Gowin APB SPI IP

用户指南

IPUG1216-1.0, 2025/04/30

免责声明

本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止反言或其它方式授予任何知识产权许可。除高云半导体在其产品的销售条款和条件中声明的责任之外，高云半导体概不承担任何法律或非法律责任。高云半导体对高云半导体产品的销售和／或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。高云半导体对文档中包含的文字、图片及其它内容的准确性和完整性不承担任何法律或非法律责任，高云半导体保留修改文档中任何内容的权利，恕不另行通知。高云半导体不承诺对这些文档进行适时的更新。

版本信息

日期	版本	说明
2025/04/30	1.0	初始版本。

目录

图目录

表目录

1 关于本手册

1.1 手册内容

Gowin APB SPI IP 用户指南主要包含功能特性、结构框图、功能描述、信号描述、参数描述、界面配置、编程模型、参考设计等内容，旨在帮助用户快速了解 APB SPI IP 的特性和使用方法。本手册中的软件界面截图参考的是 1.9.11.02 (64-bit) 版本，因软件版本升级，部分信息可能会略有差异，具体以用户软件版本的信息为准。

1.2 相关文档

通过登录高云半导体网站 www.gowinsemi.com 可以下载、查看以下相关文档：

- DS100，GW1N 系列 FPGA 产品数据手册
- DS117，GW1NR 系列 FPGA 产品数据手册
- DS821，GW1NS 系列 FPGA 产品数据手册
- DS861，GW1NSR 系列 FPGA 产品数据手册
- DS841，GW1NZ 系列 FPGA 产品数据手册
- DS961，GW2ANR 系列 FPGA 产品数据手册
- DS102，GW2A 系列 FPGA 产品数据手册
- DS226，GW2AR 系列 FPGA 产品数据手册
- DS971，GW2AN-18X &9X 器件数据手册
- DS976，GW2AN-55 器件数据手册
- DS981，GW5AT 系列 FPGA 产品数据手册
- DS1103，GW5A 系列 FPGA 产品数据手册
- DS1239，GW5AST 系列 FPGA 产品数据手册
- DS1105，GW5AS 系列 FPGA 产品数据手册
- DS1108, GW5AR 系列 FPGA 产品数据手册

- DS1118，GW5ART 系列 FPGA 产品数据手册
- SUG100，Gowin 云源软件用户指南

1.3 术语、缩略语

本手册中出现的相关术语、缩略语及相关释义如表 1-1 所示。

表 1-1 术语、缩略语

术语、缩略语	全称	含义
AHB	Advanced High Performance Bus	高性能总线
AMBA	Advanced Microcontroller Bus Architecture	高级微控制器总线架构
APB	Advanced Peripheral Bus	高级外围总线
DMA	Direct Memory Access	直接内存访问
FIFO	First In and First Out	先进先出队列
FPGA	Field Programmable Gate Array	现场可编程门阵列
IP	Intellectual Property	知识产权
LSB	Least Significant Bit	最低有效位
MSB	Most Significant Bit	最高有效位
SPI	Serial Peripheral Interface	串行外设接口

1.4 技术支持与反馈

高云半导体提供全方位技术支持，在使用过程中如有任何疑问，可直接与公司联系：

网址：www.gowinsemi.com.cn

E-mail：support@gowinsemi.com

Tel：+86 755 8262 0391

2 概述

2.1 介绍

Gowin APB SPI IP 是一个 SPI 控制器，可以用作 SPI 主设备或 SPI 从设备。作为 SPI 主设备，控制器用于连接各种 SPI 设备；而作为 SPI 从设备，控制器则响应主设备发出的数据交换请求。

表 2-1 Gowin APB SPI IP 概述

Gowin APB SPI IP	
逻辑资源	参见表 2-3
交付文件	
设计文件	Verilog(encrypted)
参考设计	Verilog
TestBench	Verilog
测试设计流程	
综合软件	GowinSynthesis
应用软件	Gowin Software（V1.9.11.02 及以上）

注！

可登录高云半导体网站查看芯片支持信息。

2.2 功能特性

Gowin APB SPI IP 主要包含以下功能特性：

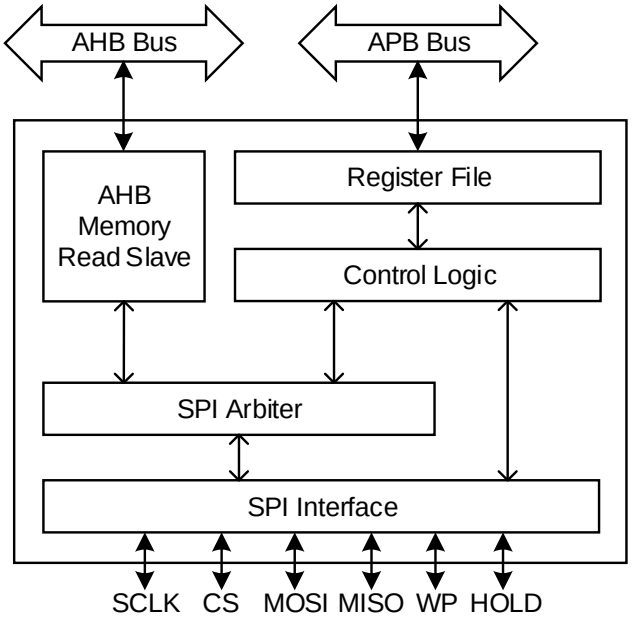
- 符合 AMBA 2 AHB 协议规范
- 符合 AMBA 3 APB 协议规范
- 支持 MSB/LSB 传输
- 支持 DMA 数据传输
- 支持可编程 SPI 时钟 SCLK
- 支持通过 AHB 总线实现只读内存映射访问
- 支持 SPI 从模式

- 支持可配置的 Dual I/O 和 Quad I/O SPI 接口
- 支持可配置的 TX/RX FIFO 深度（可选值：2、4、8、16、32、64 或 128）
- 支持 32 位 TX/RX FIFO 宽度
- 支持在 AHB/APB 接口上配置编程端口位置

2.3 结构框图

Gowin APB SPI IP 结构框图如图 2-1 所示。

图 2-1 结构框图



2.4 功能描述

Gowin APB SPI 控制器的功能可以划分为四个子部分，分别为主模式、从模式、Dual I/O 模式和 Quad I/O 模式。其中，主模式和从模式部分描述了传输启动机制和 SPI 帧格式；而 Dual I/O 模式和 Quad I/O 模式部分则介绍了相对于常规模式的另外两种传输格式。

2.4.1 主模式

Gowin APB SPI 控制器可以作为 SPI 主设备，在 SPI 总线上发起 SPI 传输。SPI 传输格式和接口时序可通过 APB 编程端口进行编程配置。SPI 传输可以通过 AHB 总线上的内存映射读访问或通过寄存器编程来启动。

对于通过内存映射 AHB 读事务发起的 SPI 传输，读事务会根据 SPI 内存访问控制寄存器（0x50）中设定的读命令格式，转换为 SPI 接口传输。该寄存器的默认复位值可通过硬件配置，以支持从 SPI ROM 启动。内存映射地址空间的大小，在 AHB 总线上可达 16MB。

从内存映射接口接收到的 AHB 地址会直接传递给 SPI 从设备，因为 APB SPI 无法得知从设备的地址和大小。如果发送给从设备的地址需要加

偏移，则应将相应的 AHB 地址位直接接到预定的值。

对于通过寄存器编程发起的 SPI 传输，包括 TX/RX 数据传输的命令、地址和数据阶段。控制器提供专用寄存器以指定命令、地址和数据字段的内容，其中 SPI 数据寄存器（0x2C）同时用于 TX 和 RX 数据传输。数据传输可通过编程 IO（PIO）或 DMA 来启动。

Gowin APB SPI 控制器提供 TX/RX FIFO 阈值中断，以便在 PIO 编程时简化流控。控制器还具备一个可编程位，可在传输完成时发出中断。

即使没有明确引用 hburst 信号，APB SPI 控制器也支持 AHB 内存端口上的所有突发传输类型。这是因为控制器对突发操作中的每次传输均进行独立处理，每次传输一个字（即一次传输）。

除了支持的传输格式外，SPI 控制器还允许直接控制 SPI 接口上的信号，从而能够与需要特殊传输格式的 SPI 设备进行通信。有关更多信息，请参见 SPI 直接 IO 控制寄存器（0x14）。

2.4.2 从模式

Gowin APB SPI 控制器也可以作为 SPI 从设备，并接受如表 2-2 所示的常用命令。此外，该控制器还支持用户自定义命令，其中从设备数据字段的格式由 SPI 传输控制寄存器（0x20）定义。

Gowin APB SPI 从设备将 SPI I/O 接口上的数据包解释为三个字段：从设备命令字段、从设备虚拟字段和从设备数据字段。从设备命令字段和从设备虚拟字段始终为 8 位长，而从设备数据字段的格式则由接收到的命令和 SPI 传输控制寄存器（0x20）共同决定。

表 2-2 从模式下所支持的命令

Slave Command Name	OP Code	Slave Data
Read Status Single IO	0x05	32-bit Status
Read Status Dual IO	0x15	32-bit Status
Read Status Quad IO	0x25	32-bit Status
Read Data Single IO	0x0B	Replay data from the Data Register in the FIFO manner
Read Data Dual IO	0x0C	Replay data from the Data Register in the FIFO manner
Read Data Quad IO	0x0E	Replay data from the Data Register in the FIFO manner
Write Data Single IO	0x51	Data saved to the Data Register in the FIFO manner
Write Data Dual IO	0x52	Data saved to the Data Register in the FIFO manner
Write Data Quad IO	0x54	Data saved to the Data Register in the FIFO manner
User-defined	Any 8-bit numbers other than the listed OP Codes	Depending on the Transfer Controller Register

