Gowin CAN Controller IP
用户指南

IPUG527-1.2.2, 2024-10-25

版权所有 © 2024 广东高云半导体科技股份有限公司

GOWIN高云、GOWIN、、GOWINSEMI、GOWIN、Gowin、高云、晨熙、小蜜蜂、LittleBee、Arora-V、GowinPnR、*GowinSynthesis*、GoBridge 均为广东高云半导体科技股份有限公司注册商标，本手册中提到的其他任何商标，其所有权利属其拥有者所有。未经本公司书面许可，任何单位和个人都不得擅自摘抄、复制、翻译本文档内容的部分或全部，并不得以任何形式传播。

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
2019/01/08	1.0	初始版本。
2019/06/28	1.1	增加 APB、Wishbone、REG 总线接口。
2020/06/01	1.2	完善 CAN IP 的功能描述。
2023/12/29	1.2.1	更新表 5-1。
2024/10/25	1.2.2	新增“Support CAN FD”选项。

目录

图目录.....iii
表目录.....iv
1 关于本手册.....1
1.1 目的.....1
1.2 相关文档.....1
1.3 术语、缩略语.....2
1.4 技术支持与反馈.....2
2 概述.....3
3 特征与性能.....4
3.1 主要特征.....4
3.2 工作频率.....5
3.3 资源利用.....5
4 结构及功能描述.....6
4.1 模块结构.....6
4.2 功能说明.....7
4.3 功能框图.....7
4.4 功能框图说明.....8
4.4.1 位时序逻辑（BTL）.....8
4.4.2 比特流处理（BSP）.....8
4.4.3 波特率预分频器（BRP）.....8
4.4.4 Rx 接收滤波器.....8
4.4.5 共享缓冲区.....8
4.4.6 传输调度程序.....13
4.4.7 配置和状态寄存器（CSR）.....14
4.4.8 CPU 接口.....14
5 接口列表.....17
5.1 PIN 引脚图.....17
5.2 引脚说明.....17
6 寄存器.....21
7 中断.....31

8 时钟.....32
9 Reset 复位.....33
10 初始化序列示例.....34
10.1 上电.....34
10.2 Initialization 初始化.....34
11 接口配置.....35
12 参考设计.....36

图目录

图 4-1 模块结构.....6
图 4-2 CAN Controller IP 功能框图.....7
图 4-3 共享缓冲区图示.....9
图 4-4 接收缓冲区/ FIFO 结构和格式.....10
图 4-5 发送高优先级缓冲区/ FIFO 结构和格式.....12
图 4-6 REG 写操作立即响应.....14
图 4-7 REG 写操作延时响应.....15
图 4-8 REG 读操作立即响应.....15
图 4-9 REG 读操作延时响应.....16
图 5-1 CAN 引脚图.....17
图 8-1 时钟结构.....32
图 11-1 CAN 配置界面.....35

表目录

表 1-1 术语、缩略语.....2
表 2-1 Gowin CAN Controller IP 概述.....3
表 3-1 资源利用.....5
表 4-1 rxbhdr1 格式.....10
表 4-2 rxhdr2 格式.....11
表 4-3 txhdr1 格式.....13
表 4-4 txhdr2 格式.....13
表 5-1 引脚说明.....17
表 6-1 寄存器的地址映射和描述.....21

1 关于本手册

1.1 目的

Gowin CAN Controller IP 用户指南主要包括功能简介、功能描述界面配置和参考设计，旨在帮助用户快速掌握 Gowin CAN Controller IP 的功能及使用方法。本手册中的软件界面截图参考的是 1.9.10.03 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.cn* 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*
- *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*
- *DS1228，Arora V FPGA 产品概述*
- *SUG100, Gowin 云源软件用户指南*

1.3 术语、缩略语

表 1-1 中列出了本手册中出现的相关术语、缩略语及相关释义。

表 1-1 术语、缩略语

术语、缩略语	全称	含义
CAN	Controller Area Network	控制器局域网络
FPGA	Field Programmable Gate Array	现场可编程门阵列
IP	Intellectual Property	知识产权
LUT	Look-up Table	查找表

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或建议，可直接与公司联系：

网址：*www.gowinsemi.com*

E-mail：*support@gowinsemi.com*

Tel：+86 755 8262 0391

2 概述

本文档描述了控制器局域网（CAN）控制器 IP，它实现了 CAN2.0A，CAN2.0B 以及更新的高性能非 ISO CAN-FD 协议。它可以集成到需要 CAN 连接的设备中，这种连接通常用于汽车和工业应用。

表 2-1 Gowin CAN Controller IP 概述

Gowin CAN Controller IP	
逻辑资源	见表 3-1
交付文件	
设计文件	Verilog (加密)
参考设计	Verilog
测试平台	Verilog
测试设计流程	
综合软件	GowinSynthesis
应用软件	Gowin Software

注！

可登录*高云半导体网站*查看芯片支持信息。

3 特征与性能

3.1 主要特征

- 符合 CAN2.0A 和 CAN2.0B 协议及 ISO 11898-1 标准
- 支持 CAN-FD
  - 非 ISO CAN FD
  - 符合博世（Bosch）协议
  - 符合 ISO/DIS 11898-1
- 独立系统时钟（SYSCLK）和 CAN 总线时钟（CANCLK）
  - SYSCLK 是 CPU 接口总线时钟和 IP 系统时钟，需为连续时钟，不可中断
  - CANCLK 可以是独立的，也可以与 SYSCLK 相连
- 灵活的共享缓冲方案，实现最佳缓冲区大小，以便在给定的应用程序中存储发送和接收消息
  - 总缓冲区深度在生成 IP 时，可通过参数配置
  - 发送缓冲区，接收缓冲区和高优先级发送缓冲区
  - CPU 可分别配置每个缓冲区的深度， 用于发送，接收和高优先级发送缓冲区
- 接收滤波器可配置为 1-16 个
- 支持 4 种方式连接 CPU
  - AHB 从接口
  - APB 从接口
  - Wishbone 从接口
  - REG 从接口
- 可编程波特率预分频器（BRP）
  - 根据 CANCLK 生成 TQ CLK
  - 8 位 BRP 寄存器支持 4 分频到 255 分频

3.2 工作频率

系统支持 2 个时钟：sysclk 和 canclk。sysclk 通常为高频时钟，例如： 50MHz。Canclk 是 CAN 总线时钟，TQ 时钟从 canclk 分频而来，具体取决于 CAN 数据位宽。在 IP 设计中，sysclk 和 canclk 完全作为异步时钟处理，不要求具有确定的相位关系。一般应用中 sysclk≥canclk 即可。

3.3 资源利用

Gowin CAN Controller IP 采用 Verilog 语言，该 IP 用于 GW1N-9、GW1NR-9、GW2A 系列和 GW2AR 系列 FPGA 器件中。不同的模式有不同的资源消耗。

表 3-1 给出了资源利用的概述。关于其它器件的资源利用请参阅相关的后期发布信息。

表 3-1 资源利用

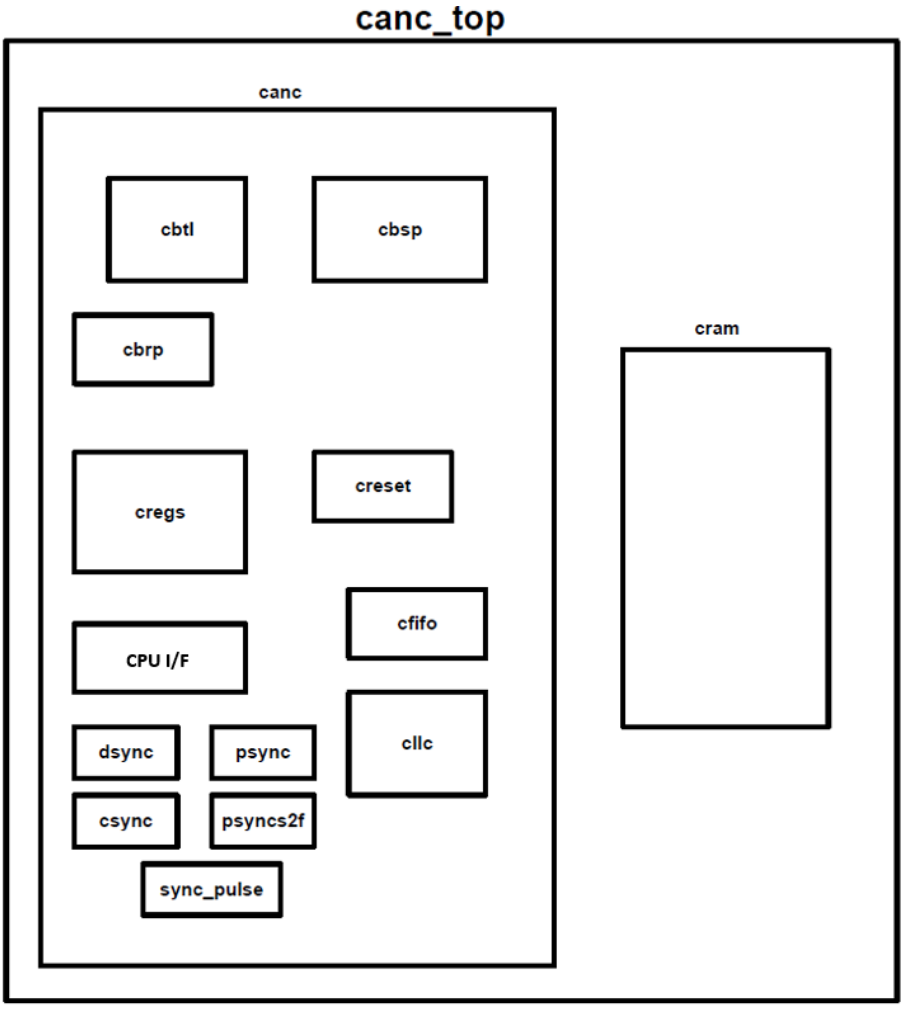
LUTs	REGs	Block Rams	Device Series	Speed Level
4109	2217	1	GW2A18	-8

