CAN-CTRL

CAN Controller Core

For CAN 2.0B and CAN FD

Design Specification

February 2019

IP Product Version
7x06N00S00

Document Signature
CAN-CTRL–DES–7x06N00S00–162

CAST, Inc.

This core Engineered by:
Fraunhofer IPMS

Document Version

This document with its associated Release Notes applies to the version(s) of the core specified on the cover. See the Release Notes for any updates and additional information not included here.

Table 1-1 Document Version History

Version	Date	Person	Changes from Previous Version
100	2011/08/24	R.H.	First release
101	2011/09/16	R.H.	Acceptance filtering clarification New synthesis parameter - NO_STBUFS
102	2011/10/20	R.H.	change of behavior of register s ACF_x New synthesis parameter - NR_OF_ACF
107	2012/09/17	R.H.	Documentation of registers RTR and DLC updated
108	2013/01/28	R.H.	AMBA APB interface added
109	2013/04/08	R.H.	documentation of TECNT, RECNT improved
110	2013/06/02	R.H.	New container for multiple CAN-CTRL instances added
111	2013/10/23	R.H.	CAN FD support added
116	2014/02/21	R.H.	Specification change for host read access
117	2014/03/04	R.H.	Improvements for host read access
118	2014/03/10	R.H.	Loop Back Mode and Standby Mode added
119	2014/08/05	R.H.	AMBA APB interface redesigned
121	2014/08/21	R.H.	AMBA AHB interface added
122	2014/09/02	R.H.	AMBA APB: byte access protection New register map (better address alignment)
123	2014/12/17	R.H.	new bits AIDEE and AIDE, new register TDC
128	2015/07/28	R.H.	Transformation to 32 bit peripheral New memory map New RBUF, TBUF and ACF organization documentation of host interfaces updated CAN FD ISO compatibility LOM and LBM updated
129	2015/08/13	R.H.	Prescalers can now be set to 1.
130	2015/09/11	R.H.	New register BITTIME_3 Bit timing documentation improved
131	2015/10/05	R.H.	New method for interrupt flag reset Unlimited number of STB slots New behaviour of TSSTAT
133	2016/01/08	R.H.	Priority decision mode for the STB Time-Triggered CAN
134	2016/0126	R.H.	Bit position of TBE and TBF in the documentation corrected.
135	2016/04/12	R.H.	New organization and extension of bit timing parameters New feature: error counter reset New synthesis parameter: SHORT_SEG2
136	2016/04/26	R.H.	Bit RRS in CAN FD frames ignored and overridden Verification documentation added
137	2016/04/27	R.H.	Wishbone 32 bit interface added
138	2016/05/23	R.H.	TTCAN can be disabled by generic parameter CiA 603 time-stamping added
139	2016/06/08	R.H.	LOM behaviour fully changed LBME does not generate a self-ACK any more
140	2016/06/17	R.H.	AMBA APB testbench added to the release package Combination of LOM+LBME created. Position of CYCLE_TIME in RBUF changed. Update behaviour of register KOER changed. Control bit RBALL added. Status bits KOER and TX added to RBUF.
141	2016/07/14	R.H.	Bit TTSEN added to the TBUF
142	2016/07/27	R.H.	Definition of CiA 603 external timer updated

			Synthesis parameter SHORT_SEG2 removed Added explanation how to turn on/off CAN FD
143	2016/08/01	R.H.	Comments for Synthesis with Xilinx ISE added
144	2016/09/20	R.H.	Improved description of REF_ID
145	2016/10/13	R.H.	New chapter “Error Handling” Chapter “Synthesis Hints” extended Top-level test outputs renamed
146	2017/01/02	R.H.	Recommended CAN bit timing configurations changed Memory synthesis parameter description corrected Supplied file list updated
147	2017/02/02	R.H.	Description of hull component <src/fpga_boundary.v(hd)> added Users Guide (USG) split into Design Specification (DES) and Integration Manual (INM)
148	2017/02/21	R.H.	CiA 603 time-stamping: hint for bit TTSEN added
149	2017/03/20	R.H.	Recommendations for CiA 603 time-stamp bit width added.
150	2017/03/21	R.H.	TPE, TSONE and TSALL: no block of set during automatic reset
151	2017/05/02	R.H.	Correction of minor typing errors
152	2017/08/29	R.H.	Exact reference to the used AMBA versions added Removal of outdated configuration parameters Behavior of RECNT changed
153	2017/09/21	R.H.	Behavior of bus off state and bit BUSOFF changed with respect to bit RESET
154	2017/10/09	R.H.	Clarification about bus-off recovery sequence handling
155	2017/11/20	R.H.	Synthesis parameter STB_PRIO added
156	2017/12/07	R.H.	Synthesis parameter RAM_TYPE added which adds the new feature to use pseudo dual port memories.
157	2017/12/20	R.H.	Linux driver added to feature list
158	2018/04/18	R.H.	Note about the pins to the transceiver added. APB read wait state removed. Parameter RAM_MEMTYPE extended. Recommended bit timing parameters corrected Calculation of memory sizes added LOM not affected by RESET any more
159	2018/06/04	R.H.	TIMEPOS can only be changed if TIMEEN=0
160	2018/07/02	R.H.	Interrupt flags routed to top-level output <host_if> Read access to ACF_x while RESET=0 is now possible, except if pseudo dual-port memory is selected New top-level output: <host_ready> (support for wait states) Upgraded definition of read / write host accesses Support for single port memory added Upgraded definition of all interface wrappers
161	2018/02/12	R.H.	Links between the descriptions of the PDP memory and the 8 bit host interface wrapper added. Guideline for parameter RAM_TYPE added. Bit EDL renamed to FDF. New generic parameters ADD_I_SYNC and ADD_O_SYNC. Asynchronous reset inputs changed. If TDC is enabled, F_PRESC is limited to 0 or 1.
162	2018/02/18	R.H.	New generic parameter ADD_R_SYNC.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 The CAN-CTRL Core

The CAN-CTRL core is a serial communications controller that performs serial communication according to the CAN protocol. This CAN bus interface uses the basic CAN principle and meets all constraints of the CAN-specification 2.0B active. Furthermore this CAN core can be configured to meet the specification of CAN with flexible data rate CAN FD.

The CAN protocol uses a multi-master bus configuration for the transfer of frames (communication objects) between nodes of the network and manages the error handling without any burden on the CPU. The CAN-CTRL bus controller enables the user to set up economic and reliable links between various components. The CAN-CTRL core appears to a microcontroller as a memory-mapped I/O device. A CPU accesses the CAN-CTRL core to control transmission or reception of frames through a two wire CAN bus system. The connection to a CAN bus is illustrated in Figure 1.

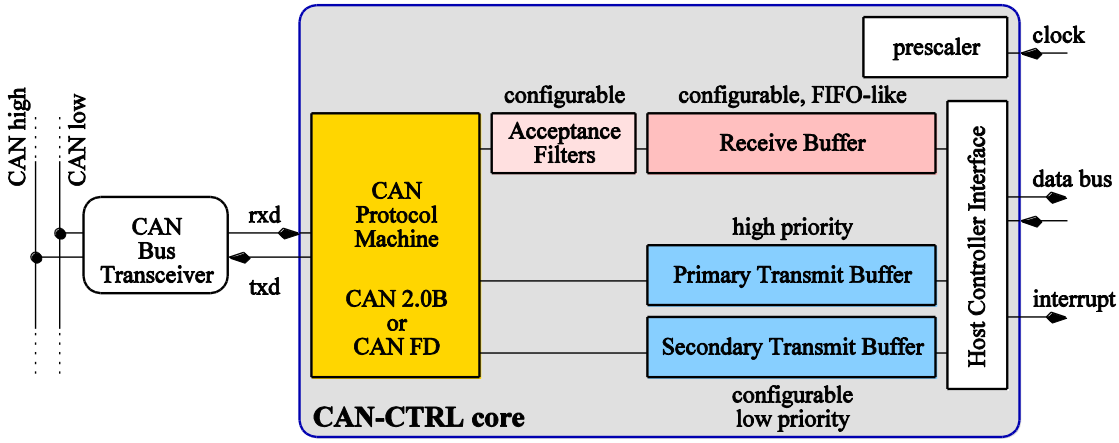

Figure 1-1 Connection to CAN Bus and Main Features of the CAN-CTRL Core

1.2 The CAN Protocol

CAN communication is organized in frames. Two types of frames exist: standard and extended frames. For CAN 2.0 the maximum data payload is up to 8 bytes while for CAN FD up to 64 bytes can be transmitted using one frame.

All CAN nodes are equal in terms of bus access. There is no super-node because the CAN is a multi-master bus.

Data addressing is done using message identifiers. In a CAN network only one node shall transmit messages with a certain identifier. All nodes receive all messages and the node host controller has to decide if it was addressed by the appropriate message identifier. To reduce the load of a host controller a CAN core may use acceptance filters. These filters compare all received message identifiers to user-selectable bit patterns. Only if a message passes an acceptance filter, it will be stored in the receive buffer and signaled to the host controller.

The identifiers of CAN frames are also used for bus arbitration. The CAN protocol machine stops transmission of a message with a low-priority identifier when a message with a higher priority identifier is transmitted by another CAN node. The CAN protocol machine automatically attempts to re-transmit the stopped message at the next possible transmit position.

CAN 2.0B defines data bit rates up to 1Mbit/s. For CAN FD there is no fixed limitation. For CAN FD the standard defines a bit rate switching. If enabled the transmission of the payload of frames can be done at higher speed while the frame header is transmitted at lower speed.

1.3 Classic CAN 2.0B and How To Enable CAN FD

CAN FD is a downward-compatible extension of the classic CAN 2.0B protocol. Every CAN FD node is able to receive and transmit classic CAN frames. CAN-CTRL can be configured to support only classic CAN or both classic CAN and CAN FD.

The synthesis parameter CAN_FD selects if CAN FD is included in the hardware or excluded (see chap. 3.2). Including CAN FD into the system will require approx. 6% more cell area and bigger receive and transmit buffers slots. (Classic CAN carries up to 8 bytes payload and CAN FD up to 64 bytes.) For modern systems it is recommended to include the CAN FD features.

If the synthesis parameter CAN_FD is set to include the CAN FD features then the input pin <can_fd_enable> can force CAN-CTRL to only classic CAN operation or can unlock CAN FD. This feature can be used to limit the system to classic CAN und unlock CAN FD later. (See chap. 3.1 for details.)

If the synthesis parameter CAN_FD is set to include the CAN FD features and the input pin <can_fd_enable> is set to enable CAN FD, then the host controller firmware can select at run-time for every frame if the frame will be transmitted as classic CAN or CAN FD frame. This is done using the configuration bits in the transmit buffer. For received frames the status bits in the receive buffer signal if a received frame was a classic CAN or CAN FD frame.

Additional info: If the synthesis parameter CAN_FD is set to exclude the CAN FD features, then CAN-CTRL is a pure classic CAN node, but it is upward-compatible or “CAN FD tolerant”. This means, that CAN-CTRL will ignore CAN FD frames and will not destroy them. Therefore it can be used in a mixed network where some nodes use classic CAN and some others use CAN FD.

1.4 Time-Stamping

1.4.1 Time-Triggered CAN

Optionally CAN-CTRL can be used for Time-Triggered CAN communication (TTCAN) according to ISO 11898-4. CAN-CTRL offers partial hardware support and requires host software interactions in real time in this mode.

The basic concept of TTCAN is to have a timer for time-stamping received messages and for triggering messages for transmission. One node in the CAN network is the time master. A time master transmits a reference message. With the reference message the cycle time starts. The time between two reference messages is a basic cycle. Inside the basic cycle messages can be transmitted in time windows. The TTCAN system administrator defines the start and duration of each time window during an offline setup.

There are three time window types:

- Exclusive time window (only one node is allowed to transmit one frame with a defined ID)
- Free time window (unused time window for further extension of the network)
- Arbitrating time window (several nodes may transmit a frame and arbitration takes place)

If a frame is received, it gets the actual cycle time as time-stamp. For transmissions CAN-CTRL offers a hardware trigger that starts transmission of a predefined message at a predefined cycle time.

CAN-CTRL automatically detects a reference message upon reception and starts the cycle time. The hardware timer is a 16 bit timer running at the CAN bit time as defined for ISO 11898-4 level 1. CAN-CTRL can be used as a time master.

Beside a hardware trigger for message transmissions CAN-CTRL offers a watch trigger to detect a missing reference message.

Partial hardware support means that the host controller needs to prepare the actions of the node for each time window. E.g. the host needs to define the message for the next transmission and define the trigger time for it. In other words: the host needs to take care of the transmission columns and the system matrix.

1.4.2 CiA 603 Time-Stamping

CAN in Automation (CiA) defines in specification 603 a method for time-stamping with at least 16 bits, which is optionally supported by CAN-CTRL. It can be used in addition to TTCAN.

The basic concept of CiA 603 is to have a free-running timer which counts clock cycles and not CAN bit times. The precision shall be least 10µs (16 bit) or 1µs (32 bit or more). Time-stamps are acquired at the SOF or EOF of a CAN / CAN FD frame.

CiA 603 is supposed to support time-stamping and time-synchronization of AUTOSAR. For AUTOSAR one node in the CAN network is the time master. A time master transmits a synchronization message (SYNC message). The time-stamp of the SYNC message is acquired by the time master and all time slaves. The difference in time between the event of commanding a SYNC message until the time when the SYNC message actually gets transmitted will be transmitted in a follow-up (FUP) message by the time master.

CiA defines rules to read out the timer as well as to modify the timer. CAN-CTRL does not include the timer, but uses an external timer. CAN-CTRL only includes the mechanism of time-stamping, the register to store one transmission time stamp (TTS) and the memory to store reception time stamps (RTS) for all received messages.

1.5 Using the CAN-CTRL Core

The Integration Manual (INM) provides a summary of the main steps on how to use the CAN-CTRL core inside the dedicated chapter “How to use CAN-CTRL Core”. It can be used as an overview during the first steps of learning how the core works as well as a summary when the main features are understood.

2. Features

2.1 Feature List

- Supports CAN specification
  - CAN 2.0B (up to 8 bytes payload, verified by Bosch reference model)
  - Optional support for CAN FD (up to 64 bytes payload, ISO 11898-1:2015 or non-ISO Bosch)
- Free programmable data rate:
  - CAN 2.0B defines data rates up to 1Mbit/s
  - CAN FD is limited by the transceiver and the clock frequency of the CAN-CTRL core.
- Programmable baud rate prescaler (1 to 1/256)
- Separate clock domains for host interface and CAN protocol machine
- Configurable receive buffer (RB) size
  - Generic parameter selects number of buffer slots.
  - FIFO-like behavior
  - Received messages which are “not accepted” or “incorrect” don’t overwrite already stored messages.
- Two Transmit Buffers
  - one Primary Transmit Buffer (PTB)
  - optional configurable Secondary Transmit Buffer (STB)
    - The STB is optional. A generic parameter selects the number of buffer slots.
    - Operation in FIFO or priority decision mode
- Independent and programmable internal 29 bit acceptance filters
  - Number of acceptance filters selectable by generic parameter in the range of 1 to 16
- Extended features
  - Single Shot Transmission Mode (for PTB and / or for STB)
  - Listen Only Mode
  - Loop Back Mode (internal and external)
  - Transceiver Standby Mode
- Extended status and error report
  - Capturing of last occurred kind of error and of arbitration lost position
  - Programmable Error Warning Limit
- Different host controller interfaces
  - 32 bit synchronous host controller interface; wrapper for 8 bit hosts
  - 32 bit AMBA APB Protocol Specification v2.0
  - 32 bit AMBA 3 AHB-Lite Protocol v1.0
  - 32 bit Wishbone
  - Optional application specific interface to the host-controller on request.
- Configurable interrupt sources
- One dual-port memory block or two pseudo dual port memory blocks for the frame buffers

- Time-stamping:
  - ISO 11898-4 Time-Triggered CAN with partial hardware support
  - CiA 603 time-stamping
- Fully synchronous and synthesizable HDL design (Verilog 2001, VHDL 93)
- Compatibility to AUTOSAR
- Optimized for SAE J1939
- Linux driver included

2.2 Upward Compatibility

The CAN specification includes reserved bits (Figure 2-3) for protocol extension. This has been used to build the CAN FD specification on top of the CAN 2.0B specification. Reserved bits are transmitted low (dominant) if not used. Unfortunately the CAN 2.0B specification defines the behavior for the case that a reserved bit is high (recessive) to accept this and proceed with the frame. Therefore if a CAN FD frame (which has a different and unexpected form compared to a CAN 2.0B frame) is received by a CAN 2.0B node that uses this behavior then this will result in an error frame by the CAN 2.0B node which destroys the frame. This behavior is called “CAN FD intolerant”.

To be upward compatible to new protocol specifications a so-called protocol exception event shall take place if a node detects a reserved bit high. This holds for CAN 2.0B as well as for CAN FD nodes. A protocol exception event results in no action for a receiver. The receiver just ignores this frame, does not generate an ACK, waits for bus idle and may transmit or receive the next frame. For a CAN 2.0B node this is called “CAN FD tolerant” and enables coexistence of CAN 2.0B and FD frames within one network.

Older CAN 2.0B conformance tests check for the “CAN FD intolerant” behavior but it is recommended to use the new “CAN FD tolerant” behavior or in general the protocol exception event to be upward compatible regardless if the node is a CAN 2.0B or CAN FD node.

CAN-CTRL provides a generic parameter to select the behavior (chap. 3.2, parameter UPWARD_COMPATIBILITY).

2.3 Message Buffers

2.3.1 Message Buffers Concept

The concept of the message buffers is illustrated in Figure 2-1. This schematic focuses on the buffers and hides other details of the CAN-CTRL core. All buffer slots are big enough to store frames with the maximum length.

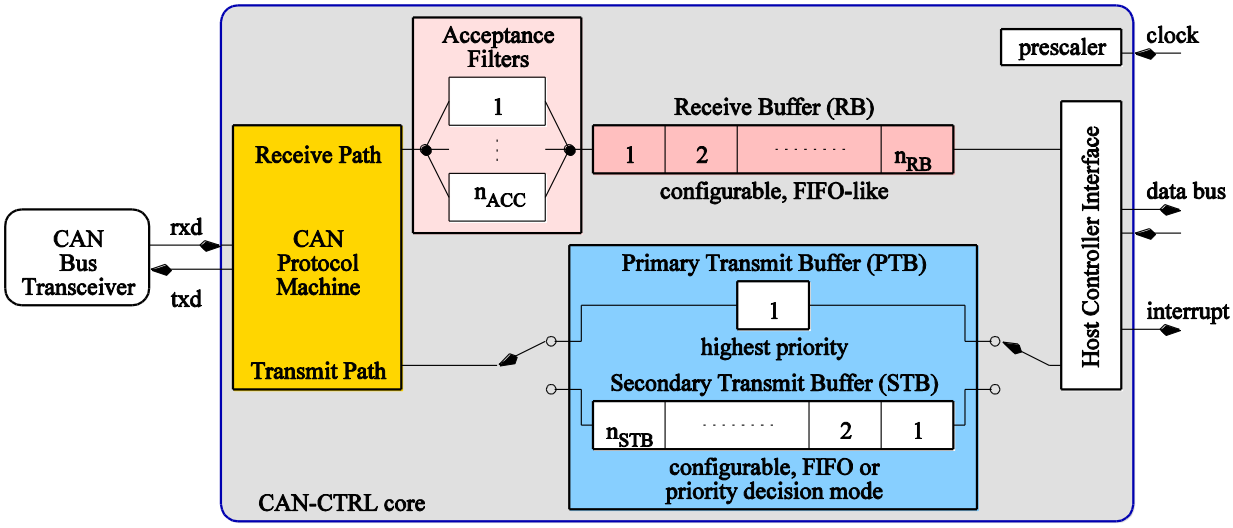

Figure 2-1 Message Buffers Concept

2.3.2 Receive Buffer

To reduce the load of received frames for the host controller, the core uses acceptance filters. The CAN-CTRL core checks the message identifier during acceptance filtering. If the received frame matches the filter criteria of one of the acceptance filters then it will be stored in the Receive Buffer (RB), which has FIFO-like behavior.

Depending on the number of available message slots, the host controller does not need to read incoming messages immediately. The CAN-CTRL core is able to generate interrupts upon every received message, when the RB is full or filled to a user-selectable “almost full” limit. Because of the FIFO-like behavior, the host controller always reads the oldest message from the RB.

2.3.3 Transmit Buffer

For frame transmission purposes, two Transmit Buffers (TB) are offered. The Primary TB (PTB) has a higher priority, but is able to buffer only one frame. The Secondary TB (STB) has a lower priority. It can act in FIFO or in priority mode. The priority decision between PTB and STB is fixed and fully independent from the CAN bus arbitration. Bus arbitration is a priority decision based on the frame identifiers.

The STB can be commanded to transmit one or all stored frames. In FIFO mode with every transmission the oldest frame inside this buffer is transmitted first. In priority mode the frame with the highest priority inside this buffer (based on the frame identifier) is transmitted first.

A frame located in the PTB has always a higher priority for the CAN protocol machine than the frames in the STB regardless of the frame identifiers. A PTB transmission stops and delays an STB transmission. The STB transmission is automatically restarted after the PTB frame has been successfully transmitted.

A PTB transmission starts at the next transmit position that is possible by the CAN protocol (after the next interframe space). Because of this, an STB transmission that has won the arbitration and is actually transmitted, will be completed before.

Interrupting STB transmissions using a PTB transmission may happen in the following cases:

1. The STB is commanded to output all stored frames and the host controller decides to command a PTB transmission before all STB transmissions are completed.
2. The STB is commanded to output a single frame and the host controller decides to command a PTB transmission before the STB transmission is completed.

If the host controller waits until each commanded transmission is completed, then it can easily decide which buffer shall transmit the next frame. As a drawback a message with a low-priority identifier may block more important messages. Then the host could abort the message (chap. 2.3.5).

2.3.4 Transmit Buffer Application Example in FIFO Mode

In the example a CAN node is used for sensor measurements. The CAN system engineer has decided to handle these sensor measurements with low priority. Therefore frame identifiers with low priority are used for CAN frames carrying the sensor data.

The host controller automatically acquires measurement results and places them as CAN frames in the STB. Because of high traffic at the CAN bus, it may happen that the sensor data frames cannot be transmitted immediately and several frames remain in the STB for some time. Later, if there is less traffic at the CAN bus, they will be transmitted.

In the situation where several frames remain in the STB, an event may happen that forces the host controller to output an important high-priority frame. In such a situation, the host can use the PTB. A frame in the PTB will be always transmitted before all frames in the STB.

The advantage of having two transmit buffers is the option to keep all messages. No message has to be aborted (discarded) in case of a high-priority event.

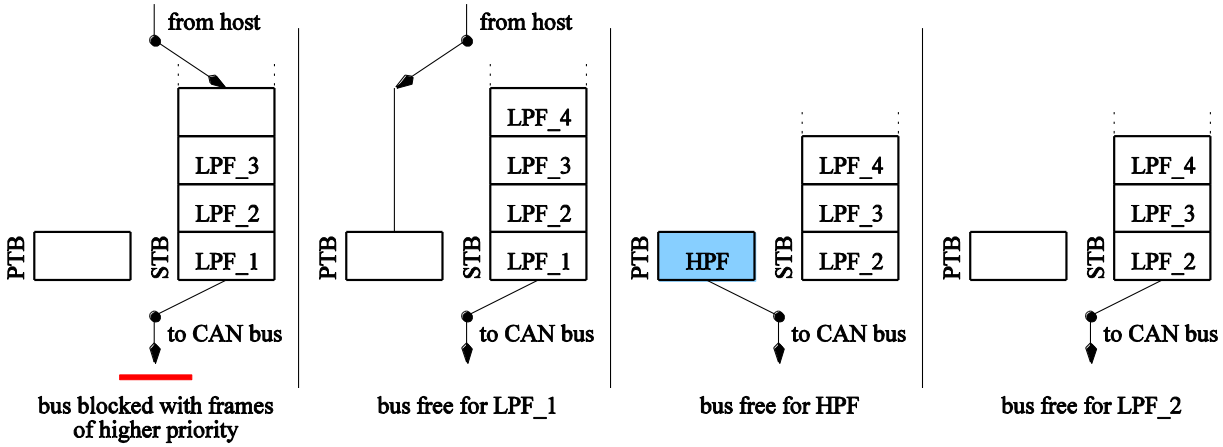

Figure 2-2 Transmit Buffer Application Example (TSALL=1)

The given application example is explained step by step in Figure 2-2. In this figure, LPF_x is the abbreviation of “low priority frame” and HPF means “high priority frame”. As can be seen, LPF_1 is blocked by other higher priority frames from other CAN nodes in the first part of the figure. The host controller puts a fourth frame into the STB (LPF_4) and then decides to output the HPF. Meanwhile, the CAN bus was free to transmit LPF_1 which is followed by HPF, LPF_2 and so on.

The priority decision between the PTB and STB is done by the CAN protocol machine, but the CAN protocol itself uses another independent priority decision mechanism which is called bus arbitration. For bus arbitration the frame identifier defines the priority level.

In the example given above, it would be possible to place a frame with a very low-priority identifier in the PTB while higher priority frames remain in the STB. For the decision between PTB and STB always the PTB wins regardless of the frame identifier. It is the task of the host controller to place frames with meaningful identifier priorities in the appropriate buffers.

2.3.5 Transmit Buffer Application Example in Priority Mode

In priority mode for the STB CAN-CTRL automatically changes the order of the frames inside the STB. As a result the frame with the highest priority is transmitted first. The priority decision is the same as for bus arbitration and therefore a frame identifier with a lower value has a higher priority. Regardless of that the PTB has always the highest priority.

Reordering the frames inside the STB is done automatically as soon as new frame is prepared for transmission and written to the STB. As a result the frame with the highest priority is selected for transmission. For the host controller it is like fire-and-forget: it just needs to write a new frame into the STB while priority mode is active.