对于状态读取命令，从设备返回 SPI 从设备状态寄存器（0x60）的

值。

对于用户自定义命令，从设备数据字段的格式由 SPI 传输控制寄存器（0x20）定义。例如，如果传输模式为{Dummy, Write}，则仅写字段会被记录到 SPI 数据寄存器（0x2C）中，而虚拟字段将被丢弃。

2.4.3 Dual I/O 模式

Dual I/O 模式通过将主设备输入/从设备输出（MISO）和主设备输出/从设备输入（MOSI）信号视为双向传输线，实现了 SPI 带宽的翻倍。

在 Dual I/O 模式下，SPI 控制器提供两种传输格式：其中一种格式中，地址阶段和数据阶段均利用两条传输线（MISO 和 MOSI）；而在另一种格式中，仅数据阶段利用两条传输线。

有关更多信息，请参见 SPI 传输格式寄存器（0x10）中的 AddrFmt 和 DualQuad 位。

2.4.4 Qual I/O 模式

Quad I/O 模式通过将主设备输入/从设备输出（MISO）、主设备输出/从设备输入（MOSI）、写保护（WP）以及 HOLD 信号视为双向传输线，使 SPI 带宽达到原来的四倍。

在 Quad I/O 模式下，SPI 控制器提供两种传输格式：在一种格式中，地址阶段和数据阶段均利用这四根传输线（MISO、MOSI、WP 和 HOLD）；在另一种格式中，仅数据阶段利用这四根传输线。

有关更多信息，请参见 SPI 传输格式寄存器（0x10）中的 AddrFmt 和 DualQuad 位。

2.5 资源使用

通过 Verilog 语言实现 Gowin APB SPI IP。因使用器件的密度、速度和等级不同，其性能和资源使用情况可能不同。以高云 GW5AT 系列 FPGA 为例，Gowin APB SPI IP 资源使用情况如表 2-3 所示。

表 2-3 资源使用情况

器件系列	资源	资源使用	配置
GW5AT	Logic	1716	Memory Map Support Memory Map Address Offset: 600000 Memory Map Read CMD Default: 0 Address Width: 32Bit Interface Mode: Quad IO Slave Mode Support Direct IO Control SPI CS2CLK Default: 0 SPI CSHT Default: 2 SPI SCLKDIV Default: 1 TX FIFO Depth: 4Word RX FIFO Depth: 4Word Enable DMA
	Register	821	
	BSRAM	0	

3 信号描述

Gowin APB SPI IP 的信号描述如表 3-1 所示。

表 3-1 信号描述

Signal Name	I/O	Width	Description
hclk	input	1	AHB clock
hresetn	input	1	AHB reset signal (Active-Low)
haddr_mem	input	[N[1]-1]	AHB address bus N is 23 if AHB address width is 24-bit. It is 31 if the AHB address width is 32-bit. Unused most significant bits must be padded with zero, otherwise undetermined behavior may occur.
hrdata_mem	output	[31:0]	AHB read data bus
hreadyin_mem	input	1	AHB bus ready
hreadyout_mem	output	1	AHB slave transfer done
hresp_mem	output	[1:0]	AHB transfer response
hsel_mem	input	1	AHB slave select
htrans_mem	input	[1:0]	AHB transfer type
hwrite_mem	input	1	AHB transfer direction
pclk	input	1	APB clock (must be synchronous with the AHB clock)
presetn	input	1	APB reset signal (Active-Low)
paddr	input	[31:0]	APB address bus
penable	input	1	APB enable signal
prdata	output	[31:0]	APB read data
pready	output	1	APB slave transfer done
psel	input	1	APB select signal
pwdata	input	[31:0]	APB write data bus
pwrite	input	1	APB write signal
spi_default_as_slave	input	1	Set the APB SPI controller as a slave after reset 0x0: Default as a master

Signal Name	I/O	Width	Description
			0x1: Default as a slave
spi_cs_n_in	input	1	Status of the SPI CS (chip select) signal
spi_hold_n_in	input	1	Status of the SPI Flash hold signal
spi_hold_n_oe	output	1	Output enable for the SPI Flash hold signal
spi_hold_n_out	output	1	Output value for the SPI Flash hold signal
spi_wp_n_in	input	1	Status of the SPI Flash write protect signal
spi_wp_n_oe	output	1	Output enable for the SPI Flash write protect signal
spi_wp_n_out	output	1	Output value for the SPI Flash write protect signal
apb2ahb_clken	input	1	APB to AHB clock enable is an AHB domain signal indicating the valid AHB clock cycles to sample and update APB domain signals when the AHB frequency is a multiple of the APB frequency.
spi_rx_dma_ack	input	1	RX FIFO DMA acknowledge
spi_rx_dma_req	output	1	RX FIFO DMA request
spi_tx_dma_ack	input	1	TX FIFO DMA acknowledge
spi_tx_dma_req	output	1	TX FIFO DMA request
spi_clock	input	1	The clock source for the SPI interface. The SCLK signal of the SPI interface is generated by dividing this clock source with a programmable value.
spi_rstn	input	1	The reset signal for the spi_clock domain (Active-Low)
spi_boot_intr	output	1	The SPI controller interrupt
spi_default_mode3	input	1	Select the default SPI mode 0x0: Default values of both CPOL and CPHA are 0x0 0x1: Default values of both CPOL and CPHA are 0x1
scan_enable	input	1	Scan enable (Active-High during ATPG scan/shift phase)
scan_test	input	1	Scan test mode (Active-High during ATPG test)
spi_clk_in	input	1	Status of the SPI SCLK signal
spi_clk_oe	output	1	Output enable for the SPI SCLK signal
spi_clk_out	output	1	Output value for the SPI SCLK signal
spi_cs_n_oe	output	1	Output enable for the SPI CS signal
spi_cs_n_out	output	1	Output value for the SPI CS signal

Signal Name	I/O	Width	Description
spi_miso_in	input	1	Status of the SPI master-input/slave-output (MISO) signal
spi_miso_oe	output	1	Output enable for the SPI MISO signal
spi_miso_out	output	1	Output value for the SPI MISO signal
spi_mosi_in	input	1	Status of the SPI master-output/slave-input (MOSI) signal
spi_mosi_oe	output	1	Output enable for the SPI MOSI signal
spi_mosi_out	output	1	Output value for the SPI MOSI signal

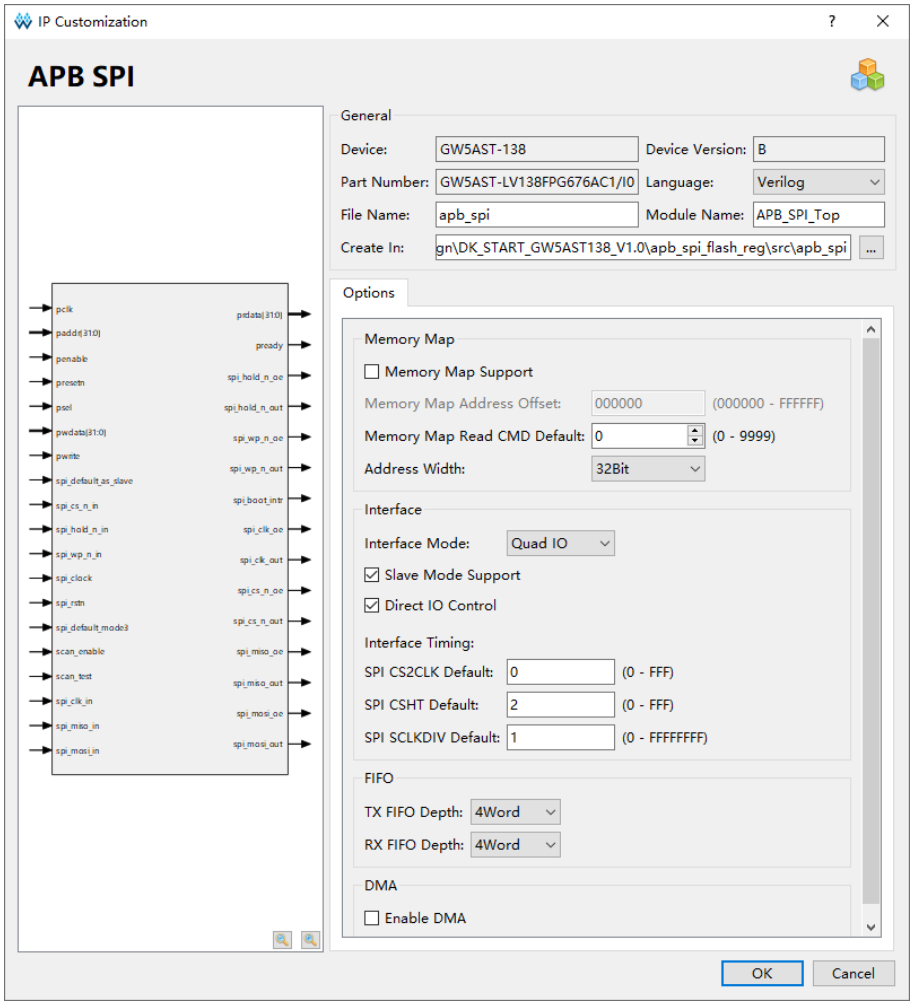
4 界面配置

用户可以在高云半导体云源软件的 IP Core Generator 工具，调用并配置 APB SPI IP。

选择菜单栏“Tool > IP Core Generator”或工具栏“”，打开 IP Core Generator，IP 列表中选择 Soft IP Core > Microprocessor System > Peripheral > APB SPI 1.0。

