (SXOSC).

1 = 外部低速振荡器（SXOSC）已准备就绪

0 = 外部低速振荡器（SXOSC）未准备就绪

WDTCLKCHGDONE — WDT clock switch done flag.

当将 WDT 时钟源从 SXOSC 更改为 SIRC 时，应在关闭 SXOSC 之前清除 WDTCLKSEL 位。

1 = WDT 时钟开关完成，WDT 可以正常工作

0 = WDT 时钟正在切换，WDT 不能正常工作

WDTCLKSEL — WDT clock selection control bit.

当将 WDT 时钟源从 SXOSC 更改为 SIRC 时，应该在关闭 SXOSC 之前清除 WDT 位。

1 = WDT 时钟源为 SXOSC（32.768KHz）

0 = WDT 时钟源是 SIRC 128Khz

SXOSCEN — SXOSC Enable Setting

1 = 为 WDT 应用程序启用了 SXOSC。

0 = SXOSC 晶体禁用 WDT 应用。另外，现在应该清除，否则 WDT 时钟将丢失。

9.5.2.3. PLL 配置和状态寄存器（PLLCSR）

地址：CLOCK_BASEADDR + 0x0000_0008

	31	30	29	28	27	26	25	24
读:	PLLCSRTEST[1:0]		0					PLLOCK
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0				PLLN[3:0]			
写:								
复位:	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8
读:	PLLOD[1:0]		PLLM[13:8]					
写:								
复位:	1	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	PLLM[7:0]							
写:								
复位:	0	0	1	1	0	0	1	0

= 写入没有影响，访问终止，没有传输错误异常。

图 9-4：PLL 配置和状态寄存器（PLLCSR）

$$\text{PLL Output Frequency} = XIN \times \frac{M}{N} \times \frac{1}{NO}$$

PLLCSRTEST[1:0] — PLLCSR Write Access Sequence In

不能更改 PLLCSR 寄存器的可写位，除非写入了正确的序列。正确的序列是：2'b01 → 2'b10 → 2'b11。在用这个序列写入这两位后，这两位的值 == 2'b11，然后可以随意改变 PLLCSR 寄存器的可写位。当值等于 2'b11 时，写入 2'b00 可以清除这两个位。写入其他值 s 没有效果，并返回 2'b11。

提示：使用时请保持以下条件：

1. 1MHz <= XIN/N <= 50MHz
2. 200MHz <= XIN*M/N <= 400MHz
3. M≥4
4. N≥1

PLLOCK — PLL Lock Flag

一旦启用了 PLL，将在等待在 PLL[PLLST]中设置的循环数之后设置 PLLOCK。

1 = PLL 已被锁定

0 = PLL 未被锁定

PLLN[3:0] — Input 4-bits divider control bits

此字段设置 PLL 时钟的输入除法值。默认值为 4'b0000。参考下表对于可用的除法器值。

表 9-6：PLL 输入分频数值

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

PLLOD[1:0] — Output divider control bits

此字段设置 PLL VCO 时钟的输出除法值。默认值为 2'b00。参考下表对于可用的除法器值。

表 9-7：PLL VCO 输出时钟的除频数值

PLLOD[1:0]	NO
00	Divide-by-1
01	Divide-by-2
10	Divide-by-4
11	Divide-by-8

PLLM[13:0] — Feedback 14-bits divider control bits

该字段设置 PLL 反馈除法值。默认值为 14'b0。参考下表对于可用的除法器值。

表 9-8：PLL 反馈的除频器数值

PLLM[13:0]	M
14’b00_0000_0000_0000	0
14’b00_0000_0000_0001	1
14’b00_0000_0000_0010	2
14’b00_0000_0000_0011	3
14’b00_0000_0000_0100	4
14’b00_0000_0000_0101	5
…..	…..
…..	…..
14’b11_1111_1111_1111	16383

9.5.2.4. 模块停止控制寄存器（MSCR）

模块停止控制寄存器（MSCR）始终为读/写状态。

地址：CLOCK_BASEADDR + 0x0000_000C

	31	30	29	28	27	26	25	24
读:	MSCRTEST[1:0]		MS[29:24]					
写:								
复位:	0	0	1	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	MS[23:16]							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	MS[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	MS[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 9-5：模块停止控制寄存器（MSCR）

MSCRTEST[1:0] — MSCR Write Access Sequence In

不能更改 MSCR 寄存器的可写位，除非写入了正确的序列。正确的序列是：2‘b01 2’b10 2‘b11。在用这个序列写入这两个位后，这两个位的值 == 2‘b11，然后可以随意更改 MSCR 寄存器的可写位。当值等于 2‘b11 时，写入 2’b00 可以清除这两个位。写入其他值 s 没有效果，并返回 2‘b11。

MS[29:0] — Module Stop Bits

MS[25:0]位禁用顶层模块的时钟。（参考下表 MS[29:0]位对应模块）。

1 = 模块时钟被禁用

0 = 模块时钟已启用

表 9-9：MS[29:0] 相应的模块

MS Bit	Corresponding Module
0	Blender
1	COMP0
2	COMP1
3	ADC
4	PIT0
5	PIT1
6	PIT2
7	PIT3
8	RTC
9	DMA
10	PWM0
11	PWM1
12	EPORT0
13	EPORT1
14	XBAR
15	OPTION
16	RESET
17	WDT
18	SCI0
19	CCM
20	I2C
21	SCI1
22	SCI2
23	CAN
24	EPORT2
25	QSPI0
26	QSPI1
27	QSPI2
28	RGB
29	USBC

9.5.2.5. EPT 外部时钟源启用控制寄存器（ECSECR）

地址：CLOCK_BASEADDR + 0x0000_0010

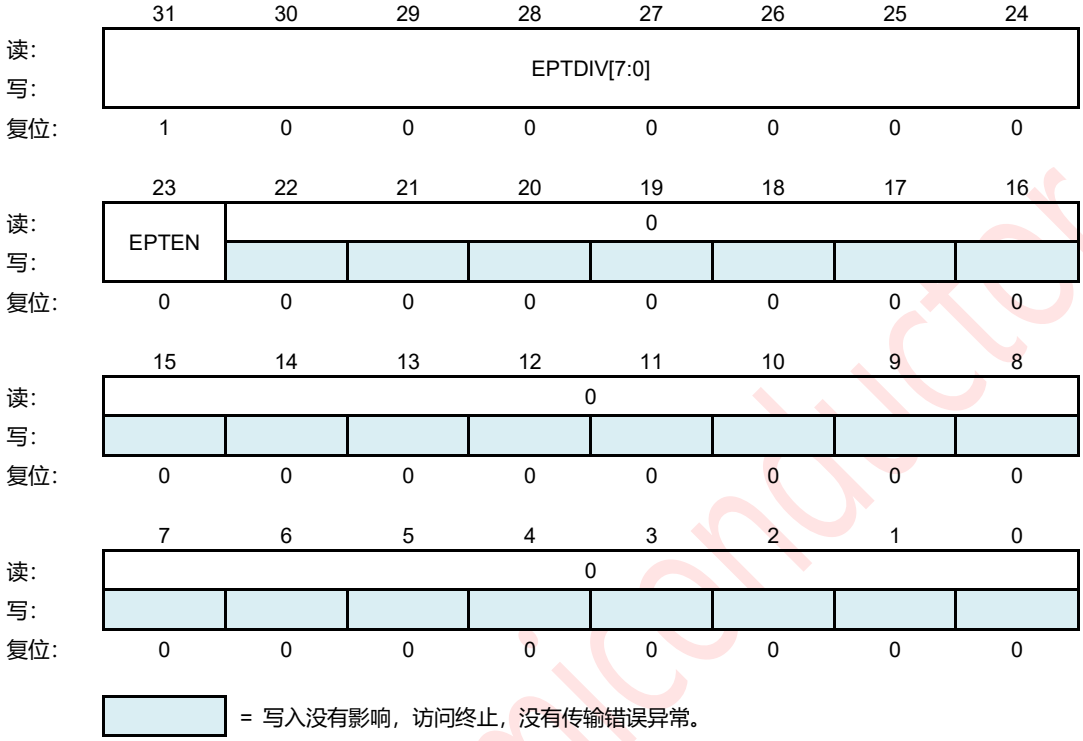


图 9-6：EPT 外部时钟源启用控制寄存器（ECSECR）

EPTDIV[7:0] — EPT Clock Divider

此字段设置 EPT 时钟的除法值。默认值为 8'h80。参考下表对于其他可用的除法器值。

表 9-10：EPT 时钟分频器

EPTDIV[7:0]	Divider Value
00000000	Divide-by-1
00000001	Divide-by-2
00000010	Divide-by-3
00000011	Divide-by-4
00000100	Divide-by-5
00000101	Divide-by-6
.....	.....
.....	.....
11111110	Divide-by-255
11111111	Divide-by-256

EPTEN — EPT Clock Enable Bit

如果设置了 EPTEN 位，EPT 时钟将与系统时钟分开。

1 = EPT 时钟已启用

0 = EPT 时钟被禁用

9.5.2.6. OSC Bist 测试配置寄存器 1（OBTCR1）

地址：CLOCK_BASEADDR + 0x0000_0014

	31	30	29	28	27	26	25	24
读: 写:	BIST_HOLD_TARGET[15:8]							
复位:	0	0	0	0	0	0	0	0

	20	19	21	20	19	18	17	16
读: 写:	BIST_HOLD_TARGET[7:0]							
复位:	0	0	0	0	0	0	0	0

	12	11	13	12	11	10	9	8
读: 写:	BIST_TARGET[15:8]							
复位:	0	0	0	0	0	0	0	0

	4	3	5	4	3	2	1	0
读: 写:	BIST_TARGET[7:0]							
复位:	0	0	0	0	0	0	0	0

图 9-7：OSC Bist 测试配置寄存器 1（OBTCR1）

BIST_HOLD_TARGET[15:0] — Bist clk hold count target value

这些位用来设置 clk 等待量，在将微调值更改后的测试为稳定后。

BIST_TARGET[15:0] — Bist clk count target value.

9.5.2.7. OSC Bist 测试配置寄存器 2（OBTCR2）

地址：CLOCK_BASEADDR + 0x0000_0018

	31	30	29	28	27	26	25	24
读: 写:	BIST_TEST_TARGET[15:8]							
复位:	0	0	0	0	0	0	0	0

	19	22	21	20	19	18	17	16
读: 写:	BIST_TEST_TARGET[7:0]							
复位:	0	0	0	0	0	0	0	0

	11	14	13	12	11	10	9	8
读: 写:	BIST_TEST_CTRIM_MARGIN[15:8]							
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读: 写:	BIST_TEST_CTRIM_MARGIN[7:0]							
复位:	0	0	0	0	0	0	0	0

图 9-8：OSC Bist 测试配置寄存器 2（OBTCR2）

BIST_TEST_TARGET[15:0] — Reserved. Not available to users.

BIST_TEST_CTRIM_MARGIN[15:0] — Reserved. Not available to users.

9.5.2.8. OSC Bist 测试控制寄存器（OBTCTLR）

地址：CLOCK_BASEADDR + 0x0000_001C

	31	30	29	28	27	26	25	24
读:	BIST_TEST_FTRIM_MARGIN[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	20	22	21	20	19	18	17	16
读:	BIST_TEST_FTRIM_MARGIN[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	BIST_START	0	BIST_RESETN	BIST_MODE	0	Reserved	BIST_IRC_SEL	BIST_EN
写:								
复位:	0	0	0	1	0	0	1	0

	7	6	5	4	3	2	1	0
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 9-9：OSC Bist 测试控制寄存器（OBTCTLR）

BIST_TEST_FTRIM_MARGIN[15:0] — Reserved. Not available to users.

BIST_START — Bist Start

1 = 启动

0 = 停止

BIST_RESETN — Bist Reset Negate

1 = 否定 Bist 复位

0 = 插入 Bist 复位

BIST_MODE — Bist Mode

1 = 跟踪模式（用于测量 PLL 或 128K 时钟的频率）

0 = 微调模式

提示：该芯片上仅实现了跟踪模式。

BIST_IRC_SEL — PLL or 128KHz clock selection

1 = 选择 PLL 时钟

0 = 选择 128K 时钟

BIST_EN — Reference Clock Enable

1 = 启用

0 = 禁用

9.5.2.9. OSC Bist 测试计数器寄存器（OBTCNTR）

地址：CLOCK_BASEADDR + 0x0000_0020

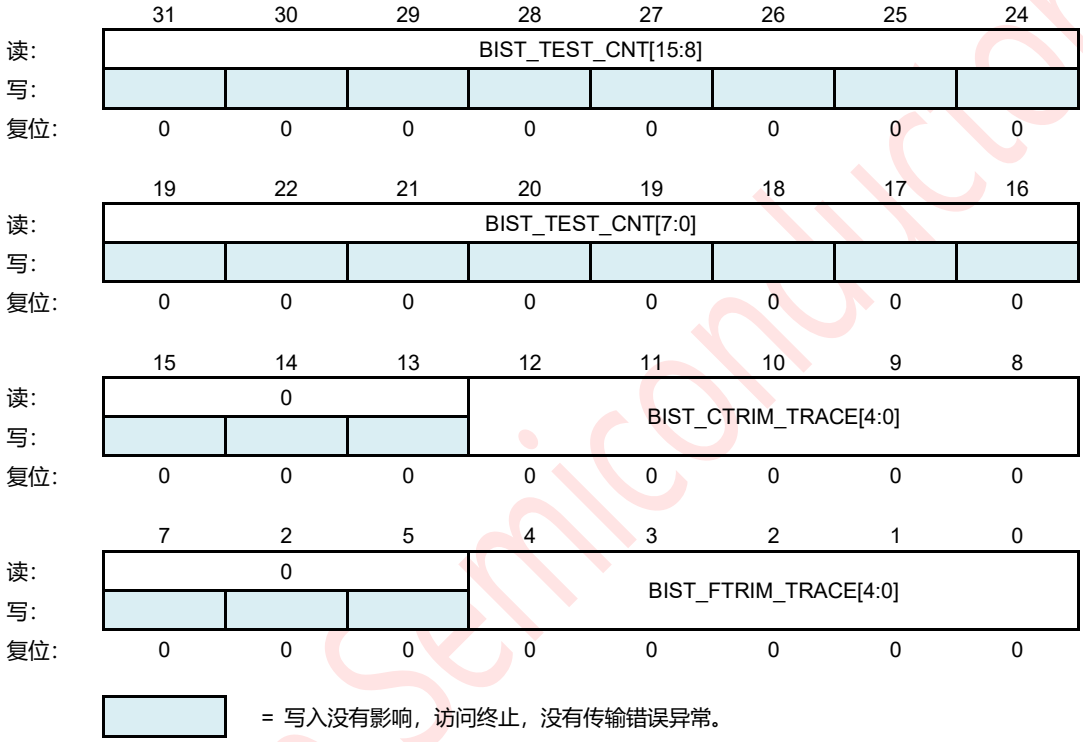


图 9-10：OSC Bist 测试计数器寄存器（OBTCNTR）

BIST_TEST_CNT[15:0] — Bist Test Counter Value

BIST_CTRIM_TRACE[4:0] — Bist Ctrim Trace Value

BIST_FTRIM_TRACE[4:0] — Bist Ftrim Trace Value

9.5.2.10. OSC Bist 测试结果寄存器（OBTRR）

地址：CLOCK_BASEADDR + 0x0000_0024

	31	30	29	28	27	26	25	24
读:	0						BIST_DONE	BIST_PASS
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							BISTRE-SULT[16]
写:								
复位:	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8
读:	BISTRESULT[15:8]							
写:								
复位:	0	0	0	0	0	1	1	1

	3	6	5	4	3	2	1	0
读:	BISTRESULT[7:0]							
写:								
复位:	0	1	1	1	1	0	1	1

= 写入没有影响，访问终止，没有传输错误异常。

图 9-11：OSC Bist 测试结果寄存器（OBTRR）

BIST_DONE — Bist Done

1 = 完成

0 = 未完成

BIST_PASS — Bist Pass

1 = 通过

0 = 失败

BISTRESULT[16:0] — Bist Trim Result

当 BIST_DONE = 1 和 BIST_PASS = 1 时有效

9.6. 功能描述

9.6.1. 打开 PLL

建议采取以下步骤：

1. 首先配置 SYNCR[PLLSRCEN]。
2. 根据目标 VCO 频率配置 PLLCSR[PLLOD]/PLLCSR[PLLN]/PLLCSR[PLLM]。
3. 配置 SYNCR[PLLEN]并打开 PLL。
4. 等待 SYNCR[PLLOCK]状态和系统时钟将被更改为 PLL 时钟。
5. 根据目标系统频率配置 SYNCR[PLLDIV]。

9.6.2. PLL 测量的频率

建议执行以下步骤：

1. 将 BIST_RESETN 位设置为 0 以进行产生复位。
2. 正在设置 BIST_TARGET 计数器的值。
3. 将 BIST_MODE 位设置为 1。
4. 为 PLL 测量，将 IRC_BIST_SEL 位设置为 1。
5. 请将 BIST _START 位设置为 1。
6. 将 BIST_RESETN 位设置为 1。
7. 正在等待 BIST_DONE 位。
8. 读取 BIST_TEST_CNT 来测量 PLL 时钟的频率。
9. 根据 BIST_TEST_CNT 计算 PLL 时钟的频率和 fxosc 时钟的频率。

PLL 时钟的频率计算方法如下：

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

9.6.3. 128KHz 测量的频率

建议执行以下步骤：

1. 将 BIST_RESETN 位设置为 0 以进行产生复位。
2. 正在设置 BIST_TARGET 计数器的值。
3. 将 BIST_MODE 位设置为 1。
4. 为 128KHz 时钟测量，将 IRC_BIST_SEL 位设置为 0。
5. 请将 BIST _START 位设置为 1。
6. 将 BIST_RESETN 位设置为 1。
7. 正在等待 BIST_DONE 位。
8. 读取 BIST_TEST_CNT，测量 128KHz 时钟的频率。
9. 根据 BIST_TEST_CNT 计算 128KHz 时钟的频率和 fxosc 时钟的频率。

128KHz 的频率计算方法如下：

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

10. 复位控制模块（RCM）

10.1. 基本概述

复位控制模块是用来确定复位的原因，向系统确认适当的复位信号，然后保持复位原因的历史记录。

10.2. 功能特点

模块功能包括：

•复位的五个来源：

-复位电源

-外部复位信号（复位编号）

-软件复位

-看门狗计时器复位

-可编程电压检测复位

•指示上次复位的原因的软件可读状态标志

10.3. 方块图

下图 10-1 说明复位控制器

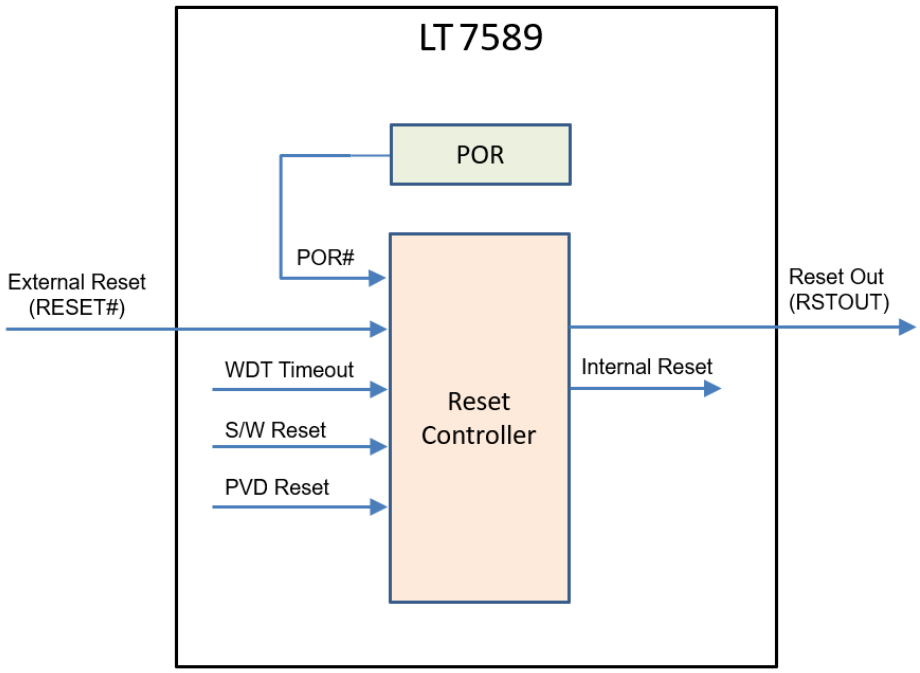


图 10-1：复位控制器的方块图

10.4. 内存映射和寄存器

10.4.1. 内存映射

复位控制器编程模型由以下几种寄存器组成：

•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）—反映最后一次复位源的状态

本章节描述了复位控制器模块的内存映射和寄存器。复位控制器的基本地址为 0x4002_0000。详见下表为地址地图和以下段落对寄存器的描述。

表 10-1：复位控制器地址映像

地址	Bits[7:0]	访问权限 [1]
0x0000	保留	S/U
0x0001	RTR—Reset Test Register	S/U
0x0002	RSR—Reset Status Register	S/U
0x0003	RCR—Reset Control Register	S/U

提示： S/U = 主管或用户模式访问。

10.4.2. 寄存器说明

10.4.2.1.复位测试寄存器（RTR）

复位测试寄存器（RTR）仅用于工厂测试。

地址：R CM_BASEADDR + 0x0000_0001

	7	6	5	4	3	2	1	0
读:	TEST[1:0]		0	0	0	0	0	FPOR
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 10-2：复位测试寄存器（RTR）

TEST[1:0] — RTR Write Access Sequence In

不能更改 FPOR 寄存器的可写位，除非写入了正确的序列。正确的序列是：2’01 → 2’10 → 2’b11。在用这个序列写入这两个位后，这两个位的值 ==2’b11，然后可以随意更改 FPOR 位的可写位。当值等于 2’b11 时，写入 2’b00 可以清除这两个位。写入其他值 s 没有效果，并返回 2’b11。

FPOR — Force Power On Reset

将 0x5B 写入 RTR 寄存器，然后设置这个位将导致系统电源复位。

复位将导致芯片再次修剪。

10.4.2.2. 复位状态寄存器（RSR）

复位状态寄存器（RSR）包含每个复位源的状态位。当输入复位时，复位条件的原因与在复位条件时没有等待处理的其他复位源的值为 0 一起被锁定。这些值随后被反映在 RSR 中。可以同时设置一个或多个状态位。任何后续复位的原因也被记录在寄存器中，覆盖来自前一个复位条件的状态。

RSR 可以随时被读取。写入到 RSR 没有任何效果。

地址：R CM_BASEADDR + 0x0000_0002

	7	6	5	4	3	2	1	0
读:	ERST	PVD	SOFT	WDTR	POR	0	0	0
写:								
复位:	0	复位相关						

= 写入没有影响，访问终止，没有传输错误异常。

图 10-3：复位状态寄存器（RSR）

ERST — External Reset

此位表示上次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 — Programmable Voltage Detect

此位表示上次复位状态是由 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 — Software Reset Flag

此位表示上次重位状态是由软件造成的。

1 = 上次复位状态是由软件引起的。

0 = 上次复位状态不是由软件造成的。

WDTR — Watchdog Timer Reset Flag

此位表示上次复位状态是由看门狗计时器超时引起的。

1 = 上次复位状态是由看门狗计时器超时引起的。

0 = 上次复位状态不是由看门狗计时器超时引起的。

POR — Power-On Reset Flag

此位表示上次复位状态是由开机复位引起的。

1 = 上次复位状态是由于开机复位所致。

0 = 上次复位状态不是由开机复位引起的。

10.4.2.3. 复位控制寄存器（RCR）

复位控制寄存器（RCR）允许软件控制请求复位。

地址：R CM_BASEADDR + 0x0000_0003

	7	6	5	4	3	2	1	0
读:	0	0	0	0	0	0	0	FRCRSTOUT
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 10-4：复位控制寄存器（RCR）

FRCRSTOUT — Force RSTOUT Pin

输出位允许软件驱动外部 RSTOUT 引脚到低或高状态。

1 = 写“1”到这个位将驱动 RSTOUT pin 到低。

0 = 写“0”到这个位将驱动 RSTOUT pin 到高。

10.5. 功能描述

10.5.1. 复位源

下表定义复位源和由复位控制器驱动的信号。

表 10-2：复位源汇总

复 位 来 源	类 型
POR	异步
ERST	异步
Watchdog timer	异步
Software	同步的
PVD	异步

为了保护数据的完整性，在当前总线周期结束之前，复位控制逻辑不会操作同步复位源。然后，在循环终止后，在系统时钟的下一个上升边缘进行复位。当复位控制逻辑必须同步复位到总线周期结束时，内部总线看门狗将自动启用。

异步复位源通常表示发生灾难性故障。因此，复位控制逻辑不等待当前总线周期完成。复位将立即声明到系统。

10.5.1.1. 通电复位（POR）

在通电时，复位控制器产生系统复位。继续进行系统复位，直到 POR 达到可接受的最低水平。

10.5.1.2. 看门狗计时器复位

看门狗计时器超时会导致计时器复位请求被识别并被锁定。

10.5.1.3. 软件复位

如果设置了 32 位 RISC 嵌入式中断控制器（EIC）中的 SYSRESTEQ 位，则将生成软件复位。复位控制器维护系统复位约 2048 个周期。然后，芯片退出复位，恢复操作。

10.5.1.4. 可编程电压检测复位

当设置了嵌入式闪存模块（EFM）中的 CCR 寄存器的 PVDRE 位时，当 VDD 超过 PVD 阈值时，PVD 将产生复位。

10.5.2. 复位控制流程

复位逻辑控制流程如下图所示给出的所有周期计数都是近似的。

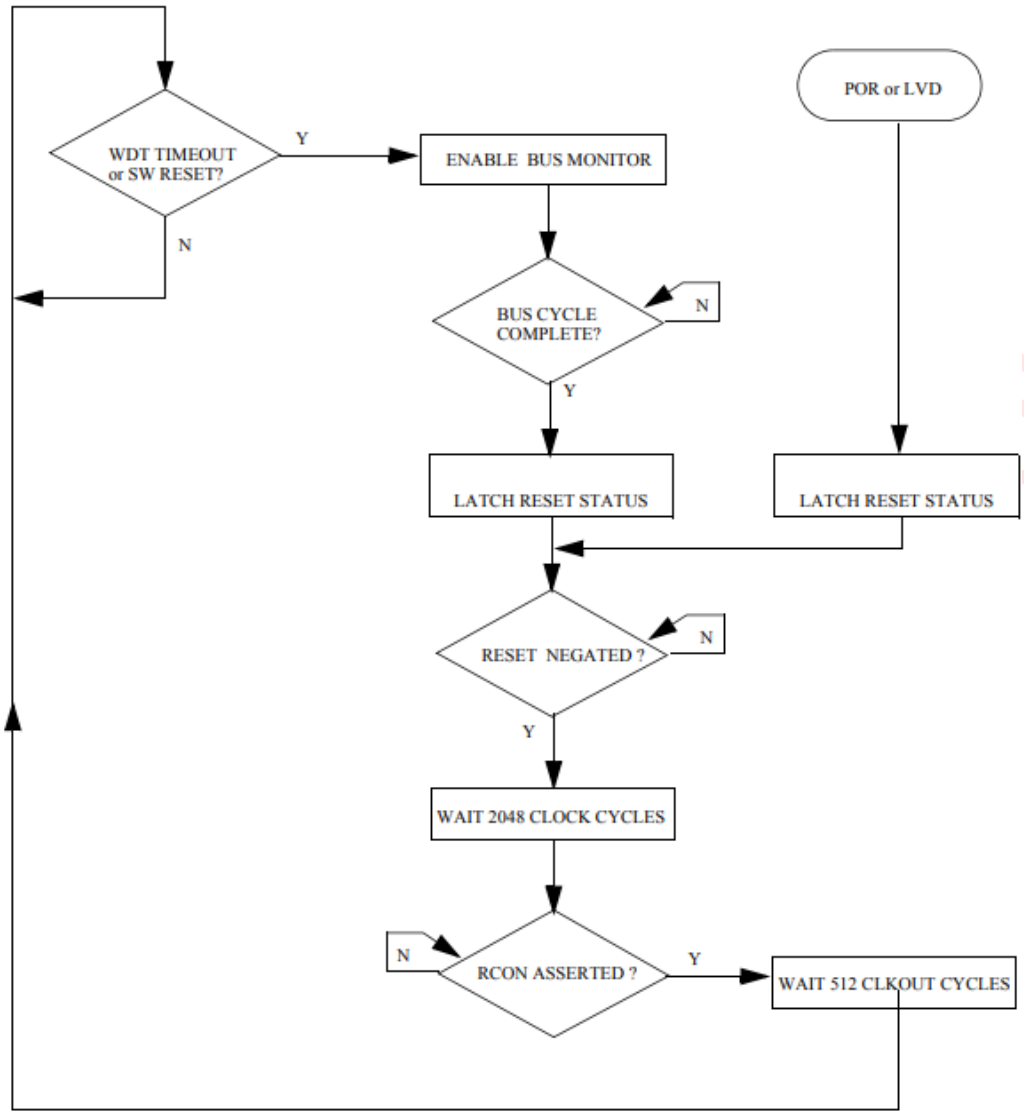


图 10-5：复位控制流程

11. 静态随机存取存储器（SRAM）

11.1. 功能介绍

LT7589 的静态随机存取存储器（SRAM）的功能包括：

- 芯片上的 64K 字节数 SRAM
- 固定地址空间
- 字节、半字（16 位）或 Word（32 位）的读/写访问
- 每次访问一个时钟（包括字节、半字和单词）
- 主管或用户模式访问

11.2. 操作模式

对 SRAM 的访问不受任何限制。该阵列可以在主管模式和用户模式下进行访问。

11.3. 低功耗模式

在等待、睡觉和停止模式下，到 SRAM 的时钟被禁用。当退出这些模式时，不需要任何恢复时间。

11.4. 复位操作

在开机复位后，SRAM 内容立即未定义。SRAM 内容不受系统复位的影响。如果在读取或写访问期间发生同步复位，则访问将正常完成，正在进行的任何管道访问都将停止，而不会损坏 SRAM 内容。

11.5. 中断说明

SRAM 模块不生成中断请求。

12. 高速缓存模块（CACHEM）

12.1. 功能介绍

缓存模块为处理器提供了紧密耦合的处理器-本地内存和到 EBI 和 QSPI0/1/2 内存空间的总线路径。

高速缓存是一个包含地址信息（通常称为标记）和相关数据的高速存储器位置块。其目的是减少内存访问的平均时间。缓存的两个原则：

- 空间局部性—对一个位置的访问可能随后访问相邻位置（例如，顺序指令执行或数据结构的利用）。
- 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）。

为了最小化存储的控制信息的数量，使用空间局部性属性将同一标签下的多个位置分组在一起。此逻辑块通常被称为高速缓存行。

当数据被加载到高速缓存中时，后续加载和存储的访问时间就会减少。这将提高整体性能。对缓存中已经存在的信息的访问称为缓存命中，其他访问称为缓存未命中。

通常，缓存是自我管理的，更新会自动发生。每当处理器想要访问可缓存位置时，都会检查缓存命中。如果访问是缓存命中，则立即发生访问。否则，将分配一个位置，并从内存中加载高速缓存行。可能会有不同的高速缓存拓扑结构和访问策略。

但是，它们必须符合底层体系结构的内存一致性模型。缓存引入了一些潜在的问题，主要原因是：

- 内存访问发生在程序员通常期望的时候；
- 存在可以保存一个数据项的多个物理位置。

缓存控制器的目标是使用任何需要缓存功能的基于 32 位 AHB 总线的应用程序。这个缓存函数可以通过提供对最近使用的代码或数据的快速访问来提高性能。

本地内存控制器支持三种操作模式：

1. 使用此缓存模式对地址空间的通写访问是可缓存的。
   - 输入总线上的通读错误导致包含所需地址的 16 字节对齐的内存地址的输出总线上读取行。此丢失数据被加载到缓存中，并标记为有效且未修改。
   - 对有效缓存位置的穿透读命中从缓存返回数据，而没有输出总线访问。
   - 通写错误绕过缓存并写入输出总线（对通写模式空间无写错误策略分配）。
   - 穿透写入命中更新缓存命中数据并写入输出总线。
2. 使用此缓存模式对地址空间的回写访问是可缓存的。
   - 输入总线上的回写读取错误将导致在包含所需地址的与 16 字节对齐的内存地址的输出总线上读取一条行。此丢失的数据被加载到缓存中，并标记为有效且未修改。
   - 对有效缓存位置的回写读命中将从缓存返回数据，而没有输出总线访问。

- 回写错误将执行“读到写”（为回写模式空间分配写错误策略）。执行在包含所需写地址的 16 字节对齐的内存地址的输出总线上读取的行。此丢失数据被加载到缓存中并标记为有效并已修改；然后写入数据将更新适当的缓存数据位置。

3. 不可缓存—对具有此缓存模式的地址空间的访问不可缓存。

- 这些访问会绕过高速缓存并访问输出总线。

缓存控制器有两个 AHB 总线接口、一个主接口和一个从接口。主接口具有解码逻辑，以确定有效地址阶段访问的高速缓存模式。然后，主访问将根据它们的缓存模式访问或绕过缓存。从属接口用于缓存丢失以及绕过缓存的访问。

缓存控制器有一个双向集关联组织。高速缓存有 32 位宽的地址和数据路径，以及 16 字节的行大小。缓存标记和数据存储使用单端口、同步 ram。该控制器具有各种读取和写的数据缓冲区，以提高性能。缓存失败和线推送生成 4 节拍 32 位封装突发访问，首先访问关键字以获得最大的性能。

12.2. 方块图

这些缓存模块提供了对 RAM 和可缓存地址空间的零等待状态访问。

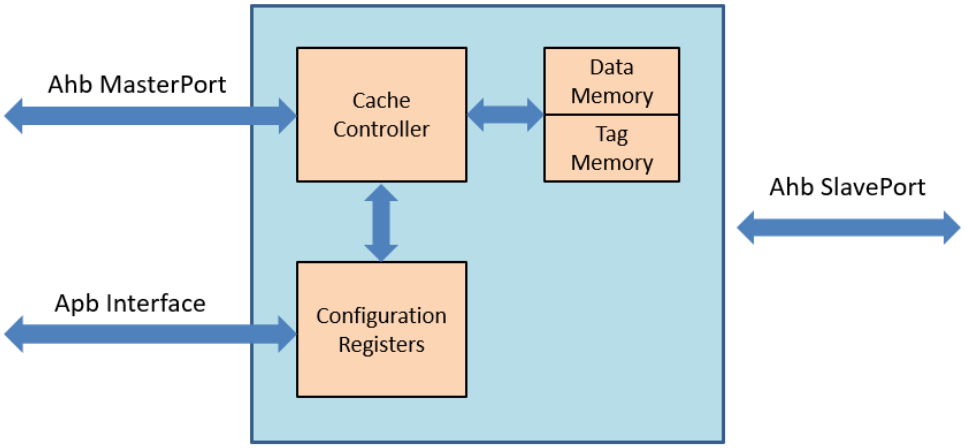


图 12-1：高速缓存模块的方块图

12.3. 内存映射和寄存器

12.3.1. 内存映射

本章节描述了高速缓存模块的内存映射和寄存器。高速缓存模块的基本地址为 0x4019_0000。下表显示了这些寄存器的偏移量地址。

表 12-1：高速缓存模块内存映射

地址偏移	Bits[31:0]	访问权限
0x0000	Cache Control Register (LMEM_CCR)	S/U
0x0004	Cache Line Control Register (LMEM_CLCR)	S/U
0x0008	Cache Search Address Register (LMEM_CSAR)	S/U
0x000C	Cache Read/Write Value Register (LMEM_CCVR)	S/U
0x0020	Cache Access Register(LMEM_ACR)	S/U
0x0180	Cache Page Invalidation Start Address	S/U
0x0184	Cache Page Invalidate Size	S/U
0x0188	Cache Clock Gate	S/U

提示：

1. S = CPU 监控模式访问，U = CPU 用户模式访问
2. 在用户模式下访问仅主管地址位置没有影响，并导致周期终止传输错误

12.3.2. 寄存器说明

本章节提供了对高速缓存模块的描述。

12.3.2.1. 高速缓存控制寄存器（LMEM_CCR）

地址：CACHE_BASEADDR + 0x0000_0000

	31	30	29	28	27	26	25	24
读:	GO	0			PUSHW1	INVW1	PUSHW0	INVW0
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0							ENCACHE
写:								
复位:	0	0	1	1	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 12-2：高速缓存控制寄存器（LMEM_CCR）

GO — Initiate Cache Command
设置此位将启动由第 27-24 位指示的高速缓存命令。读取此位表示一个命令是否处于活动状态。这个位一直保持设置，直到命令完成。写零没有任何效果。

1 = 写：由第 27-24 位指示的启动命令。读取：高速缓存命令处于激活状态。

0 = 写：没有效果。读取：没有激活的高速缓存命令。

PUSHW1 — Push way 1

1 = 设置 GO 位时，按所有修改的行 1

0 = 无操作

INVW1 — Invalidate way 1.
如果设置了 PUSHW0 和 INVW0 位，则在设置了 GO 位之后，按方式 1 推送所有修改过的行，并按方式 1 使所有行无效（清除方式 1）。

1 = 在设置 GO 位时，以 1 的方式使所有行无效。

0 = 无操作

PUSHW0 — Push way 0

1 = 设置 GO 位时，按 0 推送所有修改的行

0 = 无操作

INVW0 — Invalidate way 0.

如果设置了 PUSHW0 和 INVW0 位，则在设置 GO 位之后，按 0 推送所有修改行，按 0 使所有行无效（清除 0）。

1 = 在设置 GO 位时，将以 0 的方式使所有行无效。

0 = 无操作

ENCACHE — Cache enable.

1 = 缓存已启用

0 = 缓存已禁用

12.3.2.2. 高速缓存线控制寄存器（LMEM_CLCR）

地址：CACHE_BASEADDR + 0x0000_0004

	31	30	29	28	27	26	25	24
读:	0				LACC	LADSEL	LCMD[1:0]	
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0	LCR[2:0]			0			TDSEL
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0	WSEL	CACHE_ADDRESS [13:8]					
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	CACHE_ADDRESS [7:2]						0	LGO
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 12-3：高速缓存线控制寄存器（LMEM_CLCR）

LACC — Line access type.

0 = 读取

1 = 写入

LADSEL — Line Address Select.

当使用缓存地址时，还必须在 CLCR[WSEL]中指定指定的方式。当使用物理地址时，这两种方式都会被搜索，并且只有在命中时才会执行命令。

0 = 缓存地址

1 = 物理地址

LCMD[1:0] — Line command.

00 = 搜索和读取或写。

01 = 无效

10 = 推入

11 = 清除

LCR[2:0] — Line command current line status.

TDSEL — Tag/Data select. Select tag or data for search and read or write commands.

1 = 标签。

0 = 数据。

WSEL — Way select. Select the way of the line commands.

1 = 方式 1。

0 = 方式 0。

CACHE_ADDRESS[13:2] — Cache address.

CLCR[13:4]位用于访问标记数组；CLCR[13:2]位用于访问数据数组。

LGO — Initiate Cache Line Command.

设置此位将启动由 LMEM_CLCR[27-24]指示的高速缓存行命令。读取此位表示行命令是否处于活动状态。这个位一直保持设置，直到命令完成。写零没有任何效果。此位与 CSAR[LGO]共享。

1 = 写入：由第 27-24 位指示的初始化行命令。读取：行命令处于激活状态。

0 = 写：没有效果。读取：没有行命令。

12.3.2.3. 高速缓存搜索地址寄存器（LMEM_CSAR）

CSAR 寄存器用于为在 CLCR[LADSEL]位中指定的行大小的命令定义显式高速缓存地址或物理地址。

地址：CACHE_BASEADDR + 0x0000_0008

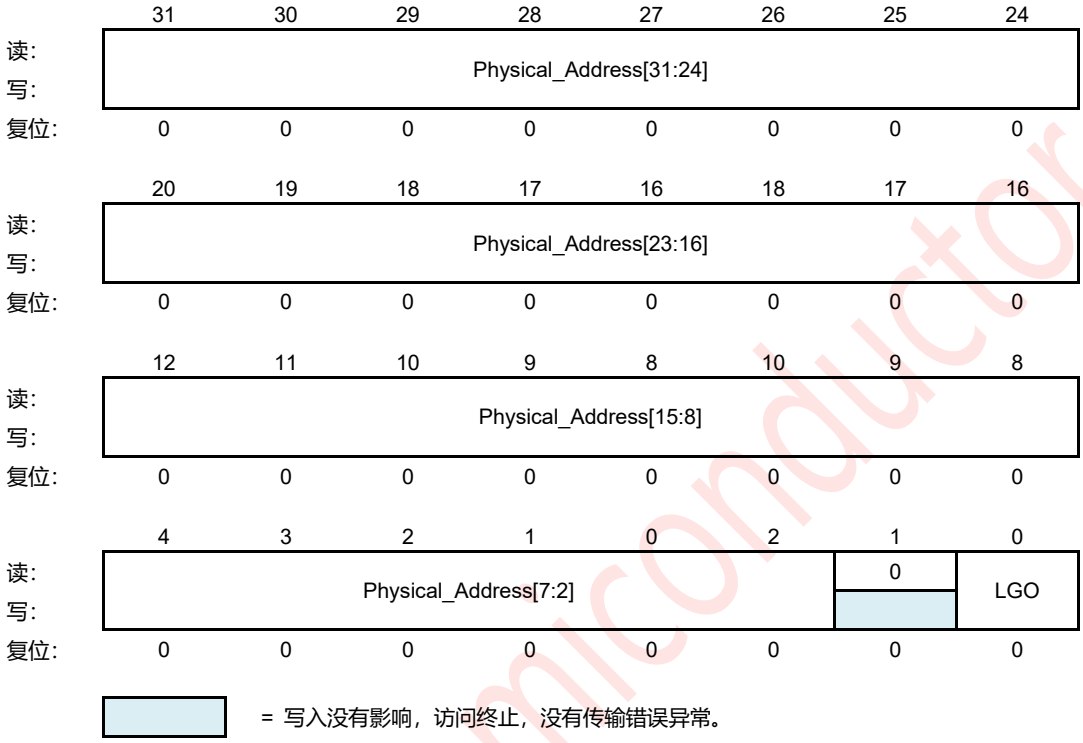


图 12-4：高速缓存搜索地址寄存器（LMEM_CSAR）

Physical_Address[31:2] — Physical Address. This represents bits [31:2] of the system address.

物理地址位用于标记比较

物理地址位用于访问标记数组

物理地址的位用于访问数据数组。

LGO — Initiate Cache Line Command.

设置此位将启动由 LMEM_CLCR[27-24]指示的高速缓存行命令。读取此位表示行命令是否处于活动状态。这个位一直保持设置，直到命令完成。写零没有任何效果。

此位与 CLCR[LGO]共享

- 写：启动由 LMEM_CLCR[27-24]指示的行命令。读取：行命令处于激活状态。
- 写：没有效果。读取：没有行命令。

12.3.2.4. 高速缓存读/写值寄存器（LMEM_CCVR）

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

地址：CACHE_BASEADDR + 0x0000_000C

	31	30	29	28	27	26	25	24
读: 写:	DATA[31:24]							
复位:	0	0	0	0	0	0	0	0

	19	22	21	20	19	18	17	16
读: 写:	DATA[23:16]							
复位:	0	0	0	0	0	0	0	0

	11	14	13	12	11	10	9	8
读: 写:	DATA[15:8]							
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读: 写:	DATA[7:0]							
复位:	0	0	0	0	0	0	0	0

图 12-5：高速缓存读/写值寄存器（LMEM_CCVR）

DATA[31:0] — Cache read/write Data

对于标签搜索、读取或写入：

- 数据的位用于标记数组的 R/W 值
- DATA[13:4]位用于读取时的标记集地址；在写入时未使用
- DATA[3:2]位保留

对于数据搜索，请读取。或写下：

- 数据数据位用于数据阵列的收发值

12.3.2.5. 高速缓存访问寄存器（LMEM_ACRG）

ACRG 用于控制 10 个不同内存区域的属性，如通写、回写和不可缓存。

地址：CACHE_BASEADDR + 0x0000_0020

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	R3_CACHEABLE	R3_WT_WB	R2_CACHEABLE	R2_WT_WB	R1_CACHEABLE	R1_WT_WB	R0_CACHEABLE	R0_WT_WB
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 12-6：高速缓存访问寄存器（LMEM_ACRG）

Region0: 0x2000_0000 ~ 0x2FFF_FFFF
Region1: 0x6000_0000 ~ 0x6FFF_FFFF
Region2: 0x7000_0000 ~ 0x7FFF_FFFF
Region3: 0x8000_0000 ~ 0x8FFF_FFFF

Rx_CACHEABLE — Region x is cacheable.

1 = 可缓存

0 = 不可缓存

Rx_WT_WB — Region x is write-through if cacheable.

1 = 回写

0 = 写入

12.3.2.6. 高速缓存页面未验证基本地址寄存器（LMEM_PAGE_INV_BADDR）

地址：CACHE_BASEADDR + 0x0000_0180

	31	30	29	28	27	26	25	24
读: 写:	Page_ Inv_ Baddr[31：24]							
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读: 写:	Page_ Inv_ Baddr[23:16]							
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读: 写:	Page_Inv_Baddr [15:8]							
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	Page_Inv_Baddr [7:4]				0	0	0	Start_Page_Inv
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 12-7：高速缓存页不验证基本地址寄存器（LMEM_PAGE_INV_BADDR）

Page_Inv_Baddr[31:4] — cache invalidate start address for system memory.

由于行对齐，将忽略较低的 4 位。

Start_Page_Inv — start to page invalidate. Cleared if the invalidation is completed.
此位在 LMEM_PAGE_INV_SIZE_REG 的第 0 位中共享。

12.3.2.7. 高速缓存页面未验证基本大小寄存器（LMEM_PAGE_INV_SIZE)

地址：CACHE_BASEADDR + 0x0000_0184

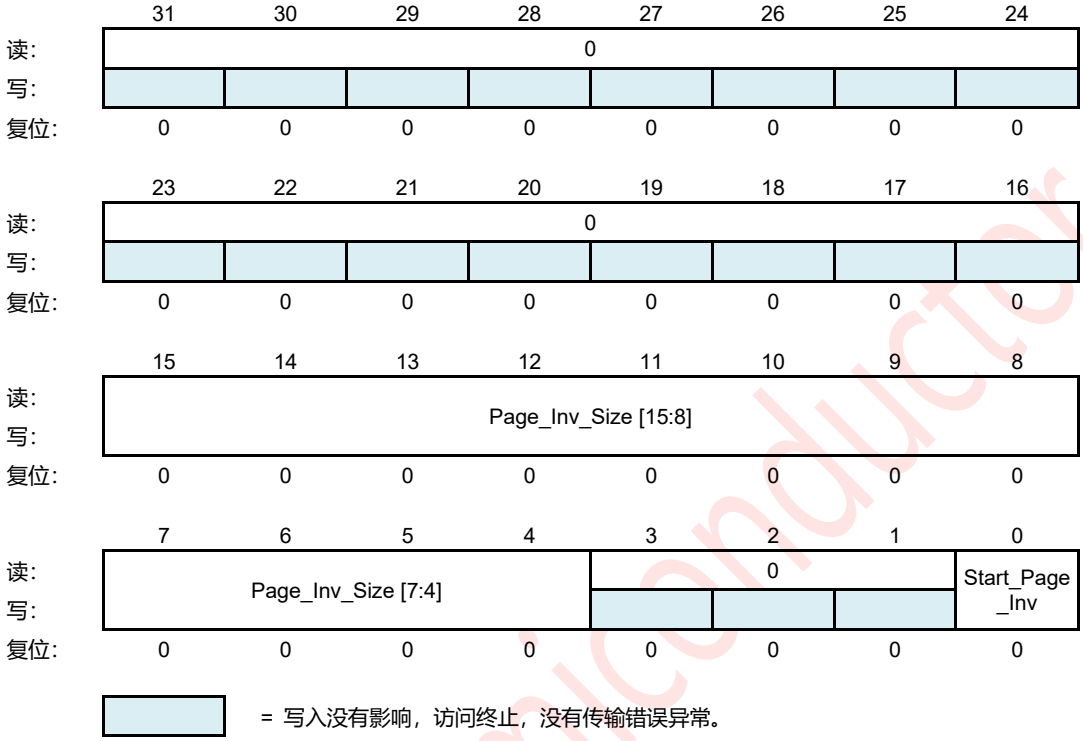


图 12-8：高速缓存页面不验证大小寄存器（LMEM_PAGE_INV_SIZE)

Page_Inv_Size[15:4] — cache invalidate size for system memory.

由于行对齐，将忽略较低的 4 位。

Start_Page_Inv — start to page invalidate. Cleared if the invalidation is completed.

此位在 LMEM_PAGE_INV_BADDR_REG 的第 0 位中共享。

12.3.2.8. 高速缓存时钟启用寄存器（LMEM_CACHE_CLK_EN）

地址：CACHE_BASEADDR + 0x0000_0188

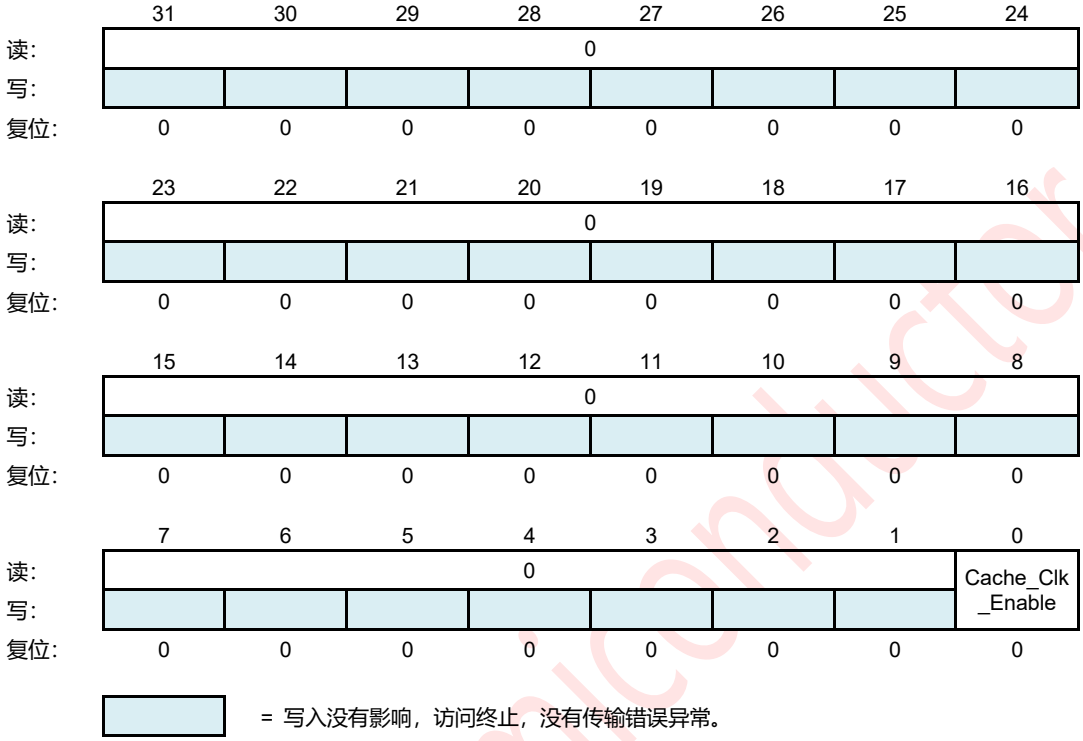


图 12-9：高速缓存时钟启用寄存器（LMEM_CACHE_CLK_EN）

Cache_Clk_Enable — cache clock is enabled. Must be ’1’ if cache is enabled.

12.4. 缓存功能

该设备上的缓存的结构如下。这两个缓存都有一个双向集关联缓存结构，总大小为 32K 字节。这些缓存有 32 位的地址和数据路径，以及 16 字节的行大小。缓存标记和数据存储使用外部单端口、同步 ram。

对于这些 32K 字节的缓存，每个缓存 TAG 使用两个 1024x20 位 RAM 数组，而缓存 DATA 使用两个 4096x32 位 RAM 数组。缓存 TAG 条目存储 18 位上地址以及每个缓存行修改的有效位。缓存数据条目存储四个字节的代码或数据。

所有正常的高速缓存访问都将使用物理地址。这将导致以下高速缓存地址的使用：

CACHE - 32K Bytes size = (1024 sets) x (16-byte lines) x (2-way set-associative)

TAG:

- Address[31:14] used in tag for compare (hit) logic
- Address[13:4] used to select 1 of 1024 sets
- Address[3:0] not used

DATA:

- Address[31:14] not used
- Address[13:4] used to select one of 1024 sets
- Address[3:2] used to select one of four 32-bits words within a set
- Aaddress[1:0] used to select the byte within the 32-bits word

缓存使用一个周期的并行 TAG 和数据访问结构。该结构在两阶段 AHB 总线管道中从地址阶段的负边排列到数据阶段的负边：

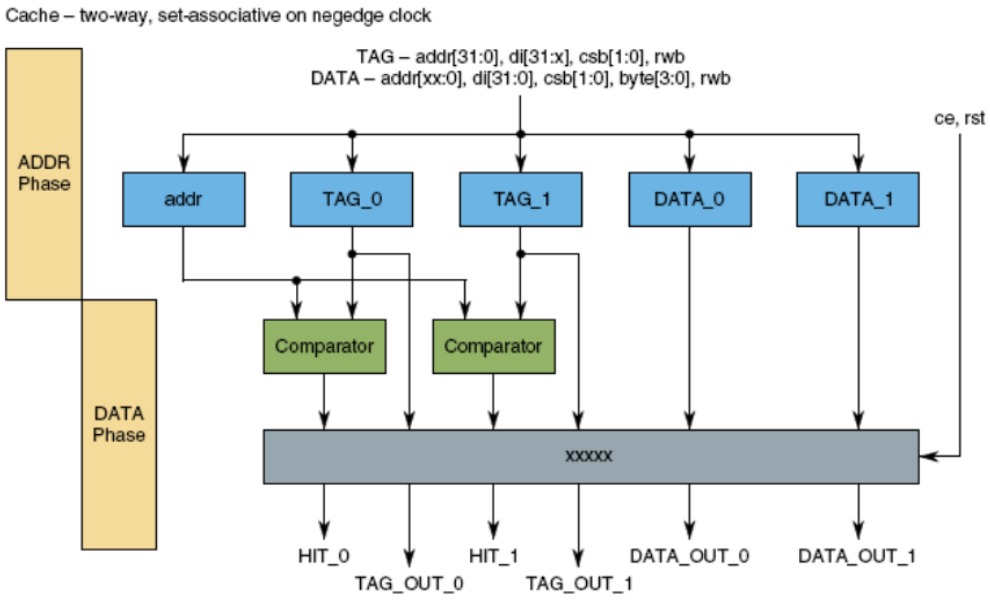


图 12-10：高速缓存标签和数据访问结构

在正常的可缓存操作中，完整地址寄存器在缓存控制逻辑中的地址阶段的负边缘上，而地址位[12:4]在 TAG 数组中寄存器，地址位[12:2]在 DATA 数组中寄存器。在从地址阶段的负边缘到数据阶段的负边缘的整个周期中，将访问 TAG 和 data 阵列并分析缓存命中。对于命中值，在数据阶段的后半段返回所需的读取数据，而写入操作则从数据阶段周期的负边缘开始完成。对于失败，数据阶段根据需要保持在缓存阶段，同时使用其从总线（CCM、CSM）通过发送到交叉条开关的行大小的数据传输访问所需的缓存线。

12.5. Cache 缓存控制

代码和系统缓存在复位时被禁用。复位时未清除缓存标记和数据数组。因此，要启用缓存，必须执行缓存命令来清除和初始化所需的标记数组位，并配置和启用缓存。

12.5.1. 缓存集命令

缓存集命令可以用于以下操作：

- 所有方式 0，
- 所有的方式 1，或
- 所有两种方式（完全缓存）。

缓存集命令将使用 CCR 寄存器中的上位来启动。缓存集命令在缓存上执行 CCR 的缓存启用位 CCR[ENCACHE]。

通过设置 CCR[GO]位来启动高速缓存集命令。此位也充当设置命令的繁忙位。当命令被激活时，它保持设置，并在设置命令完成时由硬件清除。

下表给出了支持的缓存集命令。设置命令的工作方式如下：

- 无效–无条件清除有效并修改缓存条目的位。
- 按–如果缓存条目有效并已修改 Push a cache entry，然后清除修改位。如果条目无效或未修改，请保持原样。
- 清除–推送缓存条目是否有效，并已修改，然后清除有效和修改的位。如果该项无效或未被修改，请清除该有效位。

表 12-2：缓存集命令

CCR【27:24】				命 令
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 0
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

CCR【27:24】				命 令
PUSH W1	INVW1	PUSH W0	INVW0	
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 0
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)

复位后，在使用缓存之前，先完成一个无效的缓存命令。

12.5.2. Cache Line Commands 缓存行命令

缓存行命令一次对缓存中的一行进行操作。高速缓存行命令可以使用物理地址或高速缓存地址来执行。

- 缓存地址由集合地址和方式选择组成。该行命令作用于指定的高速缓存行。
- 带有物理地址的缓存行命令首先搜索由物理地址的位[13:4]指定的缓存集的两种方式。如果它们命中，命令就会以命中的方式执行操作。

高速缓存行命令将使用 CLCR 寄存器中的上一位来指定。缓存行命令对独立于缓存启用位（CCR[enc 缓存]）执行操作。当使用高速缓存地址时，该命令可以完全由 CLCR 寄存器指定。当使用物理地址时，该命令还必须使用 CSAR 寄存器来指定物理地址。

通过设置行命令 go 位（CLCR[LGO]或 CSAR[LGO]）来启动行缓存命令。这个位也充当行命令的繁忙位。当命令处于活动时保持设置，并在命令完成时被硬件清除。

CLCR[27:24]位选择以下行命令：

表 12-3：缓存行命令

Clcr[27:24]			命令
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

12.5.2.1. 使用缓存地址执行一系列行命令

只需写入 CLCR，就可以执行一系列具有增量缓存地址的行命令。

- 将该命令放在 CLCR[27:24]中，
- 根据需要设置方式（CLCR[WSEL]）和标签/数据（CLCR[TDSEL]）控件，
- 将缓存地址放在 CLCR[缓存头数]中，和
- 设置行命令 go 位（CLCR[LGO]）。

- 当一个行命令完成后，通过以下步骤启动下一个命令：
  - 增加高速缓存地址（在第 2 位逐步通过数据或在第 4 位逐步通过行），以及
  - 设置行命令 go 位（CLCR[LGO]）。

12.5.2.2. 使用物理地址执行一系列行命令

通过以下步骤执行一系列具有增量物理地址的行命令：

- 将该命令放置在 CLCR 中[27:24]
- 设置标记/数据（CLCR[TDSEL]）控件
- 将物理地址放在 CSAR[PHYADDR]中，并设置行命令去位（CSAR[LGO]）。

当一个行命令完成时，通过以下步骤启动下一个命令：

- 增加物理地址（在第 2 位步进数据或在第 4 位步进通过行），以及
- 设置行命令 go 位（CSAR[LGO]）。

行命令 go 位在 CLCR 和 CSAR 寄存器之间共享，因此可以在对 CSAR 寄存器的一次写入中完成上述步骤。

12.5.2.3. 行命令结果

在完成行命令时，CLCR 寄存器包含有关该命令目标的行的初始状态的信息。对于具有缓存地址的行命令，在从目标缓存行执行行命令操作之前读取此信息。对于具有物理地址的行命令，在从命中缓存行执行行命令操作之前读取此信息，或者如果命令失败，则清除初始有效位。通常，如果清除了有效的指示符 CLCR[LCIVB]，则在行命令开始时目标行无效，并且没有执行行操作。

表 12-4：行命令结果

CLCR[22:20]			用于缓存地址命令	用于物理地址命令
LCWY	LCIMB	LCIVB		
0	0	0	Way 0 line 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.	

在完成除写入以外的行命令时，CCVR（缓存 R/W 值寄存器）包含有关行标记或命令目标数据的初始状态的信息。对于行命令，CLCR[TDSEL]会在标记和数据之间进行选择。如果行命令使用了物理地址，数据就不在乎。对于写命令，CCVR 保存写数据。

13. 直接内存访问控制器（DMAC）

13.1. DMA 控制器的具体信息

本章介绍了 LT7589 的直接内存访问控制器。

13.1.1. DMAC 功能

- 6 个可编程通道，以支持独立的 8、16 或 32 位单值或块传输
- 支持可变大小的队列和循环队列
- 独立配置为后增量寄存器的源地址和目标地址寄存器保持不变
- 每个传输都由外围设备、CPU、周期性定时器中断或 DMA 通道请求发起
- 外围的 DMA 请求源可能来自 QSPI，QADC
- 每个 DMA 信道能够在完成单个值或块传输时选择性地向 CPU 发送中断请求
- DMA 可能在系统内存和所有可访问的内存映像位置之间进行传输，包括外围设备和寄存器
- DMA 支持以下功能：
  - Scatter Gather
  - Channel Linking
  - Inner Loop Offset
  - Arbitration
    - Fixed Group, Fixed Channel
    - Round Robin Group, Fixed Channel
    - Round Robin Group, Round Robin Channel
    - Fixed Group, Round Robin Channel
  - Channel Preemption
  - Cancel Channel Transfer
- 中断-DMA 对每个实现的通道都有一个中断请求和一个组合的 DMA 错误中断，以标记到系统的传输错误。每个通道 DMA 中断都可以启用或禁用，并提供已完成传输的通知。有关这些中断的分配，请参考第 7 章的 EIC 章节中的中断向量。

13.1.2. 通道分配

从块到 DMA 通道的 DMA 请求之间的分配如下表所示：

表 13-1：DMA 通道分配

DMA 请求	通道	描述
ADC_ISR[EMPTY]	0	qadc_dma_req，当 FIFO 中的数据号不为空，且设置了位 ADC_CFGR1 [DMAEN]时，
QSPI0 DMA 接收请求	1	当接收 FIFO 中的有效数据条目数等于或高于 DMARDLR[DMARDL] + 1 和 DMACR[RDMAE] = 1 时，将生成 QSPI0 dma_rx_req
QSPI0 DMA 发送请求	2	当传输 FIFO 中的有效数据条目数等于或低于 DMATDLR[DMATDL]和 DMACR[TDMAE] = 1 时，将生成 QSPI0 dma_tx_req 信号
QSPI1 DMA 接收请求	3	当接收 FIFO 中的有效数据条目数量等于或高于 DMARDLR[DMARDL] + 1 和 DMACR[RDMAE] = 1 时，将生成 QSPI1 dma_rx_req
QSPI1 DMA 发送请求	4	当传输 FIFO 中的有效数据项数等于或低于 DMATDLR[DMATDL]和 DMACR[TDMAE] = 1 时，将生成 QSPI1 dma_tx_req 信号
QSPI2 DMA 接收请求	5	当接收 FIFO 中的有效数据条目数等于或高于 DMARDLR[DMARDL] + 1 和 DMACR[RDMAE] = 1 时，将生成 QSPI2 dma_rx_req
QSPI2 DMA 发送请求	6	QSPI2 dma_tx_req 信号产生时，有效的数据条目数在传输中 FIFO 等于或低于 DMATDLR[DMATDL]和 DMACR[TDMAE] = 1
PIT1 DMA 请求	7	当 PIT0 计数器达到 0x0000 和 PCSR[PDMAE] = 1 时，就会产生 pit0_dma_req 信号
PIT2 DMA 请求	8	当 PIT0 计数器达到 0x0000 和 PCSR[PDMAE] = 1 时，就会产生 pit0_dma_req 信号
PIT3 DMA 请求	9	当 PIT0 计数器达到 0x0000 和 PCSR[PDMAE] = 1 时，就会产生 pit0_dma_req 信号
SCI0 TX DMA 请求	10	当传输 FIFOis 中的数据字数等于或小于 SCI_WATER[txwater]所表示的数字时，就会产生 sci0_tx_req 信号)
SCI0 RX DMA 请求	11	当接收缓冲区中的数据字数大于 SCI_WATER[RXWATER]所表示的数字时，将生成 sci0_rx_req 信号
SCI1 TX DMA 请求	12	当传输 FIFOis 中的数据字数等于或小于 SCI_WATER[txwater]所表示的数字时，就会产生 sci1_tx_req 信号)
SCI1 RX DMA 请求	13	当接收缓冲区中的数据字数大于 SCI_WATER[RXWATER]所表示的数字时，将生成 sci1_rx_req 信号

DMA 请求	通道	描述
SCI2 TX DMA 请求	14	当传输 FIFOis 中的数据字数等于或小于 SCI_WATER[txwater]所表示的数字时，就会产生 sci2_tx_req 信号)
SCI2 RX DMA 请求	15	当接收缓冲区中的数据字数大于 SCI_WATER[RXWATER]所表示的数字时，将生成 sci2_rx_req 信号

13.2. 功能介绍

直接存储器访问控制器（DMA）能够通过 16 个可编程通道执行复杂的数据移动，而来自主机处理器的最小干预。硬件微架构包括执行源地址计算和目标地址计算的 DMA 引擎，以及实际的数据移动操作，以及包含通道的传输控制描述符（TCD）的基于 SRAM 的内存。这个实现将最小化到整体块的大小。

下图这是 DMA 模块的结构图。

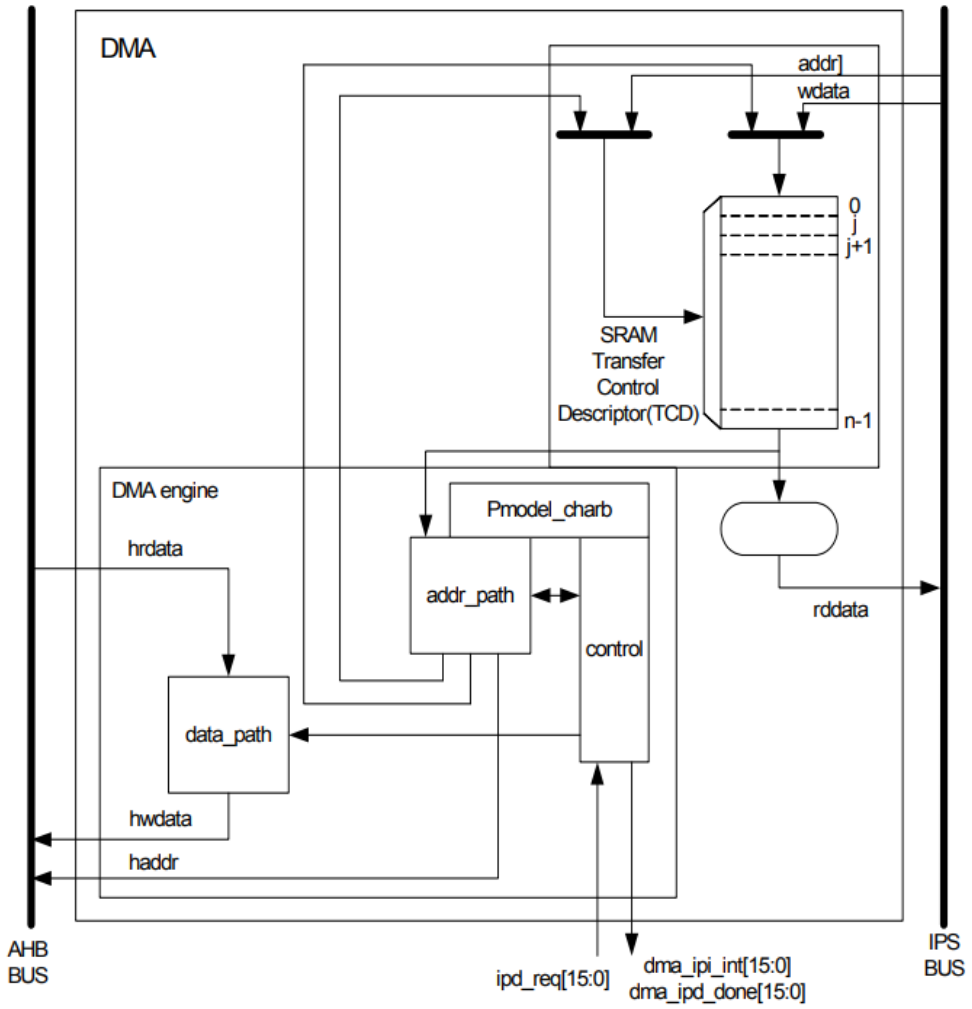


图 13-1：DMA 结构图

13.2.1. 模块特点

DMA 模块支持以下功能：

- 所有的数据移动通过双地址传输：从源代码读取，写到目的地
  - 可编程的源，目标地址，传输大小，并支持增强的寻址模式
- 传输控制描述符组织以支持双深度嵌套传输操作
  - 内部数据传输循环由一个“次要”字节传输计数定义
  - 一个外部的数据传输循环是由一个“主要的”迭代计数来定义的
- 通过以下三种方式之一提出的渠道服务请求：
  - 显式软件启动
  - 通过信道到信道的连接机制进行启动，以进行连续传输。在小回路和/或大回路的末端的独立通道连接
  - 外围设备有节奏的硬件请求（每通道一个）
  - 对于所有三种方法，每次执行小循环都需要一个服务请求
- 支持固定优先级和循环信道仲裁
- 通过可选的中断请求报告通道完成情况
  - 每个通道有一个中断，可选择在完成主要迭代计数时产生
  - 错误终止可选地启用每个通道，并逻辑求和形成少量的错误中断输出
- 支持分散/收集 DMA 处理
- 支持复杂的数据结构
- 支持通过软件或硬件进行传输

14. 选项字节（OPB）

14.1. 内存映射和寄存器

14.1.1. 内存映射

选项字节内存映射如下表所示，其基本地址为 0x4012_0000，与 Efuse 模块相同。

表 14-1：寄存器内存映射

地址偏移	Bits[31:24]	Bits[23:16]	Bits[15:8]	Bits[7:0]	访问权限 [1,2]
0x001C	CCR		PVDC		S
0x0020	PLLOCKCR		EOSCST		S
0x0024	PVDFEVR		RFEVR		S
0x002C	FCR		保留 3		S
0x0030	保留				S
0x0034	LDOTCR	VREFTCR	ADCCDISR		S
0x0038	RTCTCR		保留		S

提示：

1. 仅限 S = CPU 监控模式访问。
2. A 在用户模式下只处理主管地址位置没有影响，并导致周期终止传输错误。
3. 写入保留地址位置没有影响，读取返回 0 秒。

14.1.2. 寄存器描述

14.1.2.1. 可编程电压检测器配置寄存器（PVDC）

地址：EFM_BASEADDR + 0x0000_001C

	15	14	13	12	11	10	9	8
读:	PVDF	PVDPORRE	PVDTEST[1:0]		PVDOE	PVDRE	PVDIE	PVDE
写:								
复位:	0	注 1	0	0	1	注 1	0	1

	7	6	5	4	3	2	1	0
读:	0						PVDC[1:0]	
写:								
复位:	0	0	0	0	0	0	注 1	注 1

= 写入没有影响，访问终止，没有传输错误异常。

提示 1：由保险丝面积的值确定

图 14-1：可编程电压检测器配置寄存器（PVDC）

PVDF — Programmable Voltage Detector Flag

PVDF 表示 VDD 是否低于 PVD 阈值。POR 可以清除它。

1 = VDD33 低于 PVD 阈值

0 = VDD33 不低于 PVD 阈值

PVDPORRE — Program Voltage Detector (PVD) Power-On Reset Enable

端口显示程序电压检测器是否启用通电复位。启用后，在 PVD 复位后将设置 RSR[POR]标志。默认值将从 FUSE 区域加载。如果相应的熔断区域处于擦除状态，则清除该位，否则如果相应的熔断区域中的内容匹配，则开机后从熔断区域的相应位加载。此位可在通电复位后由软件设置。

1 = 当 VDD33 低于 PVD 阈值时，PVD 将产生接通电复位

0 = 当 VDD33 低于 PVD 阈值时，PVD 将不会产生通电复位

PVDTEST[1:0] — Write Access Enable Sequence Input

不能对 PVDC 寄存器进行更改，除非写入正确的序列。正确的序列是：2'b01 → 2'b10 → 2'b11。在用这个序列写入这两位后，这两位的值为 == 2'b11，然后就可以随意改变 PVDC 寄存器。当值等于 2'b11 时，写入 2'b00 可以清除这两个位。写入其他值没有效果，并返回 2'b11。

PVDOE — Programmable Voltage Detector Output Enable

1 = PVD 输出已启用

0 = PVD 输出已禁用

PVDRE — Program Voltage Detector (PVD) Reset Enable

PVDRE 显示是否启用了程序电压检测器复位。默认值将从 FUSE 区域加载。当启用和 PVD 孔被禁用时，在 PVD 复位后将设置 RSR[PVDF]标志。如果相应的熔断区域处于擦除状态，则清除该位，否则如果相应的熔断区域中的内容匹配，则开机后从熔断区域的相应位加载。复位后，可通过软件更改。

1 = PVD 复位

0 = PVD 复位

PVDIE — Programmable Voltage Detector Interrupt Enable

PVD IE 用于检查 VCC 是否低于 PVD 阈值

1 = 如果 VCC 低于 PVD 阈值，则将生成一个中断

0 = 如果 VCC 不低于 PVD 阈值，则不会产生任何中断。

PVDE — Programmable Voltage Detector Enable

PVDE 决定是否启用了 PVD。POR 可以清除它。

1 = 启用 PVD

0 = 禁用 PVD

PVDC[1:0] — Programmable Voltage Detector Configuration Value.