In general a frame with a higher priority will never interrupt an active transmission. To give an example in Figure 2-2 in the leftmost part the frame LPF_1 is currently under transmission while LPF_4 is written to the STB. Even if LPF_4 has higher priority than LPF_1, still LPF_1 will be transmitted until it loses arbitration on the bus. Then after the node has lost arbitration, LPF_4 will be selected for transmission because of its priority.

Priority reordering is automatically done in background, but takes some time. During heavy load when a lot of new frames with different priorities are written to the STB and the node loses arbitration or there are errors on the bus, it may be that not the frame with the highest priority is selected. This may happen if this frame has been written too shortly before the deadline for the decision which frame will be next. But this is a very rare condition and in general the host should not care about this.

If two frames with the same priority are written to the STB, then the oldest frame will be transmitted first regardless of any reordering with other lower- or higher-prioritized frames.

2.3.6 Aborting a Transmission

If the situation arises, where a message in a transmit buffer cannot be sent due to its low priority, this would block the buffer for a long time. In order to avoid this, the host controller can withdraw the

transmission request and abort the message. Aborting is possible for PTB and STB as well as for single or all frame transmission.

2.3.7 The Transmit Buffer in TTCAN Mode

In TTCAN mode each slot in the STB can be addressed by the host. Each slot can be defined as filled or empty. Therefore each slot can be dedicated to one specific message.

Transmission in TTCAN mode is started with triggers. Each trigger includes a pointer to a message slot.

The PTB has no special priority in TTCAN mode and therefore is handled just like an STB slot.

In is convenient for a host application to dedicate one message to one buffer slot. Then a task in the host can update the message at any time and then if the trigger for this message is active, it can be transmitted. But having enough slots for all messages requires a lot of hardware. Because of the partial hardware support of ISO 11898-4 it is possible to use a buffer slot for more than one message in a basic cycle. Therefore ISO 11898-4 is possible even if the STB is disabled and the core includes only the single PTB message slot.

2.4 CAN 2.0 and CAN FD Frames

CAN FD is a protocol extension of CAN 2.0. The main differences are:

- Data payload: Up to 8 bytes for CAN 2.0 and up to 64 bytes for CAN FD
- One configurable bit rate for CAN 2.0, but 2 for CAN FD: slow for arbitration and fast for data phase

All types of frames for CAN 2.0 and CAN FD are shown in Figure 2-3. The abbreviations are explained in the CAN specification and in short in Table 2-1. For Classic CAN frames (CAN 2.0) some bit names are renamed with the CAN FD ISO specification, but here the older names are still used for easier backward-reference.

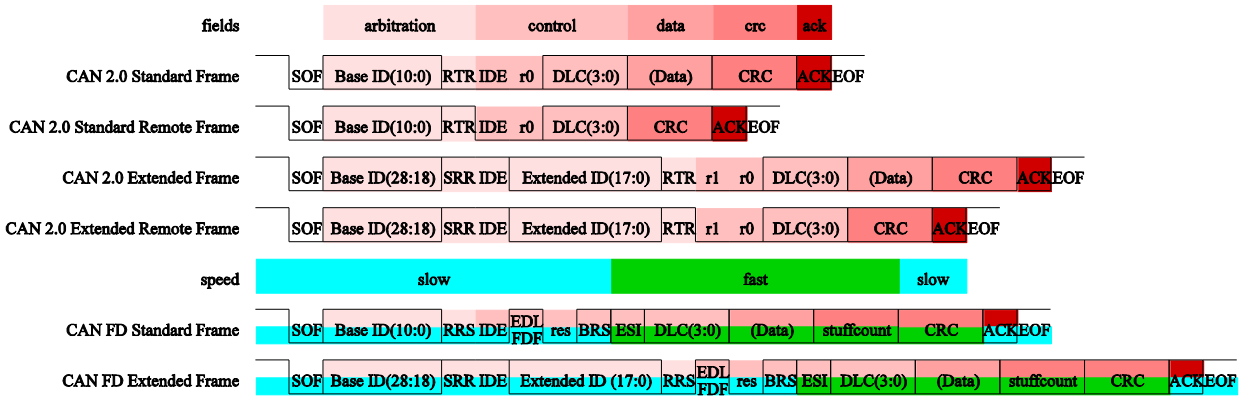

Figure 2-3 CAN 2.0 and CAN FD Frame Types

Table 2-1 CAN Bit Abbreviations

Abbreviation	Description	Comment
ID	IDentifier	
RTR	Remote Transmission Request	Remote or Data frame
SRR	Substitute Remote Request	
RRS	Remote Request Substitution	
IDE	IDentifier Extension	Standard or Extended frame
DLC	Data Length Code	Number of payload bytes
EDL	Extended Data Length	CAN 2.0 or CAN FD frame
FDF	FD Format indicator (=EDL)	CAN 2.0 or CAN FD frame
BRS	Bit Rate Switch	
ESI	Error State Indicator	
r1, r0, res	Reserved bits	

The CAN FD specification by Bosch (non-ISO) uses the name EDL while the CAN FD ISO specification uses the name FDF for the same bit. Both names are synonyms. This document uses the name FDF.

For CAN FD ISO frames the stuff count is transmitted as a part of the CRC field. For CAN FD non-ISO frames the stuff count is not part of the frame. Furthermore the CRC checkers have a different initialization for non-ISO and ISO frames. Therefore ISO and non-ISO frames are incompatible.

The CAN protocol machine in the CAN-CTRL core automatically transmits and receives the frames and embeds the appropriate control and status bits. The host controller is required to select the desired frame type (IDE, RTR, FDF), chose the identifier and set the data payload.

2.5 AUTOSAR and SAE J1939

Both AUTOSAR and SAE J1939 are software stacks which can be used with CAN-CTRL.

For AUTOSAR the number of transmission buffers should be set to at least 3 slots (generic parameter STB_SLOTS, Table 3-2) and transmissions can be aborted (bits TPA and TSA in register TCMD). Time-stamping and time-synchronization is supported using hardware time-stamping according to CiA 603.

SAE J1939 uses extended IDs (29 bits). For easy handling the acceptance filters can be configured to accept only extended IDs (bits AIDE and AIDEE in register ACF_3).

3. Interfaces

3.1 Hardware Interface

This chapter describes the core interface of CAN-CTRL. Additionally several interface wrappers are provided that connect this core interface to different host interfaces. Details about these wrappers can be found in chap. 7.

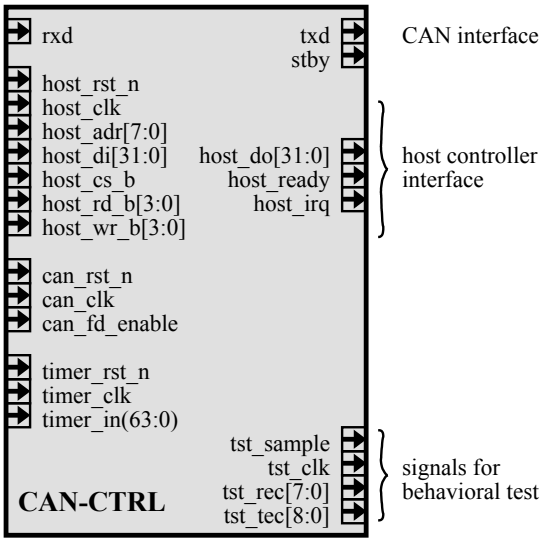

Figure 3-1 CAN-CTRL Core Pinout

Table 3-1 Pin Description

Name	Type	Polarity	Description
host_rst_n	I	Low	low-active asynchronous reset for <host_clk> domain
host_clk	I	-	clock for host interface
host_adr(7:0)	I	-	address from host
host_di(31:0)	I	-	data from host
host_do(31:0)	O	-	data to the host
host_cs_b	I	Low	chip select
host_rd_b(3:0)	I	Low	read control
host_wr_b(3:0)	I	Low	write control
host_ready	O	High	read / write access ready
host_irq	O	High	interrupt to the host
host_if(13:0)	O	High	Vector of individual interrupt flags (see chap. 3.8)
can_rst_n	I	Low	low-active asynchronous reset for <can_clk> domain
can_clk	I	-	clock for CAN protocol machine
rxd	I	-	receive data from bus – input signal from the transceiver IC (signals the bit level on the CAN bus) – Do NOT insert additional buffers! Directly connect it to the transceiver.
txd	O	-	transmit data to the bus – output signal to the transceiver IC (transmits bit to the CAN bus) – Do NOT insert additional buffers! Directly connect it to the transceiver.
stby	O	High	Transceiver Standby Mode Enable – No additional buffers are required. Directly connect it to the transceiver.
can_fd_enable	I	High	CAN FD conformance can be enabled (1) or disabled (0) at runtime if the necessary hardware is included by setting the parameter CAN_FD to 1 (Table 3-2). This input should stay unchanged as long a bit RESET in register CFG_STAT is 0 (inactive).
timer_rst_n	I	Low	low-active asynchronous reset for <timer_clk> domain

Name	Type	Polarity	Description
timer_clk	I	-	timer clock
timer_in(63:0)	I	-	timer input for CiA 603 time-stamping
tst_sample	O	-	test signal – shows the used sample points (activated one period after the sample point).
tst_clock	O	-	test signal – shows the used bit time periods (activated one period before the bit time starts)
tst_rec(7:0)	O	-	Receive Error Counter (for test purposes)
tst_tec(8:0)	O	-	Transmit Error Counter (for test purposes)

The 3 resets <host_rst_n>, <can_rst_n> and <timer_rst_n> are asynchronous resets. Reset synchronizers for each clock domain can be included depending on a generic parameter as explained in chap. 3.4. An input synchronizer for the <rxd> input can also be included.

CAN-CTRL includes three clock domains: one for the host interface, one for the CAN protocol machine and one for CiA 603 time-stamping. All top-level I/O signals in the host clock domain are named with the prefix “host_” and all signals in the timer clock domain are named with the prefix “timer_”. It is possible to use the same clock for all clock inputs.

For CAN FD operation the clock for the CAN protocol machine <can_clk> shall be set to 20MHz, 40MHz or 80MHz. This is a general recommendation for all CAN FD nodes. Further details are given in chap. 6.6. The other clocks <host_clk> and <timer_clk> are independent from <can_clk> and can be freely chosen.

If synthesis parameter CAN_FD (Table 3-2) is set to 1, then signal <can_fd_enable> can be used to disable the CAN FD functionality. As long as it is not intended to disable the CAN FD feature of the CAN FD core the signal needs to be set to '1'. The signal has no relevance when the core is built with CAN FD features disabled (generic parameter CAN_FD=0).

3.2 Configuration Parameters

The CAN-CTRL core is configurable before synthesis. Several parameters can be chosen as defined in Table 3-2. Synthesis parameters are located in the source files can_package_synparam.vhd / can_package_synparam.v in the src directory. Beside these selectable parameters, various other definitions are located in the files can_package.vhd / can_package.v, but these other shall not be changed. Comments are given inside the package for clarification.

Additionally there are generic parameters in the entity of CAN-CTRL. See chap. 3.4 for further details.

Table 3-2 Parameter Description (can_package_synparam.v(hd))

Parameter	Range	Description
RAM_MEMTYPE	1 to 5	Type of memory: 1 – Distributed RAM 2 – Block RAM for Xilinx ISE 3 – Block RAM for Xilinx Vivado 4 – Block RAM for Altera Quartus Prime 5 – Block RAM for Lattice
RAM_TYPE	0 to 1	Architecture of the memory for TBUF and RBUF : 0 – one true dual port memory 1 – two pseudo dual port memories 2 – two pseudo dual port memories with pseudo read port feature for TBUF 3 – one single port memory Details about the pseudo dual port memories are given in chap. 3.10. Details about dual port versus single port memory are given in chap. 3.11.

Parameter	Range	Description
RBUF_SLOTS	>0	Number of slots in the receive buffer Please note, that the core always uses one more (hidden) slot as defined with RBUF_SLOTS, which is used to receive a new message while the receive buffer is full and has not been released yet.
STB_ENABLE	0 to 1	0 – no STB included, STB_SLOTS ignored 1 – STB_SLOTS defines the number of STB message slots
STB_SLOTS	>0	Number of slots in the secondary transmit buffer (STB)
STB_PRIO	0 or 1	STB priority decision mode (see bit TSMODE) 0 – disabled 1 – enabled The priority decision machine may occupy much area.
ACF_NUMBER	1 to 16	Number of acceptance filters (ACF)
CAN_FD	0 to 1	0 – only CAN 2.0B conformance 1 – CAN FD conformance (an extension of CAN 2.0B)
UPWARD_COMPATIBILITY	0 to 1	0 – no protocol upward compatibility 1 – upward compatibility: protocol exception event (strongly recommended)
TTCAN	0 to 1	0 – disabled 1 – TTCAN, level 1
CIA603	0 to 2	0 – disabled 1 – CiA 603: 32 bit time-stamping 2 – CiA 603: 64 bit time-stamping

Depending on the chosen parameters the size of the memory used by CAN-CTRL calculates as follows. The memory is 32 bits wide. This calculation is also included in the Excel sheet <doc/can-ctrl-memory.xlsx>.

The memory for the acceptance filters requires the following number of words:

$$\mathrm{ACF\_WORDS} = 2 \cdot \mathrm{ACF\_NUMBER}$$

The memory for TBUF and RBUF requires the following number of words:

$$\mathrm{DATA\_BYTES} = 8 + \mathrm{CAN\_FD} \cdot (64 - 8)$$

$$\mathrm{TBUF\_BYTES} = 8 + \mathrm{DATA\_BYTES}$$

$$\mathrm{RBUF\_BYTES} = 8 + \mathrm{DATA\_BYTES} + \mathrm{CIA603} \cdot 4$$

$$\mathrm{TB\_WORDS} = (1 + \mathrm{STB\_ENABLE} \cdot \mathrm{STB\_SLOTS}) \cdot (\mathrm{TBUF\_BYTES}/4)$$

$$\mathrm{RB\_WORDS} = (1 + \mathrm{RBUF\_SLOTS}) \cdot (\mathrm{RBUF\_BYTES}/4)$$

In summary the overall number of required memory words is as follows:

$$\mathrm{MEM\_WORDS} = \mathrm{TB\_WORDS} + \mathrm{RB\_WORDS} + \mathrm{ACF\_WORDS}$$

CAN-CTRL uses one block of memory for all features except if the pseudo dual-port memory option is selected (see chap. 3.10).

3.3 The Configuration Parameter RAM_TYPE

This chapter is a short guideline about the options given by the generic parameter RAM_TYPE (see Table 3-2).

1. If the host controller supports wait states at its host interface, then selecting a single port memory (RAM_TYPE=3) will be the best choice, because a single port memory occupies the smallest amount of chip area.
2. If the target library offers a dual port memory and the host controller does not support wait states at its host interface, then RAM_TYPE=0 should be chosen. But a dual port memory occupies the

largest amount of chip area. Most Altera and Xilinx FPGAs have dual port memories built-it and therefore there is no additional cost for these FPGAs.

3. If the host controller does not support wait states and either dual port memory is not available (like in many Lattice FPGAs) or the required cell area for a dual port memory costs too much, then a pseudo dual port memory (see chap. 3.10 and chap. 3.11) is the right choice.
    a. If a true 32 bit host interface is used and it is not necessary to verify read written data in the transmit buffer, then RAM_TYPE=1 can be chosen.
    b. If the generic 8 bit host interface (chap. 7.2) is chosen and reading data, that have been written is required, then RAM_TYPE=2 needs to be chosen. Compared to RAM_TYPE=1 the amount of additional cell area is really small.

If only single port memory is available in the target library, but the selected host controller does not support wait states, then selecting a distributed memory (RAM_MEMTYPE=1) can be an option. This will result in a large array of flipflops used as memory. The required cell area will increase dramatically with the number of required memory words, but dual port functionality can be provided. Therefore this option is only recommended if small numbers are chosen for RBUF_SLOTS, STB_SLOTS and ACF_NUMBER.

3.4 Generic Parameters

Table 3-3 Generic Parameter Description (Entity of <can_ctrl.v(hd)>)

Parameter	Range	Description
ADD_R_SYNC	0 to 1	0 – no synchronizers included 1 – synchronizers included (default and recommended) Reset synchronizers are required to ensure, that all flipflops start with the same clock edge after the reset becomes inactive. Therefore reset synchronizers are always recommended, but this generic parameter offers the flexibility to add them outside of the IP core our include them inside. If reset synchronizers shall be included, then one single asynchronous reset signal can be connected to all reset inputs <host_rst_n>, <can_rst_n> and <timer_rst_n>.
ADD_I_SYNC	0 to 1	0 – no synchronizers included 1 – synchronizers included (default and recommended) Synchronizers for asynchronous inputs are required to avoid problems with meta-stability. But synchronizers add also signal delay which increases the complexity of a verification by simulation, e.g. with a verification IP. Therefore it is recommended to disable input synchronizers for behavioral verification in simulation if necessary. In contrast to this a real world application system requires input synchronizers, and a conformance test house needs to do a compensation for the signal delay.
ADD_O_SYNC	0 to 1	0 – no synchronizers included At the rising edge of <can_clk> if <tst_clock> is ‘1’ this marks the begin of a CAN bit and if <tst_sample> is ‘1’ this marks the position of the sample point. 1 – synchronizers included (default) The rising edge of <tst_clock> marks the begin of a CAN bit and the rising edge of <tst_sample> marks the position of the sample point. The test observation output <tst_clock> and <tst_sample> can by buffered using synchronizers.

3.5 Definitions

Table 3-4 Name Definitions

Abbreviation	Description
ACF	Acceptance Filter
PTB	Primary Transmit Buffer (high priority)
RB	Receive Buffer
RDC	Receiver Delay Compensation (related to TDC)
RTS	Reception Time Stamp
SP	Sample Point
SSP	Secondary Sample Point (CAN FD)
STB	Secondary Transmit Buffer (low priority)
TDC	Transmitter Delay Compensation (CAN FD specification)
TTCAN	Time-Triggered CAN (ISO 11898-4)
TTS	Transmission Time Stamp
TQ	Time Quanta (CAN specification)

All register definitions are given including an access definition. Access definitions are given using an abbreviation in the form <access>-<reset>. Possible abbreviations for the <access> attribute are “r” for read, “w” for write and “rw” for read / write access. The <reset> attribute can be “0”, “1” and “u” for uninitialized registers. Example: “rw-0” means “readable and writeable and reset to 0” while “r-u” means “readable and uninitialized”. Unused bits are always r-0.

3.6 Clock Domain Crossing

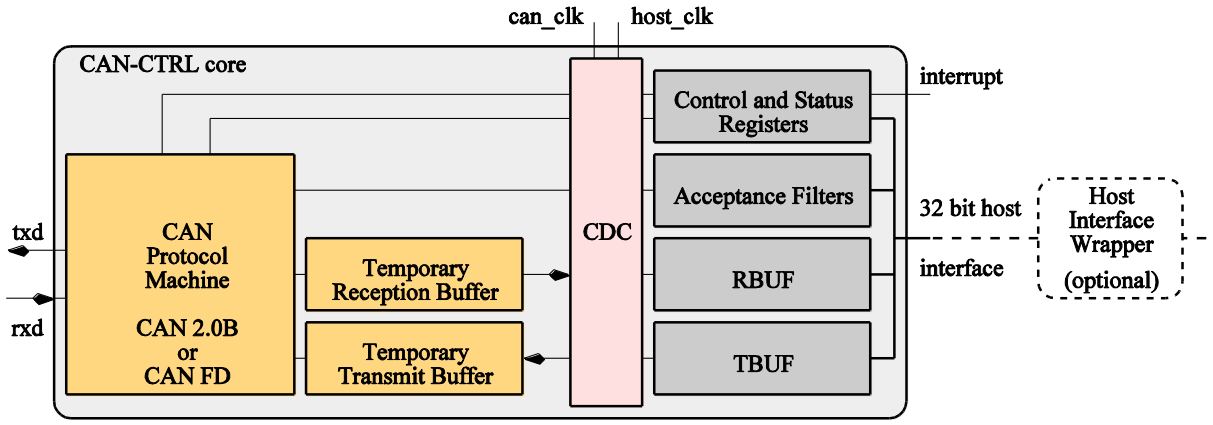

Figure 3-2 Clock Domain Crossing

The core uses two clock domains: one for the host interface and one for the CAN protocol machine. This enables the option to operate the CAN protocol machine with a clock that fits for the desired CAN bus data rates best while the host is free to operate at a different (higher) clock speed.

The host interface as well as optional host interface wrappers (e.g. for AMBA busses) operate in the host clock domain. This includes all memories and registers that are accessible using the host data bus (chap. 3.7). Clock Domain Crossing (CDC) is done with double-buffered synchronizers and bi-directional handshake mechanisms. The following gives some details about the CDC:

- Control and status registers are moved to the appropriate clock domain individually. E.g. interrupt flags use a bi-directional handshake while status registers use unidirectional synchronizers.
- Acceptance filtering is done in the host clock domain and the required control signals are transferred using CDC.

- Received frames are stored in a small temporary reception buffer. This temporary buffer holds only parts of the frames and these parts are copied to RBUF when necessary.
- Frames to be transmitted are copied step-by-step from the TBUF to a small temporary transmit buffer.

Using synchronizers and handshake mechanisms results in some latency. Up to 3 clocks in each domain are required for a bi-directional handshake. But in most cases for the host controller this will not be problem.

3.7 Software Interface

CAN-CTRL has been initially developed as peripheral component for 8 bit systems and therefore control and status registers defined as 8 bit groups (Table 3-5). Nevertheless CAN-CTRL is a 32 bit component and offers downward-compatible interfaces for 8 and 16 bit hosts. The mapping of the registers for 8 / 16 / 32 host interfaces is shown in Figure 3-3. Chapter 7 provides details about the various host interfaces.

Table 3-5 Register Map

	Bit position								Register name
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0x00 to 0x4f	Receive Buffer Registers and Reception Time Stamp								RBUF (and RTS)
0x50 to 0x97	Transmit Buffer Registers								TBUF
0x98 to 0x9f	Transmission Time Stamp								TTS
0xa0	RESET	LBME	LBMI	TPSS	TSSS	RACTIVE	TACTIVE	BUSOFF	CFG_STAT
0xa1	TBSEL	LOM	STBY	TPE	TPA	TSONE	TSALL	TSA	TCMD
0xa2	FD_ISO *)	TSNEXT	TSMODE	TTTBM	-		TSSTAT(1:0)		TCTRL
0xa3	SACK	ROM	ROV	RREL	RBALL	-	RSTAT(1:0)		RCTRL
0xa4	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	TSFF	RTIE
0xa5	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF	RTIF
0xa6	EWARN	EPASS	EPIE	EPIF	ALIE	ALIF	BEIE	BEIF	ERRINT
0xa7	AFWL(3:0)				EWL(3:0)				LIMIT
0xa8	S_Seg_1(7:0)								S_Seg_1 *)
0xa9	-	S_Seg_2(6:0)							S_Seg_2 *)
0xaa	-	S_SJW(6:0)							S_SJW *)
0xab	S_PRESC(7:0)								S_PRESC *)
0xac	-			F_Seg_1(4:0)					F_Seg_1 *)
0xad	-			F_Seg_2(3:0)					F_Seg_2 *)
0xae	-			F_SJW(3:0)					F_SJW *)
0xaf	F_PRESC(7:0)								F_PRESC *)
0xb0	KOER(2:0)			ALC(4:0)					EALCAP
0xb1	TDCEN	SSPOFF(6:0)							TDC *)
0xb2	RECNT								RECNT
0xb3	TECNT								TECNT
0xb4	-		SELMASK	-	ACFADR				ACFCTRL
0xb5	-						TIMEPOS	TIMEEN	TIMECFG
0xb6	AE_7	AE_6	AE_5	AE_4	AE_3	AE_2	AE_1	AE_0	ACF_EN_0
0xb7	AE_15	AE_14	AE_13	AE_12	AE_11	AE_10	AE_9	AE_8	ACF_EN_1
0xb8	ACODE_x or AMASK_x (7:0)								ACF_0 *)
0xb9	ACODE_x or AMASK_x (15:8)								ACF_1 *)
0xba	ACODE_x or AMASK_x (23:16)								ACF_2 *)
0xbb	-	AIDEE	AIDE	ACODE_x or AMASK_x (28:24)					ACF_3 *)
0xbc	VERSION(7:0)								VER_0
0xbd	VERSION(15:8)								VER_1
0xbe	TBE	TBF	TBPTR(5:0)						TBSLOT
0xbf	WTIE	WTIF	TEIF	TTIE	TTIF	T_PRESC(1:0)		TTEN	TTCFG
0xc0	REF_ID(7:0)								REF_MSG_0
0xc1	REF_ID(15:8)								REF_MSG_1
0xc2	REF_ID(23:16)								REF_MSG_2
0xc3	REF_IDE	-	REF_ID(28:24)						REF_MSG_3

	Bit position				Register name
0xc4	-	TTPTR(5:0)			TRIG_CFG_0
0xc5	TEW(3:0)		-	TTYPE(2:0)	TRIG_CFG_1
0xc6	TT_TRIG(7:0)				TT_TRIG_0
0xc7	TT_TRIG(15:8)				TT_TRIG_1
0xc8	TT_WTRIG(7:0)				TT_WTRIG_0
0xc9	TT_WTRIG(15:8)				TT_WTRIG_1
0xca	-				-
0xcb	-				-

*) register can only be written if bit RESET in register CFG_STAT is set.

An write access to an addressable location not shown in the register map (Table 3-5) results in no action and a read access will result in the value 0x00.

Register addresses are given as an example. They can be selected using generic parameters before synthesis. The example shows the default setting. The example for 8 / 16 / 32 hosts in Figure 3-3 assumes that every byte is addressable even if the host uses a wider native width.

Please mind the several gaps inside the register map. This is for better address segment alignment for a wider host controller interface.

Please have in mind that configuration registers that are only for CAN FD cannot be used if the IP core is reduced to only CAN 2.0B capability. Then these registers are tied to their reset values.