Gowin APB SPI IP 界面配置，如图 4-1 所示。

图 4-1 界面配置



Gowin APB SPI IP 选项配置描述如下：

- Memory Map Support：Gowin APB SPI 提供了一个内存映射访问的 AHB 总线接口，该选项用于配置是否启用内存映射访问接口。
- Memory Map Address Offset：用于内存映射访问的 ROM 的起始地址偏移，默认为 000000。例如，使用 SPI Flash Memory，设置起始地址偏移 0x600000 为启动地址，烧录引导程序或应用程序。
- Memory Map Read CMD Default：SPI 内存访问控制寄存器（0x50）的 MemRdCmd 字段的复位值，取值范围为 0 ~ 9999，默认值为 0。
- Address Width：用于配置 AHB 地址宽度，包括 24-bit 或 32-bit，默认为 32-bit。
- Interface Mode：用于配置接口模式，包括 Dual I/O 模式或 Quad I/O 模式，默认为 Quad I/O 模式。
- Slave Mode Support：用于配置是否启用 SPI 从模式，默认启用。
- Direct IO Control：用于配置是否启用直接 IO 控制，默认启用。
- SPI CS2CLK Default：SPI 接口时序寄存器（0x40）的 CS2SCLK 字段的复位值，取值范围为 0 ~ FFF，默认值为 0。
- SPI CSHT Default：SPI 接口时序寄存器（0x40）的 CSHT 字段的复位值，取值范围为 0 ~ FFF，默认值为 2。
- SPI SCLKDIV Default：SPI 接口时序寄存器（0x40）的 SCLK_DIV 字段的复位值，取值范围为 0 ~ FFFFFFFF，默认值为 1。
- TX FIFO Depth：用于配置发送 FIFO 的深度，包括 2、4、8、16、32、64 或 128-word，默认为 4-word。
- RX FIFO Depth：用于配置接收 FIFO 的深度，包括 2、4、8、16、32、64 或 128-word，默认为 4-word。
- Enable DMA：用于配置是否启用 DMA 功能，默认关闭。

5 编程模型

5.1 寄存器

5.1.1 寄存器概述

Gowin APB SPI 的寄存器定义如表 5-1 所示。Gowin APB SPI 寄存器定义，位于…\lib\driver\apb_spi.h。

表 5-1 寄存器定义

地址偏移	名称	描述
0x00	IDREV	ID 和修订寄存器
0x04~0x0C	-	保留
0x10	TRANSFMT	SPI 传输格式寄存器
0x14	DIRECTIO	SPI 直接 IO 控制寄存器
0x18~0x1C	-	保留
0x20	TRANSCTRL	SPI 传输控制寄存器
0x24	CMD	SPI 命令寄存器
0x28	ADDR	SPI 地址寄存器
0x2C	DATA	SPI 数据寄存器
0x30	CTRL	SPI 控制寄存器
0x34	STATUS	SPI 状态寄存器
0x38	INTREN	SPI 中断使能寄存器
0x3C	INTRST	SPI 中断状态寄存器
0x40	TIMING	SPI 接口时序寄存器
0x44~0x4C	-	保留
0x50	MEMCTRL	SPI 内存访问控制寄存器
0x54~0x5C	-	保留
0x60	SLVST	SPI 从设备状态寄存器
0x64	SLVDATACNT	SPI 从设备数据计数寄存器
0x68~0x78	-	保留
0x7C	CONFIG	SPI 配置寄存器

以下各节详细描述 APB SPI 寄存器定义。

寄存器类型缩略语概括如下：

- RO：Read-only
- R/W：Readable and writable
- W1C：Write 1 to clear

5.1.2 ID 和修订寄存器（0x00）

ID 和修订寄存器，用于保存 ID 和修订编号，初始值依赖于所用版本。

ID 和修订寄存器定义如表 5-2 所示。

表 5-2 ID 和修订寄存器

Name	Bit	Type	Description	Reset
ID	31:8	RO	ID number for SPI	0x000005
RevMajor	7:4	RO	Major revision number	Revision Dependent
RevMinor	3:0	RO	Minor revision number	Revision Dependent

5.1.3 SPI 传输格式寄存器（0x10）

SPI 传输格式寄存器，定义了 SPI 传输格式。

SPI 传输格式寄存器定义如表 5-3 所示。

表 5-3 SPI 传输格式寄存器

Name	Bit	Type	Description	Reset
-	31:18	-	Reserved	-
AddrLen	17:16	RW	Address length in bytes 0: 1 byte 1: 2 bytes 2: 3 bytes 3: 4 bytes	0x2
-	15:13	-	Reserved	-
DataLen	12:8	RW	The length of each data unit in bits The actual bit number of a data unit is (DataLen + 1)	0x07
DataMerge	7	RW	Enable Data Merge mode, which does automatic data split on write and data coalescing on read. This bit only takes effect when DataLen = 0x7. Under Data Merge mode, each write to the Data Register will transmit all four bytes of the write data; each read from the Data Register will retrieve four bytes of received data as a single word data. When Data Merge mode is disabled, only the least (DataLen + 1) significant bits of the Data Register are valid for read/write operations; no automatic data	0x1

Name	Bit	Type	Description	Reset
			split/coalescing will be performed.	
-	6:5	-	Reserved	-
MOSIBiDir	4	RW	Bi-directional MOSI in regular (single) mode 0: MOSI is uni-directional in regular mode. 1: MOSI is bi-directional signal in regular mode. This bi-directional signal replaces the two uni-directional data signals, MOSI and MISO	0x0
LSB	3	RW	Transfer data width the least significant bit first 0: Most significant bit first 1: Least significant bit first	0x0
SlvMode	2	RW	SPI Master/Slave mode selection 0: Master mode 1: Slave mode	
CPOL	1	RW	SPI Clock Polarity 0: SCLK is LOW in the idle states 1: SCLK is HIGH in the idle states	
CPHA	0	RW	SPI Clock Phase 0: Sampling data at odd SCLK edges 1: Sampling data at even SCLK edges	

5.1.4 SPI 直接 IO 控寄存器（0x14）

SPI 直接 IO 控制寄存器，启用 SPI 接口信号的直接控制。

SPI 直接 IO 控制寄存器定义如表 5-4 所示。

表 5-4 SPI 直接 IO 控寄存器

Name	Bit	Type	Description	Reset
-	31:25	-	Reserved	-
DirectIOEn	24	RW	Enable Direct IO 0: Disable 1: Enable	0x0
-	23:22	-	Reserved	-
HOLD_OE	21	RW	Output enable for the SPI Flash hold signal	0x0
WP_OE	20	RW	Output enable for the SPI Flash write protect signal	0x0
MISO_OE	19	RW	Output enable for the SPI MISO signal	0x0
MOSI_OE	18	RW	Output enable for the SPI MOSI signal	0x0
SCLK_OE	17	RW	Output enable for the SPI SCLK signal	0x0
CS_OE	16	RW	Output enable for the CS (chip select) signal	0x0
-	15:14	-	Reserved	-
HOLD_O	13		Output value for the SPI Flash hold signal	0x1
WP_O	12		Output value for the SPI Flash write project signal	0x1
MISO_O	11		Output value for the SPI MISO signal	0x0

Name	Bit	Type	Description	Reset
MOSI_O	10		Output value for the SPI MOSI signal	0x0
SCLK_O	9		Output value for the SPI SCLK signal	0x0
CS_O	8		Output value for the SPI CS (chip select) signal	0x1
-	7:6	-	Reserved	-
HOLD_I	5	RO	Status of the SPI Flash hold signal	
WP_I	4	RO	Status of the SPI Flash write protect signal	
MISO_I	3	RO	Status of the SPI MISO signal	
MOSI_I	2	RO	Status of the SPI MOSI signal	
SCLK_I	1	RO	Status of the SPI SCLK signal	
CS_I	0	RO	Status of the SPI CS (chip select)	

5.1.5 SPI 传输控制寄存器（0x20）

SPI 传输控制寄存器，控制 SPI 传输相位。请参考 5.1.6 SPI 命令寄存器（0x24）如何启用 SPI 传输。

SPI 传输控制寄存器定义如表 5-5 所示。

表 5-5 SPI 传输控制寄存器

Name	Bit	Type	Description	Reset
SlvDataOnly	31	RW	Data-only mode (slave mode only) 0: Disable the data-only mode 1: Enable the data-only mode **Note!** This mode only works in the uni-directional regular (single) mode so MOSIiDir, DualQuad and TransMode should be set to 0	0x0
CmdEn	30	RW	SPI command phase enable (Master mode only) 0: Disable the command phase 1: Enable the command phase	0x0
AddrEn	29	RW	SPI address phase enable (Master mode only) 0: Disable the address phase 1: Enable the address phase	0x0
AddrFmt	28	RW	SPI address phase format (Master mode only) 0: Address phase is the regular (single) mode 1: The format of the address phase is the same as the data phase (DualQuad)	0x0
TransMode	27:24	RW	Transfer mode The transfer sequence could be: 0: Write and read at the same time 1: Write only 2: Read only 3: Write, Read 4: Read, Write 5: Write, Dummy, Read 6: Read, Dummy, Write	0x0

<table>
<tr><th>Name</th><th>Bit</th><th>Type</th><th>Description</th><th>Reset</th></tr>
<tr><td></td><td></td><td></td><td>7: None Data (must enable CmdEn or AddrEn in master mode)
8: Dummy, Write
9: Dummy, Read
0xa~0xf: Reserved</td><td></td></tr>
<tr><td>DualQuad</td><td>23:22</td><td>RW</td><td>SPI data phase format
0: Regular (Single) mode
1: Dual I/O mode
2: Quad I/O mode
3: Reserved</td><td>0x0</td></tr>
<tr><td>TokenEn</td><td>21</td><td>RW</td><td>Token transfer enable (Master mode only)
Append a one-byte special token following the address phase for SPI read transfers. The value of the special token should be selected in TokenValue.
0: Disable the one-byte special token
1: Enable the one-byte special token</td><td>0x0</td></tr>
<tr><td>WrTranCnt</td><td>20:12</td><td>RW</td><td>Transfer count for write data
WrTranCnt indicates the number of units of data to be transmitted to the SPI bus from the Data Register.
The actual transfer count is (WrTranCnt + 1).
WrTranCnt only takes effect when TransMode is 0, 1, 3, 4, 5, 6 or 8.
The size (bit-width) of a data unit is defined by the DataLen field of the Transfer Format Register.
For TransMode 0, WrTranCnt must be equal to RdTranCnt.</td><td></td></tr>
<tr><td>TokenValue</td><td>11</td><td>RW</td><td>Token value (Master mode only)
The value of the one-byte special token following the address phase for SPI read transfers.
0: token value = 0x00
1: token value = 0x69</td><td>0x0</td></tr>
<tr><td>DummyCnt</td><td>10:9</td><td>RW</td><td>Dummy data count. The actual dummy count is (DummyCnt + 1).
The number of dummy cycles on the SPI interface will be (DummyCnt+1) * ((DataLen+1)/SPI IO width)
The Data pins are put into the high impedance during the dummy data phase.
DummyCnt is only used for TransMode 5, 6, 8 and 9, which has dummy data phases.
The following table shows dummy cycle settings under some common transfer formats:
<table>
<tr><th>DummyCnt+1</th><th>DataLen+1</th><th>DualQuad</th><th>#Dummy Cycles on the SPI Interface</th></tr>
<tr><td>1</td><td>8</td><td>Regular/ Single</td><td>8</td></tr>
<tr><td>1</td><td>8</td><td>Dual</td><td>4</td></tr>
</table></td><td>0x0</td></tr>
</table>