4 结构及功能描述

4.1 模块结构

IP 的内部模块结构如图 4-1 所示。

图 4-1 模块结构

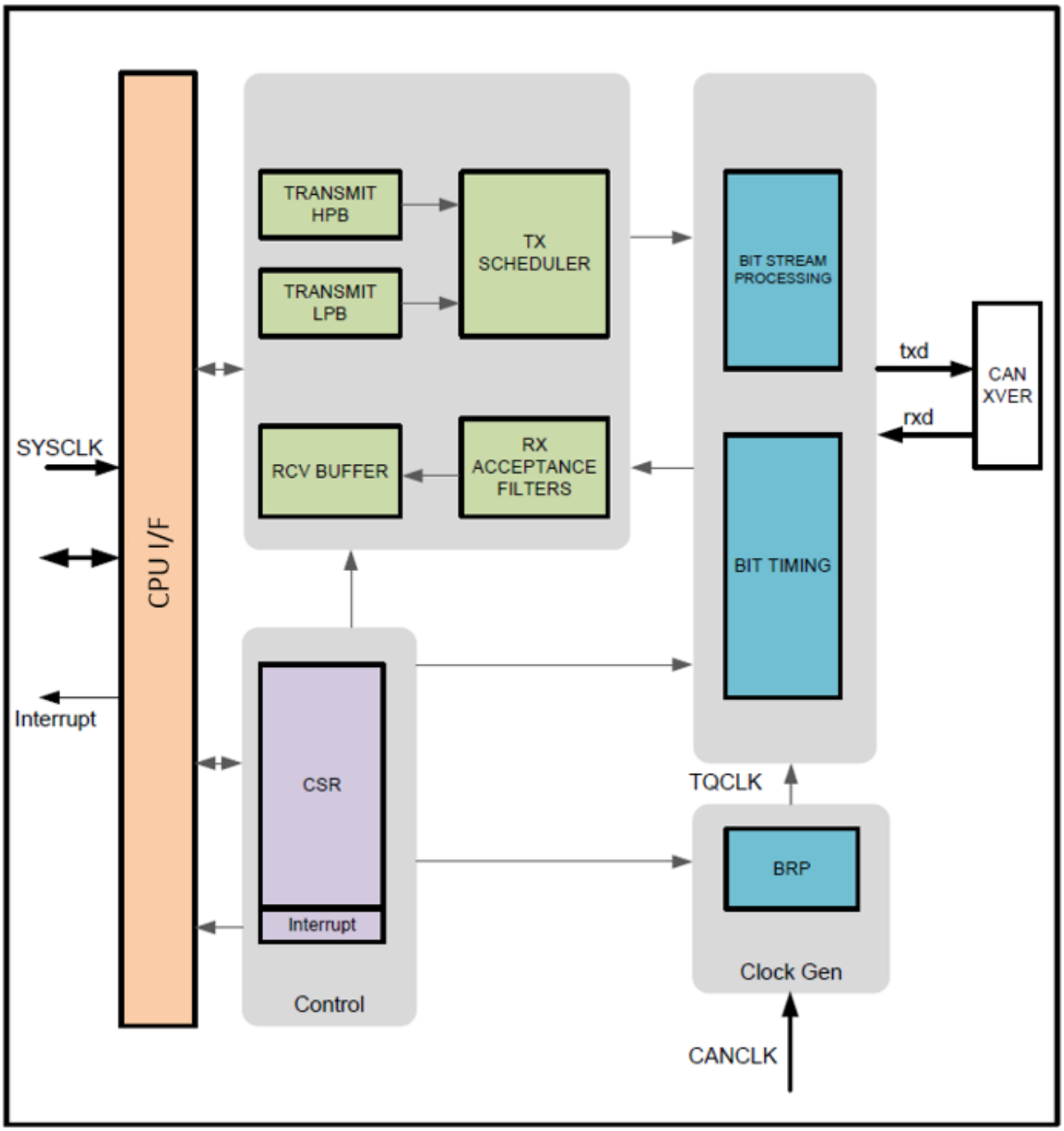


4.2 功能说明

Gowin CAN Controller IP 旨在简化汽车和工业应用中各器件的 CAN 协议的实施。

4.3 功能框图

图 4-2 CAN Controller IP 功能框图



4.4 功能框图说明

4.4.1 位时序逻辑（BTL）

位时序逻辑的主要功能是使用同步段、传播段和相位段对控制器及 CAN 通信进行位同步。用 TQ 时钟以 TQ 时间为单位测量这些段，TQ 时钟由波特率预分频器（BRP）通过将 CANCLK 除以可编程值生成。一旦实现同步，它就生成比特流处理（BSP）样本参考用于更高级别的帧解码功能。

4.4.2 比特流处理（BSP）

BSP 块执行各种更高级协议功能，通常归类为 MAC 功能，包括接收方向上的 CRC 位和填充位的检查和删除。类似地，BSP 块在发送方向上串行化数据流，根据需要添加填充位并计算和添加 CRC 位。仲裁后，它将完整的帧传输到 CAN 总线。如果在仲裁过程中丢失了访问权限，它会自动重新传输总线。

4.4.3 波特率预分频器（BRP）

波特率预分频器（BRP）模块通过将输入 CANCLK 除以编程的预分频值 BRP 来生成 TQ。该寄存器为 8 位宽，最多支持 255 分频。BTL 使用 TQ 时钟对比特流进行对齐和采样。

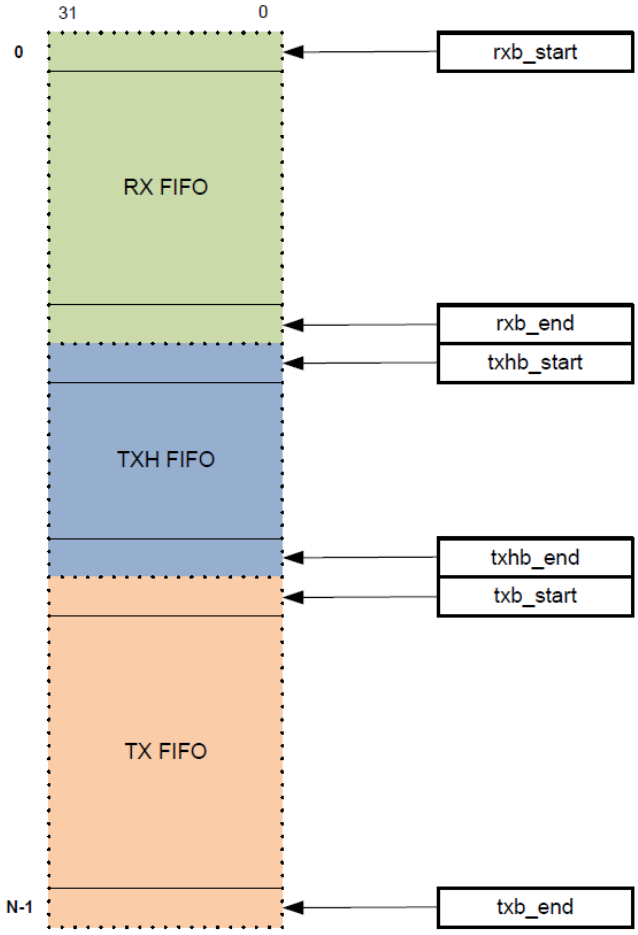
4.4.4 Rx 接收滤波器

Rx 接收滤波器通过将每条消息中的 11 位或 29 位标识符与接收滤波器寄存器（AFR）的值以及相应的接收滤波器掩码（AFM）进行比较来过滤输入帧。通过任何滤波器的消息都被接受并存储在 Rx 缓冲区中。AFR 和 AFM 由 CPU 单独配置。滤波器的数量可配置为 1-16。

4.4.5 共享缓冲区

CAN 控制器 IP 实现共享缓冲方案，以优化各种应用区域。单个存储 RAM 被软件分区为三个独立的循环 FIFO。这三个分区充当接收缓冲区（RXB）、传输高优先级缓冲区（TXHB）和传输缓冲区（TXB）。当禁用控制器时（CMD.enable =0），每个分区的大小由 CPU 配置。这为缓冲区空间优化提供了极大的灵活性，具体取决于 SoC 的应用。字节以 big-endian 格式存储，其中字节 0 存储在 31:24 位，字节 1 存储在 23:16 位，字节 2 存储在 15:8 位，字节 3 存储在 7:0 位，如图 4-3 所示。 以下小节描述了各个 FIFO 的格式和功能。