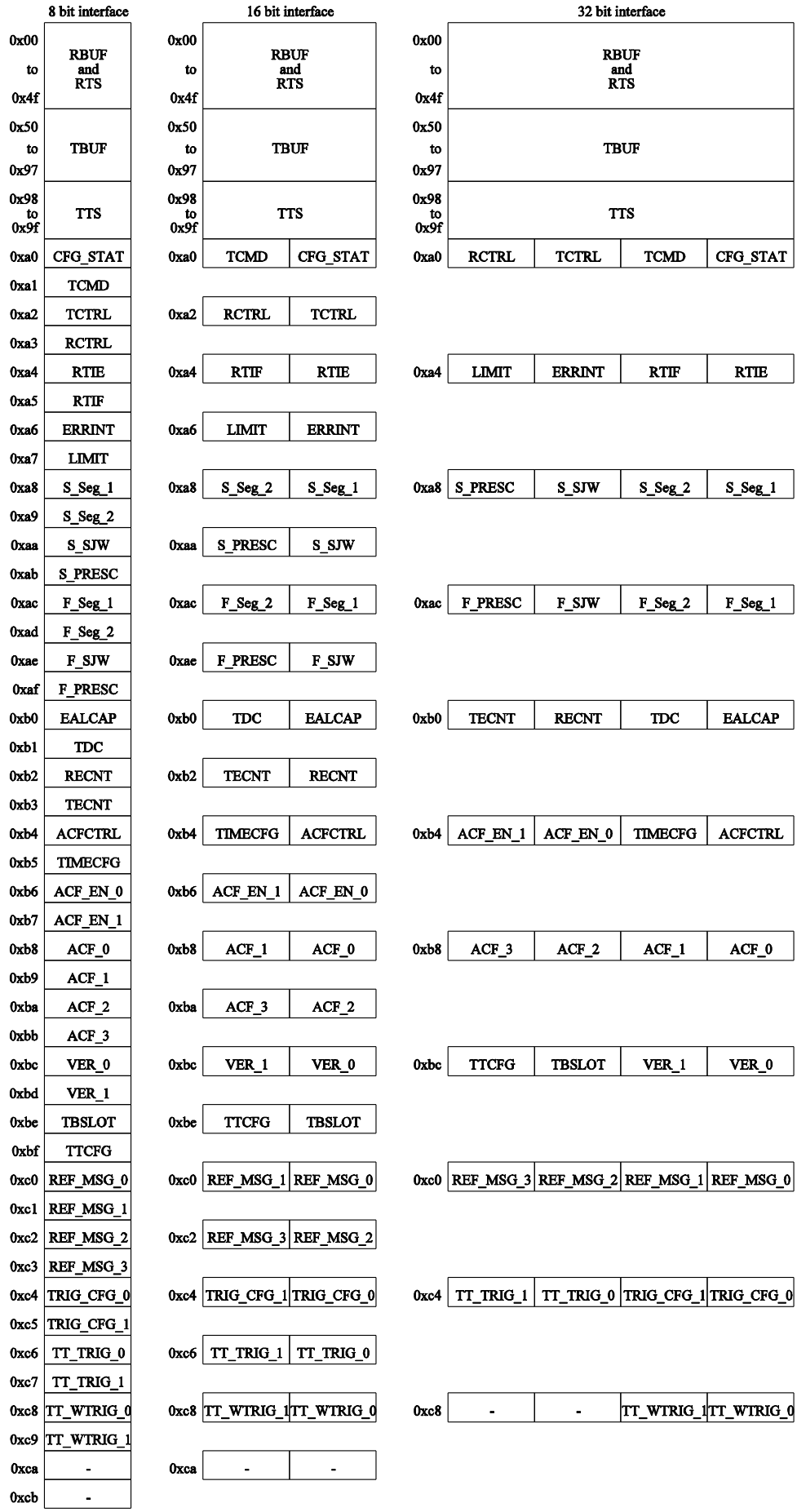

Figure 3-3 Memory Map for 8 / 16 / 32 Bit Interface

Table 3-6 Transmission Time Stamp TTS (0x98 to 0x9f)

Bits	Name	Access	Function
63:0	TTS	r-0	Transmission Time Stamp TTS holds the time stamp of the last transmitted frame for CiA 603 time stamping. Every new frame overwrites TTS if TTSEN=1. Depending on a generic parameter (chap. 3.2) the time-stamp can be 32 or 64 wide. Unused bits are forced to 0. The TTS is intended to be used by the time master to acquire the time-stamp of the SYNC message.

Table 3-7 Configuration and Status Register CFG_STAT (0xa0)

Bits	Name	Access	Function
7	RESET	rw-1	RESET request bit 1 - The host controller performs a local reset of CAN-CTRL. 0 - no local reset of CAN-CTRL The some register (e.g for node configuration) can only be modified if RESET=1. Bit RESET forces several components to a reset state. A detailed definition is given in chap. 3.9.11 . RESET is automatically set if the node enters "bus off" state (chap. 3.9.7). Note that a CAN node will participate in CAN communication after RESET is switched to 0 after 11 CAN bit times. This delay is required by the CAN standard (bus idle time). If RESET is set to 1 and immediately set to 0, then it takes some time until RESET can be read as 0 and becomes inactive. The reason is clock domain crossing from host to CAN clock domain. RESET is held active as long as needed depending on the relation between host and CAN clock.
6	LBME	rw-0	Loop Back Mode, External (chap. 3.9.10.4) 0 - Disabled 1 - Enabled LBME should not be enabled while a transmission is active.
5	LBMI	rw-0	Loop Back Mode, Internal (chap. 3.9.10.4) 0 - Disabled 1 - Enabled LBMI should not be enabled while a transmission is active.
4	TPSS	rw-0	Transmission Primary Single Shot mode for PTB (chap. 3.9.10.1) 0 - Disabled 1 - Enabled
3	TSSS	rw-0	Transmission Secondary Single Shot mode for STB (chap. 3.9.10.1) 0 - Disabled 1 - Enabled
2	RACTIVE	r-0	Reception ACTIVE (Receive Status bit) 1 - The controller is currently receiving a frame. 0 - No receive activity.
1	TACTIVE	r-0	Transmission ACTIVE (Transmit Status bit) 1 - The controller is currently transmitting a frame. 0 - No transmit activity.
0	BUSOFF	rw-0	Bus Off (Bus Status bit, chap. 3.9.7) 1 - The controller status is "bus off". 0 - The controller status is "bus on". Writing a 1 to BUSOFF will reset TECNT and RECNT. This should be done only for debugging. See chap. 3.9.10.6 for details.

Table 3-8 Command Register TCMD (0xa1)

Bits	Name	Access	Function
7	TBSEL	rw-0	Transmit Buffer Select Selects the transmit buffer to be loaded with a message. Use the TBUF registers for access. TBSEL needs to be stable all the time the TBUF registers are written and when TSNEXT is set. 0 - PTB (high-priority buffer) 1 - STB The bit will be reset to the hardware reset value if (TTEN=1 and TTTBM=1).

Bits	Name	Access	Function
6	LOM	rw-0	Listen Only Mode (chap. 3.9.10.2) 0 - Disabled 1 - Enabled LOM cannot be set if TPE, TSONE or TSALL is set. No transmission can be started if LOM is enabled and LBME is disabled. LOM=1 and LBME=0 disables all transmissions. LOM=1 and LBME=1 disables the ACK for received frames and error frames, but enables the transmission of own frames.
5	STBY	rw-0	Transceiver Standby Mode (chap. 3.9.10.5) 0 - Disabled 1 - Enabled This register bit is connected to the output signal stby which can be used to control a standby mode of a transceiver. STBY cannot be set to 1 if TPE=1, TSONE=1 or TSALL=1. If the host sets STBY to 0 then the host needs to wait for the time required by the transceiver to start up before the host requests a new transmission.
4	TPE	rw-0	Transmit Primary Enable 1 - Transmission enable for the message in the high-priority PTB 0 - No transmission for the PTB If TPE is set, the message from the PTB will be transmitted at the next possible transmit position. A started transmission from the STB will be completed before, but pending new messages are delayed until the PTB message has been transmitted. TPE stays set until the message has been transmitted successfully or it is aborted using TPA. The host controller can set TPE to 1 but can not reset it to 0. This would only be possible using TPA and aborting the message. The bit will be reset to the hardware reset value if RESET=1, STBY=1, (LOM=1 and LBME=0) or (TTEN=1 and TTTBM=1).
3	TPA	rw-0	Transmit Primary Abort 1 – Aborts a transmission from PTB which has been requested by TPE=1 but not started yet. (The data bytes of the message remains in the PTB.) 0 – no abort The bit has to be set by the host controller and will be reset by CAN-CTRL. Setting TPA automatically de-asserts TPE. The host controller can set TPA to 1 but can not reset it to 0. During the short time while the CAN-CTRL core resets the bit, it cannot be set by the host. The bit will be reset to the hardware reset value if RESET=1 or (TTEN=1 and TTTBM=1). TPA should not be set simultaneously with TPE.
2	TSONE	rw-0	Transmit Secondary ONE frame 1 – Transmission enable of one in the STB. In FIFO mode this is the oldest message and in priority mode this is the one with the highest priority. TSONE in priority mode is difficult to handle, because it is not always clear which message will be transmitted if new messages are written to the STB meanwhile. The controller starts the transmission as soon as the bus becomes vacant and no request of the PTB (bit TPE) is pending. 0 – No transmission for the STB. TSONE stays set until the message has been transmitted successfully or it is aborted using TSA. The host controller can set TSONE to 1 but can not reset it to 0. This would only be possible using TSA and aborting the message. The bit will be reset to the hardware reset value if RESET=1, STBY=1, (LOM=1 and LBME=0) or (TTEN=1 and TTTBM=1).
1	TSALL	rw-0	Transmit Secondary ALL frames 1 – Transmission enable of all messages in the STB. The controller starts the transmission as soon as the bus becomes vacant and no request of the PTB (bit TPE) is pending. 0 – No transmission for the STB. TSALL stays set until all messages have been transmitted successfully or they are aborted using TSA. The host controller can set TSALL to 1 but can not reset it to 0. This would only be possible using TSA and aborting the messages. The bit will be reset to the hardware reset value if RESET=1, STBY=1, (LOM=1 and LBME=0) or (TTEN=1 and TTTBM=1).

Bits	Name	Access	Function
			If during a transmission the STB is loaded with a new frame then the new frame will be transmitted too. In other words: a transmission initiated by TSALL is finished when the STB becomes empty.
0	TSA	rw-0	Transmit Secondary Abort 1 – Aborts a transmission from STB which has been requested but not started yet. For a TSONE transmission, only one frame is aborted while for a TSALL Transmission, all frames are aborted. One or all message slots will be released which updates TSSTAT. All aborted messages are lost because they are not accessible any more. If in priority mode a TSONE transmission is aborted, then it is not clear which frame will be aborted if new frames are written to the STB meanwhile. 0 – no abort The bit has to be set by the host controller and will be reset by CAN-CTRL. Setting TSA, automatically de-asserts TSONE or TSALL respectively. The host controller can set TSA to 1 but can not reset it to 0. The bit will be reset to the hardware reset value if RESET=1. TSA should not be set simultaneously with TSONE or TSALL.

Setting both TSONE and TSALL is meaningless. While TSALL is already set, it is impossible to set TSONE and vice versa. If both TSONE and TSALL are set simultaneously then TSALL wins and TSONE is cleared by the CAN-CTRL core.

Table 3-9 Transmit Control Register TCTRL (0xa2)

Bits	Name	Access	Function
7	FD_ISO	rw-1	CAN FD ISO mode 0 - Bosch CAN FD (non-ISO) mode 1 - ISO CAN FD mode (ISO 11898-1:2015) ISO CAN FD mode has a different CRC initialization value and an additional stuff bit count. Both modes are incompatible and must not be mixed in one CAN network. This bit has no impact to CAN 2.0B. This bit is only writeable if RESET=1.
6	TSNEXT	rw-0	Transmit buffer Secondary NEXT 0 - no action 1 - STB slot filled, select next slot. After all frame bytes are written to the TBUF registers, the host controller has to set TSNEXT to signal that this slot has been filled. Then the CAN-CTRL core connects the TBUF registers to the next slot. Once a slot is marked as filled a transmission can be started using TSONE or TSALL. It is possible to set TSNEXT and TSONE or TSALL together in one write access. TSNEXT has to be set by the host controller and is automatically reset by the CAN-CTRL core immediately after it was set. Setting TSNEXT is meaningless if TBSEL=0. In this case TSNEXT is ignored and automatically cleared. It does not do any harm. If all slots of the STB are filled, TSNEXT stays set until a slot becomes free (chap. 3.9.6). TSNEXT has no meaning in TTCAN mode and is fixed to 0.
5	TSMODE	rw-0	Transmit buffer Secondary operation MODE 0 - FIFO mode 1 - priority decision mode In FIFO mode frames are transmitted in the order in that they are written into the STB. In priority decision mode the frame with the highest priority in the STB is automatically transmitted first. The ID of a frame is used for the priority decision. A lower ID means a higher priority of a frame. A frame in the PTB has always the highest priority regardless of the ID. TSMODE shall be switched only if the STB if empty.
4	TTTBM	rw-1	TTCAN Transmit Buffer Mode If TTEN=0 then TTTBM is ignored, otherwise the following is valid: 0 - separate PTB and STB, behavior defined by TSMODE 1 - full TTCAN support: buffer slots selectable by TBPTR and TTPTR For event-driven CAN communication (TTEN=0), the system provides PTB and STB and the behavior of the STB is defined by TSMODE. Then TTTBM is ignored. For time-triggered CAN communication (TTEN=1) with full support of all features including

Bits	Name	Access	Function
			time-triggered transmissions, TTTBM=1 needs to be chosen. Then the all TB slots are addressable using TTPTR and TBPTR. For time-triggered CAN communication (TTEN=1) with only support of reception time-stamps, TTTBM=0 can be chosen. Then the transmit buffer acts as in event-driven mode and the behavior can be selected by TSMODE. TTTBM shall be switched only if the TBUF is empty.
3:2	-	r-0	reserved
1:0	TSSTAT	r-0	Transmission Secondary STATus bits If TTEN=0 or TTTBM=0: 00 – STB is empty 01 – STB is less than or equal to half full 10 – STB is more than half full 11 – STB is full If the STB is disabled using STB_DISABLE, then TSSTAT=00. If TTEN=1 and TTTBM=1: 00 – PTB and STB are empty 01 – PTB and STB are not empty and not full 11 – PTB and STB are full

Table 3-10 Receive Control Register RCTRL (0xa3)

Bits	Name	Access	Function
7	SACK	rw-0	Self-ACKnowledge 0 – no self-ACK 1 – self-ACK when LBME=1
6	ROM	rw-0	Receive buffer Overflow Mode In case of a full RBUF when a new message is received, then ROM selects the following: 1 – The new message will not be stored. 0 – The oldest message will be overwritten.
5	ROV	r-0	Receive buffer OVerflow 1 – Overflow. At least one message is lost. 0 – No Overflow. ROV is cleared by setting RREL=1.
4	RREL	rw-0	Receive buffer RELease The host controller has read the actual RB slot and releases it. Afterwards the CAN-CTRL core points to the next RB slot. RSTAT gets updated. 1 – Release: The host has read the RB. 0 – No release
3	RBALL	rw-0	Receive Buffer stores ALL data frames 0 – normal operation 1 – RB stores correct data frames as well as data frames with error (chap. 3.9.9.4)
2	-	r-0	Reserved
1:0	RSTAT	r-0	Receive buffer STATus 00 - empty 01 - > empty and < almost full (AFWL) 10 - ≥ almost full (programmable threshold by AFWL) but not full and no overflow 11 - full (stays set in case of overflow – for overflow signaling see ROV)

Table 3-11 Receive and Transmit Interrupt Enable Register RTIE (0xa4)

Bits	Name	Access	Function
7	RIE	rw-1	Receive Interrupt Enable 0 – Disabled, 1 – Enabled
6	ROIE	rw-1	RB Overrun Interrupt Enable 0 – Disabled, 1 – Enabled
5	RFIE	rw-1	RB Full Interrupt Enable 0 – Disabled, 1 – Enabled
4	RAFIE	rw-1	RB Almost Full Interrupt Enable 0 – Disabled, 1 – Enabled
3	TPIE	rw-1	Transmission Primary Interrupt Enable 0 – Disabled, 1 – Enabled
2	TSIE	rw-1	Transmission Secondary Interrupt Enable

Bits	Name	Access	Function
			0 – Disabled, 1 – Enabled
1	EIE	rw-1	Error Interrupt Enable 0 – Disabled, 1 – Enabled
0	TSFF	r-0	If TTEN=0 or TTTBM=0: Transmit Secondary buffer Full Flag 1 - The STB is filled with the maximal number of messages. 0 - The STB is not filled with the maximal number of messages If the STB is disabled using STB_DISABLE, then TSFF=0. If TTEN=1 and TTTBM=1: Transmit buffer Slot Full Flag 1 - The buffer slot selected by TBPTR is filled. 0 - The buffer slot selected by TBPTR is empty.

Table 3-12 Receive and Transmit Interrupt Flag Register RTIF (0xa5)

Bits	Name	Access	Function
7	RIF	rw-0	Receive Interrupt Flag 1 - Data or a remote frame has been received and is available in the receive buffer. 0 - No frame has been received.
6	ROIF	rw-0	RB Overrun Interrupt Flag 1 - At least one received message has been overwritten in the RB. 0 - No RB overwritten. In case of an overrun both ROIF and RFIF will be set.
5	RFIF	rw-0	RB Full Interrupt Flag 1 - All RBs are full. If no RB will be released until the next valid message is received, the oldest message will be lost. 0 - The RB FIFO is not full.
4	RAFIF	rw-0	RB Almost Full Interrupt Flag 1 - number of filled RB slots ≥ AFWL_i 0 - number of filled RB slots < AFWL_i
3	TPIF	rw-0	Transmission Primary Interrupt Flag 1 - The requested transmission of the PTB has been successfully completed. 0 - No transmission of the PTB has been completed. In TTCAN mode, TPIF will never be set. Then only TSIF is valid.
2	TSIF	rw-0	Transmission Secondary Interrupt Flag 1 - The requested transmission of the STB has been successfully completed. 0 - No transmission of the STB has been completed successfully. In TTCAN mode TSIF will signal all successful transmissions, regardless of storage location of the message.
1	EIF	rw-0	Error Interrupt Flag 1 - The border of the error warning limit has been crossed in either direction, or the BUSOFF bit has been changed in either direction. 0 - There has been no change.
0	AIF	rw-0	Abort Interrupt Flag 1 - After setting TPA or TSA the appropriated message(s) have been aborted. It is recommended to not set both TPA and TSA simultaneously because both source AIF. 0 - No abort has been executed. The AIF does not have an associated enable register. See also chap. 3.9.5 for further information.

To reset an interrupt flag, the host controller needs to write an 1 to the flag. Writing a 0 has no effect. If a new interrupt event occurs while the write access is active then this event will set the flag and override the reset. This ensures that no interrupt event is lost.

Interrupt flags will only be set if the associated interrupt enable bit is set.

Table 3-13 ERRor INTerrupt Enable and Flag Register ERRINT (0xa6)

Bits	Name	Access	Function
7	EWARN	r-0	Error WARNing limit reached 1 - One of the error counters RECNT or TECNT is equal or bigger than EWL 0 - The values in both counters are less than EWL.
6	EPASS	r-0	Error Passive mode active 0 - not active (node is error active) 1 - active (node is error passive)
5	EPIE	rw-0	Error Passive Interrupt Enable
4	EPIF	rw-0	Error Passive Interrupt Flag. EPIF will be activated if the error status changes from error active to error passive or vice versa and if this interrupt is enabled.
3	ALIE	rw-0	Arbitration Lost Interrupt Enable
2	ALIF	rw-0	Arbitration Lost Interrupt Flag
1	BEIE	rw-0	Bus Error Interrupt Enable
0	BEIF	rw-0	Bus Error Interrupt Flag

To reset an interrupt flag, the host controller needs to write an 1 to the flag. Writing a 0 has no effect. If a new interrupt event occurs while the write access is active then this event will set the flag and override the reset. This ensures that no interrupt event is lost.

Interrupt flags will only be set if the associated interrupt enable bit is set.

Table 3-14 Warning Limits Register LIMIT (0xa7)

Bits	Name	Access	Function
7:4	AFWL(3:0)	rw-0x1	receive buffer Almost Full Warning Limit AFWL defines the internal warning limit AFWL_i with n_{RB} being the number of available RB slots. $\mathrm{AFWL\_i} = \begin{cases} \mathrm{AFWL} & \mid \quad n_{RB} < 16 \\ 2 \cdot \mathrm{AFWL} & \mid \quad 16 \le n_{RB} < 32 \\ 4 \cdot \mathrm{AFWL} & \mid \quad 32 \le n_{RB} < 64 \\ \vdots & \quad \vdots \end{cases}$ $\mathrm{AFWL\_i}$ is compared to the number of filled RB slots and triggers RAFIF if equal. The valid range of $\mathrm{AFWL\_i} = [1 \ldots n_{RB}]$. $\mathrm{AFWL} = 0$ is meaningless and automatically treated as 0x1. (Note that AFWL is meant in this rule and not AFWL_i.) $\mathrm{AFWL\_i} > n_{RB}$ is meaningless and automatically treated as n_{RB}. $\mathrm{AFWL\_i} = n_{RB}$ is a valid value, but note that RFIF also exists.
3:0	EWL(3:0)	rw-0xB	Programmable Error Warning Limit = (EWL+1)*8. Possible Limit values: 8, 16, … 128. The value of EWL controls EIF. EWL needs to be transferred using CDC from host to CAN clock domain. During transfer EWL register bits are write-locked for the host for a few clocks until CDC is complete.

Table 3-15 Bit Timing Register S_Seg_1 (0xa8)

Bits	Name	Access	Function
7:0	S_Seg_1(7:0)	rw-0x3	Bit Timing Segment 1 (slow speed) The sample point will be set to $t_{Seg\_1} = (\mathrm{Seg\_1} + 2) \cdot TQ$ after start of bit time.

Table 3-16 Bit Timing Register S_Seg_2 (0xa9)

Bits	Name	Access	Function
7	-	r-0	reserved
6:0	S_Seg_2(6:0)	rw-0x2	Bit Timing Segment 2 (slow speed) Time $t_{\text{Seg\_2}} = (\text{Seg\_2} + 1) \cdot TQ$ after the sample point.

Table 3-17 Bit Timing Register S_SJW (0xaa)

Bits	Name	Access	Function
7	-	r-0	reserved
6:0	S_SJW(6:0)	rw-0x2	Synchronization Jump Width (slow speed) The Synchronization Jump Width $t_{SJW} = (SJW + 1) \cdot TQ$ is the maximum time for shortening or lengthening the Bit Time for resynchronization, where TQ is a time quanta.

Table 3-18 Bit Timing Register F_Seg_1 (0xac)

Bits	Name	Access	Function
7:5	-	r-0	reserved
4:0	F_Seg_1(4:0)	rw-0x3	Bit Timing Segment 1 (fast speed) The sample point will be set to $t_{\text{Seg\_1}} = (\text{Seg\_1} + 2) \cdot TQ$ after start of bit time.

Table 3-19 Bit Timing Register F_Seg_2 (0xad)

Bits	Name	Access	Function
7:4	-	r-0	reserved
3:0	F_Seg_2(3:0)	rw-0x2	Bit Timing Segment 2 (fast speed) Time $t_{\text{Seg\_2}} = (\text{Seg\_2} + 1) \cdot TQ$ after the sample point.

Table 3-20 Bit Timing Register F_SJW (0xae)

Bits	Name	Access	Function
7:4	-	r-0	reserved
3:0	F_SJW(3:0)	rw-02	Synchronization Jump Width (fast speed) The Synchronization Jump Width $t_{SJW} = (SJW + 1) \cdot TQ$ is the maximum time for shortening or lengthening the Bit Time for resynchronization, where TQ is a time quanta.

Table 3-21 Prescaler Registers S_PRESC (0xab) and F_PRESC (0xaf)

Bits	Name	Access	Function
7:0	S_PRESC F_PRESC	rw-0x01	Prescaler (slow and fast speed) The prescaler divides the system clock to get the time quanta clock tq_clk. Valid range PRESC=[0x00, 0xff] results in divider values 1 to 256.

Table 3-22 Transmitter Delay Compensation Register TDC (0xb1)

Bits	Name	Access	Function
7	TDCEN	rw-0	Transmitter Delay Compensation ENable TDC will be activated during the data phase of a CAN FD frame if BRS is active if TDCEN=1. For more details about TDC see chap. 6.5.
6:0	SSPOFF	rw-0x00	Secondary Sample Point OFFset The transmitter delay plus SSPOFF defines the time of the secondary sample point for TDC. SSPOFF is given as a number of TQ.

Writing to S_Seg_1, S_Seg_2, S_SJW, S_PRESC, F_Seg_1, F_Seg_2, F_SJW, F_PRESC and TDC is only possible if RESET=1. A detailed description of the CAN bus bit timing is given in chap. 3.10. The reset value sets the bit timing which is described in the example in chap. 6.3.

All timing parameters in are given for slow (prefix “S_”) and fast speed (prefix “F_”). Slow speed is used for CAN 2.0 and the CAN FD arbitration phase. Fast speed is used for the CAN FD data phase.

Table 3-23 Error and Arbitration Lost Capture Register EALCAP (0xb0)

Bits	Name	Reset Value	Function
7:5	KOER(2:0)	r-0x0	Kind Of ERror (Error code) 000 - no error 001 - BIT ERROR 010 - FORM ERROR 011 - STUFF ERROR 100 - ACKNOWLEDGEMENT ERROR 101 - CRC ERROR 110 - OTHER ERROR (dominant bits after own error flag, received active Error Flag too long, dominant bit during Passive-Error-Flag after ACK error) 111 - not used KOER is updated with each new error. Therefore it stays untouched when frames are successfully transmitted or received.
4:0	ALC(4:0)	r-0x0	Arbitration Lost Capture (bit position in the frame where the arbitration has been lost)

Table 3-24 Error Counter Registers RECNT (0xb2) and TECNT (0xb3)

Bits	Name	Access	Function
7:0	RECNT	r-0x00	Receive Error CouNT (number of errors during reception) RECNT is incremented and decremented as defined in the CAN specification. RECNT does not overflow. See chap. 3.9.7 for more details about RECNT and the “bus off” state.
7:0	TECNT	r-0x00	Transmit Error CouNT (number of errors during transmission) TECNT is incremented and decremented as defined in the CAN specification. In case of the “bus off state” TECNT may overflow. See chap. 3.9.7 for more details about TECNT and the “bus off” state.