<table>
<tr><th>Name</th><th>Bit</th><th>Type</th><th>Description</th><th>Reset</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td>1</td><td>8</td><td>Quad</td><td>2</td></tr>
<tr><td>2</td><td>8</td><td>Quad</td><td>4</td></tr>
<tr><td>3</td><td>8</td><td>Quad</td><td>6</td></tr>
<tr><td>1</td><td>32</td><td>Quad</td><td>8</td></tr>
</table>
</td><td></td></tr>
<tr><td>RdTranCnt</td><td>8:0</td><td>RW</td><td>Transfer count for read data
RdTranCnt indicates the number of units of data to be received from SPI bus and stored to the Data Register. The actual received count is (RdTranCnt+1).
RdTransCnt only takes effect when TransMode is 0, 2, 3, 4, 5, 6 or 9.
The size (bit-width) of a data unit is defined by the DataLen field of the Transfer Format Register.
For TransMode 0, WrTranCnt must equal RdTranCnt.</td><td>0x0</td></tr>
</table>

5.1.6 SPI 命令寄存器（0x24）

SPI 命令寄存器中的写操作，触发 SPI 传输。即使命令阶段没有启用，该寄存器也必须写入一个空值来启用一个 SPI 传输。当 SPI 控制器作为从机模式时，SPI 命令寄存器保存最后一次接收到的 SPI 事务的命令字段。

SPI 命令寄存器定义如表 5-6 所示。

表 5-6 SPI 命令寄存器

Name	Bit	Type	Description	Reset
-	31:8	-	Reserved	-
CMD	7:0	RW	SPI Command	0x0

5.1.7 SPI 地址寄存器（0x28）

SPI 地址寄存器定义如表 5-7 所示。

表 5-7 SPI 地址寄存器

Name	Bit	Type	Description	Reset
ADDR	31:0	RW	SPI Address (Master mode only)	0x0

5.1.8 SPI 数据寄存器（0x2C）

当控制器为数据归并模式时，SPI 数据寄存器的字节存储顺序为小端模式。

SPI 数据寄存器定义如表 5-8 所示。

表 5-8 SPI 数据寄存器

Name	Bit	Type	Description	Reset
DATA	31:0	RW	Data to transmit or the received data For writes, data is enqueued to the TX FIFO. The least significant byte is always transmitted first. If the TX FIFO is full and the SPIActive bit of the status register is 1, the ready signal hready/pready will be deasserted to insert wait states to the transfer. For reads, data is read and dequeued from the RX FIFO. The least significant byte is the first received byte. If the RX FIFO is empty and the SPIActive bit of the status register is 1, the ready signal hready/pready will be deasserted to insert wait states to the transfer. The FIFOs decouple the speed of the SPI transfers and the software's generation/consumption of data. When the TX FIFO is empty, SPI transfers will hold until more data is written to the TX FIFO; when the RX FIFO is full, SPI transfers will hold until there is more room in the RX FIFO. If more data is written to the TX FIFO than the write transfer count (WrTranCnt), the remaining data will stay in the TX FIFO for the next transfer or until the TX FIFO is reset.	0x0

5.1.9 SPI 控制寄存器（0x30）

SPI 控制寄存器定义如表 5-9 所示。

表 5-9 SPI 控制寄存器

Name	Bit	Type	Description	Reset
-	31:24	-	Reserved	-
TXTHRES	23:16	RW	Transmit (TX) FIFO Threshold The TXFIFOInt interrupt or DMA request would be issued to replenish the TX FIFO when the TX data count is less than or equal to the TX FIFO threshold.	0x0
RXTHRES	15:8	RW	Receive (RX) FIFO Threshold The RXFIFOInt interrupt or DMA request would be issued for consuming the RX FIFO when the RX data count is more than	0x0

Name	Bit	Type	Description	Reset
			or equal to the RX FIFO threshold.	
-	7:5	-	Reserved	-
TXDMAEN	4	RW	TX DMA enable	0x0
RXDMAEN	3	RW	RX DMA enable	0x0
TXFIFORST	2	RW	Transmit FIFO reset Write 1 to reset. It is automatically cleared to 0 after the reset operation completes.	0x0
RXFIFORST	1	RW	Receive FIFO reset Write 1 to reset. It is automatically cleared to 0 after the reset operation completes.	0x0
SPIRST	0	RW	SPI reset Write 1 to reset. It is automatically cleared to 0 after the reset operation completes.	0x0

5.1.10 SPI 状态寄存器（0x34）

SPI 状态寄存器定义如表 5-10 所示。

表 5-10 SPI 状态寄存器

Name	Bit	Type	Description	Reset
-	31:30	-	Reserved	-
TXNUM[7:6]	29:28	RO	Number of valid entries in the Transmit FIFO	0x0
-	27:26	-	Reserved	-
RXNUM[7:6]	25:24	RO	Number of valid entries in the Receive FIFO	0x0
TXFULL	23	RO	Transmit FIFO Full flag	0x0
TXEMPTY	22	RO	Transmit FIFO Empty flag	0x1
TXNUM[5:0]	21:16	RO	Number of valid entries in the Transmit FIFO	0x0
RXFULL	15	RO	Receive FIFO Full flag	0x0
RXEMPTY	14	RO	Receive FIFO Empty flag	0x1
RXNUM[5:0]	13:8	RO	Number of valid entries in the Receive FIFO	0x0
-	7:1	RO	Reserved	-
SPIActive	0	RO	SPI register programming is in progress. In master mode, SPIActive becomes 1 after the SPI command register is written and becomes 0 after the transfer is finished. In slave mode, SPIActive becomes 1 after the SPI CS signal is asserted and becomes 0 after the SPI CS signal is deasserted. Note that due to clock synchronization, it may take at most two spi_clock cycles for SPIActive to change when the corresponding condition happens. Note this bit stays 0 when Direct IO Control or the memory-mapped interface is used.	0x0

5.1.11 SPI 中断使能寄存器（0x38）

SPI 中断使能寄存器定义如表 5-11 所示。

表 5-11 SPI 中断使能寄存器

Name	Bit	Type	Description	Reset
-	31:6	-	Reserved	-
SlvCmdEn	5	RW	Enable the Slave Command Interrupt. Control whether interrupts are triggered whenever slave commands are received. (Slave mode only)	0x0
EndIntEn	4	RW	Enable the End of SPI Transfer interrupt. Control whether interrupts are triggered when SPI transfers end. (In slave mode, end of read status transaction doesn't trigger this interrupt.)	0x0
TXFIFOIntEn	3	RW	Enable SPI Transmit FIFO Threshold interrupt. Control whether interrupts are triggered when the valid entries are less than or equal to TX FIFO threshold.	0x0
RXFIFOIntEn	2	RW	Enable the SPI Receive FIFO Threshold interrupt. Control whether interrupts are triggered when the valid entries are greater than or equal to the RX FIFO threshold.	0x0
TXFIFOURIntEn	1	RW	Enable the SPI Transmit FIFO Underrun interrupt. Control whether interrupts are triggered when the Transmit FIFO run out of data. (Slave mode only)	0x0
RXFIFOORIntEn	0	RW	Enable the SPI Receive FIFO Overrun interrupt. Control whether interrupts are triggered when the Receive FIFO overflows. (Slave mode only)	0x0

5.1.12 SPI 中断状态寄存器（0x3C）

SPI 中断状态寄存器定义如表 5-12 所示。

表 5-12 SPI 中断状态寄存器

Name	Bit	Type	Description	Reset
-	31:6	-	Reserved	-
SlvCmdInt	5	W1C	Slave Command Interrupt. This bit is set when Slave Command interrupts occur. (Slave mode only)	0x0
EndInt	4	W1C	End of SPI Transfer interrupt. This bit is set when End of SPI Transfer interrupts occur.	0x0

Name	Bit	Type	Description	Reset
TXFIFOInt	3	W1C	TX FIFO Threshold interrupt. This bit is set when TX FIFO Threshold interrupts occur.	0x0
RXFIFOInt	2	W1C	RX FIFO Threshold interrupt. This bit is set when RX FIFO Threshold interrupts occur.	0x0
TXFIFOURInt	1	W1C	TX FIFO Underrun interrupt. This bit is set when TX FIFO Underrun interrupts occur. (Slave mode only)	0x0
RXFIFOORInt	0	W1C	RX FIFO Overrun interrupt. This bit is set when RX FIFO Overrun interrupts occur. (Slave mode only)	0x0