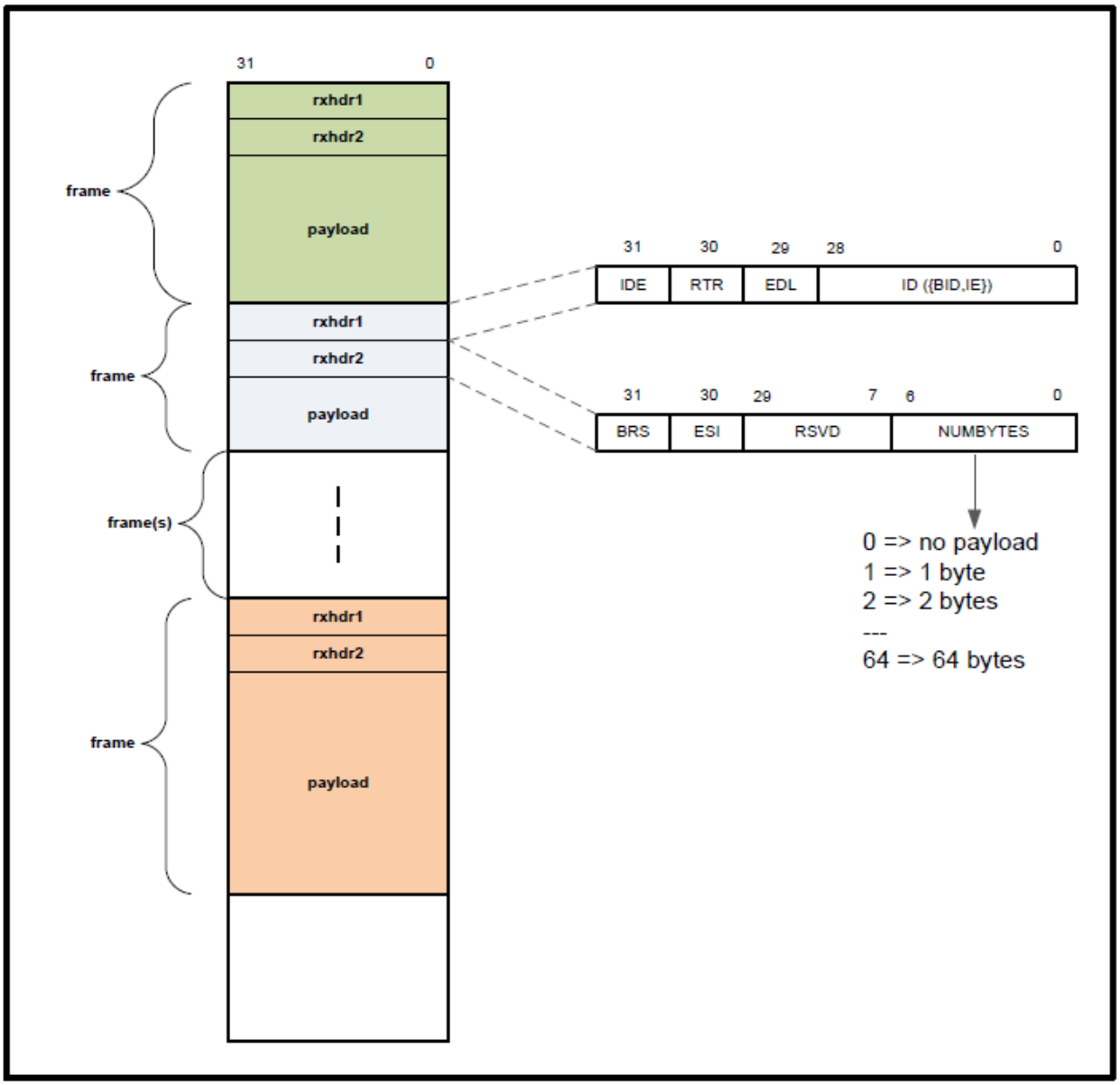
图 4-3 共享缓冲区图示



接收缓冲区

接收缓冲区（RXB）存储所接收的无错误且通过了接收滤波器的帧。CPU 通过 AHB 接口从该接收缓冲区读取这些帧。每个消息包含一对 32 位标头，该标头用来指示消息标识符、长度和其他标志，因此 CPU 知道要读取多少字节。它们通过 32 位窗口寄存器以先进先出的方式提供给 CPU。接收缓冲区深度可在运行时进行配置。存储的消息数量取决于各个消息的深度和长度。图 4-4 表示 Rx FIFO 的结构和格式。

图 4-4 接收缓冲区/ FIFO 结构和格式



可以看出，控制器写入两个 32 位标头; rxhdr1，rxhdr2 位于每个接收帧的头部。这些标头包含帧长度和接收的其他控制位。CPU 使用 rxhdr2 的 NUMBYTES 字段来了解在 rxhdr2 之后还需读取/弹出多少字节以获得下一个接收帧。读取整个帧后，CPU 检查 Rx 缓冲区状态（RXBSTS.rxbdepth）以查看是否有更多已接收的帧。或者，CPU 可以等待 RCVFVLD 中断。表 4-1 和表 4-2 分别描述了 rxhdr1，rxhdr2 的字段。

表 4-1 rxbhdr1 格式

31	30	29	28:0
IDE	RTR	EDL	ID
标识符扩展	远程传输请求 反映收到的 RTR 位	扩展数据长度 它是帧中接收的 EDL 位。	29 位标识符 28:18 => BID

31	30	29	28:0
1=>扩展帧，带有 29 位标识符 0=>普通帧，带有 11 位标识符	1 =>远程传输请求 0 =>带有效载荷的普通帧	1 => FD 格式帧 0 =>普通帧	17:0 => IE 只有 28:18 位对普通帧有效

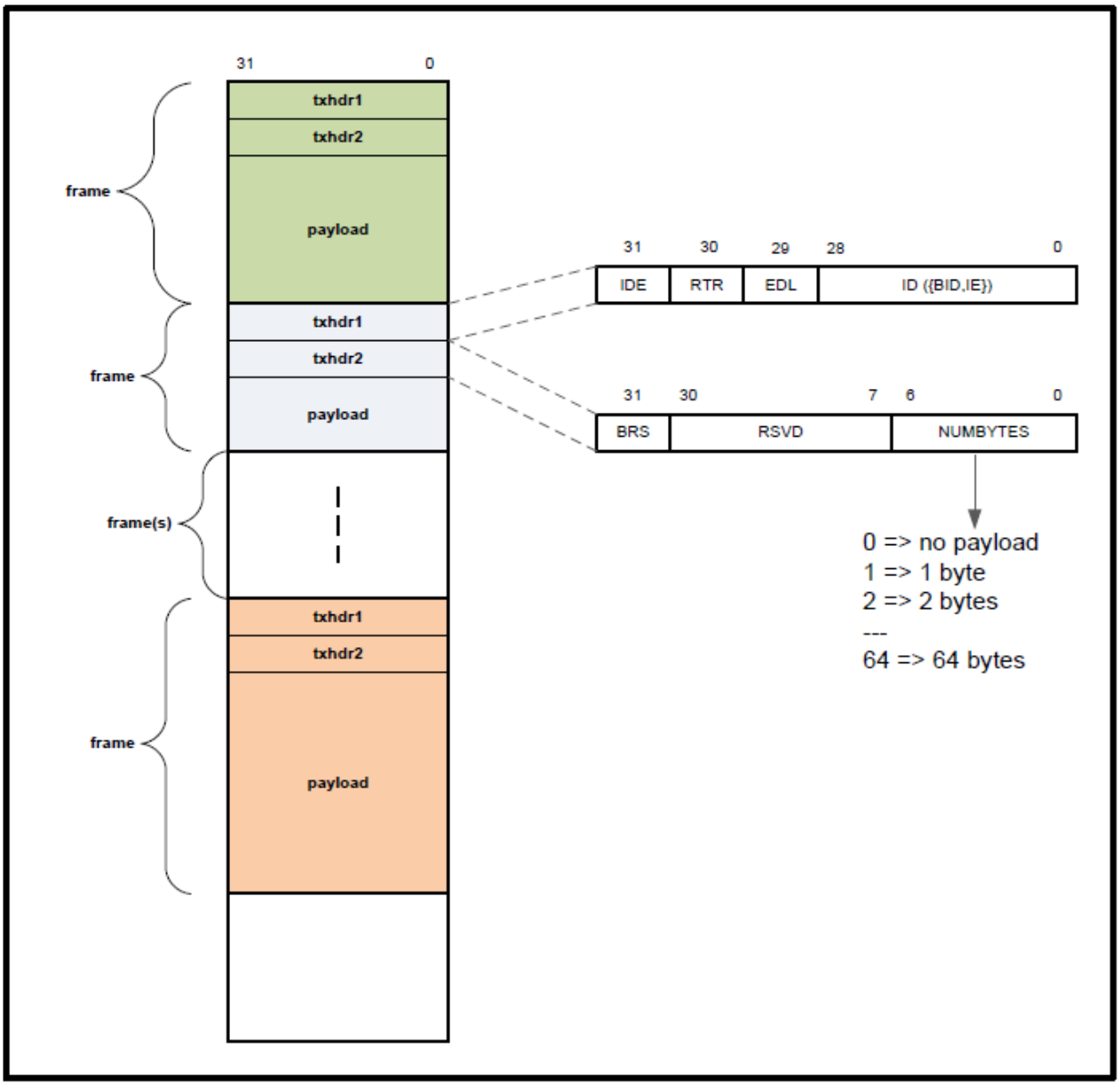
表 4-2 rxhdr2 格式

31	30	29:7	6:0
BRS	ESI	RSVD	NUMBYTES
比特率开关 仅存在于 FD 格式帧中。反映收到的 BRS 位。 1 =>在数据阶段，比特率切换到预先配置的备用速率 0 =>在数据阶段，比特率是标称的（未切换）	错误状态指示器 仅存在于 FD 格式帧中。反映收到的 ESI 位。 1 => 消息的发送器处于被动错误状态。 0 => 消息的发送器处于主动错误状态。	保留。 目前尚未使用。	帧有效载荷长度，以字节为单位 指示接收帧的有效载荷字节数，也指示 CPU 需要读取的 rxhdr2 以外的字节数。请注意，消息以 32 位字存储，CPU 读取完整字，并根据长度去掉最后一个字中的尾随字节（如果有的话）。 0 =>没有有效载荷 RTR 帧没有有效载荷 1 => 1 个字节 2 => 2 个字节 ---- 64 => 64 个字节 其他=>未使用

发送高优先级缓冲区

发送高优先级缓冲区（TXHB）用于存储 CPU 希望以更高优先级发送的高优先级消息。它作为循环 FIFO 运行。相较于存储在低（正常）优先级缓冲区中的消息，发送调度程序将先处理这些高优先级消息。TXHB 的深度可由 CPU 配置。可以存储在 TXHB 中的消息数取决于各个消息的配置深度和大小。与 Rx 缓冲区类似，TXHB 缓冲区/FIFO 中的每条消息都有两个标头，这两个标头中包含帧长度和控制器所需的其他控制信息。CPU 写入两个标头; txhdr1，txhdr2 在每个数据包的有效载荷之前。图 4-5 表示存储在 THPB 中的消息的结构和格式。

图 4-5 发送高优先级缓冲区/ FIFO 结构和格式



如图所示，CPU 写入两个 32 位标头; txhdr1，txhdr2 位于每个接收帧的头部。这些标头包含帧长度和接收的其他控制位。控制器使用 txhdr2 的 NUMBYTES 字段来获取在 txhdr2 之外还需读取/弹出多少字节以到达下一个发送帧的末尾。在写入/推送下一帧之前，CPU 将检查是否有足够空间来接收下一帧。可在 TXHBSTS.txhbspace 中获取该状态。读取的空间应大于等于要传输的下一帧的有效载荷的大小，四舍五入到字+2（以考虑两个标头）。可以在每帧的写入结束时检查 TXBSTS.txbwerr 以查看该帧是否导致 FIFO 溢出。如果在 CPU 将该帧推送至 FIFO 期间发生溢出，则丢弃该帧。表 4-3 和表 4-4 分别描述了 txhdr1，txhdr2 的字段。