Table 3-25 Acceptance Filter Control Register ACFCTRL (0xb4)

Bits	Name	Access	Function
7:6	-	r-0	reserved
5	SELMASK	rw-0	SELect acceptance MASK 0 - Registers ACF_x point to acceptance code 1 - Registers ACF_x point to acceptance mask. ACFADR selects one specific acceptance filter. (See Figure 3-4.)
4	-	r-0	Reserved
3:0	ACFADR	rw-0	acceptance filter address ACFADR points to a specific acceptance filter. The selected filter is accessible using the registers ACF_x. Bit SELMASK selects between acceptance code and mask for the selected acceptance filter. (See Figure 3-4.) A value of ACFADR>NR_OF_ACF-1 is meaningless and automatically treated as value NR_OF_ACF-1. (See chap. 0 and Figure 3-4 for details about NR_OF_ACF.)

The acceptance filter registers ACF_x provide access to the acceptance filter codes ACODE_x and acceptance filter masks AMASK_x depending on the setting of SELMASK. (See Figure 3-4 and also Table 3-5). Write access to ACF_x is only possible if RESET=1. If the pseudo dual-port memories are selected (see chap. 3.10 for details), then read access is also only possible if RESET=1.

The acceptance filters are build using a true 32 bit wide memory and therefore a write access needs to be performed as 32 bit write. See chap. 7.1.3 for further details.

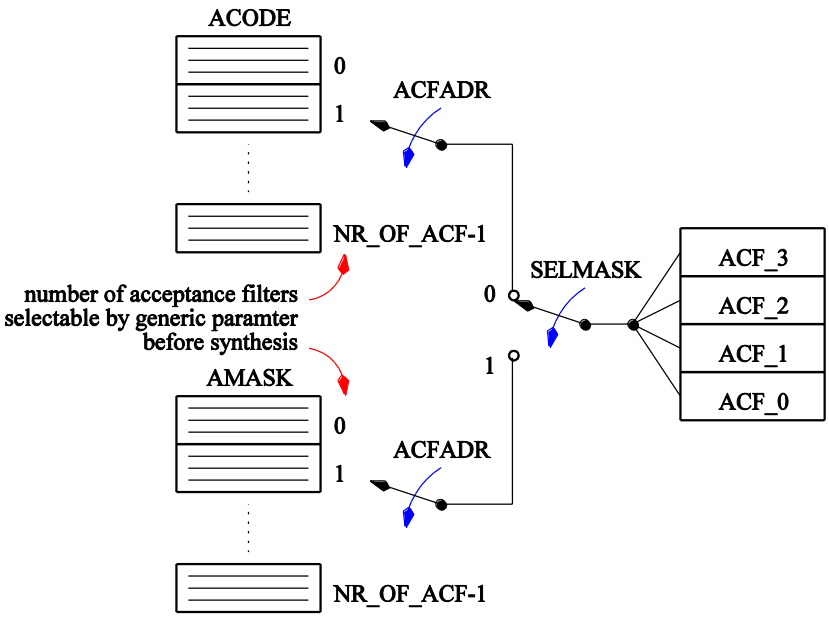

Figure 3-4 Access to the Acceptance Filters

Table 3-26 Acceptance CODE ACODE_x (register ACF_x (0xb8 to 0xbb))

Bits	Name	Access	Function
7:0	ACODE_0 ACODE_x	rw-0x00 rw-u	Acceptance CODE 1 - ACC bit value to compare with ID bit of the received message 0 - ACC bit value to compare with ID bit of the received message ACODE_x(10:0) will be used for extended frames. ACODE_x(28:0) will be used for extended frames. Only filter 0 is affected by the power-on reset. All other filters stay uninitialized. See chap. 3.9.1 for further details.

Table 3-27 Acceptance MASK AMASK_x (register ACF_x (0xb8 to 0xbb))

Bits	Name	Access	Function
7:0	AMASK_0 AMASK_x	rw-0xFF rw-u	Acceptance MASK 1 - acceptance check for these bits of receive identifier disabled 0 - acceptance check for these bits of receive identifier enable AMASK_x(10:0) will be used for extended frames. AMASK_x(28:0) will be used for extended frames. Disabled bits result in accepting the message. Therefore the default configuration after reset for filter 0 accepts all messages. Only filter 0 is affected by the power-on reset. All other filters stay uninitialized. See chap. 3.9.1 for further details.

The AMASK_x includes additional bits in register ACF_3 (Table 3-28) which can be only accessed if SELMASK=1. These bits can be used to accept only either standard or extended frames with the selected ACODE / AMASK setting or to accept both frame types. Only acceptance filter 0 is affected by the power-on reset and it is configured to accept both frame types after power-up.

Table 3-28 Bits in Register ACF_3, if SELMASK=1

Bits	Name	Access	Function
6	AIDEE	rw-0 rw-u	Acceptance mask IDE bit check enable 1 - acceptance filter accepts either standard or extended as defined by AIDE 0 - acceptance filter accepts both standard or extended frames Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.
5	AIDE	rw-0 rw-u	Acceptance mask IDE bit value If AIDEE=1 then: 1 - acceptance filter accepts only extended frames 0 - acceptance filter accepts only standard frames Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.

Table 3-29 Acceptance Filter Enable ACF_EN_0 (0xb6)

Bits	Name	Access	Function
7:0	AE_x	rw-0x01	Acceptance filter Enable 1 - acceptance filter enabled 0 - acceptance filter disable Each acceptance filter (AMASK / ACODE) can be individually enabled or disabled. Only filter number 0 is enabled by default after hardware reset. Disabled filters reject a message. Only enabled filters can accept a message if the appropriate AMASK / ACODE configuration matches. To accept all messages one filter x has to be enabled by setting AE_x=1, AMASK_x=0xff and ACODE_x=0x00. This is the default configuration after hardware reset for filter x=0 while all other filters are disabled.

Table 3-30 Acceptance Filter Enable ACF_EN_1 (0xb7)

Bits	Name	Access	Function
7:0	AE_x	rw-0x00	Acceptance filter Enable 1 - acceptance filter enabled 0 - acceptance filter disable Each acceptance filter (AMASK / ACODE) can be individually enabled or disabled. Disabled filters reject a message. Only enabled filters can accept a message if the appropriate AMASK / ACODE configuration matches.

Table 3-31 Version Information VER_0 (0xbc) and VER_1 (0xbd)

Bits	Name	Access	Function
15:0	VER_0 VER_1	r	Version of CAN-CTRL, given as decimal value. VER_1 holds the major version and VER_0 the minor version. Example: version 5x16N00S00 is represented by VER_1=5 and VER_0=16.

Table 3-32 CiA 603 Time-Stamping TIMECFG (0xb5)

Bits	Name	Access	Function
7:2	-	r-0	Reserved
1	TIMEPOS	rw-1	TIME-stamping POSition 0 – SOF 1 – EOF (see chap. 5) TIMEPOS can only be changed if TIMEEN=0, but it is possible to modify TIMPOS with the same write access that sets TIMEEN=1.
0	TIMEEN	rw-0	TIME-stamping ENable 0 – disabled 1 – enabled

Table 3-33 TTCAN: TB Slot Pointer TBSLOT (0xbe)

Bits	Name	Access	Function
7	TBE	rw-0	set TB slot to "Empty" 1 - slot selected by TBPTR shall be marked as "empty" 0 - no action TBE is automatically reset to 0 as soon as the slot is marked as empty and TSFF=0. If a transmission from this slot is active, then TBE stays set as long as either the transmission completes or after a transmission error or arbitration loss the transmission is not active any more. If both TBF and TBE are set, then TBE wins.
6	TBF	rw-0	set TB slot to "Filled" 1 - slot selected by TBPTR shall be marked as "filled" 0 - no action TBF is automatically reset to 0 as soon as the slot is marked as filled and TSFF=1. If both TBF and TBE are set, then TBE wins.
5:0	TBPTR	rw-0x00	Pointer to a TB message slot. 0x00 - Pointer to the PTB others - Pointer to a slot in the STB The message slot pointed to by TBPTR is readable / writable using the TBUF registers. Write access is only possible if TSFF=0. Setting TBF to 1 marks the selected slot as filled and setting TBE to 1 marks the selected slot as empty. TBSEL and TSNEXT are unused in TTCAN mode and have no meaning. TBPTR can only point to buffer slots, that exist in the hardware. Unusable bits of TBPTR are fixed to 0. TBPTR is limited to the PTB and 63 STB slots. More slots cannot be used in TTCAN mode. If TBPTR is too big and points to a slot that is not available, then TBF and TBE are reset automatically and no action takes place.

Table 3-34 TTCAN: Time Trigger Configuration TTCFG (0xbf)

Bits	Name	Access	Function
0	TTEN	rw-0	Time Trigger Enable 1 - TTCAN enabled, timer is running 0 - disabled
2:1	T_PRESC	rw-0x0	TTCAN Timer PRESCaler 00b - 1 01b - 2 10b - 4 11b - 8 The TTCAN time base is a CAN bittime defined by S_PRES, S_SEG_1 and S_SEG_2. With T_PRESC an additional prescaling factor of 1, 2, 4 or 8 is defined. T_PRESC can only be modified if TTEN=0, but it is possible to modify T_PRESC and set TTEN simultaneously with one write access.
3	TTIF	rw-0	Time Trigger Interrupt Flag TTIF will be set if TTIE is set and the cycle time is equal to the trigger time TT_TRIG. Writing an one to TTIF resets it. Writing a zero has no impact. TTIF will be set only once. If TT_TRIG gets not updated, then TTIF will be not set again in the next basic cycle.
4	TTIE	rw-1	Time Trigger Interrupt Enable If TTIE is set, then TTIF will be set if the cycle time is equal to the trigger time TT_TRIG.
5	TEIF	rw-0	Trigger Error Interrupt Flag The conditions when TEIF will be set, are defined in chap. 4.4. There is no bit to enable or disable the handling of TEIF.
6	WTIF	rw-0	Watch Trigger Interrupt Flag WTIF will be set if the cycle count reaches the limited defined by TT_WTRIG and WTIE is set.
7	WTIE	rw-1	Watch Trigger Interrupt Enable

To reset an interrupt flag, the host controller needs to write an 1 to the flag. Writing a 0 has no effect. If a new interrupt event occurs while the write access is active then this event will set the flag and override the reset. This ensures that no interrupt event is lost.

Interrupt flags will only be set if the associated interrupt enable bit is set.

Table 3-35 TTCAN: Reference Message REF_MSG_0 to REF_MSG_3 (0xc0 to 0xc3)

Bits	Name	Access	Function
28:0	REF_ID	rw-0x0000	REFerence message IDentifier. If REF_IDE is 1 - REF_ID(28:0) is valid (extended ID) 0 - REF_ID(10:0) is valid (standard ID) REF_ID is used in TTCAN mode to detect a reference message. This holds for time slaves (reception) as well as for the time master (transmission). If the reference message is detected and there are no errors, then the Sync_Mark of this frame will become the Ref_Mark. REF_ID(2:0) is not tested and therefore the appropriate register bits are forced to 0. These bits are used for up to 8 potential time masters. CAN-CTRL recognizes the reference message only by ID. The payload is not tested. Additional note: A time master will transmit a reference message in the same way as a normal frame. REF_ID is intended for detection of a successful transmission of a reference message.
31	REF_IDE	rw-0	REFerence message IDE bit.

Table 3-5 gives a definition of the bit positions of REF_ID and REF_IDE inside REF_MSG_0 to REF_MSG_2.

A write access to REF_MSG_3 starts a data transfer of the reference message to the CAN clock domain (clock domain crossing). During this automatic transfer a write lock for all registers REF_MSG_0 to

REF_MSG_2 is active. The transfer takes up to 6 clocks in the CAN clock domain and up to 6 clocks in the host clock domain. The write access to REF_MSG_2 is necessary to make a new reference message active.

Table 3-36 TTCAN: Trigger Configuration TRIG_CFG_0 (0xc4)

Bits	Name	Access	Function
5:0	TTPTR	rw-0x00	Transmit Trigger TB slot Pointer If TTPTR is too big and points to a slot that is not available, then TEIF is set and no new trigger can be activated after a write access to TT_TRIG_1. If TTPTR points to an empty slot, then TEIF will be set at the moment, when the trigger time is reached.
7:6	-	r-00	reserved

Table 3-37 TTCAN: Trigger Configuration TRIG_CFG_1 (0xc5)

Bits	Name	Access	Function
2:0	TTYPE(2:0)	rw-0x0	Trigger Type 000b - Immediate Trigger for immediate transmission 001b - Time Trigger for receive triggers 010b - Single Shot Transmit Trigger for exclusive time windows 011b - Transmit Start Trigger for merged arbitrating time windows 100b - Transmit Stop Trigger for merged arbitrating time windows others - no action The time of the trigger is defined by TT_TRIG. TTPTR selects the TB slot for the transmit triggers. See chap. 4.4 for more details.
3	-	r-0	reserved
7:4	TEW(3:0)	rw-0x0	Transmit Enable Window For a single shot transmit trigger there is a time of up to 16 ticks of the cycle time where the frame is allowed to start. TWE+1 defines the number of ticks. See chap. 4.4.3 for further details. TEW=0 is a valid setting and shortens the transmit enable window to 1 tick.

Table 3-38 TTCAN: Trigger Time TT_TRIG_0 and TT_TRIG_1 (0xc6 and 0xc7)

Bits	Name	Access	Function
7:0	TT_TRIG	rw-0x00	Trigger Time TT_TRIG(15:0) defines the cycle time for a trigger. For a transmission trigger the earliest point of transmission of the SOF of the appropriate frame will be TT_TRIG+1.

A write access to TT_TRIG_1 starts a data transfer of the trigger definition to the CAN clock domain (clock domain crossing) and activates the trigger. If the trigger is active, then a write lock for all registers TRIG_CFG_0, TRIG_CFG_1, TT_TRIG_0 and TT_TRIG_1 is active. The write lock becomes inactive when the trigger time is reached (TTIF gets set is TTIE is enabled) or an error is detected (TEIF gets set).

Table 3-39 TTCAN: Watch Trigger Time TT_WTRIG_0 and TT_WTRIG_1 (0xc8 and 0xc9)

Bits	Name	Access	Function
7:0	TT_WTRIG	rw-0xff	Watch Trigger Time TT_WTRIG(15:0) defines the cycle time for a watch trigger. The initial watch trigger is the maximum cycle time 0xffff. See chap. 4.5 for more details.

A write access to TT_WTRIG_1 starts a data transfer of the trigger definition to the CAN clock domain (clock domain crossing). During this automatic transfer a write lock for the registers TT_WTRIG_0 and TT_WTRIG_1 is active. The transfer takes up to 6 clocks in the CAN clock domain and up to 6 clocks in the host clock domain. The write access to TT_WTRIG_1 is necessary to make a new trigger active.

Table 3-40 Receive Buffer Registers RBUF – Standard Format (r-0)

Address	Bit position								Function
	7	6	5	4	3	2	1	0	
RBUF	ID(7:0)								Identifier
RBUF+1	-					ID(10:8)			Identifier
RBUF+2	-								Identifier
RBUF+3	ESI	-							Identifier
RBUF+4	IDE=0	RTR	FDF	BRS	DLC(3:0)				Control
RBUF+5	KOER			TX	-				Status
RBUF+6	CYCLE_TIME(7:0)								TTCAN
RBUF+7	CYCLE_TIME(15:8)								TTCAN
RBUF+8	d1(7:0)								Data byte 1
RBUF+9	d2(7:0)								Data byte 2
.	...								.
RBUF+71	d64(7:0)								Data byte 64
RBUF+72	RTS(7:0)								CiA 603
.	...								.
RBUF+79	RTS(63:56)								CiA 603

Table 3-41 Receive Buffer Registers RBUF – Extended Format (r-0)

Address	Bit position								Function
	7	6	5	4	3	2	1	0	
RBUF	ID(7:0)								Identifier
RBUF+1	ID(15:8)								Identifier
RBUF+2	ID(23:16)								Identifier
RBUF+3	ESI	-		ID(28:24)					Identifier
RBUF+4	IDE=1	RTR	FDF	BRS	DLC(3:0)				Control
RBUF+5	KOER			TX	-				Status
RBUF+6	CYCLE_TIME(7:0)								TTCAN
RBUF+7	CYCLE_TIME(15:8)								TTCAN
RBUF+8	d1(7:0)								Data byte 1
RBUF+9	d2(7:0)								Data byte 2
.	...								.
RBUF+71	d64(7:0)								Data byte 64
RBUF+72	RTS(7:0)								CiA 603
.	...								.
RBUF+79	RTS(63:56)								CiA 603

The RBUF registers (0x00 to 0x4f) point the message slot with the oldest received message in the RB as can be seen in Figure 3-5. All RBUF registers can be read in any order.

KOER in RBUF has the same meaning as the bits KOER in register EALCAP. KOER in RBUF becomes meaningful if RBALL=1 (chap. 3.9.9.4).

Status bit TX in RBUF is set to 1 if the loop back mode (chap. 3.9.10.4) is activated and the core has received its own transmitted frame. This can be useful if LBME=1 and other nodes in the network do also transmissions.

The time-stamp CYCLE_TIME will be stored in RBUF only in TTCAN mode. This is the cycle time at the SOF of this frame. The cycle time of a reference message is always 0.

The Reception Time Stamps (RTS) for CiA 603 time-stamping are stored for each received message at the end of the RBUF address range. Therefore in contrast to TTS, RTS is related to the actual selected RBUF slot.

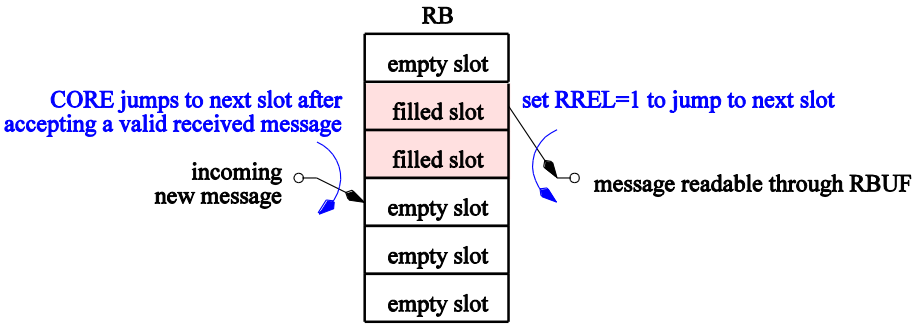

Figure 3-5 Schematic of the FIFO-like RB (example with 6 slots)

Table 3-42 Transmit Buffer Registers TBUF – Standard Format (rw-u)

Address	Bit position								Function
	7	6	5	4	3	2	1	0	
TBUF	ID(7:0)								Identifier
TBUF+1	-					ID(10:8)			Identifier
TBUF+2	-								Identifier
TBUF+3	TTSEN	-							Identifier
TBUF+4	IDE=0	RTR	FDF	BRS	DLC(3:0)				Control
TBUF+8	d1(7:0)								Data byte 1
TBUF+9	d2(7:0)								Data byte 2
.	...								.
TBUF+71	d64(7:0)								Data byte 64

Table 3-43 Transmit Buffer Registers TBUF – Extended Format (rw-u)

Address	Bit position								Function
	7	6	5	4	3	2	1	0	
TBUF	ID(7:0)								Identifier
TBUF+1	ID(15:8)								Identifier
TBUF+2	ID(23:16)								Identifier
TBUF+3	TTSEN	-		ID(28:24)					Identifier
TBUF+4	IDE=1	RTR	FDF	BRS	DLC(3:0)				Control
TBUF+8	d1(7:0)								Data byte 1
TBUF+9	d2(7:0)								Data byte 2
.	...								.
TBUF+71	d64(7:0)								Data byte 64

The TBUF registers (0x50 to 0x97) point the next empty message slot in the STB if TBSEL=1 or to the PTB otherwise. Seen in Figure 3-6 for more details. All TBUF registers can be written in any order. For the STB it is necessary to set TSNEXT to mark a slot filled and to jump to the next message slot.

Please mind the gap inside the addressing range of TBUF from TBUF+5 to TBUF+7. This is for better address segment alignment. The memory cells in the gap can be read and written, but have no meaning for the CAN protocol.

TBUF is build using a true 32 bit wide memory and therefore a write access needs to be performed as 32 bit write. See chap. 7.1.3 for further details.

Both RBUF and TBUF include some frame-individual control bits (Table 3-44). For RBUF these bits signal the status of the appropriate CAN control field bits of the received CAN frame while for TBUF these bits select the appropriate CAN control field bit for the frame that has to be transmitted.

In contrast to RTS, which is stored for every received frame, TTS is stored only for the last transmitted frame if TTSEN=1. TTS is not related to the actual selected TBUF slot.

Table 3-44 Control bits in RBUF and TBUF

Bit	Description
IDE	IDentifier Extension 0 – Standard Format: ID(10:0) 1 – Extended Format: ID(28:0)
RTR	Remote Transmission Request 0 – data frame 1 – remote frame Only CAN 2.0 frames can be remote frames. There is no remote frame for CAN FD. Therefore RTR is forced to 0 if FDF =1 in TBUF and RBUF. If a CAN FD frame is received with bit RRS=1, then this is ignored, a data payload is expected for reception instead and RTR in RBUF is overridden but the CRC of the frame is calculated with RRS=1.
FDF	CAN FD frame 0 – CAN 2.0 frame (up to 8 bytes payload) 1 – CAN FD frame (up to 64 bytes payload)
BRS	Bit Rate Switch 0 – nominal / slow bit rate for the complete frame 1 – switch to data / fast bit rate for the data payload and the CRC Only CAN FD frames can switch the bitrate. Therefore BRS is forced to 0 if FDF =0.
ESI	Error State Indicator This is a read-only status bit for RBUF and is not available in TBUF. The protocol machine automatically embeds the correct value of ESI into transmitted frames. ESI is only included in CAN FD frames and does not exist in CAN 2.0 frames. 0 – CAN node is error active 1 – CAN node is error passive ESI in RBUF is always low for CAN 2.0 frames. The error state for transmission is shown with bit EPASS in register ERRINT.
TTSEN	Transmit Time-Stamp ENable For CiA 603 time-stamping the acquisition of a transmit time stamp TTS can be selected in the TBUF: 0 – no acquisition of a transmit time stamp for this frame 1 – TTS update enabled

The Data Length Code (DLC) in RBUF and TBUF defines the length of the payload – the number of payload bytes in a frame. See Table 3-45 for further details.

Remote frames (only for CAN 2.0 frames where FDF =0) are always transmitted with 0 payload bytes, but the content of the DLC is transmitted in the frame header. Therefore it is possible to code some information into the DLC bits for remote frames. But then care needs to be taken if different CAN nodes are allowed to transmit a remote frame with the same ID. In this case all transmitters need to use the same DLC because otherwise this would result in an unresolvable collision.

Table 3-45 Definition of the DLC (according to the CAN 2.0 / FD specification)

DLC (binary)	Frame Type	Payload in Bytes
0000 to 1000	CAN 2.0 and CAN FD	0 to 8
1001 to 1111	CAN 2.0	8
1001	CAN FD	12
1010	CAN FD	16
1011	CAN FD	20
1100	CAN FD	24
1101	CAN FD	32
1110	CAN FD	48
1111	CAN FD	64

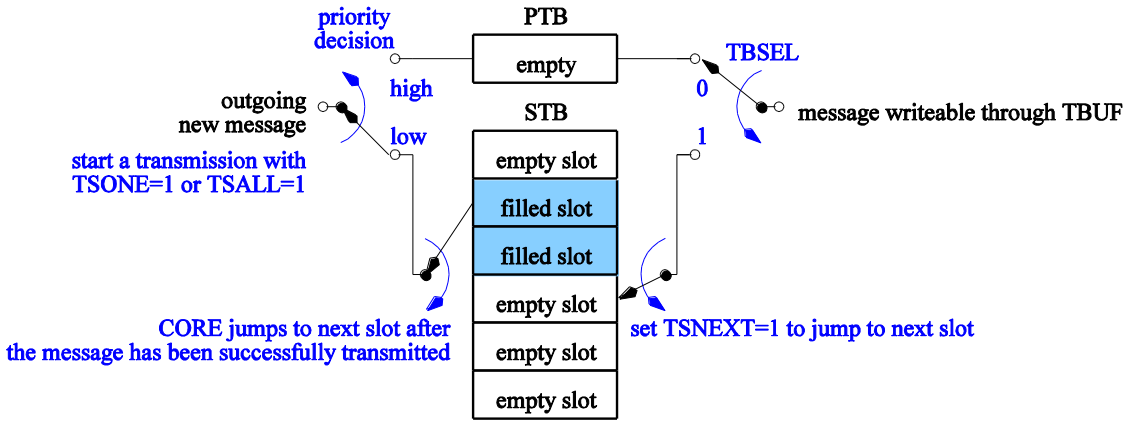

Figure 3-6 Schematic of PTB and STB in FIFO mode (empty PTB and 6 STB slots)

The TBUF registers and readable and writable. Therefore a host controller may use TBUF to successively prepare a message bit-by-bit if necessary.

3.8 Interrupt flags

CAN-CTRL provides several interrupt flags. All of them are set if the appropriate event occurs and the related interrupt-enable bit is active (if a related interrupt-enable bit exists). To reset an interrupt flag, the host controller needs to write an 1 to the flag.

The top-level output <host_irq> gets set if at least one of the interrupt flags is set. This signal is intended to be used by a host controller as single interrupt request.