表 14-2：可编程电压检测器

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

默认值将从 FUSE 区域加载。如果相应的熔断区域处于擦除状态，则复位后将清除 PVDC[1:0]，否则如果相应的熔断区域的内容匹配，则开机后从熔断区域加载这些位。这些位可以在电源复位后通过软件改变。

14.1.2.2. 客户配置寄存器（CCR）

地址：EFM_BASEADDR + 0x0000_001E

	15	14	13	12	11	10	9	8
读:	RGBEN	0	RTC_INTERFACE_EN	PGMDIS	RSTOUTDIS	CLKOUTDIS	RTC_INTERFACE_LVD_ISOEN	QSPI2EN
写:								
复位:	0	0	0	注 1	注 1	注 1	0	0

	7	6	5	4	3	2	1	0
读:	0	0	ADCCSPOR	PVDCE	0	EOSCSTE	CCRTEST[1:0]	
写:								
复位:	0	0	注 1	注 1	0	注 1	0	0

= 写入没有影响，访问终止，没有传输错误异常。

提示 1：由保险丝区域的值决定

图 14-2：客户配置寄存器（CCR）

RGBEN — RGB Interface Enable Control Bit

RGBEN 决定了 RGB 接口是否启用。

1 = RGB 接口**启用**

0 = RGB 接口禁用

表 14-3：RGB 接口启用控制

RGB 禁用	RGB 启用
EBI_D[15:0]	TFT_RGB[15:0]
EBI_WR#	TFT_VS
EBI_RD#	TFT_HS
EBI_RS	TFT_DE
EBI_CS#	TFT_PCLK

RTC_INTERFACE_EN — This bit determines whether the RTC analog module can be reconfigured or not when PRCSR[Dir] or PRENR[RTC_EN_Dir] is set.

1 = RTC 模拟模块可以重新配置。

0 = RTC 模拟模块无法重新配置。

PGMDIS — Programming Debug Interface Disable Bit

PGMDIS 显示编程调试接口是否被禁用。默认值将从 FUSE 区域加载。如果相应的保险丝区域处于擦除状态，则该位将被清除，否则，如果相应的保险丝区域中的内容被匹配，则将在开机后设置该位。复位后，可通过软件更改。

1 = 编程调试功能被禁用

0 = 编程调试功能已启用

表 14-4：编程调试接口控制

编程启用	编程禁用
PGMCK	RXD2
PGMIO	TXD2

RSTOUTDIS — RSTOUT Disable Bit

结果显示是否禁用 RSTOUT pin 功能并启用 INT1[6]。默认值将从 FUSE 区域加载。如果相应的保险丝区域处于擦除状态，则该位将被清除，否则，如果相应的保险丝区域中的内容被匹配，则将在开机后设置该位。复位后，可通过软件更改。

1 = RSTOUT Pin 功能被禁用

0 = RSTOUT Pin 功能已启用

表 14-5：RSTOUT 禁用控制

RSTOUT 启用	RSTOUT 禁用
RSTOUT	INT1[6]

CLKOUTDIS — CLKOUT Disable Bit

CLKOUTDIS 结果显示 CLKOUT pin 功能是否已禁用，INT1[0]是否已启用。默认值将从 FUSE 区域加载。如果相应的保险丝区域处于擦除状态，则该位将被清除，否则，如果相应的保险丝区域中的内容被匹配，则将在开机后设置该位。复位后，可通过软件更改。

1 = CLKOUT Pin 功能被禁用

0 = CLKOUT Pin 功能已启用

表 14-6：CLKOUT 禁用控制

CLKOUT 启用	CLKOUT 禁用
CLKOUT	INT1[0]

RTC_INTERFACE_LVD_ISOEN — This bit determines whether the RTC analog interface will be isolated or not when Program Voltage Detector event happens.

1 = RTC 模拟接口不隔离。

0 = 当发生低电压时，RTC 模拟接口将被隔离。

QSPI2EN — QSPI2 Interface Enable Control Bit

QSPI2EN 确定 QSPI2 接口是否启用 d。

1 = QSPI2 接口启用

0 = QSPI2 接口被禁用

表 14-7：QSPI2 接口启用控制

QSPI2 禁用	QSPI2 启用
PWM0[0]	QSCS2#
PWM0[1]	QSCK2
PWM0[2]	QSIO2[0]
PWM0[3]	QSIO2[1]
PWM1[0]	QSIO2[2]
PWM1[1]	QSIO2[3]

ADCCSPOR — ADC Channel Setting Bit after power-on

附件显示附件在开机后是否从保险丝区域加载。默认值将从 FUSE 区域加载。如果相应的保险丝区域处于擦除状态，则清除该位，否则，如果相应的保险丝区域中的内容相匹配，则在开机后设置该位。

1 = 附件在通电后从保险丝区域加载

0 = 电源在开机后未从保险丝区域加载

PVDCE — Program Voltage Detector (PVD) Configuration Enable

PVDCE 显示通电后是否从保险丝区域将程序电压检测器配置加载。如果相应的保险丝区域处于擦除状态，则清除该位，否则，如果相应的保险丝区域中的内容相匹配，则在开机后设置该位。

1 = PVD 配置在通电后从保险丝区域加载

0 = PVD 配置开机后未从保险丝区域加载

EOSCSTE — External High Speed Oscillator Stable Time Configuration Enable

该信号显示通电后是否从保险丝区域加载外部高速振荡器稳定时间配置。如果相应的保险丝区域处于擦除状态，则清除该位，否则，如果相应的保险丝区域中的内容相匹配，则在开机后设置该位。

1 = 外部高速振荡器（FXOSC）稳定时间配置在通电后从保险丝区域加载

0 = 外部高速振荡器（FXOSC）通电后未从保险丝区域加载

CCRTEST[1:0] — Write Access Enable Sequence Input

不能更改 CCR 寄存器的可写位，除非写入了正确的序列。正确的序列是：2'b01 → 2'b10 → 2'b11。在用这个序列写入这两个位后，这两个位的值 == 2'b11，然后可以随意改变 CCR 寄存器的可写位。当值等于 2'b11 时，写入 2'b00 可以清除这两个位。写入其他值没有效果，并返回 2'b11。

14.1.2.3. 外部高速振荡器稳定时间配置寄存器（EOSCST）

地址：EFM_BASEADDR + 0x0000_0020

	15	14	13	12	11	10	9	8
读:	EOSCTEST[1:0]		0	0	EOSCST[11:8]			
写:								
复位:	0	0	0	0	注 1	注 1	注 1	注 1

	7	6	5	4	3	2	1	0
读:	EOSCST[7:0]							
写:								
复位:	注 1	注 1	注 1	注 1	注 1	注 1	注 1	注 1

= 写入没有影响，访问终止，没有传输错误异常。

注 1： 由保险丝面积的值确定

图 14-3：外部高速振荡器稳定时间配置寄存器（EOSCST）

EOSCTEST[1:0] — Write Access Enable Sequence Input

不能更改 EOSCST 寄存器的可写位，除非写入了正确的序列。正确的序列是：2'b01 → 2'b10 → 2'b11。在用这个序列写入这两个位后，这两个位的值 == 2'b11，然后可以随意更改 EOSCST 寄存器的可写位。当值等于 2'b11 时，写入 2'b00 可以清除这两个位。写入其他值没有效果，并返回 2'b11。

EOSCST[11:0] — External High Speed Oscillator Stable Time Value.

默认值将从 FUSE 区域加载。如果相应的熔断器区域处于擦除状态，则复位后 EOSCST[11:0]值为 12'h3ff，否则如果相应的保险丝区域中的内容匹配，则开机后从熔断器区域加载。软件可以在复电后更换。在外部振荡器被启用后，它将等待 128KHz 振荡器的 EOSCST[11:0]周期，然后时钟的门将被打开。

14.1.2.4. PLL 锁定时间配置寄存器（PLLOCKCR）

地址：EFM_BASEADDR + 0x0000_0022

	15	14	13	12	11	10	9	8
读:	PLLST[15:14]		PLLST[13:8]					
写:	PLLTEST[1:0] / PLLST[15:14]							
复位:	0	0	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
读:	PLLST[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

图 14-4：PLL 锁定时间配置寄存器（PLLOCKCR）

PLLST[15:0] — PLL Lock Time Value.

启用后，PLL 将等待外部高速振荡器（FXOSC）的 PLLST[15:0]周期，然后时钟的门将被打开。

PLLTEST[1:0] — Write Access Enable Sequence Input

不能对 PLLST 寄存器进行更改，除非写入正确的序列。正确的序列是：2'b01 → 2'b10 → 2'b11。在用这个序列写入这两位后，这两位的值为 == 2'b11，然后就可以随意改变 PLLST 寄存器。当值等于 2'b11 时，任何写入操作都可以清除这两个位。

提示：PLLST[15:14]只有在其值等于 2'b11 时才可写。

14.1.2.5. 复位引脚过滤器启用和值寄存器（RFEVR）

地址：EFM_BASEADDR + 0x0000_0024

	15	14	13	12	11	10	9	8
读:	0	0	RFEVRTEST[1:0]		0	0	0	0
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	RFV[7:0]							
写:								
复位:	注 1	注 1	注 1	注 1	注 1	注 1	注 1	注 1

= 写入没有影响，访问终止，没有传输错误异常。

注 1：由保险丝面积的值确定

图 14-5：复位引脚过滤器启用和值寄存器（RFEVR）

RFEVRTEST [1:0] — Write Access Enable Sequence Input

不能更改 RFEVR 寄存器的可写位，除非写入了正确的序列。正确的序列是：2'b01 → 2'b10 → 2'b11。在用这个序列写入这两个位后，这两个位的值 == 2'b11，然后可以随意更改 RFEVR 寄存器的可写位。当值等于 2'b11 时，写入 2'b00 可以清除这两个位。写入其他值没有效果，并返回 2'b11。

RFV[7:0] — RESET Pin Filter Value

默认值将从 FUSE 区域加载。如果相应的 FUSE 区域处于擦除状态，则复电后 RFV[7:0]值为 8'h0D，否则如果相应的 FUSE 区域的内容匹配，则开机后从 FUSE 区域加载这些位。这些位可以在电源复位后通过软件改变。

14.1.2.6. 可编程电压检测器滤波器启用和值寄存器（PVDFEVR）

地址：EFM_BASEADDR + 0x0000_0026

	15	14	13	12	11	10	9	8
读:	PVDFE	0	PVDFTEST[1:0]		0			
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	PVDFV[7:0]							
写:								
复位:	注 1	注 1	注 1	注 1	注 1	注 1	注 1	注 1

= 写入没有影响，访问终止，没有传输错误异常。

注 1： 由保险丝面积的值确定

图 14-6：可编程电压检测器滤波器启用和值寄存器（PVDFEVR）

PVDFE — Programmable Voltage Detector Filter Enable.

PVDFE 显示程序电压检测器滤波器是否已启用。默认值将从 FUSE 区域加载。如果相应的熔断器区域处于擦除状态，则在开机复位后清除 PVDFE，否则，如果相应的保险丝区域中的内容相匹配，则在开机后从熔断器区域加载这些位。复位后，可通过软件更改。

1 = PVD 筛选器已启用

0 = PVD 筛选器被禁用

PVDFTEST[1:0] — Write Access Enable Sequence Input

不能更改的 PVDFEVR 寄存器的可写位，除非写了正确的序列。正确的序列是：2'b01 → 2'b10 → 2'b11。通过这个序列写入这两个位后，这两个位的值 == 2'b11，然后可以随意更改 PVDFEVR 寄存器的可写位。当值等于 2'b11 时，写入 2'b00 可以清除这两个位。写入其他值没有效果，并返回 2'b11。

PVDFV[7:0] — Programmable Voltage Detector Filter Value

默认值将从 FUSE 区域加载。如果相应的 FUSE 区域处于擦除状态，则复位后 PVDFV[7:0]值为 8'h0，否则如果相应的 FUSE 区域的内容匹配，则在上电后从 FUSE 区域加载这些位。这些位可以在电源复位后改变。

14.1.2.7. 工厂配置寄存器（FCR）

地址：EFM_BASEADDR + 0x0000_002E

	15	14	13	12	11	10	9	8
读:	0	0	RTCTE	TMDIS	RESETFTE	LDO1P2TE	VREF1P2TE	PVDFTE
写:								
复位:	0	0	注 1	注 1	注 1	注 1	注 1	注 1

	7	6	5	4	3	2	1	0
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

注 1： 由保险丝面积的值确定

图 14-7：工厂配置寄存器（FCR）

RTCTE — RTC Bias/Cap rimming Enable

RTCT E 显示是否修剪了 RTC 偏差或电容。默认值将从 FUSE 区域加载。如果相应的保险丝区域处于擦除状态，则清除该位，否则，如果相应的保险丝区域中的内容相匹配，则在开机后设置该位。

1 = RTC 偏差或电容被修剪

0 = RTC 偏差或电容没有被修剪

TMDIS — Test Mode Disable Bit

TMDIS 显示是否禁用了测试模式（STB 除外）。默认值将从 FUSE 区域加载。如果相应的保险丝区域处于擦除状态，则清除该位，否则，如果相应的保险丝区域中的内容相匹配，则在开机后设置该位。

1 =测试模式禁用

0 = 测试模式启用

RESETFTE — Reset Pin Filter Trimming Enable

复位显示是否修剪复位引脚过滤器。默认值将从 FUSE 区域加载。如果相应的保险丝区域处于擦除状态，则清除该位，否则，如果相应的保险丝区域中的内容相匹配，则在开机后设置该位。

1 = 复位引脚过滤器被修剪

0 = 复位引脚过滤器未被修剪

LDO1P2TE — LDO1P2 Trimming Enable

LDO1P2TE 显示 LDO1P2 是否被修剪。默认值将从 FUSE 区域加载。如果相应的保险丝区域处于擦除状态，则清除该位，否则，如果相应的保险丝区域中的内容相匹配，则在开机后设置该位。

1 = LDO1P2 被修剪

0 = LDO1P2 未被修剪

VREF1P2TE — VREF1P2 Module Trimming Configuration Register

VREF1P2TE 显示 VREF1P2 模块是否被修剪。默认值将从 FUSE 区域加载。如果相应的保险丝区域处于擦除状态，则清除该位，否则，如果相应的保险丝区域中的内容相匹配，则在开机后设置该位。

1 = 内部 VREF1P2 模块被修剪

0 = 内部 VREF1P2 模块未被修剪

PVDFTE — PVD Filter Trimming Enable

PVDFTE 显示 PVD 过滤器是否被修剪。默认值将从 FUSE 区域加载。如果相应的保险丝区域处于擦除状态，则清除该位，否则，如果相应的保险丝区域中的内容相匹配，则在开机后设置该位。

1 = PVD 过滤器已被修剪

0 = PVD 过滤器未被修剪

14.1.2.8. 信道禁用配置寄存器（附件）

地址：EFM_BASEADDR + 0x0000_0034

	15	14	13	12	11	10	9	8
读:	ADCCDISTEST[1:0]		0					
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	ADCCDIS[7:0]							
写:								
复位:	注 1	注 1	注 1	注 1	注 1	注 1	注 1	注 1

= 写入没有影响，访问终止，没有传输错误异常。

注 1： 由保险丝面积的值确定

图 14-8：ADC 信道禁用配置寄存器（ADCCDISR）

ADCCDISTEST[1:0] — Write Access Enable Sequence Input

除非将正确的序列写入 addctist，否则不能更改可写位[7:0]。正确的序列是：2'b01 → 2'b10 → 2'b11。用这个序列写入这两个位后，这两个位的值 == 2'b11，然后可以随意更改测试寄存器的可写位。当值等于 2'b11 时，写入 2'b00 可以清除这两个位。写入其他值没有效果，并返回 2'b11。

ADCCDIS[7:0] — ADC Channel Disable Configuration

确定 AIN[7:0]是作为 ADC 通道还是作为 GPIO 通道。如果相应的 FUSE 区域处于清除状态，则清除这些位，ADC 信道功能有效，否则，如果相应的 FUSE 区域中的内容相匹配，则从 FUSE 区域加载 ADCCDIS[7:0]。这些位可以在电源复位后改变。

表 14-8：ADC 通道禁用控制

ADC 通道禁用	ADC 通道启用
INT0[0]	AIN[0]
INT0[1]	AIN[1]
INT0[2]	AIN[2]
INT0[3]	AIN[3]
INT0[4]	AIN[4]
INT0[5]	AIN[5]
INT0[6]	AIN[6]
INT0[7]	AIN[7]

14.1.2.9. VREF1P2 修剪配置寄存器（VREFTCR）

地址：EFM_BASEADDR + 0x0000_0036

	7	6	5	4	3	2	1	0
读: 写:	VREFTCTEST[1:0]		VREFPD	VREFTRIM[4:0]				
复位:	0	0	注 1	注 1	注 1	注 1	注 1	注 1

注 1：由保险丝面积的值确定

图 14-9：VREF 修剪配置寄存器（VREFTCR）

VREFTCTEST[1:0] — WSFTCR Write Access Sequence In

除非写入正确的数据序列，否则不能更改虚拟测试寄存器的可写入位。正确的序列是：2'b01 → 2'b10 → 2'b11。在用这个序列写入这两个位后，这两个位的值 == 2'b11，然后可以随意更改测试寄存器的可写位。当值等于 2'b11 时，写入 2'b00 可以清除这两个位。写入其他值没有效果，并返回 2'b11。

VREFPD — Internal VREF1P2 Module Power Down. The setup time is 10us after Internal VREF1P2 Module is turn on.

默认值将从 FUSE 区域加载。如果相应的保险丝区域处于擦除状态，则在开机复位后设置 VREFPD，否则，如果相应的保险丝区域中的内容相匹配，则在开机后从保险丝区域加载这些位。复位后，可通过软件更改。

1 = 内部 VREF1P2 模块断电

0 = 内部 VREF1P2 模块上电

VREFTRIM[4:0] — VREF1P2 Module Trimming Value

默认值将从 FUSE 区域加载。如果相应的保险丝区域处于擦除状态，则开机复位后的 VREFTRIM[4:0]为 5'h18，否则，如果相应的保险丝区域中的内容相匹配，则这些位在开机后从保险丝区域加载。这些位可以在电源复位后通过软件改变。

14.1.2.10. LDO1P2 修剪配置寄存器（LDOTCR）

地址：EFM_BASEADDR + 0x0000_0037

	7	6	5	4	3	2	1	0
读:	LDO1P2TC[7:6]		LDO1P2TC[5:0]					
写:	LDO1P2TEST[1:0] / LDO1P2TC[7:6]							
复位:	注 1	注 1	注 1	注 1	注 1	注 1	注 1	注 1

注 1：由保险丝面积的值确定

图 14-10：LDO1P2 修剪配置寄存器（LDOTCR）

LDO1P2TC[7] — LDO over-current limit function control bit

默认值将从 FUSE 区域加载。如果相应的 FUSE 区域处于擦除状态，则开机复位后 LDO1P2TC[7]为 1'b0，否则如果相应的 FUSE 区域的内容匹配，则在开机后从 FUSE 区域加载。这个位可以在电源复位后改变。

1 = LDO 过电流限制功能将被关闭。

0 = LDO 过流限制功能。

LDO1P2TC[6] — LDO1P2 power down control bit

默认值将从 FUSE 区域加载。如果相应的 FUSE 区域处于擦除状态，则开机复位后 LDO1P2TC[6]为 1'b0，否则如果相应的 FUSE 区域的内容匹配，则在开机后从 FUSE 区域加载。这个位可以在电源复位后改变。

1 = LDO1P2 将断电。

0 = LDO1P2 将通电。

LDO1P2TC[5:0] — LDO1P2 Trimming Value

默认值将从 FUSE 区域加载。如果相应的 FUSE 区域处于擦除状态，则开机复位后 LDO1P2TC[5:0]为 6'h00，否则如果相应的 FUSE 区域中的内容相匹配，则在开机后从 FUSE 区域加载这些位。这些位可以在电源复位后通过软件改变。

LDO1P2TEST[1:0] — LDO1P2TC Write Access Sequence In

不可写的 LDO1P2TC 寄存器不能改变，除非写入正确的数据序列。正确的序列是：2'b01 → 2'b10 → 2'b11。用这个序列写入这两个位后，这两个位的值 == 2'b11，然后可以随意改变 LDO1P2TC 寄存器的可写位。当值等于 2'b11 时，任何写入操作都可以清除这两个位。

提示：只有当 LDO1P2TEST[1:0]的值等于 2'b11 时，LDO1P2TC[7:6]才能写入。

14.1.2.11. RTC 修剪配置寄存器（RTCTCR）

地址：EFM_BASEADDR + 0x0000_003A

	15	14	13	12	11	10	9	8
读:	RTCTCTEST[1:0]		0				RTC_TEST_MODE	Trim_Load
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	RTC_BIAS_TRIM[3:0]				RTC_CAP_TRIM[3:0]			
写:								
复位:	注 1	注 1	注 1	注 1	注 1	注 1	注 1	注 1

= 写入没有影响，访问终止，没有传输错误异常。

注 1：由保险丝面积的值确定

图 14-11：RTC 修剪配置寄存器（RTCTCR）

RTCTCTEST[1:0] — RTCTC Write Access Sequence In

RTCTC 寄存器的可写位不能更改，除非写入了正确的数据序列。正确的序列是：2'b01 → 2'b10 → 2'b11。在用这个序列写入这两个位后，这两个位的值 == 2'b11，然后可以随意更改 RTCTC 寄存器的可写位。当值等于 2'b11 时，写入 2'b00 可以清除这两个位。写入其他值没有效果，并返回 2'b11。

RTC_TEST_MODE — RTC Test Mode enable.

1 = RTC 测试模式已启用

0 = RTC 测试模式被禁用

Trim_Load — Low to high edge will latch RTC_BIAS_TRIM and RTC_CAP_TRIM value into RTC analog module.

RTC_BIAS_TRIM[3:0] — RTC Oscillator Bias Trimming configuration bits.

默认值将从 FUSE 区域加载。如果相应的 FUSE 区域处于擦除状态，则复位后 RTC_BIAS_TRIM[3:0]为 4'b0111，否则如果相应的 FUSE 区域的内容匹配，则开机后从 FUSE 区域加载。这些位可以在电源复位后通过软件改变。

RTC_CAP_TRIM[3:0] — RTC Oscillator Load Capacitance C1 and C2 Trimming configuration bits.

默认值将从 FUSE 区域加载。如果相应的 FUSE 区域处于擦除状态，则在电源复位后 RTC_BIAS_TRIM[3:0]为 4'b1100，否则如果相应的 FUSE 区域中的内容匹配，则在开位后从 FUSE 区域加载。这些位可以在电源复位后通过软件改变。

15. 编程中断定时器（PIT）

15.1. 功能介绍

LT7589 的可编程中断计时器（PIT）是一个 16 位计时器，提供精确的中断在定期间隔与最小的处理器干预。计时器可以从模量锁存器中写入的值开始计数，也可以是一个自由运行的下行计数器。

15.2. 方框方块图

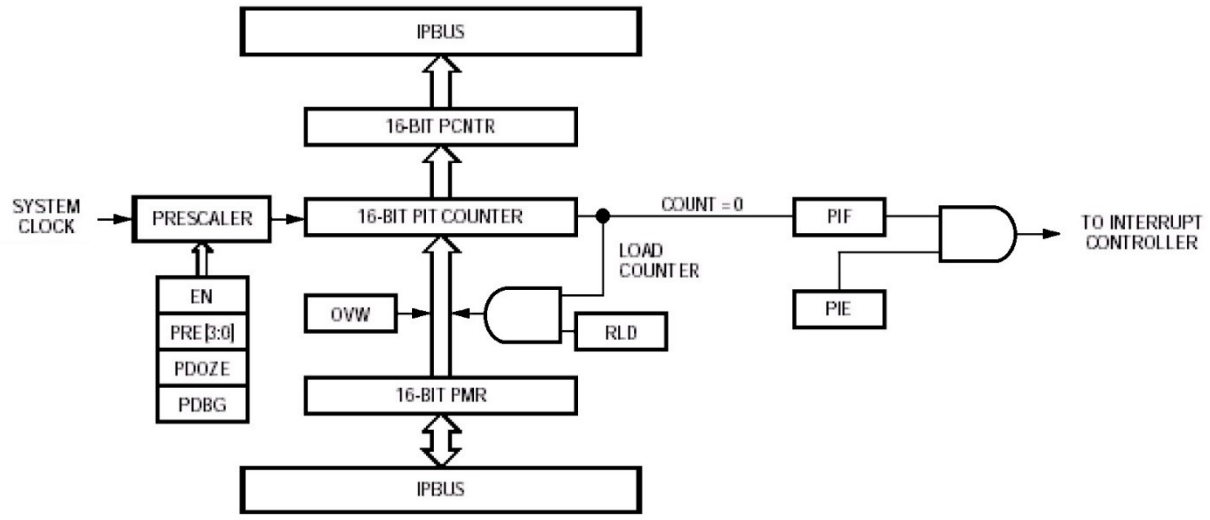


图 15-1：PIT 方块图

15.3. 操作模式

本章节介绍了三种低功耗模式和调试模式。

15.3.1. 等待模式

在等待模式下，PIT 模块继续正常运行，并且可以通过产生一个中断请求来配置为退出低功耗模式。

15.3.2. Doze 模式

在 PIT 控制和状态寄存器（PCSR）中设置 PDOZE 位的启动模式下，PIT 模块操作停止。在 PDOZE 位清除的睡觉模式下，睡觉模式不影响 PIT 操作。退出休眠模式后，PIT 操作从进入休眠模式之前的状态继续。

15.3.3. 停止模式

在停止模式下，系统时钟缺失，PIT 模块操作停止。

15.3.4. 调试模块

在 PCSR 模式中设置 PDBG 位的调试模式下，PIT 模块操作停止。在清除 PDBG 位的调试模式下，调试模式不会影响 PIT 操作。退出调试模式后，PIT 操作将从进入调试模式之前的状态继续进行，但在调试模式下进行的任何更新都将保留。

15.4. 信号说明

PIT 模块没有芯片外信号。

15.5. 内存映射和寄存器

本章节描述了 PIT 的内存映射和寄存器结构。PIT 模块内存映射如下表所示，PIT0 的基本地址为 0x4004_0000，PIT1 为 0x4005_0000，PIT2 为 0x4006_0000，PIT3 为 0x4007_0000。

15.5.1. 内存映射

见下表以获取内存映射的描述。

该设备有四个可编程的中断计时器。

表 15-1：可编程中断定时器模块内存映射

PITn 地址	Bits[15:0]	访问权限(1)
0x0	PIT Modulus Register (PMR)	S
0x2	PIT Control and Status Register (PCSR)	S
0x4	Unimplemented(2)	—
0x6	PIT Count Register (PCNTR)	S/U

提示：

(1) 仅限 S = CPU 监控模式访问。S/U = CPU 监控器或用户模式访问。在用户模式下只访问管理员地址没有影响，并导致周期终止传输错误。

(2) 对未实现的地址位置的访问没有影响，并会导致周期终止传输错误。

15.5.2. 寄存器描述

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。参考第 15.5.2.2 章节。
- PIT 模量寄存器（PMR）确定定时器模量重新加载值。参考第 15.5.2.1 章节。
- PIT 计数寄存器（PCNTR）提供了对计数器值的可见性。参考第 15.5.2.3 章节。

15.5.2.1. PIT 模量寄存器（PMR）

16 位读/写 PIT 模量寄存器（PMR）包含当计数达到 0x0000 和 RLD 位被设置时加载到 PIT 计数器的计时器模量值。

当设置了 OVW 位时，PMR 是透明的，并且写入 PMR 的值会立即加载到 PIT 计数器中。每当新值加载到 PIT 计数器以及复位计数器期间，都会复位预调节计数器。读取 PMR 将返回用模量锁存器写入的值。将 PMR 初始化为 0xFFFF。

地址：PITn_BASEADDR + 0x0000_0000

	15	14	13	12	11	10	9	8
读: 写:	PM[15:8]							
复位:	1	1	1	1	1	1	1	1

	3	6	5	4	3	2	1	0
读: 写:	PM[7:0]							
复位:	1	1	1	1	1	1	1	1

图 15-2：PIT Modulus 寄存器（PMR）

15.5.2.2. PIT 控制和状态寄存器（PCSR）

地址：PITn_BASEADDR + 0x0000_0002

	15	14	13	12	11	10	9	8
读:	0				PRE[3:0]			
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读: 写:	PDMAE	PDOZE	PDBG	OVW	PIE	PIF	RLD	EN
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 15-3：PIT 控制和状态寄存器（PCSR）

PRE[3:0] — Prescaler Bits

读写准备位选择系统时钟除数来生成 PIT 时钟，如下表所示。

要准确预测下一次计数的时间，仅在清除启用位（EN）时才更改 PRE[3:0]位。更改 PRE，将复位前置调节器计数器。系统复位和将新值加载到计数器也会复位预压器计数器。将 EN 位并写入设置为 PRE[3:0]可以在相同的写入周期中完成。清除 EN 位停止预调器计数器。

表 15-2：Prescaler Select Encoding

PRE[3:0]	IPS Clock Divisor
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 Enable Control Bit

读/写 PDMAE 位控制是指当 PIT 计数器达到 0x0000 时是否会生成 DMA 请求。

当 PIT 计数器达到 0x0000 时，将生成 1 = DMA 请求。

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

PDOZE — Doze Mode Bit

读/写 PDOZE 位控制休眠模式下的功能。复位清除 PDOZE。

1 = PIT 功能在休眠模式下停止

0 =PIT 功能休眠模式下不受影响

当退出休眠模式时，计时器操作从进入休眠模式之前的状态继续。

PDBG — Debug Mode Bit

读/写 PDBG 位在调试模式下控制 PIT 的功能。复位清除 PDBG。

1 = 在调试模式下，PIT 功能停止

0 = 在调试模式下，PIT 功能不受影响

在调试模式下，寄存器的读写访问功能正常。退出调试模式后，计时器操作将从进入调试模式之前的状态继续进行，但在调试模式下进行的任何更新都将保留。

提示： 在调试模式中将 PDBG 位从 1 改为 0 将启动 PIT 计时器。同样，在调试模式期间将 PDBG 位从 0 更改为 1 也会停止 PIT 计时器。

OVW — Overwrite Bit

读/写 OVW 位允许写入 PMR 以立即覆盖 PIT 计数器中的值。

1 = 写入 PMR 立即替换 PIT 计数器中的值。

0 = 当计数达到 0x0000 时，PMR 中的值替换 PIT 计数器中的值。

PIE — PIT Interrupt Enable Bit

读/写 PIE 位允许 PIF 标志生成中断请求。

1 = PIF 中断请求启用

0 = PIF 中断请求禁用

PIF — PIT Interrupt Flag

当 PIT 计数器达到 0x0000 时，将设置读/写 PIF 标志。通过写入 PIF 或写入 PMR 或 DMA 请求来清除 PIF。写入 0 没有效果。复位清除 PIF。

1 = PIT 计数已达到 0x0000。

0 = PIT 计数未达到 0x0000。

RLD — Reload Bit

读/写 RLD 位允许在计数达到 0x0000 时将 PMR 的值加载到 PIT 计数器中。

1 = 计数器在计算数为 0x0000 时从 PMR 重新加载

0 = 计数器在计数为 0x0000 时卷转到 0x FFFF

EN — PIT Enable Bit

读写 EN 位启用 PIT 操作。禁用该 PIT 时，计数器和预调节器将处于停止状态。

1 = PIT 启用

0 = PIT 禁用

15.5.2.3. PIT 计数寄存器（PCNTR）

16 位，只读的 PIT 控制寄存器（PCNTR）包含计数器值。

用两个 8 位读取读取 16 位计数器并不能保证是一致的。

写入 PCNTR 没有效果，写入周期正常终止。

地址：PITn_BASEADDR + 0x0000_0006

	15	14	13	12	11	10	9	8
读:	PC[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	PC[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 15-4：PIT 计数寄存器（PCNTR）

15.6. 功能描述

本章节介绍了 PIT 的功能操作。

15.6.1. 设置和忘记计时器操作

当设置了 PCSR 寄存器中的 RLD 位时，将选择此操作模式。

当 PIT 计数器达到计数数 0x0000 时，在 PCSR 中设置 PIF 标志。模量锁存器中的值被加载到计数器中，计数器开始向 0x0000 递减。如果 PIE 位设置为 PCSR 位，则 PIF 标志会向 CPU 发出中断请求。

当 OVW 位设置在 PCSR 中时，计数器可以通过写入 PMR 来直接初始化，而不必等待计数达到 0x0000。

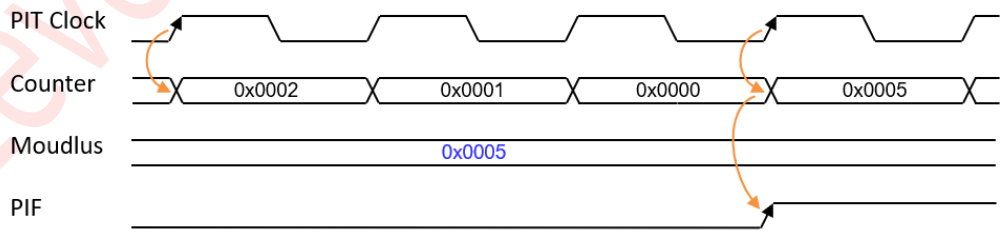


图 15-5：计数器从 Modulus Latch 重新加载

15.6.2. 自由运行计时器操作

当清除了 PCSR 中的 RLD 位时，将选择此操作模式。在此模式下，计数器从 0x0000 滚动到 0x FFFF，而不需要从模量锁存器重新加载，并继续衰减。

当计数器达到计数数 0x0000 时，在 PCSR 中设置 PIF 标志。如果 PIE 位设置为 PCSR 位，则 PIF 标志会向 CPU 发出中断请求。

当 OVW 位设置在 PCSR 中时，计数器可以通过写入 PMR 来直接初始化，而不必等待计数达到 0x0000。

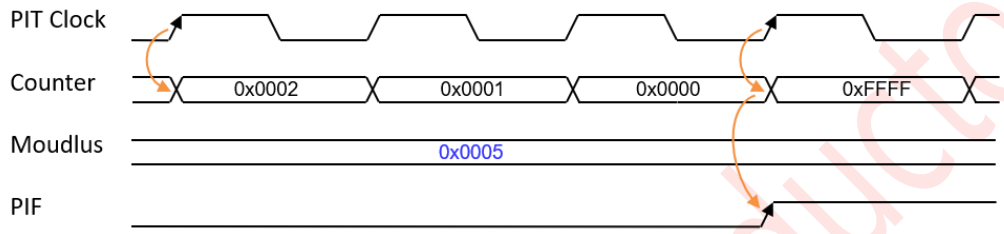


图 15-6：在自由运行模式下的计数器

15.6.3. 超时规格

16 位的 PIT 计数器和预调用器支持不同的超时时间。预计算器按 PCSR 中的 PRE[3:0]位的选择来划分系统时钟。PMR 中的 PM[15:0]位将选择超时时间。

Timeout Period = PRE[3:0] x (PM[15:0] + 1) clocks

15.7. 中断操作

下表列出了由 PIT 生成的中断请求。

表 15-3：PIT 中断请求

中断请求	标志	启用位
Timeout	PIF	PIE

16. 看门狗计时器（WDT）

16.1. 功能介绍

LT7589 的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恢复或在操作运行比预期更长时给予中断。看门狗定时器有一个自由递减计数器（看门狗计数器），在下溢时产生重置或中断。为防止重置，软件必须通过维修看门狗定期重新启动倒计时。

16.2. 操作模式

本章节介绍了看门狗定时器在低功率模式和调试模式下的操作。

16.2.1. 等待模式

在看门狗控制寄存器（WCR）中设置了等待位的等待模式下，看门狗计时器操作停止。在等待位清除的等待模式下，看门狗计时器继续正常工作。

16.2.2. Doze 模式

在 WCR 中设置 doze 位的休眠模式下，看门狗计时器模块的操作停止。在睡觉模式下，看门狗计时器继续正常工作。

16.2.3. 停止模式

在停止模式下，将停止位设置为 WCR，看门狗操作在停止模式下停止。当停止模式退出时，看门狗操作从它在进入停止模式之前的状态继续操作。在停止模式与停止位清除，看门狗计时器继续正常工作。

16.2.4. 调试模式

在 WCR 中设置 DBG 位的调试模式下，看门狗计时器模块的操作停止。在清除的 DBG 位的调试模式下，看门狗计时器继续正常运行。当调试模式退出时，看门狗计时器操作将从进入调试模式之前的状态继续进行，但在调试模式下进行的任何更新都将保留。

16.3. 方块图

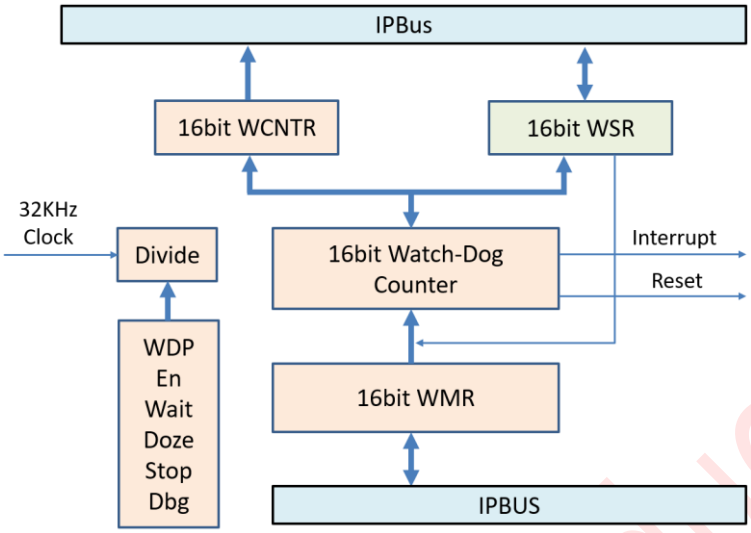


图 16-1：看门狗计时器方块图

16.4. 信号说明

看门器定时器模块没有芯片外信号。

16.5. 内存映射和寄存器

本章节描述了看门狗计时器的内存映射和寄存器。WDT 模块的基本地址为 0x4013_0000。

16.5.1. 内存映射

参考下表为看门狗记忆地图的概述。

表 16-1：看门狗计时器模块记忆地图

偏移地址	Bits[15:0]	访问权限(1)
0x0000	Watchdog Modulus Register (WMR)	S
0x0002	Watchdog Control Register (WCR)	S
0x0004	Watchdog Service Register (WSR)	S/U
0x0006	Watchdog Count Register (WCNTR)	S/U

提示(1) :仅限 S = CPU 监控模式访问。S/U = CPU 监控或用户模式访问。在用户模式下只访问管理员地址没有影响，并导致周期终止传输错误。

16.5.2. 寄存器说明

看门狗计时器编程模型由以下几种寄存器组成：

- 看门狗控制寄存器（WCR）配置看门狗计时器操作。参考第 16.5.2.2 章节
- 看门器模量寄存器（WMR）确定定时器模量重新加载值。参考第 16.5.2.1 錯誤! 找不到參照來源。章节
- 看门狗计数寄存器（WCNTR）提供了对看门狗计数器值的可见性。参考第 16.5.2.4 章节
- 看门狗服务寄存器（WSR）需要一个服务序列，以防止复位。只读的 WC 字段反映了电流看门狗计数器中的值。读取 16 位 WCNTR8 位的读取不能保证返回一个相干的值。写入到 WCNTR 没有作用，写入周期正常终止。这个寄存器是为看门狗工作域，因此读取值可能不是稳定的，请连续不断地阅读。

16.5.2.1. 看门狗模块寄存器（WMR）

地址：WDT_BASEADDR + 0x0000_0000

	15	14	13	12	11	10	9	8
读: 写:	WM[15:8]							
复位:	1	1	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
读: 写:	WM[7:0]							
复位:	1	1	1	1	1	1	1	1

图 16-2：看门狗模量寄存器（WMR）

WM[15:0] — Watchdog Modulus Field

WM 字段包含由服务序列重新加载到看门狗计数器中的模量。写入 WMR 后，会立即将新的模量值加载到看门狗计数器中。该新值也将在下一次和所有后续的重新加载时使用。

读取 WMR 返回模量寄存器中的值。将 WM[15:0]字段初始化为 0x FFFF。

16.5.2.2. 看门狗控制寄存器（WCR）

16 位读/写看门狗控制寄存器（WCR）配置看门狗计时器操作。

地址：WDT_BASEADDR + 0x0000_0002

	15	14	13	12	11	10	9	8
读:	0				WAIT	DOZE	STOP	DBG
写:								
复位:	0	0	0	0	1	1	1	1

	7	6	5	4	3	2	1	0
读:	IS	WDP[2:0]			IF	IE	0	EN
写:							CU	
复位:	0	1	1	1	0	1	0	1

= 写入没有影响，访问终止，没有传输错误异常。

图 16-3：看门狗控制寄存器（WCR）

WAIT — Wait Mode Bit

等待位控制在等待模式下的看门狗计时器的功能。复位设置等待。

1 = 看门狗计时器在等待模式下停止

0 = 看门狗计时器在等待模式下不受影响

DOZE — Doze Mode Bit

DOZE 位控制起睡模式下看门狗定时器的功能。复位设置大小。

1 = 看门狗计时器在休眠模式下停止

0 = 手表计时器在休眠模式下不受影响

STOP — STOP Mode Bit

停止位控制在停止模式下的看门狗计时器的功能。复位设置停止。

1 = 看门狗计时器在停止模式下已停止

0 = 看门狗计时器在停止模式下不受影响

DBG — Debug Mode Bit

DBG 位在调试模式下控制看门器计时器的功能。在调试模式下，可以正常地写入和读取看门狗定时器寄存器。退出调试模式后，计时器操作将从进入调试模式之前的状态继续进行，但在调试模式下进行的任何更新都将保留。

1 = 看门狗计时器在调试模式下停止

0 = 看门狗计时器在调试模式下不受影响

提示：

在调试模式下，将 DBG 位从 1 更改为 0 将启动看门狗计时器。

在调试模式下，将 DBG 位从 0 更改为 1 将停止看门狗计时器。

IS — Watchdog Clock Domain Interrupt Status Bit

这个位是只读的，如果这个位是 1‘b1，则不清除看门狗时钟域的状态，因此如果 CPU 想要休眠或停止，它将再次被唤醒。因此，在 CPU 想要休眠或停止之前，请先检查这个位。

WDP[2:0] — Watchdog Timer Prescaler

WDP[2:0]位决定了看门狗计时器运行时的看门狗计时器预压。不同的预调整值及其相应的超时时间见下表看门狗计时器。

表 16-2：看门狗计时器 Prescaler

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

IF — Watchdog Interrupt Flag Bit

写一个到这一点将清除的标志。

IE — Watchdog Interrupt Enable Bit

IE 位启用看门狗计时器中断模式。一旦产生了中断，并且 EN 位为 1，该位将被自动清除。

1 = 看门狗定时器中断模式已启用

0 = 看门狗计时器中断模式被禁用

CU — Watchdog Change Update Bit

写一个 CU 位将把 WDP[2:0]和 WMR 更新到工作锁存器。

EN — Watchdog Enable Bit

EN 位启用了看门狗计时器。

1 = 看门狗计时器已启用

0 = 看门狗计时器被禁用

16.5.2.3. 看门狗服务寄存器（WSR）

当看门狗计时器启用时，在看门狗计数器超时之前，将 0x5555 和 0x AAAA 写入看门器服务寄存器（WSR），以防止复位。如果 WSR 在超时前没有得到维修，看门狗计时器会向复位控制器或中断控制器模块发送一个信号，并产生一个系统复位或中断。

两个写入操作必须按照超时前列出的顺序发生，但是在两个写入操作之间可以执行任意数量的指令。但是，将除 0x5555 或 0xAAAA 以外的任何值写入 WSR 将复位服务序列，需要写入两个值以防止看门狗计时器导致复位。

地址：WDT_BASEADDR + 0x0000_0004

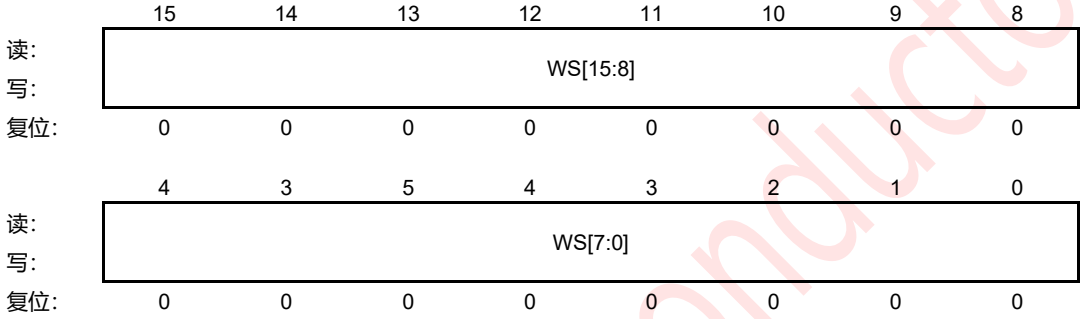


图 16-4：看门狗服务寄存器（WSR）

16.5.2.4. 看门狗计数寄存器（WCNTR）

地址：WDT_BASEADDR + 0x0000_0006

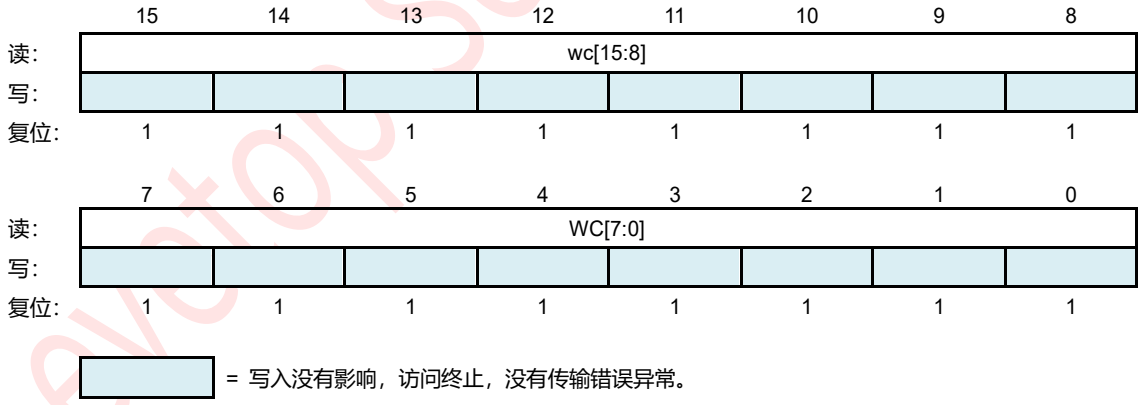


图 16-5：看门机构统计寄存器（WCNTR）

WC[15:0] — Watchdog Count Field

只读 WC[15:0]字段反映了看门狗计数器中的当前值。用两个 8 位读取读取 16 位 WCNTR 并不能保证返回一个相干值。写入 WCNTR 没有效果，写入周期正常终止。此寄存器为看门狗工作域，读取值可能不稳定，请连续读取。

17. 时钟控制器（RTC）

17.1. 功能介绍

LT7589 的实时控制器控制着一个具有 RTC 功能的综合 PMU。它可以重新配置 RTC 计数器，设置报警，并产生时间/报警中断。

17.2. RTC 特点

本模块的主要特点是：

- 重新配置 RTC 计数器，并读取来自秒、分钟、小时和天的计数器
- 支持报警设置
- 中断源：秒、分钟、小时、日中断、可编程报警中断、1KHz / 32KHz 周期性中断。

17.3. 测试模式

在测试模式下，RTC 可以通过 CPU 或芯片针脚进行配置，RTC 的状态和时钟信号可以通过芯片针脚进行检查。

17.4. 方块图

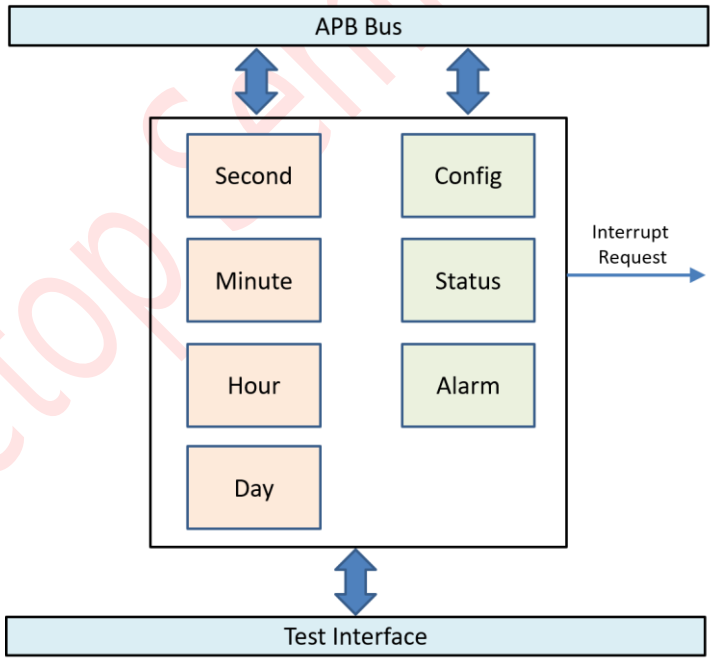


图 17-1：RTC 方块图

17.5. 应用电路

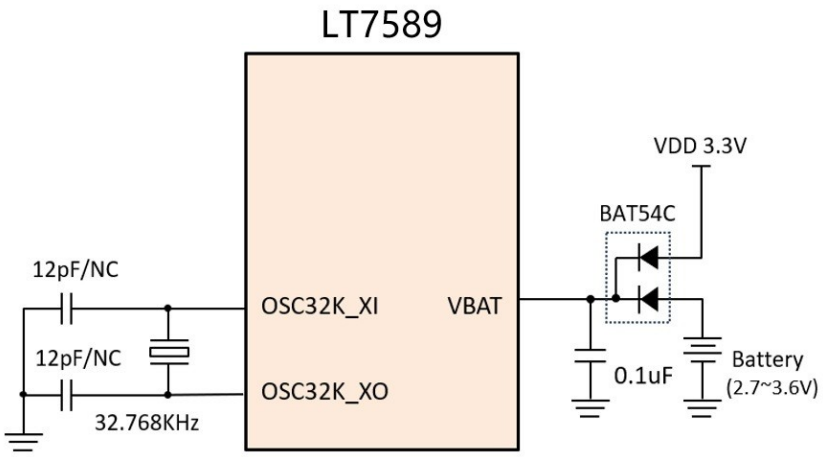


图 17-2：RTC 应用电路

18. 边缘端口模块（EPORT）

18.1. 功能介绍

LT7589 有三个边缘端口模块（EPORT）。每个 EPORT 都有 8 个相应的外部中断引脚。每个引脚可以单独配置为低电平敏感中断引脚、边缘检测中断引脚（上升边、下降边或两者）或通用输入/输出/（I/O）引脚。见下图。

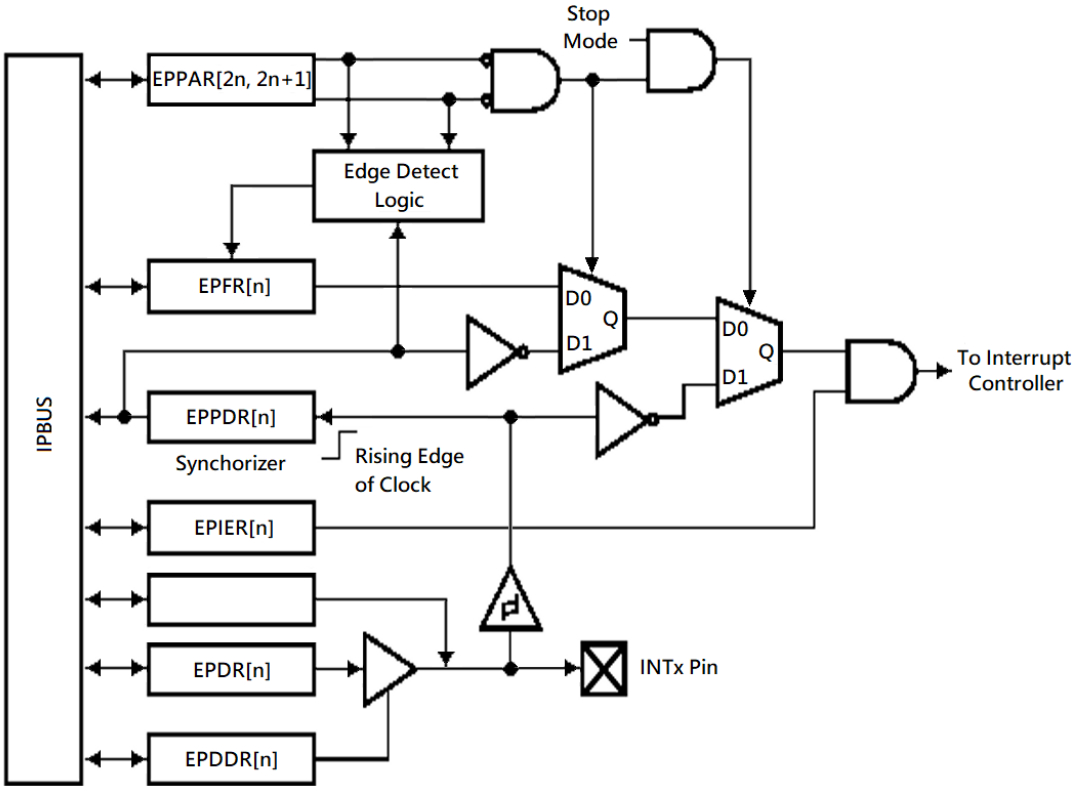


图 18-1：EPORT 方块图

18.2. 低功耗模式运行

本章节介绍了 EPORT 模块在低功耗模式下的操作情况。

18.2.1. 等待和多兹模式

在等待和睡眠模式下，EPORT 模块继续正常运行，并且可以被配置为通过在选定的边缘或在外部引脚上的低电平上产生中断请求来退出低功耗模式。

18.2.2. 停止模式

在停止模式下，没有可用来执行边缘检测功能。只有级别检测逻辑是激活的（如果配置），以允许外部中断插脚上的任何低级别生成中断（如果启用）以退出停止模式。

提示：由于没有时钟，电平检测逻辑绕过输入引脚同步器。

18.3. 中断/通用输入引脚说明

所有复位引脚时默认为通用输入引脚。当从 EPORT 引脚数据寄存器(EPPDR)读取时,引脚值被同步到 CLKOUT 的上升边缘。在边缘/级别检测逻辑中使用的值也被同步到 CLKOUT 的上升边缘。这些引脚使用施密特触发输入缓冲区，该缓冲区内置迟滞，旨在减少缓慢上升和下降输入信号产生错误边缘触发中断的概率。

18.4. 内存映射和寄存器

EPORT 模块内存映射情况见下表 EPORT0 基本地址为 0x400F_0000，EPORT1 为 0x4010_0000，EPORT2 为 0x401D_0000。本章节介绍了内存映射和寄存器结构。

18.4.1. 内存映射

有关 EPORT 内存映射的描述，请参考下表。

表 18-1：模块内存映射

偏移地址	Bits[15:8]	Bits[7:0]	访问权限(1)
0x0000	EPORT Data Direction Register (EPDDR)	EPORT Interrupt Enable Register (EPIER)	S
0x0002	EPORT Pin Assignment Register (EPPAR)		S
0x0004	EPORT Flag Register (EPFR)	EPORT Pin Pull-up enable Register (EPPUE)	S/U
0x0006	EPORT Data Register (EPDR)	EPORT Pin Data Register (EPPDR)	S/U
0x0008	EPORT Digital Filter Control Register (EPFC)	EPORT Bit Set Register (EPBSR)	S
0x000A	EPORT Level Polarity Register (EPLPR)	EPORT Open Drain Register (EPODE)	S
0x000C	保留		S
0x000E	EPORT Bit Clear Register (EPBCR)	保留	S

注意(1)：仅限 S = CPU 监控模式访问。S/U = CPU 监控或用户模式访问。在用户模式下只访问管理员地址没有影响，并导致周期终止传输错误。

18.4.2. 寄存器说明

EPORT 编程模型由以下寄存器组成：

- 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）分别控制每一个引脚的方向。
- EPORT 中断启用寄存器（EPIER）分别为每个引脚启用中断请求。
- EPORT 数据寄存器（EPDR）保存要被驱动到引脚的数据。
- EPORT 引脚数据寄存器（EPPDR）反映了引脚的当前状态。
- EPORT 标志寄存器（EPFR）单独锁存 EPORT 边缘事件。
- EPORT 引脚向上拉拉启用寄存器（EPPUE）单独控制每个引脚的拉向上。
- EPORT 电平极性寄存器（EPLPR）控制对电平敏感的每个引脚的电平极性。
- EPORT 打开排水启用寄存器（EPODE）分别控制单独输出的每个引脚的打开排水。
- EPORT 数字滤波器控制寄存器（EPFC）启用滤波器，并控制将被滤波的输入脉冲的宽度。

18.4.2.1. 边缘端口中断启用寄存器（EPIER）

地址：EPORTn_BASEADDR + 0x0000_0000

	7	6	5	4	3	2	1	0
读： 写：	EPIE[7:0]							
复位：	0	0	0	0	0	0	0	0

图 18-2：EPORT 端口中断启用寄存器（EPIER）

EPIE[7:0] — Edge Port Interrupt Enable Bits

读/写 EPIE[7:0]位启用 EPORT 中断请求。如果在 EPIER 中设置了一个位，EPORT 会在以下时候生成一个中断请求：

- EPORT 标志寄存器（EPFR）中的相应位被设置或以后被设置，或
- 相应的引脚电平较低，且引脚配置为电平敏感操作

在 EPIER 中清除一点将否定来自相应的 EPORT 引脚的任何中断请求。复位清除 EPIE[7:0]。

1 = 启用了来自相应 EPORT 引脚的中断请求

0 = 来自相应 EPORT 引脚的中断请求被禁用

18.4.2.2. Eport 数据方向寄存器（EPDDR）

地址：EPORTn_BASEADDR + 0x0000_0001

	7	6	5	4	3	2	1	0
读: 写:	EPDD[7:0]							
复位:	0	0	0	0	0	0	0	0

图 18-3：EPORT 数据方向寄存器（EPDDR）

EPDD[7:0] — Edge Port Data Direction Bits

在 EPDDR 中设置任何位都会将相应的引脚配置为输出。清除 EPDDR 中的任何位，都会将相应的引脚配置为输入。引脚的方向独立于水平/边缘检测配置。复位清除 EPDD[7:0]。

要使用 EPORT 引脚作为外部中断请求源，其在 EPDDR 中的相应位必须被清除。当 EPDDR 选择输出时，软件可以通过编程 EPORT 数据寄存器来生成中断请求。

1 = 对应的 EPORT 引脚配置为输出

0 = 对应的 EPORT 引脚配置为输入

18.4.2.3. Eport 引脚分配寄存器（EPPAR）

地址：EPORTn_BASEADDR + 0x0000_0002

	15	14	13	12	11	10	9	8
读: 写:	eppa7[1:0]		eppa6[1:0]		eppa5[1:0]		eppa4[1:0]	
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读: 写:	eppa3[1:0]		eppa2[1:0]		eppa1[1:0]		eppa0[1:0]	
复位:	0	0	0	0	0	0	0	0

图 18-4：EPORT 引脚分配寄存器（EPPAR）

EPPAx[1:0] — EPORT Pin Assignment Select Fields

读/写 EPPAx 字段配置了水平检测和升降边缘检测的 EPORT 引脚，如下表所示。

配置为电平敏感的引脚被倒置，以便外部引脚上的逻辑 0 或逻辑 1 表示有效的中断请求。电级敏感的中断输入没有被锁定。为了保证电平敏感的中断请求被确认，中断源必须保持信号直到软件确认。必须选择电平灵敏度，以使设备脱离具有 INTx 中断的停止模式。

配置为边缘触发的引脚被锁住，不需要为中断生成保持动作状态。配置为边缘检测的引脚无论配置为输入或输出都被监控。

表 18-2：EPPAx Field Settings

EPPAx	引脚配置
00	Pin INTx level-sensitive
01	Pin INTx rising edge triggered
10	Pin INTx falling edge triggered
11	Pin INTx both falling edge and rising edge triggered

在 EPORT 模块中生成的中断请求可以被中断控制器模块屏蔽。EPPAR 的功能与所选的 Pin 方向无关。

复位将清除 EPPAx 字段。

18.4.2.4. Eport 引脚向上上拉启用寄存器（EPPUE）

地址：EPORTn_BASEADDR + 0x0000_0004

	7	6	5	4	3	2	1	0
读: 写:	埃普							
复位:	1	1	1	1	1	1	1	1

图 18-5：EPORT 引脚上拉式启用寄存器（EPPUE）

EPPUE[7:0] — Edge Port Pin Pull-up enable Bits

在 EPPUE 中设置任何位都会配置相应的引脚以启用向上。该设置只有在 EPORT 处于输入模式时才可用。清除 EPPUE 中的任何位将配置相应的引脚以禁用上拉。

复位设置 EPPUE[7:0]。

1 = 相应的 EPORT 引脚，配置为启用引线向上

0 = 相应的 EPORT 引脚，配置为禁用引线向上

18.4.2.5. 边缘端口标志寄存器（EPFR）

地址：EPORTn_BASEADDR + 0x0000_0005

	7	6	5	4	3	2	1	0
读:	EPF[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 18-6：EPORT 端口标志寄存器（EPFR）

EPF[7:0] — Edge Port Flag Bits

当 EPORT 引脚配置为边缘触发时，其在 EPFR 中对应的读写位表示已检测到所选择的边缘。复位清除 EPF[7:0]。

1 = 已检测到 INTx 引脚的选定边缘。

0 = 未检测到 INTx 引脚的已选边缘。

当在相应的引脚上检测到选定的边时，将设置该寄存器中的位。一点保持设置，直到写入 1 清除。写入 0 没有效果。如果引脚配置为电级敏感（EPPARx = 00），则引脚转换不会影响该寄存器。

18.4.2.6. 边缘端口引脚数据寄存器（EPPDR）

地址：EPORTn_BASEADDR + 0x0000_0006

	7	6	5	4	3	2	1	0
读:	EPPD[7:0]							
写:								
复位:	P	P	P	P	P	P	P	P

= 写入没有影响，访问终止，没有传输错误异常。

P = 电流引脚状态

图 18-7：EPORT 端口引脚数据寄存器（EPPDR）

EPD[7:0] — Edge Port Data Bits

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

18.4.2.7. 边缘端口数据寄存器（EPDR）

地址：EPORTn_BASEADDR + 0x0000_0007

	7	6	5	4	3	2	1	0
读:	EPD[7:0]							
写:								
复位:	1	1	1	1	1	1	1	1

图 18-8：EPORT 端口数据寄存器（EPDR）

EPD[7:0] — Edge Port Data Bits

写入 EPDR 的数据存储在内部寄存器中；如果端口的任何引脚配置为输出，为该引脚存储的位被驱动到引脚上。读取 EDPR 将返回存储在寄存器中的数据。复位设置 EPD[7:0]。

18.4.2.8. Eport 端口位设置寄存器（EPBSR）

地址：EPORTn_BASEADDR + 0x0000_0008

	7	6	5	4	3	2	1	0
读:	0							
写:	EPBSR[7:0]							
复位:	0	0	0	0	0	0	0	0

图 18-9：EPORT 端口位设置寄存器（EPBSR）

EPBSR[7:0] — EPORT Port Bit Set Register

1 = 将设置相应的 EPDR 位；

0 = 相应的 EPDR 位将不会被影响；

18.4.2.9. Eport 数字滤波器控制寄存器（EPFC）

地址：EPORTn_BASEADDR + 0x0000_0009

	7	6	5	4	3	2	1	0
读: 写:	Filter_Width[6:0]							Filter_En
复位:	0	0	0	0	0	0	0	0

图 18-10：EPORT 数字滤波器控制寄存器（EPFC）

Filter_Width[6:0] — EPORT Digital Filter Pulse Width Select Bit

确定要被过滤的输入脉冲的宽度。如果输入的脉冲宽度小于（Filter_Width[6:0] + 2），则将对脉冲进行滤波。

Filter_En — EPORT Digital Filter Enable Bit

1 = EPORT 数字滤波器为启用；

0 = EPORT 数字滤波器被禁用；

18.4.2.10. Eport Open Drain 启用寄存器（EPODE）

地址：EPORTn_BASEADDR + 0x0000_000A

	7	6	5	4	3	2	1	0
读: 写:	EPODE[7:0]							
复位:	0	0	0	0	0	0	0	0

图 18-11：EPORT Open Drain 启用寄存器（EPODE）

EPODE[7:0] — Edge Port Open Drain enable Bits

如果 EPORT 配置为输出，则设置 EPODE 中的任何位都会将相应的引脚配置为“Open Drain”输出。清除 EPODE 中的任何位，都会将相应的引脚配置为 CMOS 输出。复位清除环。

1 = 对应的 EPORT 引脚配置为 Open Drain 输出

0 = 相应的 EPORT 引脚配置为 CMOS 输出

18.4.2.11. Eport 电平极性寄存器（EPLPR）

地址：EPORTn_BASEADDR + 0x0000_0000B

	7	6	5	4	3	2	1	0
读: 写:	EPLPR[7:0]							
复位:	0	0	0	0	0	0	0	0

图 18-12：EPORT 级极性寄存器（EPLPR）

EPLPR[7:0]—边缘端口级别的极性位

如果 EPORT 配置为电平敏感，则在 EPLPR 中的任何位配置相应的引脚高电平敏感。清除 EPLPR 中的任何位将配置相应的引脚低电平敏感。复位清除 EPLPR[7:0]。