表 4-3 txhdr1 格式

31	30	29	28:0
IDE	RTR	EDL	ID
标识符扩展指示控制器是以扩展格式还是以普通格式发送帧。 1=>扩展帧，带有 29 位标识符 0=>普通帧，带有 11 位标识符	远程传输请求指示它是否是远程传输请求且是否没有有效载荷。 1 =>远程传输请求 0 =>带有效载荷的普通帧	扩展数据长度，为帧中接收的 EDL 位。 指示控制器是以 FD 格式还是以普通格式发送帧 1 => FD 格式帧 0 =>普通帧	29 位标识符。 28:18 => BID 17:0 => IE 只有 28:18 位对普通帧有效

表 4-4 txhdr2 格式

31	20:7	6:0
BRS	RSVD	NUMBYTES
比特率开关仅存在于 FD 格式帧中。 指示在数据阶段是否切换了比特率。 1 =>在数据阶段，比特率切换到预先配置的备用速率 0 =>在数据阶段，比特率是标称的（未切换）	保留。 目前尚未使用。	帧有效载荷长度，以字节为单位 指示接收帧的有效载荷字节数，也指示 CPU 需要写入的 rxhdr2 以外的字节数。 请注意，消息以 32 位字存储，CPU 读取完整字，并根据长度去掉最后一个字中的尾随字节（如果有的话）。 0 =>没有有效载荷 RTR 帧没有有效载荷 1 => 1 个字节 2 => 2 个字节 ---- 64 => 64 个字节 OTHERS =>未使用

发送缓冲区（TXB）

发送缓冲区（TXB）存储了被 CPU 标记为正常优先级的数据帧。从该缓冲区输出的消息依据 FIFO 序列进行处理并以较低优先级调度，即，如果 THPB 没有任何完整消息则传送。TXB 的深度也可由 CPU 配置。发送缓冲区/ FIFO 的结构和格式类似于上一节中描述的发送高优先级缓冲区。

4.4.6 传输调度程序

Tx 调度程序在 TB 和 THPB 消息之间实现严格优先级（SP）调度，其中 THPB 中的消息具有更高的优先级。来自两个缓冲区的消息都是先进先出的。

4.4.7 配置和状态寄存器（CSR）

配置和状态寄存器（CSR）实现各种命令、控制和状态寄存器，用于对控制器进行操作并将状态传递给 CPU。另外，还包括中断状态寄存器（ISR）。

4.4.8 CPU 接口

用户可以配置 CPU 接口为 AHB 从接口、APB 从接口、Wishbone 从接口和 REG 接口之一。其中 AHB 从接口、APB 从接口和 Wishbone 从接口符合相关协议。

REG 接口分为立即响应和延时响应。立即响应：在 cpu_cs 和 cpu_write/cpu_read 有效的同时，ip 即可给出响应。延时响应：在 cpu_cs 和 cpu_write/cpu_read 有效的 n 个周期之后，ip 才会给出响应。立即响应和延时响应用户无法配置，取决于用户读取寄存器的地址。