Additionally to <host_irq> CAN-CTRL provides the top level output <host_if(13:0)> which is a vector containing each individual interrupt flag. These individual flags are intended to be used by a host system for individual interrupt processing. Table 3-46 shows the mapping of the interrupt flags to <host_if(13:0)>.

Table 3-46 Top level output signal <host_if(13:0)>

Bit position and Interrupt Flags							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF
-	**-**	**13**	**12**	**11**	**10**	**9**	**8**
-	-	WTIF	TEIF	TTIF	EPIF	ALIF	BEIF

3.9 General Operation

This chapter describes handling of CAN communication. Before communication is possible, the CAN-CTRL core has to be configured to match the CAN bus timings. A detailed description of the CAN bus bit timing is given in chap. 3.10.

3.9.1 Acceptance Filters

To reduce the load of received frames for the host controller, the core uses acceptance filters. The CAN-CTRL core checks the message identifier during acceptance filtering. Therefore the length of each acceptance filter is 29 bits.

If a message passes one of the filters then it will be accepted. If accepted, the message will be stored into the RB and finally RIF is set if RIE is enabled. If the message is not accepted, RIF is not set and the RB FIFO pointer is not increased. Messages that are not accepted will be discarded and overwritten by the next message. No stored valid message will be overwritten by any not accepted message.

Independently of the result of acceptance filtering, the CAN-CTRL core checks every message on the bus and sends an acknowledge or an error frame to the bus.

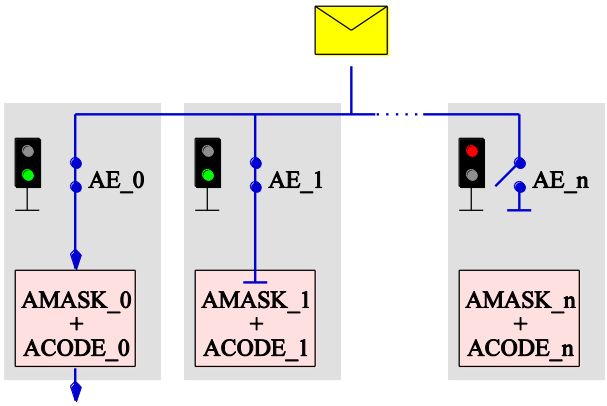

Figure 3-7 Example of acceptance filtering

The acceptance mask defines which bits shall be compared while the acceptance code defines the appropriate values. Setting mask bits to 0 enables the comparison of the selected acceptance code bits with the corresponding message identifier bits. Mask bits that are set to 1 are disabled for the acceptance check and this results in accepting the message.

The identifier bits will be compared with the corresponding acceptance code bits ACODE as follows:

- standard: ID(10:0) with ACODE(10:0)
- extended: ID(28:0) with ACODE(28:0)

Example: If AMASK_x(0)=0 and all other AMASK_x bits are 1 then the value of the last ID bit has to be equal to ACODE(0) for an accepted message. All other ID bits are ignored by the filter.

Figure 3-7 gives an example of acceptance filtering using several filters. In this example the filter 0 and 1 are enabled by the appropriate AE_x bits in the ACF_EN_x registers. All other filters are disabled and therefore do not accept any message. For the enabled filters the combination of AMASX_x and ACODE_x defines if a message is accepted (like in the example for filter 0) or not accepted (like in the example for filter 1).

Note: Disabling a filter by setting AE_x=0 blocks messages. In contrast to this disabling a mask bit in AMASK_x disables the check for this bit which results in accepting messages.

The definitions of AMASK and ACODE alone do not distinguish between standard or extended frames. If bit AIDEE=1 then the value of AIDE defines with frame type is accepted. Otherwise if AIDE=0 both types are accepted.

After power-on reset the CAN-CTRL core is configured to accept all messages. (Filter 0 is enabled by AE_0=1, all bits in AMASK_0 are set to 1 and AIDEE=0. All other filters are disabled. Filter 0 is the only filter that has defined reset values for AMASK / ACODE while all other filters have undefined reset values.)

3.9.2 Message Reception

The received data will be stored in the RB as shown in Figure 3-8. The RB is configurable by a pre-synthesis parameter and has FIFO-like behavior. Every received message that is valid and accepted sets RIF=1 if RIE is enabled. RSTAT is set depending of the fill state. When the number of filled buffers is equal to the programmable value AFWL then RAFIF is set if RAFIE is enabled. In case, when all buffers are full, the RFIF is set if RFIE is enabled.

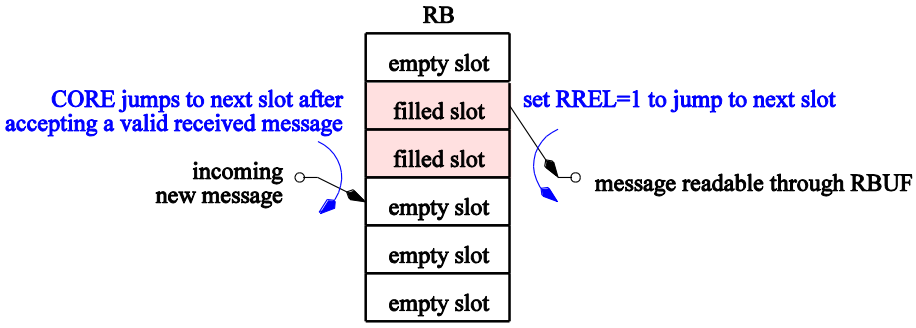

Figure 3-8 Schematic of the FIFO-like RB (example with 6 slots)

The RB always maps the message slot containing the oldest message to the RBUF registers.

The maximum payload length for CAN 2.0 messages is 8 bytes and for CAN FD messages 64 bytes. The individual length of each message is defined by the DLC. Because of this the RB provides slots for each message and the host controller is required to set RREL to jump to the next RB slot. All RBUF bytes of the actual slot can be read in any order.

If the RB is full, the next incoming message will be stored temporarily until it passes for valid (6^{th} EOF bit). Then if ROM=0 the oldest message will be overwritten by the newest or if ROM=1 the newest message will be discarded. In both cases ROIF is set if ROIE is enabled. If the host controller reads the oldest message and sets RREL before a new incoming message becomes valid then no message will be lost.

3.9.3 Handling message receptions

Without acceptance filtering the CAN-CTRL core would signal the reception of every frame and the host would be required to decide if it was addressed. This would result in quite a big load on the host controller.

It is possible to disable interrupts and use the acceptance filters to reduce the load for the host controller. For a basic operation RIF is set to 1 if RIE is enabled and the CAN-CTRL core has received a valid message. To reduce the number of reception interrupts it is possible to use RAIE / RAIF (RB Almost full Interrupt) or RFIE / RFIF (RB Full Interrupt) instead of RIE / RIF (Reception Interrupt). The “almost full limit” is programmable using AFWL.

The RB contains a number of RB slots which is selectable before synthesis using a generic parameter. Reading the RB shall be done as follows:

1. Read the oldest message from the RB FIFO using the RBUF registers.
2. Release the RB slot with RREL=1. This selects the next message (the next FIFO slot). RBUF will be updated automatically.

3. Repeat these actions until RSTAT signals an empty RB.

If the RB FIFO is full and a new received message is recognized as valid (6th EOF bit) then one message will be lost (see bit ROM). Before this event, no message is lost. This should give enough time for the host controller to read at least one frame from the RB after the RB FIFO has been filled and the selected interrupt has occurred. To enable this behavior the RB includes one more (hidden) slot than specified by the synthesis parameter RBUF_SLOTS. This hidden slot is used to receive a message, validate it and check it if it matches the acceptance filters before an overflow occurs.

3.9.4 Message Transmission

Before starting any transmission, at least one of the transmit buffers (PTB or STB) has to be loaded with a message (Figure 3-9). TPE signals if the PTB is locked and TSSTAT signals the fill state of the STB. The TBUF registers provide access to both the PTB as well as to the STB. Below is the recommended programming flow:

1. Set TBSEL to the desired value to select either the PTB or the STB.
2. Write the frame to the TBUF registers.
3. For the STB set TSNEXT=1 to finish loading of this STB slot.

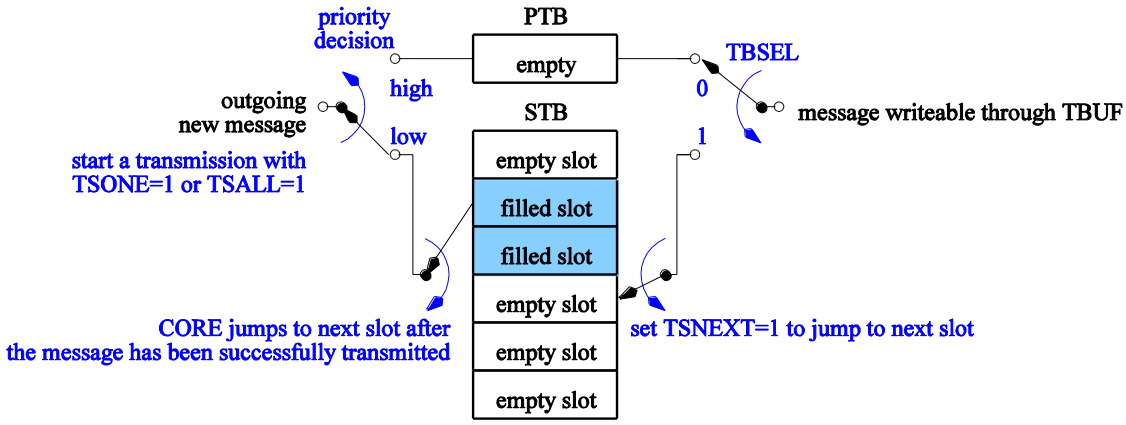

Figure 3-9 Schematic of PTB and STB in FIFO mode (empty PTB and 6 STB slots)

The maximum payload length for CAN 2.0 messages is 8 bytes and for CAN FD messages 64 bytes. The individual length of each message is defined by the DLC. For remote frames (bit RTR) the DLC becomes meaningless, because remote frames always have a data length of 0 bytes. The host controller is required to set TSNEXT to jump to the next STB slot. All TBUF bytes can be written in any order.

Setting TSNEXT=1 is meaningless if TBSEL=0 selects the PTB. In this case TSNEXT is automatically cleared and does no harm.

Bit TPE should be set to start a transmission when using the PTB. To use the STB, TSONE has to be set to start a transmission of a single message or TSALL to transmit all messages.

The PTB has always a higher priority than the STB. If both transmit buffers have the order to transmit, the PTB message will be always sent first regardless of the frame identifiers. If a transmission from the STB is already active, it will be completed before the message from the PTB is sent at the next possible transmit position (the next interframe slot). After the PTB transmission is completed or aborted, the CAN-CTRL core returns to process other pending messages from the STB. See also chap. 2.3.3 for further details.

When the transmission is completed, the following transmission interrupts are set:

- For the PTB, TPIF is set if TPIE is enabled
- For the STB using TSONE, TSIF is set if one message has been completed and TSIE is enabled.
- For the STB using TSALL, TSIF is set if all messages have been completed and if TSIE is enabled. In other words: TSIE is set if the STB is empty. Therefore, if the host controller writes

an additional message to the STB after a TSALL transmission has been started then the additional message will be also transmitted before TSIF will be set.

It is meaningless to set TSONE or TSALL while the STB is empty. In such a case TSONE respectively TSALL will be reset automatically. No interrupt flag will be set and no frame will be transmitted.

3.9.5 Message transmission abort

If the situation arises, where a message in a transmit buffer cannot be sent due to its low priority, this would block the buffer for a long time. In order to avoid this, the host controller can withdraw the transmission request by setting TPA or TSA respectively, if the transmission has not yet been started.

Both TPA and TSA source a single interrupt flag: AIF. The CAN protocol machine executes an abort only if it does not transmit anything to the CAN bus. Therefore the following is valid:

- There is no abort during bus arbitration.
    - If the node loses arbitration, the abort will be executed afterwards.
    - If the node wins arbitration, the frame will be transmitted.
- There is no abort while a frame is transmitted.
    - If a frame is transmitted successfully then a successful transmission is signaled to the host controller. In this case no abort is signaled. This is done by the appropriate interrupt and status bits.
    - After an unsuccessful transmission where the CAN node does not receive an acknowledge, the error counter is incremented and the abort will be executed.
    - If there is at least one frame left in the STB, while the host has commanded all frames to be transmitted (TSALL=1), then both the completed frame as well as the abort is signaled to the host.

Because of these facts aborting a transmission may take some time depending on the CAN communication speed and frame length. If an abort is executed, this results in the following actions:

- TPA releases the PTB which results in TPE=0.
  The frame data is still stored in the PTB after releasing the PTB.
- TSA releases one single message slot or all message slots of the STB. This depends on whether TSONE or TSALL was used to start the transmission. TSSTAT will be updated accordingly. Releasing a frame in the STB results in discarding the frame data because the host cannot access it.

Setting both TPA and TSA simultaneously is not recommended. If a host controller decides to do it anyway, then AIF will be set and both transmissions from PTB and STB will be aborted if possible. As already stated if one transmission will be completed before the abort can be executed this will result in signaling a successful transmission. Therefore the following interrupt flags may be set if enabled:

- AIF (once for both PTB and STB transmission abort)
- TPIF + AIF
- TSIF + AIF
- TPIF + TSIF (very seldom, will only happen if the host does not handle TPIF immediately)
- TPIF + TSIF + AIF (very seldom, will only happen if the host does not handle TPIF and TSIF immediately)

To clear the entire STB both TSALL and TSA need to be set. In order to detect if a message cannot be sent for a long time because it loses arbitration the host may use the ALIF / ALIE.

3.9.6 A Full STB

After writing a message to the STB, TSNEXT=1 marks a buffer slot filled and jumps to the next free message slot. TSNEXT is automatically reset to 0 by the CAN-CTRL core after this operation.

If the last message slot has been filled and therefore all message slots are occupied then TSNEXT stays set until a new message slot becomes free. While TSNEXT=1, then writing to TBUF is blocked by the CAN-CTRL core.

When a slot becomes free, then the CAN-CTRL core automatically resets TSNEXT to 0. A slot becomes free if a frame form the STB is transmitted successfully or if the host requests an abort (TSA=1). If a TSALL transmission is aborted, then TSNEXT is also reset, but additionally the complete STB is marked as empty.

3.9.7 Error Handling

On one hand CAN-CTRL does automatic error handling which means that in most cases the host controller does not care about errors. This includes automatic message retransmission and automatic deletion of received messages with errors. On the other hand if required by the host, CAN-CTRL may optionally give detailed information about errors and signal every error to the host by interrupt. This enables to run an application like a CAN bus monitor at the host.

Every CAN node has 3 states of error handling:

1. Error-Active: The node automatically transmits active error frames upon detection of an error.
2. Error-Passive: The node transmits passive error frame upon detection of an error. This means that it does not transmit a dominant value to the bus,,but expects an error frame.
3. Bus Off: After too many errors a node goes “bus off” where it stops touching the bus.

To handle the 3 error states every CAN node has two error counters: the transmit and the receive error counter. Both are incremented and decremented as defined by the CAN specification and the node enters the appropriate error state upon reaching a counter level defined by the CAN specification.

The error counters are incremented if there are errors. Dangerous errors that are probably caused by this node will result in incrementing the counter by 8, errors that are probably caused by other nodes will result in incrementing the counter by 1. Valid frame transmission or valid receptions result in decrementing the counters. All this is defined by the CAN specification and handled automatically by CAN-CTRL.

An error frame (better: an error flag) is different to a data frame. It is a dominant pulse for at least 6 consecutive dominant bits, which is a stuff bit violation for other nodes. A host controller cannot command to transmit an error frame. It is done fully automatically by CAN-CTRL.

If CAN-CTRL is commanded to transmit a frame then it automatically tries retransmissions as fast as possible until the frame gets transmitted without error or the node goes bus off. If a frame is received and CAN-CTRL detects an error, then the received data is discarded. Because of the automatically transmitted error frame the sender of the frame will retransmit the frame. Frames in RBUF are never overwritten by frames with errors. Only valid received frames may result in a RBUF overflow.

3.9.8 The Bus Off State

The “bus off” state is signaled using the status bit BUSOFF in register CFG_STAT (Table 3-7). A CAN node enters the “bus off” state automatically if its transmit error counter becomes >255. Then it will not take part in further communications until it returns into the error-active state again. Setting BUSOFF to 1 also sets the EIF interrupt if EIE is enabled. A CAN node returns to error-active state if it is reset by a power-on reset or if it receives 128 sequences of 11 recessive bits (recovery sequences). Please note that between each recovery sequence the bus may have dominant state.

In the “bus off” state, RECNT and TECNT stay unchanged. Please note that while entering bus off state TECNT rolls over and therefore may hold a small value. Therefore it is recommended to use TECNT before the node enters bus off state and status bit BUSOFF afterwards.

If the node recovers from “bus off” state then RECNT and TECNT are automatically reset to 0.

If a frame is pending for transmission but the CAN node has entered bus off state, then this frame is kept pending. If a node returns to error-active state after bus off, then the transmission of the pending frame will be tried. If this is not desired, then the frame should be aborted by the host controller.

3.9.9 Extended Status and Error Report

During CAN bus communication transmission errors may occur. The following features support detection and analysis of them. This can be used for extended bus monitoring.

3.9.9.1. Programmable Error Warning Limit

Errors during reception / transmission are counted by RECNT and TECNT. A programmable error warning limit EWL, located in register LIMIT, can be used by the host controller for flexible reaction on those events. The limit values can be chosen in steps of 8 errors from 8 to 128:

Limit of count of errors = (EWL+1)*8

The interrupt EIF will be set if enabled by EIE under the following conditions:

- the border of the error warning limit has been crossed in either direction by RECNT or TECNT or
- the BUSOFF bit has been changed in either direction.

3.9.9.2. Arbitration Lost Capture (ALC)

The core is able to detect the exact bit position in the Arbitration Field where the arbitration has been lost. This event can be signaled by the ALIF interrupt if it is enabled. The value of ALC stays unchanged if the node is able to win the arbitration. Then ALC holds the old value of the last loss of arbitration.

The value of ALC is defined as follows: A frame starts with the SOF bit and then the first bit of the ID is transmitted. This first ID bit has ALC value 0, the second ID bit ALC value 1 and so on. See Figure 2-3 for the bit order in all types of CAN 2.0 and CAN FD frames.

Arbitration is only allowed in the arbitration field (Figure 2-3). Therefore the maximum value of ALC is 31 which is the RTR bit in extended frames.

Additional hint: If a standard remote frame arbitrates with an extended frame, then the extended frame loses arbitration at the IDE bit and ALC will be 12. The node transmitting the standard remote frame will not notice that an arbitration has been taken place, because this node has won.

It is impossible to get an arbitration loss outside of the arbitration field. Such an event is a bit error.

3.9.9.3. Kind Of Error (KOER)

The core recognizes errors on the CAN bus and stores the last error event in the KOER bits. A CAN bus error can be signaled by the BEIF interrupt if it is enabled. Every new error event overwrites the previous stored value of KOER. Therefore the host controller has to react quickly on the error event. The error codes are described in the table 3-15.

KOER is updated with each new error. Therefore it stays untouched when frames are successfully transmitted or received. This opens the opportunity for a delayed error investigation.

3.9.9.4. Reception of All Data Frames (RBALL)

If RBALL=1 then all received data frames are stored even those with error. This holds also for loop back mode (chap. 3.9.10.4). Only data frames are stored in RBUF. Error frames or overload frames are not stored.

If CiA 603 time-stamping (chap. 5) is enabled (TIMEEN=1) and the timestamp is configured for the EOF (TIMEPOS=1) then in the case of an error the time-stamp is acquired at the start of the error frame.

Most of the possible errors can only occur if the node is the transmitter of the frame. In this case a frame would only be stored in RBUF if a loop back mode is activated. Depending on the type of error parts of the frame stored in an RBUF slot may be valid while other parts are unknown. Table 3-47 lists the possibilities.

Table 3-47 RBALL and KOER

KOER	Condition	Description
no error	All	Successful reception.
BIT ERROR	Receiver	Can only occur in the ACK slot. All stored data are valid.

KOER	Condition	Description
	Transmitter	The payload is always invalid. The header bits including the ID may be valid. In the arbitration phase detection of a wrong bit is part of the arbitration and therefore not a BIT ERROR. But if a stuff bit in the arbitration phase is detected wrong by the transmitter of the frame, then this is a BIT ERROR and in this case the header bits are invalid. Therefore the header bits cannot be trusted, but if the header matches to an expected header of a frame, it can be used for further investigation.
FORM ERROR	All	Only FORM ERRORs in a data frame are covered. This includes errors in the CRC delimiter, the ACK delimiter or the EOF bits. All stored data are valid.
STUFF ERROR	Receiver	The position of a STUFF ERROR is unknown. All stored data are invalid.
	Transmitter	Can only happen during arbitration. All stored data are invalid.
ACK ERROR	Receiver	Can only occur if the node is in LOM. All stored data are valid.
	Transmitter	Can only occur in loop back mode without self-ACK. All stored data are valid.
CRC ERROR	Receiver	All stored data are invalid.

Please note that even if Table 3-47 states that (parts of) the data are valid under certain conditions, then care must be taken. There is no guarantee for this and it depends on the reason for the error if this statement is correct. Example: if the node has a bad bit timing configuration then there is no correct synchronization and therefore no valid data. Therefore the statement that (parts of) the data is valid signals only the possibility that it is so.

3.9.10 Extended Features

3.9.10.1. Single Shot Transmission

Sometimes an automatic re-transmission is not desired. Therefore the order to transmit a message only once can be set separately for the transmit buffers PTB by the bit TPSS and for the transmit buffer STB by the bit TSSS. In this case no re-transmission will be performed in the event of an error or arbitration lost if the selected transmission is active.

In the case of an immediate successful transmission there is no difference to normal transmission. But in the case of an unsuccessful transmission the following will happen:

- TPIF gets set if enabled, the appropriate transmit buffer slot gets cleared
- In case of an error, KOER and the error counters get updated. BEIF gets set if BEIE is enabled and the other error interrupt flags will act accordingly.
- In case of a lost arbitration, ALIF gets set if ALIE is enable.

Therefore for single shot transmission TPIF alone does not indicate if the frame has been successfully transmitted. Therefore single shot transmission should only be used together with BEIF and ALIF.

If single shot transmission is used with TSALL and there is more than one frame in the STB then for each frame a single shot transmission is done. Regardless if any of the frames is not successfully transmitted (e.g. because of an ACK error) the CAN-CTRL advances to the next frame and stops if the STB is empty. Therefore in this scenario only the error counters indicate what has happened. This can be quite complex to evaluate because if one of two frames got errors the host cannot detect which one was the successful one.

If the bus is occupied by another frame, if a single shot transmission is started, then CAN-CTRL waits till the bus is idle and then tries to transmit the single shot frame.

3.9.10.2. Listen Only Mode (LOM)

LOM provides the ability of monitoring the CAN bus without any influence to the bus. LOM by CAN-CTRL is compatible to the bus monitoring feature defined in the CAN FD specification.

Another application is the automatic bit rate detection where the host controller tries different timing settings until no errors occur.

Errors will be monitored (KOER, BEIF) in LOM.

In LOM the core is not able to write dominant bits onto the bus (no active error flags or overload flags, no acknowledge). This is done using the following rules:

- In LOM the protocol machine acts as if in error passive mode where only recessive error flags are generated. Only the protocol machine acts as if in error passive mode. All other components including the status registers are not touched.
- In LOM the protocol machine does not generate a dominant ACK.
- The error counters stay unchanged regardless of any error condition.

Important facts regarding ACKs for LOM:

- If a frame is transmitted by a node then an ACK visible at the bus will be only generated if at least one additional node is attached to the bus that is not in LOM. Then there will be no error and all nodes (also those in LOM) will receive the frame.
- If there is an active or passive error flag after an ACK error, then the node in LOM is able to detect this as ACK error.

Activation of LOM should not be done while a transmission is active. The host controller has to take care of this. There is not additional protection from CAN-CTRL.

If LOM is enabled then no transmission can be started.

The loop back mode (external) LBME (chap. 3.9.10.4) plays an important role for the behavior of LOM. If LBME is disabled, then LOM acts as described above and the node cannot write any dominant bit to the bus. But if LBME is activated, then the node is allowed to transmit a frame. LBME allows only the transmission of an frame including the optional self-ACK (SACK), but the node will not respond with an ACK to frames from other nodes and will not generate error or overload frames. In Summary the combination of LOM and LBME is “a silent receiver, that is able to transmit if necessary”.

3.9.10.3. Bus Connection Test

To check if a node is connected to the bus the following steps shall be done:

- Transmit a frame. If the node is connected to the bus then its TX bits are visible on its RX input.
- If there are other nodes connected to the CAN bus, then a successful transmission (including an acknowledge from the other nodes) is expected. No error will be signaled.
- If the node is the only node that is connected to the CAN bus (but the connection between the bus, the transceiver and the CAN-CTRL core is fine) then the first regular error situation occurs in the ACK slot because of no acknowledge from other nodes. Then a BEIF error interrupt will be generated if enabled and KOER=“100” (ACK error).
- If the connection to the transceiver or the bus is broken then immediately after the SOF bit the BEIF error interrupt will be set and KOER=“001” (BIT error).

3.9.10.4. Loop Back Mode (LBMI and LBME)

CAN-CTRL supports two Loop Back Modes: internal (LBMI) and external (LBME). Both modes result in reception of the own transmitted frame which can be useful for self-tests. See Figure 3-10 for details.

In LBMI CAN-CTRL is disconnected from the CAN bus and the txd output is set to recessive. The output data stream is internally fed back to the input. In LBMI the node generates a self-ACK to avoid an ACK error.

In LBME CAN-CTRL stays connected to the transceiver and a transmitted frame will be visible on the bus. With the help of the transceiver CAN-CTRL receives its own frame. In LBME the node does not generate a self-ACK when SACK=0, but generates a self-ACK when SACK=1. Therefore in LBME with SACK=0 there are two possible results upon a frame transmission:

1. Another node receives the frame too and generates an ACK. This will result in a successful transmission and reception.
2. No other node is connected to the bus and this results in an ACK error. To avoid retransmissions and incrementing the error counters it is recommended to use TPSS or TSSS if it is unknown if other nodes are connected.