5.1.13 SPI 接口时序寄存器（0x40）

SPI 接口时序寄存器，控制 SPI 接口时序以满足 SPI 从机接口时序要求。仅主机模式时需要编程该寄存器。

SPI 接口时序寄存器定义如表 5-13 所示。

表 5-13 SPI 接口时序寄存器

Name	Bit	Type	Description	Reset
-	31:14	-	Reserved	-
CS2SCLK	13:12	RW	The minimum time between the edges of SPI CS and the edges of SCLK. The actual duration is (SCLK period / 2) × (CS2SCLK + 1)	Configuration dependent
CSHT	11:8	RW	The minimum time that SPI CS should stay HIGH. The actual duration is (SCLK period / 2) × (CSHT + 1)	Configuration dependent
SCLK_DIV	7:0	RW	The clock frequency ratio between the clock source and SPI interface SCLK. SCLK period = ((SCLK_DIV + 1) × 2) × (Period of the SPI clock source) The SCLK_DIV value 0xff is a special value which indicates that the SCLK frequency should be the same as the spi_clock frequency.	Configuration dependent

5.1.14 SPI 内存访问控制寄存器（0x50）

SPI 内存访问控制寄存器，为内存映射的 AHB 读访问定义的 SPI 命令。

当正在编程该寄存器或 SPI 接口时序寄存器（0x40）时，应停止内存映射的 AHB 读访问。当 MemCtrlChg 位清零后，才能继续 AHB 读访问。

SPI 内存访问控制寄存器定义如表 5-14 所示。

表 5-14 SPI 内存访问控制寄存器

Name	Bit	Type	Description	Reset
-	31:9	-	Reserved	-
MemCtrlChg	8	RO	This bit is set when this register (0x50) or the SPI Interface Timing Register (0x40) is written; it is automatically cleared when the new programming takes effect.	0
-	7:4	-	Reserved	-
MemRdCmd	3:0	RW	Selects the SPI command for serving the memory-mapped reads on the AHB bus The command encoding table is listed in 表 5-15. The latency of each command is listed in 表 5-16.	Configuration dependent

表 5-15 Supported SPI Read Commands for Memory-Mapped AHB Reads

MemRdCmd	Command	Address	Dummy	Data
0	0x03	3 bytes in Regular mode	N/A	Regular mode
1	0x0B	3 bytes in Regular mode	1 byte in Regular mode	Regular mode
2	0x3B	3 bytes in Regular mode	1 byte in Regular mode	Dual mode
3	0x6B	3 bytes in Regular mode	1 byte in Regular mode	Quad mode
4	0xBB	(3-byte address + 1-byte 0) in Dual mode	N/A	Dual mode
5	0xEB	(3-byte address + 1-byte 0) in Quad mode	2 bytes in Quad mode	Quad mode
6-7	Reserved	-	-	-
8	0x13	4 bytes in Regular mode	N/A	Regular mode
9	0x0C	4 bytes in Regular mode	1 byte in Regular mode	Regular mode
10	0x3C	4 bytes in Regular mode	1 byte in Regular mode	Dual mode
11	0x6C	4 bytes in Regular	1 byte in	Quad

MemRdCmd	Command	Address	Dummy	Data
		mode	Regular mode	mode
12	0xBC	(4-byte address + 1-byte 0) in Dual mode	N/A	Dual mode
13	0xEC	(4-byte address + 1-byte 0) in Quad mode	2 bytes in Quad mode	Quad mode
14-15	Reserved	-	-	-

表 5-16 Latency of a 4 Bytes Data Transfer through the AHB Memory Read Port

Command	Non-sequential	Sequential	Sequential (prefetched*)
0x03	8 BUS_CLK + 10 SPI_CLK + 64 SCLK	3 BUS_CLK + 32 SCLK	1 BUS_CLK
0x0B	8 BUS_CLK + 10 SPI_CLK + 72 SCLK	3 BUS_CLK + 32 SCLK	1 BUS_CLK
0x3B	8 BUS_CLK + 10 SPI_CLK + 56 SCLK	3 BUS_CLK + 16 SCLK	1 BUS_CLK
0x6B	8 BUS_CLK + 10 SPI_CLK + 48 SCLK	3 BUS_CLK + 8 SCLK	1 BUS_CLK
0xBB	8 BUS_CLK + 10 SPI_CLK + 40 SCLK	3 BUS_CLK + 16 SCLK	1 BUS_CLK
0xEB	8 BUS_CLK + 10 SPI_CLK + 28 SCLK	3 BUS_CLK + 8 SCLK	1 BUS_CLK
0x13	8 BUS_CLK + 10 SPI_CLK + 72 SCLK	3 BUS_CLK + 32 SCLK	1 BUS_CLK
0x0C	8 BUS_CLK + 10 SPI_CLK + 80 SCLK	3 BUS_CLK + 32 SCLK	1 BUS_CLK
0x3C	8 BUS_CLK + 10 SPI_CLK + 64 SCLK	3 BUS_CLK + 16 SCLK	1 BUS_CLK
0x6C	8 BUS_CLK + 10 SPI_CLK + 56 SCLK	3 BUS_CLK + 8 SCLK	1 BUS_CLK
0xBC	8 BUS_CLK + 10 SPI_CLK + 44 SCLK	3 BUS_CLK + 16 SCLK	1 BUS_CLK
0xEC	8 BUS_CLK + 10 SPI_CLK + 30 SCLK	3 BUS_CLK + 8 SCLK	1 BUS_CLK

Note!

- BUS_CLK：AHB 总线时钟周期。
- SCLK：SCLK 时钟周期。

5.1.15 SPI 从设备状态寄存器（0x60）

SPI 从设备状态寄存器，保存从设备的状态，SPI 主机可以通过读状态命令获取这些状态。

SPI 从设备状态寄存器定义如表 5-17 所示。

表 5-17 SPI 从设备状态寄存器

Name	Bit	Type	Description	Reset
-	31:19	-	Reserved	-
UnderRun	18	W1C	Data underrun occurs in the last transaction	0
OverRun	17	W1C	Data overrun occurs in the last transaction	0
Ready	16	RW	Set this bit to indicate that the SPI is ready for data transaction. When an SPI transaction other than slave status-reading command ends, this bit will be cleared to 0.	0
USR_Status	15:0	RW	User defined status flags	0

5.1.16 SPI 从设备数据计数寄存器（0x64）

SPI 从设备数据计数寄存器，描述了从模式下的读/写事务的数据数量，根据数据数量信息访问 SPI 数据寄存器。

SPI 从设备数据计数寄存器定义如表 5-18 所示。

表 5-18 SPI 从设备数据计数寄存器

Name	Bit	Type	Description	Reset
-	31:26	-	Reserved	-
WCnt	25:16	RO	Slave transmitted data count	0
-	15:10	-	Reserved	-
RCnt	9:0	RO	Slave received data count	0

5.1.17 SPI 配置寄存器（0x7C）

SPI 配置寄存器如表 5-19 所示。

表 5-19 SPI 配置寄存器

Name	Bit	Type	Description	Reset
-	31:15	-	Reserved	-
Slave	14	RO	Support for SPI Slave mode	Configuration dependent
AHBMem	12	RO	Support for memory-mapped access (read-only) through AHB bus	Configuration dependent
DirectIO	11	RO	Support for Direct SPI IO	Configuration dependent
-	10	-	Reserved	-
QuadSPI	9	RO	Support for Quad I/O SPI	Configuration dependent

Name	Bit	Type	Description	Reset
DualSPI	8	RO	Support for Dual I/O SPI	Configuration dependent
TxFIFOSize	7:4	RO	Depth of TX FIFO 0: 2 words 1: 4 words 2: 8 words 3: 16 words 4: 32 words 5: 64 words 6: 128 words	Configuration dependent
RxFIFOSize	3:0	RO	Depth of RX FIFO 0: 2 words 1: 4 words 2: 8 words 3: 16 words 4: 32 words 5: 64 words 6: 128 words	Configuration dependent

5.2 SPI 驱动函数

5.2.1 SPI 驱动函数概述

Gowin APB SPI 驱动函数定义如表 5-20 所示。Gowin APB SPI 驱动函数定义，位于…\lib\driver\apb_spi_driver.h 和 apb_spi_driver.c。

表 5-20 驱动函数定义

驱动函数	描述
apb_spix_get_capabilities	获取 APB SPI 驱动的功能信息
apb_spix_initialize	初始化 APB SPI 接口
apb_spix_uninitialize	卸载 APB SPI 接口
apb_spix_power_control	指定 APB SPI 接口的功耗模式
apb_spix_send	通过 APB SPI 驱动发送器发送数据
apb_spix_receive	从 APB SPI 驱动接收器接收数据
apb_spix_transfer	通过 APB SPI 接口传输数据
apb_spix_get_data_count	获取 APB SPI 接口传输数据的数量
apb_spix_control	配置 APB SPI 接口的设置，执行指定的操作
apb_spix_get_status	获取 APB SPI 接口的状态
apb_spix_irq_handler	APB SPI 中断处理程序
apb_spix_dma_tx_event	APB SPI 的 DMA 发送事件处理
apb_spix_dma_rx_event	APB SPI 的 DMA 接收事件处理

以下各节详细描述 SPI 的驱动函数定义。

5.2.2 apb_spix_get_capabilities

apb_spix_get_capabilities 函数定义如表 5-21 所示。

表 5-21 apb_spix_get_capabilities 函数定义

原型	APB_SPI_CAPABILITIES apb_spix_get_capabilities(APB_SPI_RESOURCES *spi)
描述	获取 APB SPI 驱动的功能信息
参数	spi：指向 APB_SPI_RESOURCES 结构体的指针
返回值	APB SPI 驱动的功能信息

5.2.3 apb_spix_initialize

apb_spix_initialize 函数定义如表 5-22 所示。

表 5-22 apb_spix_initialize 函数定义

原型	int apb_spix_initialize(APB_SPI_SignalEvent_t cb_event, APB_SPI_RESOURCES *spi)
描述	初始化 APB SPI 接口
参数	cb_event：指向 APB_SPI_SignalEvent 回调函数的指针 spi：指向 APB_SPI_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