REG 接口操作时序如下所示。

图 4-6 REG 写操作立即响应

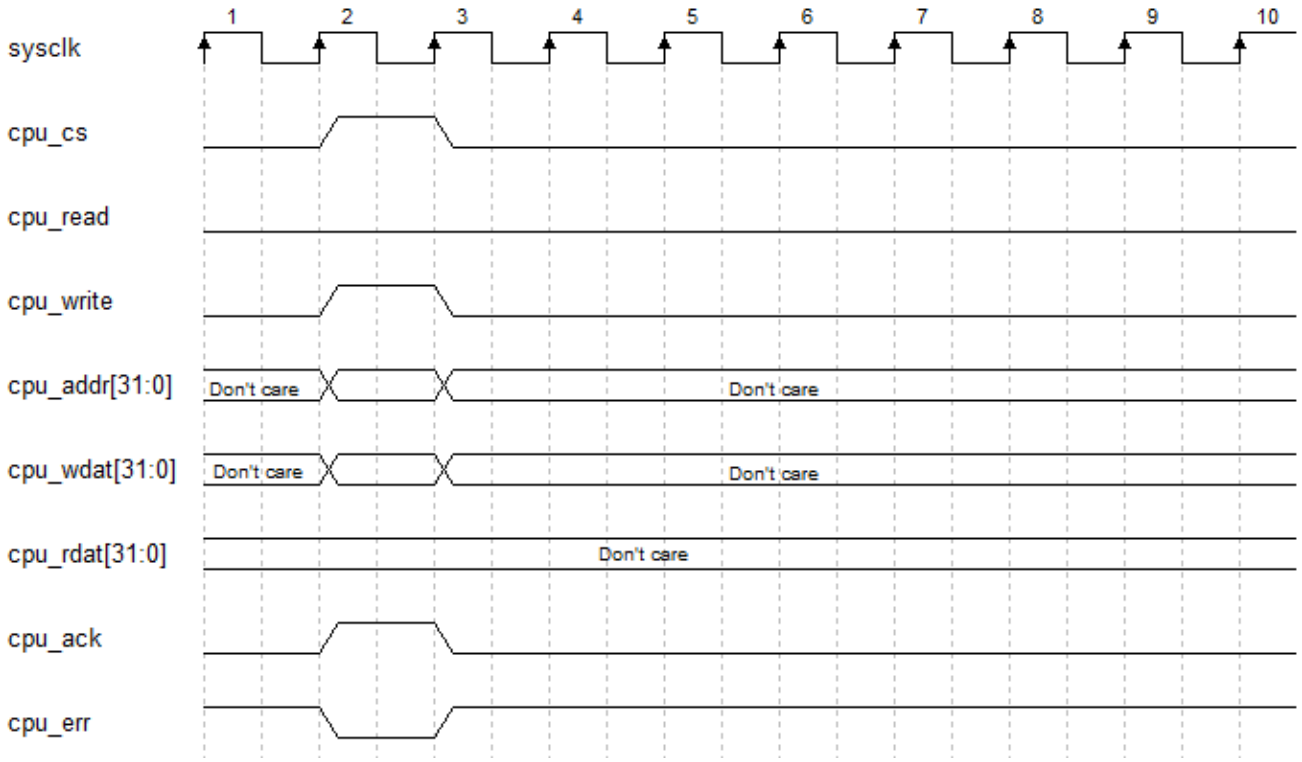


图 4-7 REG 写操作延时响应

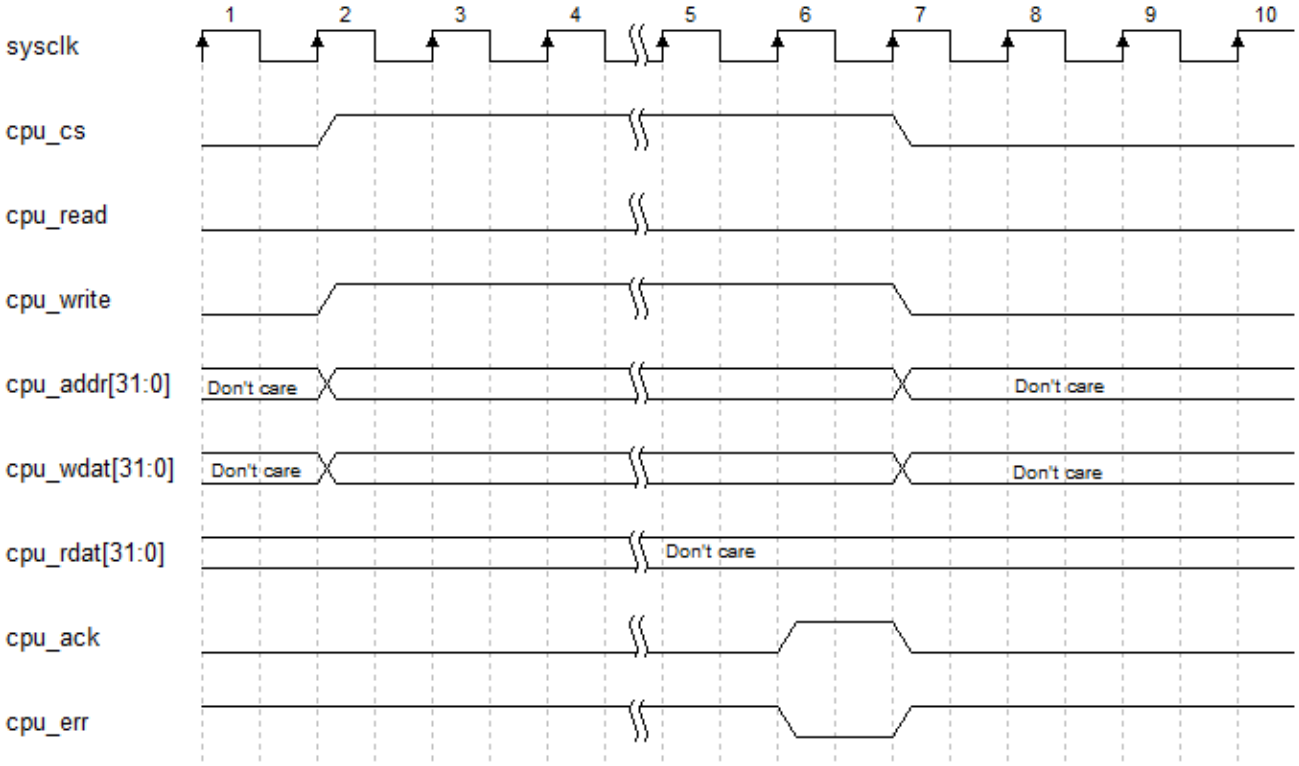


图 4-8 REG 读操作立即响应

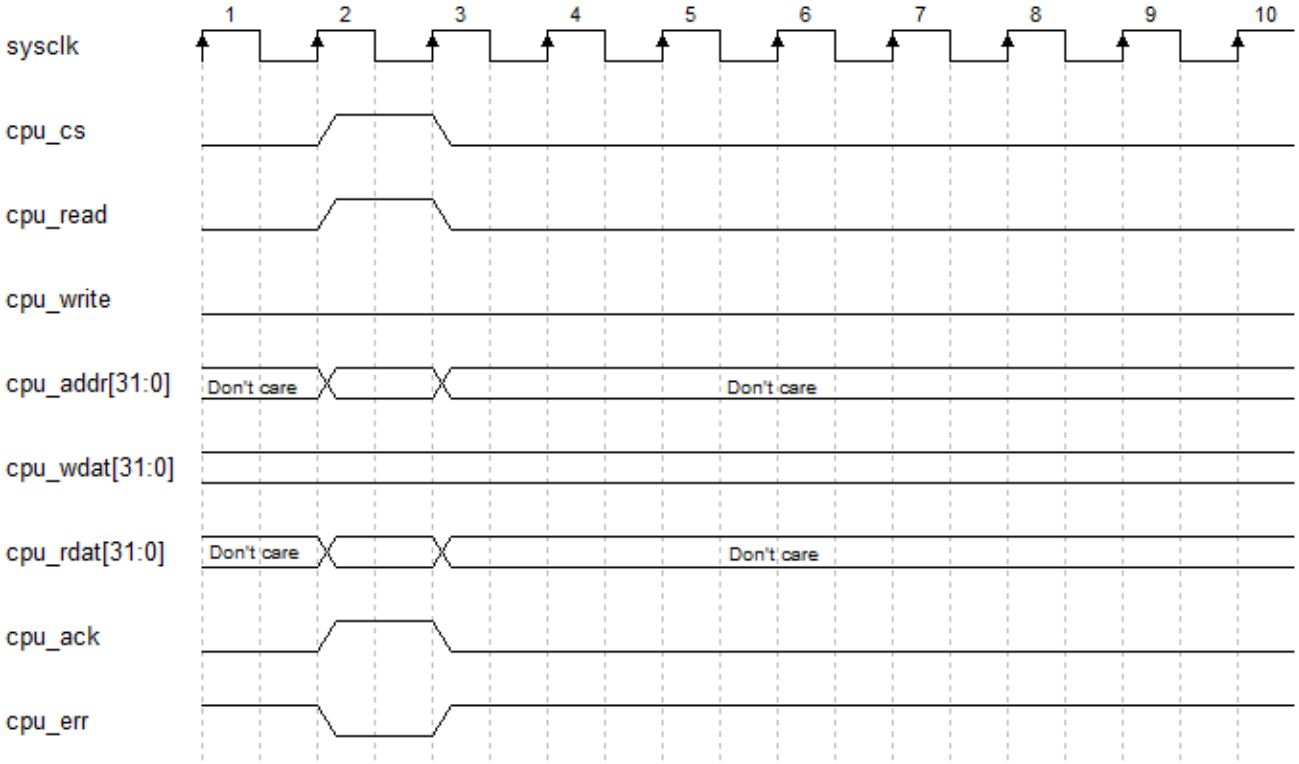
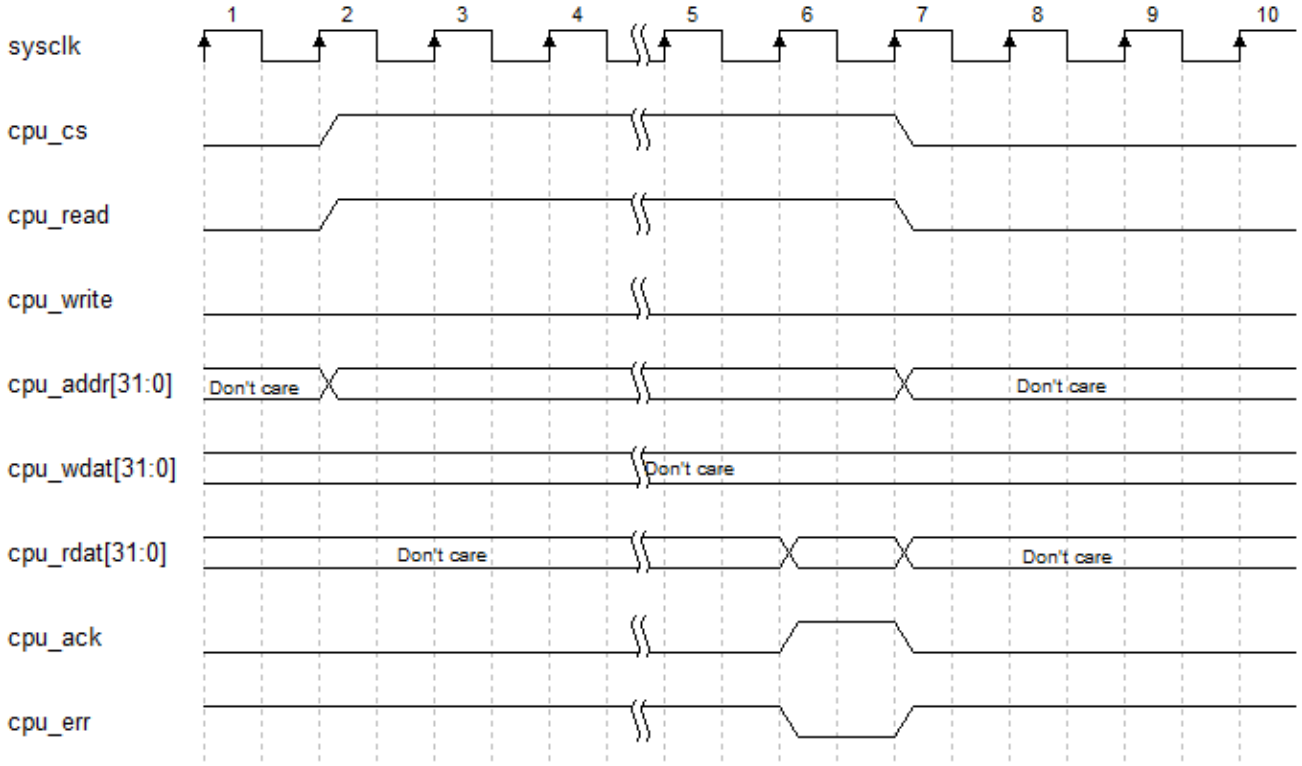


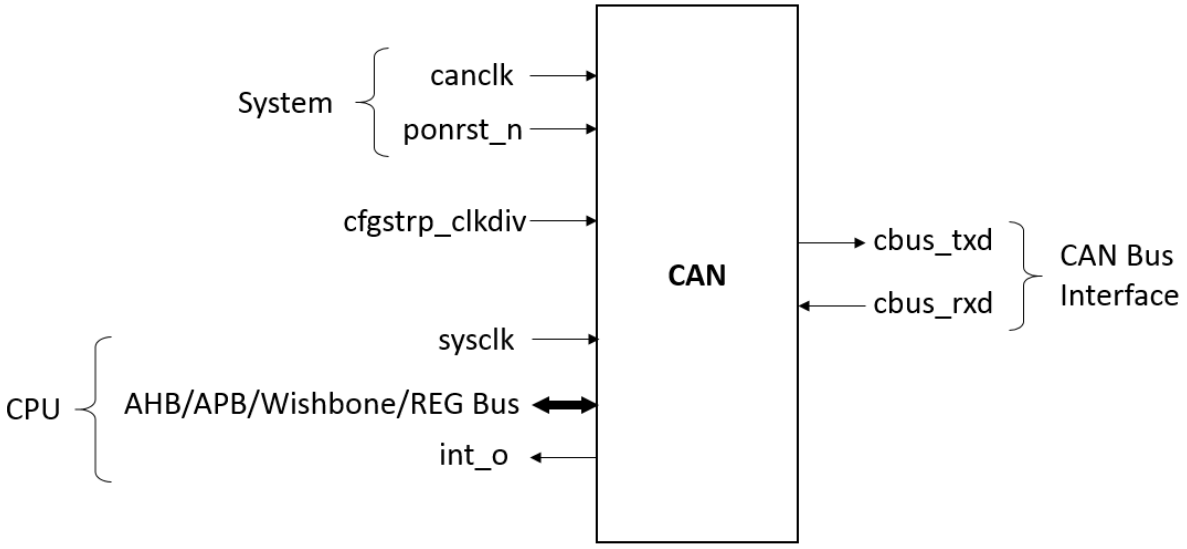
图 4-9 REG 读操作延时响应



5 接口列表

5.1 PIN 引脚图

图 5-1 CAN 引脚图



5.2 引脚说明

表 5-1 引脚说明

引脚	方向	描述
系统接口		
sysclk	输入	系统时钟输入。作为 CPU 接口时钟。 如果 CPU 是 ARM 处理器，则时钟通常情况下是 AHB 时钟。所有的 CSR 和消息的 FIFO 在 sysclk 域实现快速 CPU 访问。独立于用于 CAN 总线接口的 canclk，两者之间没有假定相位关系，满足 sysclk≥canclk 即可。
canclk	输入	CAN 总线接口时钟。

引脚	方向	描述
canclk	输入	用作内部预分频器的输入时钟，产生 TQ 时钟，该 TQ 时钟用于采样以及通过 CAN 总线传输比特流。canclk 时钟频率应至少是 TQ 时钟频率的 4 倍
ponrst_n	输入	低有效上电复位。应该在至少 4 sysclk 时钟周期内保持有效。如果 AHB 是主机 CPU 接口，则可以绑定到 hreset_n。
cfgstrp_clkdiv[7:0]	输入(Pin Strap 引脚带)	由引脚初始化的时钟分频。 解释如下： 0..3 =>保留 4 => div-by-4 5 => div-by-5 --- --- 255 => div-by_255 注！cfgstrp_clkdiv 在上电复位期间被读取并用作初始分频器值。CPU 可以通过写入 BRP（波特率预分频器）寄存器来改变频率。
CAN Bus 接口		
cbus_rxd	输入	来自收发器的 CAN 总线串行数据单端输入。 连接到收发器的 RXD 输出引脚。
cbus_txd	输出	CAN 总线串行数据单端输出。 连接到收发器的 TXD 输入引脚。
CPU 总线接口 - AHB Slave		
sysclk	输入	AHB 系统时钟。 在该 AHB 系统时钟中对所有 AHB 时钟域信号进行采样和驱动。此外，内部 CSR 和消息缓冲区也使用系统时钟来加快 CPU 访问速度。
t_hsel	输入	从设备选择
t_haddr[13:0]	输入	字节地址
t_hwdata[31:0]	输入	写数据
t_hrdata[31:0]	输出	读数据
t_hwrite	输入	写/读方向指示
t_hsize[1:0]	输入	大小（字节- 8 位，半字- 16 位，字-32 位）
t_htrans[1:0]	输入	传输类型 2’b00 -空闲 2’b10 -非顺序
t_hresp[1:0]	输出	传输响应 2’b00 –成功