1 = 对应的 EPORT 引脚配置为高电平敏感型

0 = 相应的 EPORT 引脚配置为低电平敏感的

18.4.2.12. Eport 端口位清除寄存器（EPBCR）

地址：EPORTn_BASEADDR + 0x0000_000F

	7	6	5	4	3	2	1	0
读:	0							
写:	EPBCR[7:0]							
复位:	0	0	0	0	0	0	0	0

图 18-13：EPORT 端口位清除寄存器（EPBCR）

EPBCR[7:0] — EPORT Port Bit Clear Register

1 = 相应的 EPDR 位将被清除；

0 = 相应的 EPDR 位将不会被影响；

19. CANBus 控制程序（CANBC）

19.1. 功能介绍

LT7589 的 CANBus 模块是一个基于 CAN 2.0B 协议规范实现 CAN 协议的通信控制器。一般的方块图如下图所示，它描述了在 CANBus 模块中实现的主要子块，包括两个嵌入式存储器，一个用于存储消息缓冲区（MB），另一个用于存储 Rx 单个掩码寄存器。这些子模块的功能将在后续的章节中进行描述。

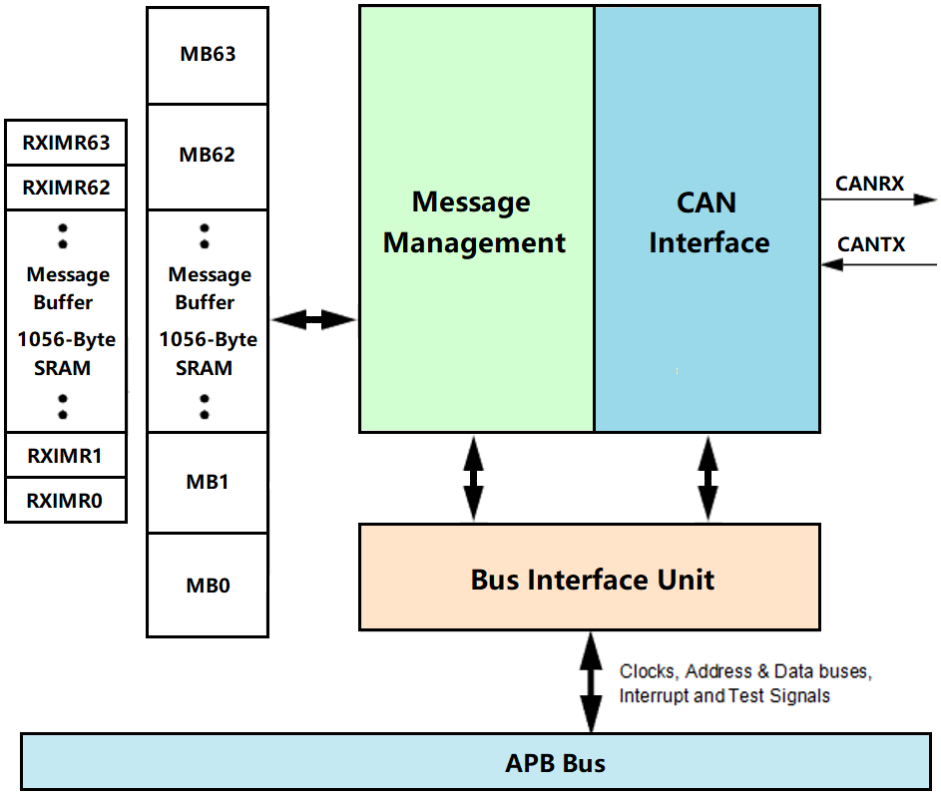


图 19-1：CANBus 方块图

19.1.1. 基本概述

CAN 协议主要但不仅仅是设计为作为车辆串行数据总线，满足该领域的具体要求：实时处理、在车辆 EMI 环境中的可靠操作、成本效益和所需的带宽。CANBus 模块是 CAN 协议规范 2.0 B 版本的完整实现，该规范支持标准的和扩展的消息帧。

最多有 64 个消息缓冲区可用。消息缓冲区存储在一个专用于 CANBus 模块的嵌入式 SRAM 中。

CAN 协议接口（CPI）子模块管理 CANBus 上的串行通信，请求 SRAM 访问以接收和发送消息帧，验证接收到的消息并执行错误处理。消息缓冲区管理（MBM）子模块处理接收和传输的消息缓冲区选择，处理仲裁和 ID 匹配算法。总线接口单元（BIU）子模块控制对内部接口总线的访问，以便建立与 CPU 和其他块的连接。时钟、地址和数据总线、中断输出和测试信号均通过总线接口单元进行访问。

19.1.2. CANBus 模块功能

CANBus 模块包括以下显着特点：

- 完全实现了 CAN 协议规范，版本 2.0B
  - 标准数据和远程帧
  - 扩展的数据和远程帧
  - 0~8 字节的数据长度
  - 可编程比特率高达 1 Mbit/s
  - 与内容相关的地址
- 64 个数据长度为 0 到 8 个字节的消息缓冲区
- 每个 MB 可配置为 Rx 或 Tx，所有支持标准和扩展消息
- 每个消息缓冲区的单独的 Rx 掩码寄存器
- 包括 1056 字节（64MB）的 SRAM 用于 MB 存储
- 包括 256 字节（64 MBs）的 SRAM，用于单个 Rx 掩码寄存器
- 全功能的 Rx FIFO 与存储容量为 6 帧和内部指针处理
- 强大的 Rx FIFO ID 过滤，能够匹配输入的 ID 与 8 个扩展，16 个标准或 32 个部分（8 位）ID，具有单独的屏蔽能力
- 可编程时钟源到 CANBUS 协议接口，或总线时钟或晶体振荡器
- 未使用的 MB 和 Rx 掩码寄存器空间可以用作通用的 SRAM 空间
- 只听模式能力
- 可编程回路返回模式，支持自检操作
- 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
- 全局网络时间，由一个特定的消息进行同步
- 可移动的中断
- 独立于传输介质（假定有一个外部收发器）
- 由于针对高优先级消息的仲裁方案而导致的延迟时间较短
- 低功耗模式
- Tx 消息缓冲区上的硬件取消

19.1.3. 操作模式

CANBus 模块有四种功能模式：正常模式（用户和主管）、冻结模式、只收听模式和回路模式。还有一个低功率模式：禁用模式。

- **正常模式（用户或主管）**

  在正常模式下，该模块操作接收和/或发送消息帧，错误处理正常，所有的 CANBUS 协议功能都被启用。用户模式和监管模式在对某些受限控制寄存器的访问方面有所不同。

- **冻结模式**

  当产生 MCR 中的 FRZ 位时，将启用它。如果启用，则在设置 MCR 中的 HALT 位或在 MCU 级别请求调试模式时，将输入冻结模式。在这种模式下，没有帧的传输或接收，对 CANBus 的同步性丢失。

- **只听模式**

  当确认控制寄存器中的 LOM 位时，模块进入此模式。在此模式下，传输被禁用，所有错误计数器被冻结，模块以 CANBUS 错误被动模式运行。将只接收由另一个 CAN 站确认的消息。如果 CANBus 检测到一个未被确认的消息，它将标记一个 BIT0 错误（不更改 REC），就像它试图承认该消息一样。

- **回路回路模式**

  当确认控制寄存器中的 LPB 位时，模块进入此模式。在这种模式下，CANBus 执行一个可用于自检操作的内部循环。发送器的比特流输出在内部反馈到接收器输入。Rx CAN 输入引脚被忽略，Tx CAN 输出进入隐性状态（逻辑“1”）。CANBus 的行为与传输时相同，并将自己传输的消息视为从远程节点接收的消息。在此模式下，CANBus 忽略 CAN 帧确认字段中 ACK 槽期间发送的位，以确保正确接收自己的消息。同时产生发送中断和接收中断。

- **模块禁用模式**

  当产生 MCR 中的 MDIS 位时，将输入此低功率模式。当禁用时，该模块将关闭到 CAN 协议接口和消息缓冲区管理子模块的时钟。通过否定 MCR 中的 MDIS 位来退出此模式。

19.2. 外部信号描述

19.2.1. 基本概述

CANBus 模块有两个 I/O 信号连接到外部 MCU 引脚。这些信号汇总在下表中并在接下来的小节中有更详细的描述。

表 19-1：CANBus 信号

信号名称	方向	描述
CANRX	输入	CANBUS 接收引脚
CANTX	输出	CANBUS 传输引脚

19.2.2. 信号描述

19.2.2.1. CANRX

此引脚是来自 CANBus 收发器的接收引脚。主导状态由逻辑级别“0”表示。隐性状态由逻辑级别“1”表示。

19.2.2.2. CANTX

此引脚是发送到 CANBus 收发器的传输引脚。主导状态由逻辑级别“0”表示。隐性状态由逻辑级别“1”表示。

以下是 Can 总线应用电路的例子。

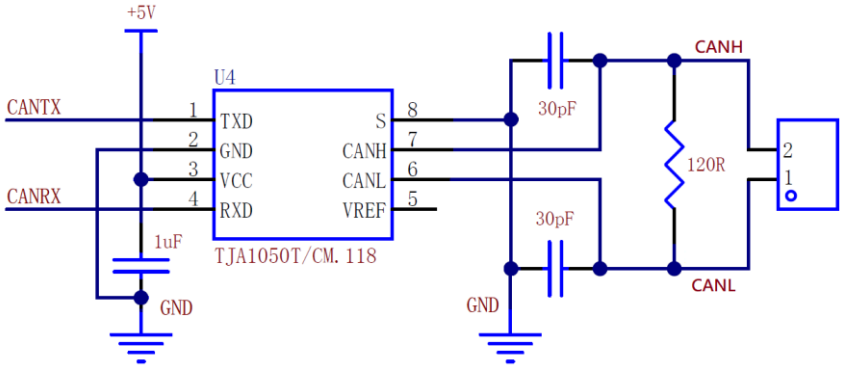


图 19-2：Canbus 电路示例

20. 串行通信接口（SCI）

20.1. 功能介绍

LT7589 的串行通信接口模块（SCI）支持基本的 UART 功能，并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 从属操作。

20.2. 通信接口特点

每个 SCI 模块的特点包括：

- 全双工，标准的非返回到零（NRZ）格式
- 可编程波特率（13 位调制分频器），具有可配置的过采样比，从 4 倍到 256 倍
- 中断，轮询操作：
  - 传输数据寄存器为空，传输完成
  - 完全接收数据寄存器
  - 接收溢出、奇偶校验误差、帧误差和噪声误差
  - 怠速接收器检测
  - 接收引脚上的上升或下降缘的动作
  - 中断检测所支持的 LIN
  - 接收数据匹配
- 硬件奇偶校验的生成和检查
- 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
- 三种接收器唤醒方法：
  - 怠速线路唤醒
  - 地址标记唤醒
  - 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
- 可选择的发送器输出和接收器输入极性

- 可选择的 IrDA 1.4 返回到零反转（RZI）格式，具有可编程的脉冲宽度
- 独立的 FIFO 结构
  - 接收和传输请求的可配置水印
  - 如果接收 FIFO 不是空，接收方在可配置的空闲字符之后产生请求的选项

20.3. 操作模式

- 停止模式
- 等待模式

20.4. 方块图

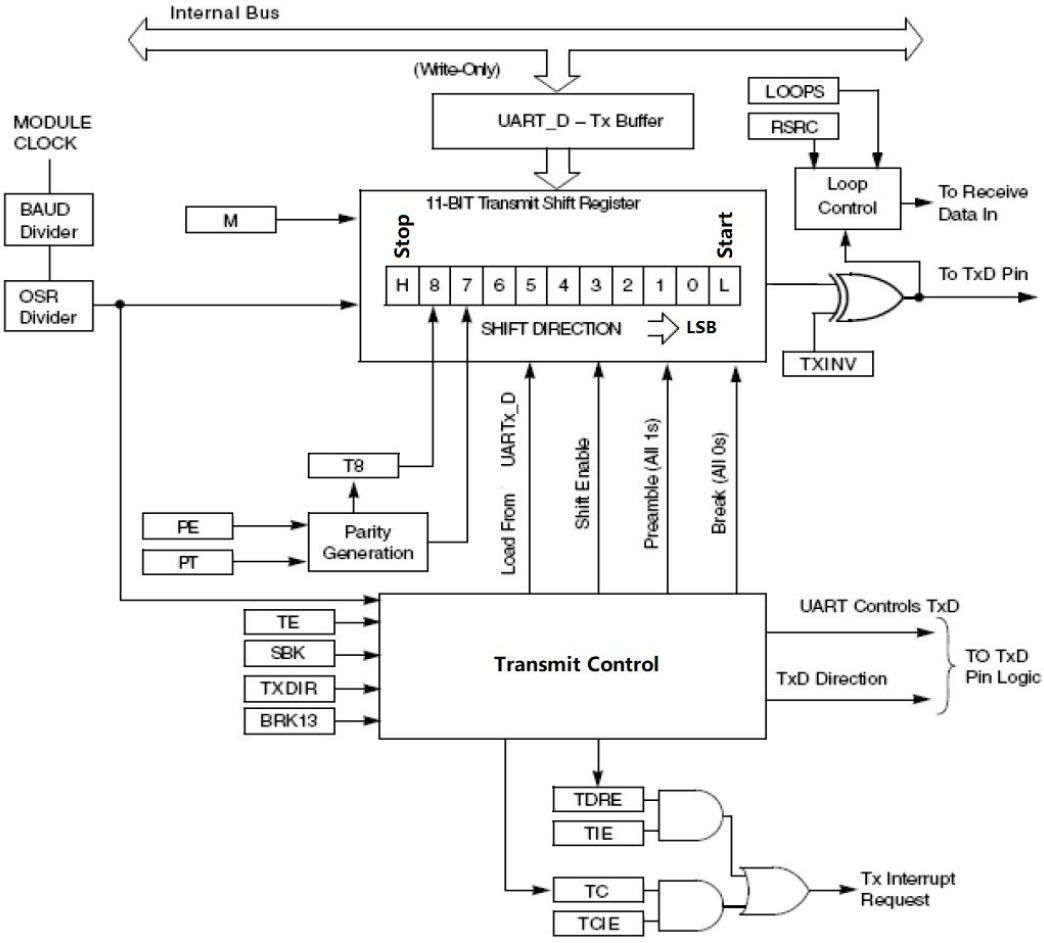


图 20-1：SCI 发送器方块图

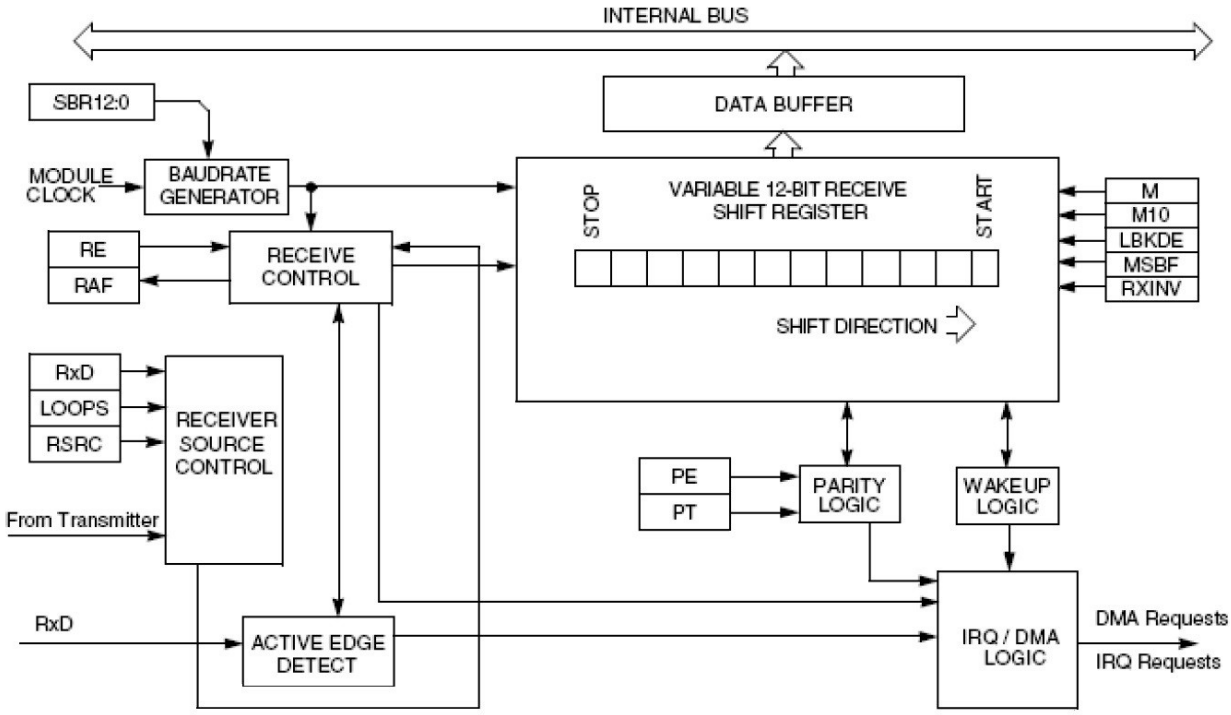


图 20-2：SCI 接收器方块图

20.5. 操作模式

20.5.1. 停止模式

在停止模式下将无法工作。

20.5.2. 等待 Mode

当设置了 DOZEEN 位时，SCI 可以被配置为在等待模式下停止。

发送器和接收器将完成当前单词的发送/接收。

20.6. 信号描述

下表给出了这里所描述的信号的概述。

表 20-1：信号属性

信号	描述	I/O
TXD	传输数据。此引脚通常是输出，但当发送器被禁用或为接收数据配置传输方向时，它是单线模式下的输入（三态）。	I/O
RXD	接收数据	I
SCI_DE	RS-485 收发信机的控制信号	O

20.7. 内存映射和寄存器

20.7.1. 内存映射

SCI 模块记忆图如下表所示 SCI0 的基本地址是 0x4009_0000，SCI1 是 0x4008_0000，SCI2 是 0x400C_0000。

表 20-2：SCI 模块记忆地图

偏移地址	Bits[31:0]
0x0000	SCI Version ID (SCI_VERID)
0x0004	SCI Parameter (SCI_PARAM)
0x0008	SCI Reset (SCI_RESET)
0x000C	SCI Pin (SCI_PIN)
0x0010	SCI Baud Rate Register (SCI_BAUD)
0x0014	SCI Status Register (SCI_STAT)
0x0018	SCI Control Register (SCI_CTRL)
0x001C	SCI Data Register (SCI_DATA)
0x0020	SCI Match Address Register (SCI_MATCH)
0x0024	SCI Modem IrDA Register (SCI_MODIR)
0x0028	SCI FIFO Register (SCI_FIFO)
0x002C	SCI Watermark Register (SCI_WATER)
0x0030	SCI Oversampling Ratio Register(SCI_OSR)

提示：

每个模块被分配了 16K 字节的地址空间，所有这些都可能无法被解码。在指定的模块内存映像之外的访问将生成总线错误异常。

20.7.2. 寄存器说明

20.7.2.1. SCI 版本 ID 寄存器（SCI_VERID）

地址：SCIn_BASEADDR + 0x0000_0000

	31	30	29	28	27	26	25	24
读:	VERID[31:24]							
写:								
复位:	0	0	0	0	0	1	0	0

	19	22	21	20	19	18	17	16
读:	VERID[23:16]							
写:								
复位:	0	0	0	0	0	0	0	0

	11	14	13	12	11	10	9	8
读:	VERID[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	VERID[7:0]							
写:								
复位:	0	0	0	0	0	0	1	1

= 写入没有影响，访问终止，没有传输错误异常。

图 20-3：SCI 版本 ID 寄存器（SCI_VERID）

VERID_ID[31:0] — SCI Version ID

20.7.2.2. SCI 参数寄存器（SCI_PARAM）

地址：SCIn_BASEADDR + 0x0000_0004

	31	30	29	28	27	26	25	24
读:	保留							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	保留							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	保留				RXFIFO_SZ[3:0]			
写:								
复位:	0	0	0	0	0	0	1	1

	7	6	5	4	3	2	1	0
读:	保留				TXFIFO_SZ[3:0]			
写:								
复位:	0	0	0	0	0	0	1	1

= 写入没有影响，访问终止，没有传输错误异常。

图 20-4：SCI 参数寄存器（SCI_PARAM）

RXFIFO_SZ[3:0] — The receive buffer/FIFO size

此只读控制位表示接收缓冲区/FIFO 的大小。

TXFIFO_SZ[3:0] — The transmit buffer/FIFO size

此只读控制位表示传输缓冲区/FIFO 的大小。

20.7.2.3. SCI 复位寄存器（SCI_RESET）

地址：SCIn_BASEADDR + 0x0000_0008

	31	30	29	28	27	26	25	24
读:	保留							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	保留							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	保留							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	保留						SWRST	保留
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 20-5：SCI 复位寄存器（SCI_RESET）

SWRST — Software Reset

这个读/写位允许软件复位 SCI IP。

1 = 软件复位已确认

0 = 未确认有任何软件复位

20.7.2.4. SCI 引脚寄存器（SCI_PIN）

地址：SCIn_BASEADDR + 0x0000_000C

	31	30	29	28	27	26	25	24
读:	保留							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	保留							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	保留							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	保留						PINCFG	
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 20-6：SCI 引脚寄存器（SCI_PIN）

PINCFG — Useless in this module

20.7.2.5. SCI 波特率寄存器（SCI_BAUD）

地址：SCIn_BASEADDR + 0x0000_0010

	31	30	29	28	27	26	25	24
读:	MAEN1	MAEN2	M10	保留				
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	TDMAE	Reserved	RDMAE	Reserved	MATCFG[1:0]		BOTHEDG	RESYNCDIS
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	LBKDIE	RXEDGIE	SBNS	SBR[12:8]				
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	SBR[7:0]							
写:								
复位:	0	0	0	0	0	1	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 20-7：SCI 波特率寄存器（SCI_BAUD）

MAEN1 — Match Address Mode Enable 1

1 = 启用自动地址匹配或数据匹配模式 MATCH[MA1]

0 = 正常操作

MAEN2 — Match Address Mode Enable 2

1 = 启用自动地址匹配或数据匹配模式 MATCH[MA2]

0 = 正常操作

M10 — 10-bits Mode select

M10 位使十分之一位成为串行传输的一部分。只有当发送器和接收器都被禁用时，才应该更改此位。

1 = 接收器和发送器使用 10 位数据字符

0 = 接收器和发送器使用 8 位或 9 位的数据字符

TDMAE — Transmitter DMA Enable

TDMAE 配置传输数据寄存器空标志 LPUART_STAT[TDRE]，以生成 DMA 请求。

1 = DMA 请求允许

0 = DMA 请求被禁用

RDMAE — Receiver Full DMA Enable

RDMAE 配置接收方数据寄存器全标志 LPUART_STAT[RDRF]，以生成 DMA 请求。

1 = DMA 请求允许

0 = DMA 请求被禁用

MATCFG[1:0] — Match Configuration

配置所使用的匹配寻址模式：

00 -- Address Match Wakeup;
01 -- Idle Match Wakeup;
10 -- Match On and Match Off;
11 -- Enable RWU on Data Match and Match On/Off for transmitter CTS input

BOTHEDGE — Both Edge Sampling

允许在波特率时钟的两侧边缘上对接收到的数据进行采样，在给定的过采样比下，有效地使接收器对输入数据的采样次数翻倍。这个位必须设置为 x4 和 x7 之间的过采样比，并且对于更高的过采样比是可选的。这个位只有在接收器被禁用时才应该改变。

1 = 接收器使用波特率时钟的上升和下降边缘采样输入数据。

0 = 接收器使用波特率时钟的上升边缘采样输入数据。

RESYNCDIS — Resynchronization Disable

当设置时，当检测到数据 1 和数据零转换时，它禁用接收到的数据字的重新同步。只有在禁用接收器的情况下，才应更改此位。

1 = 在接收数据字期间禁用重新同步

0 = 支持接收数据字期间的重新同步

LBKDIE — LIN Break Detect Interrupt Enable

LBKDIE 允许 LIN 中断检测标志 LBKDIF 来生成中断请求。

1 = 当 SCI_STAT[LBKDIF]标志为 1 时，将请求硬件中断

0 = 来自 SCI_STAT[LBKDIF]的硬件中断被禁用（使用轮询）

RXEDGIE — RX Input Active Edge Interrupt Enable

启用接收输入动作的上升或下降缘 RXEDGIF 来生成中断请求。在设置 RXEDGIE 时，更改 CTRL[LOOP]或 CTRL[RSRC]可能会导致设置 RXEDGIF。

1 = 当 SCI_STAT 标志为 1 时，请求硬件中断

0 = 个来自 SCI_STAT 的硬件中断被禁用（使用轮询）

SBNS — Stop Bit Number Select SBNS determines whether data characters are one or two stop bits. This bit should only be changed when the transmitter and receiver are both disabled.

1 = 两个停止位

0 = 一个停止位

SBR[12:0] — Baud Rate Modulo Divisor

SBR[12:0]中的 13 位数据为波特率发生器设置模除法率。当 SBR 为 1~8191 时，波特率等于“baud clock / ((OSR + 1) × SBR)”。只有当发送器和接收器都被禁用（SCI_CTRL[RE] andSCI_CTRL[TE]均为 0）时，才能更新 13 位波特率设置[SBR12:SBR0]。

20.7.2.6. SCI 状态寄存器（SCI_STAT）

地址：SCIn_BASEADDR + 0x0000_0014

	31	30	29	28	27	26	25	24
读:	LBKDIF	RXEDGIF	MSBF	RXINV	RWUID	BRK13	LBKDE	RAF
写:	w1c	w1c						
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
写:				w1c	w1c	w1c	w1c	w1c
复位:	1	1	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	MA1F	MA2F	保留					
写:	w1c	w1c						
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	保留							
写:								
复位:	0	0	0	0	0	1	0	0

= 写入没有影响，访问终止，没有传输错误异常。

w1c = .写 1 到该位将清除它

图 20-8：SCI 状态寄存器（SCI_STAT）

LBKDIF — LIN Break Detect Interrupt Flag

当 LIN 断开检测电路启用并检测到 LIN 断开字符时，设置 LBKDIF。LBKDIF 通过写 1 来清除。

1 = 已检测到 LIN 中断字符

0 = 未检测到 LIN 中断字符

RXEDGIF — RXD Pin Active Edge Interrupt Flag

RXEDGIF 是在 RXD 引脚上出现动作的上升或下降缘时设置的，如果 RXINV = 0 时下降缘，如果 RXINV = 1 时上升缘。RXEDGIF 可以通过写一个 1 来清除它。

1 = 在接收引脚上出现了一个动作的上升或下降缘

0 = 在接收引脚上没有发生动作的上升或下降缘

MSBF — MSB First

设置此位会反转在数据线上传输和接收的位的顺序。这个位不影响这些位的极性、奇偶校验位的位置或开始位或停止位的位置。只有当发送器和接收器都被禁用时，才应该更改此位。

1 = MSB（位 9、位 8、位 7 或位 6 位）是在开始位之后传输的第一个位，这取决于 CTRL[M]、CTRL[PE]和 BAUD[M10]的设置。此外，根据 CTRL[M]和 CTRL[PE]的设置，在开始位之后接收到的第一位被识别为位 9、位 8、位 7 或位 6。

0 = LSB（位 0）是在起始位之后传输的第一个位。此外，在起始位之后接收到的第一位被识别为位 0

RXINV — Receive Data Inversion

设置此位会反转接收到的数据输入的极性。设置 RXINV 会反转所有情况下的 RXD 输入：数据位，开始和停止位，中断，

和空闲。

1 = 接收数据被反向

0 = 接收数据没有反向

RWUID — Receive Wake Up Idle Detect

对于空闲字符上的 RWU，RWUID 控制唤醒接收器的空闲字符是否设置了 idle 位。对于地址匹配唤醒，RWUID 控制在地址不匹配时是否设置了 IDLE 位。只有在禁用接收器的情况下，才应更改此位。

1 = 在接收待机状态（RWU = 1）期间，在检测到空闲字符时将设置 IDLE 位。在地址匹配唤醒期间，当地址不匹配时，确实会设置 IDLE 位。

0 = 在接收待机状态（RWU = 1）期间，在检测到空闲字符时不会设置 IDLE 位。在地址匹配唤醒期间，当地址不匹配时，不会设置 IDLE 位。

BRK13 — Break Character Generation Length

BRK13 选择一个较长的传输中断字符长度。帧错误的检测不受该位的状态的影响。只有在发送器被禁用时，才应更改这个位。

1 = Break 字符的传输长度为 13 位次（如果 M = 0、SBNS = 0）或 14（如果 M = 1、SBNS = 0 或 M = 0、SBNS = 1）或 15（如果 M = 1、SBNS = 1 或 M10 = 1、SNBS = 0）或 16（如果 M10 = 1、SNBS = 1）。

0 = Break 字符的传输长度为 10 位次（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 M10 = 1、SNBS = 0）或 13（如果 M10 = 1、SNBS = 1）

LBKDE — LIN Break Detection Enable

LBKDE 选择一个较长的断点字符检测长度。当设置 LBKDE 时，接收数据不存储在接收数据缓冲区中。

1 = 中断字符检测长度为 11 位次 (if M = 0, SBNS = 0) or 12 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 14 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 15 (if M10 = 1, SNBS = 1)

0 = 中断字符检测长度为 10 位次 (if M = 0, SBNS = 0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 13 (if M10 = 1, SNBS = 1)

RAF — Receiver Active Flag

当接收方检测到有效启动位的开始时，将设置 RAF，当接收方检测到空闲线时，将自动清除 RAF。

1 = SCI 接收器激活(RXD 输入非空闲)

0 = SCI 接收器空闲，等待一个启动位

TDRE — Transmit Data Register Empty Flag

当发射 FIFO 启用时，TDRE 将设置发射 FIFO （SCI_DATA）中的数据字数等于或小于 SCI_WATER[TXWATER])。要清除 TDRE，请写入 SCI 数据寄存器（SCI_DATA），直到传输 FIFO 中的单词数大于 SCI_WATER[TXWATER]所表示的数字。当传输 FIFO 被禁用时，TDRE 将在传输数据寄存器（SCI_DATA）为空时进行设置。要清除 TDRE，请写入 SCI 数据寄存器（SCI_DATA）。

TDRE 不受传输过程中的字符的影响，它在每个传输字符开始时更新。

1 = 传输数据缓冲区为空

0 = 传输数据缓冲区已满

TC — Transmission Complete Flag

当传输正在进行中或加载了前导字符或断开字符时，TC 将被清除。当传输缓冲区为空且没有传输数据、前导或中断字符时，设置 TC。当 TC 设置时，发射数据输出信号变为空闲（逻辑 1）。TC 通过写入 SCI_DATA 传输新数据，通过清除并设置 SCI_CTRL[TE]，通过写入 1 写入 SCI_CTRL[SBK]排队中断字符。

1 = 发送器空闲（传输活动完成）

0 = 发送器处于活动状态（发送数据、前导码或中断）

RDRF — Receive Data Register Full Flag

当启用接收 FIFO 时，当接收缓冲区中的数据字数大于指示的数字 bySCI_WATER[RXWATER]时，将设置 RDRF。要清除 RDRF，请读取 SCI_DATA，直到接收数据缓冲区中的数据字数等于或小于 SCI_WATER[RXWATER]所指示的数字。当接收 FIFO 被禁用时，当接收缓冲区（SCI_DATA）已满时，将设置 RDRF。若要清除 RDRF，请读取 SCI_DATA 寄存器。

正在接收过程中的字符在接收到整个字符之前不会导致 RDRF 的更改。即使设置了 RDRF，该字符也将继续被接收，直到接收到整个字符后出现溢出情况为止。

1 = 接收数据缓冲区已满

0 = 接收数据缓冲区为空

IDLE — Idle Line Flag

当 SCI 接收行在一段活动后完全字符空闲时，设置 IDLE。当 ILT 被清除时，接收器在开始位之后开始计算空闲位时间。如果接收字符全部为 1s，这些位时间和停止位时间计入逻辑高的全部字符时间，10 到 13 位时间，需要接收器检测空闲线。当设置 ILT 时，接收器直到停止位之后才开始计算空闲位时间。前一个字符结尾的停止位和任何逻辑高位时间不计入接收器检测空闲行所需的逻辑高的全部字符时间。

要清除 IDLE，请将逻辑 1 写入 IDLE 标志。清除 IDLE 后，只有在接收缓冲区中存储了一个新字符或一个 LIN 中断字符设置了 LBKDIF 标志之后，才能再次设置它。即使接收行长时间空闲，也只设置一次。

1 = 检测到空闲线（Idle Line）

0 = 未检测到空闲线

OR — Receiver Overrun Flag

当软件无法防止接收数据寄存器溢出数据时，将设置 OR。在完全收到溢出缓冲区的数据字的停止位后，立即设置 OR 位，并阻止所有其他错误标志（FE、NF 和 PF）进行设置。移位寄存器中的数据丢失，但 SCI 数据寄存器中的数据不受影响。如果启用了 LBKDE 并且检测到 LIN Break，则如果在接收下一个数据字符之前没有清除 LBKDIF，LBKDE 字段将产生。

在设置 OR 标志时，即使存在足够的空间，数据缓冲区中也不会存储额外的数据。若要清除 OR，请将逻辑 1 写入 OR 标志。

1 = 接收溢出（新的 SCI 数据丢失）

0 = 无溢出

NF — Noise Flag

接收器中使用的先进采样技术在每个接收的比特中采集三个样本。如果这些样本中的任何一个与帧中的任何比特时间内的其他样本不一致，则检测到该字符的噪声。每当从 SCI_DATA 中读取的下一个字符在字符中检测到噪声时，就设置 NF。要清除 NF，请将逻辑 1 写入 NF。

1 = 在 SCI_DATA 中的接收字符中检测到噪声

0 = 未检测到噪声

FE — Framing Error Flag

当接收到从 SCI_DATA 读取的下一个字符，检测到需要停止位时，就设置 FE。要清除 FE，请将逻辑 1 写入 FE。

1 = 数据框架错误

0 = 未检测到数据框架。这并不能保证数据是正确的

PF — Parity Error Flag

当启用奇偶校验（PE = 1）时，当接收到要从 SCI_DATA 读取的下一个字符，并且接收到的字符中的奇偶校验位与预期的奇偶校验值不一致时，就会设置 PF。要清除 PF，请向 PF 写入一个逻辑 1。

1 = 奇偶校验错误

0 = 无奇偶校验错误

MA1F — Match 1 Flag

当从 SCI_DATA 读取的下一个字符与 MA1 匹配时，将设置 MA1F。要清除 MA1F，请向 MA1F 写入一个逻辑 1。

1 = 接收到的数据等于 MA1

0 = 接收到的数据不等于 MA1

MA2F — Match 2 Flag

当从 SCI_DATA 读取的下一个字符与 MA2 匹配时，将设置 MA2F。要清除 MA2F，请向 MA2F 写入一个逻辑 1。

1 = 接收到的数据等于 MA2

0 = 接收到的数据不等于 MA2

20.7.2.7. SCI 控制寄存器（SCI_CTRL）

地址：SCIn_BASEADDR + 0x0000_0018

	31	30	29	28	27	26	25	24
读: 写:	R8T9	R9T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读: 写:	TIE	TCIE	RIE	ILIE	TE	RE	RUW	SBK
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读: 写:	MA1IE	MA2IE	Reserved			IDLECFG		
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读: 写:	LOOPS	DOZEEN	RSRC	M	WAKE	ILT	PE	PT
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 20-9：SCI 控制寄存器（SCI_CTRL）

R8T9 — Receive Bit 8 / Transmit Bit 9

R8 是当 SCI 配置为 9 位或 10 位数据格式时接收到的第 9 个数据位。当读取 9 位或 10 位的数据时，在读取 SCI_DATA 之前先读取 R8。

T9 是 SCI 配置为 10 位数据格式时接收的第 1 个数据位。当写入 10 位数据时，在写入 SCI_DATA 之前写入 T9。如果 T9 不需要改变它以前的值，例如当它被用于生成地址标记或奇偶校验时，那么它不需要被写入每个 timeSCI_DATA。

R9T8 — Receive Bit 9 / Transmit Bit 8

R9 是当 SCI 配置为 10 位数据格式时接收到的第 10 个数据位。当读取 10 位数据时，在读取 SCI_DATA 之前读取 R9

T8 是当 SCI 配置为 9 位或 10 位数据格式时接收到的第 9 个数据位。当写入 9 位或 10 位数据时，在 writingSCI_DATA 之前写入 T8。如果 T8 不需要改变它以前的值，例如当它被用于生成地址标记或奇偶校验时，那么它就不需要在每次写 SCI_DATA 时都被写出来。

TXDIR — TXD Pin Direction in Single-Wire Mode

当 SCI 配置为单线半双工操作（LOOPS = RSRC = 1）时，该位确定了 TXD 引脚处的数据方向。当清除 TXDIR 时，发送器将在接收器开始从 TXD 引脚接收数据之前完成接收当前字符（如果有的话）。

1 = TXD 引脚是在单线模式下的输出

0 = TXD 引脚是一个在单线模式下的输入

TXINV — Transmit Data Inversion

设置此位会反转传输数据输出的极性。设置 TXINV 会反转所有情况下的 TXD 输出：数据位、启动和停止位、中断和空闲。

1 = 传输数据被反向

0 = 传输数据没有反向

ORIE — Overrun Interrupt Enable

此位允许溢出标志（OR）生成硬件中断请求。

1 = 在设置 OR 时，请求硬件中断

0 = OR 中断被禁用；使用轮询

NEIE — Noise Error Interrupt Enable

此位使噪声标志（NF）能够生成硬件中断请求。

1 = 当设置 NF 时，请求硬件中断

0 = NF 中断被禁用；使用轮询

FEIE — Framing Error Interrupt Enable

此位允许框架错误标志（FE）生成硬件中断请求。

1 = 当设置 FE 时，请求硬件中断

0 = FE 中断被禁用；使用轮询

PEIE — Parity Error Interrupt Enable

此位使奇偶校验错误标志（PF）能够生成硬件中断请求。

1 = 当设置 PF 时，请求硬件中断

0 = PF 中断被禁用；使用轮询

TIE — Transmit Interrupt Enable

此位使 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 = 来自 TDRE 的硬件中断被禁用；使用轮询

TCIE — Transmission Complete Interrupt Enable for

TCIE 允许传输完成标志 TC 生成中断请求。

1 = 当 TC 标志为 1 时，请求硬件中断

0 = 来自 TC 的硬件中断被禁用；使用轮询

RIE — Receiver Interrupt Enable

此位使 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 = 来自 RDRF 的硬件中断被禁用；使用轮询

ILIE — Idle Line Interrupt Enable

ILIE 允许空闲行标志 STAT[IDLE]来生成中断请求。

1 = 当 IDLE 标志为 1 时，请求硬件中断

0 = 来自 IDLE 的硬件中断被禁用；使用轮询

TE — Transmitter Enable

此位可启用 SCI 发送器。TE 还可以通过清除然后设置 TE 来设置空闲前导队列。当 TE 被清除时，这个寄存器位将读取为 1，直到发送器完成当前字符，TXD 引脚是三态的。

1 = 发送器启用

0 = 发送器被禁用

RE — Receiver Enable

此位可启用 SCI 接收器。当 RE 写入 0 时，此寄存器位将读取为 1，直到接收器完成接收到当前字符（如果有的话）。

1 = 接收器启用

0 = 接收器被禁用

RWU — Receiver Wakeup Control

此字段可以设置为将 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 IDLE 事件，或当 CTRL[WAKE]与 STAT[RWUID]清除时地址匹配。

如果通道当前没有空闲，则 RWU 必须仅设置为 CTRL[WAKE] = 0（空闲时唤醒）。这可以由国家统计局来确定。如果该标志被设置为唤醒一个 IDLE 事件，而该通道已经空闲，则 SCI 可能会丢弃数据。这是因为在允许重新插入 IDLE 之前，必须在检测到 IDLE 之后接收数据或检测到 LIN 中断。

1 = SCI 接收器处于备用状态，等待唤醒状态

0 = 接收器正常工作

SBK — Send Break

将 1 和 0 写入 SBK 在传输数据流中的中断字符。额外的中断字符为 10 到 13，或 13 到 16，如果设置了 SCI_STATBRK13]，只要设置了 SBK，逻辑 0 的位时间就会排队。根据设置的时间和 SBK 相对于当前正在传输的信息的清除情况，在软件清除 SBK 之前，可能会排队等待第二个中断字符。

1 = 要发送的队列中断字符(s)

0 = 传送器正常工作

MA1IE — Match 1 Interrupt Enable

1 = MA1F 中断启用

0 = MA1F 中断禁用

MA2IE — Match 2 Interrupt Enable

1 = MA2F 中断启用

0 = MA2F 中断禁用

IDLECFG — Idle Configuration

此位确认了在设置 idle 标志之前必须接收到的空闲字符数。

000 --1 idle character;
001 -- 2 idle characters;
010 -- 4 idle characters;
011 -- 8 idle characters;
100 -- 16 idle characters;
101 -- 32 idle characters;
110 -- 64 idle characters;

111 -- 128 idle characters

LOOPS — LOOP Mode Select

当设置环路时，RXD 引脚与 SCI 断开，发送器输出在内部连接到接收器输入。发送器和接收器必须启用才能使用循环功能。

1 = 回路模式或单线模式，其中发送器输出内部连接到接收器输入（参考 RSRC 位）

0 = 正常操作- RXD 和 TXD 使用单独的引脚

DOZEEN — Doze Enable

1 = SCI 在打盹模式下被禁用

0 = SCI 在打盹模式下启用

RSRC — Receiver Source Select

除非设置了循环字段，否则此字段没有任何意义或效果。当设置循环时，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 = 提供的循环被设置，RSRC 被清除，选择内环回路模式，SCI 不使用 RXD 引脚

M — 9-Bit or 8-Bit Mode Select

1 = 接收器和发送器使用 9 位数据字符

0 = 接收器和发送器使用 8 位数据字符

WAKE — Receiver Wakeup Method Select

这个位确定当 RWU = 1：地址标记在接收数据字符的最重要位位置或接收引脚输入信号的空闲状态时，哪个状态唤醒 SCI。

1 = 配置 RWU 与 address-mark 唤醒

0 = 配置 RWU 以进行 idel-line 唤醒

ILT — Idle Line Type Select

该位确定接收器何时开始将逻辑 1 计算为空闲字符位。计数可以在有效的开始位之后或在停止位之后开始。如果计数在开始位之后开始，那么在停止位之前的逻辑串 1 可能会导致对空闲字符的错误识别。在停止位后开始计数可以避免错误的空闲字符识别，但需要正确的同步传输。如果 SCI 是用 ILT = 1 编程的，那么一个逻辑 0 会在接收到的停止位后被自动移位，因此会复位空闲计数。

1 = 空闲字符位计数在停止位之后开始

0 = 空闲字符位计数在开始位之后开始

PE — Parity Enable

此位支持硬件奇偶校验生成和检查。当启用奇偶校验时，停止位之前的位立即被视为奇偶校验位。

1 = 启用奇偶校验

0 = 无硬件奇偶校验生成或检查

PT — Parity Type

如果启用了奇偶校验（PE = 1），该位将选择偶数或奇数奇偶校验。奇数奇偶性是指数据字符中的 1s 的总数，包括奇偶性位，是奇数。偶数意味着数据字符中的 1 总数，包括奇偶校验位，是偶数。

1 = 奇数校验

0 = 偶数校验

20.7.2.8. SCI 数据寄存器（SCI_DATA）

地址：SCIn_BASEADDR + 0x0000_001C

	31	30	29	28	27	26	25	24
读:	保留							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	保留							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	NOISY	PARITYE	FRETSC	RXEMPT	IDLINE	Reserved	R9T9	R8T8
写:								
复位:	0	0	0	1	0	0	0	0

	7	6	5	4	3	2	1	0
读:	R7T7	R6T6	R5T5	R4T4	R3T3	R2T2	R1T1	R0T0
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 20-10：SCI 数据寄存器（SCI_DATA）

NOISY — The current received dataword contained in DATA[R9:R0] was received with noise

1 = 数据接收有噪声

0 = 数据字接收到无噪声

PARITYE — The current received dataword contained in DATA[R9:R0] was received with a parity error

1 = 该数据字收到了一个奇偶校验错误

0 = 接收到的数据字没有出现奇偶校验错误

FRETSC — Frame Error / Transmit Special Character

对于读取，此位表示 DATA[R9：R0]中包含的当前接收到的数据字存在帧错误。对于写入，此位表示要传输一个中断字符或空闲字符，而不是 DATA[T9：T0]中的内容。T9 用于表示 0 时的中断字符，以及表示 1 时的空闲字符，则 DATA[T8：T0]的内容应该为零。

1 = 数据字收到帧错误，传输空闲或中断字符

0 = 接收到的数据字在读取时没有出现帧错误，在写入时传输一个正常字符

RXEMPT — Receive Buffer Empty

当接收缓冲区中没有数据时，此位会起作用。此字段不考虑包含在接收移位寄存器中的数据。

1 = 接收缓冲区为空，读取时返回的数据无效

0 = 接收缓冲区包含有效的数据

IDLINE — Idle Line

此位表示在接收 DATA[9:0]中的字符之前接收线处于空闲状态。与 IDLE 标志不同，这个位可以为接收器第一次接收的第一个字符设置。

1 = 接收器在接收此字符之前已空闲

0 = 接收器在接收此字符之前没有空闲状态

R9T9 — Read receive data buffer 9 or write transmit data buffer 9
R8T8 — Read receive data buffer 8 or write transmit data buffer 8
R7T7 — Read receive data buffer 7 or write transmit data buffer 7
R6T6 — Read receive data buffer 6 or write transmit data buffer 6
R5T5 — Read receive data buffer 5 or write transmit data buffer 5
R4T4 — Read receive data buffer 4 or write transmit data buffer 4
R3T3 — Read receive data buffer 3 or write transmit data buffer 3
R2T2 — Read receive data buffer 2 or write transmit data buffer 2
R1T1 — Read receive data buffer 1 or write transmit data buffer 1
R0T0 — Read receive data buffer 0 or write transmit data buffer 0

20.7.2.9. SCI 匹配地址寄存器（SCI_MATCH）

地址：SCIn_BASEADDR + 0x0000_0020

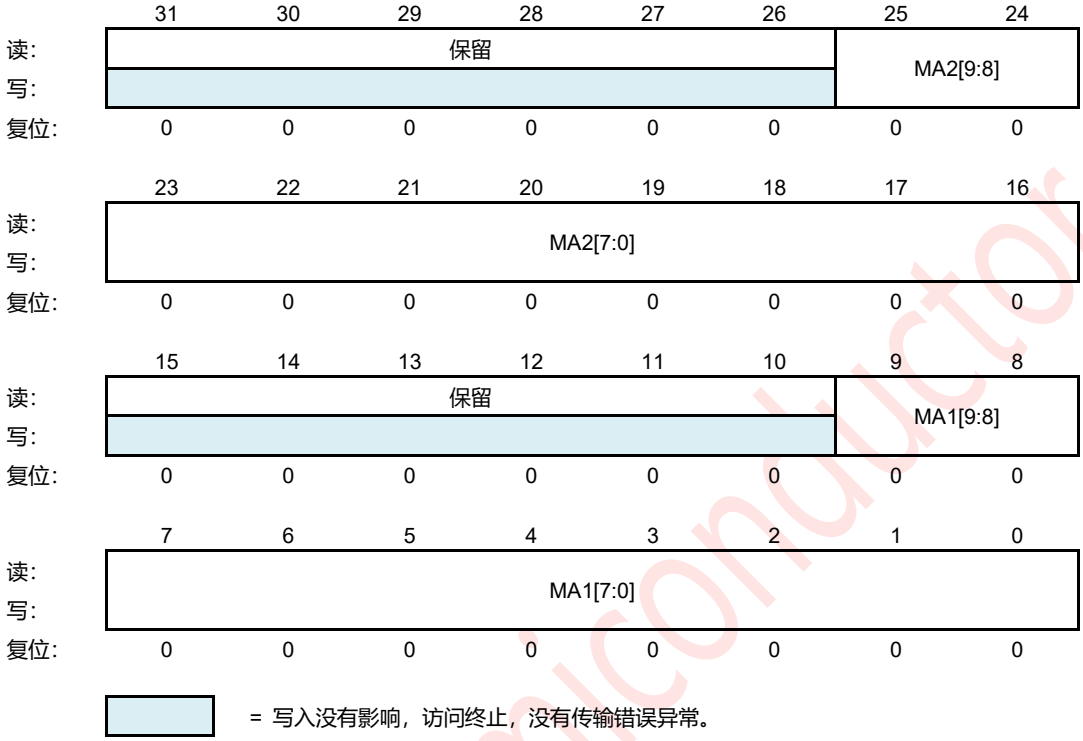


图 20-11：SCI 匹配地址寄存器（SCI_MATCH）

MA2[9:0] — Match Address 2

当设置了最显着的位并设置了相关的 BAUD[MAEN]位时，将 MA1 和 MA2 寄存器与输入数据地址进行比较。如果发生匹配，以下数据将被传输到数据寄存器。如果匹配失败，则将丢弃以下数据。仅在关联的 BAUD[MAEN]位被清除时，软件才应写入 MA 寄存器。

MA1[9:0] — Match Address 1

当设置了最显着的位并设置了相关的 BAUD[MAEN]位时，将 MA1 和 MA2 寄存器与输入数据地址进行比较。如果发生匹配，以下数据将被传输到数据寄存器。如果匹配失败，则将丢弃以下数据。仅在关联的 BAUD[MAEN]位被清除时，软件才应写入 MA 寄存器。

20.7.2.10. SCI 调制解调器 IrDA 寄存器（SCI_MODIR）

地址：SCIn_BASEADDR + 0x0000_0024

	31	30	29	28	27	26	25	24
读:	保留							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	保留					IREN	TNP	
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	RTSWATER							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	保留		TXCTSSRC	TXCTSC	RXRTSE	TXRTSPOL	TXRTSE	TXCTSE
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 20-12：SCI 调制解调器 IrDA 寄存器（SCI_MODIR）

IREN — Infrared enable

此位可启用/禁用红外线调制/解调。

1 = IR 启用

0 = IR 被禁用

TNP — Transmitter narrow pulse

该位允许 SCI 是否传输 1/OSR、2/OSR、3/OSR 或 4/OSR 窄脉冲。

0 0 = 1/OSR;
0 1 = 2/OSR;
1 0 = 3/OSR;
1 1 = 4/OSR

RTSWATER — Receive RTS Configuration

此位根据可以存储在接收 FIFO 中的附加字符数来配置 RX RTS 输出被否定的点。当配置为 0 时，当检测到将导致 FIFO 变满的字符的起始位时，RTS 将否定。

1 = 当接收 FIFO 小于或接收 FIFO 等于 RXWATER 配置时，RTS 起作用，当接收 FIFO 大于 RXWATER 配置时否定。

0 = 当接收方 FIFO 满或接收到导致 FIFO 满的字符时，RTS 起作用。

TXCTSSRC — Transmit CTS Source

此位配置 CTS 输入的源。

1 = CTS 输入是反向接收器匹配结果

0 = CTS 输入为 SCI_CTS 引脚

TXCTSC — Transmit CTS Configuration

此位配置是否在每个字符开始时检查或仅在发送器空闲时检查 CTS 状态。

1 = 在发送器空闲时，对 CTS 输入进行采样

0 = CTS 输入在每个字符的开头进行采样

RXRTSE — Receiver request-to-send enable

这个位允许 RTS 输出控制发送设备的 CTS 输入，以防止接收器溢出。不要同时设置 RXRTSE 和 TXRTSE。

1 = 如果接收数据寄存器满或检测到起始位将导致接收数据寄存器满，则取消 RTS。如果接收数据寄存器没有满，并且没有检测到会导致接收数据寄存器变为满的开始位，则 RTS 产生作用。RTS 的产生由 RTSWATER 字段配置

0 = 接收器对 RTS 没有影响

TXRTSPOL — Transmitter request-to-send polarity

这个位控制发送器 RTS 的极性。TXRTSPOL 不影响接收器 RTS 的极性。在活动的低状态下，RTS 将保持否定，除非设置了 TXRTSE。

1 = 发送器 RTS 是高位动作

0 = 发送器 RTS 是低位动作

TXRTSE — Transmitter request-to-send enable

这个位控制传输前后的 RTS。

1 = 当一个字符被放入一个空的发送器数据缓冲区时，RTS 在传输起始位之前产生一个位时间。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）完全发送后取消一位时间。

0 = 发送器对 RTS 没有影响

TXCTSE — Transmitter clear-to-send enable

TXCTSE 控制变送器的操作。TXCTSE 可以独立于 TXRTSE 和 RXRTSE 的状态进行设置。

1 = 清除发送操作。每次它准备好发送一个字符时，发送器都会检查 CTS 的状态。如果产生了 CTS，则会发送该字符。如果取消了 CTS，信号 TXD 保持标记状态，传输延迟，直到 CTS 产生。发送的 CTS 的变化不影响其传输

0 = CTS 对发送器没有影响

20.7.2.11. SCI FIFO 寄存器（SCI_FIFO）

地址：SCIn_BASEADDR + 0x0000_0028

	31	30	29	28	27	26	25	24
读:	保留							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	TXEMPT	RXEMPT	保留				TXOF	RXUF
写:							w1c	w1c
复位:	1	1	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0	0	保留	Rxiden			TXOFE	RXUFE
写:	TXFLUSH	RXFLUSH						
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	TXFE	TXFIFOSIZE[2:0]			RXFE	RXFIFOSIZE[2:0]		
写:								
复位:	0	0	1	0	0	0	1	0

= 写入没有影响，访问终止，没有传输错误异常。

w1c = 写 1 到该位将清除它

图 20-13：SCI FIFO 寄存器（SCI_FIFO）

TXEMPT — Transmit Buffer/FIFO Empty

当传输 FIFO/缓冲区中没有数据时，此位起作用。此字段不考虑包含在传输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 — Receive Buffer/FIFO Empty

当接收 FIFO/缓冲区中没有数据时，此位起作用。此字段不考虑包含在接收移位寄存器中的数据。

1 = 接收缓冲区为空

0 = 接收缓冲区不为空

TXOF — Transmitter Buffer Overflow Flag

这位表示写入传输缓冲区的数据超过了它所能容纳的数据。此字段将起作用，而不管 TXOFE 的值如何。但是，只有在设置了 TXOFE 时，才会向主机发出一个中断。通过写入一个 1，可以清除此标志。

1 = 自上次清除该标志以来，至少发生了一次传输缓冲区溢出

0 = 自上次清除该标志以来，没有发生传输缓冲区溢出

RXUF — Receiver Buffer Underflow Flag

此位表示从接收缓冲区读取的数据比现有的数据更多。此字段将起作用，而不管 RXUFE 的值如何。但是，只有在设置了 RXUFE 时，才会向主机发出一个中断。通过写入一个 1，可以清除此标志。

1 = 自上次清除该标志以来，至少发生了一个接收缓冲区下溢(Underflow)

0 = 自上次清除该标志以来，没有发生接收缓冲区下溢

TXFLUSH — Transmit FIFO/Buffer Flush

写入此字段会导致存储在传输 FIFO/缓冲区中的所有数据被刷新。这不会影响传输移位寄存器中的数据。

1 = 传输 FIFO/缓冲区中的所有数据都将被清除

0 = 不发生冲洗操作 (Flush Operation)

RXFLUSH — Receive FIFO/Buffer Flush

写入此字段会导致存储在接收 FIFO/缓冲区中的所有数据被刷新。这不会影响接收移位寄存器中的数据。

1 = 接收的 FIFO/缓冲区中的所有数据都将被清除

0 = 不发生冲洗操作

RXIDEN — Receiver Idle Empty Enable

当设置时，当接收器空闲了一些空闲字符且 FIFO 不是空时，它将启用 RDRF 的作用。

000 = Disable RDRF assertion due to partially filled FIFO when receiver is idle;
001 = Enable RDRF assertion due to partially filled FIFO when receiver is idle for 1 character;
010 = Enable RDRF assertion due to partially filled FIFO when receiver is idle for 2 characters;
011 = Enable RDRF assertion due to partially filled FIFO when receiver is idle for 4 characters;
100 = Enable RDRF assertion due to partially filled FIFO when receiver is idle for 8 characters;
101 = Enable RDRF assertion due to partially filled FIFO when receiver is idle for 16 characters;
110 = Enable RDRF assertion due to partially filled FIFO when receiver is idle for 32 characters;
111 = Enable RDRF assertion due to partially filled FIFO when receiver is idle for 64 characters.

TXOFE — Transmit FIFO Overflow Interrupt Enable

当设置此字段时，TXOF 标志将生成一个对主机的中断。

1 = TXOF 标志生成一个对主机的中断

0 = TXOF 标志不会对主机产生中断

RXUFE — Receive FIFO Underflow Interrupt Enable

当设置此字段时，RXUF 标志将生成一个对主机的中断。

1 = TRXUF 标志生成一个对主机的中断

0 = RXUF 标志不会对主机产生中断

TXFE — Transmit FIFO Enable

设置此字段后，将启用传输缓冲区(Buffer) 的内置 FIFO 结构。FIFO 结构的大小用 TXFIFOSIZE 表示。如果没有设置此字段，传输缓冲区作为深度为一个数据字的 FIFO 操作，而不管 TXFIFOSIZE 的值。在更改此字段之前，必须同时选择 CTRL[TE]和 CTRL[RE]。

1 = 传输 FIFO 已启用。缓冲区用 TXFIFOSIZE 表示

0 = 传输 FIFO 被禁用。缓冲区为深度 1。 (Legacy Support)

TXFIFOSIZE[2:0] — Transmit FIFO/Buffer Depth

此字段表示可以存储在传输缓冲区中的最大传输数据字数。此字段为只读状态。

000 = 传输 FIFO/Buffer 深度 = 1 个数据字;

001 = 传输 FIFO/Buffer 深度 = 4 个数据字;

010 = 传输 FIFO/Buffer 深度 = 8 个数据字;

011 = 传输 FIFO/Buffer 深度 = 16 个数据字;

100 = 传输 FIFO/Buffer 深度 = 32 个数据字;

101 = 传输 FIFO/Buffer 深度 = 64 个数据字;

110 = 传输 FIFO/Buffer 深度 = 128 个数据字;

111 = 传输 FIFO/Buffer 深度 = 256 个数据字。

RXFE — Receive FIFO Enable

设置此字段后，启用接收缓冲区的 FIFO 结构。FIFO 结构的大小由 RXFIFOSIZE 表示。如果未设置此字段，则接收缓冲区作为深度为一个数据字的 FIFO 操作，而不管 RXFIFOSIZE 中的值如何。在更改此字段之前，必须同时选择 CTRL[TE]和 CTRL[RE]。

1 = 接收 FIFO 已启用。缓冲区是由 RXFIFOSIZE 表示的深度

0 = 接收 FIFO 被禁用。缓冲区为深度 1。 (Legacy support)

RXFIFOSIZE[2:0] — Receive FIFO/Buffer Depth

可以存储在接收缓冲区中的最大传输数据字数。此字段为只读状态。

000 = 接收 FIFO/缓冲区深度 = 1 个数据字;

001 = 接收 FIFO/缓冲区深度 = 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/缓冲区深度 = 256 个数据字；

20.7.2.12. SCI 水印寄存器（SCI_WATER）

地址：SCIn_BASEADDR + 0x0000_002C

	31	30	29	28	27	26	25	24
读:	RXCOUNT[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	RXWATER[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	TXCOUNT[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	TXWATER[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 20-14：SCI 水印寄存器（SCI_WATER）

RXCOUNT[7:0] — Receive Counter

此寄存器中的值表示接收 FIFO/缓冲区中的数据字数。如果正在接收到一个数据字，即在接收移位寄存器中，则它不包括在计数中。此值可与 FIFO[RXFIFOSIZE]一起使用，以计算在接收 FIFO/缓冲区中还有多少空间。

RXWATER[7:0] — Receive Watermark

当接收 FIFO/缓冲区中的数据字数大于此寄存器字段中的值时，将生成一个中断。为了正确操作，RXWATER 中的值必须设置为小于 FIFO[FIFO 尺寸]和 FIFO[RXFE]所指示的接收 FIFO/缓冲器大小，并且必须大于 0。

TXCOUNT[7:0] — Transmit Counter

该寄存器中的值表示传输 FIFO/缓冲区中的数据字数。如果正在传输一个数据字，即在传输移位寄存器中，则它不包括在计数中。此值可与 FIFO[TXFIFOSIZE]一起使用，以计算在传输的 FIFO/缓冲区中还有多少空间。

TXWATER[7:0] — Transmit Watermark

当发送 FIFO/缓冲区中的数据字数等于或小于该寄存器字段中的值时，将生成一个中断。为了正确操作，TXWATER 中的值必须设置为小于传输缓冲区的大小/由[FIFO 大小]和 FIFO 大小[TXFE]所示的 FIFO 大小。

20.7.2.13. SCI 过采样比寄存器（SCI_OSR）

地址：SCIn_BASEADDR + 0x0000_0030

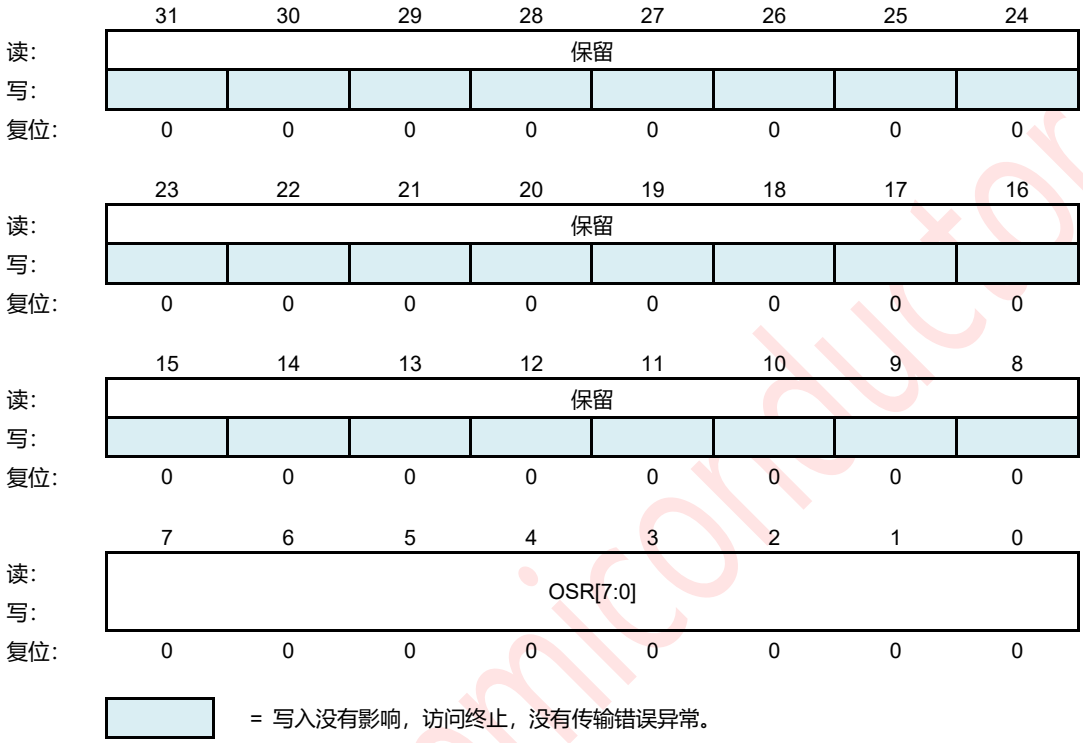


图 20-15：SCI 过采样比寄存器（SCI_OSR）

OSR[7:0] — Oversampling Ratio

该字段为接收器配置 4x（00000011）和 256x（111111111）之间的过采样比。写入无效的过采样比（例如，不介于 4x 和 256x 之间的值）将默认为过采样比 16（00001111）。只有当发送器和接收器都被禁用时，OSR 字段。过采样比 = OSR + 1。

20.8. 功能描述

SCI 支持全双工、异步和 NRZ 串行通信，并包括一个波特率发生器、发送器和接收器块。发送器和接收器独立工作，尽管它们使用相同的波特率发生器。下面描述了 SCI 的每个方块。

20.9. 波特率生成

波特率发生器中的一个 13 位模量计数器推导出接收器和发送器的波特率。从 1 到写入 SBR[12:0]的 8191 的值确定了异步 SCI 波特时钟的波特时钟除数。SBR 位在 SCI 波特率寄存器、BDH 和 BDL 中。波特率时钟驱动接收器，而发送器则由波特率时钟除以过采样比来驱动。根据过采样比，接收器的采集速率为 4 到 256 个样本。

波特率的产生有两个误差来源：

- 异步 SCI 波特时钟的整数除法可能不能给出准确的目标频率。
- 与异步 SCI 波特时钟的同步可能会导致相移

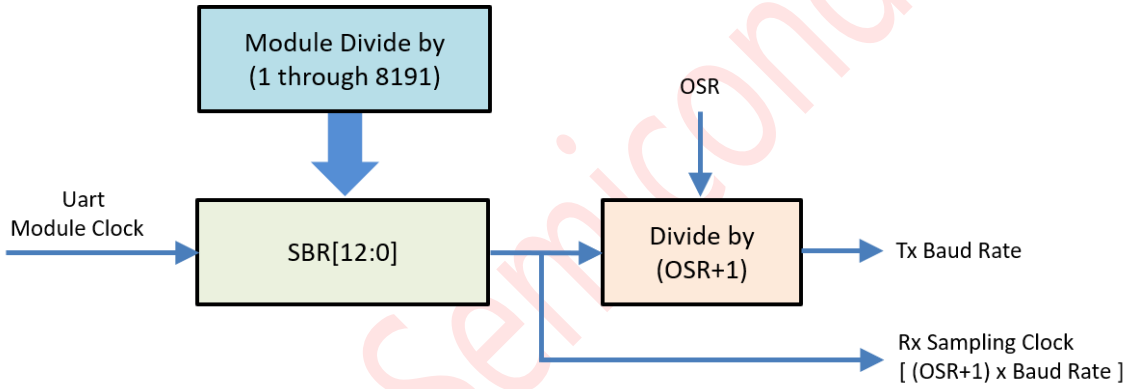


图 20-16：SCI 波特率生成

$$\text{Baud Rate} = \frac{\text{Uart Module Clock}}{\text{SBR[12:0]} \times (\text{OSR+1})}$$

提示：如果 SBR[12:0] = 0，则关闭波特率生成器。

20.10. 发送器功能描述

本章节描述了 SCI 发送器的整体方块图，以及用于发送中断和空闲字符的专门功能。

变送器输出（TXD）空闲状态默认为逻辑高，CTRL[TXINV]在复位后被清除。通过设置 CTRL[TXINV]来反转发送器输出。通过设置 CTRL[TE]位来启用发送器。这将为一个属于空闲状态的完整字符框架的前导字符进行排队。然后发送器保持空闲，直到发射数据缓冲区中的数据可用。程序通过写入 SCI 数据寄存器，将数据存储到传输数据缓冲区中。

SCI 发送器的中心组件是发射移位寄存器，它从 10 位到 13 位长，这取决于 CTRL[M]、BAUD[M10]和 BAUD[SBNS]控制位的设置。对于本章节的其余部分，假定 CTRL[M]、BAUD[M10]和 BAUD[SBNS]已被清除，并选择正常的 8 位数据模式。在 8 位数据模式中，移位寄存器包含一个起始位、8 个数据位和一个停止位。当传输移位寄存器可用于新字符时，在传输数据寄存器中等待的值被传输到移位寄存器，与波特率时钟同步，并且发送数据寄存器空（STAT[TDRE]）状态标志被设置为指示另一个字符可以写入传输数据缓冲区。

如果在停止位移出 TXD 引脚后，没有在传输数据缓冲区中等待新字符，则发送机设置发送完整标志并进入空闲模式，TXD 高，等待更多字符传输。

将 0 写入 CTRL[TE]并不会立即禁用发送器。当前正在进行的传输活动必须首先完成（这可能包括一个数据字符、空闲字符或中断字符），尽管发送器将不会开始传输另一个字符。

20.10.1. 发送休息和排队的空闲时间

SCI_CTRL[SBK]位发送中断字符，最初用于获得旧电传类型接收器的注意。中断字符是逻辑 0 的完整字符时间，10 位到 12 位的时间，包括开始位和停止位。通过设置 SCI_STAT[BRK13]，可以启用更长的 13 位时间的中断。通常，程序会等待 SCI_STAT[TDRE]设置为指示消息的最后一个字符已移动到传输移位器，写入 1，然后将 0 写入 SCI_CTRL[SBK]位。一旦转换器可用，此动作将排队排列要发送的中断字符。如果 SCI_CTRL[SBK]在排队中断移动到移位器时保持 1，同步到波特率时钟，则另一个中断字符排队。如果接收设备是另一个飞思卡尔半导体 SCI，则在所有数据位中接收到中断字符为 0s，并发生帧错误（SCI_STAT[FE] = 1）。

还可以通过写入设置为 13 位和数据位被清除的 SCI_DATA 寄存器来传输中断字符。这支持将中断字符作为正常数据流的一部分进行传输。

当使用空闲行唤醒时，消息之间需要整个空闲字符时间（逻辑 1）来唤醒任何休眠的接收器。通常，程序会等待 SCI_STAT[TDRE]设置为指示消息的最后一个字符已移动到传输移位器，然后写入 0，然后将 1 写入 SCI_CTRL[TE]位。一旦转换器可用，此操作将立即发送空闲字符。只要在 SCI_CTRL[TE]被清除时，移位器中的字符没有完成，SCI 发送器就不会真正释放对 TXD 引脚的控制。