5.2.4 apb_spix_uninitialize

apb_spix_uninitialize 函数定义如表 5-23 所示。

表 5-23 apb_spix_uninitialize 函数定义

原型	int apb_spix_uninitialize (APB_SPI_RESOURCES *spi)
描述	卸载 APB SPI 接口
参数	spi：指向 APB_SPI_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

5.2.5 apb_spix_power_control

apb_spix_power_control 函数定义如表 5-24 所示。

表 5-24 apb_spix_power_control 函数定义

原型	int apb_spix_power_control (APB_SPI_POWER_STATE state, APB_SPI_RESOURCES *spi)
描述	指定 APB SPI 接口的功耗模式
参数	spi：指向 APB_SPI_RESOURCES 结构体的指针 state：APB SPI 接口功耗模式，包括： APB_SPI_POWER_FULL：to set up peripherals for data transfers, enable interrupts and DMA APB_SPI_POWER_LOW：to enable power-saving APB_SPI_POWER_OFF：to terminate pending data transfers and disable peripherals, related interrupts and DMA
返回值	如果发生执行错误，返回一个负值

5.2.6 apb_spix_send

apb_spix_send 函数定义如表 5-25 所示。

表 5-25 apb_spix_send 函数定义

原型	int apb_spix_send (const void *data, unsigned int num, APB_SPI_RESOURCES *spi)
描述	通过 APB SPI 驱动发送器发送数据
参数	data：指向发送数据缓存区的指针 num：发送数据的长度 spi： 指向 APB_SPI_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

5.2.7 apb_spix_receive

apb_spix_receive 函数定义如表 5-26 所示。

表 5-26 apb_spix_receive 函数定义

原型	int apb_spix_receive (void *data, unsigned int num, APB_SPI_RESOURCES *spi)
描述	从 APB SPI 驱动接收器接收数据
参数	data：指向接收数据缓存区的指针 num：接收数据的长度 spi： 指向 APB_SPI_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

5.2.8 apb_spix_transfer

apb_spix_transfer 函数定义如表 5-27 所示。

表 5-27 apb_spix_transfer 函数定义

原型	int apb_spix_transfer (const void *data_out, void *data_in, unsigned int num, APB_SPI_RESOURCES *spi)
描述	通过 APB SPI 接口传输数据
参数	data_out：指向发送数据缓存区的指针 data_in：指向接收数据缓存区的指针 num：传输数据的长度 spi：指向 APB_SPI_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

5.2.9 apb_spix_get_data_count

apb_spix_get_data_count 函数定义如表 5-28 所示。

表 5-28 apb_spix_get_data_count 函数定义

原型	int apb_spix_get_data_count(APB_SPI_RESOURCES *spi)
描述	获取 APB SPI 接口传输数据的数量
参数	spi：指向 APB_SPI_RESOURCES 结构体的指针
返回值	APB SPI 最后一次执行数据传输时，传输数据的数量

5.2.10 apb_spix_control

apb_spix_control 函数定义如表 5-29 所示。

表 5-29 apb_spix_control 函数定义

原型	int apb_spix_control(unsigned int control, unsigned int arg, APB_SPI_RESOURCES *spi)
描述	配置 APB SPI 接口的设置，执行指定的操作
参数	control：APB SPI 驱动接口的一种设置或执行的一种操作 arg：指定设置或操作的附加信息 spi：指向 APB_SPI_RESOURCES 结构体的指针
返回值	如果发生执行错误，返回一个负值

“control”和“arg”设置与操作如表 5-30 所示。

表 5-30 Control Settings or Operations

Options for control	arg specifies	Settings or operations
Mode controls (Bits: 0~7)		
APB_SPI_MODE_INACTIVE		Sets the SPI to inactive
APB_SPI_MODE_MASTER	Bus speed in bps	Sets the SPI to the master (output on MOSI, and input on MISO)
APB_SPI_MODE_MASTER_SIMPLEX	Bus speed in bps	Sets the SPI to the master (output and input on MOSI)
APB_SPI_MODE_SLAVE		Sets the SPI to the slave (output on MISO, and input on MOSI)
APB_SPI_MODE_SLAVE_SIMPLEX		Sets the SPI to the slave (output and input on MISO)
Clock polarity (Frame format) (Bits: 8~11)		
APB_SPI_CPOL0_CPHA0 (default)		Sets the clock polarity to 0 and clock phase to 0
APB_SPI_CPOL0_CPHA1		Sets the clock polarity to 0 and clock phase to 1
APB_SPI_CPOL1_CPHA0		Sets the clock polarity to 1 and clock phase to 0
APB_SPI_CPOL1_CPHA1		Sets the clock polarity to 1 and clock phase to 1
APB_SPI_TI_SSI		Uses the Texas Instruments Frame Format
APB_SPI_MICROWIRE		Uses the National Microwire Frame Format
Parity bit (Bits: 12~17)		
APB_SPI_DATA_BITS(N)		Sets the number of bits per SPI frame; N ranges from 1 to 32. This is the minimum required parameter.

Options for control	arg specifies	Settings or operations
Bit order (Bits: 18)		
APB_SPI_MSB_LSB (default)		Sets the bit order from MSB to LSB
APB_SPI_LSB_MSB		Sets the bit order from LSB to MSB
Slave select mode (Bits: 19~21)		
APB_SPI_SS_MASTER_UNUSED (default)		Sets the Slave Select mode for the master to “Not used”. It is specified along with the option APB_SPI_MODE_MASTER. The master does not drive or monitor the SS line.
APB_SPI_SS_MASTER_SW		Sets the Slave Select mode for the master to “Software controlled”. It is specified along with the option APB_SPI_MODE_MASTER. The Slave Select line is configured as output and controlled via APB_SPI_CONTROL_SS. By default, the Slave Select line is not active (i.e., high), and is not affected by transfer, send, or receive functions.
APB_SPI_SS_MASTER_HW_OUTPUT		Sets the Slave Select mode for the master to “Hardware controlled output”. It is specified along with the option APB_SPI_MODE_MASTER. The Slave Select line is configured as output and controlled by the hardware. The transfers via the line is activated or deactivated by the hardware and is not affected by APB_SPI_CONTROL_SS.
APB_SPI_SS_MASTER_HW_INPUT		Sets the Slave Select mode for the master to “Hardware monitored input”. It is specified along with the option APB_SPI_MODE_MASTER and used in multi-master configuration where a master monitors the Slave Select but not drives it. The Slave Select is configured as input. When another master activates this line, the previous active master backs off. This causes a Mode Fault APB_SPI_EVENT_MODE_FAULT and makes the SPI switch

Options for control	arg specifies	Settings or operations
		to be inactive.
APB_SPI_SS_SLAVE_HW (default)		Sets the Slave Select mode for the slave to “Hardware monitored”. It is specified along with the option APB_SPI_MODE_SLAVE. The hardware monitors the Slave Select line and accepts transfers only when the line is active. Transfers are ignored while the Slave Select line is inactive.
APB_SPI_SS_SLAVE_SW		Sets the Slave Select mode for the slave to “Software controlled”. It is specified along with the option APB_SPI_MODE_SLAVE when the Slave Select line is not used. For example, when a single master and slave are connected in the system, the Slave Select line will not be needed. The software controls whether the slave respond or not (not respond by default) and enables/disables transfers by APB_SPI_CONTROL_SS.
Other controls (Bits: 0~21)		
APB_SPI_SET_BUS_SPEED	Bus speed in bps	Sets the bus speed
APB_SPI_GET_BUS_SPEED		Gets the bus speed
APB_SPI_SET_DEFAULT_TX_VALUE	Transmission value	Sets the default transmission value
APB_SPI_CONTROL_SS	available values are APB_SPI_SS_INACTIVE, APB_SPI_SS_ACTIVE	Controls the Slave Select (SS) signal
APB_SPI_ABORT_TRANSFER		Aborts the current data transfer

5.2.11 apb_spix_get_status

apb_spix_get_status 函数定义如表 5-31 所示。

表 5-31 apb_spix_get_status 函数定义

原型	APB_SPI_STATUS apb_spix_get_status(APB_SPI_RESOURCES *spi)
描述	获取 APB SPI 接口的状态

参数	spi：指向 APB_SPI_RESOURCES 结构体的指针
返回值	APB SPI 接口的当前状态

5.2.12 apb_spix_irq_handler

apb_spix_irq_handler 函数定义如表 5-32 所示。

表 5-32 apb_spix_irq_handler 函数定义

原型	void apb_spix_irq_handler(APB_SPI_RESOURCES *spi)
描述	APB SPI 中断处理程序
参数	spi：指向 APB_SPI_RESOURCES 结构体的指针
返回值	无

5.2.13 apb_spix_dma_tx_event

apb_spix_dma_tx_event 函数定义如表 5-33 所示。

表 5-33 apb_spix_dma_tx_event 函数定义

原型	void apb_spix_dma_tx_event (unsigned int event, APB_SPI_RESOURCES * spi)
描述	APB SPI 的 DMA 发送事件处理
参数	event：DMA 发送事件 spi：指向 APB_SPI_RESOURCES 结构体的指针
返回值	无

5.2.14 apb_spix_dma_rx_event

apb_spix_dma_rx_event 函数定义如表 5-34 所示。

表 5-34 apb_spix_dma_rx_event 函数定义

原型	void apb_spix_dma_rx_event (unsigned int event, APB_SPI_RESOURCES * spi)
描述	APB SPI 的 DMA 接收事件处理
参数	event：DMA 接收事件 spi：指向 APB_SPI_RESOURCES 结构体的指针
返回值	无