引脚	方向	描述
t_hresp[1:0]	输出	IP 不发送错误响应
t_hready	输出	传输完成
t_hready_in	输入	从接口 ready 信号输入
CPU 总线接口 - APB Slave		
sysclk	输入	APB 系统时钟。 在该 APB 系统时钟中对所有 APB 时钟域信号进行采样和驱动。此外，内部 CSR 和消息缓冲区也使用系统时钟来加快 CPU 访问速度。
PADDR[13:0]	输入	字节地址
PWDATA[31:0]	输入	写数据
PRDATA[31:0]	输出	读数据
PSEL	输入	从设备选择
PWRITE	输入	写/读方向指示
PENABLE	输入	使能信号
PREADY	输出	准备好
PSLVERR	输出	传输失败
CPU 总线接口 - Wishbone Slave		
sysclk	输入	Wishbone 系统时钟。 在该 Wishbone 系统时钟中对所有 Wishbone 时钟域信号进行采样和驱动。此外，内部 CSR 和消息缓冲区也使用系统时钟来加快 CPU 访问速度。
CYC_I	输入	总线周期信号
STB_I	输入	选通信号
WE_I	输入	写/读方向指示
ADR_I[13:0]	输入	地址信号
DAT_I[31:0]	输入	写数据
ACK_O	输出	确认信号
DAT_O[31:0]	输出	读数据
CPU 总线接口 - REG		
sysclk	输入	REG 系统时钟。 在该 REG 系统时钟中对所有 REG 时钟域信号进行采样和驱动。此外，内部 CSR 和消息缓冲区也使用系统时钟来加快 CPU 访问速度。
cpu_cs	输入	片选信号

引脚	方向	描述
cpu_read	输入	读使能
cpu_write	输入	写使能
cpu_addr[13:0]	输入	地址信号
cpu_wdat[31:0]	输入	写数据
cpu_rdat[31:0]	输出	读数据
cpu_ack	输出	传输完成响应
cpu_err	输出	传输错误指示
CPU 中断信号		
int_o	输出	中断信号，高有效

6 寄存器

CAN 控制器 IP 使用 64KB 地址空间，因此将较低 16 位的 CPU /系统总线地址解码为寄存器组的偏移值。它将所有控制和状态寄存器以及用于访问 Tx 和 Rx 缓冲区的窗口寄存器映射到此地址空间。所有寄存器 DWORD（32 位）可访问。寄存器的地址映射和描述如表 6-1 所示。

表 6-1 寄存器的地址映射和描述

偏移/范围	寄存器名称	访问限制	描述
0x0000	SRST	无	软件复位寄存器 CPU 写“1”到 bit0 启动控制器的硬复位，类似上电复位可复位整个控制器。 写'0'取消复位。 0 =>软复位 1 => 完全重置控制器，包括配置和状态寄存器。BRP 未重置并继续生成 tqclk 0 =>复位被取消 Others => RSVD 注！内部软复位与 tqclk 域同步，tqclk 域通常以低得多的频率运行，可以是 KHz 的量级。因此，CPU 必须在较长时间内保持 SRST 有效，以完全重置控制器。最短持续时间至少为 4 tqclk 时钟周期。在进行重新初始化之前，它必须在失效后再等待 4 个 tqclk 时钟周期。由于控制 tqclk 时钟周期的 BRP 寄存器本身在软复位期间复位，因此默认为（canclk 频率/ BRP.brp [7：0]）。 寄存器的 BRP.brp [7：0]默认值设置为 0x4（canclk 频率的 div-by-4）
0x0004	CMD	无	命令寄存器 0 =>使能（默认值：0x0） 1'b1 :工作模式 1'b0 :命令模式

偏移/范围	寄存器名称	访问限制	描述
0x0004	CMD	无	Others => RSVD 写入'1'至使能位会使控制器进入工作状态。在配置运行参数（例如 BRP，BTN 和 BTD）之后，将其写入“1”。
0x0008	BRP	无	波特率预分频寄存器。 7:0 => brp (默认 : 0x0) Others => RSVD 用于通过整数除法从 canclk 生成的 TQ 时钟的预分频值。 8'd0, 8'd1, 8'd2, 8'd3 =>保留 (默认情况下选择 div_by_4) 8'd4 => div-by-4 8'd5 => div-by-5 --- 8'd255 => div-by-255 注！在通过向 CMD.enable 位写入 1 来使能控制器之前对其进行配置
0x000C	BTN	无	位定时（标称）寄存器。 该寄存器定义了标称或正常速率的位定时参数。单位为 TQ 时钟周期。 5:0 => phseg1_nom (默认: 0x0) 定义 PROP_SEG + PHASE_SEG1 窗口的宽度。 13:8 => phseg2_nom (默认: 0x0) 定义 PHASE_SEG2 窗口的宽度。 28:24 => sjw_nom (默认 : 0x0) 重新同步跳转宽度
0x0010	BTD	无	位定时（数据）寄存器。 该寄存器定义了 FDS 中允许的可选更高数据速率的位定时参数，并在数据阶段使用。较高速率的数据阶段由隐性 BRS 位表示。单位为 TQ 时钟周期。 3:0 => phseg1_d (默认 : 0x0) 定义 PROP_SEG + PHASE_SEG1 窗口的宽度。 11:8 => phseg2_d (默认 : 0x0) 定义 PHASE_SEG2 窗口的宽度。 26:24 => sjw_d (默认 : 0x0) 重新同步跳转宽度
0x0020	IS	只读	中断状态寄存器 指示各种事件的状态。 CPU 将“1”写入中断清除（IC）寄存器以分别清除每一位。 0 => RCVFVLD (默认: 0x0)

偏移/范围	寄存器名称	访问限制	描述
0x0020	IS	只读	Rx FIFO 有效。当读为 1 时，表示接收 FIFO 至少有一个有效帧接收并可供 CPU 通过 Rx FIFO 窗口寄存器 RXB 读取。“0”表示 Rx FIFO 缓冲区为空或不包含任何有效的 Rx 消息。 1 => RXBFULL (默认 : 0x0) Rx FIFO 溢出。当读为“1”时，表示在一个有效消息正在写入时，RX 缓冲区发生溢出，这个消息会被丢弃 4 => TXFOVF (默认 : 0x0) Tx FIFO 溢出指示。如果 Tx 缓冲区在发送帧的中途溢出，则丢弃部分帧并且不发送部分帧。这是错误情形，因为正常访问过程不会导致此位被置位。在开始将帧写入 Tx FIFO 之前，CPU 应检查 Tx 缓冲区/ FIFO 中的空间 5 => TXHFOVF (默认 : 0x0) Tx 高优先级 FIFO 溢出指示。如果 Tx 缓冲区在发送帧的中途溢出，则丢弃部分帧并且不发送部分帧。这是错误情形，因为正常访问过程不会导致此位被置位。在开始将帧写入 Tx HP FIFO 之前，CPU 应检查 Tx 缓冲区/ FIFO 中的空间
0x0020	IS	只读	8 => ERRSTS (默认 : 0x0) 读为“1”时，表示 ERRSTS 寄存器（0x60）中的一个或多个位已置 1。 在检测时，CPU 读取 ERRSTS 寄存器内容以获得更多细节，即，知道已经设置的错误类型。 CPU 首先清除在 ERRSTS 寄存器中置位的所有位，然后清除该位本身。 27 => TXMSGOK 读为“1”时，表示上一条 Tx 消息发送成功。消息 ID 记录在 TXMSGSTS 寄存器中 26 => TXMSGRETRY 读为“1”时，表示由于先前的尝试未成功而重新尝试发送上一条 Tx 消息。消息 ID 记录在 TXMSGSTS 寄存器中 25 => TXMSGFAIL 读为“1”时，表示包括重新尝试在内的上一次 Tx 消息传输失败，并且丢弃了消息。消息 ID 记录在 TXMSGSTS 寄存器中。 23 => TXHMSGOK 读为“1”时，表示上次成功传输 HP Tx 消息。消息 ID 记录在 TXHMSGSTS 寄存器中 22 => TXHMSGRETRY 读为“1”时，表示由于先前的尝试未成功而重新尝试发送上一条 HP Tx 消息。消息 ID 记录在 TXHMSGSTS 寄存器中 21 => TXHMSGFAIL

偏移/范围	寄存器名称	访问限制	描述
0x0020	IS	只读	读为“1”时，表示包括重试尝试在内的上一次 HP Tx 消息传输失败，并且丢弃了消息。消息 ID 记录在 TXHMSGSTS 寄存器中。 31 => BUSOFF (默认 : 0x0) 读为“1”时，表示控制器处于总线关闭状态，原因是发送错误过多，即发送错误计数（TEC）大于等于 256。 OTHERS => RSVD
0x0024	IE	无	中断使能寄存器 IS 寄存器中列出的每个中断源使能。 0 => RCVFVLDE (默认: 0x0) RCVFVLD 使能 1 => RXBFULLE (默认: 0x0) RXBFULL 使能 4 => TXFOVFE (默认: 0x0) TXFOVF 使能 5 => TXHFOVFE (默认: 0x0) TXHFOVF 使能 8 => ERRSTSE (默认: 0x0) ERRSTS 使能 21 => TXHMSGFAILE (默认: 0x0) TXHMSGFAIL 使能 22 => TXHMSGRETRYE (默认: 0x0) TXHMSGRETRY 使能 23 => TXHMSGOKE (默认: 0x0) TXHMSGOK 使能 25 => TXMSGFAILE (默认: 0x0) TXMSGFAILE 使能 26 => TXMSGRETRYE (默认: 0x0) TXMSGRETRY 使能 27 => TXMSGOKE (默认: 0x0) TXMSGOK 使能 31 => BUSOFFE (默认: 0x0) BUSOFF 使能 OTHERS => RSVD
0x0028	IC	只写	中断清除寄存器 清除 IS 寄存器中列出的每个中断源。 0 => RCVFVLD (默认 : 0x0) 清除 RCVFVLD 1 => RXBFULL 清除 RXBFULL 4 => TXFOVF (默认: 0x0)