空闲字符也可以通过将第 13 位和数据位写入 SCI_DATA 寄存器来传输。这支持将空闲字符作为正常数据流的一部分进行传输。

断裂字符的长度受到 SCI_STAT[BRK13]、SCI_CTRL[M]、SCI_BAUD[M10]和 SCI_BAUD[SNBS]位的影响，如下表所示。

表 20-3：断开字符长度

BRK13	M	M10	SBNS	Break Character Length
0	0	0	0	10-bits times
0	0	0	1	11-bits times
0	1	0	0	11-bits times
0	1	0	1	12-bits times
0	X	1	0	12-bits times
0	X	1	1	13-bits times
1	0	0	0	13-bits times
1	0	0	1	13-bits times
1	1	0	0	14-bits times
1	1	0	1	14-bits times
1	X	1	0	15-bits times
1	X	1	1	15-bits times

20.11. 接收器功能说明

在本章节中，接收器方块图是整体接收器功能描述的指南。接下来，将更详细地描述了用于重构接收数据的数据采样技术。最后，解释了接收器唤醒功能的不同变化。

通过设置 SCI_STAT[RXINV]来反转接收器输入。通过设置 SCI_CTRL[RE]位来启用接收器。字符帧由逻辑 0 的起始位、8 到 10 个数据位（MSB 或 LSB 优先）和逻辑 1 的一个或两个停止位组成。有关 9 位或 10 位数据模式的信息，请参阅 8 位、9 位和 10 位的数据模式。在本讨论的其余部分中，假设 SCI 被配置为正常的 8 位数据模式。

在接收到停止位进入接收移动器后，如果接收数据寄存器尚未满，则将数据字符传输到接收数据寄存器，并设置接收数据寄存器满（SCI_STAT[RDRF]）状态标志。如果 SCI_STAT 已设置[RDRF]，则表示接收数据寄存器（缓冲区）已满，已设置溢出（OR）状态标志，且新数据丢失。因为 SCI 接收器是双缓冲的，所以在必须读取接收数据缓冲器中的数据之前，在设置了 SCI_STAT[RDRF]之前，程序有一个完整的字符时间，以避免接收器溢出。

当程序检测到接收数据寄存器已满时（SCI_STAT[RDRF] = 1），它通过读取 SCI_DATA 从接收数据寄存器获取数据。有关标志清除的详细信息，请参阅中断标志和状态标志。

20.11.1. 数据采样技术

SCI 接收器支持 4×到 256×的波特率时钟的可配置过采样率。接收器首先以过采样率乘以波特率进行逻辑级采样，以搜索 RXD 串行数据输入引脚上的下降边。一个下降边被定义为经过三个连续的逻辑 1 样本之后的逻辑 0 样本。过采样波特率时钟将比特时间从 1 到 OSR 分成 4 到 256 段（其中 OSR 是配置的过采样比）。当找到一个下降的边缘时，将在（OSR/2）、（OSR/2）+ 1 和（OSR/2）+ 2 处再采集三个样本，以确保这是一个真正的起始位，而不仅仅是噪声。如果这三个样本中至少有两个为 0，则接收方假定它与接收到的字符同步。如果在认为接收器同步之前检测到另一个下降边，则接收器从第一段重启采样。

然后，接收器在（OSR/2）、（OSR/2）+ 1 和（OSR/2）+ 2 处采样每个位时间，包括开始位和停止位，以确定该位的逻辑级别。逻辑级别被解释为在比特时间内采集的大部分样本。如果在字符帧中的任何位时间，包括开始位和停止位，与该位的逻辑级别不一致，当接收字符传输到接收数据缓冲区时设置噪声标志（SCI_STAT[NF]）。

当 SCI 接收器被配置为在波特率时钟的两边进行采样时，每个接收位中的段数有效地增加了一倍（从 1 到 OSR×2）。然后在 OSR、OSR + 1 和 OSR + 2 处对起始位和数据位进行采样。对于 4×到 7×的过采样率，必须启用时钟两侧的采样，并且对于更高的过采样率是可选的。

下降边检测逻辑不断地寻找下降边。如果检测到边缘，则样本时钟将重新同步到位时间（除非已禁用重新同步）。这提高了接收器在存在噪声或不匹配的波特率的情况下的可靠性。它不会改善最坏情况分析，因为某些字符在字符框架中的任何地方都没有任何额外的下降边。

在帧错误的情况下，如果接收到的字符不是断开字符，则搜索下降边的采样逻辑充满三个逻辑 1 样本，以便几乎可以立即检测到新的开始位。

20.11.2. 接收器唤醒操作

接收器唤醒和接收器地址匹配是一种硬件机制，它允许 SCI 接收器忽略针对不同接收器的消息中的字符。

在接收器唤醒期间，所有接收器评估每个消息的第一个字符(s)，一旦他们确定消息是针对不同的接收器，他们将逻辑 1 写入接收器唤醒控制位（SCI_CTRL[RWU]）。当设置 RWU 位和 SCI_S2[RWUID]位时，与接收器相关的状态标志，除了空闲位的空闲位，被禁止设置，从而消除了处理不重要消息字符的软件开销。在消息结束或下一个消息的开始，所有接收器自动强制 SCI_CTRL[RWU]为 0，以便所有接收器及时醒来以查看下一个消息的第一个字符(s)。

在接收器地址匹配过程中，在硬件中进行地址匹配，SCI 接收器将忽略所有不满足地址匹配要求的字符。

接收器唤醒受到 SCI_CTRL[RWU]、SCI_MATCH [MA1]、SCI_MATCH [MA2]、SCI_BAUD[MATCFG]和 SCI_STAT[RWUID]位的影响，如下表所示。

表 20-4：接收器唤醒选项

RWU	MA1 | MA2	MATCFG	WAKE: RWUID	Receiver Wakeup
0	0	X	X	Normal operation
1	0	00	00	Receiver wakeup on idle line, IDLE flag not set
1	0	00	01	Receiver wakeup on idle line, IDLE flag set
1	0	00	10	Receiver wakeup on address mark
1	1	11	X0	Receiver wakeup on data match
0	1	00	X0	Address mark address match, IDLE flag not set for discarded

RWU	MA1 | MA2	MATCFG	WAKE: RWUID	Receiver Wakeup
				characters
0	1	00	X1	Address mark address match, IDLE flag set for discarded characters
0	1	01	X0	Idle line address match
0	1	10	X0	Address match on and address match off, IDLE flag not set for discarded characters
0	1	10	X1	Address match on and address match off, IDLE flag set for discarded characters

20.11.2.1. 急速排队

当唤醒被清除时，接收器被配置为空闲线路唤醒。在此模式下，当接收方检测到空闲线级别的完整字符时间时，将自动清除 SCI_CTRL[RWU]。SCI_CTRL[M]和 SCI_BAUD[M10]控制位选择 8 位到 10 位数据模式，SCI_BAUD[SBNS]位选择 1 位或 2 位停止位数，决定需要多少位空闲，因为开始位和停止位需要 10 到 13 位。

当 SCI_CTRL[RWU]为 1 且 SCI_STAT[RWUID]为零时，唤醒接收方的空闲条件不会设置 SCI_STAT[IDLE]标志。接收方唤醒并等待下一个消息的第一个数据字符，该消息设置了 SCI_STAT[RDRF]标志，并在启用时生成一个中断。当 SCI_STAT[RWUID]为 1 时，任何空闲条件都会设置 SCI_STAT[IDLE]标志，并在启用时生成一个中断，而不管 SCI_CTRL[RWU]为 0 还是 1。

空闲线类型（SCI_CTRL[ILT]）控制位选择两种方法之一来检测空闲线。当 SCI_CTRL[ILT]被清除时，空闲位计数器在开始位之后开始，因此停止位和字符结束计数时的任何逻辑 1 接近空闲的全部字符时间。当设置 SCI_CTRL[ILT]时，空闲位计数器直到停止位时间之后才开始，因此空闲检测不受上一条消息最后一个字符中的数据的影响。

20.11.2.2. 地址标记唤醒

设置 SCI_CTRL[WAKE]时，接收器配置为地址标记唤醒。在此模式下，当接收方检测到接收字符的最重要位中的逻辑 1 时，SCI_CTRL 将自动被[RWU]清除。

地址标记唤醒允许消息包含空闲字符，但需要保留 MSB 以便在地址帧中使用。地址帧的 MSB 中的逻辑 1 在接收到停止位之前清除 SCI_CTRL[RWU]位，并设置 SCI_STAT[RDRF]标志。在这种情况下，即使接收器在字符大部分时间睡觉，MSB 集的字符也会被接收。

20.11.2.3. 数据匹配唤醒

当设置了 SCI_CTRL[RWU]并且 SCI_BAUD[MATCFG]等于 11 时，接收器被配置为数据匹配唤醒。在此模式下，当接收方检测到设置 BAUD[MAEN1]时匹配[MA1]字段，或设置 BAUD[MAEN2]时匹配[MA2]的字符时，将自动清除 SCI_CTRL[RWU]。

20.11.2.4. 地址匹配操作

当设置了 SCI_BAUD[MAEN1]或 SCI_BAUD[MAEN2]位，并且 SCI_BAUD[MATCFG]等于 00 时，将启用地址匹配操作。在这个功能中，RXD 引脚通过逻辑 1 接收到的一个字符被认为是一个地址，并与相关的匹配[MA1]或匹配[MA2]字段进行比较。如果比较匹配，字符只转移到接收缓冲区，并设置 SCI_STAT[RDRF]。在停止位之前的位位置用逻辑 0 接收的所有后续字符被认为是与地址相关联的数据，并被传输到接收数据缓冲区。如果没有标记的地址匹配，则不会传输到接收数据缓冲区，并且在停止位之前的位位置的逻辑为零的以下字符也将被丢弃。如果 SCI_BAUD[MAEN1]和 SCI_BAUD[MAEN2]位都被否定，则接收器将正常工作，并且接收到的所有数据都将被传输到接收数据缓冲区。

匹配[MA1]和匹配[MA2]字段的相同方式的地址匹配操作功能。

- 如果只 SCI_BAUD[MAEN1]和 SCI_BAUD[MAEN2]中的一个起作用，则只将标记的地址与关联的匹配寄存器进行比较，并且仅在匹配时将数据传输到接收数据缓冲区。
- 如果 SCI_BAUD[MAEN1]和 SCI_BAUD[MAEN2] 都起作用，则将标记的地址与两个匹配寄存器进行比较，并且仅在与任何一个寄存器的匹配上传输数据。

20.11.2.5. 空闲匹配操作

当设置了 SCI_BAUD[MAEN1]或 SCI_BAUD[MAEN2]位，并且 SCI _BAUD[MATCFG]等于 01 时，将启用空闲匹配操作。在这个功能中，在空闲行条件之后由 RXD 引脚接收到的第一个字符被认为是一个地址，并与关联的 MA1 或 MA2 寄存器进行比较。如果比较匹配，字符只转移到接收缓冲区，并设置 SCI_STAT[RDRF]。所有后续字符都被认为是与地址关联的数据，并被传输到接收数据缓冲区，直到检测到下一个空闲行状态。如果没有地址匹配，则不会向接收数据缓冲区传输，在下一个空闲状态之前的所有帧也将被丢弃。如果 SCI_BAUD[MAEN1]和 SCI_BAUD[MAEN2]位都被否定，则接收器将正常工作，并且接收到的所有数据都将被传输到接收数据缓冲区。

对于 MA1 和 MA2 寄存器，以相同的方式操作。

- 如果只 SCI_BAUD[MAEN1]和 SCI_BAUD[MAEN2]中的一个起作用，则只将空闲行之后的第一个字符与关联的匹配寄存器进行比较，并且仅在匹配时将数据传输到接收数据缓冲区。
- 如果 SCI_BAUD[MAEN1]和 SCI_BAUD[MAEN2] 都起作用，则将空闲行之后的第一个字符与两个匹配寄存器进行比较，并且只在与任何一个寄存器的匹配上传输数据。

20.11.2.6. 匹配匹配关闭操作

当设置 SCI_BAUD[MAEN1]和 SCI_BAUD[MAEN2]且 SCI_BAUD[MATCFG]等于 10 时，将启用 “匹配打开，匹配关闭” 操作。在此功能中，接收到匹配匹配的 RXD 引脚[MA1]的字符并将其转移到接收缓冲区，并设置 SCI_STAT[RDRF]。所有后续字符都被认为是数据，也被传输到接收数据缓冲区，直到接收到与匹配[MA2]寄存器匹配的字符。丢弃匹配[MA2]的字符，直到收到另一个匹配[MA1]匹配的字符。如果 SCI_BAUD[MAEN1]和 SCI_BAUD[MAEN2]位都被否定，则接收器将正常工作，并且接收到的所有数据都将被传输到接收数据缓冲区。

20.11.3. 红外解码器

红外解码器将接收到的字符从 IrDA 格式转换为接收器使用的 NRZ 格式。它也有一个 OSR 过采样波特率时钟计数器，过滤噪声，并指示何时接收到 1。

20.11.3.1. 开始位检测

当清除 STAT[RXINV]时，接收字符的第一个下降边对应于起始位。红外解码器复位其计数器。此时，接收器也开始了它的开始位检测过程。在检测到开始位后，接收器将其位时间同步到此开始位时间。对于其余的字符接收，红外解码器的计数器和接收器的位时间计数器的计数相互独立。

20.11.3.2. 噪声滤波

在红外解码器计数器的前半部分中检测到的任何进一步上升的边缘都被解码器忽略。任何小于一个过采样波特时钟的脉冲都不能被它检测到，无论它是在计数的前半部分或后半部分被看到。

20.11.3.3. 低位检测

在解码器计数的后半部分，一个上升的边被解码为一个 0，它被发送到接收器。解码器计数器也被复位。

20.11.3.4. 高比特检测

在 OSR 的过采样波特率时钟后，如果没有看到一个上升的边缘，则解码器向接收器发送一个 1。

如果下一位是 0，到达较晚，则根据低位检测检测到低位。发送到接收方的值将从 1 更改为 0。然后，如果噪声脉冲发生在接收器的比特时间采样周期之外，则 0 的延迟不被记录为噪声。

20.12. 额外的 SCI 功能

下面的部分描述了额外的 SCI 功能。

20.12.1. 8 位、9 位和 10 位的数据模式

通过设置 SCI_ CTRL[M]或通过设置 SCI_ BAUD[M10]的 10 位数据模式，SCI 发送器和接收器可以配置为以 9 位数据模式运行。在 9 位模式下，有第九个数据位，而在 10 位模式下，有第 10 个数据位。对于传输数据缓冲区，这些位被存储在 SCI_CTRL[T8]和 SCI_CTRL[T9]中。对于接收器，这些位被保存在 SCI_CTRL[R8]和 SCI_CTRL[R9]中。它们也可以通过对 SCI_DATA 寄存器的 16 位或 32 位的访问来访问。

对于对传输数据缓冲区的相干 8 位写入，请在写入 SCI_DATA[7:0]之前写入 SCI_CTRL[T8]和 SCI_CTRL[T9]。对于对 SCI_DATA 寄存器的 16 位和 32 位写入，所有 10 个传输位同时被写入传输数据缓冲区。

如果作为新字符的第九位和第十位传输的位值与前一个字符的位值相同，则无需再次写入 SCI_CTRL[T8]和 SCI_CTRL[T9]。当数据从传输数据缓冲区传输到传输移位器时，将复制 SCI_CTRL[T8]和 SCI_CTRL[T9]中的值，同时将数据从 SCI_DATA[7:0]传输到移位器。

9 位数据模式通常与奇偶校验一起使用，以允许 8 位数据加上第九位的奇偶校验，或者它与地址标记唤醒一起使用，因此第九位数据位可以作为唤醒位。10 位数据模式通常与奇偶校验和地址标记唤醒一起使用，因此第九数据位可以作为唤醒位，第十数据位可以作为奇偶校验位。在自定义协议中，第九位和/或第十位也可以作为软件控制的标记。

20.12.2. 空闲长度

空闲字符是指起始位、所有数据位和停止位都处于标记位置的字符。CTRL[ILT]寄存器可以配置为从前一开始位（指向空闲字符检测的任何数据位和停止位计数）或从前一停止位开始检测空闲字符。

还可以使用 CTRL[IDLECFG]字段对在检测到空闲行条件之前必须接收到的空闲字符数进行配置。此字段配置在设置 STAT[IDLE]标志、清除 STAT[RAF[RAF]标志和 DATA[IDLINE]标志使用下一个接收字符设置之前必须接收的空闲字符数。

空闲线唤醒和空闲匹配操作也会受到 CTRL 字段的影响。当启用地址匹配或匹配开/关操作时，设置 STAT[RWUID]位将导致将丢弃的字符视为空闲字符。独立于外部系统中的连接，以帮助隔离系统问题。在这种模式下，发送器输出内部连接到接收器输入，RXD 引脚不使用 SCI。

20.12.3. 单线操作

当设置 SCI_CTRL[循环]时，同一寄存器中的 SCI_CTRL[RSRC]位会在循环模式（SCI_CTRL[RSRC] = 0）和单线模式(SCI_CTRL[RSRC] = 1]之间进行选择。循环模式有时用于检查软件，独立于外部系统中的连接，以帮助隔离系统问题。在这种模式下，发送器输出内部连接到接收器输入，RXD 引脚不使用 SCI。

20.12.4. 循环模式

当设置了 SCI_CTRL[循环]时，在同一寄存器中的 RSRC 位会在循环模式（SCI_CTRL[RSRC] = 0）或单线模式（SCI_CTRL[RSRC] = 1）之间进行选择。单线模式实现了一个半双工串行连接。接收器内部连接到发送器输出和 TXD 引脚（不使用 RXD 引脚）。

在单线模式下，SCI_CTRL[TXDIR]位控制 TXD 引脚上的串行数据的方向。当 SCI_CTRL[TXDIR]被清除时，TXD 引脚是接收器的输入，发送器暂时与 TXD 引脚断开，以便外部设备可以向接收器发送串行数据。当设置 SCI_CTRL[TXDIR]时，TXD 引脚是由发送器驱动的输出，内环回连接被禁用，因此接收器无法接收发送器发送的字符。

20.13. 红外线接口

SCI 提供了将窄脉冲传输到红外 LED 并接收窄脉冲并将其转换为串行位的能力，并被发送到 SCI 的能力。IrDA 物理层规范定义了一个用于交换数据的半双工红外通信链路。完整的标准包括高达 16 Mbits/s 的数据速率。该设计仅涵盖了 2.4 kbits/s 到 115.2 kbits/s 之间的数据速率。

SCI 有一个红外发射编码器和接收解码器。SCI 传输由红外子模块编码的串行数据位，每一个零位传输一个窄脉冲。每一位不发送脉冲。当接收数据时，使用红外光二极管检测红外脉冲，并由来自 SCI 的外部的红外接收解码器转换到 CMOS 水平。然后，窄脉冲由红外接收解码器拉伸，以返回到要由 UART 接收的串行比特流。传输脉冲和预期接收脉冲的极性可以反向，从而可以直接连接到使用有源高脉冲的外部 IrDA 收发器模块。

红外线子模块从 SCI 接收其时钟源。在红外子模块中选择这两个时钟中的一个，以在传输过程中产生 1/OSR、2/OSR、3/OSR 或 4/OSR 的窄脉冲。

20.13.1. 红外发射编码器

红外发射编码器将数据从发射移位寄存器的串行位转换到 TXD 信号。一个窄脉冲传输零位，没有脉冲。窄脉冲在比特开始时发送，其持续时间为 1/OSR、2/OSR、3/OSR 或 4/OSR 的比特时间。当清除 SCI_CTRL[TXINV]时，发送零位窄低脉冲，而当设置 SCI_CTRL[TXINV]时，为零位发送窄高脉冲。

20.13.2. 红外接收解码器

红外接收块将来自 RXD 信号的数据转换到接收移位寄存器。每个接收到一个零都期望有一个窄脉冲，每个接收到一个都期望没有脉冲。设置零位 whenSCI_STAT[RXINV]时，预计清除窄低脉冲，而设置 SCI_STAT[RXINV]时，预计零位清除窄高脉冲。该接收解码器满足 IrDA 串行红外物理层规范所定义的边缘抖动要求。

20.14. 中断和状态标志

SCI 发送器有两个状态标志，可以选择性地生成硬件中断请求。发送数据寄存器为空的 SCI_（STAT[TDRE]）位表示发送数据缓冲区中是否有空间将另一个发送字符写入 SCI_DATA。如果设置了传输中断启用 SCI_CTRL[TIE])位，则在设置 SCI_STAT[TDRE]时请求硬件中断。传输完成（SCI_STAT[TC]）位表示传输机完成了所有数据、前导和断开字符的传输，且 TXD 处于非活动级别。这个标志通常用于具有调制解调器的系统，以确定何时关闭调制解调器是安全的。如果设置了发送完全中断启用（SCI_CTRL[TCIE]）位，则在设置 SCI_STAT[TC]时请求硬件中断。如果清除了相应的 SCI_CTRL[TIE]或 SCI_CTRL[TCIE]本地中断掩码，则可以使用软件轮询来监视 SCI_STAT[TDRE]和 SCI_STAT[TC]状态标志，而不是硬件中断。

当程序检测到接收数据寄存器已满时（SCI_STAT[RDRF] = 1），它通过读取 SCI_DATA 从接收数据寄存器获取数据。通过读取 SCI_DATA，将清除 SCI_STAT[RDRF]标志。

IDLE 状态标志包含了防止在 RXD 行长时间空闲时重复设置它的逻辑。通过将 1 写入 SCI _STAT[IDLE]标志来清除 IDLE。清除 SCI_STAT[IDLE]后，在接收方接收到至少一个新字符并设置了 SCI_STAT[RDRF]之前，才能再次设置它。

如果在接收到的字符中检测到关联的错误，导致设置 SCI_STAT[RDRF]，则错误标志-噪声标志（SCI_STAT[NF]）、帧错误（SCI_STAT[FE]）和奇偶校验错误标志（SCI_STAT[PF]）-与 SCI_STAT[RDRF]同时设置。这些标志不会在溢出的情况下设置。

如果当一个新字符准备从接收移位器传输到接收数据缓冲区时，已经设置了 SCI_STAT[RDRF]，则将设置溢出（SCI_STAT[OR]）标志，而不是伴随任何相关的 NF、FE 或 PF 条件的数据。

如果接收到的字符与匹配[MA1]和/或匹配[MA2]的内容匹配，则在设置 SCI_STAT[MA1F]和/或 SCI_STAT[RDRF]的同时设置 SCI_STAT[MA2F]标志。

在任何时候，RXD 串行数据输入引脚上的上升或下降缘动作都会导致设置 SCI_STAT[SCI_STAT]标志。通过写入 1 来清除 SCI_STAT 标志。此功能取决于已启用的接收器（SCI_CTRL[RE] = 1）。

21. 同步串行接口（SSI）

21.1. 功能介绍

SSI 是一个可编程的同步串行接口（SSI）外设。这是一个符合 AMBA2.0 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 还可以使用一组可选的 DMA 信号与 DMA 控制器进行接口，该信号可以在配置期间进行选择。

LT7589 的 SSI 模块提供了到 SPI Flash 或其他 SPI 设备的 QSPI0、QSPI1 和 QSPI2 接口。QSPI0 信号直接连接到嵌入式 SPI 闪存上，QSPI1 和 QSPI2 信号可供用户使用。

21.2. 串行接口特点

串行接口包括：

- 串行主操作。
- DMA 控制器接口-这使得 SSI 能够通过使用针对传输请求的握手接口，通过总线与 DMA 控制器进行接口。
- 时钟拉伸支持在增强的 SPI 传输。
- 数据项大小（4 到 32 位）—在程序员控制下的每个数据传输的项目大小。
- FIFO 深度-发射和接收 FIFO 缓冲区是 8 个字的深度。FIFO 的宽度固定在 32 位。
- 增强的 SPI 支持。
- 就地执行（XIP）模式支持。

21.3. 操作模式

SSI 在这三种模式下发挥作用：

1. 运行模式 - 运行模式是正常的操作模式。
2. 打盹模式 - 打盹模式是一种可配置的低功耗模式。
3. 停止模式 - SSI 在停止模式下处于非活动状态。

21.4. 方块图

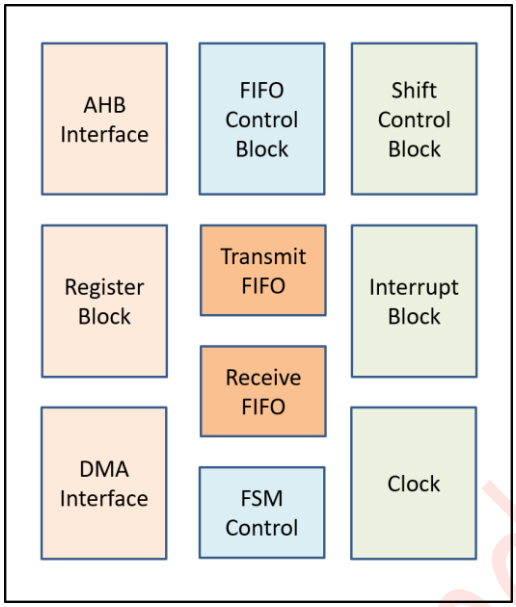


图 21-1：SSI 方块图

21.5. 应用图

以下是 LT7589 的 SSI 模块的应用程序。

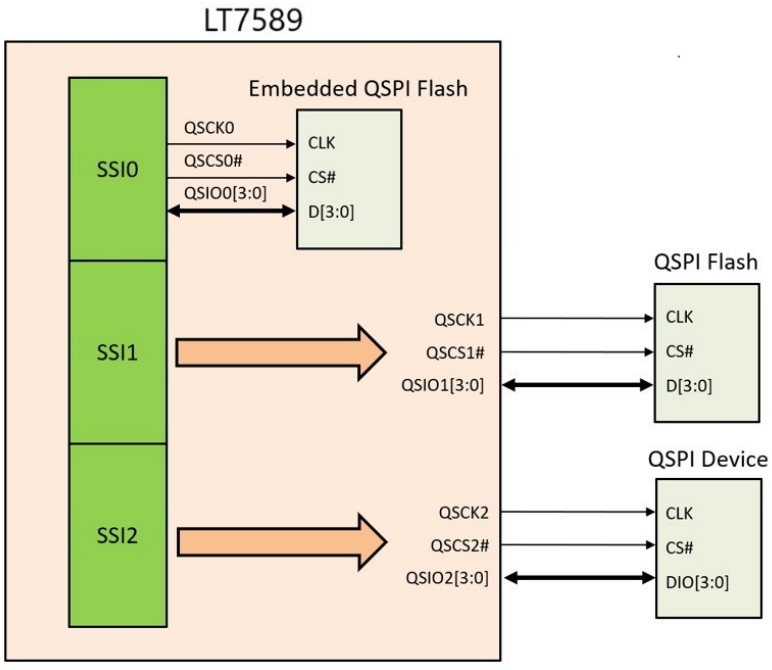


图 21-2：SSI 应用图

21.6. 内存映射和寄存器

21.6.1. 内存映射

SSI 模块内存映射如下表所示 QSPI0 的基本地址为 0x6000_0000，QSPI1 为 0x7000_0000，QSPI2 为 0x8000_0000。

表 21-1：SSI 内存映射

偏移地址	Bits[31:0]	访问权限
0x0000	Control Register 0 (CTRLR0)	S/U
0x0004	Control Register 1 (CTRLR1)	S/U
0x0008	SSI Enable Register (SSIENR)	S/U
0x000C	Microwire Control Register (MWCR)	S/U
0x0010	Slave Select Register (SER)	S/U
0x0014	Baud Rate Select Register (BAUDR)	S/U
0x0018	Transmit FIFO Threshold Level Register (TXFTLR)	S/U
0x001C	Receive FIFO Threshold Level Register (RXFTLR)	S/U
0x0020	Transmit FIFO Level Register (TXFLR)	S/U
0x0024	Receive FIFO Level Register (RXFLR)	S/U
0x0028	Status Register (SR)	S/U
0x002C	Interrupt Mask Register (IMR)	S/U
0x0030	Interrupt Status Register (ISR)	S/U
0x0034	Raw Interrupt Status Register (RISR)	S/U
0x0038	Transmit FIFO Overflow Interrupt Clear Register (TXOICR)	S/U
0x003C	Receive FIFO Overflow Interrupt Clear Register (RXOICR)	S/U
0x0040	Receive FIFO Underflow Interrupt Clear Register (RXUICR)	S/U
0x0048	Interrupt Clear Register (ICR)	S/U
0x004C	DMA Control Register (DMACR)	S/U
0x0050	DMA Transmit Data Level Register (DMATDLR)	S/U
0x0054	DMA Receive Data Level Register (DMARDLR)	S/U
0x0058	Identification Register (IDR)	S/U
0x005C	Version ID Register (VIDR)	S/U
0x0060 + i*0x4	SSI Data Register (DRx)	S/U
0x00F0	RX Sample Delay Register (RXSDR)	S/U
0x00F4	SPI Control Register 0 (SPICTRLR0)	S/U
0x00FC	XIP Mode Bits (XIPMBR)	S/U
0x0100	XIP Incr Inst Register (XIPIIR)	S/U
0x0104	XIP Wrap Inst Register (XIPWIR)	S/U
0x0108	XIP Control Register (XIPCR)	S/U
0x010C	XIP Slave Enable Register (XIPSER)	S/U
0x0110	XIP Receive FIFO Overflow Interrupt Clear Register (XRXIOCR)	S/U
0x0114	XIP Continus Transfer Time Out Register (XIPCTTOR)	S/U

21.6.2. 寄存器描述

21.6.2.1.控制寄存器 0（CTRLR0）

地址：QSPIn_BASEADDR + 0x0000_0000

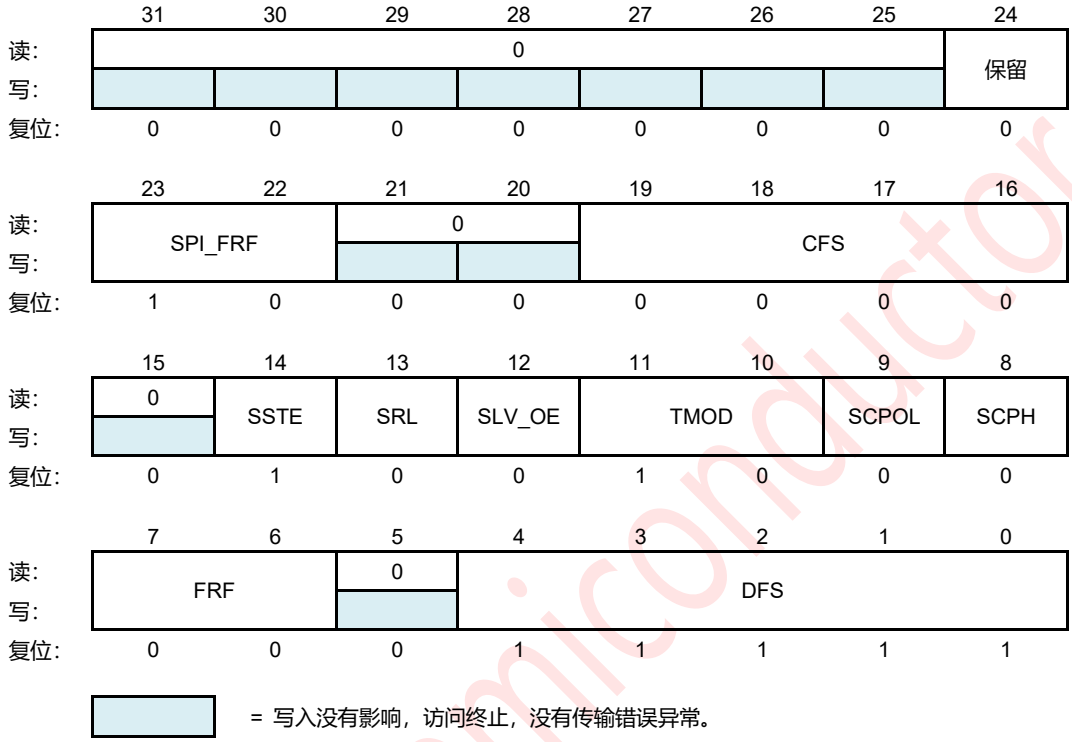


图 21-3：控制寄存器 0（CTRLR0）

此寄存器控制串行数据传输。当启用 SSI 时，不可能写入此寄存器。通过写入 SSIENR 来启用和禁用 SSI。

SPI_FRF — SPI Frame Format

仅当 SSIC_SPI_MODE 设置为“双”、“四”或“八度”模式时，选择“传输/接收数据位”的数据帧格式。

0x0（SPI_STANDARD）：标准 SPI 格式
0x1（SPI_DUAL）：双 SPI 格式
0x2（SPI_QUAD）：四轴 SPI 格式
0x3（SPI_OCTAL）：八进制 SPI 格式

CFS — Control Frame Size

选择微线帧格式的控制字的长度。

0x0（SIZE_01_BIT）：1 位控制 Word
0x1（SIZE_02_BIT）：2 位控制 Word
0x2（SIZE_03_BIT）：3 位控制 Word
0x3（SIZE_04_BIT）：4 位控制 Word
0x4（SIZE_05_BIT）：5 位控制 Word
0x5（SIZE_06_BIT）：6 位控制 Word
0x6（SIZE_07_BIT）：7 位控制 Word

0x7（SIZE_08_BIT）：8 位控制 Word
0x8（SIZE_09_BIT）：9 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 — Slave Select Toggle Enable.

在时钟阶段（SCPH）设置为 0 的 SPI 模式下操作时，此寄存器控制数据帧之间的从属选择线（ss_*_n）的行为。

1 =（TOGGLE_EN）：ss_*_n 行将在连续的数据帧之间切换，串行时钟（sclk）被保持在其默认值，而 ss_*_n 为高

0 =（TOGGLE_DISABLE）：ss_*_n 将保持低，sclk 将在传输期间持续运行

SRL — Shift Register Loop.

仅用于测试目的。当内部活动时，将发射移位寄存器输出连接到接收移位寄存器输入。可在串行从模式和串行主模式中使用。当 SSI 以环回模式配置为从端时，ss_in_n 和 ssi_clk 信号必须由外部源提供。在这种模式下，从服务器不能生成这些信号，因为没有什么可以返回的内容。

1 =（TESTING_MODE）：测试模式操作

0 =（NORMAL_MODE）：正常模式操作

SLV_OE — Slave Output Enable.

仅当 SSI 被配置为串行从属设备时才相关。当配置为串行主节点时，此位字段没有任何功能。

1 = 从输出

0 = 从输出

TMOD — Transfer Mode.

选择串行通信的传输方式。此字段不影响转移的两面性性。仅指示接收数据或传输数据是否有效。

0x0（TX_AND_RX）：传输和接收；不适用于增强型 SPI 工作模式
0x1（TX_ONLY）：仅传输模式；或以增强的 SPI 操作模式写入
0x2（RX_ONLY）：仅接收模式；或在增强的 SPI 操作模式下读取
0x3（EEPROM_READ）：EEPROM 读取模式；不适用于增强型 SPI 操作模式

SCPOL — Serial Clock Polarity.

当帧格式（FRF）设置为摩托罗拉 SPI 时有效。用于选择非活动串行时钟的极性，当 SSI 主服务器不在串行总线上主动传输数据时，该时钟保持为非活动状态。

1 =（INACTIVE_HIGH）：串行时钟的非活动状态较高

0 =（INACTIVE_LOW）：串行时钟的非活动状态较低

SCPH — Serial Clock Phase.

当帧格式（FRF）设置为摩托罗拉 SPI 时有效。串行时钟相位选择串行时钟与从机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缘捕获数据。当 SCPH = 1 时，在从属选择线被激活后，串行时钟开始切换一个周期，并在串行时钟的第二个边缘捕获数据。

1 =（START_BIT）：在第一位开始时进行串行时钟切换

0 =（MIDDLE_BIT）：串行时钟在第一位的中间进行切换

FRF — Frame Format.

选择由哪个串行协议传输数据。

0x0（SPI）：摩托罗拉 SPI 帧格式

0x1（SSP）：德州仪器公司的 SSP 帧格式

0x2（微线）：国家半导体微线帧格式

0x3（保留）：保留

DFS — Data Frame Size.

选择数据帧长度。当数据帧大小被编程为小于 32 位时，接收数据被接收逻辑自动对正，接收 FIFO 的上限被零填充。

在写入传输 FIFO 之前，您必须正确证明传输数据。传输逻辑在传输数据时忽略上部未使用的位。

0x0（DFS_01_BIT）：保留

0x1（DFS_02_BIT）：保留

0x2（DFS_03_BIT）：保留

0x3（DFS_04_BIT）：4 位串行数据传输

0x4（DFS_05_BIT）：5 位串行数据传输

0x5（DFS_06_BIT）：6 位串行数据传输

0x6（DFS_07_BIT）：7 位串行数据传输

0x7（DFS_08_BIT）：8 位串行数据传输

0x8（DFS_09_BIT）：9 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

21.6.2.2.控制寄存器 1（CTRLR1）

只有当 SSI 被配置为主设备时，此寄存器才存在。当 SSI 配置为串行从服务器时，写入此位置无效；从此位置读取返回 0。控制寄存器 1 在 RX-ONLY 模式和 TX-模式 ONLY 模式下控制串行传输的结束。当启用 SSI 时，不可能写入此寄存器。通过写入 SSIENR 寄存器来启用和禁用 SSI。

地址：QSPIn_BASEADDR + 0x0000_0004

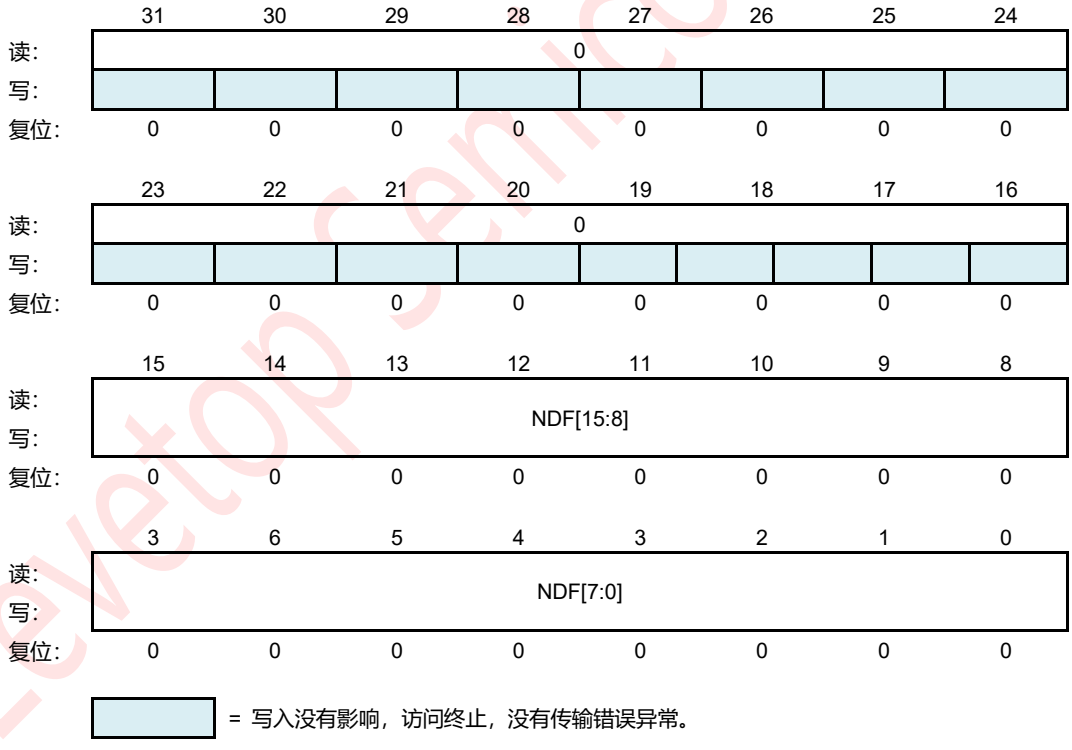


图 21-4：控制寄存器 1（CTRLR1）

NDF[15:0] — Number of Data Frames.

当 TMOD = 01 或 TMOD = 10 或 TMOD = 11 时，该寄存器字段设置了 SSI 要连续接收或传输的数据帧数。SSI 继续接收或传输串行数据，直到数据帧数等于该寄存器值加上 1。当 SSI 被配置为串行从属服务器时，这个寄存器没有任何作用，也不存在。当 TMOD = 01 时，必须设置 SPI_CTRLR0 中的 CLK_STRETCH_EN。

21.6.2.3. SSI 启用寄存器（SSIENR）

地址：QSPIn_BASEADDR + 0x0000_0008

图 21-5：SSI 启用寄存器（SSIENR）

SSIC_EN — SSI Enable.

启用和禁用所有 SSI 操作。禁用后，所有串行传输都将立即停止。当设备被禁用时，将清除传输和接收 FIFO 缓冲区。当启用时，不可能对某些 SSI 控制寄存器进行编程。当禁用时，设置 SSI 睡眠输出（延迟后），通知系统删除 ssi_clk 是安全的，从而节省系统的功耗。

1 = SSI 已启用。

0 = SSI 禁用。

21.6.2.4. 微线控制寄存器（MWCR）

地址：QSPIn_BASEADDR + 0x0000_000C

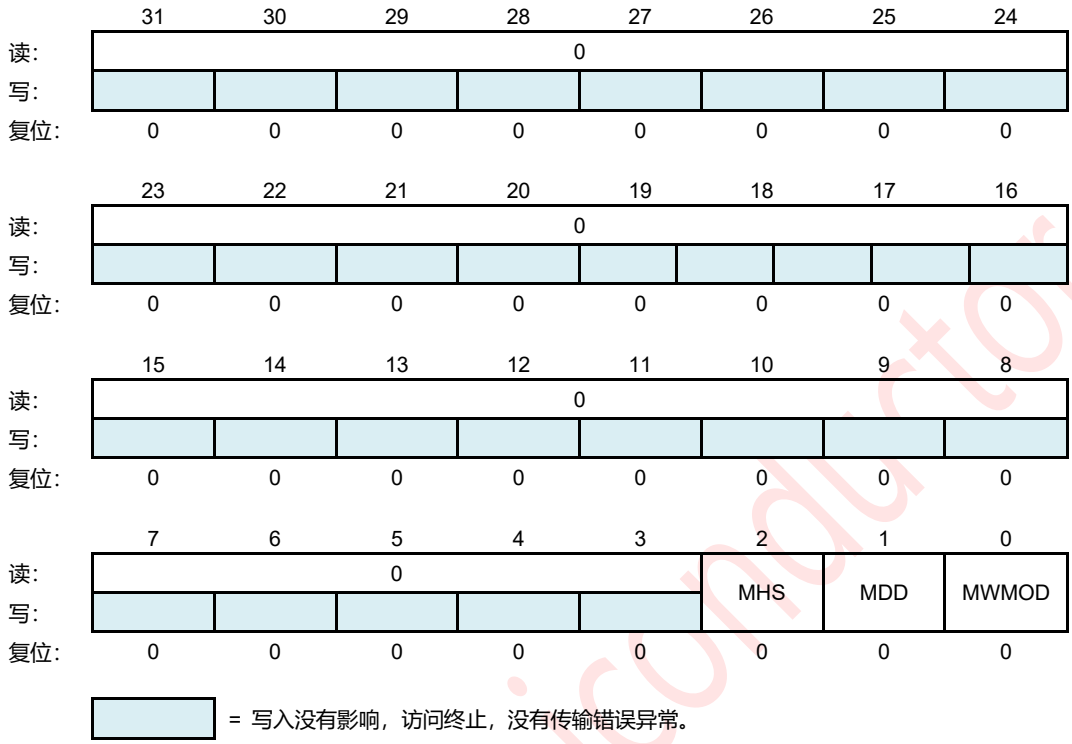


图 21-6：微线控制寄存器（MWCR）

该寄存器控制半双工微线串行协议的数据字的方向。当启用 SSI 时，不可能写入此寄存器。通过写入 SSIENR 寄存器来启用和禁用 SSI。

MHS — Microwire Handshaking.

仅当 SSI 被配置为串行主设备时才相关。当配置为串行从属服务器时，此位字段没有任何功能。用于启用和禁用微线协议的繁忙/准备就绪握手接口。当启用时，SSI 在传输最后一个数据/控制位之后，在清除 SR 寄存器中的 BUSY 状态之前，从目标从服务器检查就绪状态。

1 = 已启用了握手界面

0 = 握手界面禁用

MDD — Microwire Control.

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 MacroCell 从外部串行设备接收数据字。当这个位被设置为 1 时，数据字从 SSI 宏单元传输到外部串行设备。

1 = SSI 传输数据

0 = SSI 接收数据

MWMOD — Microwire Transfer Mode.

定义微线传输是顺序的还是非顺序的。当使用顺序模式时，只需要一个控制字来发送或接收一个数据字块。当使用非顺序模式时，对于被传输或接收的每个数据字，必须有一个控制字。

1 = 顺序转移

0 = 非顺序传输

21.6.2.5. 从机启用寄存器（SER）

地址：QSPIn_BASEADDR + 0x0000_0010

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0							SER
写:								
复位:	0	0	0	0	0	0	0	1

= 写入没有影响，访问终止，没有传输错误异常。

图 21-7：从属服务器启用寄存器（SER）

此寄存器仅在将 SSI 配置为主设备时有效。当 SSI 配置为串行从服务器时，写入此位置无效；从此位置读取返回 0。该寄存器允许单个从服务器从 SSI 主服务器中选择输出行。在 SSI 主服务器上最多有 16 个从属选择输出引脚。当 SSI 忙于和 SSIC_EN = 1 时，您无法写入此寄存器。

SER — Slave Select Enable Flag.

1 = 选择

0 = 未选择

21.6.2.6. 波特率选择（BAUDR）

地址：QSPIn_BASEADDR + 0x0000_0014

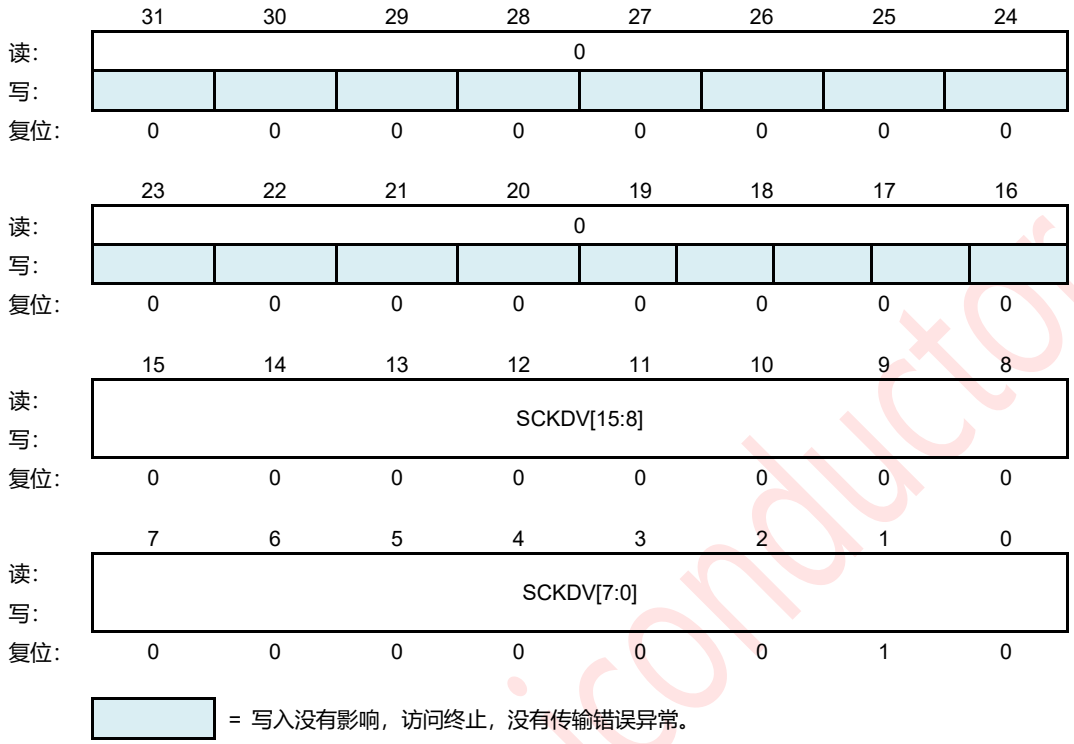


图 21-8：波特率选择（BAUDR）

SCKDV[15:0] — SSI Clock Divider.

此字段的 LSB 总是设置为 0，并且不受写操作的影响，该写操作确保在此寄存器中保持偶数值。如果该值为 0，则会禁用串行输出时钟（sclk_out）。sclk_out 的频率由以下公式得到：

Fsclk_out = Fssi_clk / SCKDV

其中，SCKDV 是介于 2 到 65534 之间的任何偶数值。例如：适用于 Fssi_clk = 3.6864MHz 和 SCKDV = 2 Fsclk_out = 3.6864/2 = 1.8432MHz。

21.6.2.7. 传输 FIFO 阈值等级（TXFTLR）

地址：QSPIn_BASEADDR + 0x0000_0018

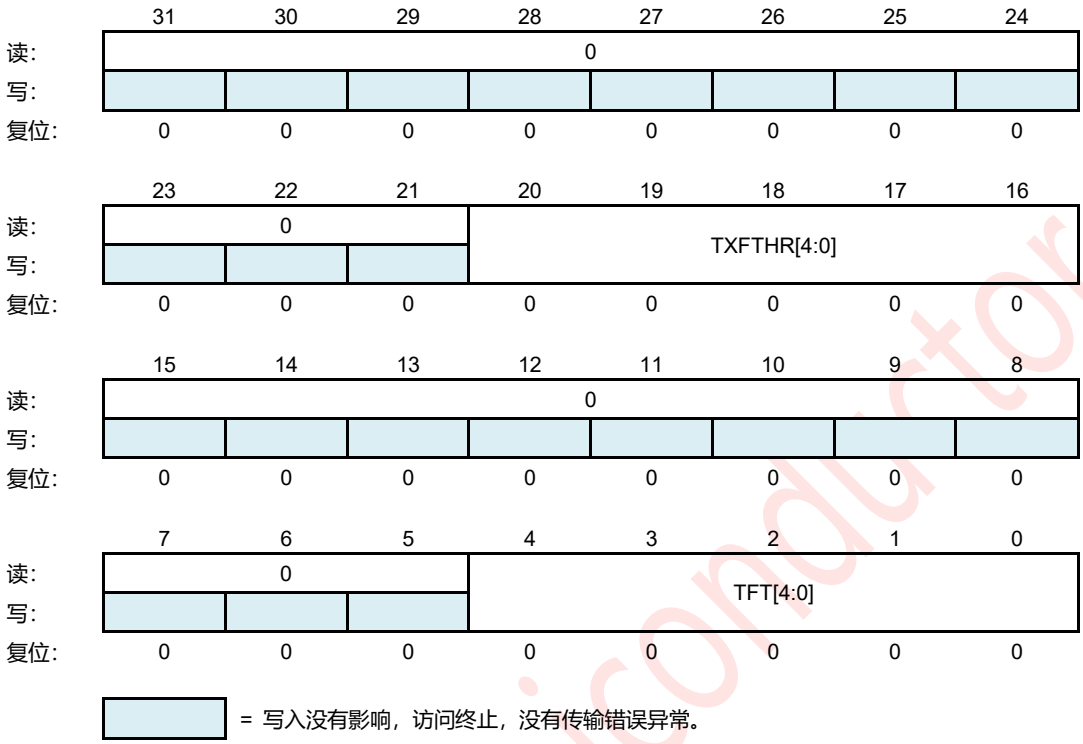


图 21-9：传输 FIFO 阈值级别（TXFTLR）

该寄存器控制传输 FIFO 存储器的阈值。通过写入 SSIENR 寄存器来启用和禁用 SSI。

TXFTHR[4:0] — Transfer start FIFO level.

用于控制传输 FIFO 中的入口级别，超过该级别的传输将从串行线开始。此寄存器可用于确保在开始在串行线上的写入操作之前，在传输 FIFO 中存在足够的数据。这些字段仅对主操作模式有效。

TFT[4:0] — Transmit FIFO Threshold.

控制传输 FIFO 控制器触发中断的条目的级别（或以下）。FIFO 深度可在 8-256 范围内配置；该寄存器的大小符合访问 FIFO 所需的地址位数。如果您试图将此值设置为大于或等于 FIFO 的深度，则不会写入此字段，并保留其当前值。当发送 FIFO 条目的数量小于或等于该值时，将触发发送 FIFO 空中断。

21.6.2.8. 接收 FIFO 阈值级别（RXFTLR）

地址：QSPIn_BASEADDR + 0x0000_001C

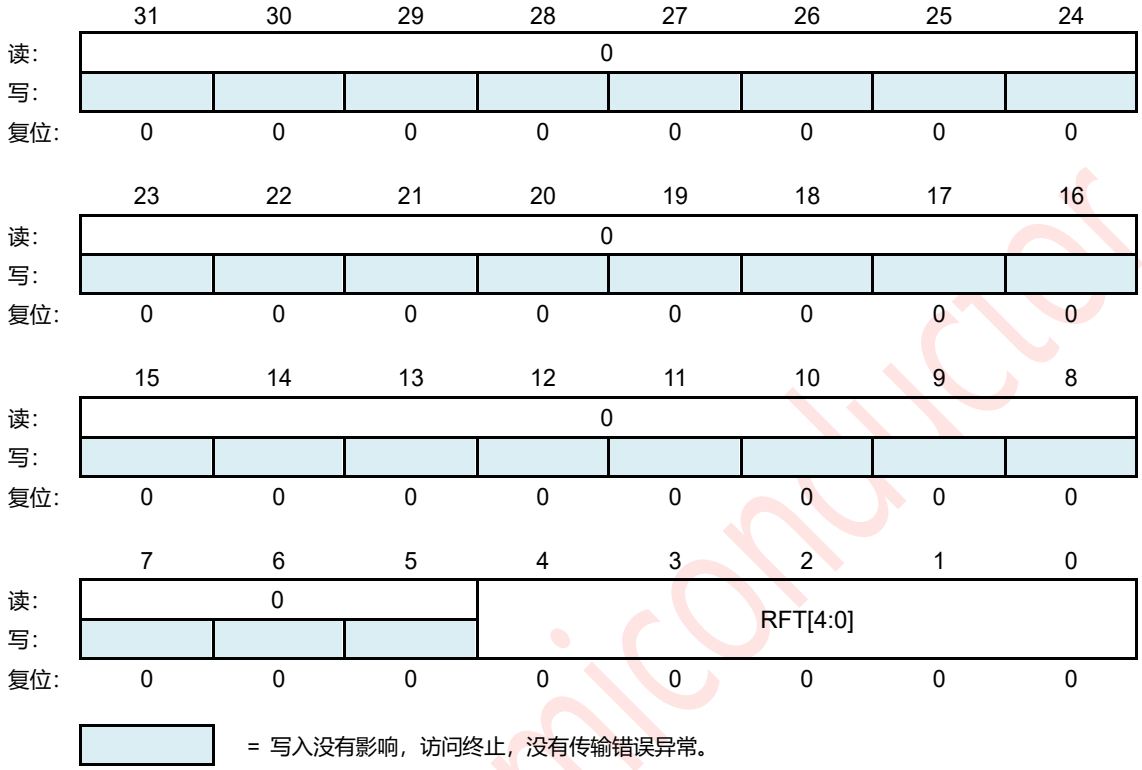


图 21-10：接收 FIFO 阈值级别（RXFTLR）

该寄存器控制接收 FIFO 存储器的阈值。通过写入 SSIENR 寄存器来启用和禁用 SSI。

RFT[4:0] — Receive FIFO Threshold.

控制接收 FIFO 控制器触发中断的条目的级别（或以上）。这个寄存器的大小被调整为访问 FIFO 所需的地址位数。如果您试图将此值设置为大于 FIFO 的深度，则不会写入此字段，并保留其当前值。当接收的 FIFO 条目的数量大于或等于该值 + 1 时，将触发接收的 FIFO 完全中断。

21.6.2.9.发射 FIFO 液位寄存器（TXFLR）

地址：QSPIn_BASEADDR + 0x0000_0020

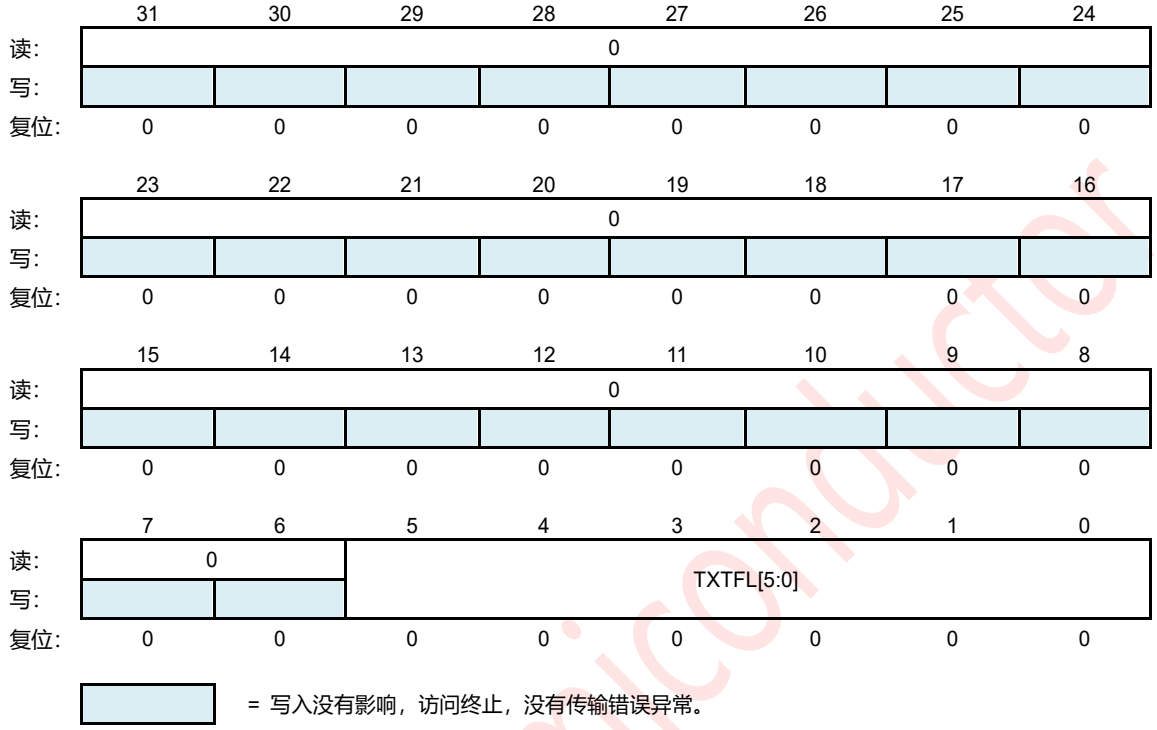


图 21-11：传输 FIFO 液位寄存器（TXFLR）

TXTFL[5:0] — Transmit FIFO Level.

包含传输 FIFO 中的有效数据条目的数量。

21.6.2.10. 接收 FIFO 级别寄存器（RXFLR）

地址：QSPIn_BASEADDR + 0x0000_0024

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0		RXTFL[5:0]					
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 21-12：接收 FIFO 级别寄存器（RXFLR）

RXTFL[5:0] — Receive FIFO Level.

包含接收 FIFO 中的有效数据条目的数量。

21.6.2.11. 状态寄存器（SR）

地址：QSPIn_BASEADDR + 0x0000_0028

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0	保留	TXE	RFF	RFNE	TFE	TFNF	BUSY
写:								
复位:	0	0	0	0	0	1	1	0

= 写入没有影响，访问终止，没有传输错误异常。

图 21-13：状态寄存器（SR）

TXE — Transmission Error.

设置在启动传输时，传输 FIFO 是否为空。只有当 SSI 被配置为从设备时，才能设置此位。来自以前传输的数据将在 TXD 线上重新发送。读取时，将清除此位。

1 =（TX_ERROR）：传输错误

0 =（NO_ERROR）：无错误

RFF — Receive FIFO Full.

当接收 FIFO 完全满时，将设置此位。当接收 FIFO 包含一个或多个空位置时，该位将被清除。

1 =（FULL）：接收 FIFO 已满

0 =（NOT_FULL）：接收 FIFO 未满

RFNE — Receive FIFO Not Empty.

当接收 FIFO 包含一个或多个条目时设置，当接收 FIFO 为空时清除。这个位可以通过软件轮询以完全清空接收 FIFO。

1 =（NOT_EMPTY）：接收 FIFO 不是空的

0 =（EMPTY）：接收 FIFO 为空

TFE — Transmit FIFO Empty.

当传输 FIFO 完全为空时，将设置此位。当传输 FIFO 包含一个或多个有效条目时，该位将被清除。此位字段不请求中断。

1 = （EMPTY）：传输 FIFO 为空

0 =（NOT_EMPTY）：传输 FIFO 不是空的

TFNF — Transmit FIFO Not Full.

当传输 FIFO 包含一个或多个空位置时设置，并在 FIFO 满时清除。

1 =（NOT_FULL）：Tx FIFO 未满

0 = Tx FIFO 已满

BUSY — SSI Busy Flag.

设置时，表示正在进行串行传输；清除时表示 SSI 空闲或禁用。

1 = （ACTIVE）：SSI 正在主动传输数据

0 = （INACTIVE）：SSI 已空闲或已禁用

21.6.2.12. 中断掩码寄存器（IMR）

地址：QSPIn_BASEADDR + 0x0000_002C

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0	XRXOIM	Reserved	RXFIM	RXOIM	RXUIM	TXOIM	TXEIM
写:								
复位:	0	1	1	1	1	1	1	0

= 写入没有影响，访问终止，没有传输错误异常。

图 21-14：中断掩码寄存器（IMR）

XRXOIM — XIP Receive FIFO Overflow Interrupt Mask

1 = （UNMASKED）：ssi_xrxo_intr 中断未被屏蔽

0 = （MASKED）：ssi_xrxo_intr 中断被屏蔽

RXFIM — Receive FIFO Full Interrupt Mask

1 = （UNMASKED）：ssi_rxf_intr 中断未被屏蔽

0 = （MASKED）：ssi_rxf_intr 中断被屏蔽

RXOIM — Receive FIFO Overflow Interrupt Mask

1 = （UNMASKED）：ssi_rxo_intr 中断未被屏蔽

0 = （MASKED）：ssi_rxo_intr 中断被屏蔽

RXUIM — Receive FIFO Underflow Interrupt Mask

1 = （UNMASKED）：ssi_rxu_intr 中断未被屏蔽

0 = （MASKED）：ssi_rxu_intr 中断被屏蔽

TXOIM — Transmit FIFO Overflow Interrupt Mask

1 = （UNMASKED）：ssi_txo_intr 中断未被屏蔽

0 = （MASKED）：ssi_txo_intr 中断被屏蔽

TXEIM — Transmit FIFO Empty Interrupt Mask

1 = （UNMASKED）：ssi_txe_intr 中断未被屏蔽

0 = （MASKED）：ssi_txe_intr 中断被屏蔽

21.6.2.13. 中断状态寄存器（ISR）

地址：QSPIn_BASEADDR + 0x0000_0030

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0	XRXOIS	保留	RXFIS	RXOIS	RXUIS	TXOIS	TXEIS
写:								
复位:	0	0	0	0	0	0	0	1

= 写入没有影响，访问终止，没有传输错误异常。

图 21-15：中断状态寄存器（ISR）

此寄存器会在屏蔽 SSI 中断后报告它们的状态。

XRXOIS — XIP Receive FIFO Overflow Interrupt Status

1 = （ACTIVE）：掩蔽后的 ssi_xrxo_intr 中断被激活

0 = （INACTIVE）：屏蔽后 ssi_xrxo_intr 中断未激活

RXFIS — Receive FIFO Full Interrupt Status

1 = （ACTIVE）：掩蔽后的 ssi_rxf_intr 中断被激活

0 = （INACTIVE）：屏蔽后 ssi_rxf_intr 中断未激活

RXOIS — Receive FIFO Overflow Interrupt Status

1 = （ACTIVE）：掩蔽后的 ssi_rxo_intr 中断被激活

0 = （INACTIVE）：屏蔽后 ssi_rxo_intr 中断未激活

RXUIS — Receive FIFO Underflow Interrupt Status

1 = （ACTIVE）：掩蔽后的 ssi_rxu_intr 中断被激活

0 = （INACTIVE）：屏蔽后 ssi_rxu_intr 中断未激活

TXOIS — Transmit FIFO Overflow Interrupt Status

1 = （ACTIVE）：掩蔽后的 ssi_txo_intr 中断被激活

0 = （INACTIVE）：屏蔽后 ssi_txo_intr 中断未激活

TXEIS — Transmit FIFO Empty Interrupt Status

1 = （ACTIVE）：掩蔽后的 ssi_txe_intr 中断被激活

0 = （INACTIVE）：屏蔽后 ssi_txe_intr 中断未激活

21.6.2.14. 原始中断状态寄存器（RISR）

地址：QSPIn_BASEADDR + 0x0000_0034

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0	XRXOIS	保留	RXFIS	RXOIS	RXUIS	TXOIS	TXEIS
写:								
复位:	0	0	0	0	0	0	0	1

= 写入没有影响，访问终止，没有传输错误异常。

图 21-16：原始中断状态寄存器（RISR）

XRXOIS — XIP Receive FIFO Overflow Raw Interrupt Status

1 = （ACTIVE）：ssi_xrxo_intr 中断在屏蔽之前是激活的

0 = （INACTIVE）：ssi_xrxo_intr 中断不是活动的之前的屏蔽

RXFIS — Receive FIFO Full Raw Interrupt Status

1 = （ACTIVE）：ssi_rxf_intr 中断在屏蔽之前是激活的

0 = （INACTIVE）：ssi_rxf_intr 中断不是活动的之前的屏蔽

RXOIS — Receive FIFO Overflow Raw Interrupt Status

1 = （ACTIVE）：ssi_rxo_intr 中断在屏蔽之前是激活的

0 = （INACTIVE）：ssi_rxo_intr 中断不是活动的之前的屏蔽

RXUIS — Receive FIFO Underflow Raw Interrupt Status

1 = （ACTIVE）：ssi_rxu_intr 中断在屏蔽之前是激活的

0 = （INACTIVE）：ssi_rxu_intr 中断不是活动的之前的屏蔽

TXOIS — Transmit FIFO Overflow Raw Interrupt Status

1 = （ACTIVE）：ssi_txo_intr 中断在屏蔽之前是激活的

0 = （INACTIVE）：ssi_txo_intr 中断不是活动的之前的屏蔽

TXEIS — Transmit FIFO Empty Raw Interrupt Status

1 = （ACTIVE）：ssi_txe_intr 中断在屏蔽之前是激活的

0 = （INACTIVE）：ssi_txe_intr 中断不是活动的之前的屏蔽

21.6.2.15. 传输 FIFO 溢出中断清除寄存器（TXOICR）

地址：QSPIn_BASEADDR + 0x0000_0038

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0							TXOICR
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 21-17：传输 FIFO 溢出中断清除寄存器（TXOICR）

TXOICR — Clear Transmit FIFO Overflow Interrupt.

这个寄存器反映了中断的状态。从这个寄存器读取会清除 ssi_txo_intr 中断；写入没有效果。

21.6.2.16. 接收 FIFO 溢出中断清除寄存器（RXOICR）

地址：QSPIn_BASEADDR + 0x0000_003C

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0							RXOICR
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 21-18：接收 FIFO 溢出中断清除寄存器（RXOICR）

RXOICR — Clear Receive FIFO Overflow Interrupt.

这个寄存器反映了中断的状态。从这个寄存器读取会清除 ssi_rxo_intr 中断；写入没有效果。

21.6.2.17. 接收 FIFO 下溢中断清除寄存器（RXUICR）

地址：QSPIn_BASEADDR + 0x0000_0040

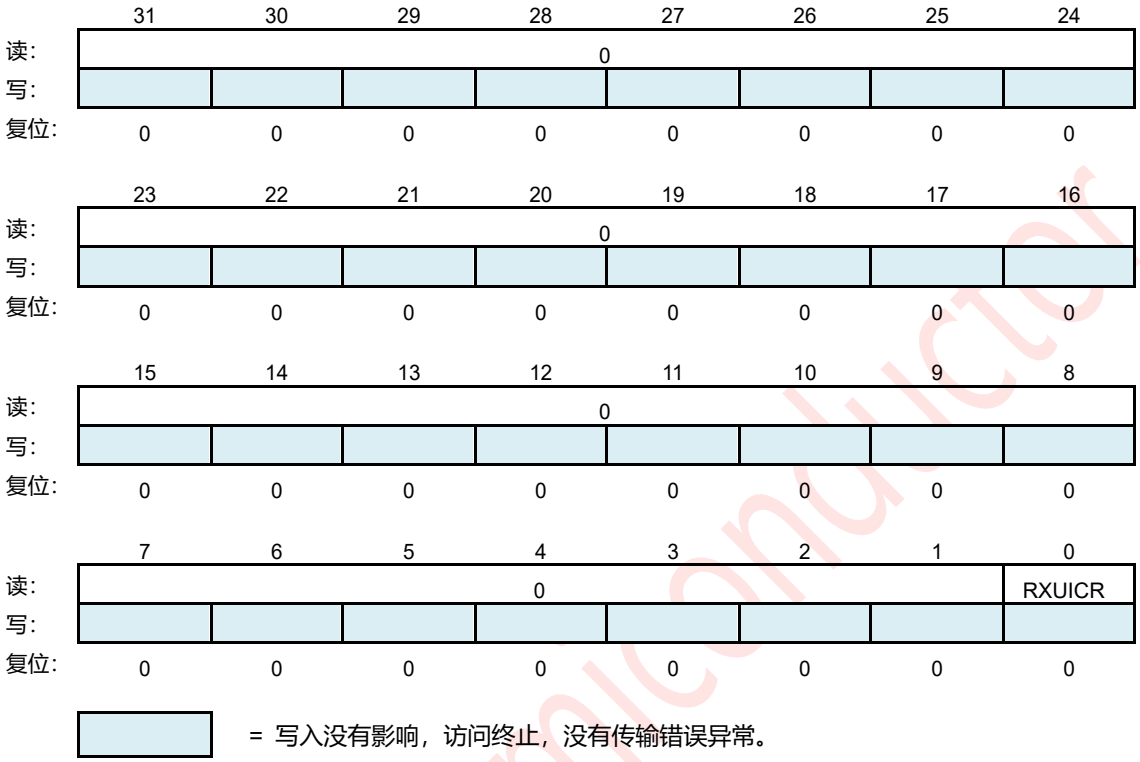


图 21-19：接收 FIFO 下溢（UnderFlow)中断清除寄存器（RXUICR）

RXUICR — Clear Receive FIFO Underflow Interrupt.