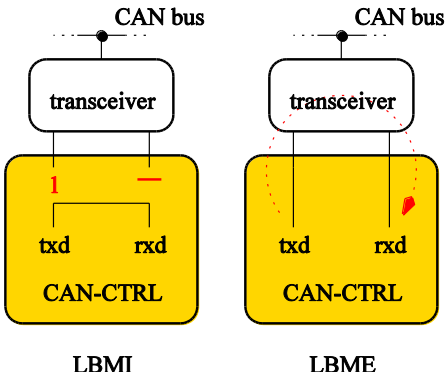

Figure 3-10 Loop Back Mode: Internal and External

In Loop Back Mode the core receives its own message, stores it in the RBUF and sets the appropriate receive and transmit interrupt flags if enabled.

LBMI can be useful for chip-internal and software tests while LBME can test the transceiver and the connections to it.

Activation of LBMI or LBME should not be done while a transmission is active. The host controller has to take care of this. There is not additional protection from CAN-CTRL.

If the node is connected to a CAN bus then switching back from LBMI to normal operation must not be done by simply setting LBMI to 0, because then it may be that this occurs just at the moment while another CAN node is transmitting. In this case switching back to normal operation shall be done by setting bit RESET to 1. This automatically clears LBMI to 0. Finally RESET can be disabled and the core returns back to normal operation. In contrast to this LBME can be disabled every time.

LBME can be used in combination with LOM. Details are given in chap. 3.9.10.2.

3.9.10.5. Transceiver Standby Mode

Using the register bit STBY the output signal stby is driven. It can be used to activate a standby mode for a transceiver. The behavior is compatible to the NXP TJA1049 transceiver.

Once standby is activated no transmission is possible and therefore TPE, TSONE and TSALL cannot be set. On the other hand CAN-CTRL does not allow STBY to be set if a transmission is already active (TPE, TSONE or TSALL is set).

If STBY is set the transceiver enters a low-power mode. In this mode it is unable to receive a frame at full speed but monitors the CAN bus for a dominant state. If the dominant state is active for a time which is defined in the transceivers data sheet the transceiver will pull the rxd signal low. If rxd is low CAN-CTRL automatically clears STBY to 0 which disables standby mode for the transceiver. This is done silently without an interrupt to the host controller.

Switching from standby mode to active mode takes some time for the transceiver and therefore the initial wakeup frame cannot be received successfully. Therefore the node recently being in standby will not respond with an ACK. If no CAN node at the bus responds to the wakeup frame with an ACK then this results in an ACK error for the transmitter of the wakeup frame. Then the transmitter will automatically repeat the frame. During the repetition the transceiver will be back in active mode and CAN-CTRL will receive the frame and will respond with an ACK.

In summary: One node transmits a frame for wakeup. If all others nodes are in standby mode, then the transmitter gets an ACK error and will automatically repeat the frame. During the repetition the nodes are back in active mode and will respond with an ACK.

STBY is not affected by bit RESET.

3.9.10.6. Error Counter Reset

According to the CAN / CAN FD standard RECNT counts receive errors and TECNT counts transmit errors. After too many transmit errors a CAN node has to go to bus off state (chap. 3.9.7). Bit RESET does not modify the errors counters or the bus off state. The CAN / CAN FD specification defines rules how to disable bus off state and to decrease the error counters. A good node will recover from all of this

automatically if only a temporary error has caused the problems. The classic CAN 2.0B specification requires this automatic behavior without host controller interaction to avoid the babbling idiot problem.

The CAN FD specification relaxes this and allows the host controller to override the automatic error counter handling. But this should be used with extra care and is suggested to use only for debugging purposes.

Writing a 1 to the bit BUSOFF resets the error counters and therefore forces the node to leave the bus off state. This is done without setting EIF.

3.9.11 Software Reset

If bit RESET in CFG_STAT is set to 1 then the software reset is active. Several components are forced to a reset state if RESET=1 but no all components are touched by RESET. Some components are only sensitive to a hardware reset. The reset values of all bits are always the same for software and hardware reset.

Table 3-48 Software Reset

Register	RESET	Comment
ACFADR	No	-
ACODE_x	No	Register is writeable if RESET=1 and write-locked if 0.
AE_x	No	-
AFWL	No	-
AIF	Yes	-
ALC	Yes	-
ALIE	No	-
ALIF	Yes	-
AMASK_x	No	Register is writeable if RESET=1 and write-locked if 0.
BEIE	No	-
BEIF	Yes	-
BUSOFF	(No)	An error counter reset by setting BUSOFF=1 also resets BUSOFF.
EIE	No	-
EIF	No	-
EPASS	No	-
EPIE	No	-
EPIF	Yes	-
EWARN	No	-
EWL	Yes	-
FD_ISO	No	Register is writeable if RESET=1 and write-locked if 0.
F_PRESC	No	Register is writeable if RESET=1 and write-locked if 0.
F_Seg_1	No	Register is writeable if RESET=1 and write-locked if 0.
F_Seg_2	No	Register is writeable if RESET=1 and write-locked if 0.
F_SJW	No	Register is writeable if RESET=1 and write-locked if 0.
KOER	Yes	-
LBME	Yes	-
LBMI	Yes	-
LOM	No	-
RACTIVE	Yes	Reception is immediately cancelled even if a reception is active. No ACK will be generated.
RAFIE	No	-
RAFIF	Yes	-
RBALL	Yes	-
RBUF	(Yes)	All RB slots are marked as empty. RBUF contains unknown data.
RECNT	No	An error counter reset is possible by setting BUSOFF=1.
REF_ID	No	-

Register	RESET	Comment
REF_IDE	No	-
RFIE	No	-
RFIF	Yes	-
RIE	No	-
RIF	Yes	-
ROIE	No	-
ROIF	Yes	-
ROM	No	-
ROV	Yes	All RB slots are marked as empty.
RREL	Yes	-
RSTAT	Yes	All RB slots are marked as empty.
SACK	Yes	-
SELMASK	No	-
STBY	No	-
S_PRESC	No	Register is writeable if RESET=1 and write-locked if 0.
S_Seg_1	No	Register is writeable if RESET=1 and write-locked if 0.
S_Seg_2	No	Register is writeable if RESET=1 and write-locked if 0.
S_SJW	No	Register is writeable if RESET=1 and write-locked if 0.
TACTIVE	Yes	All transmissions are immediately terminated with RESET. If a transmission is active this will result in an erogenous frame. Other nodes will generate error frames.
TBE	Yes	-
TBF	Yes	-
TBPTR	No	-
TBSEL	Yes	TBUF fixed to point to PTB.
TBUF	(Yes)	All STB slots are marked as empty. Because of TBSEL TBUF points to the PTB.
TECNT	No	An error counter reset is possible by setting BUSOFF=1.
TEIF	Yes	-
TIMEEN	No	-
TIMEPOS	No	-
TPA	Yes	-
TPE	Yes	-
TSA	Yes	-
TSALL	Yes	-
TSMODE	No	-
TSNEXT	Yes	-
TSONE	Yes	-
TPIE	No	-
TPIF	Yes	-
TPSS	Yes	-
TSFF	Yes	All STB slots are marked as empty.
TSIE	No	-
TSIF	Yes	-
TSSS	Yes	-
TSSTAT	Yes	All STB slots are marked as empty.
TTEN	Yes	-
TTIF	Yes	-
TTIE	No	-
TTPTR	No	-
TTS	No	-
TTTBM	No	-

Register	RESET	Comment
TTYPE	No	-
TT_TRIG	No	-
TT_WTRIG	No	-
T_PRESC	No	-
WTIE	No	-
WTIF	Yes	-

3.10 Pseudo Dual Port Memories

Generic parameter RAM_TYPE selects between the following options:

- one true dual port memory block for TBUF, RBUF and ACF
- two pseudo dual port memory blocks for TBUF and RBUF, one single port memory for ACF
- one single port memory block for TBUF, RBUF and ACF

Using a true dual port memory enables access to the memory for the host controller and the CAN protocol machine at any time without the need of having wait states at the host interface, which are required if a single port memory is chosen. (Not all host controllers support wait states.) A true dual port memory is easy to use but it occupies a lot of area and may not be available for all target technologies.

A pseudo dual port memory (PDP memory) is an option if no true dual port memory is available but the host controller does not support wait states. PDP memory has also two independent ports, but one port is only for write and the other port is only for read access. For the reception data path the CAN protocol machine writes received frames to RBUF and the host reads them. For the transmission data path the host writes frames for transmission to TBUF and the CAN protocol machine reads them. Therefore PDP memory can be used for TBUF and RBUF. The acceptance filters need to be configured before CAN communication will be active and therefore this only requires a single port memory. In summary 3 memory blocks for TBUF (PDP memory), RBUF (PDP memory) and ACF (single port memory) can be used.

As can be seen this results in the fact, that for using PDP memories TBUF cannot be read by the host. For a typical application this is not an issue and for testing TBUF it is recommended to use the loop back features of the IP core (chap. 3.9.10.4).

Using 32 bit wide write accesses to TBUF can be done without a problem when using PDP memories, but if an host interface wrapper is used, that translates the native 32 bit wide interface of CAN-CTRL to a smaller width like the generic 8 bit wide host interface (chap. 7.2), then a read-modify-write access is required. But the PDP memory for TBUF has only a write port for the host side. This is compensated by a pseudo read port for the host side. This pseudo read port is a small buffer that enables the required read-modify-write access and it is mandatory if the generic 8 bit wide host interface (chap. 7.2) is used. This buffer is explained in detail in the following:

Four bytes form a 32 bit wide word, e.g. the addresses 0x50, 0x51, 0x52 and 0x53 form the first word of TBUF and 0x54, 0x55, 0x56 and 0x57 the second one. The pseudo read port buffer stores the bytes, that already have been written to a word if and only if the address points to the same word for consecutive accesses. If this condition is fulfilled, then the pseudo read port buffer makes the data content available for read access. If the address points to a different 32 bit word, then the buffer is cleared to zero. Clearing because of this condition is done for both read and write accesses. Examples are given in Table 3-49.

Table 3-49 Examples for the Pseudo Read Port of TBUF

Step	Action	Address	Data	Pseudo Read Port Buffer
1	Write	0x50	0x11	0x00000011
2	Write	0x52	0x33	0x00330011
3	Write	0x55	0xBB	0x0000BB00
4	Read	0x54	-	0x0000BB00
5	Read	0x52	-	0x00000000

An access to any address outside of the address space of TBUF does not have any impact to the pseudo read port buffer.

If PDP memories are selected by the generic parameter RAM_TYPE, then the pseudo read port buffer can be optionally included into the system and then provides the described behavior in a fully transparent way without any need for configuration or status monitoring.

Parameter RAM_TYPE also provides the option to exclude the pseudo read port feature if not needed. As stated in chap. 7.1.3 a write access to TBUF or ACF has to access a full 32 bit wide word. As a result the pseudo read port feature should be only enabled if the generic 8 bit wide host interface (chap. 7.2) is used.

If the PDP memory option is selected, then the memory for ACF is implemented as single port memory. This is possible, because the acceptance filters shall be set before the CAN node takes part in CAN communication and while CAN communication is active there is no need to change the acceptance filters. As a result the host controller can only access the ACF memory if bit RESET=1.

As described in chap. 3.2 the required memory size for the ACF is rather small. As a result if the PDP option is chosen and therefore the ACF memory is a separate memory block, then it can be advantageous to build the ACF memory as an array of flipflops (“distributed memory”). Modern FPGA synthesis tool will do this automatically to save BRAM blocks for other applications.

3.11 Dual Port versus Single Port Memory

Using a true dual port memory enables access to the memory for the host controller and the CAN protocol machine at any time without the need of having wait states at the host interface, which are required if a single port memory is chosen. (Not all host controllers support wait states.)

Dual port memory consumes more area than single port memory and may not be available for all target technologies. Many FPGAs from Altera and Xilinx support dual port memory but e.g. many Lattice FPGAs don’t support it.

Due to the fact, that the host controller may want to access the memory simultaneously to the CAN protocol machine which transmits or receives CAN frames meanwhile either a dual port solution needs to be used or it is necessary to wait until the access can be granted which results in the requirement for wait states at the host interface.

CAN-CTRL includes an arbiter, which manages access to the memory for the 3 CAN protocol machines for transmission, reception and acceptance filtering for every selected memory type. If single port memory is selected, then this arbiter is extended to manage the accesses from the host controller as well. Accesses from the host controller will be granted if there is no access from one of the CAN protocol machines but after a successful access from the host the following access will be granted to one of the CAN protocol machines. As a result the number of wait cycles is minimized for all parties.

All 3 CAN protocol machines will never request accesses continuously. There will be always a pause cycle between the accesses where an access from the host can be inserted.

4. Time-Triggered CAN

Time-Triggered CAN (TTCAN) is an operation mode according to ISO 11898-4 where frames will be transmitted only at predefined time windows. There are three time window types:

- Exclusive time window (only one node is allowed to transmit one frame with a defined ID)
- Free time window (unused time window for further extension of the network)
- Arbitrating time window (several nodes may transmit a frame and arbitration takes place)

TTCAN is optional and can be disabled using a generic parameter (chap. 3.2).

The timing is defined offline by a TTCAN system administrator and organized in a system matrix as shown in Figure 4-1. A row is called a basic cycle and it starts with a reference message. The reference message is transmitted by the time master. Other messages can be transmitted by any node including the time master.

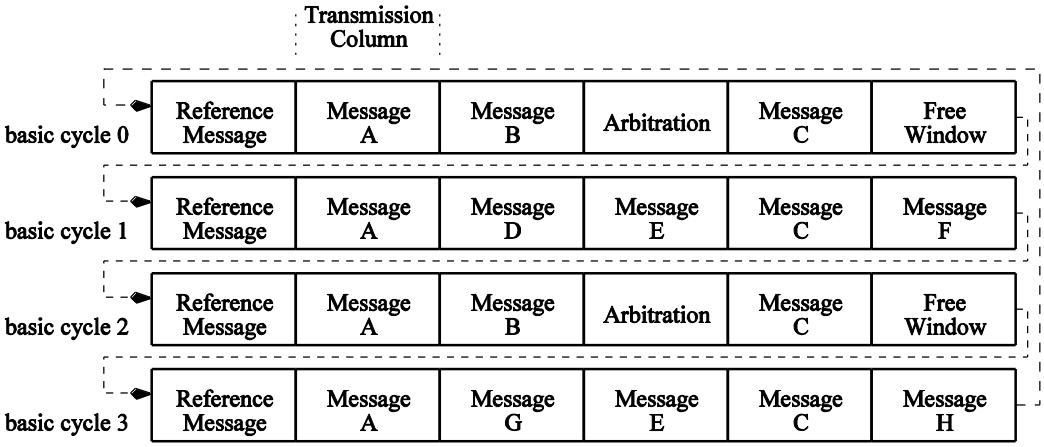

Figure 4-1 Example System Matrix

The time of the SOF of a message is a Sync_Mark. The Sync_Mark of a reference message is the Ref_Mark. Timing is done by a free-running 16 bit timer. The difference between the timer and the value of the Ref_Mark is the cycle time. In other words: each basic cycle starts the cycle time at the Ref_Mark. The cycle time is stored in the RBUF as a time stamp.

The length of a time window is defined by the TTCAN administrator and it is long enough to carry one message. Therefore messages must be transmitted in single shot mode. There is only one exception from this rule: if several arbitrating time window are merged then it is possible to enable retransmission. But this needs to be disabled early enough to not influence the following time window.

To transmit a frame in a time window, CAN-CTRL offers a hardware trigger. This trigger needs to be configured by the host. The desired frame needs to be stored in a TBUF slot and the trigger needs to point to this slot. Then if the cycle time comes to the time that is defined for the trigger, CAN-CTRL automatically transmits the frame in single shot mode.

If the trigger gets active this will be signaled to the host by an interrupt. Then the host needs to configure the next trigger for the next time window. This requires a real-time response by the host. The duration of one time window is the time that the host is allowed to use.

If CAN-CTRL needs to operate as time master, then the reference message needs to be placed in a TBUF slot like all other messages and will be transmitted when the hardware trigger gets active. Detection of a reference message is done automatically by all of the time slaves and the time master. This is done by setting the correct REF_ID and REF_IDE bit of the reference message in the REF_MSG_x registers. Upon detection CAN-CTRL automatically updates the Ref_Mark which starts the cycle time.

Additionally to the trigger for messages, CAN-CTRL includes a watch trigger. This should be used to check whether the time since the last reference message has been too long. The host shall configure the watch trigger for the periodic case where each basic cycle is followed by the next basic cycle or for the

event-triggered case where there is a time gap between the basic cycles and the next cycle is started upon an event. If the watch trigger gets active, this results in the watch interrupt.

Beside support for ISO 11898-4, CAN-CTRL offers a mixture of event-driven CAN communication combined with time-stamping for received messages. This mode is selected if register bit TTTBM=0. In this mode CAN-CTRL acts similar to event-driven CAN communication (TTEN=0), but reference messages can be detected and time-stamps for received messages are provided. Additionally only time triggers and the watch trigger are supported in this mode.

4.1 The TBUF in TTCAN Mode

The behavior if the TBUF depends on the register bit TTTBM. If TTTBM=1, then each TBUF slot can be addressed by the host controller which is required to use transmission triggers. Otherwise if TTTBM=0, then only time-stamping and time triggers are supported.

4.1.1 TBUF in TTCAN Mode if TTTBM=1

In TTCAN mode with TTTBM=1 the STB is used as an array of message slots. Each slot can be addressed by TBPTR. The host can mark a slot as filled or as empty using TBF and TBE. Filled slots are write-locked. TBSEL and TSNEXT have no meaning in TTCAN mode and are ignored.

The PTB can be addressed by TBPTR=0. This makes the PTB useable as any other STB slot. In fact, the PTB has no special properties in this mode and is associated with the STB. This furthermore results in the fact that a successful transmission is always signaled using TSIF.

There is no FIFO operation and no priority decision for the TBUF in TTCAN mode. Furthermore only one frame can be selected for transmission.

A message trigger defines the time when a message needs to be transmitted (the beginning of a time window) and it selects the message using the pointer TTPTR. If the trigger event takes place, then selected message transmission is started. Finally the trigger interrupt is set to signal the host that the next action needs to be prepared.

All transmissions can only be started using a trigger. See chap. 4.4 for details. TPE, TSONE, TSALL, TPSS and TPA are fixed to 0 and ignored in TTCAN mode.

4.1.2 TBUF in TTCAN Mode if TTTBM=0

TTTBM=0 offers the combination of event-driven CAN communication and time-stamping for received messages.

In this mode the PTB and the STB provide the same behavior as if TTEN=0. Then the PTB always has higher priority than the STB and the STB can operated in FIFO mode (TSMODE=0) or in priority mode (TSMODE=1).

4.2 TTCAN Operation

After power-up a time master needs to do the initialization as defined in ISO 11898-4. There may be up to 8 potential time masters in a CAN network. Each one has it's own reference message ID (the last 3 bits of the ID). Potential time masters may try to transmit their reference message according to their priority. Lower-prioritized time masters shall try to transmit their reference messages later.

If TTEN is set, then the 16 bit timer is running. If a reference message is detected or if a time master successfully transmits it's reference message, then CAN-CTRL copies the Sync_Mark of this message to the Ref_Mark which sets the cycle time to 0. The reception of the reference message will set RIF respectively the successful transmission will set TPIF or TSIF. Then the host needs to prepare the trigger for the next action.

A trigger for an action can be a reception trigger. This just triggers the interrupt and it can be used to detect if an expected message is not received. Such a trigger can also be used for other actions, but this depends on the host application.

A different trigger is a transmission trigger. This starts the frame in the TBUF slot, where the trigger points to with TTPTR. If the TBUF slot is marked as empty, then no transmission is started, but the trigger interrupt is set.

It is possible that one host task updates a message slot with a new message if it is necessary by the host application. This could be e.g. a new sensor value. Later then if TTCAN requires this message to be transmitted, it will be activated by the trigger. In other words: one host task needs to be responsible for TTCAN and a different host task may update the message slots. This requires that one message slot is exclusively used for one message (e.g. slot 1 for a temperature sensor). If not enough TBUF slots are available then slots need to be shared by different messages.

The TTCAN host application needs to keep track of the system matrix. If a trigger gets active, then the host application needs to prepare the next transmission column. The host needs to handle also the basic cycles. The reference message includes the cycle count.

Most operations in ISO 11989-4 require single shot transmission. See chap. 3.9.10.1 for further details about handling successful and unsuccessful transmissions.

4.3 TTCAN Timing

CAN-CTRL supports ISO 11898-4 level 1. This includes a 16 bit timer running at the speed of a CAN bit time (defined by S_PRESC, S_SEG_1, S_SEG_2). An additional prescaler is defined by T_PRESC. If TTEN=1 then the timer is counting continuously.

At the SOF of the message, the timer value is the Sync_Mark. If the message was a reference message, then this value is copied to the Ref_Mark. The cycle time is the timer value minus the Ref_Mark. It is the time that is used as time-stamp for received messages or as trigger time for messages, that need to be transmitted. An overflow protection is included and therefore the cycle time is always strictly monotonic inside a basic cycle.

ISO 11898-4 does not include CAN FD baud rate switching. CAN-CTRL therefore always operates the timer with slow nominal bit rate. The timer is running freely and it is not influenced by synchronization or baud rate switching. Therefore a timer tick is not synchronous to the start of a CAN bit.

The timer is running in the CAN clock domain while all control and status bits are in the host clock domain. Therefore the pure timer value is not readable for the host as this would require clock domain crossing. Trigger events need to be used to synchronize actions required by the host to the timer.

Interrupts because of trigger events, reception or transmission require clock domain crossing and therefore include some clocks delay. This only becomes relevant if after a trigger the host application decides to start a transmission. (Therefore it is recommended to use a transmission trigger instead.) In all other cases the host application has enough time to prepare all actions for the next trigger event (which is the next time window). Almost all hosts are fast enough to do a lot of actions during the duration of a CAN frame (which is the duration of a time window).

Timing according ISO 11898-4 level 1 is not perfect. The cycle time counts not synchronously with the CAN bits, because a CAN bit can be shortened or lengthened because of CAN synchronization. Therefore the cycle time of one node may differ compared to the cycle time of other nodes. In many cases this will be +/- 1 tick but it is not limited to this. The begin of a new basic cycle with the transmission of a reference message will resynchronize the nodes.

If a transmission trigger becomes active, then the appropriate transmission can only be started with the next CAN bit. Therefore the earliest point of transmission of the SOF of the frame will be at TT_TRIG+1.

4.4 TTCAN Trigger Types

The trigger type is defined by TTYPE. TTPTR is a pointer to a TB message slot and TT_TRIG defines the cycle time of the trigger.

Triggers and the related actions shall finish before the maximum cycle time 0xffff is reached as this is the maximum length of a basic cycle.

Except for the immediate trigger, all triggers set TTIF if TTIE is enabled.

If TTTBM=1, only time triggers are supported. Using other triggers in this mode will result in setting TEIF.

If a trigger is activated after a write access to TT_TRIG_1, then write access to TT_CFG_0, TT_CFG_1, TT_TRIG_0 and TT_TRIG_1 are locked until the trigger time is reached (TTIF is set if TTIE is set) or an error is detected (TEIF is set). Therefore no new trigger can override an active trigger. The write lock is also removed if TTEN=0.

4.4.1 Immediate Trigger

An immediate trigger starts the immediate transmission of a frame that is pointed to by TTPTR. TTIF is not set. To start a trigger, TT_TRIG_1 needs to be written. The value that is written, does not care for an immediate trigger.

In TTCAN mode TPE, TSONE, TSALL cannot be used. The immediate trigger is the replacement. Only the transmission of one frame can be started with an immediate trigger. The host must not command a second immediate trigger before the first transmission has been completed successfully or unsuccessfully.

For an immediate trigger the single shot mode can be selected by TSSS. A transmission can be aborted using TSA. (TPSS and TPA have no meaning.)

If TTPTR of an immediate trigger points to an empty slot, then TEIF is set.

4.4.2 Time Trigger

A time trigger just generates an event by setting TTIF and therefore generates an interrupt. No other actions are done.

The time trigger can be used as a reception trigger. If the node expects a message to be received in a time window, then if the message is missing and RIF is not set, a reception trigger can be used to signal this. A reception trigger shall be set after the latest moment when the message is expected to be successfully received.

If TT_TRIG is lower than the actual cycle time, then TEIF is set.

A time trigger can be used if TTTBM=1. This is the only trigger type, that is available in this mode.

4.4.3 Single Shot Transmit Trigger

A single shot transmit trigger is intended to be used for exclusive time windows, where a message needs to be transmitted in single shot mode. The selected message is defined by TTPTR.

Single shot mode is automatically used regardless of the state of TSSS. The register bit TSSS is ignored and stays unchanged.

Single shot transmit triggers are intended to be used for exclusive time windows. For this ISO 11898-4 defines a transmit enable window of up to 16 ticks of the cycle time. The register bits TEW(3:0)+1 define the number of ticks. A frame cannot be started if the bus is occupied by another frame. This should not happen in an exclusive time window, but the transmit enable window ensures that there is no delayed start which would result in a violation of the next time window. If the transmit enable window closes and the frame could not be started, then it is aborted. As a result the TB slot of the frame will be marked as empty and AIF will be set if AIE is set. The data of the frame in the TB slot will not be touched and therefore the slot only needs to be marked as filled again if the same data shall be transmitted in a next attempt.

If TT_TRIG is lower than the actual cycle time, then TEIF is set and no action is done.

4.4.4 Transmit Start Trigger

A transmit start trigger is intended to be used for merged arbitrating time windows, where several nodes may transmit messages and CAN arbitration takes place. The selected message is defined by TTPTR.

TSSS defines if retransmission or single shot mode is used.

If the selected frame cannot be transmitted (arbitration loss, several transmissions after an error), then the transmission can be stopped using the transmit stop trigger (chap. 4.4.5).