5.3 SPI Flash 驱动函数

5.3.1 SPI Flash 驱动函数概述

SPI Flash 驱动函数定义如表 5-35 所示。

SPI Flash 驱动函数定义，位于…\lib\driver\apb_spi_flash_driver.h 和 apb_spi_flash_driver.c。

表 5-35 驱动函数定义

驱动函数	描述
apb_spi_flash_init	初始化 SPI Flash
apb_spi_flash_switch_mode	切换 SPI Flash 烧录/内存读写模式
apb_spi_flash_read	读数据
apb_spi_flash_write	写数据
apb_spi_flash_program	以页为单元写数据
apb_spi_flash_erase_4ksec	以扇区（4KB）为单元擦除数据
apb_spi_flash_erase_64ksec	以块（64KB）为单元擦除数据

SPI Flash 的命令定义如表 5-36 所示。

表 5-36 SPI Flash 命令定义

宏定义	命令码	描述
APB_SPI_FLASH_WRITE_CMD	0x01	写数据
APB_SPI_FLASH_PROGRAM_CMD	0x02	页编程
APB_SPI_FLASH_READ_CMD	0x03	读数据
APB_SPI_FLASH_WRITE_DISABLE_CMD	0x04	写禁用
APB_SPI_FLASH_READ_STATUS_CMD	0x05	读状态
APB_SPI_FLASH_WRITE_ENABLE_CMD	0x06	写使能
APB_SPI_FLASH_ERASE_4K_CMD	0x20	扇区擦除
APB_SPI_FLASH_ERASE_64K_CMD	0xD8	块擦除
APB_SPI_FLASH_ERASE_CHIP_CMD	0x60	整片擦除

以下各节详细描述 SPI Flash 的驱动函数定义。

5.3.2 apb_spi_flash_init

apb_spi_flash_init 函数定义如表 5-37 所示。

表 5-37 apb_spi_flash_init 函数定义

原型	int apb_spi_flash_init(APB_SPI_RegDef *DEV_FLASH)
描述	初始化 SPI Flash
参数	DEV_FLASH：指向 APB_SPI_RegDef 结构体的指针
返回值	如果发生错误，返回一个负值

5.3.3 apb_spi_flash_switch_mode

apb_spi_flash_switch_mode 函数定义如表 5-38 所示。

表 5-38 apb_spi_flash_switch_mode 函数定义

原型	int apb_spi_flash_switch_mode(APB_SPI_RegDef *DEV_FLASH)
描述	切换 SPI Flash 烧录/内存读写模式
参数	DEV_FLASH：指向 APB_SPI_RegDef 结构体的指针
返回值	如果发生错误，返回一个负值

5.3.4 apb_spi_flash_read

apb_spi_flash_read 函数定义如表 5-39 所示。

表 5-39 apb_spi_flash_read 函数定义

原型	int apb_spi_flash_read(APB_SPI_RegDef *DEV_FLASH, unsigned int len, unsigned int cmd, unsigned int address, unsigned char *buf)
描述	读数据
参数	DEV_FLASH：指向 APB_SPI_RegDef 结构体的指针 len：读取数据的长度 cmd： SPI Flash 命令 address：读取数据的 SPI Flash 的起始地址 buf：指向读取数据缓存区的指针
返回值	如果发生错误，返回一个负值

5.3.5 apb_spi_flash_write

apb_spi_flash_write 函数定义如表 5-40 所示。

表 5-40 apb_spi_flash_write 函数定义

原型	int apb_spi_flash_write(APB_SPI_RegDef *DEV_FLASH, unsigned int len, unsigned int cmd, unsigned int address, unsigned char *buf)
描述	写数据
参数	DEV_FLASH：指向 APB_SPI_RegDef 结构体的指针 len：数据的长度 cmd： SPI Flash 命令 address：写入数据的 SPI Flash 的起始地址 buf：指向写入数据缓存区的指针
返回值	如果发生错误，返回一个负值

5.3.6 apb_spi_flash_program

apb_spi_flash_program 函数定义如表 5-41 所示。

表 5-41 apb_spi_flash_program 函数定义

原型	int apb_spi_flash_program(APB_SPI_RegDef *DEV_FLASH, unsigned int len, unsigned int address, unsigned char *buf)
描述	以页为单元写数据
参数	DEV_FLASH：指向 APB_SPI_RegDef 结构体的指针 len：写入数据的长度 address：写入数据的 SPI Flash 的起始地址 buf：指向写入数据缓存区的指针
返回值	如果发生错误，返回一个负值

5.3.7 apb_spi_flash_erase_4ksec

apb_spi_flash_erase_4ksec 函数定义如表 5-42 所示。

表 5-42 apb_spi_flash_erase_4ksec 函数定义

原型	int apb_spi_flash_erase_4ksec(APB_SPI_RegDef *DEV_FLASH, unsigned int address)
描述	以扇区（4KB）为单元擦除数据
参数	DEV_FLASH：指向 APB_SPI_RegDef 结构体的指针 address：擦除数据的 SPI Flash 的起始地址
返回值	如果发生错误，返回一个负值

5.3.8 apb_spi_flash_erase_64ksec

apb_spi_flash_erase_64ksec 函数定义如表 5-43 所示。

表 5-43 apb_spi_flash_erase_64ksec 函数定义

原型	int apb_spi_flash_erase_64ksec(APB_SPI_RegDef *DEV_FLASH, unsigned int address)
描述	以块（64KB）为单元擦除数据
参数	DEV_FLASH：指向 APB_SPI_RegDef 结构体的指针 address：擦除数据的 SPI Flash 的起始地址
返回值	如果发生错误，返回一个负值

6 编程序列

本章描述了通过寄存器编程发起 SPI 传输的编程序列。在 AHB 总线上进行内存映射读操作时，不应执行寄存器编程；同样，在进行寄存器编程期间，也不应执行内存映射读操作。

6.1 主模式

6.1.1 SPI 写操作（使用 DMA）

6.1.1.1 场景

以下示例编程序列设置控制器，以实现：

- 传输 2-byte 地址和 8-bit 数据宽度，
- 总传输计数为 16，
- 将四个字节的数据合并成一个字，
- 使用硬件握手的 DMA 数据传输，
- 在 SPI 传输结束时触发中断，
- SPI SCLK 频率为 SPI 时钟源频率的一半，
- 向 ROM 发出“页编程”命令（0x02）。

6.1.1.2 编程序列

SPI 传输格式设置

1. 读取 SPI 配置寄存器（0x7C）中的 TX/RX FIFO 深度。
2. 等待之前的 SPI 传输完成，方法是等待 SPI 状态寄存器（0x34）中的 SPIActive 位变为零。
3. 如下设置 SPI 传输格式寄存器（0x10）：
   - AddrLen = 1（地址长度 - 1）
   - DataLen = 7（数据长度 - 1）
   - DataMerge = 1
   - 其他字段保持复位值。

4. 如下设置 SPI 传输控制寄存器（0x20）：
   - CmdEn = 1
   - AddrEn = 1
   - TransMode = 1（只写）
   - WrTranCnt = 15（总传输计数 - 1）
   - 其他字段保持复位值
5. 设置 SPI 控制寄存器（0x30）：启用 DMA，复位 TX FIFO，并根据 TX FIFO 配置指定 TX FIFO 阈值，例如设置为 TX FIFO 深度的一半。
6. 设置 SPI 中断使能寄存器（0x38）：启用 EndIntEn 中断。
7. 设置 SPI 接口时序寄存器（0x40）：将 SCLK_DIV 设为 0。

SPI 传输执行

1. 设置 DMA 控制器，将数据从内存传输到 SPI 数据寄存器（0x2C）。
2. 设置 SPI 地址寄存器（0x28）。
3. 向 SPI 命令寄存器（0x24）写入“页编程”命令（0x02），以触发 SPI 传输。该命令编码可能会因 SPI 设备而异。
4. 通过检查 SPI 中断状态寄存器（0x3C）中的 EndInt 位，等待 EndInt 中断。
5. 写入 1 以清除 SPI 中断状态寄存器（0x3C）中的 EndInt 位。

6.1.2 SPI 读操作（使用 DMA）

6.1.2.1 场景

以下示例编程序列设置控制器，以实现：

- 从两字节地址接收 16 字节数据，
- 将四个字节数据合并成一个字，
- 使用硬件握手的 DMA 数据传输，
- 在 SPI 传输结束时触发中断，
- SPI SCLK 频率为 SPI 时钟源频率的一半，
- 向 ROM 发出“读数据”命令（0x03）

6.1.2.2 编程序列

SPI 传输格式设置

1. 检查 SPI 配置寄存器（0x7C）中的 TX/RX FIFO 深度。
2. 等待之前的 SPI 传输完成，即等待 SPI 状态寄存器（0x34）中的 SPIActive 位变为 0。
3. 如下设置 SPI 传输格式寄存器（0x10）：

- AddrLen = 1（地址长度 - 1）
- DataLen = 7（数据长度 - 1）
- DataMerge = 1
- 其他字段保持复位值

4. 如下设置 SPI 传输控制寄存器（0x20）：
   - CmdEn = 1
   - AddrEn = 1
   - TransMode = 2（只读）
   - RdTranCnt = 15（总传输计数 - 1）
   - 其他字段保持复位值
5. 设置 SPI 控制寄存器（0x30）：启用 DMA，并指定 RX FIFO 阈值。
6. 设置 SPI 中断使能寄存器（0x38）：启用 EndIntEn 中断。
7. 设置 SPI 接口时序寄存器（0x40）：将 SCLK_DIV 设为 0。

SPI 传输执行

1. 设置 DMA 控制器，将数据从 SPI 数据寄存器（0x2C）传输到内存。
2. 设置 SPI 地址寄存器（0x28）。
3. 向 SPI 命令寄存器（0x24）写入“读数据”命令（0x03），以触发 SPI 传输。该命令编码可能会因 SPI 设备而异。
4. 通过检查 SPI 中断状态寄存器（0x3C）中的 EndInt 位，等待 EndInt 中断。
5. 写入 1 以清除 SPI 中断状态寄存器（0x3C）中的 EndInt 位。