这个寄存器反映了中断的状态。从这个寄存器读取会清除 ssi_rxu_intr 中断；写入没有效果。

21.6.2.18. 中断清除寄存器（ICR）

地址：QSPIn_BASEADDR + 0x0000_0048

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0							ICR
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 21-20：中断清除寄存器（ICR）

ICR — Clear Interrupts.

如果下面的任何一个中断都是活动的，则会设置此寄存器。读取将清除 ssi_txo_intr、ssi_rxu_intr、ssi_rxo_intr 和 ssi_mst_intr 中断。写入此寄存器没有效果。

21.6.2.19. DMA 控制寄存器（DMACR）

地址：QSPIn_BASEADDR + 0x0000_004C

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0						TDMAE	RDMAE
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 21-21：DMA 控制寄存器（DMACR）

此寄存器仅在 SSI 配置了一组 DMA 控制器接口信号（SSIC_HAS_DMA = 1）时有效。当 SSI 未配置为 DMA 操作时，此寄存器将不存在，并且写入该寄存器的地址将无效；从此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 — Transmit DMA Enable

此位可启用/禁用传输 FIFO DMA 信道。

1 = 传输 DMA

0 = 传输 DMA 被禁用

RDMAE — Receive DMA Enable

此位可启用/禁用接收 FIFO DMA 通道。

1 = 接收 DMA

0 = 接收 DMA 被禁用

21.6.2.20. DMA 传输数据级别（DMATDLR）

地址：QSPIn_BASEADDR + 0x0000_0050

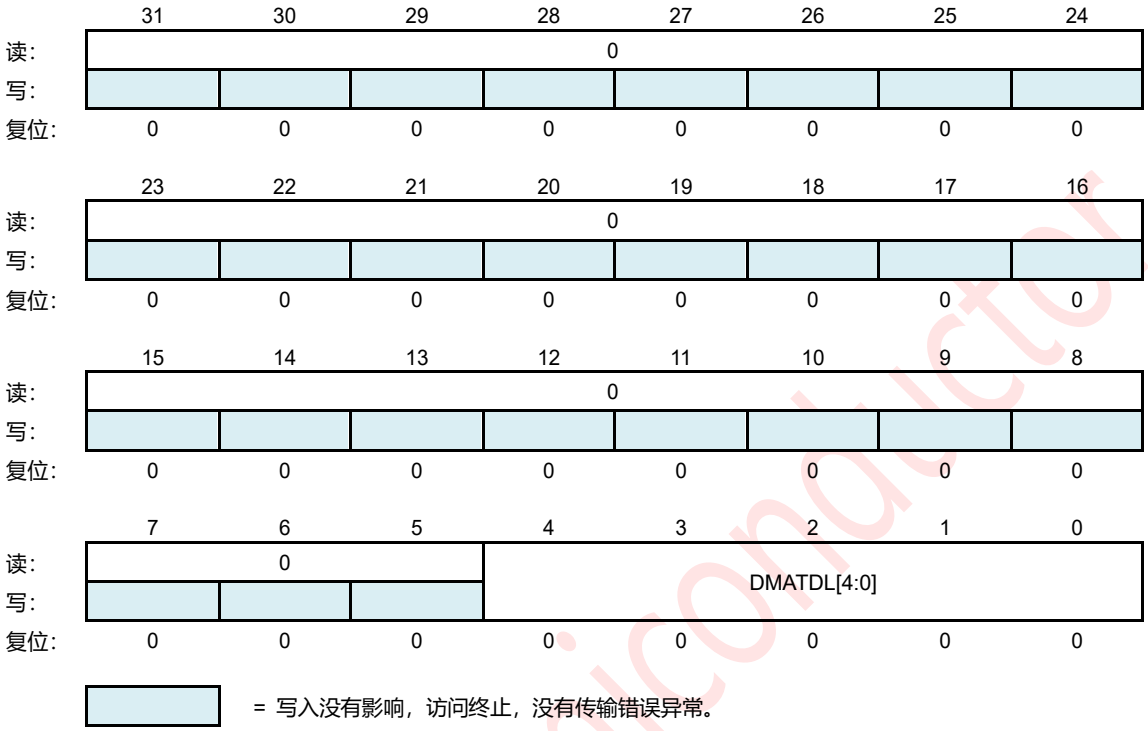


图 21-22：DMA 传输数据级别（DMATDLR）

此寄存器仅在 SSI 配置了一组 DMA 接口信号（SSIC_HAS_DMA = 1）时有效。当未为 DMA 操作配置 SSI 时，此寄存器将不存在，并且写入其地址将无效；从其地址读取将返回零。

DMATDL[4:0] — Transmit Data Level.

该位字段控制由发送逻辑发出 DMA 请求的级别。等于水印电平，即当发送 FIFO 中的有效数据项数等于或低于该字段值时，以及 TDMAE = 1，即产生 dma_tx_req 信号。

21.6.2.21. DMA 接收数据级别（DMARDLR）

地址：QSPIn_BASEADDR + 0x0000_0054

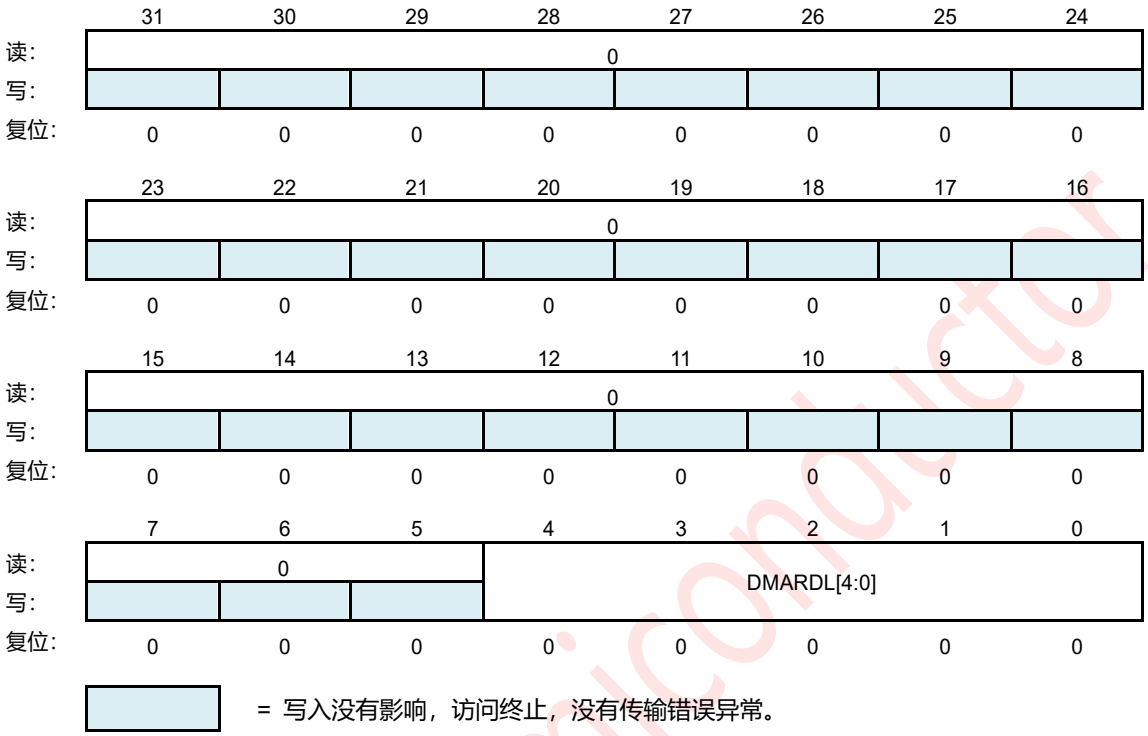


图 21-23：DMA 接收数据级别（DMARDLR）

此寄存器仅在 SSI 配置了一组 DMA 接口信号（SSIC_HAS_DMA = 1）时有效。当未为 DMA 操作配置 SSI 时，此寄存器将不存在，并且写入其地址将无效；从其地址读取将返回零。

DMARDL[4:0] — Receive Data Level.

该位字段控制由接收逻辑发出 DMA 请求的级别。水印级别 = DMARDL + 1，即，当接收 FIFO 中的有效数据条目数等于或高于该字段值 + 1 和 RDMAE = 1 时，生成 dma_rx_req。

21.6.2.22. 识别寄存器（IDR）

地址：QSPIn_BASEADDR + 0x0000_0058

	31	30	29	28	27	26	25	24
读:	IDCODE[31:24]							
写:								
复位:	1	1	1	1	1	1	1	1

	23	22	21	20	19	18	17	16
读:	IDCODE[23:16]							
写:								
复位:	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8
读:	IDCODE[15:8]							
写:								
复位:	1	1	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
读:	IDCODE[7:0]							
写:								
复位:	1	1	1	1	1	1	1	1

= 写入没有影响，访问终止，没有传输错误异常。

图 21-24：识别寄存器（IDR）

IDCODE[31:0] — Identification code.

寄存器包含外围设备的标识代码，在配置时写入寄存器。

21.6.2.23. 版本 ID 寄存器（VIDR）

地址：QSPIn_BASEADDR + 0x0000_005C

	31	30	29	28	27	26	25	24
读:	SSIC_COMP_VERSION[31:24]							
写:								
复位:	0	0	1	1	0	0	0	1

	19	22	21	20	19	18	17	16
读:	SSIC_COMP_VERSION[23:16]							
写:								
复位:	0	0	1	1	0	0	0	0

	11	14	13	12	11	10	9	8
读:	SSIC_COMP_VERSION[15:8]							
写:								
复位:	0	0	1	1	0	0	0	1

	3	6	5	4	3	2	1	0
读:	SSIC_COMP_VERSION[7:0]							
写:								
复位:	0	0	1	0	1	0	1	0

= 写入没有影响，访问终止，没有传输错误异常。

图 21-25：版本 ID 寄存器（VIDR）

SSIC_COMP_VERSION[31:0]

包含同步系统组件版本的十六进制表示形式。由版本中每个数字的 ASCII 值组成，后面跟着*。例如，32_30_31_2A 表示版本 2.01*。

21.6.2.24. SSI 数据寄存器（DRx）

地址：QSPIn_BASEADDR + 0x0000_0060

	31	30	29	28	27	26	25	24
读: 写:	DR[31:24]							
复位:	0	0	0	0	0	0	0	0

	19	22	21	20	19	18	17	16
读: 写:	DR[23:16]							
复位:	0	0	0	0	0	0	0	0

	11	14	13	12	11	10	9	8
读: 写:	DR[15:8]							
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读: 写:	DR[7:0]							
复位:	0	0	0	0	0	0	0	0

图 21-26：SSI 数据寄存器（DRx）

SSI 数据寄存器是发送/接收 FIFOs 的 32 位读/写缓冲区。当读取寄存器时，将访问接收 FIFO 缓冲区中的数据。当它被写入时，数据被移动到传输 FIFO 缓冲区；只有当 SSIC_EN = 1 时才能发生写入。当 SSIC_EN = 为 0 时，fifo 会被复位。

DR[31:0] — Data Register.

在写入此寄存器时，必须正确数据。读取数据将自动被正确证明。

读取 = 接收 FIFO 缓冲区

写入 = 传输 FIFO 缓冲区。

21.6.2.25. RX 取样延迟寄存器（RXSDR）

地址：QSPIn_BASEADDR + 0x0000_00F0

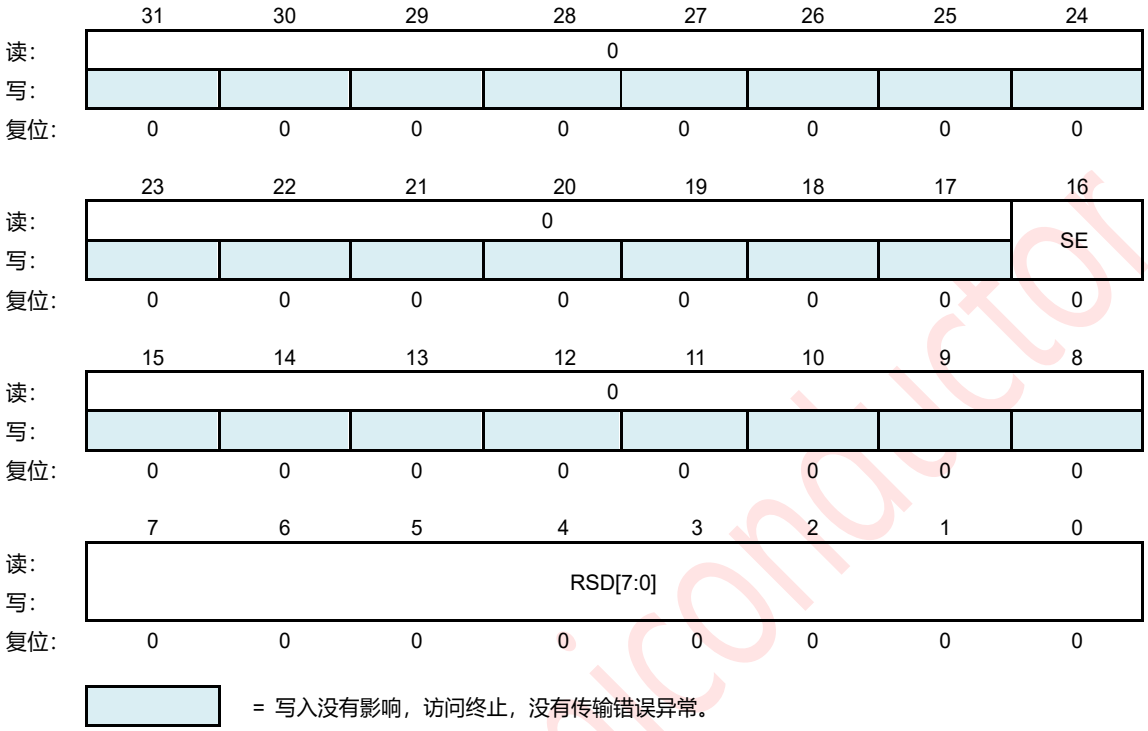


图 21-27：RX 取样延迟寄存器（RXSDR）

SE — Receive Data (RXD) Sampling Edge.

1 = 使用 ssi_clk 的负缘对输入数据进行采样

0 = 使用 ssi_clk 的正缘对输入的数据进行采样

RSD[7:0] — Receive Data (RXD) Sample Delay.

此寄存器用于延迟 RXD 输入端口的样本。每个值代表 RXD 样本上的一个 ssi_clk 延迟。

21.6.2.26. SPI 控制寄存器 0（SPICTRLR0）

地址：QSPIn_BASEADDR + 0x0000_00F4

	31	30	29	28	27	26	25	24
读:	0	CLK_STRETCH_EN	保留	0	保留			
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0	0	保留					
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	WAIT_CYCLES[4:0]					0	INST_L[1:0]	
写:								
复位:	0	0	0	0	0	0	1	0

	7	6	5	4	3	2	1	0
读:	保留	0	ADDR_L[3:0]				TRANS_TYPE[1:0]	
写:								
复位:	0	0	0	1	1	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 21-28：SPI 控制寄存器 0（SPICTRLR0）

CLK_STRETCH_EN

在 SPI 传输中启用时钟拉伸能力。

在写的情况下，如果 FIFO 变成空的，SSI 将拉伸时钟，直到 FIFO 有足够的数据来继续传输。在读取的情况下，如果接收的 FIFO 变成满的，SSI 将停止时钟，直到从 FIFO 读取数据。

提示：建议始终设置此位

1 = clk_stretch_enable

0 = clk_stretch_disable

WAIT_CYCLES[4:0] — Wait cycles in Dual/Quad/Octal mode between control frames transmit and data reception. Specified as number of SPI clock cycles.

INST_L[1:0] — Dual/Quad/Octal mode instruction length in bits.

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[3:0] — Length of Address to be transmitted

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[1:0] — Address and instruction transfer format.

选择 SSI 是否将在标准 SPI 模式或在 CTRLR0 中选择的 SPI 模式下传输指令/地址。SPI_FRF 字段。

0x0（TT0）：指令和地址将以标准 SPI 模式发送。
0x1（TT1）：指令将以标准 SPI 模式发送，地址将以 CTRLR0 指定的模式发送。SPI_FRF.
0x2（TT2）：指令和地址都将以 SPI_FRF 指定的方式发送。
0x3（TT3）：保留。

21.6.2.27. XIP 模式位（XIPMBR）

地址：QSPIn_BASEADDR + 0x0000_00FC

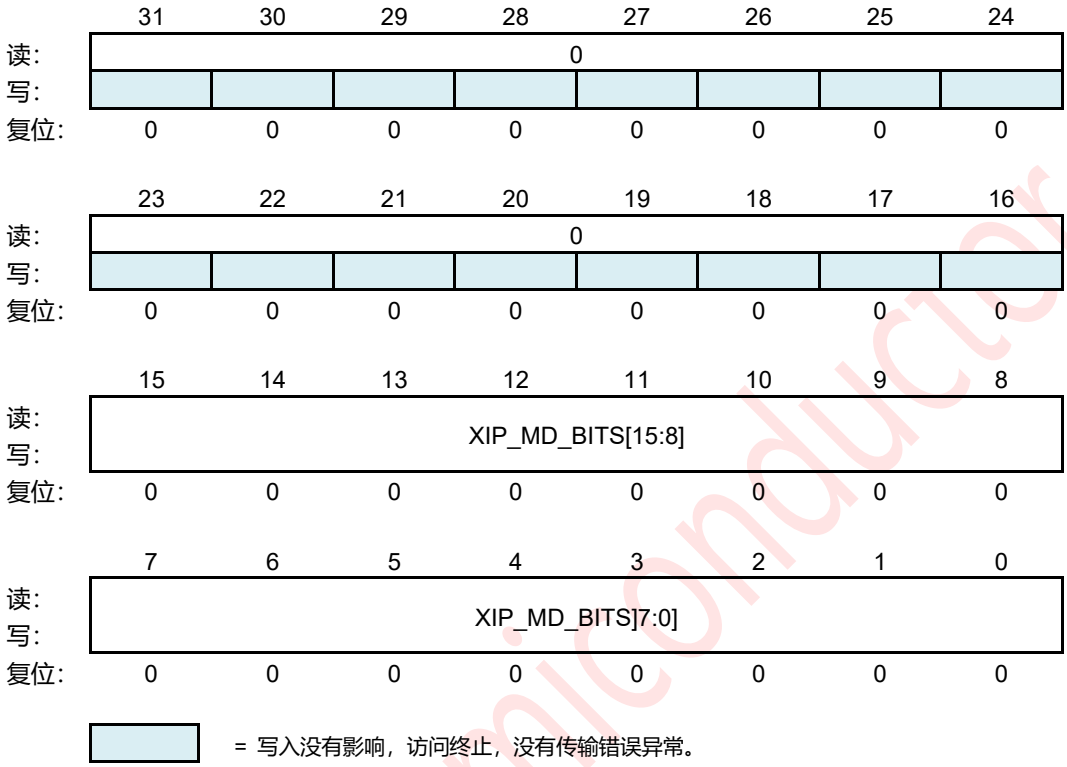


图 21-29：XIP 模式位（XIPMBR）

该寄存器携带地址阶段后 XIP 模式发送的模式位。这是一个 8 位寄存器，只有在 SSIENR 寄存器设置为 0 时才能写入。

XIP_MD_BITS[15:0]

XIP 传输地址阶段后发送 XIP 模式位。

21.6.2.28. XIP 中初始寄存器（XIPIIR）

地址：QSPIn_BASEADDR + 0x0000_0100

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	INCR_INST[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	INCR_INST[7:0]							
写:								
复位:	0	1	1	0	1	0	1	1

= 写入没有影响，访问终止，没有传输错误异常。

图 21-30：XIP 国际寄存器（XIPIIR）

此寄存器仅在 SSIC_XIP_INST_EN 等于 1 时有效。此寄存器用于存储在 AHB 接口上请求 INCR 事务中使用的指令操作代码。当启用了 SSI（SSIC_EN = 1）时，不可能写入此寄存器。通过写入 SSIENR 寄存器来启用和禁用 SSI。

INCR_INST[15:0] — XIP INCR transfer opcode.

当 SPI_CTRLR0。XIP_INST_EN 位设置为 1，SSI 为所有 XIP 传输发送指令，该寄存器字段存储在 AHB 总线上请求 INCR 类型传输时要发送的指令操作码。在指令阶段要发送的位数由 SPI_CTRL0 决定。INST_L 字段。

21.6.2.29. XIP 封装初始寄存器（XIPWIR）

地址：QSPIn_BASEADDR + 0x0000_0104

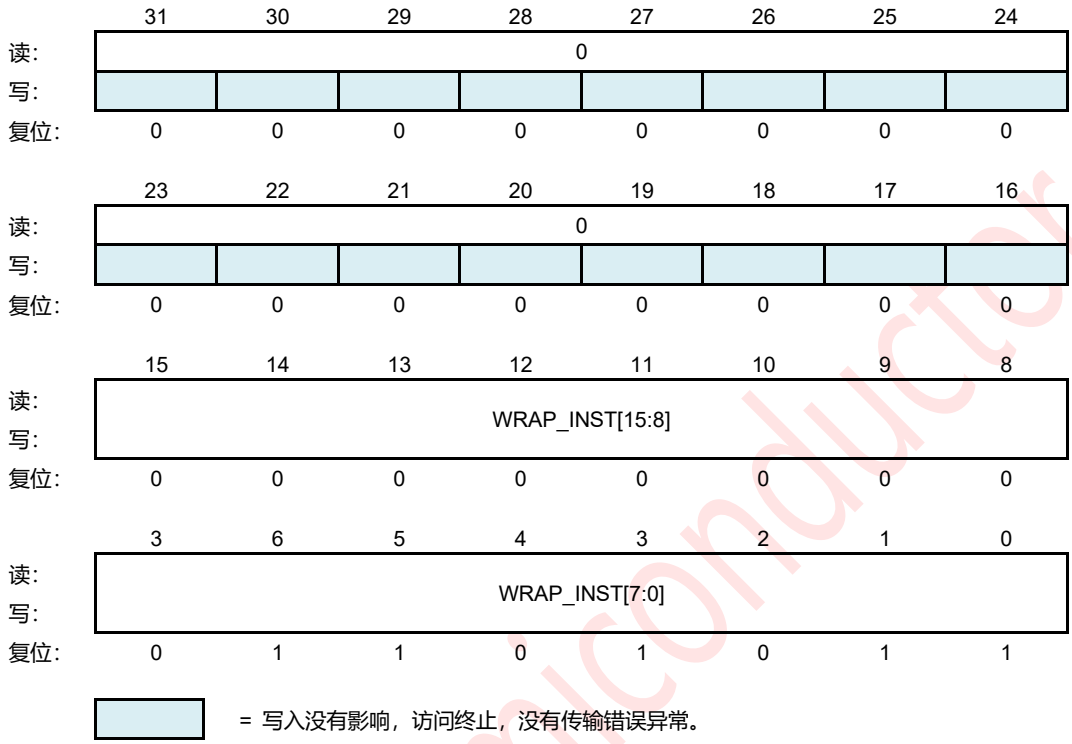


图 21-31：XIP 封装初始寄存器（XIPWIR）

此寄存器仅在 SSIC_XIP_INST_EN 等于 1 时有效。此寄存器用于存储在 AHB 接口上请求时在 WRAP 事务中使用的指令操作代码。当启用了 SSI（SSIC_EN = 1）时，不可能写入此寄存器。通过写入 SSIENR 寄存器来启用和禁用 SSI。

WRAP_INST[15:0] — XIP WRAP transfer opcode.

当 SPI_CTRLR0。XIP_INST_EN 位被设置为 1，SSI 为所有的 XIP 传输发送指令，这个寄存器字段存储在 AHB 总线上请求 WRAP 类型传输时要发送的指令操作码。在指令阶段要发送的位数由 SPI_CTRL0 决定。INST_L 字段。

21.6.2.30. XIP 控制寄存器（XIPCR）

地址：QSPIn_BASEADDR + 0x0000_0108

	31	30	29	28	27	26	25	24
读:	0	0	XIP_PREFETCH_EN	0	XIP_MBL[1:0]		保留	
写:								
复位:	0	0	1	0	1	0	0	0

	23	22	21	20	19	18	17	16
读:	CONT_XFER_EN	INST_EN	保留				WAIT_CYCLES[4:3]	
写:								
复位:	1	1	0	0	0	0	0	1

	15	14	13	12	11	10	9	8
读:	WAIT_CYCLES[2:0]			MD_BITS_EN	0	INST_L[1:0]		0
写:								
复位:	0	0	0	0	0	1	0	0

	7	6	5	4	3	2	1	0
读:	ADDR_L[3:0]				TRANS_TYPE		FRF	
写:								
复位:	0	1	1	0	0	0	1	0

= 写入没有影响，访问终止，没有传输错误异常。

图 21-32：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 XIP 传输将在并发模式下使用的控制信息。

XIP_PREFETCH_EN

1 = 在 SSI 中启用 XIP 预取功能。

0 = 在 SSI 中禁用 XIP 预取功能。

XIP_MBL[1:0] — XIP Mode bits length.

设置 XIP 模式操作中的模式位的长度。这些位仅在 XIP_CTRL 时有效。XIP_MD_BIT_EN 被设置为 1。

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 = 在 XIP 模式下启用连续传输。

0 = 禁用 XIP 模式下的连续传输。

INST_EN

1 = XIP 传输将有指令阶段。

0 = XIP 传输将没有指令阶段。

WAIT_CYCLES[4:0] — Wait cycles in Dual/Quad/Octal mode between control frames transmit and data reception. Specified as number of SPI clock cycles.

MD_BITS_EN — Mode bits enable in XIP mode.

1 = 在地址阶段之后的插入模式位。

0 = 在地址阶段后的无模式位。

INST_L[1:0]

双/八进制指令长度。

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[3:0]

此位定义了要传输的地址的长度。只有在这么多的位被编程到 FIFO 后，传输才能开始。

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[1:0]

地址和指令传输格式。

选择 SSI 是否将在标准 SPI 模式或在 CTRLR0 中选择的 SPI 模式下传输指令/地址。SPI_FRF 字段。

0x0（TT0）：指令和地址将以标准 SPI 模式发送。

0x1（TT1）：指令将以标准 SPI 模式发送，地址将以 XIP_CTRL 指定的模式发送。SPI_FRF.

0x2（TT2）：指令和地址都将以 XIP_CTRL 指定的方式发送。FRF.

0x3（TT3）：保留。

FRF[1:0] — SPI Frame Format

选择要传输/接收数据的数据帧格式。

0x0（RSVD）：保留

0x1（SPI_DUAL）：双 SPI 格式

0x2（SPI_QUAD）：四轴 SPI 格式

0x3（SPI_OCTAL）：八进制 SPI 格式

21.6.2.31. XIP 从属服务器启用寄存器（XIPSER）

地址：QSPIn_BASEADDR + 0x0000_010C

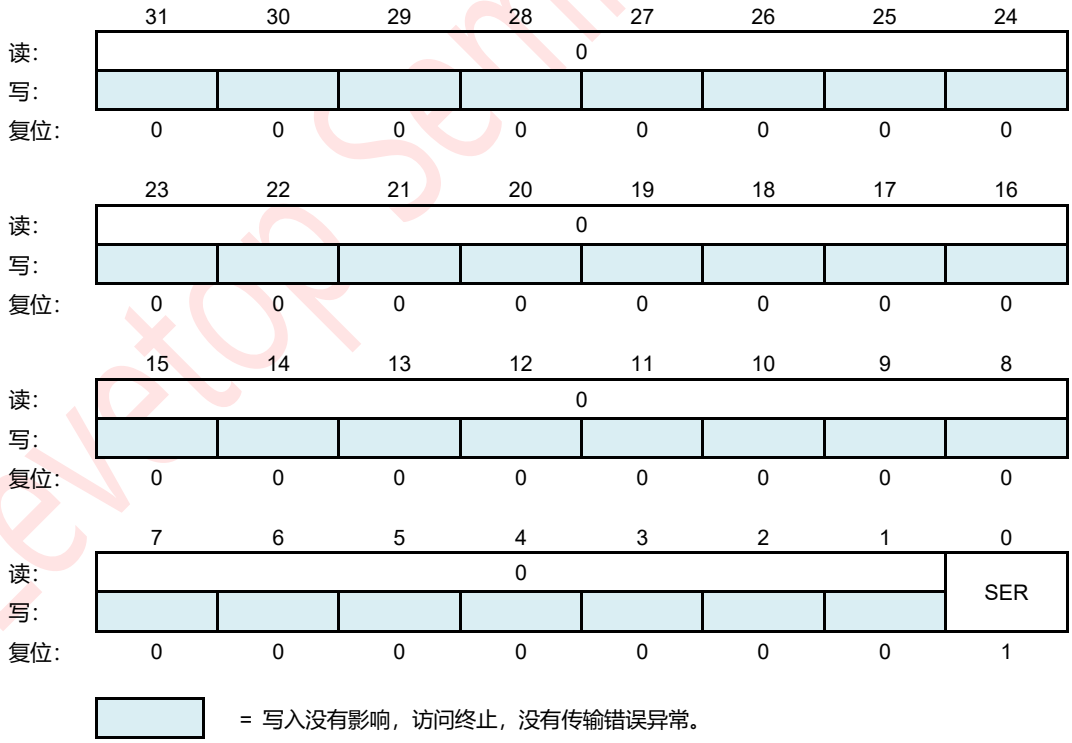


图 21-33：XIP 从属服务器启用寄存器（XIPSER）

此寄存器仅在 SSIC_CONCURRENT_XIP_EN 等于 1 时有效。该寄存器使单个从服务器能够从 SSI 主服务器中选择输出行 XIP 操作模式。在 SSI 主服务器上最多有 16 个从属选择输出引脚。当 SSI 繁忙或 SSIC_EN = 1 时，您无法写入此寄存器。

SER — Slave Select Enable Flag.

该寄存器中的每个位对应于 SSI 主服务器的从选择行（ss_x_n）。当设置这个寄存器中的一个位(1)时，当 XIP 传输开始时，主服务器对应的从选择行被激活。应该注意的是，在 XIP 传输开始之前，设置或清除这个寄存器中的位对相应的从属选择输出没有影响。在开始传输之前，您应该启用此寄存器中与主服务器希望与其通信的从设备相对应的位。当不在广播模式下操作时，在此字段中只能设置一个位。

1 = 选择

0 = 未选择

XIP 接收 FIFO 溢出中断清除寄存器（XRXIOCR）

21.6.2.32. XIP 接收 FIFO 溢出中断清除寄存器（XRXIOCR）

地址：QSPIn_BASEADDR + 0x0000_0110

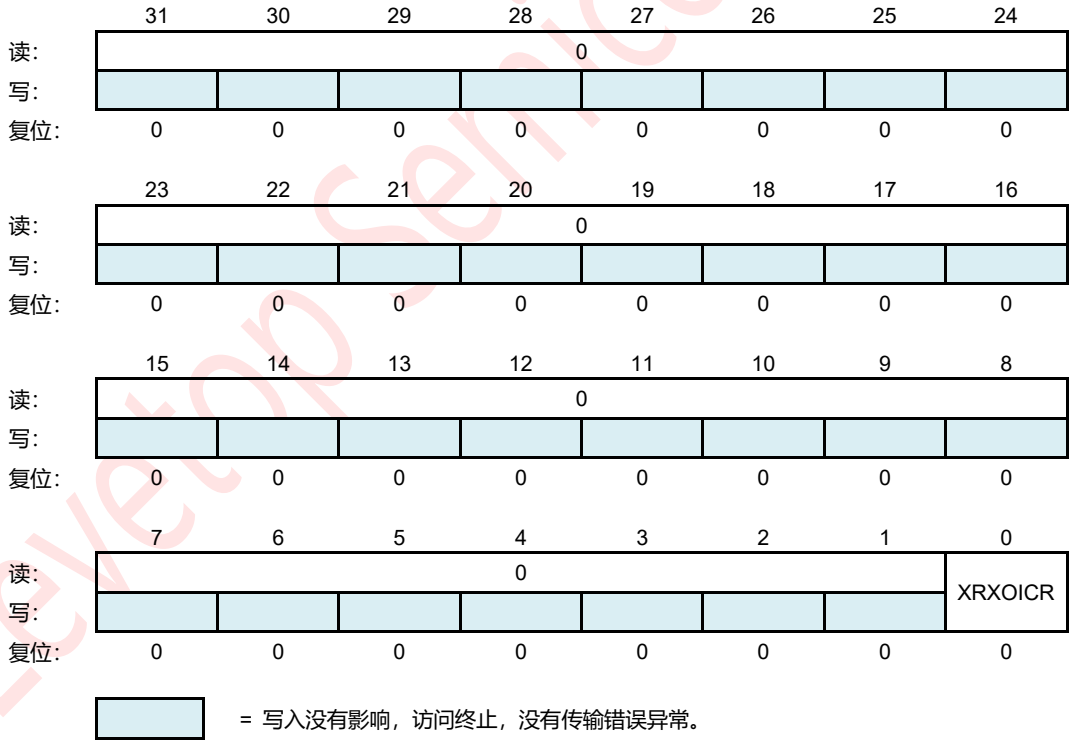


图 21-34：XIP 接收 FIFO 溢出中断清除寄存器（XRXIOCR）

XRXOICR — Clear XIP Receive FIFO Overflow Interrupt.

这个寄存器反映了中断的状态。从这个寄存器读取会清除 ssi_xrxo_intr（_n）中断；写入没有效果。

21.6.2.33. XIP 连续传输超时寄存器（XIPCTTOR）

地址：QSPIn_BASEADDR + 0x0000_0114

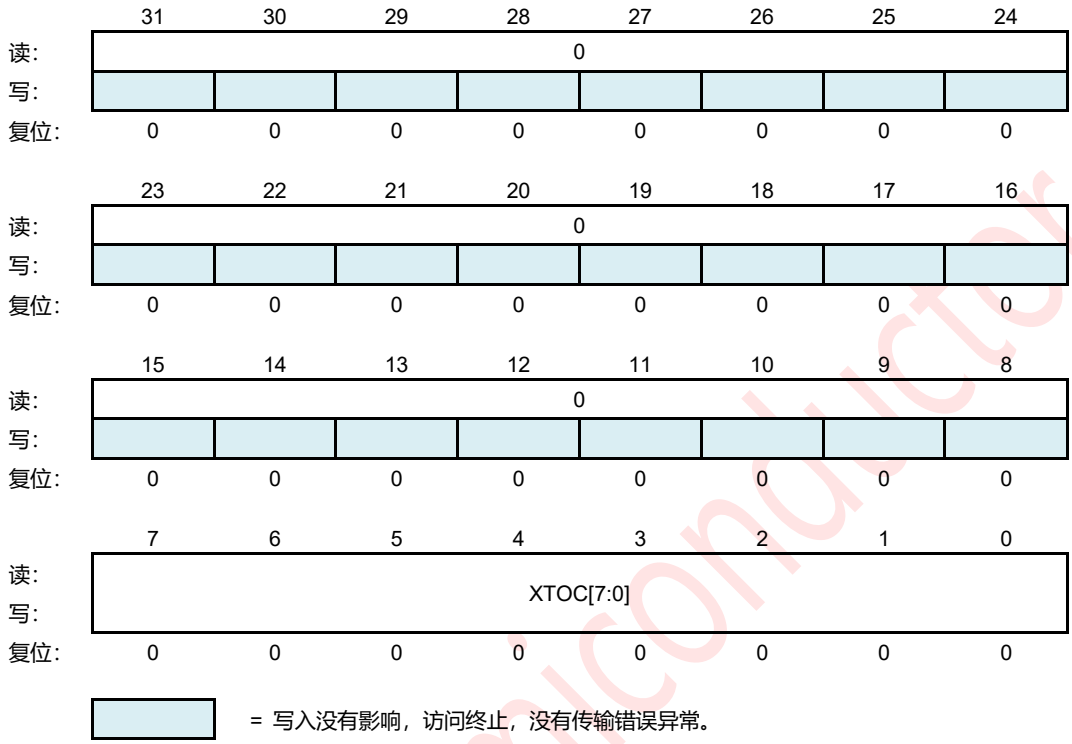


图 21-35：XIP 连续传输超时寄存器（XIPCTTOR）

XIP 计数下寄存器为连续模式。该计数器用于在连续传输模式下取消选择从属服务器。当启用 SSI 时，不可能写入此寄存器（SSIC_EN = 1）。通过写入 SSIENR 寄存器来启用和禁用 SSI。

XTOC[7:0] — XIP time out value in terms of hclk.

一旦在连续 XIP 模式中选择了从属器，如果在计数器中没有指定的时间请求，该计数器将用于取消选择从属器。

21.7. 功能描述

21.7.1. 主模式

此模式允许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。下图显示了 SSI 配置为串行主服务器的示例，以及串行总线上配置为串行从线服务器的所有其他设备。

由 SSI 产生和控制的串行位率时钟在 sclk_out 线上被输出。当 SSI 被禁用（SSIC_EN = 0）时，可能不会发生串行传输，并且 sclk_out 处于“非活动”状态，这是由它操作的串行协议定义的。

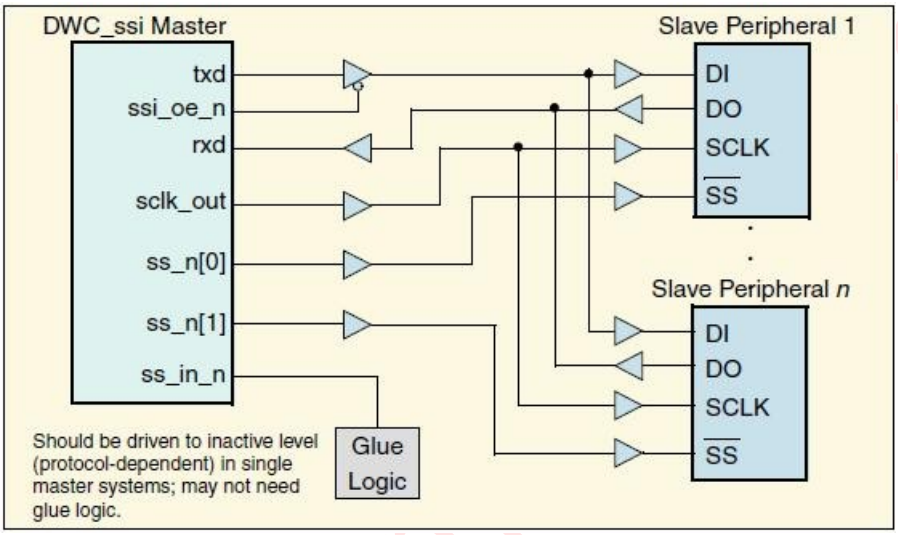


图 21-36：SSI 配置为主设备

21.7.2. 时钟比率

SSI 致力于一个过采样的体系结构。对于主操作模式，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）的最大频率为 ssi_clk 频率的一半。这是为了允许移位控制逻辑捕获 sclk_out 的一个时钟边缘上的数据，并在相反的边缘上传播数据。

sclk_out 的频率可以由以下方程推导出。

$$F_{sclk\_out} = \frac{F_{ssi\_clk}}{SCKDV}$$

SCKDV 是一个可编程寄存器，保持在 0-65,534 范围内的任何偶数值。如果是 SCKDV = 0，则 sclk_out 将被禁用。

21.7.3. 接收和传输 FIFO 缓冲区

SSI 使用的 FIFO 缓冲区是内部的 d 型人字拖。由于串行规范，传输和接收 FIFO 缓冲器的宽度都固定在 32 位，这说明串行传输（数据帧）的长度可以是 4 到 32 位。当写入传输 FIFO 缓冲区时，大小小于 32 位的数据帧必须是正确正确的。移位控制逻辑自动正确证明接收 FIFO 缓冲区中的接收数据。

21.7.3.1. 传输 FIFO

传输的 FIFO 通过 AHB 写入命令加载到 SSI 数据寄存器（DR）。数据通过移位控制逻辑从发射 FIFO 中弹出（移除）到发射移位寄存器中。当 FIFO 中的条目数小于或等于 FIFO 阈值时，发送 FIFO 生成 FIFO 空中断请求（ssi_txe_intr）。通过可编程寄存器 TXFTLR 设置的阈值决定了产生中断的 FIFO 条目的级别。该阈值允许您向处理器提供发送 FIFO 几乎为空的早期指示。如果您试图将数据写入已经完整的传输 FIFO，则会生成传输 FIFO 溢出中断（ssi_txo_intr）。

21.7.3.2. 接收 FIFO

数据通过 AHB 读取命令从接收的 FIFO 弹出到 SSI 数据寄存器（DR）。通过移位控制逻辑从接收移位寄存器加载接收 FIFO。当 FIFO 中的条目数大于或等于 FIFO 阈值加 1 时，接收 FIFO 生成一个 FIFO-满中断请求（ssi_rxf_intr）。通过可编程的寄存器 RXFTLR 设置的阈值决定了生成中断的 FIFO 条目的级别。

该阈值允许您向处理器提供接收 FIFO 已接近满的早期指示。当接收移位逻辑尝试将数据加载到一个完全完整的接收 FIFO 中时，将生成一个接收 FIFO 溢出中断（ssi_rxo_intr）。然而，这些新收到的数据却丢失了。如果您试图从空的接收 FIFO 中读取，则会生成接收 FIFO 下溢（UnderFlow)中断（ssi_rxu_intr）。这将提醒处理器读取的数据无效。

21.7.4. DMA 操作

SSI 具有可选的内置 DMA 功能，可以在配置时进行选择；它有一个到 DMA 控制器的握手接口来请求和控制传输。AHB 总线用于执行与 DMA 之间的数据传输。虽然 SSI DMA 操作以一种通用的方式设计，以尽可能容易地适应任何 DMA 控制器，但它被设计为无缝地工作，并最好地使用。要在 SSI 上启用 DMA 控制器接口，您必须写入 DMA 控制寄存器（DMACR）。将一个 1 写入 DMACR 寄存器的 TDMAE 位字段，可以实现 SSI 传输握手接口。将 1 写入 DMACR 寄存器的 RDMAE 位字段，允许 SSI 接收握手接口。

21.7.5. SSI 中断

SSI 中断的描述如下：

传输 FIFO 空中断（ssi_txe_intr）-当传输 FIFO 等于或低于其阈值，并需要服务以防止运行不足时进行设置。通过软件可编程寄存器设置的阈值决定了产生中断的传输 FIFO 条目的级别。当数据被写入传输 FIFO 缓冲区时，硬件会清除此中断，使其超过阈值级别。

传输 FIFO 溢出中断（ssi_txo_intr）-当 AHB 访问在传输 FIFO 被完全填充后尝试写入时进行设置。当设置时，从 AHB 写入的数据将被丢弃。这个中断保持设置，直到您读取传输 FIFO 溢出中断清除寄存器（TXOICR）。

接收 FIFO 完全中断（ssi_rxf_intr）-当接收 FIFO 等于或高于其阈值加上 1，并且需要提供服务以防止溢出时进行设置。通过软件可编程寄存器设置的阈值决定了产生中断的接收 FIFO 条目的级别。当从接收 FIFO 缓冲区读取数据时，硬件会清除此中断，使其低于阈值水平。

接收 FIFO 溢出中断（ssi_rxo_intr）-当接收逻辑在尝试将数据完全填充后的数据输入接收 FIFO 时进行设置。在设置后，新接收到的数据将被丢弃。此中断保持设置，直到读取接收 FIFO 溢出中断清除寄存器（RXOICR）。

接收 FIFO 下溢中断（ssi_rxu_intr）-当 AHB 访问试图从接收 FIFO 为空时读取时设置。设置后，将从接收 FIFO 读回零。这个中断保持设置，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）-上述所有中断请求的结果。要屏蔽此中断信号，您必须屏蔽所有其他 SSI 中断请求。

21.7.6. 增强的 SPI 模式

SSI 支持使用 SSIC_SPI_MODE 配置参数的 SPI 的双、四元和八进制模式。该参数的可能值是标准、双 SPI、四 SPI 和八进 SPI 模式。当该参数选择双、四、四或八进制模式时，TXD、RXD 和 ssi_oe_n 信号的宽度分别变为 2、4 或 8。因此，数据被转移到多个线路上，从而增加了总体吞吐量。除了 TXD、RXD 和 ssi_oe_n 信号的宽度外，双 SPI、四位或八位 SPI 模式的功能类似。可以使用 CTRLR0 选择操作模式（写/读取）。TMOD 字段。

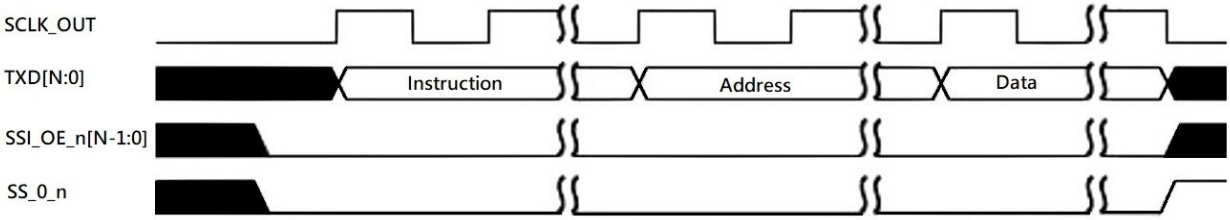


图 21-37：典型的写入操作双/四/八进制 SPI 模式

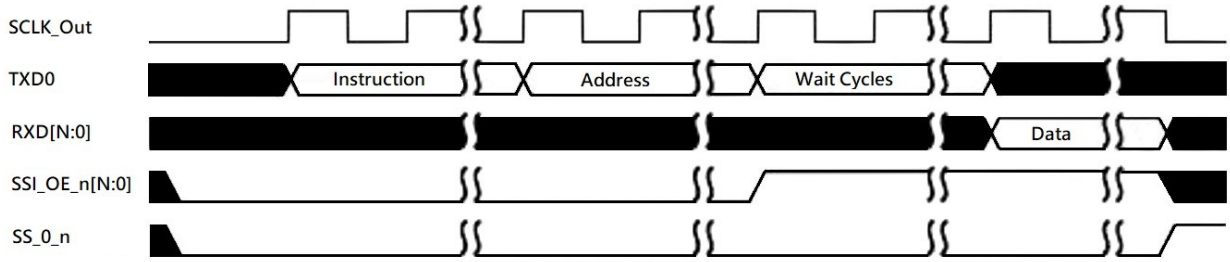


图 21-38：典型的读取操作双/四/八进制 SPI 模式

21.7.7. 就地执行（XIP）模式

SSI 提供了一个可以直接从 AHB 事务中执行内存读取操作的功能。这被称为就地执行模式，在其中，SSI 充当到 SPI 内存的内存映射接口。通过选择配置参数 SSIC_XIP_EN，可以在 SSI 中启用 XIP 模式。这包括一个 AHB 接口上额外的边带信号 xip_en。这个信号电平决定了 AHB 传输是寄存器读写还是 XIP 读取。在 XIP 操作期间，只支持 AHB read。如果 xip_en 信号被驱动为 1，那么 SSI 期望在 AHB 接口上发出一个读取请求。此请求被转换为串行接口上的 SPI 读取。一旦收到数据，它就会返回到同一事务中的 AHB 接口。haddr 用于推导出要在 SPI 接口上发送的地址。某些设备期望在 XIP 传输期间存在指令阶段。SSI 支持在 XIP 操作模式期间包含一些固定的指令集。

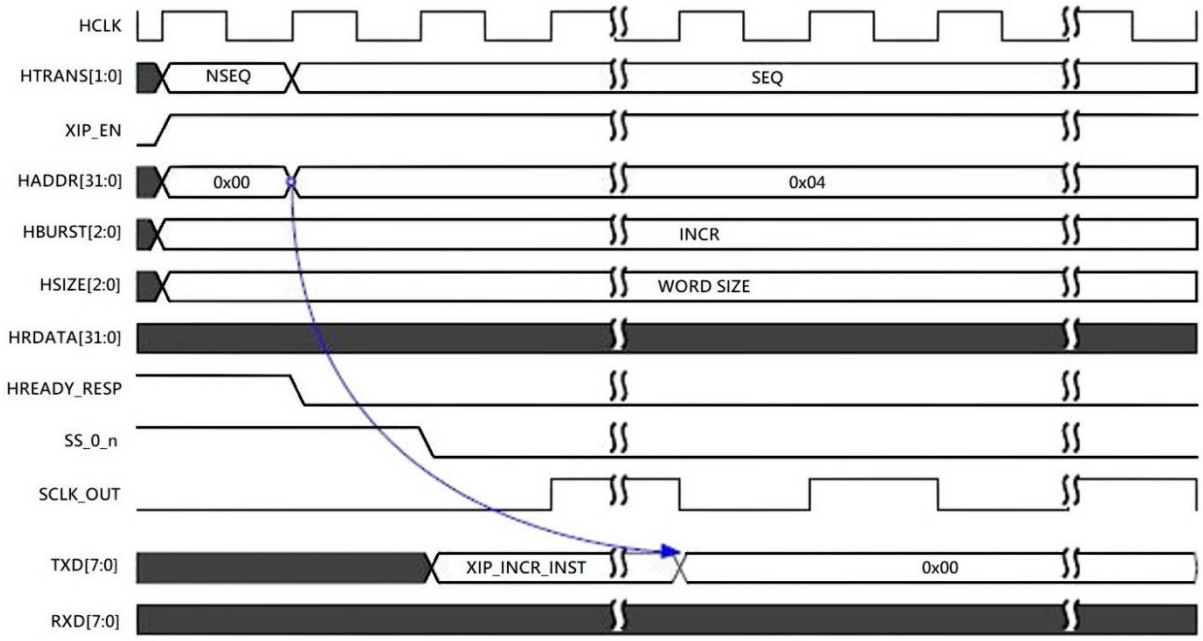


图 21-39：XIP 传输与指令阶段

21.7.8. 在 XIP 中的连续传输模式

当 SSI 接收到一个 XIP 请求时，来自 AHB 接口的地址将被直接传输到 SPI 接口上。AHB 接口上的每个新传输（XIP Read）都以相同的方式处理。因此，对于每个请求，都必须向设备发送一个新的地址，从而导致系统的延迟。

如果一个存储器设备允许在 XIP 读取传输之间拉伸从属选择信号，那么 SSI 可以被编程为连续的 XIP 模式，以实现更高的性能。在这种模式下，主机通过确保该命令和地址不被重新传输，并且主机控制器不需要等待这些突发之间的任何虚拟周期，将两个或多个 AHB 突发请求融合到单个 SPI 命令中。

当此功能被启用时，一旦收到第一个 XIP 命令，SSI 功能就处于连续 XIP 模式。对于第一个 XIP 传输，该地址将在 SPI 接口上发送。在接收到请求的数据后，SSI 继续保持从从被选择，时钟（sclk_out）仍然处于默认状态。对于 AHB 接口上的后续 XIP 传输，SSI 恢复时钟（sclk_out），命令和地址都不会传输到 SPI 接口，以及将立即从设备获取的数据（没有虚拟周期）。

•在连续读取模式下，不支持未定义的 INCR（hburst = 001）突发。

在连续传输过程中，由于从设备一直被选择，所以在从设备上消耗了大量的功率。为了避免这种情况，SSI 提供了一个配置选项，使看门狗计时器在计数器用完后取消选择从属器。

SSI 可以在以下条件下取消选择从属服务器：

- 在 XIP 接口上接收非 XIP 命令（实际上任何 xip_en 驱动为 0 的 AHB 事务）。
- 当 AHB 事务是一个非连续的地址时，从属选择将被删除，然后 SSI 启动一个新的 XIP 请求。
- 在 XIP_CNT_TIME_OUT 寄存器中指定的时间段内，SSI 没有检测到 AHB 接口上的任何 XIP 传输。

21.7.9. 在 XIP 操作中的数据预取

使用 SSI 中的数据预取功能，控制器在当前 XIP 事务中对连续突发的数据进行预取。如果对连续地址发出下一个事务请求，则可以直接从 RX FIFO 读取数据，而不是等待将新的地址和数据发送到设备。这提高了系统的整体性能。

要预取的数据量应该等于最后一个 AHB 请求的突发长度或 FIFO 深度（以较低者为准）。例如，如果 AHB 定义了从地址 0x00 开始的 16 个突发长度，那么 SSI 获取 16 个突发以完成当前传输，然后再次获取 16 个数据帧并保存到数据寄存器中。如果 AHB 总线再次请求从结束地址开始的数据以进行最后一次传输，那么其余的数据将从 RX FIFO 本身发送到设备。同时，SSI 再次启动向设备的 XIP 传输，以预取下一个数据块。如果 AHB 主服务器没有定义连续地址，则当前数据将从 FIFO 中刷新，并且控制器将启动一个新的事务。

当 SSI 完成当前的突发，并为下一个突发预取数据时，可能会放置一个新的 XIP 请求。在这种情况下，SSI 可以终止当前的传输，或者根据地址增加要获取的数据拍数。

- 如果启用了 XIP 预抓取，则不允许请求增量传输未定义长度（hburst = 3‘b001）的 AHB 请求。

22. 脉冲宽度调制器（PWM）

22.1. 功能介绍

LT7589 嵌入了 4 个 PWM 定时器。4 个 PWM-定时器有 2 个预刻度，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。每个都可以用作计时器和发出中断。

每个两个 PWM-Timers 共享相同的预比例尺。时钟分频器为每个计时器提供 5 个时钟源（1、1/2、1/4、1/8、1/16)。每个计时器中的 16 位计数器接收来自时钟选择器的时钟信号，并可用于处理一个 PWM 周期。16 位比较器将计数器中的数与之前加载的寄存器中的阈值数进行比较，以生成 PWM 占空比。来自时钟分频器的时钟信号被称为 PWM 时钟。死区发生器利用 PWM 时钟作为时钟源。一旦启用了死区发生器，两个 PWM 定时器的输出将被阻塞。两个输出引脚均作为死区发电机的输出信号，用以控制片外电源装置。比较器的值用于脉宽调制。当下计数器的值与比较寄存器的值相匹配时，计数器控制逻辑会改变输出电平。

每个 PWM-Timer 都包含一个捕获通道。捕获 0 和 PWM 0 共享一个包含在 PWM 0 中的计时器；而捕获 1 和 PWM 1 共享另一个计时器，等等。因此，用户必须在打开捕获特性之前设置 PWM-Timer。启用捕获特性后，当输入通道有上升转换时，捕获总是锁定 PWM 计数器到 CRLR，当输入通道有下降转换时，捕获总是锁定 PWM 计数器到 CFLR。捕获通道 0 中断可以通过设置 CCR0[1]（上升锁存中断启用）和 CCR0[2]（下降锁存中断启用）来确定中断触发条件。捕获通道 1、2 和 3 也具有相同的功能。每当捕获问题中断 0/1/2/3 时，PWM 计数器 0/1/2/3 将在此时重新加载。最大捕获频率应由中断进程时间来决定。如果中断处理时间为 T0，则捕获信道输入信号在 T0 不变化，最大捕获频率为 1/T0。

从 PWM 到中断控制器（INTC）只有四个中断。PWM 0 和捕获 0 共享相同的中断；PWM1 和捕获 1 共享相同的中断，等等。因此，PWM 功能和捕获功能不能同时使用。

22.2. PWM 特点

脉冲宽度调制器包括以下显著特点：

- 可编程周期
- 可编程占空比
- 两个死区产生器
- 捕获功能
- 引脚可以配置为通用 I/O

22.3. 方块图

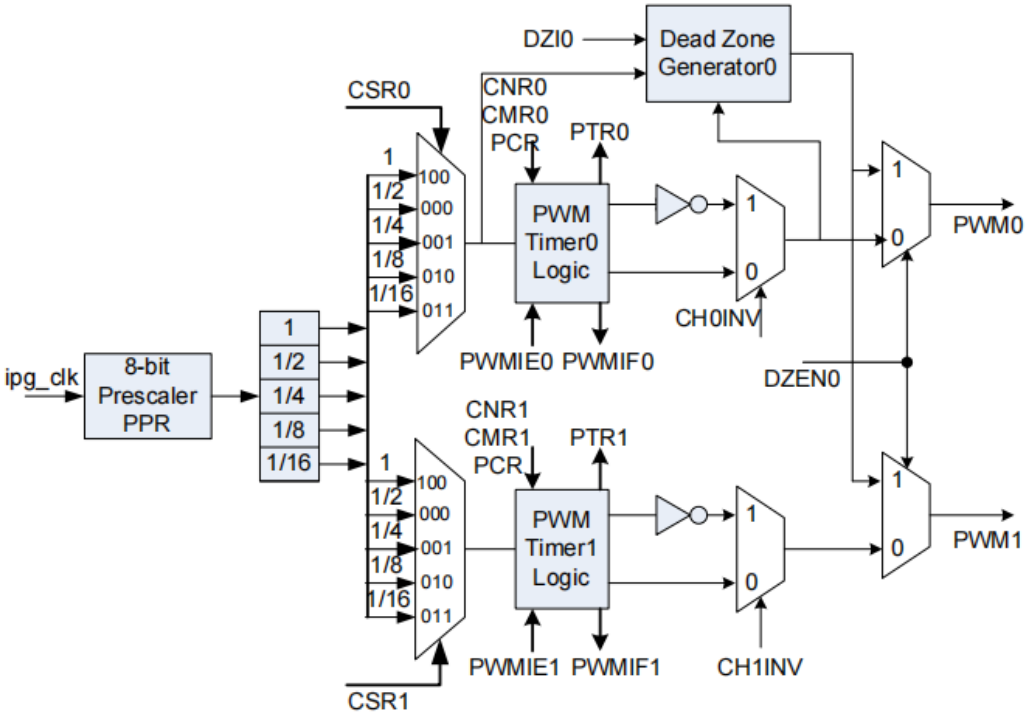


图 22-1：PWM 方块图

22.4. 信号描述

表 22-1：PWM 信号描述

名称	I/O	宽度	复位状态	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 用作通用的输入/输出，也用作 PWM 发送输出或捕获输入。

在默认状态下，它被用作通用的输入端口。

22.5. 内存映射和寄存器

PWM 模块内存映射情况见下表 PWM0 的基本地址为 0x400D_0000，PWM1 为 0x400E_0000。本章节描述了 PWM 的内存映射和寄存器结构。

22.5.1. 内存映射

表 22-2：模块内存映射

地址	Bit[31:0]	访问权限(1)
0x0000	PWM Pre-scale Register (PPR)	S/U
0x0004	PWM Clock Select Register (PCSR)	S/U
0x0008	PWM Control Register (PCR)	S/U
0x000C	PWM Counter Register0 (PCNR0)	S/U
0x0010	PWM Comparator Register0 (PCMR0)	S/U
0x0014	PWM Timer Register0 (PTR0)	S/U
0x0018	PWM Counter Register1 (PCNR1)	S/U
0x001C	PWM Comparator Register1 (PCMR1)	S/U
0x0020	PWM Timer Register1 (PTR1)	S/U
0x0024	PWM Counter Register2 (PCNR2)	S/U
0x0028	PWM Comparator Register2 (PCMR2)	S/U
0x002C	PWM Timer Register2 (PTR2)	S/U
0x0030	PWM Counter Register3 (PCNR3)	S/U
0x0034	PWM Comparator Register3 (PCMR3)	S/U
0x0038	PWM Timer Register3 (PTR3)	S/U
0x003C	PWM Interrupt Enable Register (PIER)	S/U
0x0040	PWM Interrupt Flag Register (PIFR)	S/U
0x0044	PWM Capture Control Register0 (PCCR0)	S/U
0x0048	PWM Capture Control Register1 (PCCR1)	S/U
0x004C	PWM Capture Rising Latch Register0 (PCRLR0)	S/U
0x0050	PWM Capture Falling Latch Register0 (PCFLR0)	S/U
0x0054	PWM Capture Rising Latch Register1 (PCRLR1)	S/U
0x0058	PWM Capture Falling Latch Register1 (PCFLR1)	S/U
0x005C	PWM Capture Rising Latch Register2 (PCRLR2)	S/U
0x0060	PWM Capture Falling Latch Register2 (PCFLR2)	S/U
0x0064	PWM Capture Rising Latch Register3 (PCRLR3)	S/U
0x0068	PWM Capture Falling Latch Register3 (PCFLR3)	S/U
0x006C	PWM Port Control Register (PPCR)	S/U

注意(1)：S/U = CPU 主管或用户模式访问。

22.5.2. 寄存器描述

22.5.2.1. PWM 预缩放寄存器（PPR）

寄存器（PPR）用于设置预调节器和设置死区长度。

地址：PWMn_BASEADDR + 0x0000_0000

	31	30	29	28	27	26	25	24
读: 写:	DZI1[7:0]							
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读: 写:	DZI0[7:0]							
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读: 写:	CP1[7:0]							
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读: 写:	CP0[7:0]							
复位:	0	0	0	0	0	0	0	0

图 22-2：PWM 比例前寄存器（PPR）

DZI1[7:0] — Dead zone interval register 1 for PWM2 and PWM3

这 8 位决定了死区的长度。从时钟选择器 1 接收到死区长度的 1 单位时间。

DZI0[7:0] — Dead zone interval register 0 for PWM0 and PWM1

这 8 位决定了死区的长度。从时钟选择器 0 接收到死区长度的 1 个单位时间。

CP1[7:0] — Clock pre-scale 1 for PWM Timer 2 & 3

时钟输入除以（CP1 + 1），然后输入到 PWM 计时器 2 和 3 的计数器。

CP0[7:0] — Clock pre-scale 0 for PWM Timer 0 & 1

时钟输入除以（CP0 + 1），然后输入到 PWM 计时器 0 和 1 的计数器。

22.5.2.2. PWM 时钟选择寄存器（PCSR）

时钟分频器为每个计时器提供 5 个时钟源（1、1/2、1/4、1/8、1/16)。每个计时器从时钟分频器接收它自己的时钟信号，该分频器接收来自 8 位预缩放的时钟。

地址：PWMn_BASEADDR + 0x0000_0004

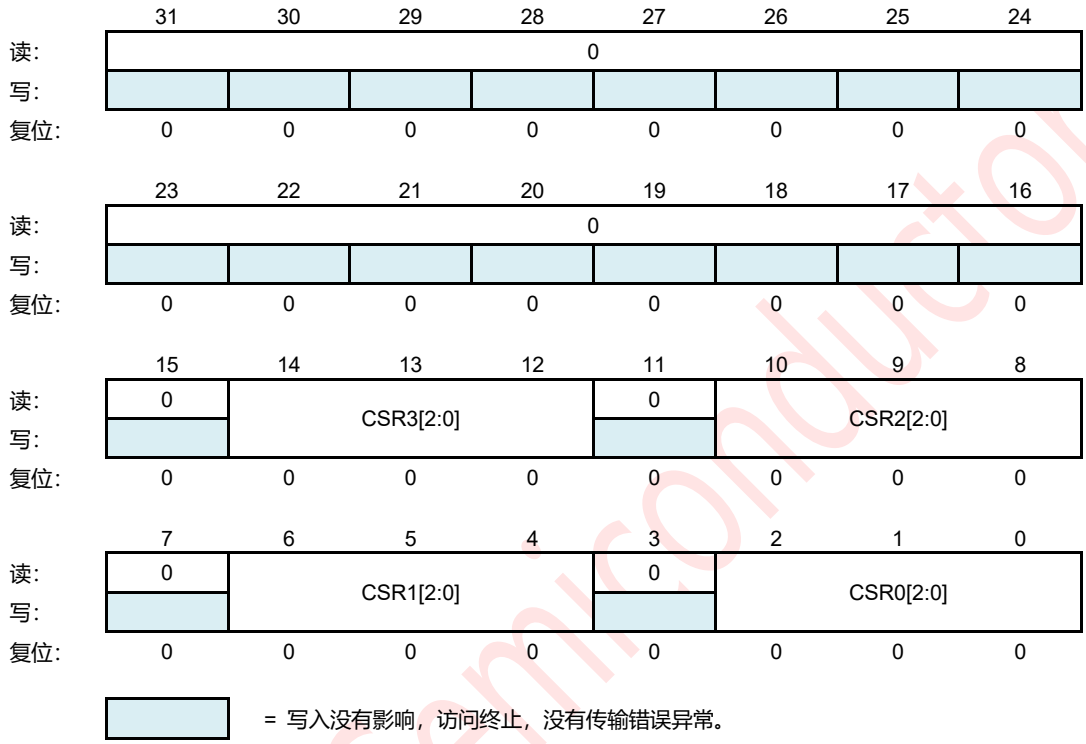


图 22-3：PWM 时钟选择寄存器（PCSR）

CSR3[2:0] — Timer 3 Clock Source Selection

选择计时器 3 选择时钟输入。

表 22-3：计时器 3 时钟源选择

CSR3[2:0]	输入时钟除以
100~111	1
011	16
010	8
001	4
000	2

CSR2[2:0] — Timer 2 Clock Source Selection

选择计时器 2 选择时钟输入。与 CSR3 相同。

CSR1[2:0] — Timer 1 Clock Source Selection

为计时器 1 选择时钟输入。与 CSR3 相同。

CSR0[2:0] — Timer 0 Clock Source Selection

选择计时器 0 选择时钟输入。与 CSR3 相同。

22.5.2.3. PWM 对照寄存器（PCR）

这个寄存器是 PWM 控制寄存器。

地址：PWMn_BASEADDR + 0x0000_0008

	31	30	29	28	27	26	25	24
读:	0				CH3MOD	CH3INV	0	CH3EN
写:								
复位:	0	0	0	0	0	0	0	0

	23	19	18	20	16	18	17	16
读:	0				CH2MOD	CH2INV	0	CH2EN
写:								
复位:	0	0	0	0	0	0	0	0

	15	11	10	12	8	10	9	8
读:	0				CH1MOD	CH1INV	0	CH1EN
写:								
复位:	0	0	0	0	0	0	0	0

	7	3	2	4	0	2	1	0
读:	0		DZEN1	DZEN0	CH0MOD	CH0EN	0	CH0EN
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-4：PWM 控制寄存器（PCR）

CH3MOD — Timer 3 Auto-load/One-Shot Mode

1 = 自动加载模式

0 = 单次模式

提示：如果在该位有一个上升或下降的转换，它将导致 CNR3 和 CMR3 被清除。

CH3INV — Timer 3 Inverter ON/OFF

1 = 反向器打开

0 = 反向器关闭

CH3EN — Timer 3 Enable/Disable

1 = 启用

0 = 禁用

CH2MOD — Timer 2 Auto-load/One-Shot Mode

1 = 自动加载模式

0 = 单次模式

提示：如果在该位有上升或下降的转换，则 CNR2 和 CMR2 将被清除。

CH2INV — Timer 2 Inverter ON/OFF

1 = 反向器打开

0 = 反向器关闭

CH2EN — Timer 2 Enable/Disable

1 = 启用

0 = 禁用

CH1MOD — Timer 1 Auto-load/One-Shot Mode

1 = 自动加载模式

0 = 单次模式

提示：如果在该位有上升或下降的转换，则 CNR1 和 CMR1 将被清除。

CH1INV — Timer 1 Inverter ON/OFF

1 = 反向器打开

0 = 反向器关闭

CH1EN — Timer 1 Enable/Disable

1 = 启用

0 = 禁用

DZEN1 — Dead-Zone 1 Generator Enable/Disable

1 = 启用

0 = 禁用

提示：当启用 DZEN1 时，应禁用 CH3EN。因为通道 3 和通道 2 的输出都由通道 2 决定的。

DZEN0 — Dead-Zone 0 Generator Enable/Disable

1 = 启用

0 = 禁用

提示：当启用 DZEN0 时，应禁用 CH1EN。因为通道 1 和通道 0 的输出都由通道 0 决定的。

CH0MOD — Timer 0 Auto-load/One-Shot Mode

1 = 自动加载模式

0 = 单次模式

提示：如果在此位有一个上升或下降的转换，它将导致清除 CNR0 和 CMR0。

CH0INV — Timer 0 Inverter ON/OFF

1 = 反向器打开

0 = 反向器关闭

CH0EN — Timer 0 Enable/Disable

1 = 启用

0 = 禁用

22.5.2.4. PWM 计数器寄存器（PCNRn）

这些寄存器通过定义该周期内的计数脉冲的数量来控制 PWM 的周期。

地址：PWMn_BASEADDR + 0x0000_000C

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CNR0[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	CNR0[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

地址：PWMn_BASEADDR + 0x0000_0018

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CNR1[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	CNR1[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

地址：PWMn_BASEADDR + 0x0000_0024

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CNR2[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	CNR2[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

地址：PWMn_BASEADDR + 0x0000_0030

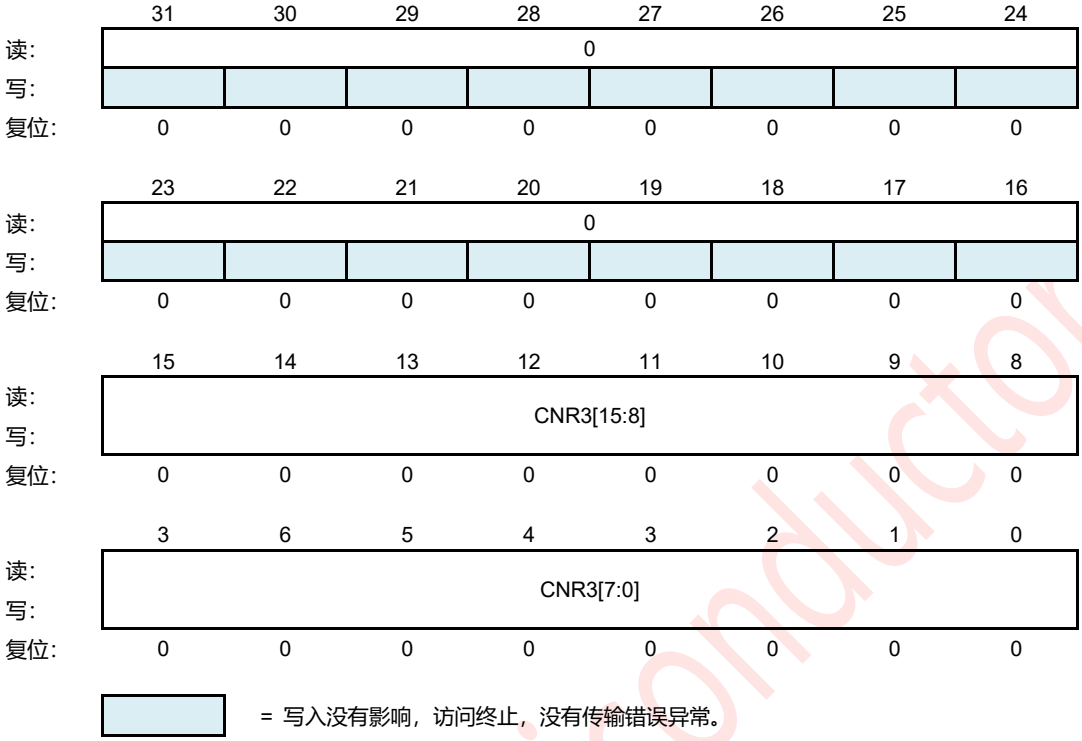


图 22-5：PWM 计数器寄存器（PCNR）

CNRx[15:0] — Loaded Value for PWM Counter/Timer

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，则 PWM 计数器/计时器将被停止。

2: 每当一个值被写入 CNR 时，它将在下一个 PWM 计数器周期中生效。

22.5.2.5. PWM 比较器寄存器（PCMRn）

这些寄存器定义了脉冲的宽度。当计数器与此寄存器中的值匹配时，输出将在此期间内复位。

地址：PWMn_BASEADDR + 0x0000_0010

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CMR0[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	CMR0[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

地址：PWMn_BASEADDR + 0x0000_001C

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CMR1[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	CMR1[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

地址：PWMn_BASEADDR + 0x0000_0028

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CMR2[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	CMR2[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

地址：PWMn_BASEADDR + 0x0000_0034

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CMR3[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	CMR3[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-6：PWM 比较器寄存器（PCMR）

CMRx[15:0] — PWM Comparator Register

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：高

CMR >= CNR ： PWM 的输出量总是很高的

CMR < CNR ： PWM 输出高 =（CMR + 1）单元

CMR = 0 ： PWM 输出高 = 1 单元

提示：

1: PWM tuty = CMR + 1。如果 CMR 等于零，则 PWM 占空比为 1

2: 每当一个值被写入 CMR 时，它将在下一个 PWM 计数器周期中生效。

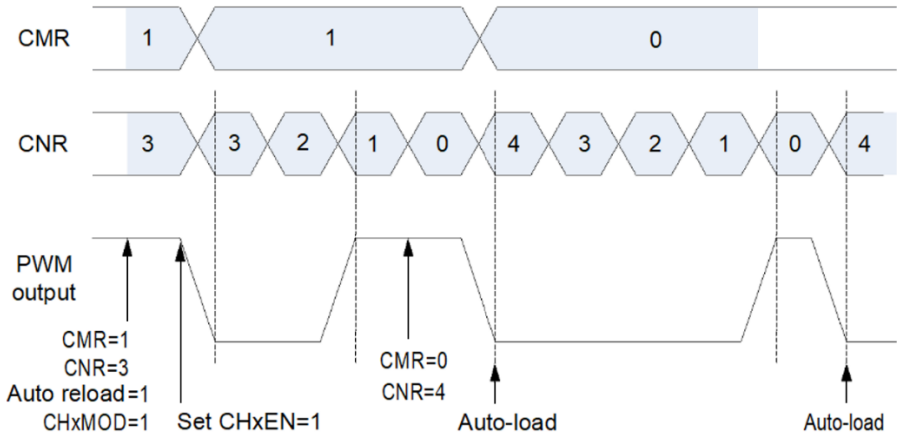


图 22-7：PWM 输出

调制 PWM 控制器输出占空比（CNR = 150）

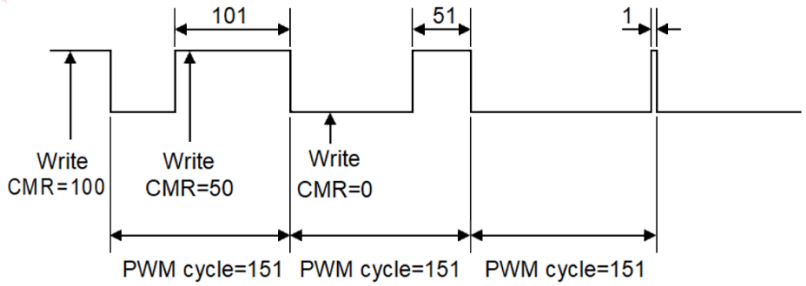


图 22-8：PWM 占空比

22.5.2.6. PWM 计时器寄存器（PTRn）

只读的 PWM 计时器寄存器保存当前的计数值。它可以随时阅读而不干扰柜台。

地址：PWMn_BASEADDR + 0x0000_0014

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	PTR0[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	PTR0[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

地址：PWMn_BASEADDR + 0x0000_0020

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	PTR1[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	4	3	5	4	3	2	1	0
读:	PTR1[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

地址：PWMn_BASEADDR + 0x0000_002C

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	PTR2[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	PTR2[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

地址：PWMn_BASEADDR + 0x0000_0038

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	PTR3[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	PTR3[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-9：PWM 计时器寄存器（PTR）

PTRx[15:0] — PWM Timer value

只读的 PTR 位保存当前的计数值。用户可以监控 PTR，以获得 16 位下计数器中的当前值。

22.5.2.7. PWM 中断启用寄存器（PIER）

这个寄存器用于启用 PWM 计时器中断。

地址：PWMn_BASEADDR + 0x0000_003C

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0				PIER3	PIER2	PIER1	PIER0
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-10：PWM 中断启用寄存器（PIER）

PIER3 — PWM Timer 3 Interrupt Enable

1 = 启用

0 = 禁用

PIER2 — PWM Timer 2 Interrupt Enable

1 = 启用

0 = 禁用

PIER1 — PWM Timer 1 Interrupt Enable

1 = 启用

0 = 禁用

PIER0 — PWM Timer 0 Interrupt Enable

1 = 启用

0 = 禁用

22.5.2.8. PWM 中断标志寄存器（PIFR）

这个寄存器用于指示 PWM 计时器中断标志。

地址：PWMn_BASEADDR + 0x0000_0040

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0				PIFR3	PIFR2	PIFR1	PIFR0
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-11：PWM 中断标志寄存器（PIFR）

PIFR3 — PWM Timer 3 Interrupt Flag.