If TT_TRIG is lower than the actual cycle time, then TEIF is set and no action is done.

4.4.5 Transmit Stop Trigger

A transmit stop trigger is intended to be used to stop a transmission, that is started with the transmit start trigger (chap. 4.4.4).

If a transmission is stopped, then the frame is aborted. Therefore AIF is set is AIE is set and the TB slot of the frame is marked as empty. The data of the frame in the TB slot will not be touched and therefore the slot only needs to marked as filled again if the same data shall be transmitted in a next attempt. Chap. 3.9.5 gives further details.

The behavior of a transmit stop trigger is similar to the end of the transmit enable window used by the single shot transmit trigger (chap. 4.4.3). If a transmit stop trigger points to an empty slot (which means, that the message has already been transmitted), then not action is done an TEIF is not set.

If TT_TRIG is lower than the actual cycle time, then TEIF is set but the stop is executed.

4.5 TTCAN Watch Trigger

In contrast to the generic trigger types explained in chap. 4.4 the watch trigger has a dedicated interrupt flag WTIF. If the cycle count equal to the value defined by TT_WTRIG, then WTIF is set is WTIE is set.

The watch trigger is intended to be used if the time since the last valid reference message has been too long. Reference messages can be received in a periodic cycle or after an event. The host application needs to take care of this and has to adjust the watch trigger accordingly.

The default value of WTIE is 1 and therefore the default watch trigger 0xffff is automatically valid. To turn off the watch trigger, WTIE needs to be set to 0.

If TT_WTRIG is updated and it is lower than the actual cycle time, then TEIF is set.

The watch trigger can be used if TTTBM=1.

4.6 Disabling TTCAN

If TTCAN is disabled using a generic parameter (chap. 3.2), then the appropriate registers are removed and the bits are fixed to the reset values. This holds for TTTBM, TBSLOT, TTCFG, REF_MSG_x, TRIG_CFG_x, TT_TRIG_x and TT_WTRIG_x. Furthermore the logic cells responsible to TTCAN are also removed which decreases the area consumption of the core.

5. CiA 603 Time-Stamping

5.1 Time-Stamping

CAN in Automation (CiA) defines in specification 603 a method for time-stamping with at least 16 bits, which is optionally supported by CAN-CTRL. It can be used in addition to TTCAN or stand-alone.

The basic concept of CiA 603 is to have a free-running timer which counts clock cycles and not CAN bit times. The precision shall be at least 10µs (16 bit) or 1µs (32 bit or more). CAN-CTRL requires an external free-running timer (up to 64 bit) connected to the input timer_in and running with timer_clk. Unused bits of timer_in shall be fixed to 0.

The clock timer_clk can be

- Asynchronous to can_clk and host_clk
- Synchronous to can_clk
- Synchronous to host_clk
- Synchronous to can_clk and host_clk

CDC is used as necessary. Therefore if timer_clk is not synchronous to can_clk, then the acquired time-stamps depend on the relation of these both clocks and there will be a delay because of the CDC.

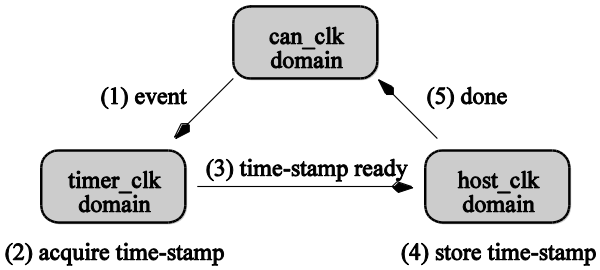

Figure 5-1 Time-Stamping and CDC

As shown in Figure 5-1 after an event a copy of the timer is taken. This is the time-stamp. This time-stamp is stored into RTS or TTS respectively where it can be read by the host. After it is stored, a handshake signal signals the completion of the action. This describes a CDC mechanism with 3 clock domains.

Time-stamps are acquired at the sample point of SOF or the EOF bit where the frame is taken to be valid. This can be selected by the configuration bit TIMEPOS. Seven recessive bits after the ACK delimiter form the EOF of a CAN / CAN FD frame. A frame gets valid for a receiver at the last but one bit of EOF, but for a transmitter at the last bit of EOF.

Software-based time-stamping, which is widely used in many systems, relies on the reception and transmission interrupt. Therefore time-stamping at EOF is recommended.

CiA 603 is supposed to support time-stamping and time-synchronization of AUTOSAR. For AUTOSAR one node in the CAN network is the time master. A time master transmits a synchronization message (SYNC message). The time-stamp of the SYNC message is acquired by the time master and all time slaves. The difference in time between the event of commanding a SYNC message until the time when the SYNC message actually gets transmitted will be transmitted in a follow-up (FUP) message by the time master. Therefore CAN-CTRL supports only one time-stamp for transmitted frames (TTS) but individual time-stamps for all received frames (RTS). Generation of a time-stamp for transmitted frames can be enabled or disabled for each frame individually using bit TTSEN inside a TBUF slot.

CiA defines rules to read out the timer as well as to modify the timer. CAN-CTRL does not include the timer, but uses an external timer. CAN-CTRL only includes the mechanism of time-stamping, the register to store TTS and the memory to store RTS for each frame.

Register bit TIMEEN enables or disables time-stamping. If disabled TTS and RTS are invalid.

5.2 The External Timer

An external timer needs to be used for CiA 603 time-stamping. This timer is not included in CAN-CTRL because it may be that such a timer already exists in a system which can be re-used for CiA 603 time-stamping or that a CAN multicore instance (chap. 9) uses one timer for all multicores.

The external timer shall be at least a free running counter, that increments with timer_clk. Such a very simple timer is also included in the testbench. The bit width of the timer is free to chose. CAN-CTRL stores either 32 or 64 bits when a time-stamp is acquired. If the timer has less bits, the unused bits shall be filled with zeros.

A timer for CiA 603 time-stamping shall have a precision of at least 10µs (16 bit) or 1µs (32 bit or more) and at most 1ns. It is based on a physical second and not on the CAN bit time. The timer shall start from zero and shall counter upward continuously.

CiA 603 defines a prescaler for the timer to compensate different levels of precision from different nodes in a network. As an alternative and replacement for the prescaler for CAN-CTRL is suggested to use a 64 bit timer and do the precision adaption using software if necessary.

CiA 603 defines the option to modify the timer value. If a modification is possible, then it shall take place as fast as possible or immediately in the ideal case. If the timer is not in the same clock domain as the host controller, then a modification may take some time, but such a delay shall be minimized.

CiA 603 requires the ability to read the timer at any time.

In summary such a timer is a free-running counter, which can be read and written by the host while such an access may include some delays because of clock domain crossing. Such a timer together with the CIiA 603 time-stamping provided by CAN-CTRL support time synchronization as defined by AUTOSAR.

5.3 Timer Bit Width

CiA 603 time-stamping focuses on support for AUTOSAR. AUTOSAR handles time synchronization and therefore time-stamps have to be converted into real time. With the recommendation, that the precision shall be at least 10µs (16 bit) or 1µs (32 bit or more) a bit width of 32 bits is sufficient for good AUTOSAR support. It has to be noted, that the conversion into real time results in some computational effort, that has to be done by the host processor, but this is part of the AUTOSAR concept.

Using a bit width of 64 bits can be preferable in custom applications where the time-stamp can be used immediately as real time without conversion. 64 bits are enough for many thousand years without overflow. Such custom applications need to take care of the fact that all nodes in a network need to count with the same <timer_clk> frequency or otherwise the clock rate ratios need to be considered if time-stamps of other nodes are used for calculations.

6. CAN Bit Time

6.1 Data Bit Rates

CAN 2.0B defines data bit rates up to 1Mbit/s. For CAN FD there is no fixed limitation. For real life systems the data rates are limited by the used transceiver and the achievable clock frequency for the CAN-CTRL core which depends on the used target cell library.

The CAN-CTRL core can be programmed to arbitrarily chosen data rates only limited by the range of the bit settings in the appropriate bit timing and prescaler registers.

6.2 Definitions

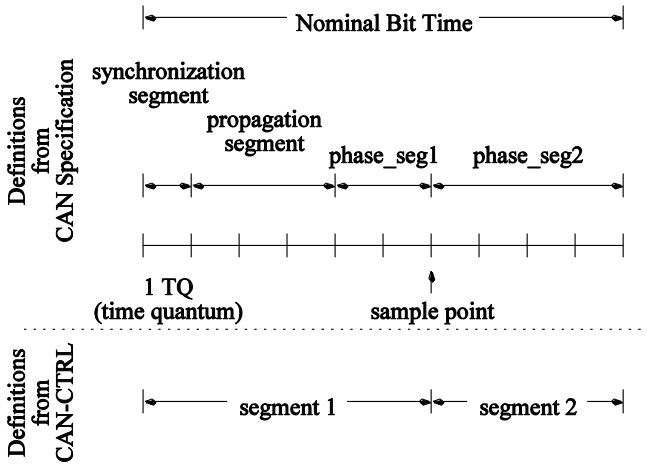

Figure 6-1 CAN Bit Timing Specifications

The CAN bit time BT consists of several segments as shown in Figure 6-1. Each segment consists of a number of time quanta units n_{TQ}. The duration of a time quanta TQ is:

$$TQ = \frac{n_{prescaler}}{f_{CLOCK}}$$

The values of n_{TQ} and $n_{prescaler}$ have to be chosen depending on the system clock frequency f_{CLOCK} to match BT_{real} as close as possible to $BT_{ideal} = 1/BR$ where BR is the CAN bus baud rate:

$$BT_{ideal} \approx BT_{real} = \frac{n_{prescaler} \cdot n_{TQ}}{f_{\mathrm{CLOCK}}} = t_{\mathrm{Seg\_1}} + t_{\mathrm{Segt\_2}}$$

The CAN specification requires several relationships between the segment lengths (Table 6-1) which results in relationships between t_{Seg_1}, t_{Seg_2} and the maximum synchronization jump width t_{SJW}. Please note that Table 6-1 lists the minimum configuration ranges defined by the CAN / CAN FD specification.

Table 6-1 CAN Timing Segments (Minimum Configuration Ranges)

Segment	Description		
SYNC_SEG	Synchronization Segment = 1 TQ		
PROP_SEG	Propagation Segment [1...8] TQ [1...48] TQ [1…48] TQ [0...8] TQ	 CAN 2.0 bit rate CAN 2.0 bit rate CAN FD nominal bit rate CAN FD data bit rate	 CAN FD not enabled CAN FD enabled
PHASE_SEG1	Phase Buffer Segment 1 [1...8] TQ [1...16] TQ [1...16] TQ [1...8] TQ	 CAN 2.0 bit rate CAN 2.0 bit rate CAN FD nominal bit rate CAN FD data bit rate	 CAN FD not enabled CAN FD enabled
PHASE_SEG2	Phase Buffer Segment 2 [2...8] TQ [2...16] TQ [2...16] TQ [2...8] TQ	 CAN 2.0 bit rate CAN 2.0 bit rate CAN FD nominal bit rate CAN FD data bit rate	 CAN FD not enabled CAN FD not enabled
SJW	Synchronization Jump Width [1...4] TQ [1...16] TQ [1...16] TQ [1...8] TQ	 CAN 2.0 bit rate CAN 2.0 bit rate CAN FD nominal bit rate CAN FD data bit rate	 CAN FD not enabled CAN FD not enabled
IPT	Information Processing Time = [0...2] TQ PHASE_SEG2 ≥ IPT		

As can be seen in Figure 6-1 the CAN-CTRL core collects SYNC_SEG, PROP_SEG and PHASE_SEG1 into one group and the length of the group is configurable with t_{Seg_1}. Table 6-2 lists the available configuration ranges. Please note that the CAN-CTRL core does not check if all rules are met and offers a wider configuration range than defined by the CAN / CAN FD specification.

Table 6-2 CAN-CTRL Timing Settings (Available Configuration ranges)

Setting	Requirements			
t_{Seg_1}	[2…65] TQ [2…65] TQ [2…17] TQ		CAN 2.0 bit rate CAN FD nominal bit rate CAN FD data bit rate	(slow) (slow) (fast)
t_{Seg_2}	[1…8] TQ [1…32] TQ [1…8] TQ	$t_{Seg\_1} \geq t_{Seg\_2} + 2$ $t_{Seg\_1} \geq t_{Seg\_2} + 2$ $t_{Seg\_1} \geq t_{Seg\_2} + 1$	CAN 2.0 bit rate CAN FD nominal bit rate CAN FD data bit rate	(slow) (slow) (fast)
t_{SJW}	[1…16] TQ [1…16] TQ [1…8] TQ	$t_{Seg\_2} \geq t_{SJW}$ $t_{Seg\_2} \geq t_{SJW}$ $t_{Seg\_2} \geq t_{SJW}$	CAN 2.0 bit rate CAN FD nominal bit rate CAN FD data bit rate	(slow) (slow) (fast)

For CAN 2.0 bit rate and CAN FD (slow) nominal bit rate the register settings S_Seg_1, S_Seg_2, S_SJW and S_PRESC define the appropriate segment lengths. For CAN FD (fast) data bit rate the register F_Seg_1, F_Seg_2, F_SJW and F_PRESC are valid.

$$t_{Seg\_1} = (S\_Seg\_1 + 2) \cdot TQ \qquad t_{Seg\_1} = (F\_Seg\_1 + 2) \cdot TQ$$
$$t_{Seg\_2} = (S\_Seg\_2 + 1) \cdot TQ \qquad t_{Seg\_2} = (F\_Seg\_2 + 1) \cdot TQ$$
$$t_{SJW} = (S\_SJW + 1) \cdot TQ \qquad t_{SJW} = (F\_SJW + 1) \cdot TQ$$
$$n_{prescaler} = S\_PRESC + 1 \qquad n_{prescaler} = F\_PRESC + 1$$

A CAN FD core switches from slow nominal bit rate to fast data bit rate at the sample point in the BRS bit and switches back at the sample point in the CRC delimiter bit.

An illustration a suitable bit rate configuration is given in Figure 6-2 where $f_{\text{tq\_clk}} = \frac{f_{\text{CLOCK}}}{n_{prescaler}}$. The bit time BT_{real}, the sample point and the synchronization jump width SJW will be derived from tq_clk.

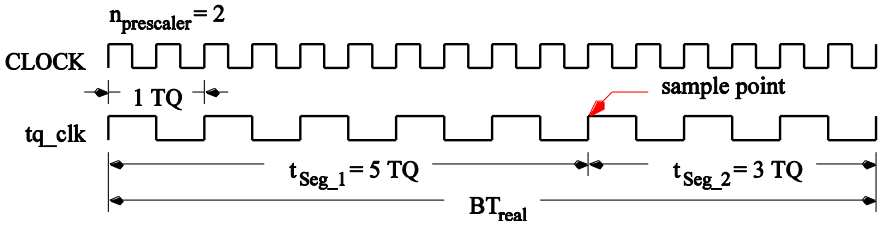

Figure 6-2 Clock Division for Bit Sampling

Having the requirements from Table 6-2 in mind the host controller has to define the length of segment 1, segment 2 and the synchronization jump width for the slow bit rate and if required for CAN FD for the fast bit rate.

Some suggestions, given as rules of the thumb:

- Segment 1 must be slightly larger than segment 2. Then the sample point is in the a little bit later than in the middle of the bit time.
- The synchronization jump width must not be bigger than segment 2. If SJW is too small the CAN node may be too slow to resynchronize, if SJW is too big then the CAN node may resynchronize too often. SJW being half as long as segment 2 seems to be a suitable value.
- All CAN nodes connected to a CAN bus should choose similar settings if possible.

The fastest CAN bus speed can be configured using the minimum timing values. But there are some things that need to be considered:

- If the prescalers are bigger than one then all other timing parameters could be set to zero, but this violates the rule $t_{Seg\_1} \geq t_{Seg\_2} + 2$ (slow speed) resp. $t_{Seg\_1} \geq t_{Seg\_2} + 1$ (fast speed) and therefore S_Seg_1 and F_Seg_1 should be at least set to 1. But such a selecting of timing parameters works in theory, but may not be robust enough for real nets.
- If the prescalers are set to one then synchronization becomes difficult. In general the CAN specification requires one bit time to be at least 8 TQ long for CAN 2.0 and CAN FD slow nominal bit rate. For CAN-CTRL one bit time of fast data bit rate needs to be also at least 8 TQ long if the fast prescaler is set to 1.
- In summary the fastest CAN bus frequency is the can_clk frequency divided by 8: prescalers set to 1 and 8 TQ bit time.

6.3 Example Configuration

This example refers to Figure 6-2. It is an example for CAN 2.0 / slow bit rate configuration but all statements can be easily translated to CAN FD (fast) data bit rate. The following steps need to be carried for the configuration of the CAN-CTRL core:

1. Set bit RESET=1.
2. Set registers S_Seg_1 and S_Seg_2:
   In the example the data rate on the bus $f_{BUS} = 1\,\text{Mbaud}$ and the system clock is $16\,\text{MHz}$.
   The values of n_{TQ} and $n_{prescaler}$ have to be selected to fit BT_{real} as close as possible to BT_{ideal}.
   In this example $n_{prescaler} = 2$ and $n_{\text{TQ}} = 8$ are chosen which results in a perfect match:
   $BT_{ideal} = BT_{real} = 8\,\text{TQ}$.
   With $BT_{real} = t_{\text{Seg\_1}} + t_{\text{Segt\_2}}$ and the time segment definitions given in chapter 6.1 $t_{\text{Seg\_1}} = 5\,\text{TQ}$

and $t_{\text{Seg\_2}} = 3\,\text{TQ}$ can be chosen as suitable values which finally results in the register settings S_Seg_1=3 and S_Seg_2=2.

3. Load the acceptance code and mask registers (optional).
4. Set register S_SJW:
   With $t_{\text{Seg\_2}} \geq t_{SJW}$ one is free to chose $t_{\text{SJW}} = 3$ which finally results in 2.
5. Load the clock prescaler register S_PRESC: $n_{prescaler} = \text{PRESC} + 1$ results in S_PRESC=1.
6. Set bit RESET=0.
   The given order is not mandatory. It is merely necessary to set bit RESET=1 at the beginning, as it is otherwise not possible to load the bit timing, ACODE and AMASK registers. RESET=0 is required upon completion of the configuration. The controller then waits for 8 recessive bits (frame end) and resumes its normal operation.
7. Continue with configuration of interrupts other configuration bits and execute commands.

6.4 CAN Bit Timing Calculator (CBC)

The release package includes a software tool for MS Windows for the calculation of the CAN bit timings. This tool can be used to calculate the CAN 2.0 as well as the CAN FD nominal and data bit rates.

Starting with the setting of the clock frequency of the CAN-CTRL core and the desired Baud rate this tool outputs all possible settings for the prescaler, segment 1 and segment 2. For SJW the tool gives the maximum value. Furthermore the tool outputs the position of the sample point as a factor of a bit time.

The tool gives on one hand the time in multiples of TQ as well as the register value. For example “t_seg1=7” means that segment 1 is 7 TQ long and the register value needs to be set to S_Seg_1=5.

By default the tool tries to calculate the settings with a tolerance of 0%. In other words: With the given clock speed and these settings the desired baud rate will be synthesized exactly. It is possible to modify the tolerance setting to get also settings that will not match exactly. Please note that the tool does not check if the selected tolerance is inside of the CAN specification.

This tool performs and exhaustive search for possible settings. The calculation time will increase for fast baud rates and a big tolerance value. The tools searches for settings in steps of 1 Baud.

Additionally to the software tool a MS Excel sheet is included in the release package. This sheet can be useful if CAN-CTRL needs to be adapted to an existing CAN network where the bit timing configuration is already fixed. This Excel sheet includes the configuration parameters recommended by CAN in Automation (CiA).

6.5 Bit Rate Switching and the Sample Point

In a CAN network when CAN FD frames with bit rate switching are used the exact position of the sample point is important and needs to be for all nodes as similar as possible.

The bit rate is switched after the sample point of the BRS bit and the sample point of the CRC delimiter. Therefore the lengths of these bits are intermediate. As can be seen in Figure 6-3 the position of the sample point makes a big difference for the absolute length of the BRS bit. (1 TQ of the slow nominal bit rate may be much bigger than several TQs of the fast data rate.) If the data rate is much higher than the nominal bit rate then an earlier or later sample point may lead to false samples at fast data bit rate.

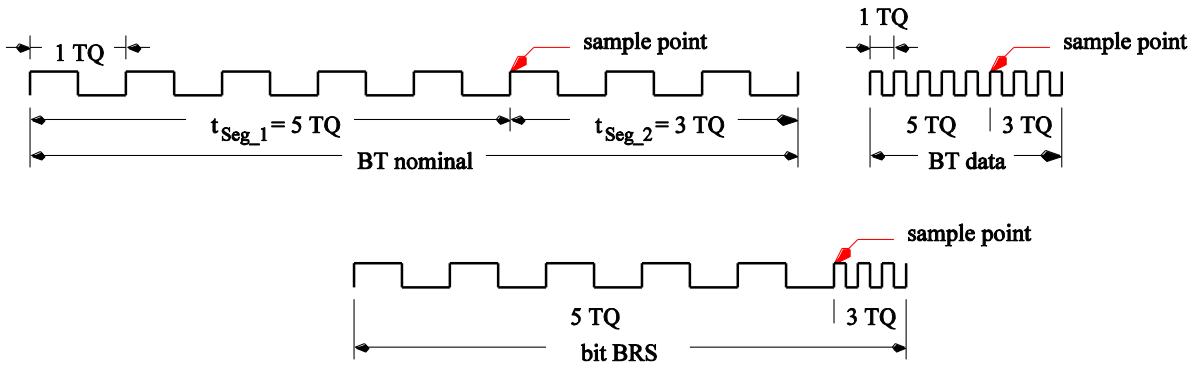

Figure 6-3 Bit Rate Switching at bit BRS $(S\_PRESC \neq F\_PRESC)$

6.6 Bit Timing Configuration for CAN FD Nodes

As stated in chap. 6.5 the position of the sample point is very important for bit rate switching. Therefore there is the general recommendation for all CAN FD nodes to use exactly the same timing parameters in a CAN network. In contrast to classic CAN where only the data rate needs to be the same for all nodes, CAN FD requires the sample point at the same position. In others words segment 1 and segment 2 need to be configured to the same length for all nodes. Furthermore it is recommended to use also the same length of one TQ to make synchronization for all nodes equal.

This requirement can only be achieved if all nodes run with the same base clock frequency (called can_clk for CAN-CTRL) or at least with compatible clock frequencies. The recommended settings are 20MHz, 40MHz or 80MHz.

6.7 TDC and RDC

For CAN FD nodes transmitter delay compensation (TDC) can be optionally enabled while receiver delay compensation (RDC) is automatically active. These features have the following background: For communication with CAN FD data bit rate it may be that the delay of the transmitting transceiver or the delay of the bus is bigger than a bit time. This needs to be compensated. Without TDC, the bit rate in the data-phase of FD frames is limited by the fact that the transmitter detects a bit error if it cannot receive its own transmitted bit latest at the sample point of that bit.

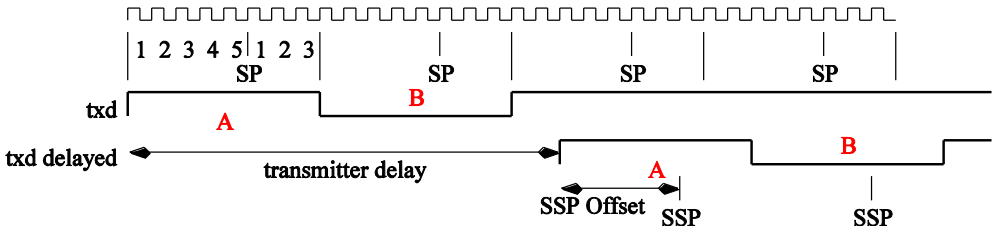

Figure 6-4 Transmitter Delay

Figure 6-4 gives an example of the effect of a big transmitter delay. In this figure a txd data stream is shown starting with bits A and B. One bit time consists of $t_{Seg\_1} = 5\,TQ$ before the Sample Point (SP) and $t_{Seg\_2} = 3\,TQ$ behind. In this example the transmitter delay is longer than 2 bit times. Therefore the original SP cannot be used to sample the correct bit value and the CAN FD specification defines an additional Secondary Sample Point (SSP). If TDC is enabled with bit TDCEN=1 then CAN-CTRL automatically determines the transmitter delay. The position of the SSP is the transmitter delay plus the SSP Offset which is defined by the configuration bits SSPOFF. SSPOFF must be given as a number of TQ and it is suggested to set t_{Seg_1} equal to SSPOFF. (Please remember that F_SEG_1 defines t_{Seg_1} during data bit rate. Therefore in the example given in Figure 6-4 SSPOFF=5 is chosen, because F_SEG_1=3.)

ISO 11898-1:2015 requires to use only F_PRESC=0 or 1 if that TDC is used. With this requirement CAN-CTRL is capable of automatically determining a transmitter delay of up to 3 bit times of the fast data bit rate.

A delay bigger than a bit time needs to be taken into account also for receiving nodes. CAN FD defines an additional hard synchronization at the falling edge between bit FDF =1 and the following r0 bit. (At this time the slow nominal bit rate is active.) Synchronization for CAN and CAN FD is in general defined to operate in time steps of one TQ. But steps of one TQ at nominal bit rate may be to coarse for a synchronization for the fast data bit rate. Therefore the additional hard synchronization at the FDF bit needs to violate this rule and needs to synchronize as exact as possible and only limited by the clock frequency of the system. CAN-CTRL uses this special synchronization and this is called RDC. RDC is automatically done during reception if FDF =1, no matter if TDC is enabled or not.

6.8 Bit Timing Recommendations

CAN FD timing configuration requires to use the same data rate and the same sample point for all nodes in a CAN network. Therefore it is recommended to use 20MHz, 40MHz or 80MHz as source for the CAN protocol machine (can_clk). In Table 6-4, Table 6-5 and Table 6-6 there are some recommendations for the bit timing settings that should apply to all nodes in a CAN FD network. These settings are recommendations from CAN in Automation (CiA).