偏移/范围	寄存器名称	访问限制	描述
0x0028	IC	只写	清除 TXFOVF 5 => TXHFOVF (默认: 0x0) 清除 TXHFOVF 8 => ERRSTS (默认: 0x0) 清除 ERRSTS 21 => TXHMSGFAIL (默认: 0x0) 清除 TXHMSGFAIL 22 => TXHMSGRETRY (默认: 0x0) 清除 TXHMSGRETRY 23 => TXHMSGOK (默认: 0x0) 清除 TXHMSGOK 25 => TXMSGFAIL (默认 : 0x0) 清除 TXMSGFAIL 26 => TXMSGRETRY (默认: 0x0) 清除 TXMSGRETRY 27 => TXMSGOK (默认: 0x0) 清除 TXMSGOK 31 => BUSOFF (默认: 0x0) 清除 BUSOFF OTHERS => RSVD
0x0030	CFG	无	配置寄存器 0 => isofd (默认 0x0) 0 :非 ISO FD 模式（Bosch） 1 : ISO FD 模式 如果 ISO FD 模式在 CRC 字段的开头使能填充计数器，如 ISO_DIS_11898-1 规范中所述 4 => disprotexceponres (默认为 0x0) 0 :隐性'res'位被视为协议异常事件 1 :隐性'res'位被视为 FORM-ERROR 仅当节点在 ISO FD 模式下运行时才适用和解释
0x0040	RXBCFG	无	Rx 缓冲区/ FIFO 配置寄存器 定义接收缓冲区的 Rx 缓冲区大小（以 32 位字为单位）。它是 TXBUF 和 TXHBUF 共享的共享缓冲区中的软分区。 15:0 => rxb_start (默认: 0x0) 缓冲区的起始偏移量 31:16 => rxb_end (默认: 0x0) 缓冲区的结束偏移量 注意：使用/编程的位数应与共享缓冲区地址宽度匹配，例如，如果共享缓冲区深度为 256 个字，表示总缓冲区空间为 1K Bytes，则仅使用低 8 位，若高位被配置，则有可能与其他两个 FIFO 重叠。

偏移/范围	寄存器名称	访问限制	描述
0x0044	TXBCFG	无	RegisterTx 缓冲区/ FIFO 配置寄存器 定义接收缓冲区的 Tx 缓冲区大小（以 32 位字为单位）。它是 RXBUF 和 TXHBUF 共享的共享缓冲区中的软分区。 15:0 => txb_start (默认: 0x0) Rx 缓冲区的起始偏移量 31:16 => txb_end (默认: 0x0) 缓冲区的结束偏移量 注意：使用/编程的位数应与共享缓冲区地址宽度匹配，例如，如果共享缓冲区深度为 256 个字，表示总缓冲区空间为 1K Bytes。，则仅使用低 8 位，若高位被配置，则有可能与其他两个 FIFO 重叠。
0x0048	TXHBCFG	无	Tx 高优先级缓冲区/ FIFO 配置寄存器 定义接收缓冲区的 Tx HP 缓冲区大小（以 32 位字为单位）。它是 RXBUF 和 TXBUF 共享的共享缓冲区中的软分区。 15:0 => txhb_start (默认: 0x0) Rx 缓冲区的起始偏移量 31:16 => txhb_end (默认: 0x0) 缓冲区的结束偏移量 注意：使用/编程的位数应与共享缓冲区地址宽度匹配，即，例如，如果共享缓冲区深度为 256 个字，表示总缓冲区空间为 1K Bytes。，则仅使用低 8 位，若高位被配置，则有可能与其他两个 FIFO 重叠。
0x0050	TXBRETRY	无	Tx 缓冲区重试计数（默认值：0x0） 定义传输错误时 tx 消息的重试次数。默认值（0x0）意味着在成功传输消息之前无限次重试。 注意：如果在有限重试模式下重试 x 次后消息传输失败，则会丢弃该消息。 重试次数可设置为： 0 => 无限次 1 => 1 次 2 => 2 次 … 15 => 15 次
0x0054	TXHBRETRY	无	Tx 高优先级缓冲区重试计数（默认值：0x0） 定义传输错误时高优先级 tx 消息的重试次数。默认值（0x0）意味着在成功传输消息之前无限次重试。 如果在有限重试模式下重试 x 次后消息传输失败，则丢弃该消息。 重试次数可设置为： 0 => 无限次

偏移/范围	寄存器名称	访问限制	描述
0x0054	TXHBRETRY	无	1 => 1 次 2 => 2 次 … 15 => 15 次
0x0058	TXMSGSTS	只读	传输消息状态寄存器 记录上次传输消息的状态 31:30 => txmsgsts 消息状态 2'b00 : OK -消息传输成功 2'b10 : RETRIED -重试-先前尝试失败，正在重试 2'b11 : FAILED -失败且消息被丢弃 28:0 => txmsgid 消息 ID。 只有 28:18 位对没有扩展标头的帧有效
0x005C	TXHMSGSTS	只读	传输高优先级消息状态寄存器 记录上次 HP 传输消息的状态 31:30 => txhmsgsts 消息状态 2'b00 : OK -消息传输成功 2'b10 : RETRIED - 之前的尝试失败并正在重试 2'b11 : FAILED -失败且消息被丢弃 28:0 => txhmsgid 消息 ID。 只有 28:18 位对没有扩展标头的帧有效
0x0060	ERRSTS	无	错误状态寄存器。 记录规范中定义的错误事件，直到 CPU 通过向相应位写入'1'清除。 0 => 填充错误 1 => 位错误 2 => 形式错误 3 => 应答错误 4 => CRC 错误
0x0064	ERRCNTR	只读	错误计数器寄存器 反映发送和接收错误计数器 TEC 和 REC 的值。TEC 和 REC 按规范中的定义递增和递减。该寄存器显示这些计数器的简介。 8:0 => REC (默认: 0x00) 24:16 => TEC (默认 : 0x00)
0x0100-	AF	无	接收接收滤波器寄存器

偏移/范围	寄存器名称	访问限制	描述
0x100+(4*N)	AF	无	每个滤波器都有相应的位掩码寄存器，CPU 可以对其进行编程以选择性地匹配标识符位或其他字段的子集。 每个 AF 都是 32 位寄存器，第一个滤波器从偏移量 0x100 开始，后续滤波器间隔 4 个字节。 接收滤波器数量，N = 1-16 AF 寄存器的每位定义如下： 31 =>使能 '1' =>滤波器有效并用于匹配传入帧 '0' =>滤波器无效并不用于匹配传入帧 28:18 =>基本标识符 (BID) 17:0 =>标识符扩展 (IE) 如果通过置位 IDE 位将滤波器配置为扩展标识符，则 IE 与传入扩展帧匹配/使用 30 => IDE 选择滤波器是用于扩展帧还是普通帧 '1' =>用作扩展帧 '0' =>用作普通帧 29 => EDL (扩展数据长度) EDL bit 用于匹配 FD 格式的帧，由隐性 EDL 位指示 '1' =>匹配 FD 帧，普通帧无法通过此滤波器 '0' =>匹配普通帧，FD 格式帧无法通过此滤波器
0x0140-0x140+(4*N)	AFM	无	接收接收滤波器掩码寄存器 每个接受滤波器的掩码向量。在每个滤波器中只比较/匹配掩码为 1 的位，掩码为 0 的位被当作通配符，不做比较/匹配。 每个 AFM 都是 32 位寄存器，第一个滤波器从偏移量 0x100 开始，后续滤波器间隔 4 个字节。 接收滤波器数量，N = 1-16。 AF 寄存器的每位定义如下。 28:18 =>基本标识符(BID)掩码 BID 字段的位掩码 17:0 =>用于 IE 字段的标识符扩展（IE）掩码位掩码。 OTHERS => RSVD
0x0200	RXB	只读	接收缓冲区/ FIFO 窗口寄存器。 提供对 Rx FIFO 缓冲区的间接访问。 每个 CPU 读取都被视为 FIFO 读取，其中加载了来自 Rx FIFO 的下一个字。当 CPU 获知 Rx FIFO 有效（包含至少一个有效帧）时，CPU 开始读取，直到弹出或读取帧的所有字。前 2 个字包含由控制器插入的特殊标头，包含控制信息，如长度（DLC 的解码值），和其他控制标志，如 IDE，EDL，BRS 等。

偏移/范围	寄存器名称	访问限制	描述
0x0204	TXB	只写	发送缓冲区/ FIFO 窗口寄存器。 提供对 Tx FIFO 的间接访问。 每个至该寄存器的 CPU 写入将一个字写入 Tx FIFO，内部 FIFO 写指针递增。一旦获知空间可用性，CPU 就会执行重复写入，直到 Tx 帧的所有字节都被写入（推入）到 Tx FIFO 中。前两个写入包含有助于控制器格式化和传输有效载荷的头字。标头包含长度（由控制器编码到 DLC 字段中）和其他控制信息，例如 BID，IE，IDE，RTR，EDL，和 BRS。
0x0208	TXHB	只写	传输高优先级缓冲区/ FIFO 窗口寄存器。 提供对 THPB FIFO 的间接访问。 每个至该寄存器的 CPU 写入将一个字推入 THPB FIFO，内部 FIFO 写指针递增。一旦了解了空间可用性，CPU 就会执行重复写入，直到 Tx 帧的所有字节都被写入（推入）到 THPB FIFO 中。前两个写入包含有助于控制器格式化和传输有效载荷的头字。头包含长度（由控制器编码到 DLC 字段中）和其他控制信息，例如 BID，IE，IDE，RTR，EDL 和 BRS。
0x020C	TXBSTS	只读	发送缓冲区/FIFO 状态。 指示在将新发送帧写入/推入 FIFO 之前由 CPU 读取的 Tx FIFO 的当前状态。 15:0 => txbspace 可用 32 位字空间。CPU 将在写入每个新发送帧之前检查空间。空间应大于等于有效载荷中的字数+ 2 个字（考虑由 CPU 填写/写入的发送头） 31 => txbwerr 如果 CPU 写入（推入）数据引起了 TX FIFO 的溢出，那么该位会被置位。建议在每一次发送的最后一个帧被写入 TX FIFO 后，CPU 读取该位，以确保该位是零。注意：如果发生溢出，发送数据包将被丢弃，并且不会传输到 CAN 总线上。
0x0210	TXHBSTS	只读	发送高优先级缓冲区/FIFO 状态。 指示在将新发送帧写入/推入 FIFO 之前由 CPU 读取的 Tx HP FIFO 的当前状态。 15:0 => txhbspace 空间可用 32 位字。CPU 将在写入每个新发送帧之前检查空间。空间应大于等于有效载荷中的字数+ 2 个字（考虑由 CPU 填写/写入的发送头） 31 => txhbwerr 如果 CPU 写入（推入）数据引起了 TXH fifo 的溢出，那么该位会被置位。。建议：在每一次发送的的最后一个帧被写入 TX fifo 后，CPU 读取该位，以确保该位是零。注意：如果发