当 PWM 计时器 3 计数为 0，而 PIER3 = 1 时，PIFR3 将被设置为 1。向此位写入 1 将清除 PIFR3。

1 = 中断标志已打开

0 = 中断标志已关闭

PIFR2 — PWM Timer 2 Interrupt Flag.

当 PWM 计时器 2 计数为 0，而 PIER2 = 1 时，PIFR2 将被设置为 1。向此位写入 1 将清除 PIFR2。

1 = 中断标志已打开

0 = 中断标志已关闭

PIFR1 — PWM Timer 1 Interrupt Flag.

当 PWM 计时器 1 计数为 0，并且 PIER1 = 1 时，PIFR1 将被设置为 1。写入 1 将清除 PIFR1。

1 = 中断标志已打开

0 = 中断标志已关闭

PIFR0 — PWM Timer 0 Interrupt Flag.

当 PWM 计时器 0 计数为 0，并且 PIER0 = 1 时，PIFR0 将被设置为 1。写入 1 将清除 PIFR0。

1 = 中断标志已打开

0 = 中断标志已关闭

22.5.2.9. PWM 捕获控制寄存器（PCCR0/1）

这些寄存器用于控制捕获功能。

地址：PWMn_BASEADDR + 0x0000_0044

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	CFLRD1	CRLRD1	0	CAPIF1	CAPCH1EN	FL_IE1	RL_IE1	INV1
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	CFLRD0	CRLRD0	0	CAPIF0	CAPCH0EN	FL_IE0	RL_IE0	INV0
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-12：PWM 捕获控制寄存器（PCCR0）

地址：PWMn_BASEADDR + 0x0000_0048

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	CFLRD3	CRLRD3	0	CAPIF3	CAPCH3 EN	FL_IE3	RL_IE3	INV3
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	CFLRD2	CRLRD2	0	CAPIF2	CAPCH2 EN	FL_IE2	RL_IE2	INV2
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-13：PWM 捕获控制寄存器（PCCR1）

CFLRDx — Capture Falling Latch Register load flag

1 = 当输入通道 x 有一个下降的转换时，CFLRx 被更新，这个位是“1”。

0 = 当输入通道 x 没有一个下降的转换时。

写下 1 来清除这一位。

CRLRDx — Capture Rising Latch Register load flag

1 = 当输入通道 x 有一个上升的转换时，CRLRx 被更新，这个位是“1”。

0 = 当输入通道 x 没有一个上升的转换时。

写下 1 来清除这一位。

CAPIFx — Capture Channel x interrupt flag

1 = 当输入通道 x 有一个下降的转换，并且启用了 FL_IEx 位时，将设置这个中断标志。当输入通道 x 有一个上升的转换，并且启用了 RL_IEx 位时，也将设置这个中断标志。

0 = 未设置捕获通道 x 中断标志。

写下 1 来清除这一位。

CAPCHxEN — Capture Channel x Enable/Disable

1 = 启用

0 = 禁用

当此位设置为 1 时，捕获锁定 PMW 计数器，该值保存到 CRLR（上升锁存）和 CFLR（下降锁存）。
当此位设置为 0 时，捕获不会更新 CRLR 和 CFLR，并禁用通道 x 中断。

FL_IEx — Channel x Falling Interrupt Enable ON/OFF

1 = 启用

0 = 禁用

当这个位被设置为 1 时，如果捕获检测到通道 x 有一个下降的转换，捕获将发出一个中断。

RL_IEx — Channel x Rising Interrupt Enable ON/OFF

1 = 启用

0 = 禁用

当此位设置为 1 时，如果捕获检测到通道 x 有一个上升的转换，捕获将发出一个中断。

INVx — Channel x Inverter ON/OFF

1 = 反向器打开

0 = 反向器关闭

22.5.2.10. PWM 捕获上升锁存寄存器（PCRLRn）

这些寄存器用于在捕获上升转换时锁定 PWM 计数器。

地址：PWMn_BASEADDR + 0x0000_004C

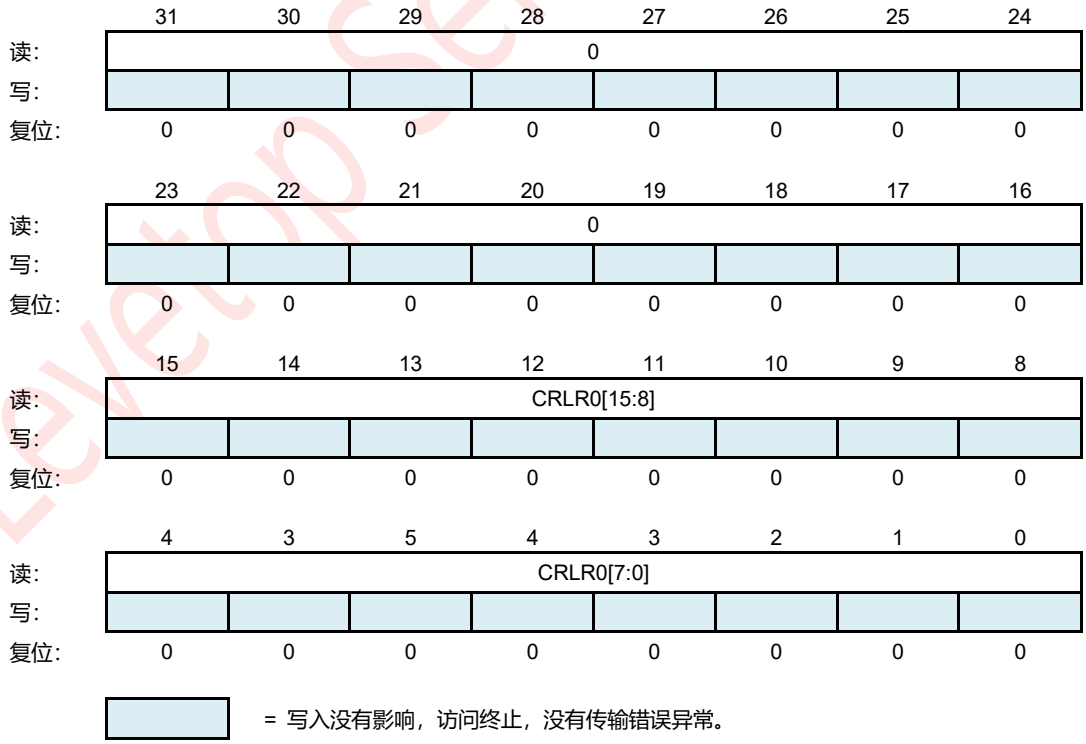


图 22-14：PWM 捕获上升锁存寄存器（PCRLR0）

地址：PWMn_BASEADDR + 0x0000_0054

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CRLR1[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	CRLR1[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-15：PWM 捕获上升锁存寄存器（PCRLR1）

地址：PWMn_BASEADDR + 0x0000_005C

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CRLR2[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	CRLR2[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-16：PWM 捕获上升锁存寄存器（PCRLR2）

地址：PWMn_BASEADDR + 0x0000_0064

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CRLR3[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	CRLR3[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-17：PWM 捕获上升锁存寄存器（PCRLR3）

CRLRx[15:0] — Capture Rising Latch Registerx

当通道 x 的转换阶段出现上升时，锁住 PWM 计数器。

22.5.2.11. PWM 捕获下降锁存寄存器（PCFLRn）

这些寄存器用于在捕获一个下降的转换时锁存 PWM 计数器。

地址：PWMn_BASEADDR + 0x0000_0050

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CFLR0[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	4	3	5	4	3	2	1	0
读:	CFLR0[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-18：PWM 捕获下降锁存寄存器（PCFLR0）

地址：PWMn_BASEADDR + 0x0000_0058

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CFLR1[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	CFLR1[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-19：PWM 捕获下降锁存寄存器（PCFLR13）

地址：PWMn_BASEADDR + 0x0000_0060

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CFLR2[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	CFLR2[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-20：PWM 捕获下降锁存寄存器（PCFLR2）

地址：PWMn_BASEADDR + 0x0000_0068

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	CFLR3[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	CFLR3[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 22-21：PWM 捕获下降锁存寄存器（PCFLR3）

CFLRx[15:0] — Capture Falling Latch Registerx

当通道 x 的转换过程下降时，锁住 PWM 计数器。

22.5.2.12. 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。

地址：PWMn_BASEADDR + 0x0000_006C

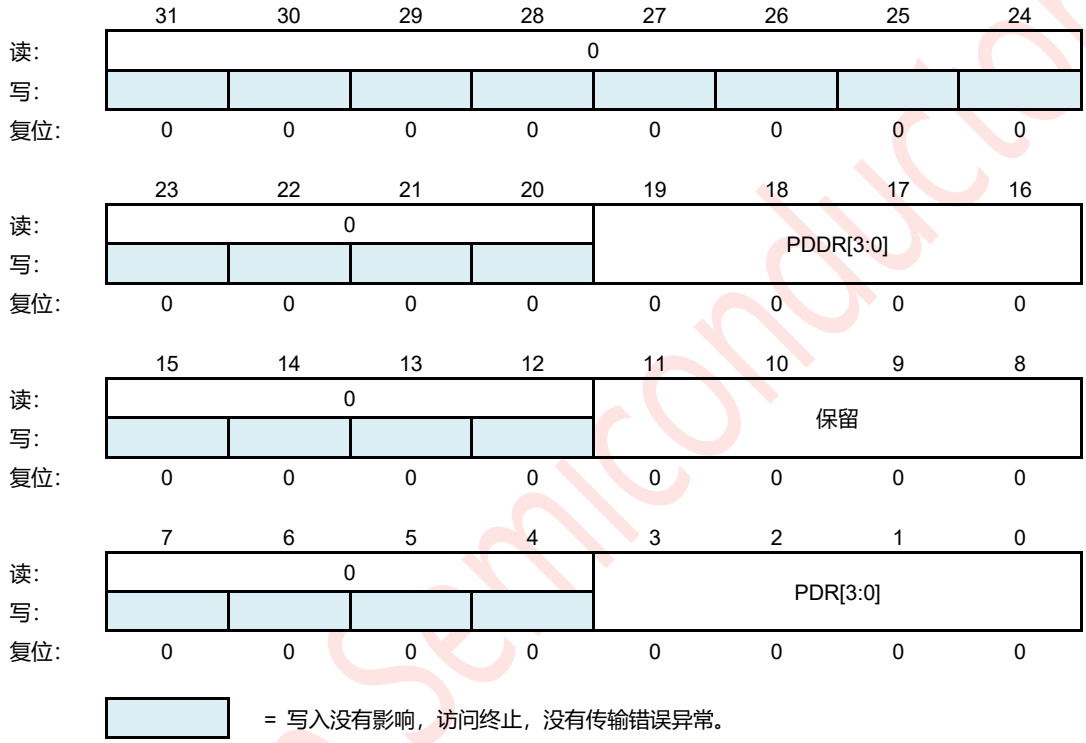


图 22-22：PWM 端口控制寄存器（PPCR）

PDDR[3:0] — Port Data Direction Register

PDDR[3:0]位控制 PWM 引脚的方向。复位清除 PDDR[3:0]。

1 = 对应的引脚配置为输出

0 = 对应的引脚配置为输入

PDR[3:0] — Port Data Register

将数据写入 PDR[3:0]可以驱动配置为通用输出的相应引脚。读取输入(PDDR 位清除）将返回引脚电平。

22.6. 功能描述

本章节介绍了 PWM 的功能操作。

22.6.1. PWM 双缓冲和自动重新加载

PWM-计时器具有双重缓冲功能，允许为下一个计时器操作更改的重新加载值，而不停止当前计时器操作。虽然设置了新的计时器值，但当前计时器操作仍然成功操作。计数器值可以写入 CNR0~3 中，当前的计数器值可以从 PTR0~3 中读取。当下计数器达到 0 时，自动重新加载操作将将数据从 CNR0~3 复制到下计数器。如果 CNR0~3 设置为零，则当计数器计数为零时，计数器将停止。如果自动重装位设置为零，计数器将立即停止。

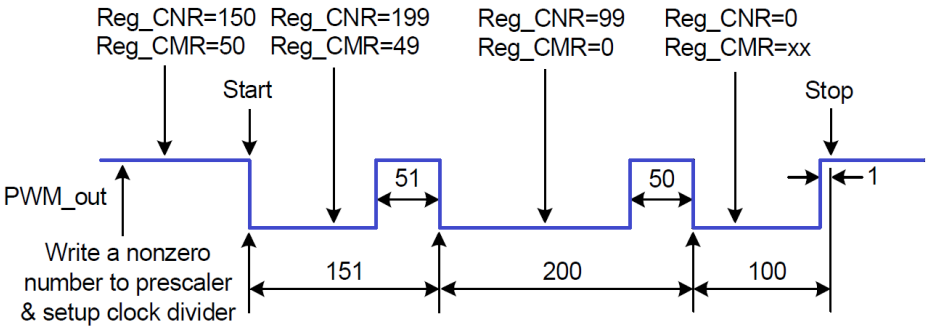


图 22-23：PWM 双缓冲功能说明

22.6.2. 调节占空比

双缓冲功能允许在当前周期中的任何时间点上写入 CMR。所加载的值将从下一个周期开始生效。

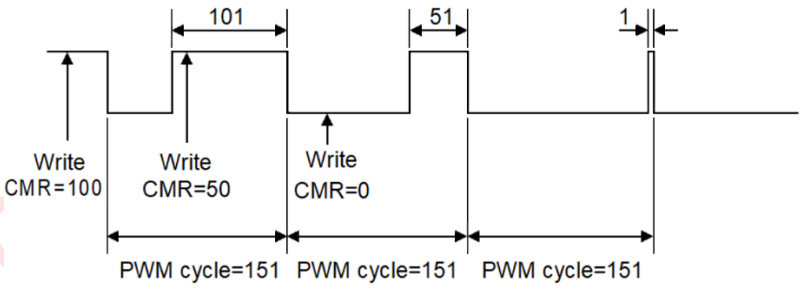


图 22-24：PWM 控制器输出负荷比

22.6.3. 死区发电机

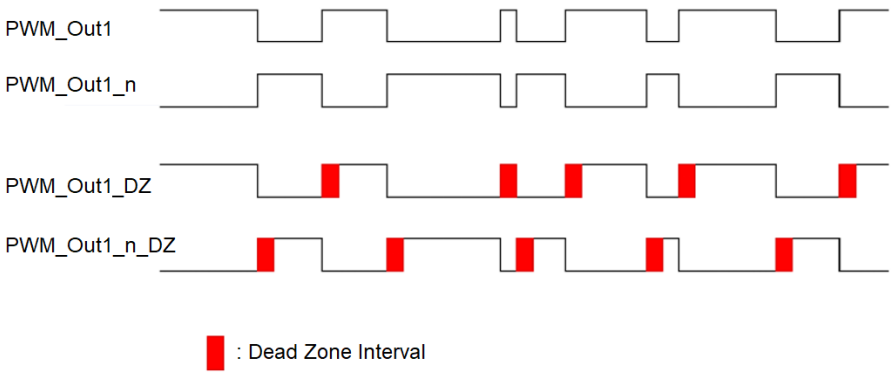


图 22-25：死区生成操作

22.6.4. PWM 计时器启动过程

1. 设置时钟选择器（CSR）
2. 设置比例前 r 和死区间隔（PPR）
3. 设置反向器开/关，死区发生器开/关，切换模式/一次性模式，和 PWM 定时器关闭。（PCR）
4. 设置比较器寄存器（CMR）
5. 设置计数器寄存器（CNR）
6. 设置中断启用寄存器（PIER）
7. 设置 PWMx 作为输出引脚（PPCR）
8. 启用 PWM 计时器（PCR）

22.6.5. PWM 计时器停止程序

方法 1:
将 16 位下计数器（CNR）设置为 0，并监控 PTR。当 PTR 达到 0 时，禁用 PWM 计时器（PCR）。（推荐使用）

方法 2:
将 16 位下计数器（CNR）设置为 0。当中断请求发生时，禁用 PWM 计时器（PCR）。（推荐使用）

方法 3:
直接禁用 PWM 定时器（PCR）。（不推荐使用）

22.6.6. 捕获启动过程

1. 设置时钟选择器（CSR）
2. 设置预比例尺（PPR）
3. 设置反向器开/关，死区发生器开/关，自动加载模式/一次性模式，和 PWM 定时器关闭。（PCR）
4. 设置计数器寄存器（CNR）
5. 设置捕获寄存器（CCR）
6. 设置 PWMx 作为输入引脚（PPCR）
7. 启用 PWM 计时器（PCR）

22.6.7. 捕获基本计时器操作

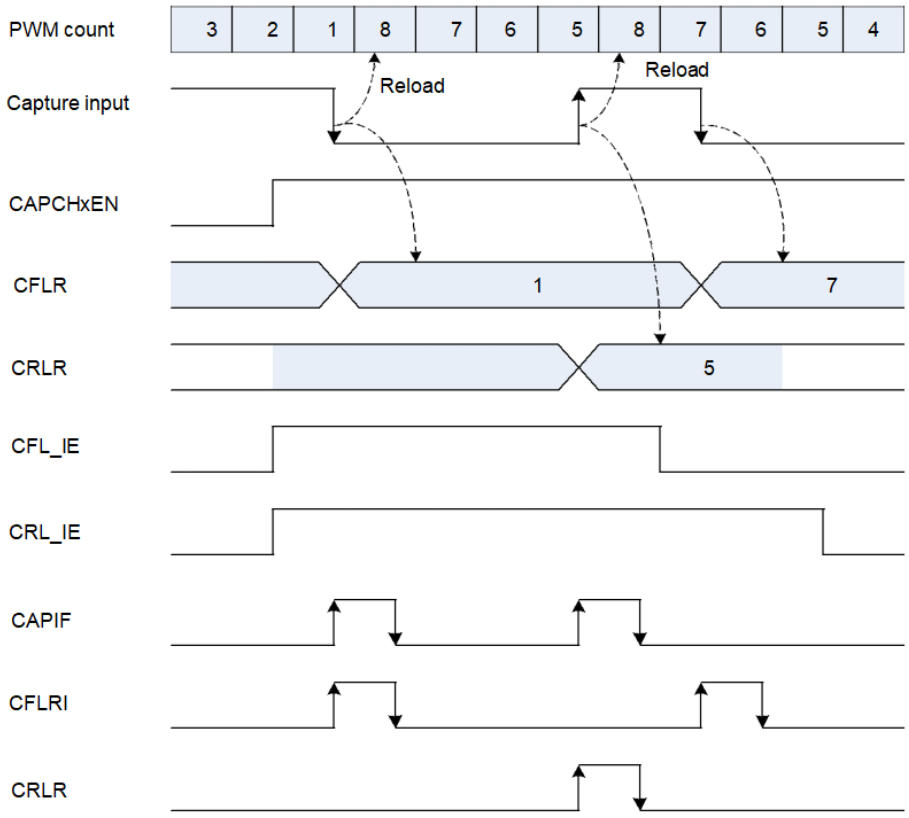


图 22-26：捕获基本计时器操作

在这种情况下，中国北车是 8 号：

1. 当 CAPIFx 设置为 1 时，PWM 计数器 CNRx 将被重新加载。
2. 通道低脉冲宽度为（CNR + 1-CRLR）。
3. 通道高脉冲宽度为（CNR + 1 - CFLR）。

23. 模拟比较器（COMP）

23.1. 功能介绍

模拟比较器提供可编程的响应时间，迟滞，一个模拟输入多路复用器，和两个输出，在端口引脚可选：一个同步“过滤”输出（CP），或一个异步“原始”输出（CPA）。即使在系统时钟处于非活动状态，异步 CPA 信号也可用。这允许比较器在停止模式下操作并生成输出。

23.2. 方块图

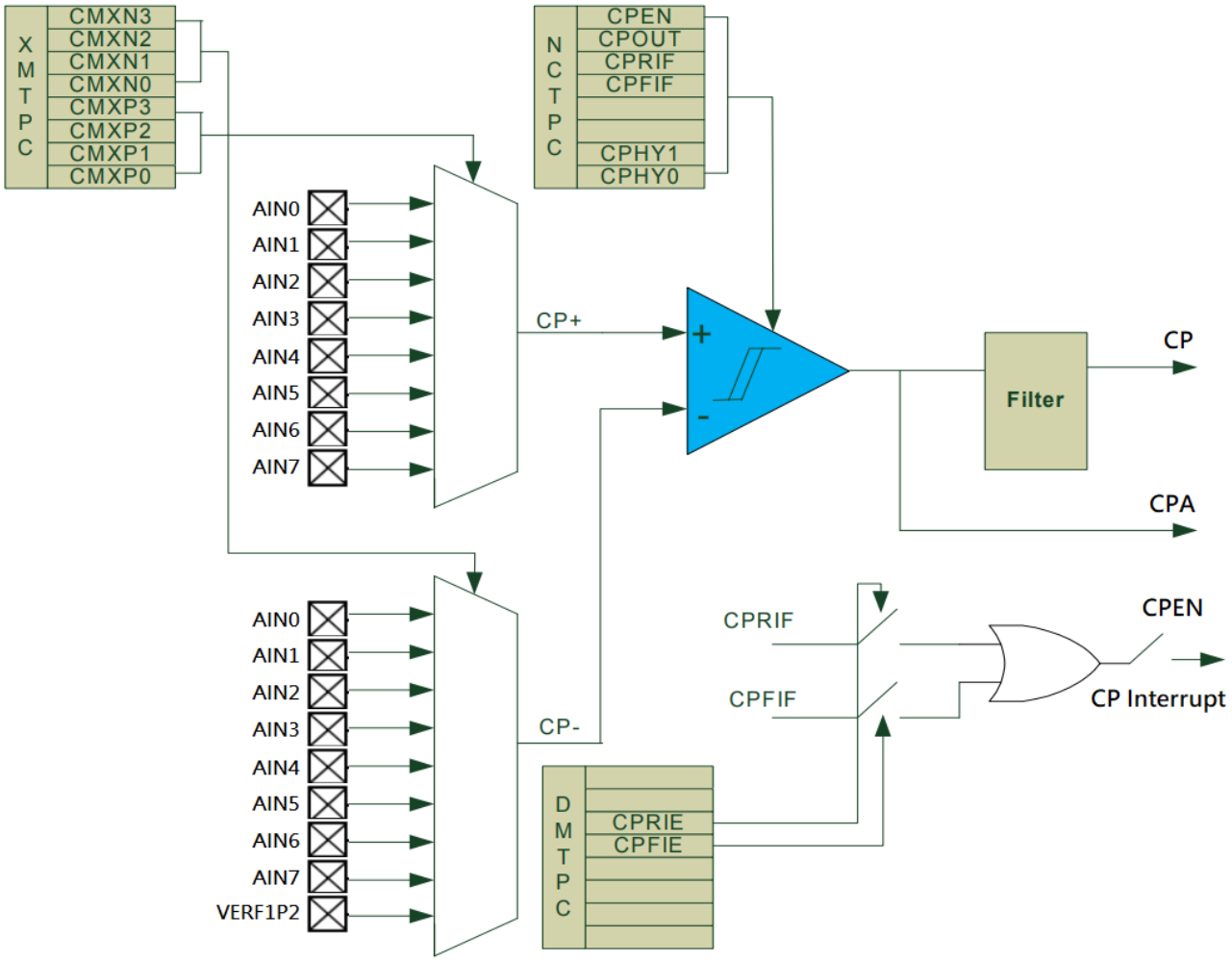


图 23-1：比较器方块图

23.3. 操作模式

本章节介绍了这三种低功耗模式。

23.3.1. 等待模式

在等待模式下，比较器模块可以通过设置 CPEN 位来继续正常运行，并且可以通过配置为通过产生中断请求来退出低功耗模式。

23.3.2. Doze 模式

在起启动模式下，比较器模块可以通过设置 CPEN 位来继续正常运行，并且可以被配置为通过产生中断请求来退出低功耗模式。

23.3.3. 停止模式

在停止模式下，系统时钟不存在，比较器模块可以通过设置 CPEN 位来继续正常运行，并且可以通过产生一个异步的“原始”输出（CPA）唤醒信号来配置为退出低功耗模式。

23.4. 内存映射和寄存器

比较器模块记忆映射如下表所示.COMP0 的基本地址为 0x400A_0000，COMP1 为 0x400B_0000。本章节介绍了比较器的内存映射和寄存器结构。

23.4.1. 内存映射

见下表以获取内存映射的描述。
该设备设有两个比较器模块。

表 23-1：比较器模块内存贴图

偏移地址	Bits[7:0]	访问权限(1)
0x3	Comparator Control Register(CPTCN)	S/U
0x2	Comparator0 Mode Selection Register(CPTMD)	S/U
0x1	Comparator MUX Selection Register(CPTMX)	S/U
0x0	Comparator Output Filter Selection Register(CPTFS)	S/U

提示（1）： 仅限 S = CPU 监控模式访问。S/U = CPU 监控器或用户模式访问。在用户模式下只访问管理员地址没有影响，并导致周期终止传输错误。

23.4.2. 寄存器描述

比较器编程模型由以下寄存器组成：

•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：比较器输出过滤器选择寄存器

23.4.2.1. 比较器控制寄存器（CPTCN）

地址：COMP0_BASEADDR + 0x0000_0003，COMP1_BASEADDR + 0x0000_0003

	7	6	5	4	3	2	1	0
读:	CPEN	CPOUT	CPRIF	CPFIF	0	0	CPHY[1:0]	
写:								
复位:	0	0	0	0	0	0	0	1

= 写入没有影响，访问终止，没有传输错误异常。

图 23-2：比较器控制寄存器（CPTCN）

CPEN — Comparator Enable Bit

读/写 CPEN 位启用比较器操作。当禁用比较器时，CPOUT 处于低状态。

1 = 已启用比较器

0 = 已禁用比较器

CPOUT — Comparator Output State Flag.

1 = 电压在 CP+ > CP-

0 = 电压在 CP+ < CP-。

CPRIF — Comparator Rising-Edge Flag. Must be cleared by writing one to this bit.

1 = 比较器上升边缘产生。

0 = 自上次清除此标志以来，未发生比较器上升边缘。

CPFIF — Comparator Falling-Edge Flag. Must be cleared by writing one to this bit.

1 = 比较器下降边缘产生。

0 = 自上次清除此标志以来，未发生比较器掉边。

CPHY[1:0] — Comparator Hysteresis Control Bits.

00 = 滞后被禁用。

01 = 迟滞 = 8 mV。

10 = 迟滞性 = 12 mV。

11 = 迟滞 = 15.4 mV。

23.4.2.2. 比较器模式选择寄存器（CPTMD）

地址：COMP0_BASEADDR + 0x0000_0002，COMP1_BASEADDR + 0x0000_0002

	7	6	5	4	3	2	1	0
读:	CPOEN	0	CPRIE	CPFIE	0	WKUPLEVEL	0	CPMD
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 23-3：比较器模式选择寄存器（CPTMD）

CPOEN — Comparator Output to pad control bit.

1 = 比较器输出可以在 PWM1[0]的 COMP0 中观察到，在 COMP1 的 PWM1[1]中观察到。

0 = 比较器输出禁用。

CPRIE — Comparator Rising-Edge Interrupt Enable.

1 = 比较器上升缘中断允许。

0 = 比较器上升缘中断禁用。

CPFIE — Comparator Falling-Edge Interrupt Enable.

1 = 比较器下降缘中断允许。

0 = 比较器下降缘中断禁用。

WKUPLEVEL — Comparator WakeUp level control bit.

1 = CP+ > CP– 上的电压将在停止模式期间产生唤醒请求。

0 = CP+ < CP– 上的电压将在停止模式期间产生唤醒请求。

CPMD — Comparator Mode Select. These bits select the response time for Comparator.

表 23-2：比较器模式选择

模式	CPMD 响应时间的功耗		
低速	0	500ns	3.3uA
高速	1	80ns	22uA

23.4.2.3. 比较器 MUX 选择寄存器（CPTMX）

地址：COMP0_BASEADDR + 0x0000_0001，COMP1_BASEADDR + 0x0000_0001

	7	6	5	4	3	2	1	0
读: 写:	CMXN[3:0]				CMXP[3:0]			
复位:	1	1	1	1	1	1	1	1

图 23-4：比较器 MUX 选择寄存器（CPTMX）

CMXN[3:0] — Comparator Negative Input MUX Select. These bits select which Port pin is used as the Comparator negative input.

表 23-3：比较器负向输入 MUX 选择

CMXN3	CMXN2	CMXN1	CMXN0	负输入
0	0	0	0	AIN0
0	0	0	1	AIN 1
0	0	1	0	AIN 2
0	0	1	1	AIN 3
0	1	0	0	AIN 4
0	1	0	1	AIN 5
0	1	1	0	AIN 6
0	1	1	1	AIN 7
1	0	0	0	Vref1p2
其他值				None

CMXP[3:0] — Comparator Positive Input MUX Select. These bits select which Port pin is used as the Comparator positive input.

表 23-4：比较器正输入 MUX 选择

CMXP3	CMXP2	CMXP1	CMXP0	正输入
0	0	0	0	AIN0
0	0	0	1	AIN 1
0	0	1	0	AIN 2
0	0	1	1	AIN 3
0	1	0	0	AIN 4
0	1	0	1	AIN 5
0	1	1	0	AIN 6
0	1	1	1	AIN 7
1	x	x	x	None

23.4.2.4. 比较器输出过滤器选择寄存器（CPTFLS）

地址：COMP0_BASEADDR + 0x0000_0000，COMP1_BASEADDR + 0x0000_0000

	7	6	5	4	3	2	1	0
读: 写:	CMFLEN	CMFL_WIDTH[6:0]						
复位:	0	0	0	0	0	0	0	0

图 23-5：比较器输出过滤器选择寄存器（CPTFLS）

CMFLEN — Comparator Output Digital Filter Enable

1 = 比较器输出数字滤波器，滤波器输出产生 CPMRIF 和 CPMFIF；

0 = 比较器输出数字滤波器，CPMRIF 和 CPMFIF 由原始输出生成；

CMFL_WIDTH[6:0] — Comparator Output Digital Filter Pulse Width Selection.

CMFL_WIDTH[6:0]确定将被过滤的输入脉冲的宽度。如果输入脉冲宽度小于（CMFL_WIDTH[6:0] + 2）*固定周期，则脉冲将被过滤。

23.5. 功能描述

在 CPTMX 寄存器中选择比较器输入。CMXP3-CMXP0 位选择比较器正输入；CMXN3-CMXN0 位选择比较器负输入。关于比较器输入的重要提示：选择为比较器输入的端口引脚应在其相关的端口配置寄存器中配置为模拟输入，并且应禁用输入、输出、上拉和下拉控制。

比较器的输出可以在软件中进行轮询，用作中断源，和/或路由到端口引脚。当路由到端口引脚时，比较器输出可用，并且与系统时钟具有异步性或同步性；异步输出即使在停止模式（无活动系统时钟）下也可用。禁用时，比较器输出默认为逻辑低状态，其电源电流下降小于 100nA。

比较器滞后是软件可编程通过其比较器控制寄存器 CPTCN。用户可以编程的迟滞电压的值。比较器迟滞现象使用比较器控制寄存器 CPTCN 中的位 1-0 进行编程。

比较器中断可以在上升边缘和下降边缘的输出转换上产生。在比较器下降边出现时，CPFIF 标志被设置为逻辑 1，而在比较器上升边出现时，CPRIF 标志被设置为逻辑 1。一旦设置，这些位将保持设置，直到被软件清除。通过将 CPRIE 设置为逻辑 1 来启用比较器上升边缘中断掩码。通过将 CPFIE 设置为逻辑 1 来启用比较器下降边缘中断掩码。

通过读取 CPOUT 位，可以随时获得比较器的输出状态。比较器通过设置 CPEN 位为逻辑 1 而被启用，并通过清除该位为逻辑 0 而被禁用。

请注意，当比较器第一次通电时，或如果对磁滞控制位进行了更改时，可以检测到错误的上升边和下降边。因此，建议在比较器启用或改变其模式位后的短时间内，将上升边和下降边标志明确清除为逻辑 0。比较器的最小启动时间小于 5 我们。

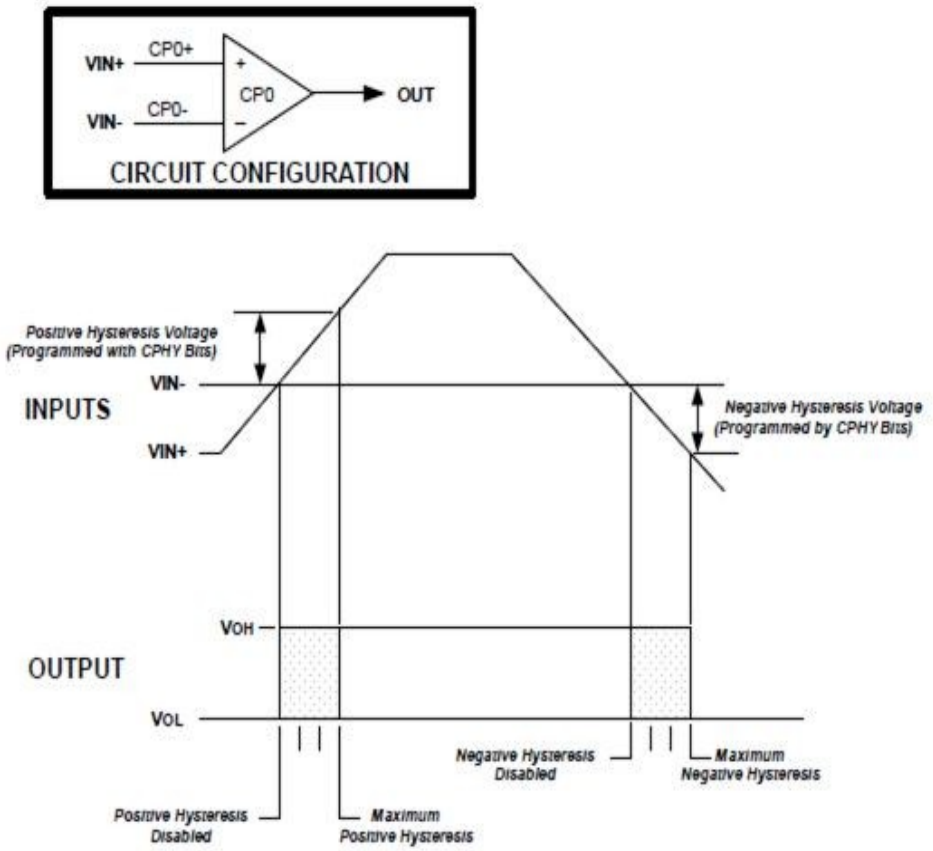


图 23-6：比较器滞后图

24. USB2.0 控制器（USBC）

24.1. 功能介绍

本章节介绍 USB2.0 全速控制器。本模块中的设备实现提供了实现 USB 2.0 全速/低速兼容外围设备的解决方案。

24.2. USB 模块特点

USB 模块的功能包括：

- USB2.0 仅设备功能控制器
- USB 2.0 兼容
  - 12Mbps 全速（FS）数据速率
  - USB 数据控制逻辑：
    分组识别和解码/生成
    CRC 生成和检查
    NRZI（非返回零倒置）编码/解码
    小填料
    同步检测
    数据包端检测
- 8 个 USB 端点
- USB 内存
  - 2048 字节的缓冲区 RAM，由系统和 USB 模块共享
  - RAM 可以被分配为 USB 控制器或额外的系统 RAM 资源的缓冲区
- USB 复位选项
  - 由 MCU 生成的 USB 模块复位
  - 由主机生成的总线复位，从而触发 CPU 中断
- 使用远程唤醒支持来暂停和恢复操作
- 收发器功能
  - 将 USB 差分电压转换为数字逻辑信号水平
  - 芯片上的 USB 拔拉电阻器
- 3.3V 调节器

24.3. 方块图

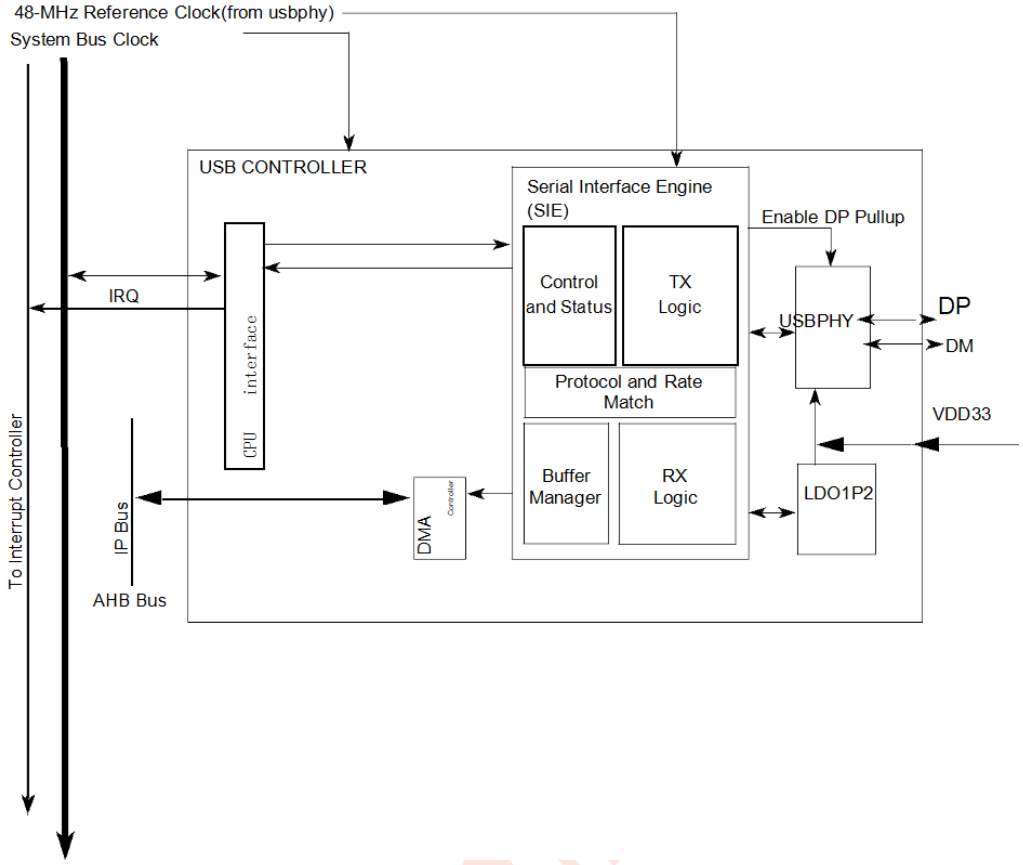


图 24-1：USB 全速设备(USB 模块）方块图

24.4. 操作模式

本章节介绍了这三种低功耗模式。

24.4.1. 等待模式

在等待模式下，USB 模块可以继续正常运行，并可以通过产生一个中断请求来配置为退出低功耗模式。

24.4.2. Doze 模式

在 Doze 模式下，USB 模块可以继续正常运行，并且可以通过产生一个中断请求来配置为退出低功耗模式。

24.4.3. 停止模式

USB 模块可选择在停止模式下使用。根据 USB 规范版本，USB 暂停模式可能需要减少当前消耗模式。2.0，停止模式可以降低 MCU 的电流消耗，从而降低整个 USB 设备。在通过固件进入停止之前，用户必须确保设备设置配置为停止，以实现 USB 暂停当前的消耗目标。

当设置休眠标志时，通知 USB 模块进入暂停模式；这发生在 USB 总线空闲 3 ms 之后。USB 设备等待处理模式当前消耗级别要求由 USB 规范版本定义。2.0（低功耗 500uA，大功率 2.5 mA）。
如果设置了用户，并且在 USB 总线上检测到 K 状态（恢复信令），则恢复位将被设置。这将触发一个异步中断，将唤醒 MCU 从停止模式，并启用时钟到 USB 模块。

24.5. 内存映射和寄存器

USB 模块内存映射如下表所示 USB 控制器的基本地址为 0x4016_0000。本章节介绍了 USB 模块的内存映射和寄存器结构。

24.5.1. 内存映射

见下表以获取内存映射的描述。

表 24-1：USB 模块内存地图

偏移地址	Bits[31:0]	访问权限(1)
0x0000	保留	S/U
0x0004	保留	S/U
0x0008	保留	S/U
0x000C	保留	S/U
0x001C	USBPHY Control Register 1 (USBPHY_CTRL1)	S/U
0x0080	Interrupt Status Register (INT_STAT)	S/U
0x0084	Interrupt Enable Register (INT_ENB)	S/U
0x0088	Error Interrupt Status Register (ERR_STAT)	S/U
0x008C	Error Interrupt Enable Register (ERR_ENB)	S/U
0x0090	Status Register (STAT)	S/U
0x0094	Control Register (CTL)	S/U
0x0098	Address Register (ADDR)	S/U
0x009C	EBT Page Register 1 (EBT_PAGE_01)	S/U
0x00A0	Frame Number Register (FRMNUML)	S/U
0x00A4	Frame Number Register (FRMNUMH)	S/U
0x00A8	Token Register (TOKEN)	S/U
0x00AC	SOF Threshold Register (SOF_THLD)	S/U
0x00B0	EBT Page Register 2 (EBT_PAGE_02)	S/U
0x00B4	EBT Page Register 3 (EBT_PAGE_03)	S/U
0x00C0	Endpoint Control Registers (ENDPT0)	S/U
0x00C4	Endpoint Control Registers (ENDPT1)	S/U
0x00C8	Endpoint Control Registers (ENDPT2)	S/U
0x00CC	Endpoint Control Registers (ENDPT3)	S/U
0x00D0	Endpoint Control Registers (ENDPT4)	S/U
0x00D4	Endpoint Control Registers (ENDPT5)	S/U
0x00D8	Endpoint Control Registers (ENDPT6)	S/U
0x00DC	Endpoint Control Registers (ENDPT7)	S/U
0x0100	USBPHY Control Register 2 (USBPHY_CTRL2)	S/U
0x0104	USB PHY Observe Register	S/U

偏移地址	Bits[31:0]	访问权限(1)
	(USB_PHY_OBSERVE)	
0x0108	USB PHY GPIO Register	S/U
0x010C	USB Resume Wakeup Enable Register	S/U
	(USB_RESMEN)	
0x0118	USBPHY Control Register 3 (USBPHY_CTRL3)	S/U
0x011C	USBPHY Control Register 4 (USBPHY_CTRL4)	S/U

注意(1)：仅限 S = CPU 监控模式访问。S/U = CPU 监控器或用户模式访问。在用户模式下只访问管理员地址没有影响，并导致周期终止传输错误。

24.5.2. 寄存器描述

24.5.2.1. USBPHY 控制寄存器 1（USBPHY_CTRL1）

USBPHY 控制寄存器控制数据线终端电阻的操作。

地址：USBC_BASEADDR + 0x0000_001C

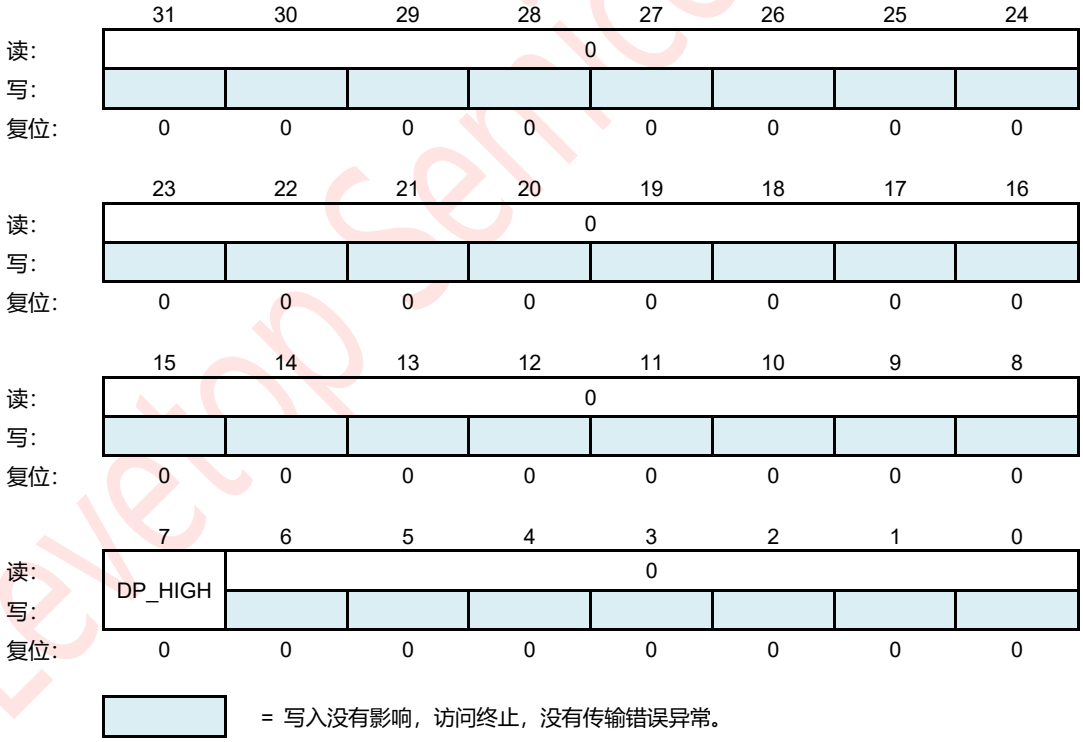


图 24-2：USBPHY 控制寄存器 1（USBPHY_CTRL1）

DP_HIGH — D+ Data Line pullup resistor enable

0 = D+上拉电阻被禁用

1 = 启用 D+上拉电阻。

24.5.2.2. 中断状态寄存器（INT_STAT）

中断状态寄存器包含了 USB 模块中的每个中断源的位。这些位都用各自的中断启用位限定。这个寄存器的所有位在逻辑上与 OTG 中断状态寄存器（OTG_STAT）一起使用，为处理器的中断控制器形成一个单一的中断源。在设置了中断位之后，只能通过将中断位写入相应的中断位来清除。此寄存器包含复位后的值 0x00。

地址：USBC_BASEADDR + 0x0000_0080

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	STALL	0	RESUME	SLEEP	TOKDNE	SOF_TOK	ERROR	USB_RST
写:								
复位:	0	0	0	1	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 24-3：中断状态寄存器（INT_STAT）

STALL — Stall Interrupt, and cleared by writing 1 to this bit.

在设备模式下，当 USB 模块发送停止握手时，此位将起作用。在主机模式下，当 USB 模块在 USB 事务的握手阶段检测到停止确认时，将设置此位。此中断可用于确定最后一个 USB 事务是否已成功完成或是否停止。

RESUME — Resume Interrupt, and cleared by writing 1 to this bit.

此位根据 DP/DM 信号设置，可用于信号在 USB 总线上的远程唤醒信号。当未处于等待处理模式时，应禁用此中断。

SLEEP — Sleep Interrupt, and cleared by writing 1 to this bit.

当 USB 模块检测到 USB 总线上持续空闲 3 毫秒时，将设置此位。睡眠计时器通过 USB 总线上的活动进行复位。

TOKDNE — Token Done Interrupt, and cleared by writing 1 to this bit.

在正在处理的当前 TOKEN 完成时设置此位。处理器应立即读取 STAT 寄存器，以确定用于此 TOKEN 的端点和 EB 条目。清除此位（通过写入一个位）会导致清除 STAT 寄存器，或将 STAT 保持寄存器加载到 STAT

寄存器中。

SOF_TOK — SOF Token Interrupt, and cleared by writing 1 to this bit.

当 USB 模块接收到帧的开始（SOF）TOKEN 时，将设置此位。在主机模式下，当达到 SOF 阈值时设置这个位，以便软件可以为下一个 SOF 做准备。

ERROR — Error Interrupt, and cleared by writing 1 to this bit.

当 ERR_STAT 寄存器中的任何错误条件发生时，都将设置此位。然后，处理器必须读取 ERR_STAT 寄存器，以确定错误的来源。

USB_RST — USB RST Interrupt, and cleared by writing 1 to this bit.

当 USB 模块解码有效 USB 复位时设置此位。这将通知微处理器，它应该将 0x00 写入地址寄存器，并启用端点 0。在检测到 USB 复位 2.5 微秒后，将设置 USB_RST。在删除 USB 复位条件并重新插入之前，不会再次起作用。

24.5.2.3. 中断启用寄存器（INT_ENB）

中断启用寄存器包含了 USB 模块中的每个中断源的启用位。设置这些位中的任何一个都可以启用 INT_STAT 寄存器中各自的中断源。此寄存器包含复位后的值 0x00。

地址：USBC_BASEADDR + 0x0000_0084

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	STALL_EN	ATTACH_EN	RESUME_EN	SLEEP_EN	TOKDNE_EN	SOF_TOK_EN	ERROR_EN	USB_RST_EN
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 24-4：中断启用寄存器（INT_ENB）

STALL_EN — Stall Interrupt Enable

0 = Stall 中断被禁用

1 = Stall 中断已启用。

ATTACH_EN — Attach Interrupt Enable.

0 = Attach 中断被禁用。

1 = Attach 中断已启用。

RESUME_EN — Resume Interrupt Enable.

0 = Resume 中断被禁用。

1 = Resume 已启用。

SLEEP_EN — Sleep Interrupt Enable.

0 = 休眠中断被禁用。

1 = 休眠中断已启用。

TOKDNE_EN — Token Done Interrupt Enable.

0 = Token Done 中断被禁用。

1 = Token Done 中断已启用。

SOF_TOK_EN — SOF Toke Interrupt Enable.

0 = SOF Toke 中断被禁用。

1 = SOF Toke 中断已启用。

ERROR_EN — Error Interrupt Enable.

0 = 错误中断被禁用。

1 = 错误中断已启用。

USB_RST_EN — USB Reset Interrupt Enable.

0 = USB 复位中断被禁用。

1 = USB 复位中断已启用。

24.5.2.4. 错误中断状态寄存器（ERR_STAT）

错误中断状态寄存器包含对 USB 模块中的每个错误源的启用位。这些位都用它们各自的错误启用位进行限定。这个寄存器的所有位在逻辑上在一起，结果放置在 INT_STAT 寄存器的错误位中。在设置了中断位之后，只能通过将中断位写入相应的中断位来清除。一旦检测到错误条件，就会设置每个位。因此，该中断通常并不对应于正在处理的 TOKEN 的结束。此寄存器包含复位后的值 0x00。

地址：USBC_BASEADDR + 0x0000_0088

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	BTS_ERR	0	DMA_ERR	BTO_ERR	DFN8	CRC16	CRC5_EOF	PID_ERR
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 24-5：错误中断状态寄存器（ERR_STAT）

BTS_ERR — This bit is set when a bit stuff error is detected. If set, the corresponding packet is rejected due to the error.

DMA_ERR — 如果 USB 模块请求 DMA 访问以读取新的 EBT，但在需要接收或传输数据之前没有得到总线，则设置此位。如果处理 TX 传输，这将导致传输数据下溢状态。如果正在处理 RX 传输，这将导致接收数据溢出的情况。当为微处理器和 USB 模块开发设备仲裁硬件以最小化总线请求和总线授予延迟时，此中断非常有用。如果到主机或来自主机的数据包大于 EBT 中分配的缓冲区大小，也会设置此位。在这种情况下，数据包在放入缓冲区内存中时被截断。

BTO_ERR — 在发生总线周转超时错误时设置此位。USB 模块包含一个总线周转计时器，它跟踪 TOKEN 与设置或输出 TOKEN 的数据阶段或 IN TOKEN 的数据和握手阶段之间使用的时间量。如果在从 IDLE 转换之前从前一个 EOP 计算了超过 16 位的时间，则会发生总线周转超时错误。

DFN8 — 如果接收到的数据字段的长度不是 8 位，则设置此位。USB 规范 1.0 要求数据字段是一个整数字节数。如果数据字段不是整数字节数，则设置此位。

CRC16 — 当数据包由于 CRC16 错误而被拒绝时，设置此位。

CRC5_EOF — 这个错误中断有两个功能。当 USB 模块在设备模式（HOST_MODE_EN = 0）下运行时，此中断会检测到主机生成的 TOKEN 数据包中的 CRC5 错误。如果设置，则由于 CRC5 错误，TOKEN 包被拒绝。当 USB 模块在主机模式（HOST_MODE_EN = 1）下运行时，此中断会检测到帧结束（EOF）错误情况。当 USB 模块正在传输或接收数据，而 SOF 计数器为零时，就会发生这种情况。在开发 USB 数据包调度软件时，这个中断很有用，以确保没有 USB 事务跨越下一帧的开始。

PID_ERR — 当 PID 检查字段失败时，将设置此位。

24.5.2.5. 错误中断启用寄存器（ERR_ENB）

错误中断启用寄存器包含对 USB 模块中的每个错误中断源的启用位。设置这些位中的任何一个都可以启用 ERR_STAT 寄存器中各自的中断源。一旦检测到错误条件，就会设置每个位。因此，该中断通常并不对应于正在处理的 TOKEN 的结束。此寄存器包含复位后的值 0x00。

地址：USBC_BASEADDR + 0x0000_008C

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	BTS_ERR_EN	0	DMA_ERR_EN	BTO_ERR_EN	DFN8_EN	CRC16_EN	CRC5_EOF_EN	PID_ERR_EN
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 24-6：错误中断启用寄存器（ERR_ENB）

BTS_ERR_EN — BTS_ERR Interrupt Enable

0 = BTS_ERR 中断被禁用。

1 = BTS_ERR 中断已启用。

DMA_ERR_EN — DMA_ERR Interrupt Enable.

0 = DMA_ERR 中断被禁用。

1 = DMA_ERR 中断已启用。

BTO_ERR_EN — BTO_ERR Interrupt Enable.

0 = BTO_ERR 中断被禁用。

1 = BTO_ERR 中断已启用。

DFN8_EN — DFN8 Interrupt Enable.

0 = DFN8 中断被禁用。

1 = DFN8 中断已启用。

CRC16_EN — CRC16 Interrupt Enable.

0 = CRC16 中断被禁用。

1 = CRC16 中断已启用。

CRC5_EOF_EN — CRC5_EOF Interrupt Enable.

0 = CRC5_EOF 中断被禁用。

1 = CRC5_EOF 中断已启用。

PID_ERR_EN — PID_ERR Interrupt Enable.

0 = PID_ERR 中断被禁用。

1 = PID_ERR 中断已启用。

24.5.2.6. 状态寄存器（STAT）

状态寄存器报告 USB 模块内的事务状态。当处理器的中断控制器接收到 TOK_DNE 中断时，应读取状态寄存器以确定先前端点通信的状态。当 OK_DNE 中断位起作用时，状态寄存器中的数据将有效。STAT 寄存器实际上是由 USB 模块维护的状态 FIFO 中的一个读取窗口。当 USB 模块使用 EB 条目时，它会更新状态寄存器。如果在 TOK_DNE 中断之前执行了另一个 USB 事务，USB 模块将在 STATF 中存储下一个事务的 FIFO 的状态。因此，STAT 寄存器实际上是一个四字节的 FIFO，它允许处理器核心在处理 SIE 处理下一个事务时处理一个事务。清除 INT_STAT 寄存器中的 TOK_DNE 位会导致 SIE 用下一个 STAT 值的内容更新 STAT 寄存器。如果 STAT 保持寄存器中的数据有效，则 SIE 立即重新插入 TOK_DNE 中断。

地址：USBC_BASEADDR + 0x0000_0090

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	ENDP[3:0]				TX	ODD	0	0
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 24-7：状态寄存器（STAT）

ENDP[3:0] — Endpoint Number.

这四个位对接收或传输前一个 TOKEN 的端点地址进行编码。这允许微控制器确定最后一个 EBT 事务更新了哪个 EBT 条目。

0000 : Endpoint 0
0001 : Endpoint 1
0010 : Endpoint 2
0011 : Endpoint 3
0100 : Endpoint 4
0101 : Endpoint 5
0110 : Endpoint 6
0111 : Endpoint 7

TX — Transmit Indicator.

0 = 最近的数据转换是一个接收操作。

1 = 最近的数据转换是一个传输操作。

ODD — Odd/Even Transaction.

如果更新的最后一个端点缓冲区表在 EBT 的奇数组中，则设置此位。

24.5.2.7. 控制寄存器（CTL）

控制寄存器为 USB 模块提供各种控制和配置信息。

地址：USBC_BASEADDR + 0x0000_0094

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	JSTATE	SE0	TXSUSPEND/TOKENBUSY	RESET	HOST_MODE_EN	RESUME	ODD_RST	USB_EN/SOF_EN
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 24-8：控制寄存器（CTL）

JSTATE — Live USB differential receiver JSTATE signal

该信号的极性受到 LS_EN 的当前状态的影响。

SE0 — Live USB Single Ended Zero signal

该信号的极性受到 LS_EN 的当前状态的影响。

TXSUSPEND / TOKENBUSY

当 USB 模块处于主机模式时，当 USB 模块忙于执行 USB TOKEN 时，将设置 TOKEN_BUSY，并且不再将 TOKEN 命令写入 TOKEN 寄存器。软件应该在将任何 TOKEN 写入 TOKEN 寄存器之前检查此位，以确保 TOKEN 命令不会丢失。在设备模式下，当 SIE 禁用了数据包传输和接收时，将设置 tx 挂起。清除此位将允许 SIE 继续进行 TOKEN 处理。此位在接收到安装 TOKEN 时由 SIE 设置，允许软件在恢复 TOKEN 处理之前删除 EBT 中任何挂起的数据包事务。

RESET — This bit is invalid for current device-role-only function.

设置此位可使 USB 模块能够生成 USB 复位信号。这允许 USB 模块复位 USB 外设。此控制信号仅在主机模式下有效（HOST_MODE_EN = 1）。软件必须在所需的时间内设置为 1，然后将其清除为 0 以结束复位信号。

HOST_MODE_EN — This bit is read-only for this device-role-only function.

当设置为 1 时，此位使 USB 模块能够在主机模式下运行。在主机模式下，USB 模块在主机处理器的编程控制下执行 USB 事务。

RESUME

当设置为 1 时，此位使 USB 模块能够执行恢复信令。

这允许 USB 模块执行远程唤醒。软件必须将所需时间设置为 1，然后将其清除为 0。如果设置了 HOST_MODE_EN 位，当恢复位被清除时，USB 模块附加到恢复信令。

ODD_RST

将此位设置为 1，将将所有 EBT ODD 乒乓球/乒乓球位复位为 0，然后指定 EVEN EBT 库。

USB_EN/SOF_EN — USB Enable.

设置此位会导致 SIE 将其所有的 ODD 位复位为 ebt。因此，设置这个位会复位 SIE 中的大部分逻辑。当启用主机模式时，清除此位将导致 SIE 停止发送 SOF TOKEN。

0 = USB 模块已禁用。

1 = USB 模块已启用。

24.5.2.8. 地址寄存器（ADDR）

地址寄存器保存 USB 模块在外设模式下解码的唯一 USB 地址（HOST_MODE_EN = 0）。当在主机模式（HOST_MODE_EN = 1）下操作时，USB 模块使用 TOKEN 数据包传输此地址。这使得 USB 模块能够唯一地处理一个 USB 外设。在任何一种模式下，都必须设置控制寄存器内的 USB_EN 位。在复置输入激活或 USB 模块解码 USB 复置信号后，地址寄存器被复位为 0x00。此操作会初始化地址寄存器，以根据 USB 规范的要求解码地址 0x00。

地址：USBC_BASEADDR + 0x0000_0098

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	LS_EN	ADDR[6:0]						
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 24-9：地址寄存器（ADDR）

LS_EN — Low Speed Enable bit.

此位通知 USB 模块，写入 TOKEN 寄存器的下一个 TOKEN 命令必须以低速执行。这使得 USB 模块能够执行低速数据传输所需的必要的前导码。

ADDR[6:0] — USB address.

这个 7 位值定义了 USB 模块在外设模式下解码或在主机模式时传输的 USB 地址。

24.5.2.9. EBT 页面寄存器 1（EBT_PAGE_01）

地址：USBC_BASEADDR + 0x0000_009C

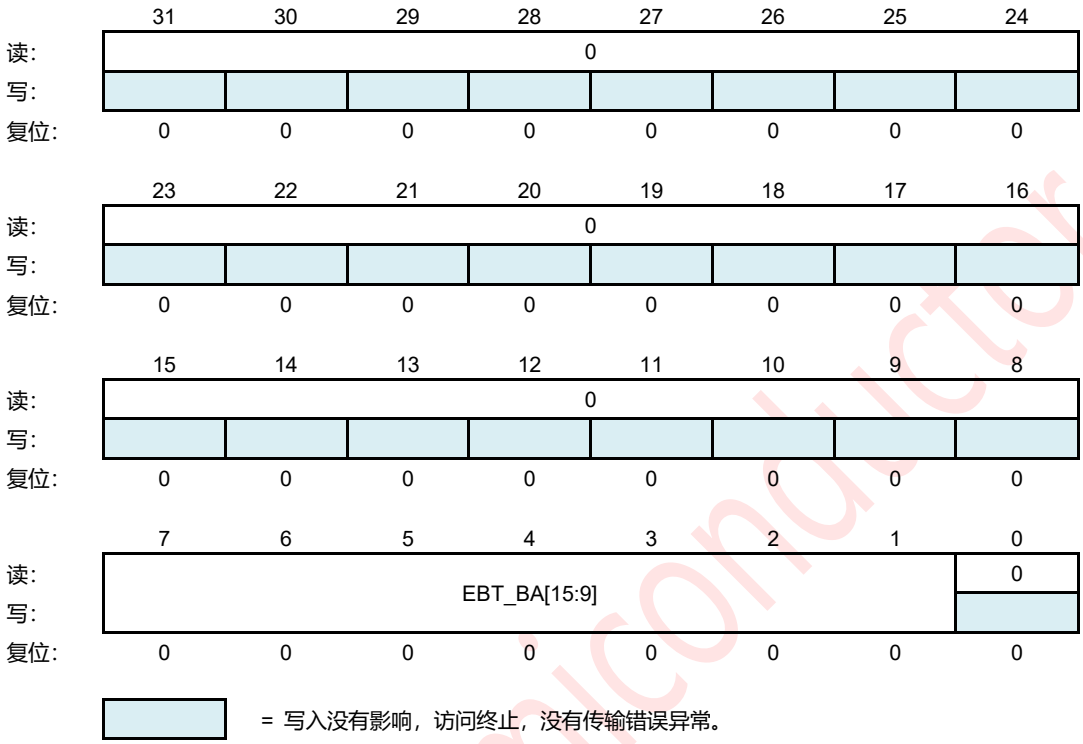


图 24-10：EBT 页面寄存器 1（EBT_PAGE_01）

EBT_BA[15:9] — This 7-bits field provides address bits 15 through 9 of the EBT base address, which defines where the Buffer Descriptor Table resides in system memory.