6.1.3 7.1.3 通过内存映射接口发起的 SPI 活动停止

内存映射接口（AHB）将总线读操作转换为 SPI 接口上的 SPI 访问，该接口经过优化以加速顺序访问。APB SPI 控制器总是预取 SPI 设备的顺序数据字，直到 RX FIFO 满，此时通过暂停 SPI SCLK（同时保持 SPI CS 有效）来停止数据预取。如果后续内存接口的访问仍为顺序且 RX FIFO 满状态被清除，则数据传输将继续进行，而无需重新启动带有命令和地址阶段的新 SPI 请求；否则，控制器将结束当前传输，并以新的非顺序地址启动新的传输。此外，如果通过寄存器编程接口发出新请求，控制器也会结束当前传输。

为了防止 SPI CS 在长时间无传输活动的情况下持续保持断言状态，可以显式执行以下操作来取消 SPI CS 的断言，并使控制器退出活跃的内存映射访问状态（当近期不会再进行内存映射访问时）。请注意，一旦内存映射接口处于活跃状态，控制器会保护 FIFO 不受寄存器编程干扰，因此需要按以下三步程序操作：

1. 读取 SPI 内存访问控制寄存器（0x50）的当前值。

2. 将该值写回 SPI 内存访问控制寄存器（0x50）。
3. 等待直到 MemCtrlChg 变为 0。

6.2 从模式

6.2.1 接收来自 SPI 主设备的数据

6.2.1.1 场景

假设 SPI 传输格式为：

- 8 位数据宽度，
- Quad I/O 写命令（0x54），
- 传输 20 字节数据。

由于默认启用了数据合并，这将对应于对 SPI 数据寄存器（0x2C）进行 5 次数据访问。

6.2.1.2 编程序列

1. 通过向 RXFIFORST 位写入 1 来复位 RX FIFO，并等待 RXFIFORST 位清零（变为 0）。
2. 根据 RX FIFO 的配置，在 SPI 控制寄存器（0x30）中设置 RX FIFO 阈值（RXTHRES），例如设置为 RX FIFO 深度的一半。
3. 在 SPI 中断使能寄存器（0x38）中使能以下中断：从设备命令中断（SlvCmdEn）、接收 FIFO 阈值中断（RXFIFOIntEn）、传输结束中断（EndIntEn）。
4. 在中断服务程序中：
   - 通过检查 SPI 中断状态寄存器（0x3C）中的 SlvCmdInt 位，等待从设备命令中断。
     a) 根据 SPI 命令寄存器（0x24）中记录的内容，准备处理接收到的 SPI 请求。
     b) 写入 1 以清除 SPI 中断状态寄存器（0x3C）中的 SlvCmdInt 位。
   - 通过检查 SPI 中断状态寄存器（0x3C）中的 RXFIFOInt 位，等待 RX FIFO 中断。
     a) 根据设定的 RX FIFO 阈值（RXTHRES），从 SPI 数据寄存器（0x2C）弹出数据字。
     b) 写入 1 以清除 SPI 中断状态寄存器（0x3C）中的 RXFIFOInt 位。
   - 通过检查 SPI 中断状态寄存器（0x3C）中的 EndInt 位，等待传输结束中断。

a) 检查 SPI 状态寄存器（0x34）中 RX FIFO 的条目数（RXNUM）。
b) 从 SPI 数据寄存器（0x2C）弹出所有剩余条目。
c) 写入 1 以清除 SPI 中断状态寄存器（0x3C）中的 EndInt 位。

6.2.2 向 SPI 主设备传输数据

6.2.2.1 场景

在提交读取命令之前，SPI 主设备应安排从设备提前准备好传输数据。假设这是通过一个用户自定义命令完成，该命令用于指示下一次读取传输的地址偏移和数据计数。

该用户自定义命令之后紧跟一个读取状态命令（0x05），用于检查从设备处理用户自定义命令的进度（即用户自定义命令要求的数据是否已准备就绪）。

最后，SPI 主设备发出数据读取命令以启动数据传输。假设数据传输格式为：

- 8 位数据宽度，
- Quad I/O 读命令（0x0E），
- 用户自定义命令请求传输 32 字节数据。

由于默认启用了数据合并，这将对应于对 SPI 数据寄存器（0x2C）进行 8 次数据访问。

6.2.2.2 编程序列

1. 当接收到请求传输 32 字节数据的用户自定义命令时：
   - 通过向 TXFIFORST 位写入 1 来复位 TX FIFO，并等待 TXFIFORST 位清零（变为 0）。
   - 根据 TX FIFO 配置，在 SPI 控制寄存器（0x30）中设置 TX FIFO 阈值（TXTHRES），例如设置为 TX FIFO 深度的一半。
   - 准备好所需数据（8 个字 = 32 字节），并将数据字写入 SPI 数据寄存器（0x2C）直至达到 TX FIFO 阈值（TXTHRES）。
   - 在 SPI 中断使能寄存器（0x38）中使能以下中断：从设备命令中断（SlvCmdEn）、传输 FIFO 阈值中断（TXFIFOIntEn）以及传输结束中断（EndIntEn）。
   - 在 SPI 从设备状态寄存器（0x60）中设置 Ready 位。
2. 在中断服务程序中：
   - 通过检查 SPI 中断状态寄存器（0x3C）中的 SlvCmdInt 位，等待从设备命令中断。

a) 通过检查 SPI 命令寄存器（0x24）验证接收到的命令是否为读取命令。

b) 写入 1 以清除 SPI 中断状态寄存器（0x3C）中的 SlvCmdInt 位。

- 通过检查 SPI 中断状态寄存器（0x3C）中的 TXFIFOInt 位，等待 TX FIFO 中断。
  a) 将数据字写入 SPI 数据寄存器（0x2C）直至达到 TX FIFO 阈值（TXTHRES）。
  b) 如果读取传输的数据计数已全部写入 SPI 数据寄存器（0x2C），则在 SPI 中断使能寄存器（0x38）中清除传输 FIFO 阈值中断（TXFIFOIntEn），以避免冗余中断。
  c) 写入 1 以清除 SPI 中断状态寄存器（0x3C）中的 TXFIFOInt 位。
- 通过检查 SPI 中断状态寄存器（0x3C）中的 EndInt 位，等待传输结束中断。
  a) 写入 1 以清除 SPI 中断状态寄存器（0x3C）中的 EndInt 位。

6.2.3 仅数据传输

6.2.3.1 场景

当 SPI 的 CS 和时钟信号激活时，主设备和从设备会同时交换数据。这些传输仅包含数据传输阶段，没有命令和虚拟阶段。以下将描述如何在从模式下设置 APB SPI 以进行此类通信。在从模式下，数据将被加入到 RX FIFO，并从 TX FIFO 发送。

6.2.3.2 编程序列

1. 在 SPI 传输格式寄存器（0x10）中，将 MOSIBiDir 设为 0，SlvMode 设为 1，其他位设置为与 SPI 主设备传输格式匹配的适当值。
2. 在 SPI 传输控制寄存器（0x20）中，将 SlvDataOnly 设为 1，TransMode 设为 0，DualQuad 设为 0，以指示 APB SPI 在从设备仅数据模式下操作。
3. 通过将 RXFIFORST 位写为 1 来重置 RX FIFO，并等待 RXFIFORST 位清除为 0。
4. 根据 RX FIFO 的配置，在 SPI 控制寄存器（0x30）中设置 RX FIFO 阈值（RXTHRES），例如设为 RX FIFO 深度的一半。

5. 向 TX FIFO 写入适当的数据，以匹配与 SPI 主设备的协议。
6. 在 SPI 中断使能寄存器（0x38）中，启用接收 FIFO 阈值中断（RXFTHIE）、发送 FIFO 阈值中断（TXFTHIE）和传输结束中断（EndIntEn）。
7. 在中断服务程序中：
   - 通过检查 SPI 中断状态寄存器（0x3C）的 RXFIFOInt 位，等待 RX FIFO 中断。
     a) 根据设置的 RX FIFO 阈值（RXTHRES），从 SPI 数据寄存器（0x2C）中弹出数据字。
     b) 通过将 SPI 中断状态寄存器（0x3C）的 RXFIFOInt 位写为 1 来清除该位。
   - 通过检查 SPI 中断状态寄存器（0x3C）的 TXFIFOInt 位，等待 TX FIFO 中断。
     a) 向 SPI 数据寄存器（0x2C）写入数据字，直到满足 TX FIFO 阈值（TXTHRES）。
     b) 如果读取传输的数据已完全填充到 SPI 数据寄存器（0x2C）中，清除 SPI 中断使能寄存器（0x38）中的发送 FIFO 阈值中断（TXFIFOIntEn），以避免冗余中断。
     c) 将 SPI 中断状态寄存器（0x3C）的 TXFIFOInt 位写 1，以清除该中断标志。
   - 通过检查 SPI 中断状态寄存器（0x3C）的 EndInt 位，等待传输结束中断。
     a) 如果接收到数据，检查 SPI 状态寄存器（0x34）的 RX FIFO 条目数（RXNUM），并从 SPI 数据寄存器（0x2C）中读取所有剩余条目。
     b) 将 SPI 中断状态寄存器（0x3C）的 EndInt 位写 1，以清除该中断标志。

注！

如果从设备仅需要在某一时刻接收或发送数据，则可以跳过与另一方向相关的操作。尽管在这种情况下可能发生 TX FIFO 下溢或 RX FIFO 上溢，但可以忽略或禁用相应的中断。

7 参考设计

详细信息请参见高云半导体网站 Gowin APB SPI IP 相关参考设计：

- 硬件参考设计：
  - …\ref_design\FPGA_RefDesign\DK_START_GW5AST138_V1.0\apb_spi
  - …\ref_design\FPGA_RefDesign\DK_START_GW5AST138_V1.0\apb_spi_flash_mem
  - …\ref_design\FPGA_RefDesign\DK_START_GW5AST138_V1.0\apb_spi_flash_reg
- 软件参考设计：

  …\ref_design\MCU_RefDesign\apb_spi


GOWIN高云
智慧逻辑 定制未来