偏移/范围	寄存器名称	访问限制	描述
0x0210	TXHBSTS	只读	生溢出，发送数据包将被丢弃，并且不会传输到 CAN 总线上。
0x0214	RXBSTS	只读	接收缓冲区/ FIFO 状态。 指示 Rx FIFO 的当前状态。CPU 可以进行读取以确认 Rx FIFO 是否有任何等待读取的帧。这是等待 RCVFVLD 中断的另一种方法。 注意：由于控制器在帧边界上更新了状态，任何大于 0 的值表示至少有一个完整帧。 以 32 位字为单位。 15:0 => rxbdepth Rx FIFO 的占用等级

7 中断

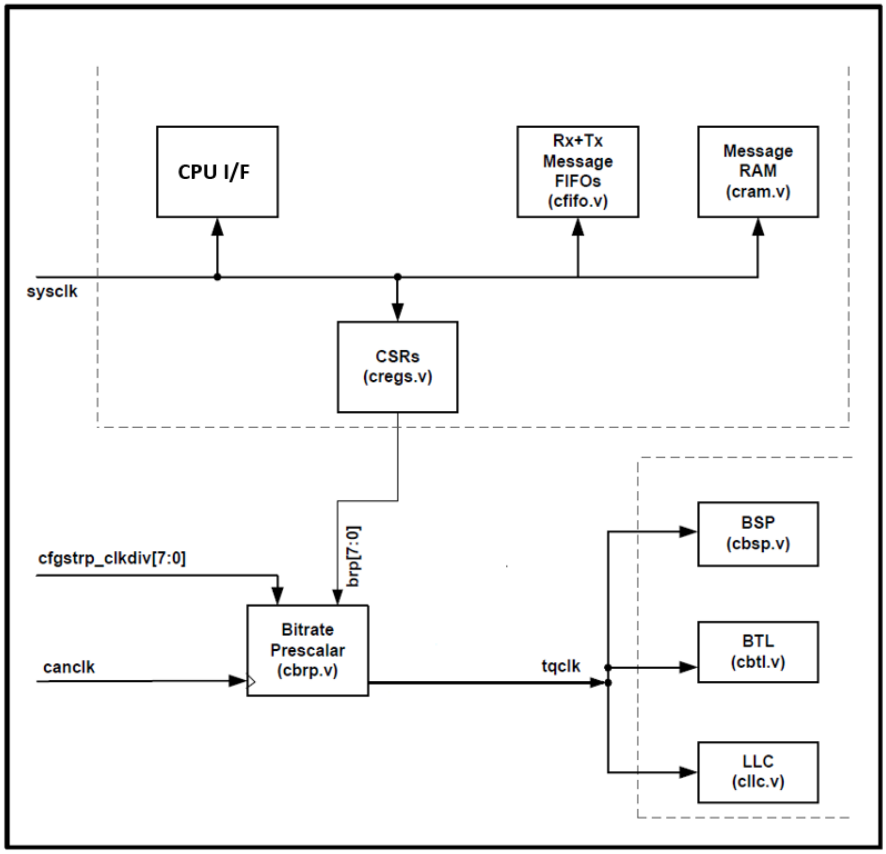
CAN 控制器 IP 实现高电平有效的电平中断。它是一种标准机制，其中所有使能的单个中断源或在一起，产生最终的中断输出。中断源在中断状态（IS）寄存器中定义，中断使能具有相应的掩码。

8 时钟

CAN 控制器 IP 中使用的时钟方案如图 8-1 所示。IP 有两个时钟输入源：canclk 和 sysclk。canclk 作为 CAN 总线比特流传输的参考时钟。canclk 被波特率预分频器（BRP）分频以产生时间 TQ 时钟（tqclk），它是串行位定时的基础。BRP 支持整数除法，可以达到 255 分频。canclk 的频率必须至少是所需 tqclk 的四倍。

sysclk 是 IP 的第二个输入源时钟。它用作 CPU 接口时钟和 IP 系统时钟，因此如果使用 ARM CPU，则与 AHB 接口时钟相关联，通常其运行速度远高于 canclk。通常 sysclk 等于或者比 canclk 更快。sysclk 和 canclk 为异步时钟，且没有相位关系的要求。

图 8-1 时钟结构



9 Reset 复位

CAN 控制器 IP 使用异步复位机制，其中所有寄存器都异步复位。进入触发器的复位输入被预先同步到相应的时钟域，因此复位的取消与时钟的边沿同步。IP 有两个复位源：低电平有效上电复位和 CPU 启动的软件复位。在基于 AHB 的系统中，上电复位与 hreset_n 连接。两个复位都用作硬复位，使得 IP 完全复位，但以下情况除外：软件复位不会复位 CPU 接口块和软件复位寄存器。

10 初始化序列示例

以下是 CPU 初始化 IP 顺序的实例。

10.1 上电

1. 将管脚 cfgstrp_clkdiv 的值预设为所需值
2. 启动 sysclk 和 canclk
3. 使上电复位 ponrst_n 为低电平
4. 等待至少 8 个 canclk 时钟周期
5. 释放 ponrst_n
6. 等待至少 32 tqclk 时钟周期并继续执行 11.2 节中概述的初始化序列。请注意，tqclk 时钟周期取决于 canclk 频率和 cfgstrp_clkdiv[7:0]输入引脚预设的分频值。例如，对于分频器值 0x4，请等待 128 个 canclk 时钟周期。

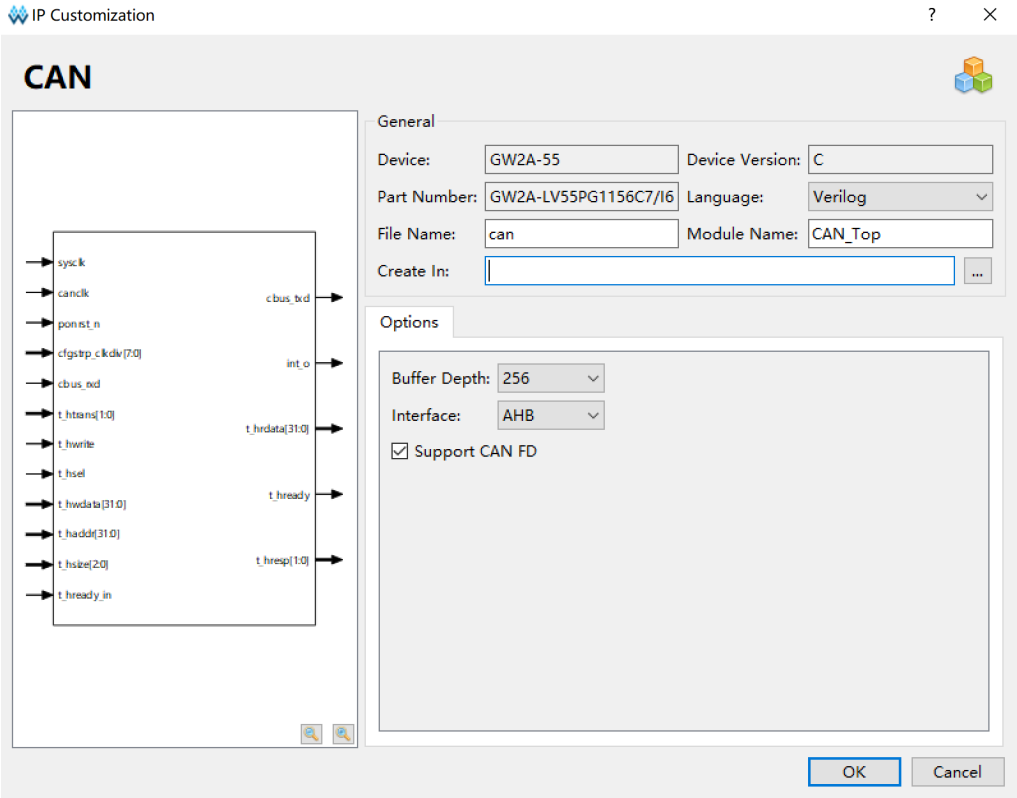
10.2 Initialization 初始化

1. 设置 BRP 寄存器的值为预期值
2. 设置帧相关寄存器 BTN、BTD
3. 在寄存器 RXBCFG，TXBCFG 和 TXHBCFG 上配置缓冲区边界，用来初始化发送和接收缓冲区分区
4. 对 IE 寄存器中的中断源使能进行设置。
5. 在配置寄存器 AF 中设置接收滤波器，在 AFM 中设置相应的掩码。
6. 向 CMD.enable 位写入'1'，从而将 IP 置于操作模式。IP 变为可操作后，IP 准备发送和接收消息。

11 接口配置

用户可以使用 IDE 中的 IP 内核生成器工具调用和配置 Gowin CAN Controller IP。CAN 配置界面如图 11-1 所示。

图 11-1 CAN 配置界面



- 可通过修改“Buffer Depth”（缓冲区深度）一栏的值来配置缓冲区的深度。
- 可通过修改“Interface”一栏的选项来配置 CPU 接口类型。
- 可通过勾选“Support CAN FD”选项，配置 IP 是否支持 CAN FD。

12 参考设计

详细信息请参见高云半导体官网给出的 CAN 相关*参考设计*。

GOWIN高云

智慧逻辑 定制未来