24.5.2.10. 帧数寄存器（FRMNUML）

帧号寄存器中包含 11 位的帧号。帧号寄存器需要两个 8 位寄存器来实现。低阶字节包含在 FRMNUML 中，高阶字节包含在 FRMNUMH 中。每当收到一个 SOF TOKEN 时，这些寄存器就会用当前的帧号进行更新。

地址：USBC_BASEADDR + 0x0000_00A0

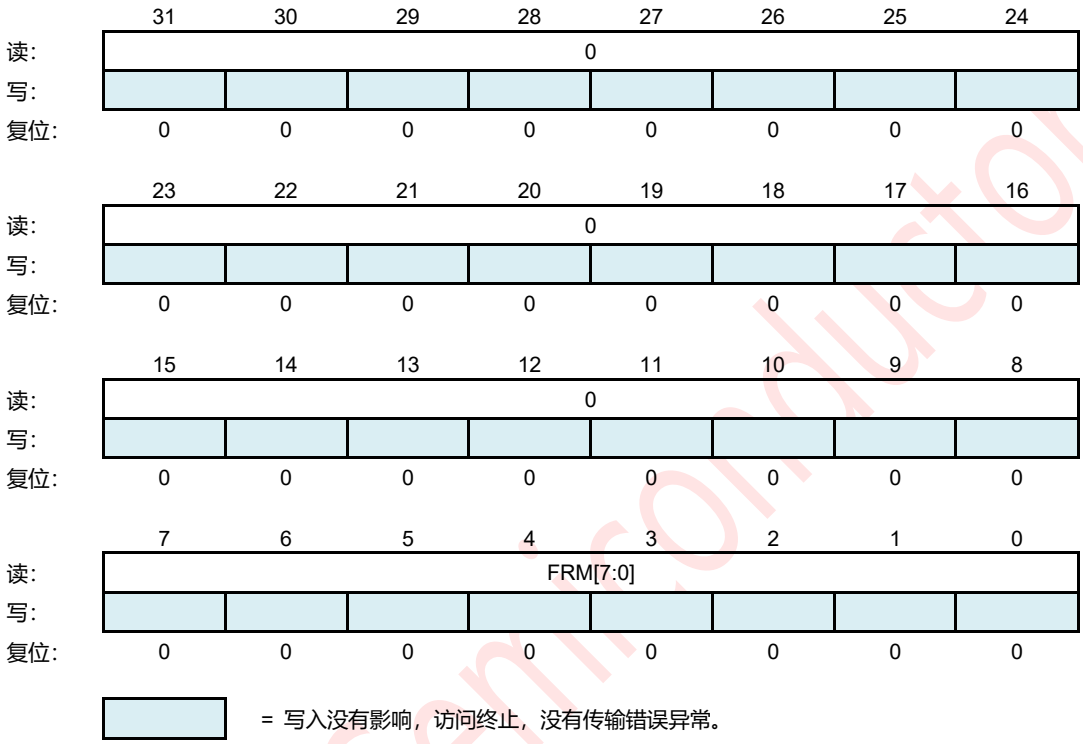


图 24-11：帧号寄存器（FRMNUML）

FRM[7:0] — Frame Number.

这些位表示 11 位帧数的低阶位。

24.5.2.11. 帧数寄存器（FRMNUMH）

帧号寄存器中包含 11 位的帧号。帧号寄存器需要两个 8 位寄存器来实现。低阶字节包含在 FRMNUML 中，高阶字节包含在 FRMNUMH 中。每当收到一个 SOF TOKEN 时，这些寄存器就会用当前的帧号进行更新。

地址：USBC_BASEADDR + 0x0000_00A4

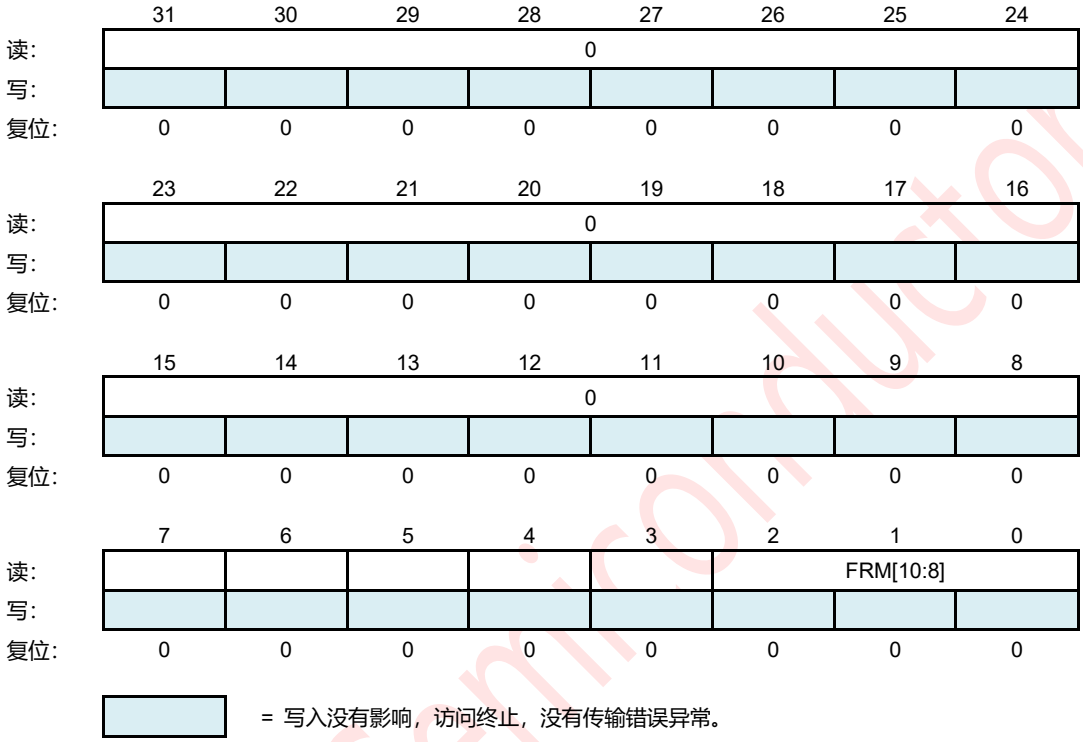


图 24-12：帧号寄存器（FRMNUMH）

FRM[10:8] — Frame Number.

这些位表示 11 位帧数的高阶位。

24.5.2.12. TOKEN 寄存器（TOKEN）

TOKEN 寄存器用于在主机模式下执行 USB 事务（HOST_MODE_EN = 1）。当处理器核心希望对外围设备执行 USB 事务时，它会将 TOKEN 类型和端点写入此寄存器。在写入此寄存器之后，USB 模块将对地址寄存器中包含的地址开始指定的 USB 事务。处理器核心在对 TOKEN 寄存器执行写入之前，应始终检查控制寄存器中的 TOKEN_BUSY 位是否没有设置。这可以确保在执行 TOKEN 命令之前不会被覆盖。在执行 TOKEN 命令时也使用地址寄存器和端点控制寄存器 0，因此也必须写入 TOKEN 寄存器之前。地址寄存器用于正确选择由 TOKEN 命令传输的 USB 外设地址。端点控制寄存器确定在传输期间使用的握手和重试策略。

地址：USBC_BASEADDR + 0x0000_00A8

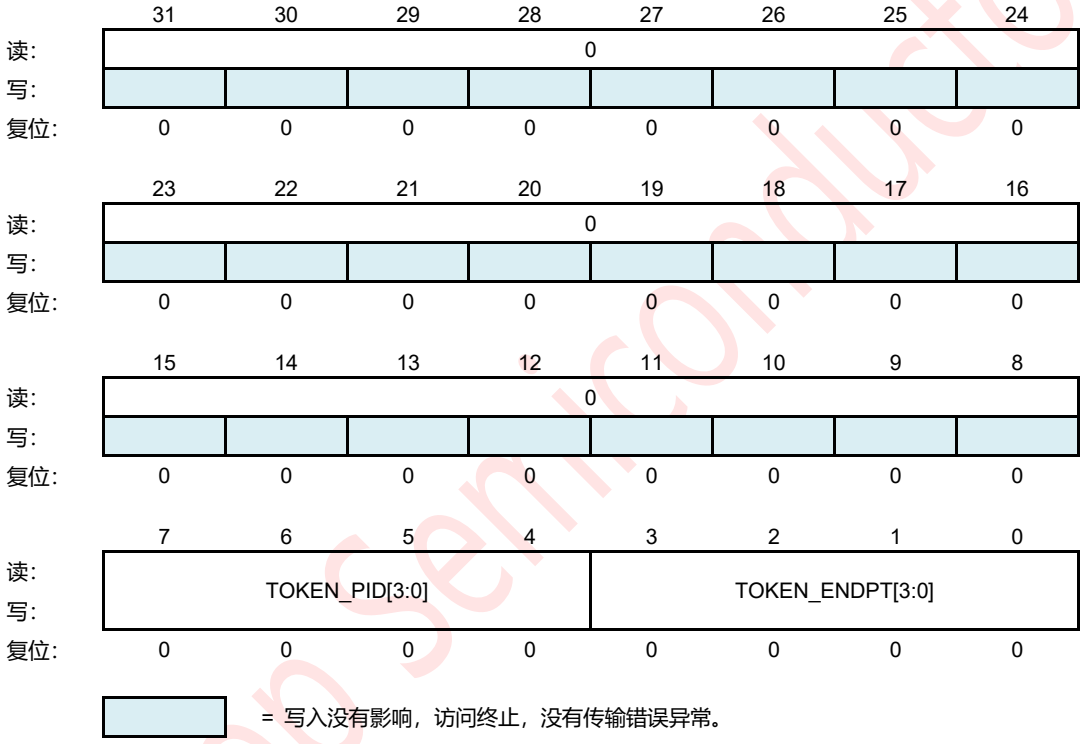


图 24-13：TOKEN 寄存器（TOKEN）

TOKEN_PID[3:0] — This 4-bits field contains the token type executed by the USB Module.

0001 OUT Token。USB 模块执行 OUT（TX）事务。

1001 IN Token。USB 模块执行一个 IN（RX）事务。

1101 SETUP Token。USB 模块执行 SET（TX）事务

TOKEN_ENDPT[3:0] — This 4-bits field holds the Endpoint address for the token command. The 4-bits value written must be a valid endpoint.

24.5.2.13. SOF 阈值寄存器（SOF_THLD）

SOF 阈值寄存器仅在主机模式下使用（HOST_MODE_EN = 1）。在主机模式下，14 位 SOF 计数器计算 SOF 帧之间的间隔。SOF 必须每 1 毫秒传输一次，因此 SOF 计数器的加载值为 12000。当 SOF 计数器达到零时，将发送帧开始（SOF）TOKEN。SOF 阈值寄存器用于编程在 SOF 之前的 USB 字节次数，以停止启动 TOKEN 数据包事务。此寄存器必须设置为一个值，以确保当 SOF 时间计数为零时不会主动地传输其他数据包。当 SOF 计数器达到阈值时，直到传输 SOF 之后才传输更多的 TOKEN。编程到阈值寄存器的值必须保留足够的时间，以确保最坏情况下的事务完成。一般来说，最坏的情况下的事务是一个 IN TOKEN，然后是来自目标的数据包，然后是来自主机的响应。实际所需的时间是总线上最大数据包大小的函数。SOF 阈值的典型值有：

64 字节数据包　　= 74；

32 字节数据包　　= 42；

16 字节数据包　　= 26；

8 字节数据包　　= 18。

地址：USBC_BASEADDR + 0x0000_00AC

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	CNT[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 24-14：SOF 阈值寄存器（SOF_THLD）

CNT[7:0] — This 8-bits field represents the SOF count threshold in byte times. This register is read only in this device.

24.5.2.14. EBT 页面寄存器 2（EBT_PAGE_02）

端点缓冲区表页面寄存器 2 包含一个 8 位值，用于计算当前端点缓冲区表（EBT）在系统内存中的地址。

地址：USBC_BASEADDR + 0x0000_00B0

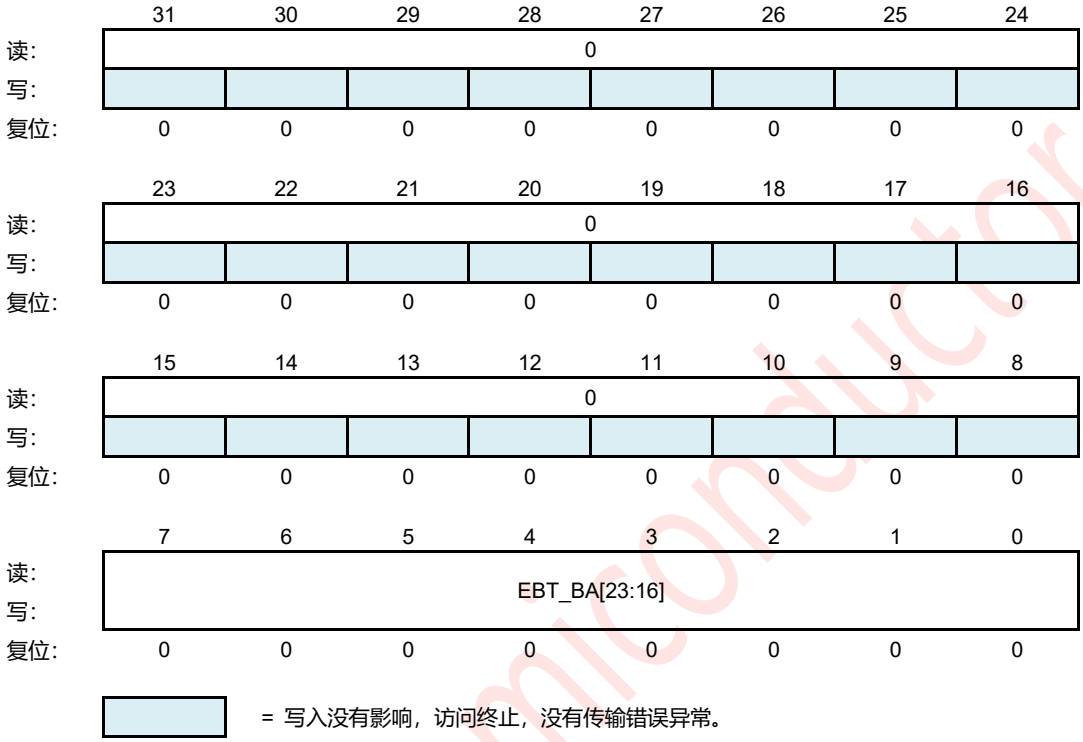


图 24-15：EBT 页面寄存器 2（EBT_PAGE_02）

EBT_BA[23:16]

这个 8 位字段提供了 EBT 基地址的地址位 23 到 16，它定义了缓冲区描述器表在系统内存中所在的位置。

24.5.2.15. EBT 页面寄存器 3（EBT_PAGE_03）

端点缓冲区表页面寄存器 3 包含一个 8 位值，用于计算当前端点缓冲区表（EBT）所在在系统内存中所在的地址。

地址：USBC_BASEADDR + 0x0000_00B4

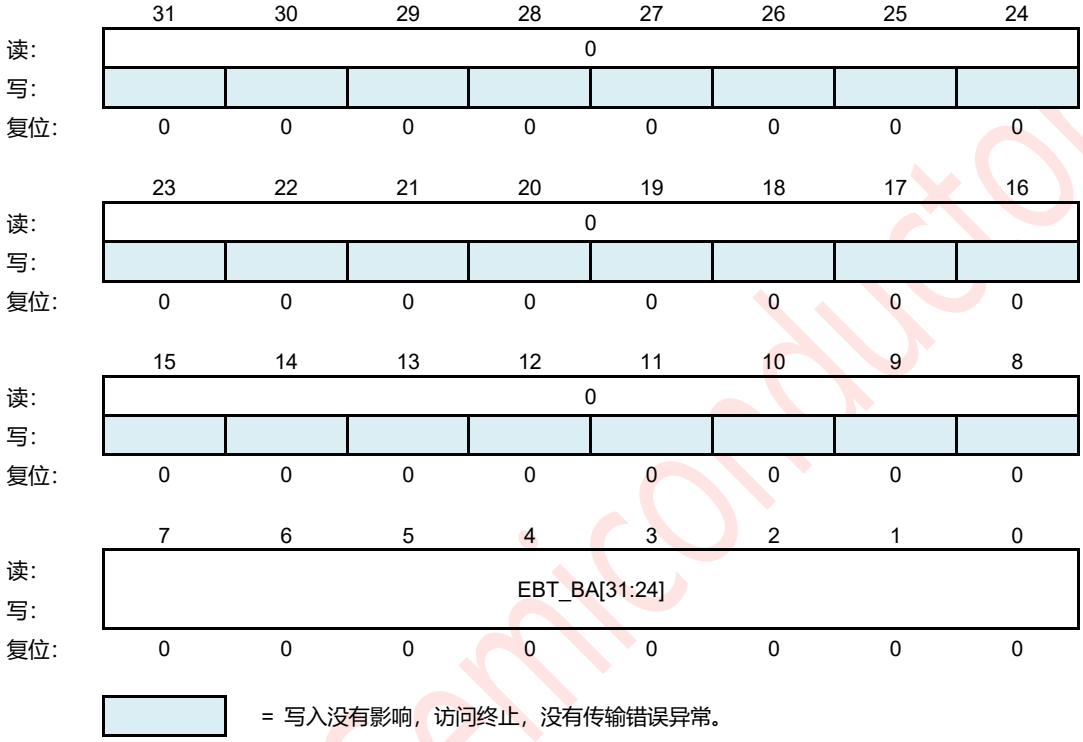


图 24-16：EBT 页面寄存器 3（EBT_PAGE_03）

EBT_BA[31:24]

这个 8 位字段提供了 EBT 基地址的地址位 31 到 24，它定义了缓冲区描述器表在系统内存中所在的位置。

24.5.2.16. 端点控制寄存器（ENDPTn）

端点控制寄存器包含在 USB 模块中可用于解码地址的 8 个端点中的每个端点控制位。这些寄存器的格式如下图所示。端点 0（ENDPT0）与控制管道 0 相关联，这是所有 USB 功能所必需的。因此，在 USB_RST 中断后，处理器核心应该将 ENDPT0 寄存器设置为包含 0x0D。

在主机模式下，ENDPT0 用于确定主机传输的握手、重试和低速特性。对于主机模式控制、批量和中断传输，EP_HSHK 位应该设置为 1。对于等时传输，它应设置为 0。在主机模式下，ENDPT0 的常用值是，控制、批量和中断传输为 0x4D，同步传输为 0x4C。

寄存器偏移地址：

ENDPT0 Address: USBC_BASEADDR + 0x0000_00C0
ENDPT1 Address: USBC_BASEADDR + 0x0000_00C4

ENDPT2 Address: USBC_BASEADDR + 0x0000_00C8
ENDPT3 Address: USBC_BASEADDR + 0x0000_00CC
ENDPT4 Address: USBC_BASEADDR + 0x0000_00D0
ENDPT5 Address: USBC_BASEADDR + 0x0000_00D4
ENDPT6 Address: USBC_BASEADDR + 0x0000_00D8
ENDPT7 Address: USBC_BASEADDR + 0x0000_00DC

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	HOST_WO_HUB	RETRY_DIS	0	EP_CTL_DIS	EP_RX_EN	EP_TX_EN	EP_STALL	EP_HSHK
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 24-17：端点控制寄存器（ENDPTn，n = 0-7）

HOST_WO_HUB — This is a Host mode only bit and is only present in the control register for endpoint 0 (ENDPT0). This bit is read only in this device.

当设置时，此位允许主机与直接连接的低速设备进行通信。当清除后，主机将生成 PRE_PID，然后在根据需要向低速设备发送信号以通过集线器与低速设备进行通信时，切换到低速信令。

RETRY_DIS — This is a Host mode only bit and is only present in the control register for endpoint 0 (ENDPT0). This bit is read only in this device.

设置时，此位将导致主机不重试 NAK 的（否定确认）事务。当事务为 NAK 时，EBT PID 字段用 NAK PID 更新，并设置 TOKEN_DNE 中断。清除此位后，将在硬件中重试 NAKed 事务。当主机试图轮询中断端点时，必须设置此位。

EP_CTL_DIS — This bit, when set, disables control (SETUP) transfers. When this bit is cleared, control transfers are enabled. This applies if and only if the EP_RX_EN and EP_TX_EN bits are also set.

EP_RX_EN — This bit, when set, enables the endpoint for RX transfers.

EP_TX_EN — This bit, when set, enables the endpoint for TX transfers.

EP_STALL — 当设置时，此位表示端点已停止。这个位优先于端点启用寄存器中的所有其他控制位，但它仅在 EP_TX_EN = 1 或 EP_RX_EN = 1 时有效。任何对此端点的访问都会导致 USB 模块返回一个停止握手。在端点停止后，它需要从主机控制器中进行干预。

EP_HSHK — When set, this bit enables an endpoint to perform handshaking during a transaction to this endpoint. This bit is generally set unless the endpoint is Isochronous.

表 24-2：端点启用/方向控制

Bit Name			端点启用/方向控制
EP_CTL_DIS	EP_CTL_DIS	EP_CTL_DIS	
X	0	0	Disable endpoint
X	0	1	Enable endpoint for IN(TX) transfers only
X	1	0	Enable endpoint for OUT(RX) transfers only
0	1	1	Enable endpoint for IN, OUT and SETUP transfers.
1	1	1	保留

24.5.2.17. USBPHY 控制寄存器 2（USBPHY_CTRL2）

地址：USBC_BASEADDR + 0x0000_0100

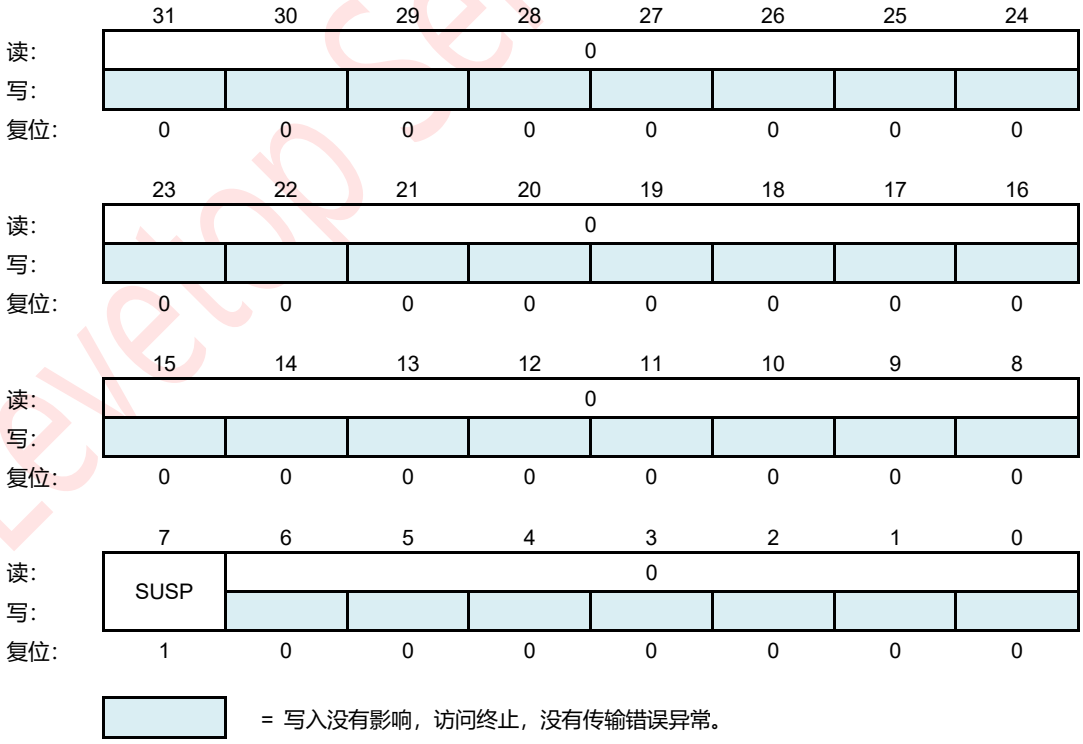


图 24-18：USBPHY 控制寄存器 2（USBPHY_CTRL2）

SUSP — Places the USB transceiver into the suspend state.

0 = USB 收发器未处于等待处理状态。

1 = USB 收发器处于等待处理状态。

24.5.2.18. USB PHY 观察寄存器（USB_PHY_OBSERVE）

地址：USBC_BASEADDR + 0x0000_0104

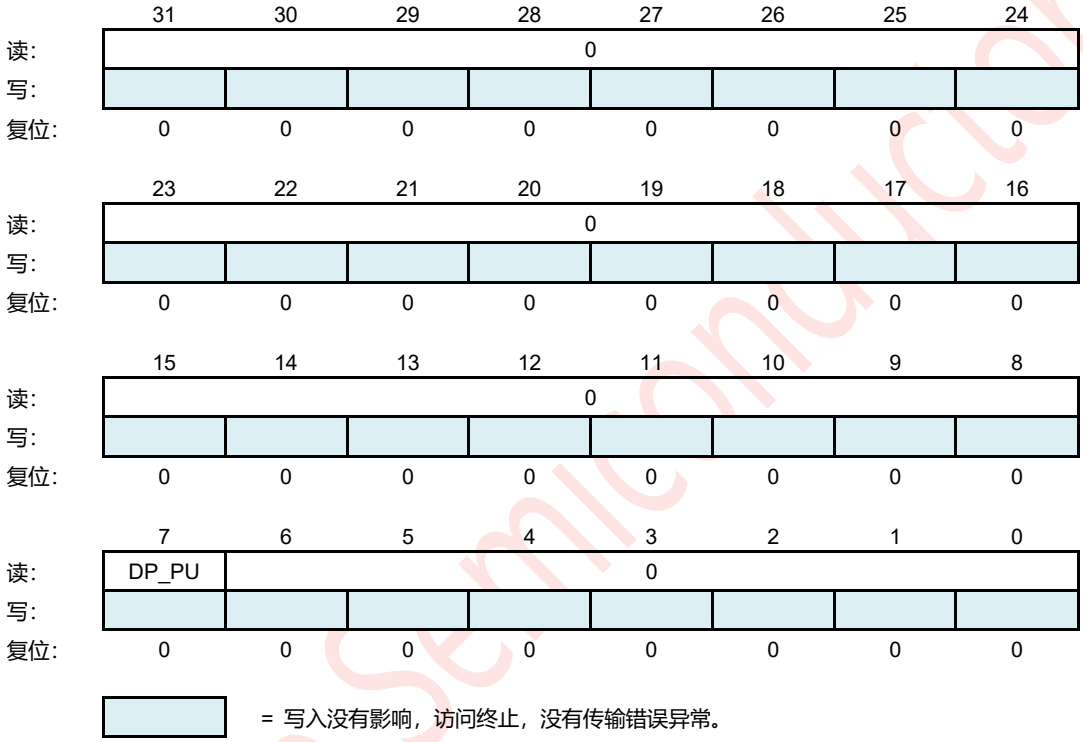


图 24-19：USB PHY 观察寄存器（USB_PHY_OBSERVE）

DP_PU — Provides observability of the D+ Pull Up signal output from the USB OTG module. This bit is useful when interfacing to an external OTG control module via a serial interface.

0 = D+上拉已禁用。

1 = D+上拉已启用。

24.5.2.19. USBPHY GPIO 寄存器（USB_PHY_GPIO）

地址：USBC_BASEADDR + 0x0000_0108

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	GPIOEN	RX_RCV	RX_DM	RX_DP	TX_DM_EN	TX_DP_EN	TX_DM	TX_DP
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 24-20：USB PHY GPIO 寄存器（USB_PHY_GPIO）

GPIOEN — USB PHY GPIO Mode control.

0 = USB PHY 不处于 GPIO 模式

1 = USB PHY 处于 GPIO 模式

RX_RCV — USB PHY Differential input data during USBPHY GPIO mode.

RX_DM — USB PHY DM input data during USBPHY GPIO mode.

RX_DP — USB PHY DP input data during USBPHY GPIO mode.

TX_DM_EN — USB PHY DM output direction control during USBPHY GPIO mode.

0 = DM 为输入方向

1 = DM 为输出方向

TX_DP_EN — USB PHY DP output direction control during USBPHY GPIO mode.

0 = DP 为输入方向

1 = DP 为输出方向

TX_DM — USB PHY DM output data during USBPHY GPIO mode.

TX_DP — USB PHY DP output data during USBPHY GPIO mode.

24.5.2.20. USB 恢复启用寄存器（USB_RESMEN）

地址：USBC_BASEADDR + 0x0000_010C

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0	0	USBRESMEN	USBPHYCLKEN	0	0	USBIRQ	RESUME
写:	USB_SFT_RESET							
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 24-21：USB 简历启用寄存器（USB_RESMEN）

USB_SFT_RESET — USB Soft Reset control bit.

USB 模块将通过写入这个位来复位。

USBRESMEN — USB resume wakeup enable control bit

如果设置了用户，并且在 USB 总线上检测到 K 状态（恢复信令），则恢复位将被设置。这将触发一个异步中断，将唤醒 MCU 从停止模式，并启用时钟到 USB 模块。

USBPHYCLKEN — USB PHY clock enable control bit.

USBIRQ — All enabled interrupt in INT_STAT register.

如果设置了 INT_STAT 中的任何一个，并且设置了相应的 INT_ENB，则将设置此位。

RESUME — USB DP/DM resume status bit.

如果设置了用户位，并且在 USB 总线上检测到 k 状态（恢复信令），则在 CPU 从停止模式唤醒并清除用户位位时设置该位。

24.5.2.21. USB PHY 控制寄存器 3（USBPHY_CTRL3）

地址：USBC_BASEADDR + 0x0000_0118

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	PHY_1.5V PDnote1	PHY_OEN note1	PHY_RESETb	USB_BYTE_SWAP	PLL_PD	PHY_SUSPEND	PHY_SUSPEND_SEL	48_60M_SEL
写:								
复位:	0	0	0	0	1	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

note1 这些位在这个芯片中没有什么意义

图 24-22：USB PHY 控制寄存器 3（USBPHY_CTRL3）

PHY_1.5VPD — USB PHY core power control.

0 = USB PHY 核心电源将不会丢失。

1 = USB PHY 核心电源将丢失。

PHY_OEN — USB PHY output isolate control.

0 = USB PHY 输出将不会被隔离。

1 = USB PHY 输出将被隔离。

PHY_RESETb — USB PHY RESET control.

0 = USB PHY 将处于复位状态。

1 = USB PHY 将为正常状态。

USB_BYTE_SWAP — When this bit is set, the data received/transmitted by USB will be swapped when reading from or writing to the system memory.

0 = USB 已接收或传输的数据将不会被交换。

1 = USB 已接收或传输的数据将被交换。

PLL_PD — USB PHY PLL power down mode control.

0 = USB PHY PLL 未处于断电状态。

1 = USBPHY PLL 处于断电状态。

PHY_SUSPEND — USB PHY Suspend control

0 = USB PHY 未处于暂停状态。

1 = USBPHY 处于暂停状态。

PHY_SUSPEND_SEL — USB PHY Suspend control selection.

0 = USB PHY 等待处理由 USBPHY_CTRL2[SUSP]控制。

1 = USB PHY 等待处理系统由 USBPHY_CTRL3[PHY_SUSPEND]控制。

48_60M_SEL — USB PHY 48/60Mhz clock selection.

0 = 48Mhz 被用作 USB SIE 解码时钟。

1 = 60Mhz 被用作 USB SIE 解码时钟。

24.5.2.22. USB PHY 控制寄存器 4（USBPHY_CTRL4）

地址：USBC_BASEADDR + 0x0000_011C

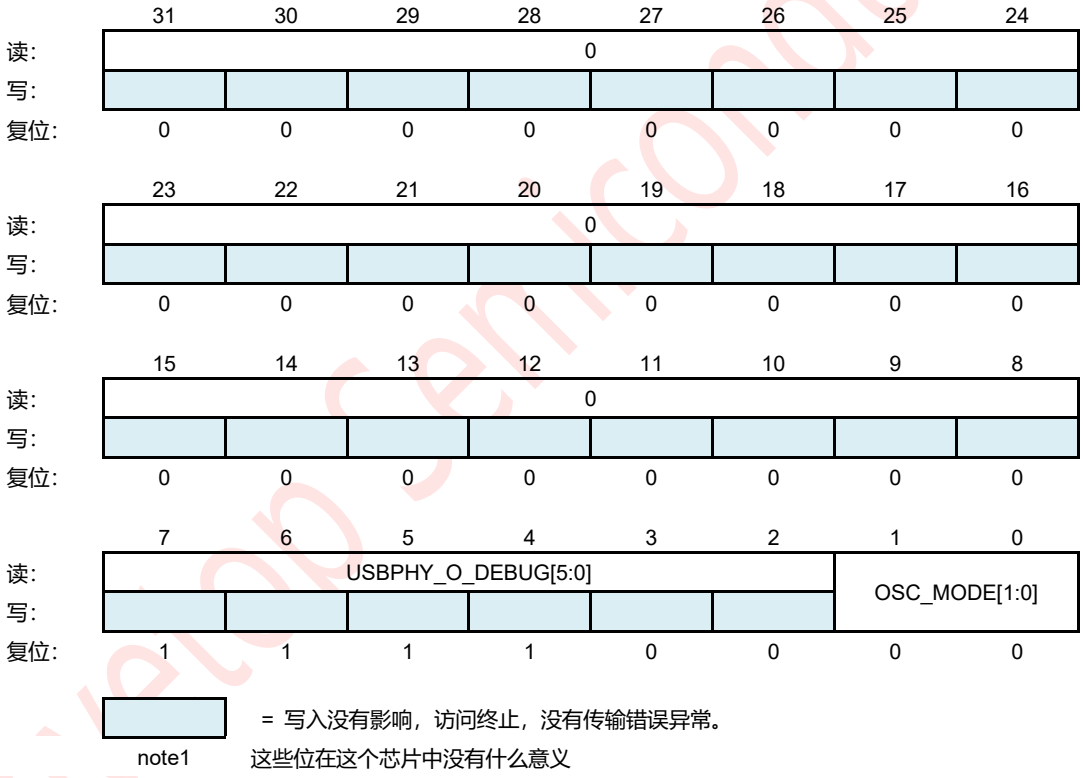


图 24-23：USB PHY 控制寄存器 4（USBPHY_CTRL4）

USBPHY_O_DEBUG[5:0] — USBPHY Output signals are for debug purpose and are meaningless during normal work.

OSC_MODE[1:0] — USB PHY clock mode selection.

表 24-3：USB PHY 振荡器模式选择

OSC_MODE[1:0]	振荡器模式选择
00	自动检测振荡器
01	自动检测振荡器（快速模拟模式）
10	选择内部振荡器
11	选择外部振荡器

24.6. 功能描述

USB-FS 2.0 全速/低速模块通过状态寄存器、控制寄存器和内存中的数据结构与处理器核心进行通信。

24.6.1. 数据结构

设备操作的功能是将存储器映像中的请求传输到通用串行总线和从通用串行总线传输。为了有效地管理 USB 端点通信，USB 模块在系统内存中实现了一个端点缓冲区表（EBT），见图 24-24。

24.6.2. 端点缓冲区表

为了有效地管理 USB 端点通信，USB 模块在系统内存中实现了一个缓冲区描述符表（EBT）。EBT 位于系统内存中的 512 字节边界上，并由 EBT 页面寄存器指向。每个端点方向都需要两个 8 字节的端点缓冲区表条目。

因此，一个有 16 个完全双向端点的系统将需要 512 字节的系统内存来实现 EBT。两个端点缓冲表（EBT）条目允许每个端点方向的 EVEB 条目和 ODD EB 条目。这允许微处理器在处理另一个 EB 条目时，USB 模块正在处理另一个 EB 条目。双缓冲 EB 入口以这种方式允许 USB SIE 轻松地以 USB 提供的最大吞吐量传输数据。

该软件 API 通过在需要时更新 EBT 来智能地管理 USB SIE 的缓冲区。这允许 USB SIE 有效地管理数据传输和接收，而微处理器执行通信开销处理和其他功能依赖的应用程序。因为缓冲区在微处理器和 USB 模块之间共享，所以使用一个简单的信号量机制来区分谁被允许更新系统内存中的 EBT 和缓冲区。当 EB 条目属于微处理器时，一个信号量位，即 OWN 位，被清除为 0。当 OWN 位为 0 时，微处理器允许对系统内存中的缓冲区进行读写访问。当 OWN 位设置为 1 时，系统内存中的 EB 入口和缓冲区归 USB 模块所有。USB 模块现在有完全的读写访问权限，微处理器不应该修改 EB 条目或其相应的数据缓冲区。EB 条目还包含指向实际缓冲区在系统内存中所在位置的间接地址指针。这种间接地址机制如下图所示。

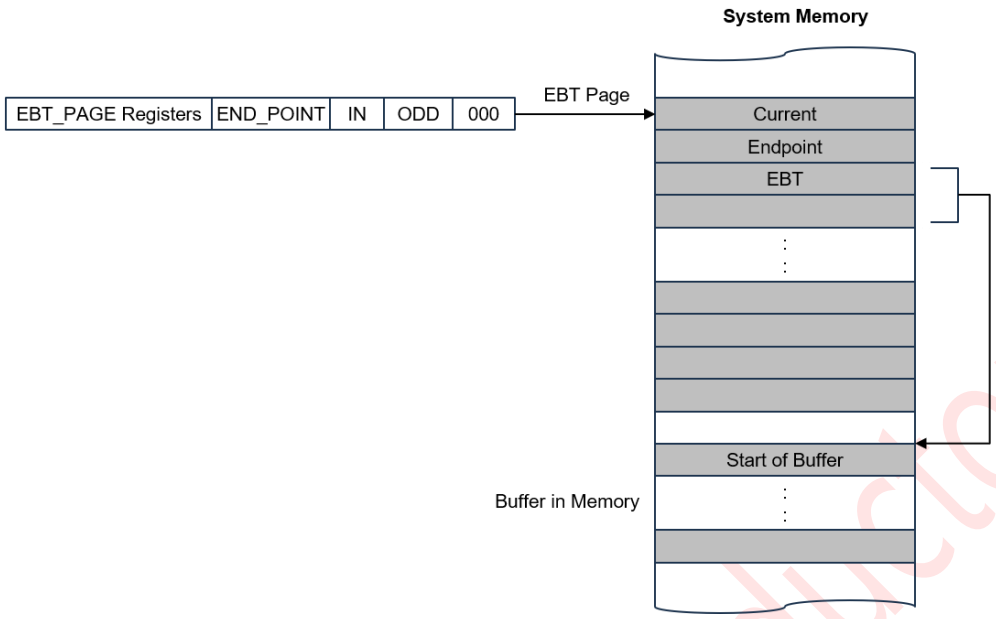


图 24-24：端点缓冲区表

24.6.3. Rx vs。Tx 作为一个 USB 目标设备

中心命名法用于描述 USB SIE 核心和 USB 主机之间的数据传输方向：

Rx（或接收）：描述将数据从 USB 移动到内存的传输。

Tx（或传输）：描述将数据从内存移动到 USB 的传输。

下表显示了数据方向如何对应于主机和目标设备应用程序中的 USB TOKEN 类型。

表 24-4：USB 目标设备的数据方向

	RX	TX
Device	OUT or Setup	IN

24.6.4. 寻址端点缓冲区表条目

当通过 USB 模块或微处理器访问端点数据时，了解端点缓冲区表的寻址机制是很有用的。一些值得关注的关注点是：

- 端点缓冲区表占用多达 512 字节的系统内存。
- 16 个双向端点的全 EBT 为 512 个字节。
- 每个 USB 端点方向需要 16 个字节。
- 端点少于 16 个端点的应用程序需要更少的 RAM 来实现 EBT。
- EBT 页面寄存器指向 EBT 的起始位置。

- EBT 必须位于系统内存中的 512 字节边界上。
- 所有启用的 TX 和 RX 端点 EB 条目都被索引到 EBT 中，以允许通过 USB 模块或 CPU 轻松访问。

当接收到启用端点上的 USB TOKEN 时，USB 模块使用其集成的 DMA 控制器来询问 EBT。USB SIE 读取相应的端点 EB 条目，以确定它是否拥有系统内存中的 EB 条目和相应的缓冲区。

为了计算 EBT 中的入口点，将 EBT_PAGE 寄存器与当前端点以及 TX 和 ODD 字段连接起来，形成一个 32 位的地址。此地址机制如下图所示：

表 24-5：EBT Address Calculation Fields

Field	Description
EBT_PAGE	EBT_PAGE registers in the Control Register Block
END_POINT	END POINT field from the USB TOKEN
TX	1 for an TX transmit transfers and 0 for an RX receive transfers
ODD	This bit is maintained within the USB SIE. It corresponds to the buffer currently in use.
	The buffers are used in a ping-pong fashion.

24.6.5. 端点缓冲区表格式

端点缓冲区表（EBT）为 USB 模块和微处理器提供端点控制信息。端点缓冲区表根据它是 USB 模块还是微处理器而具有不同的含义。

USB SIE 控制器使用存储在 EB 入口中的数据来确定：

- 谁拥有系统内存中的缓冲区
- 数据 0 或数据 1PID
- 包完成后释放自己 w
- 无地址增量(FIFO 模式)
- 启用数据切换同步
- 要传输或接收多少数据
- 缓冲区存放在系统内存中的位置

而微处理器使用存储在 EB 入口中的数据来确定：

- 谁拥有系统内存中的缓冲区
- 数据 0 或数据 1PID
- 接收到的 TOKEN PID
- 传输或接收了多少数据
- 缓冲区存放在系统内存中的位置

EB 条目的格式如下图所示。

表 24-6：端点缓冲区表字节格式

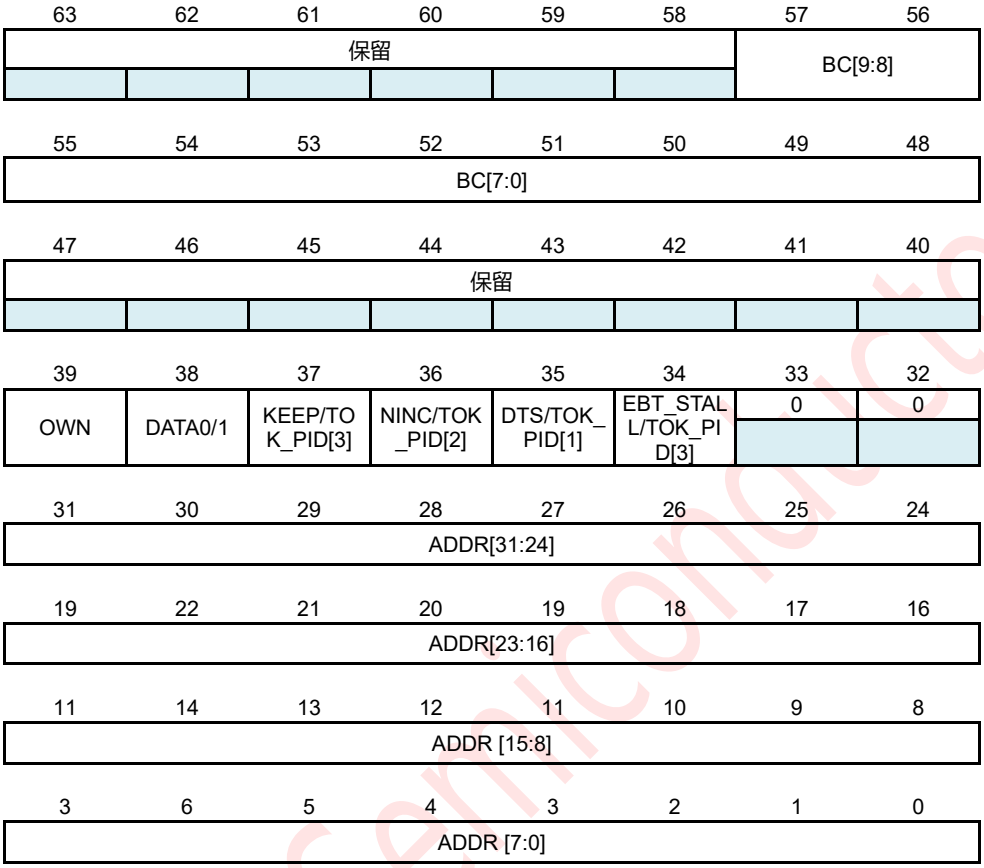


表 24-7：端点缓冲区表字节字段

Field	描述
OWN	**OWN** — This OWN bit determines who currently owns the buffer. Except when KEEP = 1, the USB SIE writes a 0 to this bit when it has completed a token. The USB module ignores all other fields in the EB entry when OWN = 0. Once the EB entry has been assigned to the USB module (OWN = 1), the MCU should not change it in any way. This byte of the EB entry should always be the last byte the MCU (firmware) updates when it initializes a EB entry. Although the hardware will not block the MCU from accessing the EB entry while owned by the USB SIE, doing so may cause undefined behavior and is generally not recommended. 0: The MCU has exclusive access to the entire EB entry 1: The USB module has exclusive access to the EB entry
DATA0/1	**Data Toggle** — This bit defines if a DATA0 field (DATA0/1 = 0) or a DATA1 (DATA0/1 = 1) field was transmitted or received. It is unchanged by the USB module. 0: Data 0 packet 1: Data 1 packet

Field	描述
KEEP/ TOK_PID[3]	**KEEP / EB entry Token PID [3]** — Typically this bit is set (that is, 1) with ISO endpoints feeding a FIFO. The microprocessor is not informed that a token has been processed, the data is simply transferred to or from the FIFO. If KEEP is set, normally the NINC bit is also set to prevent address increment. 0: Bit 3 of the current token PID is written back in to the EB entry by the USB SIE. Allows the USB SIE to release the EB entry when a token has been processed. 1: This bit is unchanged by the USB SIE. If the OWN bit also is set, the EB entry remains owned by the USB SIE forever.
NINC/ TOK_PID[2]	**No Increment / EB entry Token PID [2]** — The No Increment (NINC) bit disables the DMA engine address increment. This forces the DMA engine to read or write from the same address. This is useful for endpoints when data needs to be read from or written to a single location such as a FIFO. Typically this bit is set with the KEEP bit for ISO endpoints that are interfacing to a FIFO. 0: The USB SIE writes bit 2 of the current token PID to the EB entry. 1: This bit is unchanged by the USB SIE.
DTS/ TOK_PID[1]	**Data Toggle Synchronization / EB entry Token PID [1]** — This bit enables data toggle synchronization. Setting this bit enables the USB SIE to perform Data Toggle Synchronization. • If KEEP = 0, bit 1 of the current token PID is written back to the EB entry. • If KEEP = 1, this bit is unchanged by the USB SIE. 0: No data toggle synchronization is performed. 1: Data toggle synchronization is performed.
EBT_STALL/ TOK_PID[0]	**EBT Stall / EB entry Token PID [0]** — Setting this bit will cause the USB module to issue a STALL handshake if a token is received by the SIE that would use the EBT in this location. The EBT is not consumed by the SIE (the OWN bit remains and the rest of the EB entry is unchanged) when the EBT_STALL bit is set. • If KEEP = 0, bit 0 of the current token PID is written back to the EB entry. • If KEEP = 1, this bit is unchanged by the USB SIE. 0: EBT stall is disabled 1: USB will issue a STALL handshake if a token is received by the SIE that would use the EBT in this location
TOK_PID[n]	Bits [37:34] can also represent the current token PID. The current token PID is written back in to the EB entry by the USB SIE when a transfer completes. The values written back are the token PID values from the USB specification: • 0x1 for an OUT token. • 0x9 for an IN token. • 0xD for a SETUP token.
BC[9:0]	**Byte Count** — The Byte Count bits represent the 10-bits byte count. The USB module serial interface engine (SIE) will change this field upon the completion of a RX transfer with the byte count of the data received.
ADDR[31:0]	The Address bits represent the 32-bits buffer address in system memory. These bits are unchanged by the USB SIE.

24.6.6. USB Transaction

当 USB SIE 传输或接收数据时，它使用“寻址端点缓冲区表项”表中显示的地址生成来计算 EBT 地址。

如果为 OWN = 1，则会发生以下进程：

1. USB SIE 读取 EBT。
2. SIE 通过 DMA 将数据传输到 EB 条目的 ADDR 字段指向的缓冲区。
3. 当 TOKEN 完成时，USB SIE 将更新 EBT，如果是 KEEP = 0，则会将 OWN 位更改为 0。
4. 将更新 STAT 寄存器，并设置 TOK_DNE 中断。
5. 当微处理器处理 TOK_DNE 中断时，它从状态寄存器中读取处理端点所需的所有信息。
6. 此时，微处理器分配一个新的 EB 条目，以便可以为该端点传输或接收额外的 USB 数据，然后处理最后一个 EB 条目。

下图显示了在读取 EBT 和 OWN = 1 后如何处理典型 USBTOKEN 的时间轴。

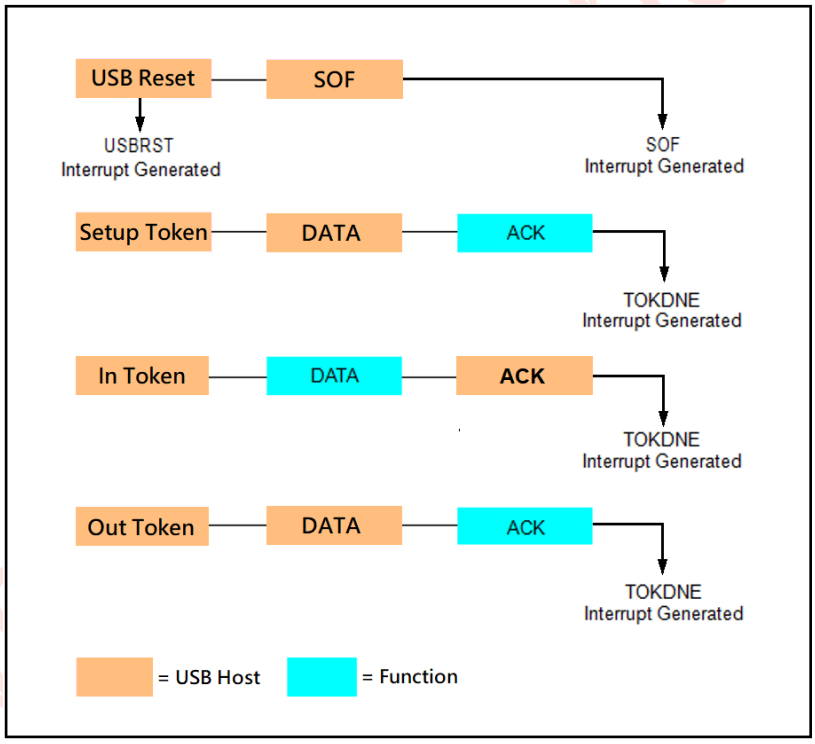


图 24-25：USB Token Transaction

该 USB 有两个来源的 DMA 溢出错误：

Memory Latency 内存延迟

DMA 接口上的内存延迟可能过高，并导致接收 FIFO 溢出。这主要是一个硬件性能问题，通常由瞬时内存访问问题引起。

Oversized Packets 超大数据包

接收到的数据包可能大于协商的最大数据包大小。通常，这是由一个软件错误引起的。对于由于数据包超大而导致的 DMA 溢出错误，USB 规范并不明确。它假定双方都有正确的软件驱动程序。打开数据包可能导致已经超大的数据包数据的重新传输。因此，为了响应超大的数据包，USB 核心继续继续进行非等时传输。

表 24-8：USB 响应的 DMA 覆盖错误

由于内存延迟而导致的错误	由于超大号的数据包而导致的错误
不确认（NAK）或总线超时（BTO）—请参考“错误中断状态寄存器（ERR_STAT）”中的第 4 位。	继续确认（确认）非同步用户的数据包。
—	写入内存的数据被剪辑到 MaxPacket 大小，以免损坏系统内存。
DMA_ERR 位是在 ERR_STAT 寄存器中设置的。根据 INT_ENB 和 ERR_ENB 寄存器的值，核心可能会发出一个中断，以通知处理器的 DMA 错误。	ERR_STAT 寄存器的 DMA_ERR 位起作用（可以触发中断），并触发 TOK_DNE 中断。 **提示：EBT 的 TOK_PID 字段不是 1111，因为 DMA_ERR 不是由于延迟所致。**
EBT 没有被写回，也不会触发 TOK_DNE 中断，因为它假设第二次尝试已经排队，并将在未来成功。	写入 EBT 的包长度字段是 MaxPacket 值，它表示实际写入内存的裁剪数据的长度。
从这里开始，软件可以为未来的事务决定一个适当的行动方案，如停止端点、取消转移、禁用端点等。	

25. 模数转换器（ADC）

25.1. 功能介绍

12 位 ADC 是一个连续的近似模数转换器。它有多达 9 个通道，允许它测量来自 8 个外部源和 2 个内部源的信号。各种通道的 A/D 转换可以在单个、连续、扫描或不连续模式下进行。ADC 的结果存储在一个 12 位 x8 深度的 FIFO 中，数据格式可以是左对齐或向右对齐的。

模拟看门狗功能允许应用程序检测输入电压是否超出用户定义的高或低阈值。

有效的低功率模式允许低频功耗。

25.2. ADC 主要功能

- 高性能
  - 12 位、10 位、8 位或 6 位的可配置分辨率
  - ADC 转换时间：12 位分辨率为 1.0μs（1MHz），10 位分辨率为 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
  - 具有内置数据一致性的数据对齐
  - DMA 支持
- 低功耗
  - 应用程序可以降低低功率操作时的 PLCK 频率，同时仍然保持最佳的 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：防止具有低频 PLCK 的应用程序中的 ADC 溢出
  - 自动关闭模式：ADC 是自动关闭，除了在活动转换阶段。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 个通道的内部参考电压
  - 内部温度传感器的 1 个通道
- 可以启动转换的开始时间：
  - 通过软件
  - 通过具有可配置极性的硬件触发器
- 转换模式
  - 可以转换单个通道或可以扫描一系列通道。
  - 单模式为每个触发器转换选定的输入一次
  - 连续模式连续转换选定的输入

-不连续模式

- 在采样结束、转换结束、序列转换结束以及发生模拟看门器或溢出事件时中断生成。
- 模拟看门狗
- 单端和差分输入配置
- 转换器使用内部引用或外部引用

25.3. ADC 功能描述

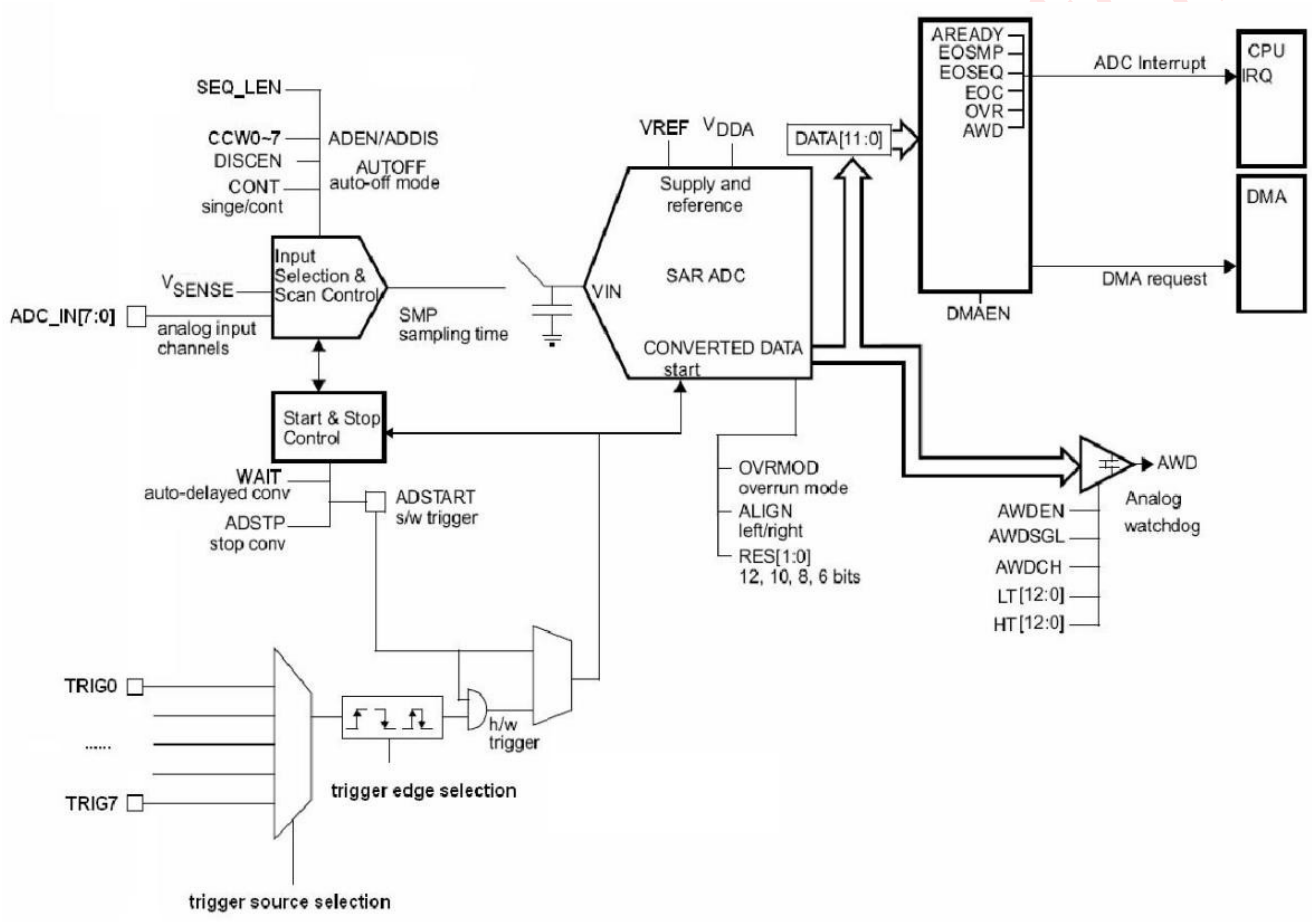


图 25-1：ADC 方块图

25.3.1. ADC 开关控制（ADEN、ADDIS、ADRDY）

在 MCU 通电时，ADC 被禁用并处于断电模式（ADEN = 0）。如下图所示 ADC 需要一个 tSTAB 的稳定时间（~2.0μs）才能开始准确转换。

有两个控制位用于启用或禁用 ADC：

- 设置 ADEN = 1 以启用 ADC。一旦 ADC 准备好进行操作，就会设置 ADRDY 标志。
- 设置 ADDIS = 1 以禁用 ADC，并将 ADC 置于断电模式。一旦 ADC 被完全禁用，硬件就会自动清除 ADEN 和 ADDIS 位。

然后，可以通过设置已启动 = 1，或者在启用触发器时发生外部触发器事件来开始转换。

按照以下步骤启用 ADC：

- 在 ADC_CR 寄存器中设置 ADEN = 1。
- 等待到 ADC_ISR 寄存器中的 ADRDY = 1（ADRDY 在 ADC 启动时间之后设置）。如果通过在 ADC_IER 寄存器中设置 ADRDYIE 位来启用中断，则可以通过中断来处理。

按照以下步骤禁用 ADC：

- 检查 ADC_CR 寄存器中启动 = 0 以确保没有正在进行的转换。如果需要，请将 1 写入 ADC_CR 寄存器中的 ADSTP 位，并等待，以停止任何正在进行的转换，直到在 0 处读取这个位。
- 在 ADC_CR 寄存器中设置 ADDIS = 1。
- 如果应用程序需要，请等待到 ADC_CR 寄存器中的 ADEN = 0，这表明 ADC 已被完全禁用（一旦 ADEN = 0，ADdis 将自动复位）。

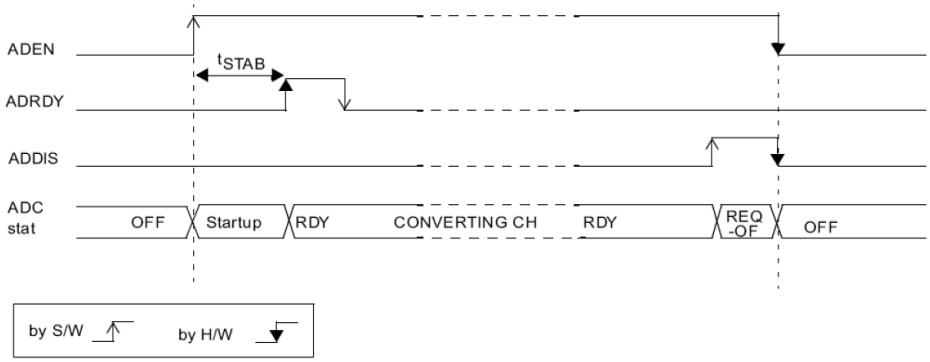


图 25-2：启用/禁用 ADC

提示： 在自动关闭模式（自动关闭 = 1）下，通过硬件自动执行开关阶段，并且不设置 ADRDY 标志。

25.3.2. ADC 时钟

ADC 具有双时钟域架构，如下图所示，其优点是无论选择 IPG 时钟方案，都能达到最大 ADC 时钟频率。

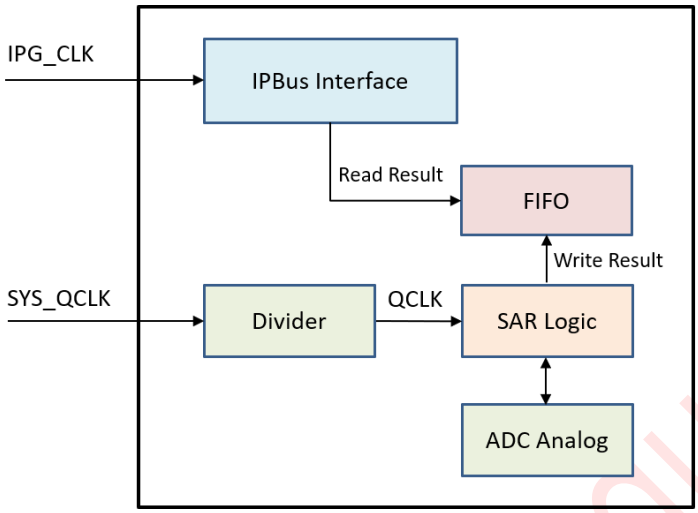


图 25-3：ADC 时钟方案

25.3.3. 配置 ADC

如果 ADC 被禁用（ADEN 必须为 0），则软件必须写入 ADC_CR 寄存器中的 ADEN 位。如果 ADC 已启用，并且没有等待请求禁用 ADC（ADEN = 1 和 ADDIS = 0），则软件只能写入 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，软件只能在没有正在进行的转换时写入配置控制位（附加启动 = 0）。

如果 ADC 已启用（可能转换）并且没有禁用 ADC 的待定请求（启动 = 1 和 ADDIS = 0），软件必须只写入 ADC_CR 寄存器中的 ADSTP 位。

提示： 没有硬件保护，防止软件进行上述规则所禁止的写操作。如果发生这样种禁止的写访问，ADC 可能进入未定义状态。在这种情况下，要恢复正确的操作，必须禁用 ADC（ADDIS = 1）。

25.3.4. 通道选择（CCWi）

最多有 10 个多路复用通道：

- 8 个模拟输入从引脚(ADC_IN0……ADC_IN7)
- 2 个内部模拟输入（细化和温度传感器）

可以转换单个通道或自动扫描通道序列。

要转换的通道的序列被组织为 CCW[0]，然后是 CCW[1]，……，然后是 CCW[7]。CCWi 是用 ADC_CHSELRi 编写的。序列长度在 ADC_CFGR2 的 SEQ_LEN[2:0]中编程。例如，如果序列长度设置为 3，则序列被组织为 CCW[0]，然后是 CCW[1]，然后是 CCW[2]。

通道解码情况如下表所示。

表 25-1：通道解码

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

25.3.5. 可编程采样时间（SMP）

在开始转换之前，ADC 需要在要测量的电压源和 ADC 的嵌入式采样电容器之间建立一个直接的连接。该采样时间必须足以使输入电压源为样品充电，并将电容器保持到输入电压电平。

具有可编程的采样时间，允许根据输入电压源的输入电阻来调整转换速度。ADC 对多个 ADC 时钟周期的输入电压进行采样，这些周期可以使用 ADC_SMPR 寄存器中的 SMP[3:0]位进行修改。这个可编程的采样时间对所有通道都是通用的。

ADC 通过设置 EOSMP 标志来表示采样阶段的结束。

25.3.6. 单次转换模式（CONT = 0）

在单转换模式下，ADC 对序列转换一次。当 ADC_CFGR1 寄存器中的 CONT = 0 时，将选择此模式。转换由以下一个开始：

- 在 ADC_CR 寄存器中设置已启动位
- 硬件触发事件

在序列内，每次转换完成后：

- 转换后的数据存储在 FIFO 中
- 已设置了 EOC（转换结束）标志
- 如果设置了 EOCIE 位，则会生成一个中断

在转换顺序完成后：

- 已设置 EOSEQ（序列结束）标志
- 如果设置了 EOSEQIE 位，则会生成一个中断

然后 ADC 停止，直到发生新的外部触发事件或重新设置已启动位。

提示：要转换单个通道一次，请编写一个长度为 1 的序列。

25.3.7. 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，然后自动重新启动并连续执行相同的转换序列。当 ADC_CFGR1 寄存器中的 CONT = 1 时，将选择此模式。转换由以下一个开始：

- 在 ADC_CR 寄存器中设置已启动位
- 硬件触发事件

在序列内，每次转换完成后：

- 转换后的数据存储在 FIFO 中
- 已设置了 EOC（转换结束）标志
- 如果设置了 EOCIE 位，则会生成一个中断

在转换顺序完成后：

- 已设置 EOSEQ（序列结束）标志
- 如果设置了 EOSEQIE 位，则会生成一个中断

然后，一个新的序列立即重新启动，ADC 不断重复转换序列。

提示：不可能同时启用不连续模式和连续模式：
禁止同时设置位 DISCEN = 1 和 CONT = 1。

25.3.8. 正在启动转换（ADSTART）

软件通过设置已启动的 = 1 来启动 ADC 转换。
当设置了 ADSTART 时，转换为：

- 如果触发模式 = 0x0（软件触发器），则立即启动
- 在选定的硬件触发器的下一个动作的上升或下降缘，如果触发模式 = 0x0

该启动位还用于指示当前是否正在进行 ADC 操作。在启动 = 0 时，可以重新配置 ADC，这表明 ADC 处于空闲状态。

重新启动位由硬件清除：

- 在单个模式下与软件触发器
  _在转换序列的任何端（EOSEQ = 1）
- 在不连续的模式下与软件触发器
  _在转换的任何一个结束处
- 在所有情况下
  _在执行了由软件调用的 ADSTP 过程后

提示：

- 在连续模式（CONT = 1）中，当设置 EOSEQ 标志时，启动启动位不会被硬件清除，因为序列会自动重新启动。
- 当在单一模式下选择硬件触发器时，当设置 EOSEQ 标志时，硬件不会清除 ADSTART。这避免了软件需要再次设置重新启动位，并确保不会错过下一个触发事件。

25.3.9. 时间安排

转换开始和转换结束之间经过的时间是配置的采样时间加上根据数据分辨率的连续近似时间的总和：

tADC = tSMPL + tSAR = [4|min + 12|12-bits] x tQCLK = 1µs |min (for fQCLK = 16MHz)

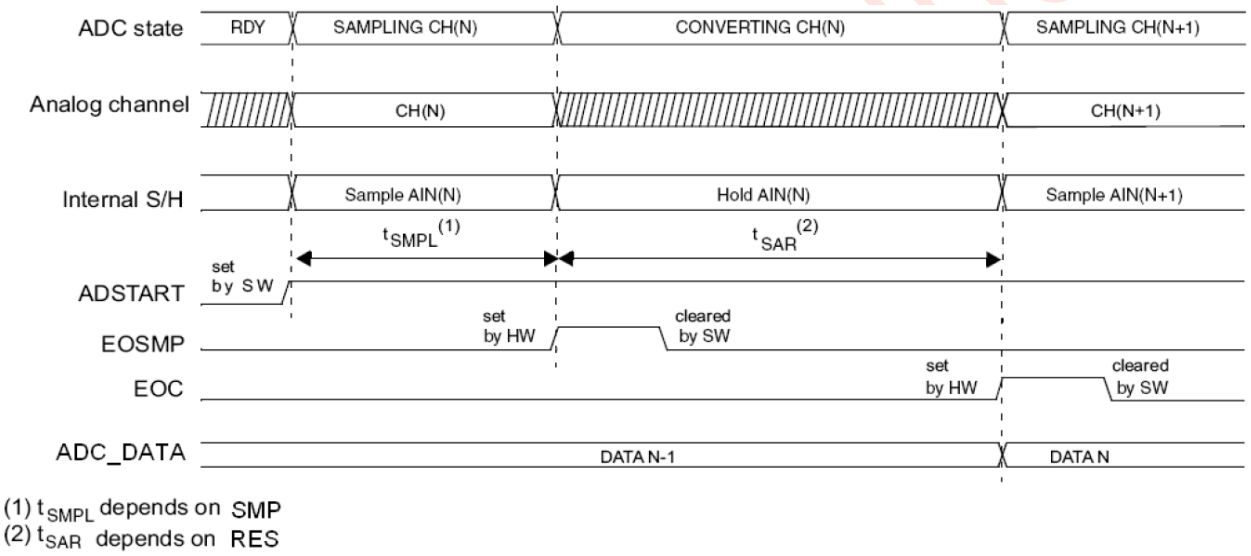


图 25-4：Analog to Digital 转换时间

25.3.10. 停止一个持续的转换（ADSTP）

该软件可以通过在 ADC_CR 寄存器中设置 ADSTP = 1 来决定停止任何正在进行的转换。

这将复位 ADC 操作，ADC 将空闲，准备进行新的操作。

当由软件设置 ADSTP 位时，任何正在进行的转换都会被中止，结果也会被丢弃（FIFO 不会用当前的转换进行更新）。扫描序列也会中止和复位（这意味着重新启动 ADC 将重新启动新序列）。

一旦此过程完成，ADSTP 和 ADSTART 都被硬件清除，软件必须等到 ADSTART = 0 才开始新的转换。

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。

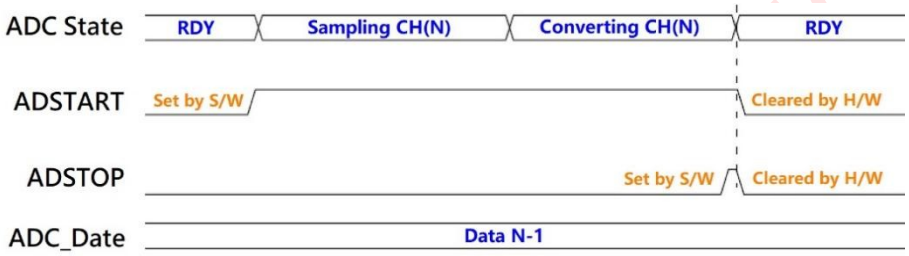


图 25-5：停止一个正在进行的转换

25.4. 在外部触发器和触发器极性上的转换

转换或转换序列可以由软件或外部事件触发。如果三角模式控制位不等于“0”，则外部事件能够触发具有所选极性的转换。一旦软件设置位启动 = 1，触发选择有效。

在正在进行的转换时发生的任何硬件触发器都将被忽略。

如果位启动 = 0，则忽略发生的任何硬件触发器。

下表提供了三角模式值与触发器极性之间的对应关系。

表 25-2：正在配置触发器的极性。

TRIGMODE[2:0]	Source
3’b000	触发器检测被禁用，软件触发器
3’b001	上升缘检测
3’b010	下降缘检测
3’b011	对上升和下降边缘的检测
3’b100	在高电平电压上的检测
3’b101	在低水平电压上的检测
3’b110	PIT 0 检测
3’b111	在 PWM 0 上检测

提示：

只有当 ADC 没有转换时，外部触发器的极性才能被改变（附加启动 = 0）。

触发器控制位用于选择 8 个可能的事件中的哪一个可以触发转换。下表为常规转换提供了可能的外部触发器。软件源触发器事件可以通过在 ADC_CR 寄存器中设置动态启动位来生成。

表 25-3：配置触发器的极性

TRIGSCR[2:0]	Name	Source
3’b000	TRG0	
3’b001	TRG1	
3’b010	TRG2	
3’b011	TRG3	
3’b100	TRG4	
3’b101	TRG5	
3’b110	TRG6	
3’b111	TRG7	

提示： 只有当 ADC 没有转换时，才能更改触发器选择（已启动 = 0）

25.4.1. 不连续模式（DISCEN）

通过在 ADC_CFGR1 寄存器中设置 DISCEN 位来启用此模式。

在此模式下 (DISCEN = 1) 中，需要一个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，则单个硬件或软件触发事件依次启动序列中定义的所有转换。

样例：

- DISCEN = 1，通道要转换为 = 0、3、7、10
  - 第 1 个触发器：通道 0 转换并生成 EOC 事件
  - 第 2 个触发器：转换通道 3 并生成一个 EOC 事件

    外部触发器和触发器极性的转换（三角模式、触发模式）
  - 第 3 个触发器：转换通道 7 并生成一个 EOC 事件
  - 第 4 个触发器：转换通道 10，并生成 EOC 和 EOSEQ 事件。
  - 第 5 个触发器：通道 0 被转换，生成一个 EOC 事件
  - 第 6 个触发器：第 3 通道被转换，并生成一个 EOC 事件

  – .....
- DISCEN = 0，要转换的通道 = 0、3、7、10
  - 第 1 个触发器：完整的序列被转换为：通道 0，然后是通道 3、通道 7 和通道 10。每个转换都会生成一个 EOC 事件，最后一个转换也会生成一个 EOSEQ 事件。
  - 任何后续的触发事件都将重新启动整个序列。

25.4.2. 可编程分辨率（RES）- 快速转换模式

通过降低 ADC 的分辨率，可以获得更快的转换时间（tSAR）。通过在 ADC_CFGR1 寄存器中编程 RES[1:0]位，可以将分辨率配置为 12、10、8 或 6 位。对于不需要高数据精度的应用程序，较低的分辨率允许更快的转换时间。

提示：

只有在复位 ADEN 位时，才能更改 RES[1:0]位。

转换的结果总是是 13 位宽的，任何未使用的 LSB 位都被读取为零。

较低的分辨率减少了连续近似步骤所需的转换时间。

25.4.3. 转换结束，采样阶段结束(EOC、EOSMP Flag)

ADC 表示每次转换结束（EOC）事件。

一旦有一个新的转换数据结果可用，ADC 就会在 ADC_ ISR 寄存器中设置 EOC 标志。如果在 ADC_IER 寄存器中设置了 EOCIE 位，则可以产生一个中断。软件可以通过写入 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。软件可以通过写入 1 来清除 EOSMP 标志。如果在 ADC_IER 寄存器中设置了 ADC_IER 位，则可以生成一个中断。

25.4.4. 转换序列的结束(Eoseq Flag)

ADC 通知每个序列（EOSEQ）事件的应用程序。

一旦转换序列的最后一个数据结果在 FIFO 中可用，ADC 就会在 ADC_ISR 寄存器中设置 EOSEQ 标志。如果在 ADC_IER 寄存器中设置了 ADC_IER 位，则可以生成一个中断。软件可以通过写入 1 来清除 EOSEQ 标志。

25.4.5. 时序图范例

本章节显示了硬件或软件触发条件下的单个模式和连续模式。

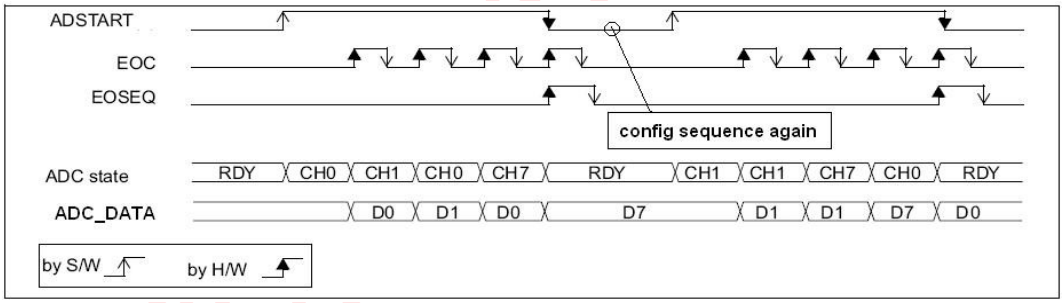


图 25-6：一个序列的单一转换，软件触发器

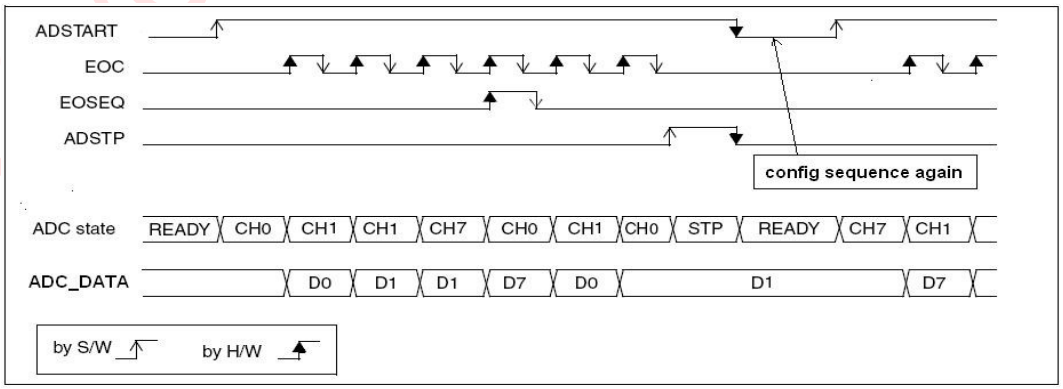


图 25-7：一个序列的连续转换，软件触发器

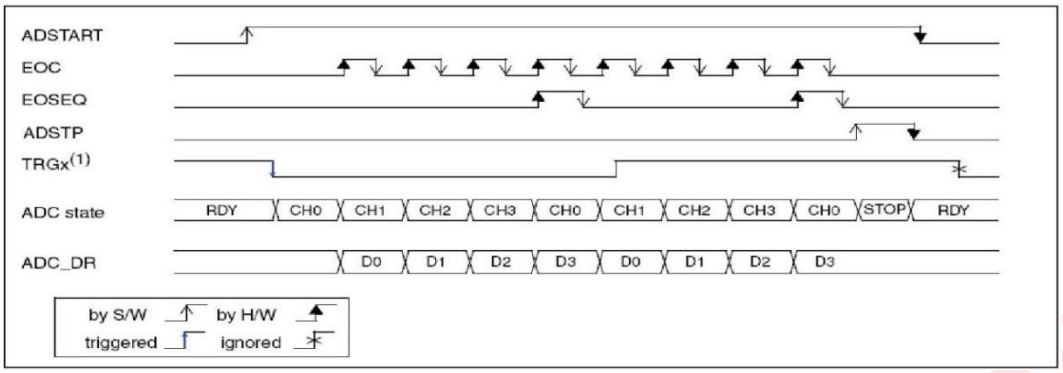


图 25-8：一个序列的单一转换，硬件触发器

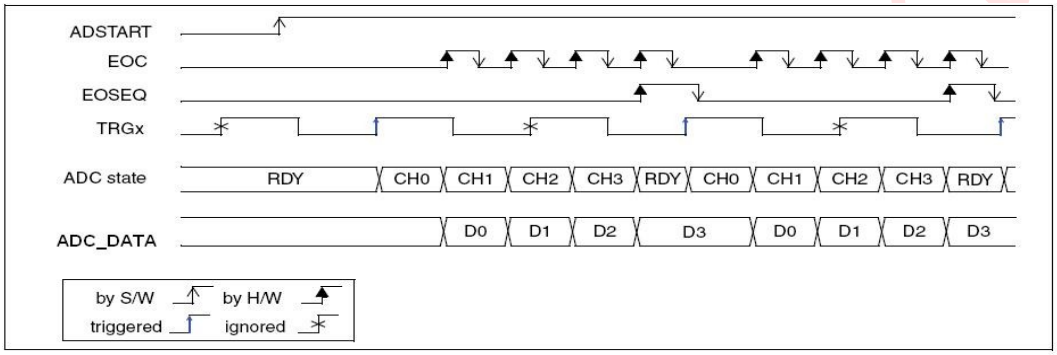


图 25-9：一个序列的连续转换，硬件触发器

25.5. 数据管理

25.5.1. 数据 FIFO 和数据对齐(ADC_FIFO，对齐)

在每次转换结束时（当 EOC 事件发生时），转换数据的结果存储在 ADC_FIFO 中，它为 13 位宽 x8 深度。

读出数据的格式取决于所配置的数据对齐和分辨率。

ADC_CFGR1 寄存器中的对齐位选择转换后存储的数据的对齐方式。数据可以是右对齐 (对齐 = 0)或左对齐 (对齐 = 1)，如下表所示。

表 25-4：数据对齐和分辨率

Align	RES[1:0]	31	30	….	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0x0			….					Data[11:0]											
	0x1			….							Data[9:0]									
	0x2			….									Data[7:0]							
	0x3			….											Data[5:0]					
1	0x0			….	Data[11:0]															
	0x1			….	Data[9:0]															
	0x2			….									Data[7:0]							
	0x3			….									Data[5:0]							

FIFO 支持字节、半字和字的读取，但地址偏移量应始终为 0x4C。
对于不同的数据对齐和分辨率，用户应提示：

- 如果按字读取，则数据格式为数据[31:0]，如上表所示。
- 如果按半字读取，则数据格式为数据[15:0]，如上表所示。
- 如果在数据大于 8 位时按字节读取，则将首先读出高字节，然后应该再次读取低字节。
- 如果在数据不超过 8 位时按字节读取，则数据格式为数据[7:0]，如上表所示。

25.5.2. ADC 超运行（OVR、OVRMOD）

溢出标志（OVR）表示在 FIFO 满之前没有及时读取转换后的数据时的数据溢出事件。

如果在一个新的转换完成时，FULL 标志仍然在“1”，则会在 ADC_ISR 寄存器中设置 OVR 标志。如果在 ADC_IER 寄存器中设置了 OVRIE 位，则可以产生一个中断。

当发生溢出情况时，ADC 将继续运行，并可以继续转换，除非软件决定通过在 ADC_CR 寄存器中设置 ADSTP 位来停止和复位序列。

软件可以通过写入 1 来清除 OVR 标志。

可以通过对 ADC_CFGR1 寄存器中的 OVRMOD 位进行编程，来配置，在溢出事件发生时，数据是否被保留或覆盖：

•OVRMOD = 0

-溢出事件保留数据寄存器不被覆盖：维护旧数据，丢弃新的转换。如果 OVR 保持在 1，则可以执行进一步的转换，但结果数据将被丢弃。

•OVRMOD = 1

-数据寄存器被最后的转换结果所覆盖。如果 OVR 保持在 1，则可以执行进一步的转换，并且 FIFO 总是包含来自最新转换的数据。

25.5.3. 管理不使用 DMA 而转换的数据序列

如果转换速度足够慢，则转换顺序可以由软件来处理。在这种情况下，软件可以使用 EOC 标志及其相关的中断来处理每个数据结果。每次转换完成时，在 ADC_ISR 寄存器中设置 EOC 位，并且可以读取 FIFO 寄存器。

软件还可以使用 FIFO 空标志来处理每个数据结果。如果空不是“0”，这意味着 FIFO 有新的数据。

ADC_CFGR1 寄存器中的 OVRMOD 位应该配置为 0，以作为一个错误来管理溢出事件。

25.5.4. 管理转换的数据而不使用 DMA

让 ADC 在一个或多个通道转换后不读取数据可能很有用。在这种情况下，OVRMOD 位必须配置为 1，并且软件应该忽略 OVR 标志。当 OVRMOD = 1 时，溢出事件不会阻止 ADC 继续转换，而 FIFO 始终包含最新的转换数据。

25.5.5. 使用 DMA 管理转换后的数据

一旦 FIFO 中的数据号不是空的，并且设置了位 DMAEN，QADC 将向 DMA 发送请求。这允许将转换后的数据从 FIFO 传输到软件选择的目标位置。

尽管如此，如果由于 DMA 无法及时服务 DMA 传输请求而发生溢出（OVR = 1），ADC 将停止生成 DMA 请求，与新转换对应的数据不会被 DMA 传输。这意味着传输到 RAM 的所有数据都可以被认为是有效的。

根据 OVRMOD 位的配置，数据会被保留或覆盖。

DMA 传输请求被阻止，直到软件清除 OVR 位。

25.6. 低功耗功能

25.6.1. 等待模式转换

等待模式转换可用于简化软件，并优化低频应用程序的性能，这可能会有发生 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 的速度以适应读取数据的系统的速度的方法。

提示： 在转换进行或读访问之前的等待期间发生的任何硬件触发器将被忽略。

25.6.2. 自动关闭模式（AUTOFF）

ADC 具有一个自动电源管理功能，它被称为自动关闭模式，并通过在 ADC_CFGR1 寄存器中设置自动关闭 = 1 来启用。

当自动关闭 = 1 时，ADC 在不转换时总是关闭，并且当转换启动时自动唤醒（通过软件或硬件触发器）。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 将被自动禁用。

自动关闭模式可以导致显著降低功耗的应用程序需要相对较少的转换或当转换请求的时间足够远的距离（例如，低频硬件触发）来证明额外的权力和额外的时间用于开关 ADC。

自动关闭模式可以与等待模式转换（WAIT = 1）结合低频。如果 ADC 在等待阶段自动关闭，并在应用程序读取 FIFO 后立即重新启动，那么这种组合可以显著节省电力

25.7. 模拟窗口看门狗（AWD）

通过在 ADC_CFGR1 寄存器中设置 AWDEN 位来启用 AWD 模拟看门狗功能。它用于监控一个选定通道或所有启用通道是否保持在配置的电压范围（窗口）内，如图 25-10 所示。

如果由 ADC 转换的模拟电压低于较低的阈值或高于较高的阈值，则设置 AWD 模拟看门狗状态位。这些阈值在 ADC_TR 寄存器中被编程。可以通过在 ADC_IER 寄存器中设置 AWDIE 位来启用中断。

AWD 标志被软件通过写入 1 来清除。

当转换分辨率小于 12 位的数据时（根据位 RES[1:0]），必须清除编程阈值的 LSB，因为总是对完整的 12 位原始转换数据（左对齐）进行内部比较。

表 25-5 显示如何在 theADC_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。

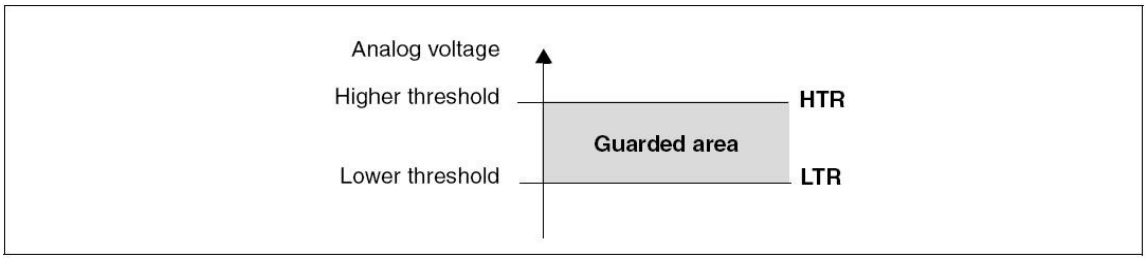


图 25-10：模拟式看门狗的监视区域

表 25-5：模拟监控器通道选择

Channels Guarded by the Analog Watchdog	AWDSGL Bit	AWDEN Bit
没有一个	X	0
所有通道	0	1
单个(1)通道	1	1

注(1)：由 AWDCH 选择

25.8. 温度传感器

温度传感器内部连接到 ADC1_IN15 输入通道，用于将传感器的输出电压转换为数字值。

25.9. ADC 中断

以下任一事件都可以产生中断：

- 当 ADC 准备就绪时，ADC 通电（ADRDY 标志）
- 任何转换的结束(EOC 标志)
- 转换序列的结束(EOSEQ 标志)
- 当发生模拟看门狗检测时(AWD 标志)
- 当采样阶段结束时（EOSMP Flag）
- 当发生数据溢出(OVR 标志）时，单独的中断启用位可提供灵活性。

表 25-6：ADC 中断

Interrupt Event	Event Flag	Enable Control Bit
ADC Ready	ADRDY	ADRDYIE
End of Conversion	EOC	1EOCIE
End of Sequence of Conversions	EOSEQ	1EOSEQIE
Analog Watchdog Status bit is Set	AWD	AWDIE
End of Sampling Phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE

25.10. 内存映射和寄存器

ADC 模块内存映射如下表 25-7 所示 ADC 基地址为 0x4011_0000。本章节描述了 ADC 的内存映射和寄存器结构。

25.10.1. 内存映射

见下表于 QADC 内存映射的描述。

表 25-7：ADC 模块内存映射

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	EOSEQ	EOC	EOSMP	ADRDY
0x04	ADC_IER																									AWDIE			OVRIE	EOSEQIE	EOCIE	EOSMPIE	ADRDYIE
0x08	ADC_CR																												ADSTP	ADSTART	ADDIS	ADEN	
0x0C	ADC_CFGR1	DIFF	OVRMOD				SEQLEN[2:0]			DISCEN	AUTOFF	WAIT	CONT		TRIGSCR[2:0]						TRIGMODE[2:0]			ALIGN	RES[1:0]								DMAEN
0x10	ADC_CFGR2																					QPR[3:0]				STCNT[7:0]							
0x14	ADC_SMPR																									SMP[7:0]							
0x18	ADC_WDG																									AWDEN	AWDSGL			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[3:0]								CCW6[3:0]								CCW5[3:0]								CCW4[3:0]			
0x4C	ADC_FIFO																	DATA[15:0]															

提示： 1.所有的寄存器都是 CPU 管理或用户模式是可访问的。

2.灰色位是保留的，并且必须保持在复位值。

25.10.2. 寄存器描述

25.10.2.1. ADC 中断和状态寄存器（ADC_ISR）

地址：ADC_BASEADDR + 0x0000_0000

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	AWD	EMPTY	FULL	OVR	EOSEQ	EOC	EOSMP	ADRDY
写:	w1c			w1c	w1c	w1c	w1c	w1c
复位:	0	1	0	0	0	0	0	0

	= 写入没有影响，访问终止，没有传输错误异常。
w1c	= 写 1 到该位将清除它

图 25-11：ADC 中断和状态寄存器（ADC_ISR）

Read: Anytime

Write: Anytime

AWD — Analog watchdog flag

当转换后的电压超过 ADC_TR 寄存器中编程的值时，该位由硬件设置。它可以通过软件写入 1 来清除。

1 = 模拟看门狗事件

0 = 没有发生模拟看门狗事件（或标志事件已被软件确认和清除）

EMPTY — FIFO empty status

当 FIFO 为空时，此位由硬件设置。当 FIFO 不为空时，它将由硬件清除。

1 = FIFO 为空

0 = FIFO 不是空的

FULL — FIFO full status

当 FIFO 已满时，此位将由硬件设置。当 FIFO 未满时，它将由硬件清除。

1 = FIFO 已满

0 = FIFO 未满

OVR — ADC overrun

这个位是在发生溢出时由硬件设置的，这意味着在已经设置了满标志时，一个新的转换已经完成。它可以通过软件写入 1 来清除。

1 = 已发生溢出

0 = 没有发生溢出（或标志事件已被软件确认和清除）

EOSEQ — End of sequence flag

这个位是由硬件在一个序列的转换结束时设置的。它可以通过软件写入 1 来清除。

1 = 转换序列已完成

0 = 转换序列未完成（或标志事件已被软件确认和清除）

EOC — End of conversion flag

当 ADC_FIFO 寄存器中有一个新的数据结果时，由硬件在通道的每次转换结束时设置这个位。通过软件写入 1 或读取 ADC_FIFO 寄存器来清除。

1 = 通道转换完成

0 = 通道转换未完成（或标志事件已被软件确认和清除）

EOSMP — End of sampling flag

这个位是在采样阶段结束时由硬件设置的。

1 = 采样阶段结束

0 = 未在采样阶段结束时（或标志事件已被软件确认和清除）

ADRDY — ADC ready

这个位是在启用 ADC 之后（位 ADEN = 1）和当 ADC 达到它准备好接受转换请求的状态时由硬件设置的。它可以通过软件写入 1 来清除。

1 = ADC 已准备好开始转换

0 = ADC 尚未准备好开始转换（或标志事件已被软件确认和清除）

25.10.2.2. ADC 中断启用寄存器（ADC_IER）

地址：ADC_BASEADDR + 0x0000_0004

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	AWDIE	0	0	OVRIE	EOSEQIE	EOCIE	EOSMPIE	ADRDYIE
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 25-12：ADC 中断启用寄存器（ADC_IER）

Read: Anytime

Write: Before ADC start

AWDIE — Analog watchdog interrupt enable
此位由软件设置和清除，以启用/禁用模拟看门狗中断。
1 = 已启用模拟看门狗中断
0 = 模拟看门狗中断被禁用

OVRIE — Overrun interrupt enable
软件已设置并清除此位，以启用/禁用溢出中断。
1 = 溢出中断已启用。当设置了 OVR 位时，就会产生一个中断。
0 = 溢出中断被禁用

EOSEQIE — End of conversion sequence interrupt enable
该位由软件设置和清除，以启用/禁用转换中断序列的结束。
1 = EOSEQ 中断已启用。当设置了 EOSEQ 位时，就会产生一个中断。
0 = EOSEQ 中断被禁用

EOCIE — End of conversion interrupt enable

此位由软件设置和清除，以启用/禁用转换中断的结束。

1 = EOC 中断已启用。当设置了 EOC 位时，就会产生一个中断。

0 = EOC 中断被禁用

EOSMPIE — End of sampling flag interrupt enable

该位由软件设置和清除，以启用/禁用采样阶段中断的结束。

1 = EOSMP 中断已启用。当设置了 EOSMP 位时，就会产生一个中断。

0 = EOSMP 中断被禁用。

ADRDYIE — ADC ready interrupt enable

此位被软件设置和清除，以启用/禁用 ADC 准备中断。

1 = ADRDY 中断已启用。当设置了 ADRDY 位时，就会产生一个中断。

0 = ADRDY 中断被禁用。

25.10.2.3. ADC 控制寄存器（ADC_CR）

地址：ADC_BASEADDR + 0x0000_0008

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	1	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0				ADSTP	ADSTART	ADDIS	ADEN
写:					rs	rs	rs	rs
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

rs = 软件可以读取和设置，写 0 没有影响

图 25-13：ADC 控制寄存器（ADC_CR）

Read: Anytime

Write: See each bit description

ADSTP — ADC stop conversion command

这个位是由软件设置为停止和丢弃一个正在进行的转换（ADSTP 命令）。当转换被有效地丢弃并且 ADC 准备接受新的开始转换命令时，硬件清除。

1 = 写入 1 以停止 ADC。读取 1 表示一个 ADSTP 命令正在进行中。

0 = 没有 ADC 停止转换命令正在进行中

提示： 软件只允许在 ADSTP 启动 = 1 和 ADDIS = 0 时设置 ADSTP（ADC 已启用，可能正在转换，没有等待的请求来禁用 ADSTP）

ADSTART — ADC start conversion command

这个位是由软件设置来启动 ADC 转换的。根据三叉模式配置位的不同，转换可以立即开始 (软件触发配置)，或者一旦硬件触发事件发生（硬件触发配置）。

通过硬件进行清除：

- 在选择软件触发器时，单一转换模式：在转换顺序结束（EOSEQ）标志。
- 选择软件触发器时，处于停止转换模式：显示转换结束（EOC）标志。
- 在所有情况下：在执行 ADSTP 命令后，同时，ADSTP 位被硬件清除。

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 = 没有正在进行的 ADC 转换。

提示： 只有当 ADEN = 1 和 ADDIS = 0（启用 ADC 且没有禁用 ADC 的待定请求）时，软件才允许设置已启动

ADDIS — ADC disable command

此位由软件设置为禁用 ADC（ADDIS 命令），并将其置于断电状态(OFF 状态）。一旦 ADC 被有效地禁用，硬件就会清除它（此时 ADEN 也会被硬件清除）。

1 = 写入 1 以禁用 ADC。读取 1 表示正在执行一个 ADDIS 命令。

0 = 没有 ADDIS 命令正在进行中

提示： 只有当 ADEN = 1 和 ADSTART = 0 时，软件才允许设置 ADDIS（这将确保不进行转换）

ADEN — ADC enable command

此位由软件设置为启用 ADC。一旦设置了 ADRDY 标志，ADC 将有效地准备好操作。执行 ADDIS 命令后，禁用 ADC 后，硬件清除。

1 = Write 1 以启用 ADC。

0 = ADC 已禁用（关闭状态）。

提示： 只有当 ADC_CR 寄存器的所有位都为 0（ADSTP = 0、ADSTART = 0、ADDIS = 0 和 ADEN = 0）时，软件才允许设置 ADEN

25.10.2.4. ADC 配置寄存器 1（ADC_CFGR1）

地址：ADC_BASEADDR + 0x0000_000C

	31	30	29	28	27	26	25	24
读:	DIFF	OVRMOD	0			SEQ_LEN[2:0]		
写:								
复位:	0	0	0	0	0	1	1	1

	23	22	21	20	19	18	17	16
读:	DISCEN	AUTOFF	WAIT	CONT	0	TRIGSCR[2:0]		
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0		TRIGMODE[2:0]			ALIGN	RES[1:0]	
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0							DMAEN
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 25-14：ADC 配置寄存器 1（ADC_CFGR1）

Read: Anytime

Write: Before ADC start

DIFF — Select differential-input

此位确定输入是单端输入还是差分输入。

1 = 模拟输入进行差分采样。

0 = 模拟输入为单采样

OVRMOD — Overrun management mode

此位由软件设置和清除，并配置数据溢出的方式。

1 = 当检测到溢出时，ADC_DR 寄存器将被最后一个转换结果覆盖。

0 = 当检测到溢出时，ADC_DR 寄存器将与旧数据一起保存。

SEQ_LEN[2:0] — Sequence length

这些位定义了序列的长度。序列长度为 = 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 — Discontinuous mode

此位通过软件设置和清除，以启用/禁用不连续模式。

1 = 不连续模式已启用

0 = 不连续模式被禁用

AUTOFF — Auto-off mode

此位通过软件设置和清除，以启用/禁用自动关闭模式。

1 = 已启用了自动关闭模式

0 = 自动关闭模式被禁用

WAIT — Wait conversion mode

此位通过软件设置并清除，以启用/禁用等待转换模式。

1 = 等待转换模式开启

0 = 等待转换模式被禁用

CONT — Single / continuous conversion mode

此位元已由软件设置和清除。如果设置，转换持续进行，直到被清除

1 = 连续转换模式

0 = 单次转换模式

TRIGSCR[2:0] — External trigger source

这些位用于选择 8 个可能的事件中哪些可以触发转换，见表 25-3。

TRIGMODE[2:0] — Trigger mode select

这些位用于选择软件触发模式或外部触发极性，见表 25-2。

ALIGN — Data alignment

这个位被软件设置和清除，以选择右或左对齐，见表 25-4。

0 = 右对齐

1 = 左对齐

RES[1:0] — Data resolution

这些位是由软件编写来选择转换的分辨率，见表 26-4。

DMAEN — Direct memory access enable

该位由软件设置和清除，以启用生成 DMA 请求。这允许使用 DMA 控制器来自动管理转换后的数据。

1 = DMA 已启用

0 = DMA 被禁用

25.10.2.5. ADC 配置寄存器 2（ADC_CFGR2）

地址：ADC_BASEADDR + 0x0000_0010

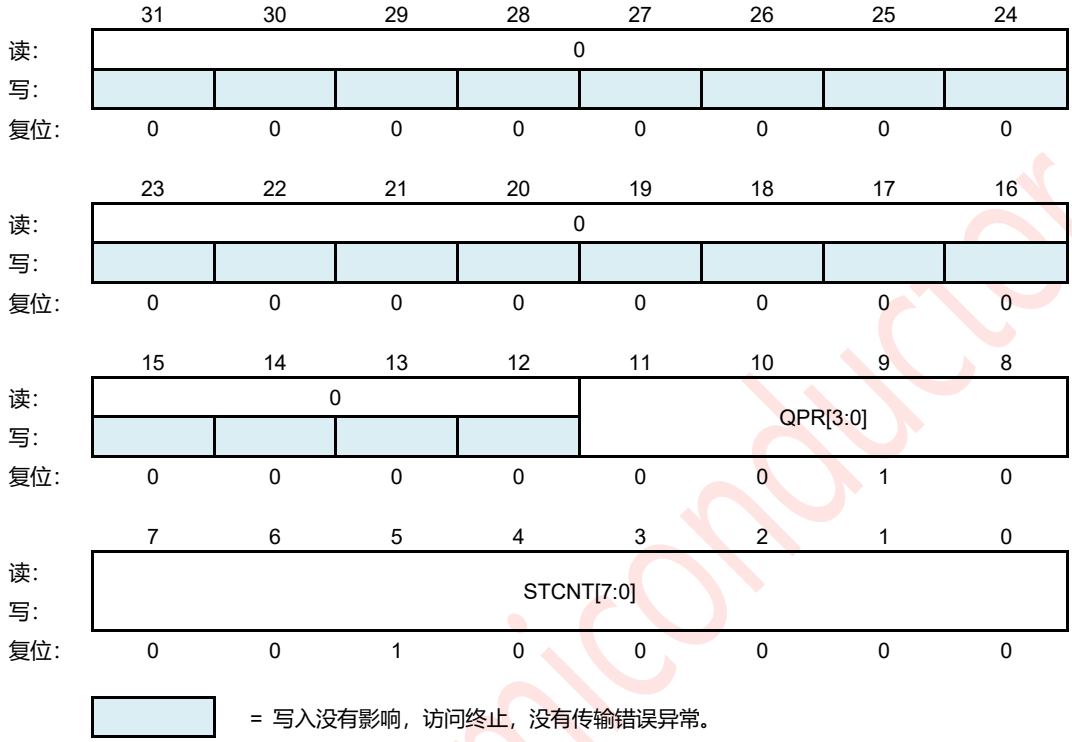


图 25-15：ADC 配置寄存器 2（ADC_CFGR2）

Read: Anytime
Write: Before ADC enable

QPR[3:0] — Prescaler Clock Divider Bits
这些位选择系统时钟除数来生成 QADC 时钟如下：

FQCLK = Fsys_QCLK / (QPR[3:0] + 1)

where:

0 < QPR[3:0] <= 15 and the value 1 is not allowed.

STCNT[7:0] — ADC startup counter bits
ADC 需要 tSTAB（~2us）才能开始精确转换。这个时间是通过计算 QCLK 循环来计算的，直到内部计数器到达 STCNT[7:0]。所以用户应该在 ADC 启用之前设置这些位。例如，如果 QCLK = 为 16MHz，则用户应该设置 STCNT[7:0] = 2000/（1000/16）= 32。

25.10.2.6. ADC 采样时间寄存器（ADC_SMPR）

地址：ADC_BASEADDR + 0x0000_0014

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	SMP[7:0]							
写:								
复位:	0	0	0	0	0	0	1	0

= 写入没有影响，访问终止，没有传输错误异常。

图 25-16：ADC 采样时间寄存器（ADC_SMPR）

Read: Anytime

Write: Before ADC start

SMP[7:0] — Sampling time selection

这些位是由软件编写，以选择适用于所有通道的采样时间。采样时间计算为（SMP[7:0] + 2）QCLKs

例如：SMP[7:0] = 0x2 表示示例时间为 4 个 QCLKs

25.10.2.7. ADC 看门狗寄存器（ADC_WDG）

地址：ADC_BASEADDR + 0x0000_0018

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	AWDEN	AWDSGL	0	0	AWDCH[3:0]			
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 25-17：ADC 看门狗寄存器（ADC_WDG）

Read: Anytime

Write: Before ADC start

AWDEN — Analog watchdog enable
此位元已由软件设置和清除。

1 = 已启用模拟看门狗

0 = 模拟看门器已禁用

AWDSGL — Enable the watchdog on a single channel or on all channels
此位由软件设置和清除，以启用由 AWDCH[4:0]位识别的通道上或所有通道上的模拟看门狗

1 = 模拟看门狗已在单个通道上启用

0 = 在所有通道上都启用了模拟看门狗

AWDCH[3:0] — Analog watchdog channel selection
这些位是由软件设置和清除。它们选择要由模拟看门狗保护的输入通道。

- 0000：由 AWD 监控的 ADC 模拟输入通道 0
- 0001：ADC 模拟输入通道 1
- .....
- .....

- .....
- 0111：ADC 模拟输入通道 7
- 1111：由 AWD 监控的温度传感器
- 其他值：已保留，不能使用

25.10.2.8. ADC 看门狗阈值寄存器（ADC_TR）

地址：ADC_BASEADDR + 0x0000_001C

	31	30	29	28	27	26	25	24
读:	0			HT[12:8]				
写:								
复位:	0	0	0	1	1	1	1	1

	23	22	21	20	19	18	17	16
读:	HT[7:0]							
写:								
复位:	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8
读:				LT[12:8]				
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	LT[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 25-18：ADC 看门狗阈值寄存器（ADC_TR）

Read: Anytime

Write: Before ADC start

HT[12:0] — Analog watchdog higher threshold

这些位是由软件编写的，以定义模拟看门狗的更高的阈值。

LT[12:0] — Analog watchdog lower threshold

这些位是由软件编写来定义模拟看门狗的下阈值。

25.10.2.9. ADC 通道选择寄存器（ADC_CHSELR）

地址：ADC_BASEADDR + 0x0000_002C

	31	30	29	28	27	26	25	24
读:	0				CCW3[3:0]			
写:								
复位:	0	0	0	0	0	0	0	0

	23	19	18	17	16	18	17	16
读:	0				CCW2[3:0]			
写:								
复位:	0	0	0	0	0	0	0	0

	15	11	10	9	8	10	9	8
读:	0				CCW1[3:0]			
写:								
复位:	0	0	0	0	0	0	0	0

	7	3	2	1	0	2	1	0
读:	0				CCW0[3:0]			
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 25-19：ADC 通道选择寄存器 1（ADC_CHSELR1）

地址：ADC_BASEADDR + 0x0000_0030

	31	30	29	28	27	26	25	24
读:	0				CCW7[3:0]			
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0				CCW6[3:0]			
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	0				CCW5[3:0]			
写:								
复位:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读:	0				CCW4[3:0]			
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 25-20：ADC 通道选择寄存器 2（ADC_CHSELR2）

Read: Anytime

Write: Before ADC start

CCWx[3:0] — The number x conversion select channel, refer to **Table 31-1**.

25.10.2.10. ADC FIFO 访问寄存器（ADC_FIFO）

地址：ADC_BASEADDR + 0x0000_004C

	31	30	29	28	27	26	25	24
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	23	22	21	20	19	18	17	16
读:	0							
写:								
复位:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8
读:	DATA[15:8]							
写:								
复位:	0	0	0	0	0	0	0	0

	3	6	5	4	3	2	1	0
读:	DATA[7:0]							
写:								
复位:	0	0	0	0	0	0	0	0

= 写入没有影响，访问终止，没有传输错误异常。

图 25-21：ADC FIFO 访问寄存器（ADC_FIFO）

Read: Anytime

Write: Never

DATA[15:0] — Converted data

Refer to **Table 31-4.**

26. TFT LCD 控制器的寄存器

TFT LCD 显示控制器由内部 32-bits MCU 所控制，但是其有保留 2 个 PWM 及数个 IO 口让使用者运用，作为 MCU 接口的延伸，通常是 LT7589 做二次开发时才会用到，使用者需要透过这些寄存器才能控制这些接口。寄存器读写的方式可以参考 乐升半导体 提供的程序代码。

26.1. PWM 控制寄存器

REG[84h] PWM Prescaler Register（PSCLR）

Bit	说　明	默认值	存取模式
7-0	**PWM Prescaler Register** 此寄存器为 Timer-0 及 Timer-1 的 Prescaler 值。基频是： **Core_Freq / （Prescaler + 1）**	0	RW

REG[85h] PWM Clock Mux Register（PMUXR）

Bit	说　明	默认值	存取模式
7-6	**PWM Timer-1 除频器设定（Select 2nd Clock Divider’s MUX Input for PWM Timer-1）** 00b = 1。 01b = 1/2。 10b = 1/4。 11b = 1/8。	0	RW
5-4	**PWM Timer-0 除频器设定（Select 2nd Clock Divider’s MUX Input for PWM Timer-0）** 00b = 1。 01b = 1/2。 10b = 1/4。 11b = 1/8。	0	RW
3-2	**PWM-1 功能设定（PWM[1] Function Control）** 0xb：PWM[1] 输出系统错误旗标（Scan FIFO pop 错误或是内存存取超过范围）。 10b：PWM[1] 输出 PWM 计数器 1 的波形或是 PWM 计数器 0 的反相波形（dead zone 使能）。 11b：PWM[1] 输出 Oscillator 晶振频率（OSC）。 如果 TEST[0] 为 High，则 PWM[1] 将会是屏幕扫描频率的输入。	0	RW
1-0	**PWM-0 功能设定（PWM[0] Function Control）** 0xb：PWM[0] 为 GPIO-C[7]。 10b：PWM[0] 输出 PWM 计数器 0。 11b：PWM[0] 输出系统频率。	0	RW

REG[86h] PWM Configuration Register（PCFGR）

Bit	说　明	默认值	存取模式
7	未使用 -- 必须保持在 0	0	RW
6	**PWM1 输出准位控制（PWM Timer-1 Output Inverter On/Off)** PWM1 的输出是否反相。 0：反相关闭。 1：PWM1 反相开启。	0	RW
5	**Timer-1 自动重载控制（PWM Timer-1 Auto Reload On/Off)** Timer-1 的自动重载开启与关闭。 0：单击模式（One-Shot）。 1：自动重载模式。	1	RW
4	**Timer-1 计数器开始与停止（PWM Timer-1 Start/Stop)** 0：停止。 1：开始。 在自动重载模式，MCU 若要停止 PWM 计数器，必须写 0。在单击模式中，这个 bit 会自动被清除。MCU 可以读取这个 bit，以便得知 PWM 是执行中还是停止中。	0	RW
3	**Timer-0 死区控制（PWM Timer-0 Dead Zone Enable)** 0：禁止。 1：使能。	0	RW
2	**PWM0 输出准位控制（PWM Timer-0 Output Inverter On/Off)** PWM0 的输出是否反相。 0：反相关闭。 1：PWM0 反相开启。	0	RW
1	**Timer-0 自动重载控制（PWM Timer-0 Auto Reload On/Off)** Timer-0 的自动重载开启与关闭。 0：单击模式（One-Shot）。 1：自动重载模式。	1	RW
0	**Timer-0 计数器开始与停止（PWM Timer-0 Start/Stop)** 0：停止。 1：开始。 在自动重载模式，MCU 若要停止 PWM 计数器，必须写 0。在单击模式中，这个 bit 会自动被清除。MCU 可以读取这个 bit，以便得知 PWM 是执行中还是停止中。	0	RW

REG[87h] Timer-0 Dead Zone Length Register [DZ_LENGTH]

Bit	说　明	默认值	存取模式
7-0	**Timer-0 死区长度寄存器（Timer-0 Dead Zone Length Register)** 此寄存器为 Dead Zone 的长度，以计数器 0 的计数完整的一个周期为 Dead Zone 的一个单位时间长度。	0	RW

REG[89h-88h] Timer-0 Compare Buffer Register [TCMPB0]

Bit	说　明	默认值	存取模式
15-0	**Timer-0 计数比较寄存器 (Timer-0 compare Buffer Register)** REG[89h] 对应到 TCMPB0 [15:8]。 REG[88h] 对应到 TCMPB0 [7:0]。 Timer-0 计数比较寄存器总共是 16-bits，当计数器等于或小于此寄存器的值，并且在 PWM 计数器 0 反相关闭情况下，PWM0 输出为 High。	0	RW

REG[8Bh-8Ah] Timer-0 Count Buffer Register [TCNTB0]

Bit	说　明	默认值	存取模式
15-0	**Timer-0 计数寄存器（Timer-0 Count Buffer Register [15:0])** REG[8Bh] 对应到 TCNTB0 [15:8]。 REG[8Ah] 对应到 TCNTB0 [7:0]。 Timer-0 计数寄存器总共有 16-bit。当计数器等于 0 时，并且 Reload_EN 是使能的情况下，PWM 会重载此寄存器的值到计数器中。当 PWM 开始计数后，可以通过这个寄存器读回目前的计数值。	0	RW

REG[8Dh-8Ch] Timer-1 Compare Buffer Register [TCMPB1]

Bit	说　明	默认值	存取模式
15-0	**Timer-1 计数比较寄存器 (Timer-1 compare Buffer Register)** REG[8Dh] 对应到 TCMPB1 [15:8]。 REG[8Ch] 对应到 TCMPB1 [7:0]。 Timer-1 计数比较寄存器总共是 16-bits，当计数器等于或小于此寄存器的值，并且在 PWM 计数器 1 反相关闭情况下，PWM1 输出为 High。	0	RW

REG[8Fh-8Eh] Timer-1 Count Buffer Register [TCNTB1]

Bit	说　明	默认值	存取模式
15-0	**Timer-1 计数寄存器（Timer-1 Count Buffer Register [15:0])** REG[8Fh] 对应到 TCNTB1 [15:8]。 REG[8Eh] 对应到 TCNTB1 [7:0]。 Timer-1 计数寄存器总共有 16-bit。当计数器等于 0 时，并且 Reload_EN 是使能的情况下，PWM 会重载此寄存器的值到计数器中。当 PWM 开始计数后，可以通过这个寄存器读回目前的计数值。	0	RW

26.2. GPIO 寄存器

REG[F0h] LCD_IOA Direction（LCD_IOADIR)

Bit	说　明	默认值	存取模式
7-2	**LCD_IOA[7:2] 输出/输入控制（LCD_IOA In/Out Control)** 0：输出。 1：输入。	FFh	RW
1-0	保留	--	--

REG[F1h] LCD_IOA（LCD_IOA)

Bit	说　明	默认值	存取模式
7-2	**LCD_IOA[7:2] 数据（LCD_IOA Data)** Write：设定 LCD_IOA 的输出数据。 Read：由 LCD_IOA 读取输入数据。 LCD_IOA[77:2] 为通用型 I/O，这些引脚只有 LT7589B 才可以使用。	NA	RW
1-0	保留	--	--

REG[F3h] GPIO-C Direction（GPIOCD)

Bit	说　明	默认值	存取模式
7	**GPIOC[7] 输出/输入控制（GPIOC[7] In/Out Control)** 0：输出。 1：输入。	FFh	RW
6-0	保留		

REG[F4h] GPIO-C（GPIOC）

Bit	说　明	默认值	存取模式
7	**GPIOC[7] 数据（GPIOC[7] Data）** Write：设定 GPIOC[7] 的输出数据。 Read：由 GPIOC[7] 读取输入数据。 提示：GPIOC[7] 的输出数据与 PWM[0] 共享引脚。 GPIOC 功能只有在 TFT LCD 控制器的 PWM 与 SPI Master 的功能被禁止时才能使用。	NA	RW
6-0	保留	NA	RW

REG[F5h] GPIO-D Direction（GPIODD）

Bit	说　明	默认值	存取模式
7-0	**GPIO D 组输出/输入控制（GPIOD In/Out Control）** 0：输出。 1：输入。	FFh	RW

REG[F6h] GPIO-D（GPIOD）

Bit	说　明	默认值	存取模式
7-0	**GPIO-D 组数据（GPIOD Data）** Write：设定 GPIOD[7:0] 的输出数据。 Read：由 GPIOD[7:0] 读取输入数据。 GPIOD[7:0] 与 PD[18, 2, 17, 16, 9, 8, 1, 0] 共享引脚，GPIOD 只有在 TFT LCD 屏幕数据总线设成 16-bits（RGB：565）时才能使用，GPIOD[7,6,3,2] 则只有在 LCD 屏幕数据总线设成 16-bits 时才能使用。	NA	RW

27. 电气特性

27.1. 极限参数

表 27-1：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	-0.3 ~ 4.0	V
V_{IN}	逻辑输入电压	-0.3 ~ VDD33+0.3	V
V_{OUT}	逻辑输出电压	-0.3 ~ VDD33+0.3	V
P_D	最大功耗	≦300	mW
T_{OPR}	工作温度范围（@120MHz)	-40 ~ 85	℃
T_{JT}	工作结温范围	-40 ~ 105	℃
T_{ST}	储存温度范围	-40 ~ 125	℃
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。

27.2. DC 电气参数

表 27-2：DC 电气参数表

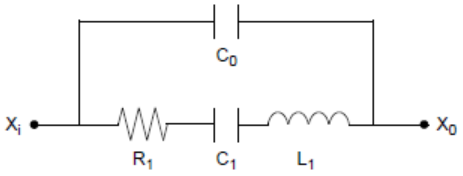
符 号	参 数 描 述	条 件	最小值	典型值	最大值	单位
V_{DD33} , V_{DD33_IO}	工作电压		3.0	3.3	3.6	V
C_{VDD}	负载电容		1	-	10	uF
I_{OPR}	工作电流	提示 1		60		mA
I_{STB}	待机电流	提示 1		30		mA
I_{SUSP}	休眠电流	提示 1		10		mA
I_{SLP}	睡眠电流	提示 1		7		mA
T_{RMP}	电源上升时间	VDD Ramp Up to 3.3 V			35	ms
振荡时钟与 PLL						
F_{OSC}	晶振（OSC）频率	V_{DD33} = 3.3 V，提示 2		12		MHz
F_{VCO}	VCO 输出频率		100		500	MHz
T_{LOCK}	Lock Time	提示 3			500	us
CLK_{MPLL}	MPLL 输出频率（MCLK）	V_{DD33} = 3.3 V			133	MHz
CLK_{CPLL}	CPLL 输出频率（CCLK）	V_{DD33} = 3.3 V			100	MHz

符 号	参 数 描 述	条 件	最小值	典型值	最大值	单位
CLK_{PPLL}	PPLL 输出频率（PCLK）	V_{DD33} = 3.3 V			80	MHz
串口 MCU 界面						
CLK_{SPI}	SPI 输入频率				50	MHz
其他输出输入界面 (CMOS 3-State Output pad with Schmitt Trigger Input, Pull-Up/Down)						
V_{IH}	输入高电位		2		3.6	V
V_{IL}	输入低电位		-0.3		0.8	V
V_{OH}	输出高电位		2.4			V
V_{OL}	输出低电位				0.4	V
R_{PU}	上拉电阻		33	41	62	KΩ
R_{PD}	下拉电阻		33	42	68	KΩ
V_{TP}	施密特触发由低到高的阈值		1.5		2.1	V
V_{TN}	施密特触发由高到低的阈值		0.8		1.3	V
V_{HVS}	迟滞电压		200			mV
I_{LEAK}	输入漏电流		-10		+10	μA
V_{SLEW}	电压上升/下降斜率			1.5		V/ns

(条件：V_{DD} = 3.3V，T_A = 25 ℃)

提示 1：在无额外负载情况下，以串口 SPI 接口测试。

提示 2：使用晶振时的寄生电容效应。



标准值：R1 = 50Ω（25-100Ω），L1 = 3.4mH，C1 = 13fF，C0 = 2.8pF

图 27-1：晶振等效电路图

提示 3：从电源启动到内部 PLL 有稳定的时钟输出时所需要的时间。

表 27-3：电源特性

项目	符号	最小值	典型值	最大值	单位
芯片电源	VDD33A	2.97	3.3	3.63	V
	VDD33B	2.97	3.3	3.63	V
	VDD33_IO	2.97	3.3	3.63	V
ADC 工作电压	AVDD	2.97	3.3	3.63	V
MCU 内核工作电压 (LDO O/P)	VDD12	1.1	1.2	1.3	V
LCD 控制器内核工作电压 (LDO O/P)	LCD_V12	1.1	1.2	1.3	V
RTC 工作电压	VBAT	2.7	3.3	3.6	V

表 27-4：热阻参数 (Thermal Characteristics)

符 号	参 数 描 述	参 数 范 围	单 位
$R_{\Theta}JC$	热阻：Junction to Case	3 ~ 8	℃/W
$R_{\Theta}JA$	热阻：Junction to Ambient	20 ~25	℃/W

27.3. ESD 保护规格

表 27-5：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	800	V	ANSI/ESDA/JEDEC JS-002-2022
Latch Up	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. 封装信息

28.1. LT7589A (QFN-96pin)

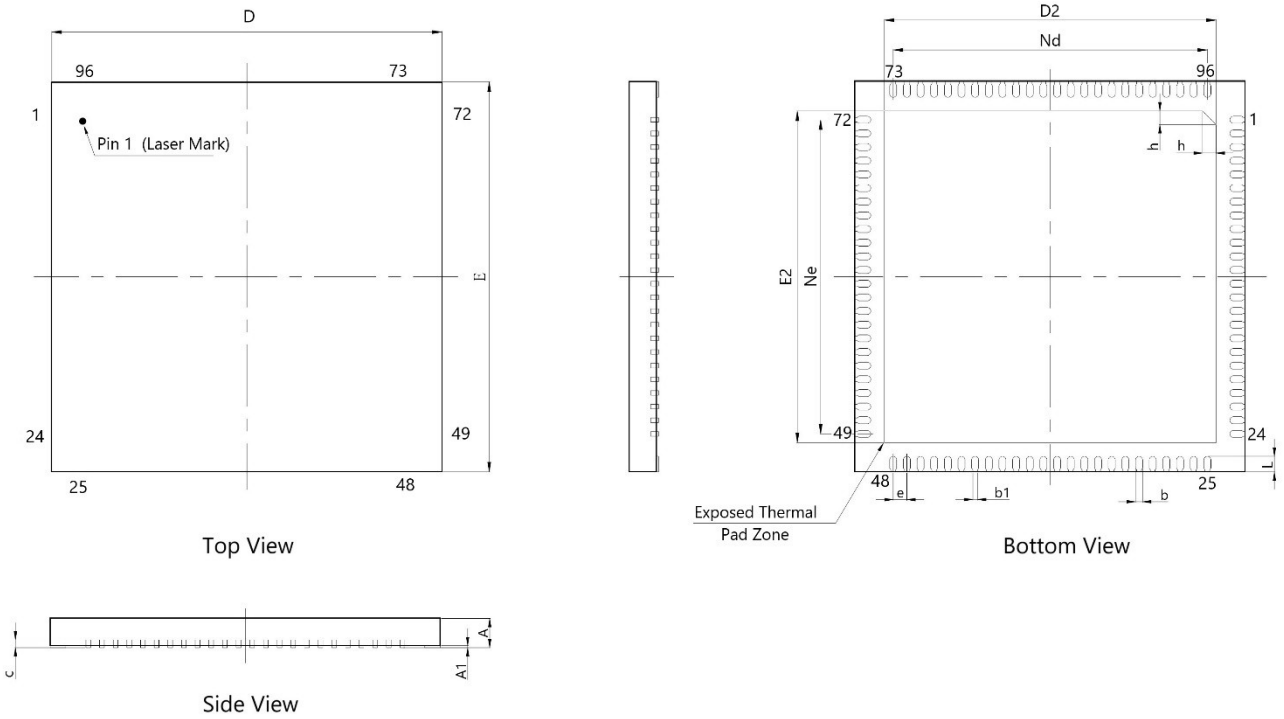


图 28-1：QFN-96Pin 外观尺寸图

提示：PCB 布局时，LT7589A 背部的散热焊盘（Thermal Pad Zone）必须直接接地。

表 28-1：QFN-96Pin 尺寸参数

Symbol	Millimeter			Symbol	Millimeter		
	Min.	Nom.	Max		Min.	Nom.	Max
A	0.80	0.85~1.10	1.15	E	9.90	10.00	10.10
A1	-	0.02	0.05	Ne	8.05BSC		
b	0.13	0.18	0.23	L	0.35	0.40	0.45
b1	0.12REF			E2	8.40	8.50	8.60
c	0.18	0.20	0.25	h	0.30	0.35	0.40
D	9.90	10.00	10.10	Nd	8.05BSC		
D2	8.40	8.50	8.60				
e	0.35BSC						

28.2. LT7589B (LQFP-128pin)

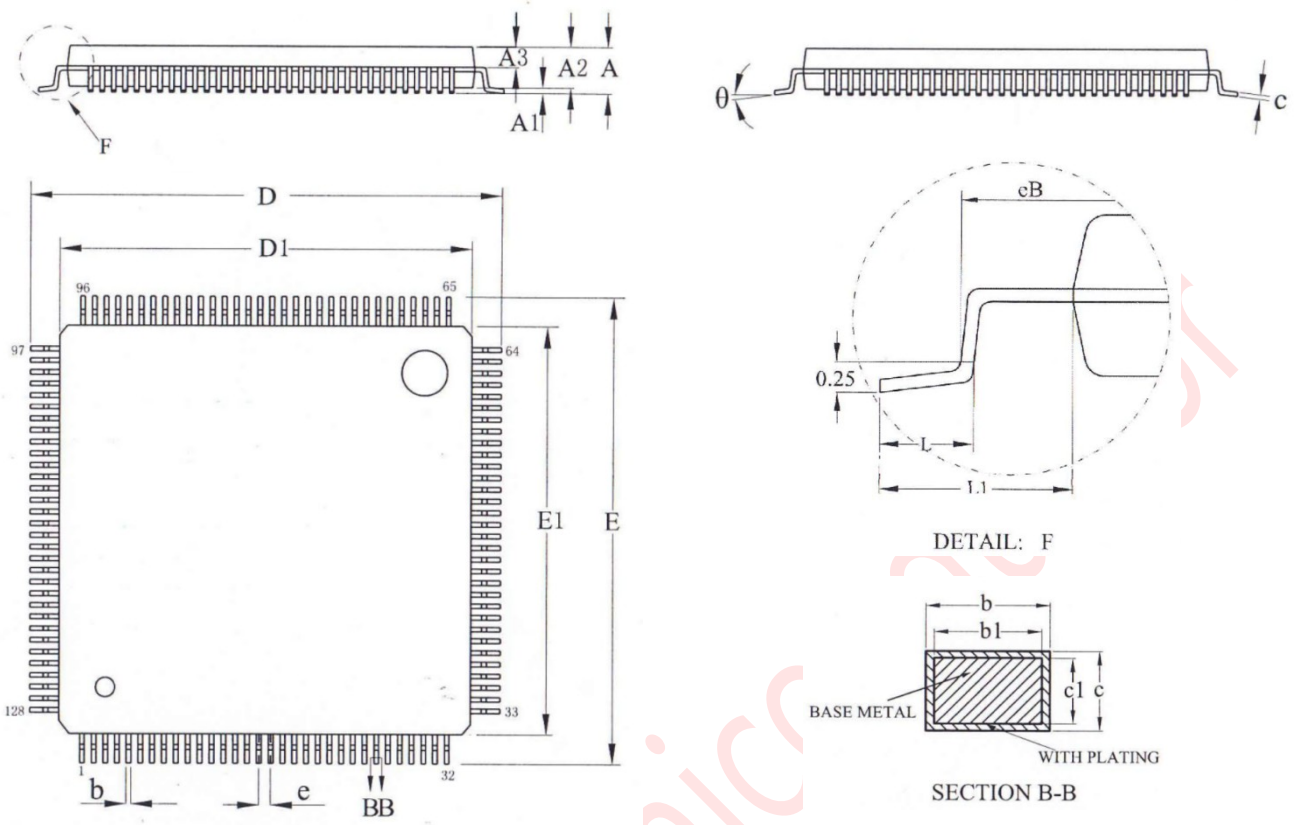


图 28-2：LQFP-128Pin 外观尺寸图

表 28-2：LQFP-128Pin 尺寸参数

Symbol	Millimeter			Symbol	Millimeter		
	Min.	Nom.	Max		Min.	Nom.	Max
A	-	-	1.60	**D1**	13.9	14.0	14.1
A1	0.05	-	0.15	**E**	15.8	16.0	16.2
A2	1.35	1.40	1.45	**E1**	13.9	14.0	14.1
A3	0.59	0.64	0.69	**eB**	15.05	-	15.35
b	0.14	-	0.22	**e**	0.40BSC		
b1	0.13	0.16	0.19	**L**	0.45	-	0.75
c	0.13	-	0.17	**L1**	1.00REF		
c1	0.12	0.13	0.14	**θ**	0		7
D	15.8	16.00	16.2				

28.3. LT7589A PCB 板布局建议

T7589A 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 LT7589A 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 LT7589A 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 2 个 LT7589A 范例，中间浅黄色区是 LT7589A 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。

提示：其他 PCB 板布局注意事项请务必参考乐升半导体官网的“LT7589_PCB_EMI_CH_Vxx.pdf”文件说明。

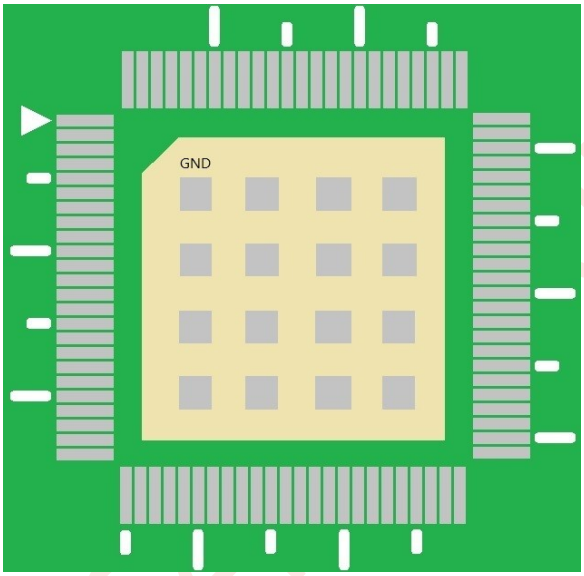


图 28-3：LT7589A 底部焊盘 PCB 的设计建议-1

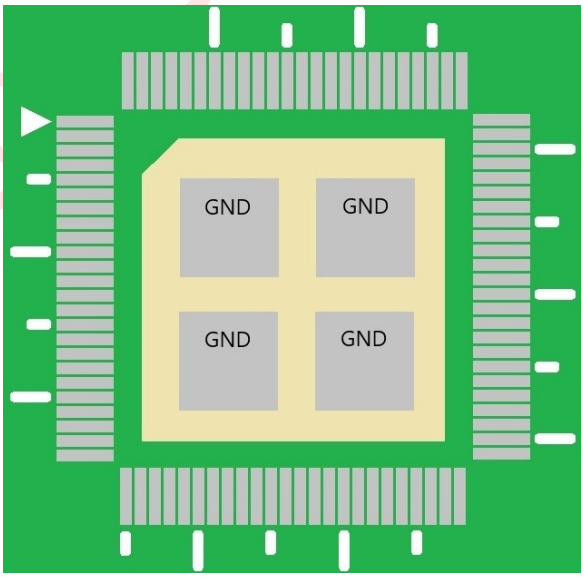


图 28-4：LT7589A 底部焊盘 PCB 的设计建议-2

29. 原理图

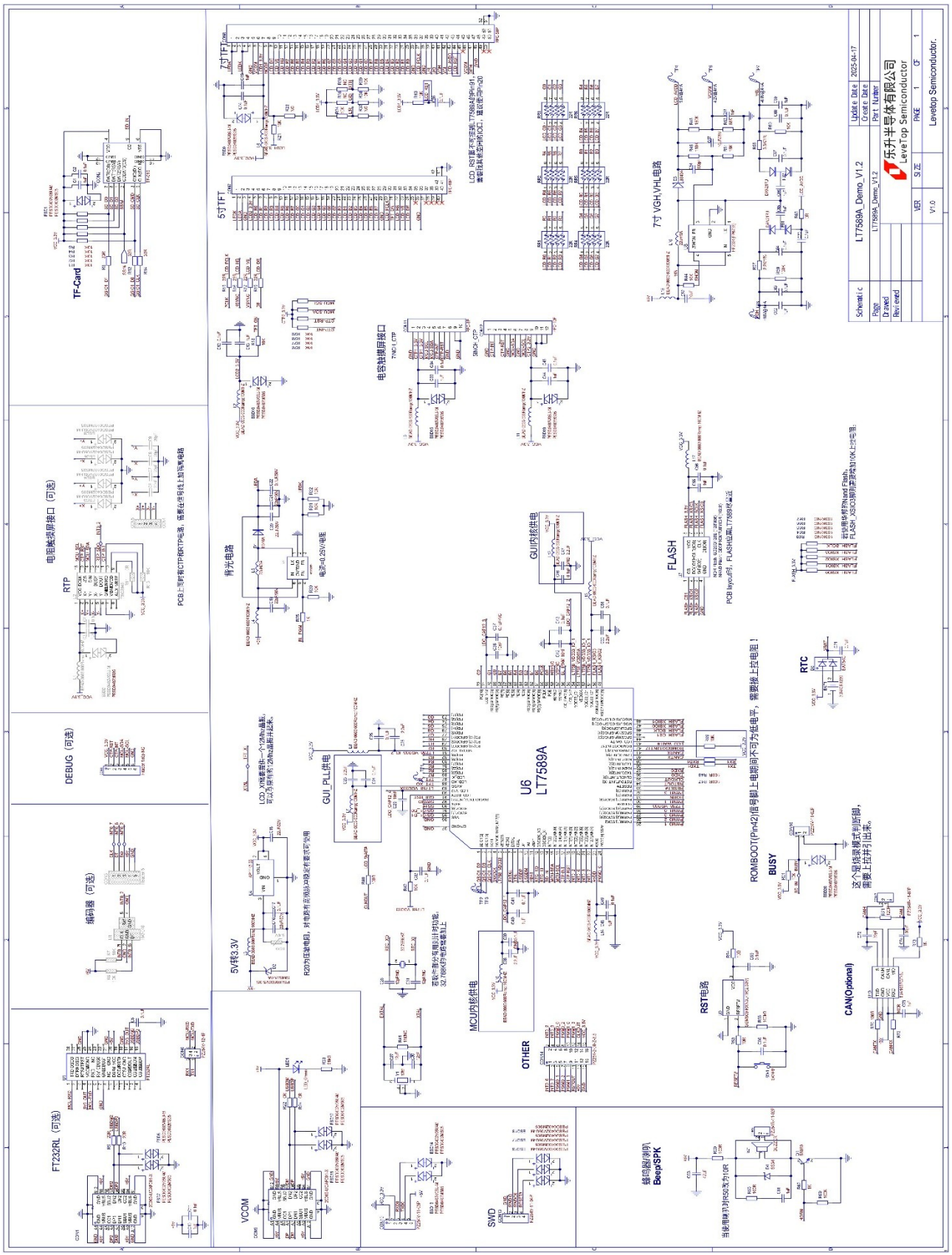


图 29-1：LT7589A 参考原理图

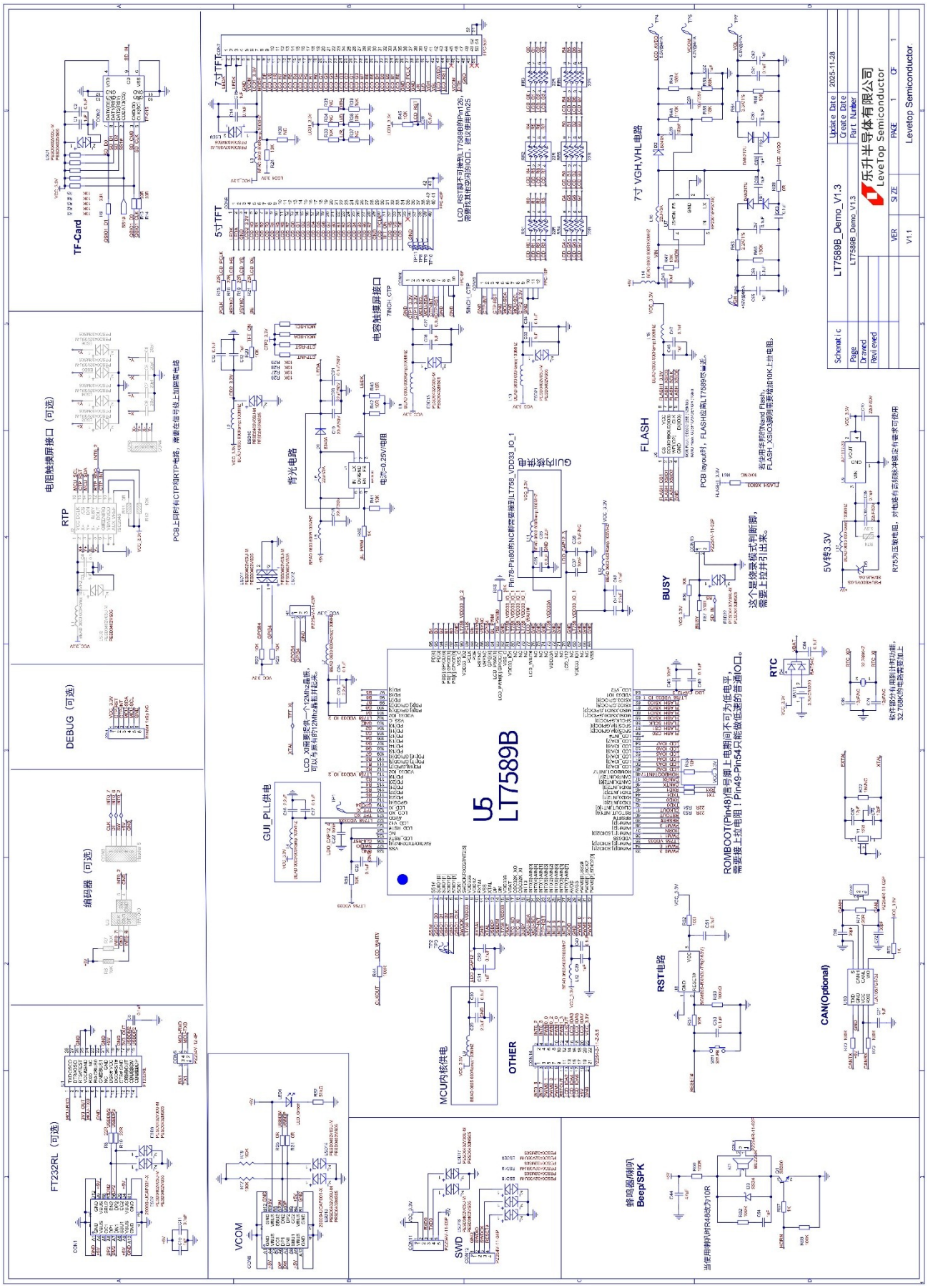


图 29-2：LT7589B 参考原理图