Please note that the bit time should be at least 8 TQ if the prescaler is set to 1 for a stable communication. Lower settings are possible, but may cause problems.

The decision if TDC is used or not depends on the CAN network. The tables only give a suggestion and TDC can be enabled or disabled as needed.

Table 6-3 Abbreviations for Table 6-4, Table 6-5 and Table 6-6

Abbreviation	Description
PSP	Primary Sample Point
SSP	Secondary Sample Point
Seg 1	Segment 1
Seg 2	Segment 2
TDC	Transmitter Delay Compensation (see SSPOFF)

Table 6-4 Recommendations for 20MHz can_clk

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC [clk]
0.25 (arbitration)	80	-	1	80	64	16	16	-
0.5 (arbitration)	80	-	1	40	32	8	8	-
0.5	80	(disable)	1	40	32	8	8	-
0.833	79	(disable)	1	24	19	5	5	-
1	80	80	1	20	16	4	4	16
1.538	77	77	1	13	10	3	3	10
2	80	80	1	10	8	2	2	8
4	80	80	1	5	4	1	1	4
5	75	75	1	4	3	1	1	3

Table 6-5 Recommendations for 40MHz can_clk

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC [clk]
0.25 (arbitration)	80	-	2	80	64	16	16	-
0.5 (arbitration)	80	-	1	80	64	16	16	-
0.5	80	(disable)	2	40	32	8	8	-
0.833	79	(disable)	2	24	19	5	5	-
1	80	80	1	40	32	8	8	32
1.538	77	77	1	26	20	6	6	20
2	80	80	1	20	16	4	4	16
3.077	77	77	1	13	10	3	3	10
4	80	80	1	10	8	2	2	8
5	75	75	1	8	6	2	2	6
6.667	83	83	1	6	5	1	1	5
8	80	80	1	5	4	1	1	4
10	75	75	1	4	3	1	1	3

Table 6-6 Recommendations for 80MHz can_clk

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC [clk]
0.25 (arbitration)	80	-	4	80	64	16	16	-
0.5 (arbitration)	80	-	2	80	64	16	16	-
0.5	80	(disable)	4	40	32	8	8	-
0.833	79	(disable)	4	24	19	5	5	-
1	80	80	2	40	32	8	8	64
1.538	77	77	2	26	20	6	6	40
2	80	80	2	20	16	4	4	32
3.077	77	77	2	13	10	3	3	20
4	80	80	1	20	16	4	4	16
5	75	75	1	16	12	4	4	12
6.667	83	83	1	12	10	2	2	10
8	80	80	1	10	8	2	2	8
10	75	75	1	8	6	2	2	6

7. Host Interfaces

CAN-CTRL is a 32 bit peripheral component which provides a generic 32 bit host interface. Additionally various host interface wrappers are provided. These wrappers are described in the following chapters and can be used in addition to CAN-CTRL.

None of the host interfaces uses tri-state busses. Dashed lines in the figures of the following subchapters mean “don’t care”.

7.1 Generic 32 Bit Wide Synchronous Host Interface

The generic 32 bit wide host interface is the native interface of CAN-CTRL. All other interfaces are implemented as wrappers and therefore all definitions of this chapter are valid for the wrappers too.

Every byte of the register map (Table 3-5) is assigned to an address. Using the 32 bit host interface therefore the 2 LSBs of the address (host_adr) are ignored by CAN-CTRL.

The host interface may insert wait states where signal host_ready shows if a wait state is inserted.

If dual port memory is selected by generic parameter (chap. 3.2), then only one fixed wait state for the read access is required. In other words: a read access always executes in two clock cycles and a write access always executes in one clock cycle. Because of this fixed number of clock cycles this document speaks about “no wait states if dual port memory is selected”.

If single port memory is selected by generic parameter (chap. 3.2), then additional wait states may be inserted for both read and write accesses. In most cases zero additional wait state are necessary.

7.1.1 Write Access

The Address (host_adr), Data (host_di), Chip Select (host_cs_b) and Write Enable (host_wr_b) signals must be valid throughout the write process. Writing into the registers occurs at the rising edge of the system clock (host_clk). All mentioned signals have to be stable around the positive edge of host_clk with setup and hold times t_{setup} and t_{hold} depending on the used technology.

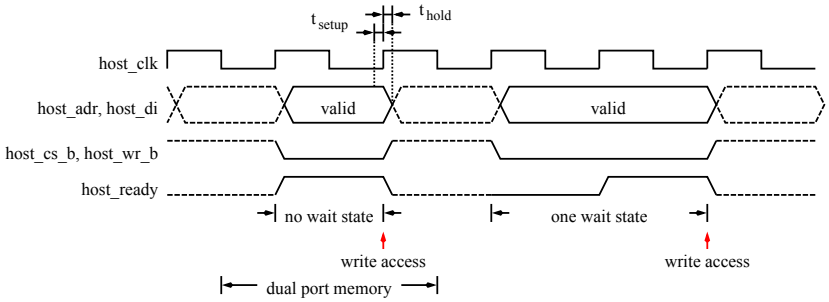

Figure 7-1 Write Operation

Figure 7-1 shows two write accesses: one without wait states and one with one wait state.

Signal host_wr_b is 4 bits wide and controls the bytes to be written. See chap. 7.1.3 for details.

7.1.2 Read Access

The Address (host_adr), Chip Select (host_cs_b) and Read Enable (host_rd_b) signals must be valid throughout the read process. Reading occurs at the rising edge of the system clock (host_clk). All mentioned signals have to be stable around the positive clock edge with setup and hold times t_{setup} and t_{hold} depending on the used technology.

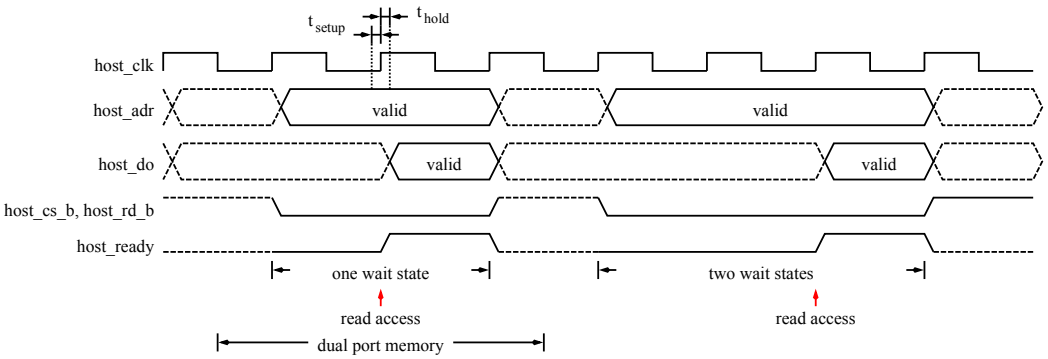

Figure 7-2 Read Operation

Figure 7-2 shows two read accesses: one with one wait state and one with two wait states. The read data value stays valid after the read access for only one clock cycle. As a consequence each read access executes in at least 2 cycles of host_clk.

Signal host_rd_b is 4 bits wide to control the read bytes. See chap. 7.1.3 for details. Although CAN-CTRL always executes 32 bit wide reads, those 4 control bits are necessary to optimize the access arbiter if single port memory is used.

7.1.3 Data Alignment

Table 7-1 Active Byte Lanes (little endian)

host_rd_b host_wr_b	host_adr[1:0]	data[31:24]	data[23:16]	data[15:8]	data[7:0]
1110b	00b	-	-	-	Active
1101b	01b	-	-	Active	-
1011b	10b	-	Active	-	-
0111b	11b	Active	-	-	-
1100b	00b	-	-	Active	Active
0011b	10b	Active	Active	-	-
0000b	00b	Active	Active	Active	Active

The 32 bit interface provides the capability of accessing a byte, a half word or a full 32 bit word as shown in Table 7-1. Signal host_adr is given only for completeness as the 2 LSBs are ignored by CAN-CTRL and only the host_rd_b and host_wr_b signals are relevant.

TBUF and ACF are implemented as true 32 bit wide memories. Therefore a write access to TBUF or ACF is only executed if it is performed as a full 32 bit word access (host_wr_b=0000b). Read accesses and write accesses to other memory locations are not restricted.

7.2 Generic 8 Bit Wide Synchronous Host Interface

The generic 8 bit wide interface (component can_sync_08) is a wrapper that enables CAN-CTRL to be used by an 8 bit host controller.

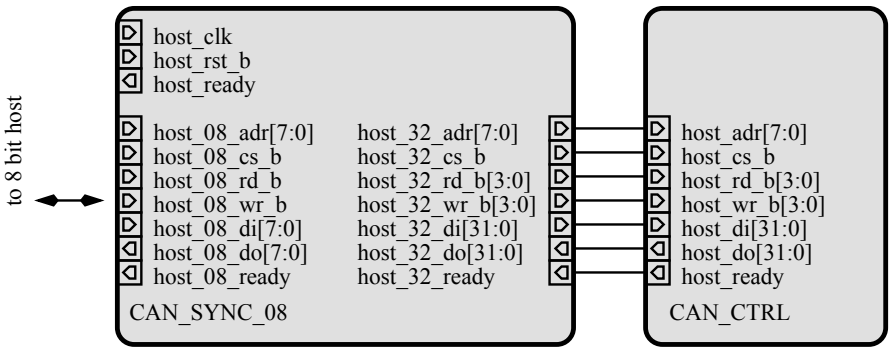

Figure 7-3 Generic 8 Bit Wide Host Interface (only relevant signals shown)

The interface maps 8 bit wide accesses to 32 bit wide accesses. For a read operation this is straightforward and the wrapper extracts the selected byte in the same way as shown in Figure 7-2.

For a write operation the wrapper needs a read-modify-write access where the full 32 bits of data are read with the first rising edge of host_clk. Then the 32 bits are modified (the new 8 bits are embedded) and written back with the second rising edge of host_clk. (See Figure 7-4 for details.) Therefore a write always takes at least two clocks and the next access can only be performed if signal host_ready is high.

If pseudo dual port memories are used (see generic parameter RAM_TYPE and chap. 3.10), then there is a limitation to do a read access from the TBUF memory area. But inside this area the generic 8 bit wide interface requires read-modify-write access to perform a 32 bit grouped write access. A read-modify-write is only possible if a read access is possible. As a consequence when using the generic 8 bit wide host interface it is necessary to add the pseudo read port to the pseudo dual-port memory. Chap. 3.10 explains further details.

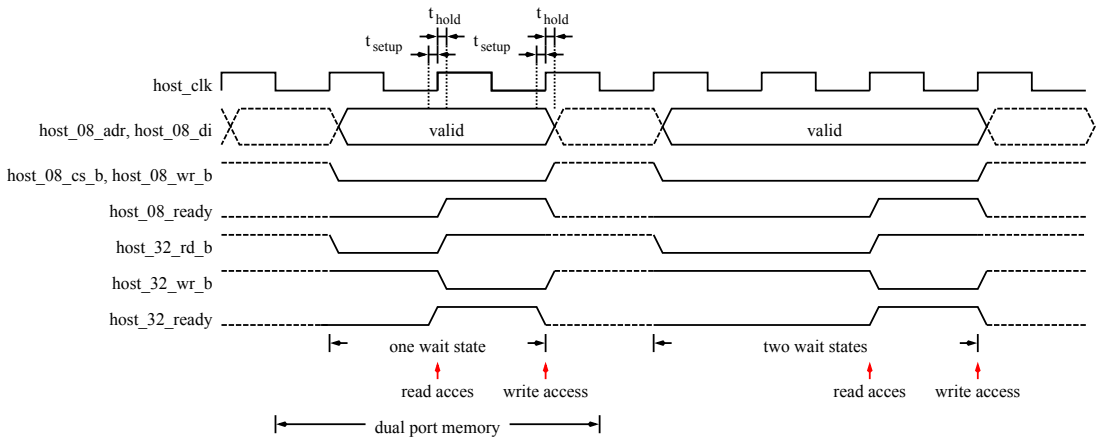

Figure 7-4 Write Operation with 8 Bit Host Interface

7.3 AMBA APB

The memory interface to the CAN core, which has been described in chap. 7.1, can be connected to a 32 bit AMBA APB interface using a wrapper. Source code for the wrapper is provided in the release package (compatible to: AMBA APB Protocol Specification v2.0 (ARM IHI 0024C (ID041610))). The wrapper shall be connected to the CAN-CTRL core as shown in Figure 7-5. The wrapper does not use the APB signals <presetn> and <pprot>.

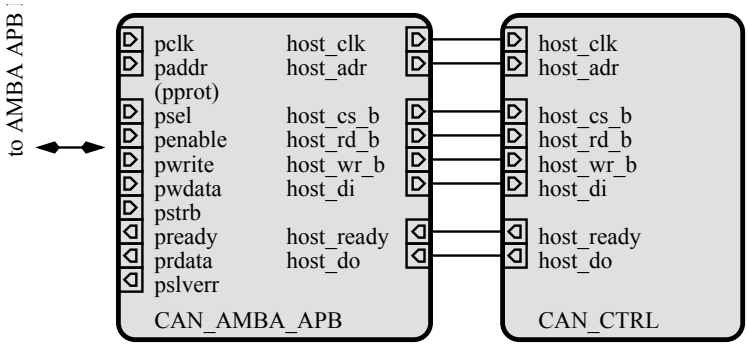

Figure 7-5 AMBA APB Wrapper (only relevant bits shown)

The wrapper provides a 32 bit APB interface and executes both read and write accesses with no wait state with respect to the APB specification. But the definition of what is a wait state in the APB specification is different to what is used in this document. To avoid misunderstandings Figure 7-6 shows APB accesses according to the APB specification. As can be seen, APB requires two cycles of <pclk> for both the write and the read access and calls this “no wait states”.

To avoid any additional cycles this interface wrapper executes the read access when <psel> is set and ignores <penable> resulting in a read access similar to Figure 7-2. In Figure 7-6 the points in time are shown with the red arrows, when the accesses take place.

CAN-CTRL may insert additional wait states if single port memory is used.

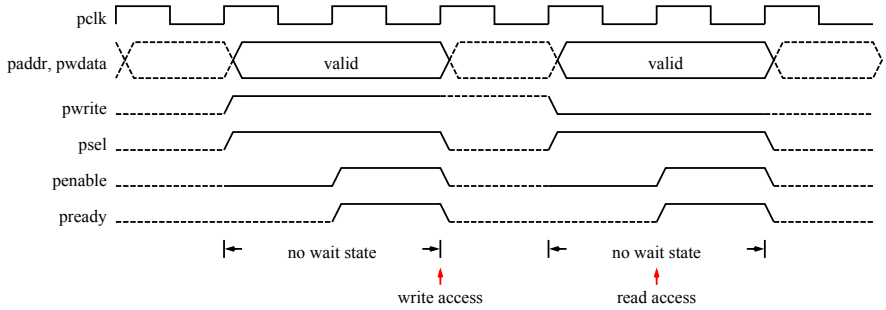

Figure 7-6 AMBA APB Accesses According to the APB Specification

The address <paddr> maps to the addresses of the register map (Table 3-5, chap. 3.6) with signal <host_adr>. As stated in chap. 7.1 the two LSBs are ignored by CAN-CTRL.

If only one byte or some bytes shall be written then signal <pstrb> shall be used to mask the bytes. See chap. 7.1.3 for details. (Signal <host_wr_b> maps to the inverse of <pstrb>.)

Attention: as defined by the APB specification <pstrb> only works with write accesses but not for read accesses! Therefore always 4 bytes are read.

As stated in chap. 7.1.3 a write access to TBUF or ACF has to access a full 32 bit wide word (pstrb=1111b). If not then an error is reported using signal <pslverr> and the write access is not executed.

The APB reset <presetn> can be connected to the asynchronous reset <host_rst_n>.

7.4 AMBA AHB

The memory interface to the CAN core, which has been described in chap. 7.1, can be connected to an AMBA AHB interface using a wrapper. Source code for the wrapper is provided in the release package (compatible to: AMBA 3 AHB-Lite Protocol v1.0 (ARM IHI 0033A)). The wrapper shall be connected to the CAN-CTRL core as shown in Figure 7-7. The wrapper does not use the AHB signals<hburst>,<hprot> and <hmastlock>.

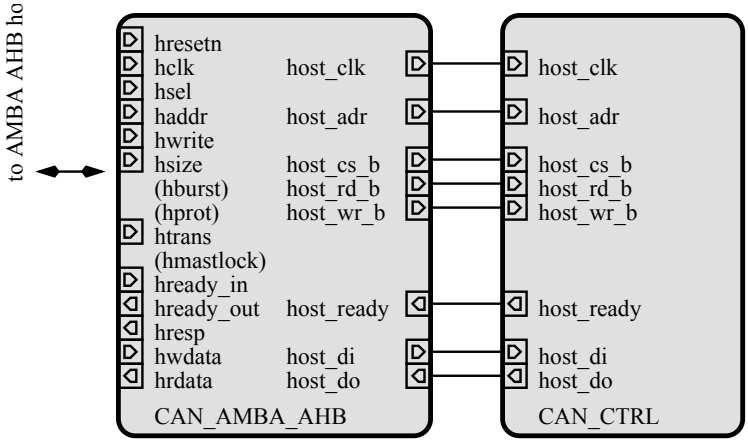

Figure 7-7 AMBA AHB Wrapper (only relevant bits shown)

The wrapper provides a 32 bit AHB interface and executes write accesses with no wait state and read accesses with one wait state. Please note that AHB uses a pipelined access consisting of an address and a data phase. Both phases take at least one <hclk>. The data phase can be extended using wait states. CAN-CTRL may insert additional wait states if single port memory is used.

AHB is designed for high throughput. The required wait state has an impact to the system performance. If this is a problem for the system, then it is suggested to use the APB interface (chap. 7.3) instead of AHB. (The AHB specification suggests to use AHB for fest peripherals and APB for slow peripherals.)

The address <haddr> maps to the addresses of the register map (Table 3-5, chap. 3.6) with signal <host_adr>. As stated in chap. 7.1 the two LSBs are ignored by CAN-CTRL.

AHB requires address alignment if more than one byte is accessed. For halfword accesses haddr[0]=0b and for word accesses haddr[1:0]=00b is required. CAN_AMBA_AHB responds with an error using signal <hresp> if the address alignment is wrong. A correct AHB master will never command such wrong accesses. CAN_AMBA_AHB is a little-endian component. See chap. 7.1.3 and Table 7-2 for details.

Table 7-2 AHB Signals Mapping to CAN-CTRL

hsize	haddr[1:0]	host_wr_b host_rd_b
000b	00b	1110
000b	01b	1101
000b	10b	1011
000b	11b	0111
001b	00b	1100
001b	10b	0011
010b	00b	0000

CAN_AMBA_AHB is designed as 32 bit AHB wrapper and therefore supports byte, halfword or word accesse (hsize = 000b, 001b or 010b). Other values are answered with an error using signal <hresp>.

All accesses can be made in a random way. CAN_AMBA_AHB does not require specific burst accesses and therefore signal <hburst> is not used. Protected accesses (<hprot>) or locked accesses (<hmastlock>) are not supported and therefore these signals are not used too.

Transfer types defined by <htrans> are supported but both types NONSEQ and SEQ will result in single unrelated accesses. CAN_AMBA_AHB is unable to draw any advantage from burst accesses. Types IDLE and BUSY will result in no access. No access will be done also if hsel=0.

Each AHB slave signals hready_out=1 if an access is completed. The AHB decoder collects all <hready_out> signals from all AHB slaves and generates a final <hready> signal. This final <hready> needs to be connected to the AHB master and to the inputs <hready_in> of all AHB slaves.

As stated in chap. 7.1.3 a write access to TBUF or ACF has to access a full 32 bit wide word (hsize=010b). If not then an error is reported using signal <hresp> and the write access is not executed.

The AHB reset <hresetn> can be connected to the asynchronous reset <host_rst_n>.

7.5 Wishbone

The memory interface to the CAN core, which has been described in chap. 7.1, can be connected to a 32 bit wide Wishbone interface using a wrapper. Source code for the wrapper is provided in the release package (compatible to Wishbone specification B.3). The wrapper shall be connected to the CAN-CTRL core as shown in Figure 7-8.

This chapter and Table 7-3 provides the mandatory Wishbone datasheet.

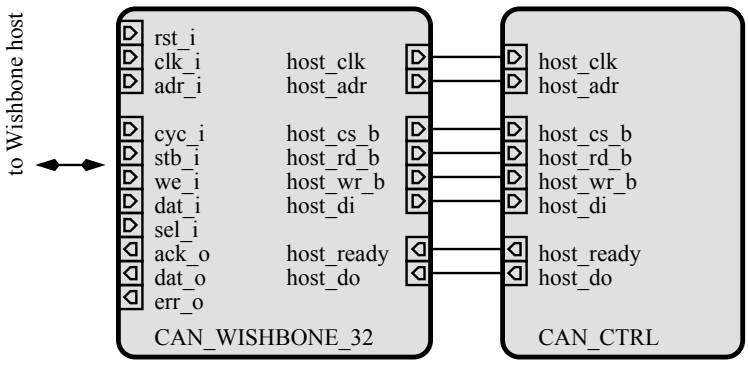

Figure 7-8 Wishbone Wrapper (only relevant bits shown)

Table 7-3 Wishbone datasheet of CAN_WISHBONE_32

Description	Specification
General description	32 bit slave peripheral (CAN protocol machine)
Supported Cycles	Slave, read / write Slave, block read / write Slave, RMW (read, modify, write)
Data port, size Data port, granularity Data port, maximum operand size Data transfer, ordering Data transfer, sequencing	32 bit 8 bit 32 bit Little endian Undefined
Supported signal list and cross reference to equivalent Wishbone signals	rst_i, clk_i, adr_i, cyc_i, stb_i, we_i, sel_i(3:0), dat_i(31:0), dat_o(31:0), ack_o, err_o (All signal names match with the Wishbone signal names.)

The wrapper provides a 32 bit little endian Wishbone interface and executes write accesses with no wait state and read accesses with one wait state. Data alignment is done as described in chap. 7.1.3 .

The address adr_i maps to the addresses of the register map (Table 3-5, chap. 3.6) with signal host_adr. As stated in chap. 7.1 the two LSBs are ignored by CAN-CTRL.

CAN_WISHBONE_32 is designed as 32 bit Wishbone wrapper. Signal sel_i is 4 bits wide and therefore provides byte granularity for accesses. All combinations of the bits of sel_i are supported for accesses. Signal sel_i is only relevant for write accesses. Read accesses are always executed as 32 bit reads.

All accesses can be made in a random way. CAN_WISHBONE_32 is able to execute block accesses and read-modify-write accesses, but there will be no benefit in terms of speed.

As stated in chap. 7.1.3 a write access to TBUF or ACF has to access a full 32 bit wide word (sel_i=1111b). If not then an error is reported using signal err_o and the write access is not executed. The Wishbone master is free to monitor signal err_o or to ignore this signal.

All other signals of CAN-CTRL, that are not shown in Figure 7-8 are outside of the wishbone interface. This includes the asynchronous resets <host_rst_n>, <can_rst_n> and <timer_rst_n>. The Wishbone reset rst_i is a synchronous reset. For the asynchronous resets it is suggested to use reset synchronizers.

8. Functional Description

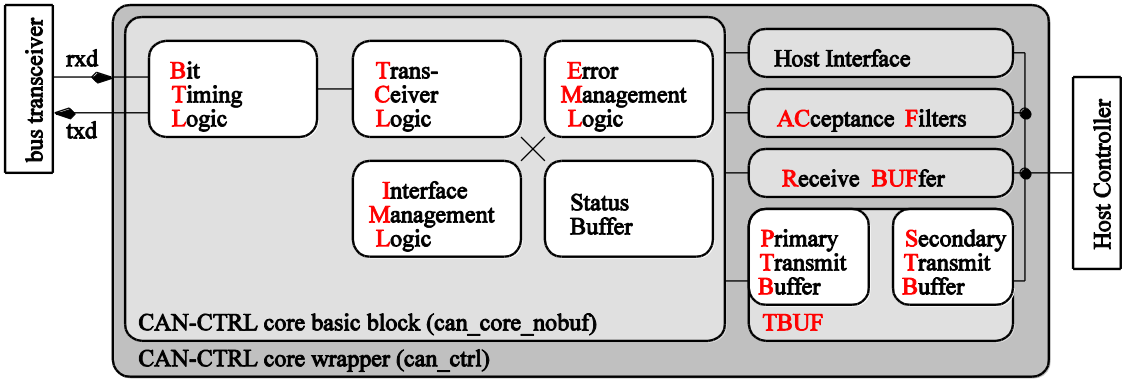

Figure 8-1 Block Diagram

8.1 CAN-CTRL Core Basic Block (can_core_nobuf)

8.1.1 Bit Timing Logic (BTL)

The BTL monitors the bus and synchronizes the internal actions to the bit stream on the CAN bus. Hard synchronization is executed if a “recessive-to-dominant“ bus line transition occurs at the beginning of a message. Re-synchronization takes place on further “recessive-to-dominant“ transitions during the reception of a message. The timing registers (provided by host_interface) control the time segments to compensate propagation delay times and phase shifts and define the sample point.

The BTL also includes the Baud Rate Prescaler. The external system clock will be divided by a programmed value. The resulting period is called time quanta (TQ).

8.1.2 Transceiver Logic (TCL)

The TCL consists of the protocol state machine. It controls all functions for the rreception and transmission of frames:

- checking of received bits and transferring to the receive buffer
- generation of bits to transmit to the bus (ID-, RTR-, IDE-, DLC-, Data-, CRC-, …)
- arbitration
- stuff bit handling
- error handling
- generation of error and overload frames

8.1.3 Interface Management Logic (IML)

The Interface Management Logic controls the behavior of the entire core depending on the commands (register settings in host interface). The IML generates the addresses for the message buffers and provides interrupts and status information to the host controller.

8.1.4 Error Management Logic (EML)

The EML consists of counters for receive and transmit errors. The counters are controlled depending of the corresponding type of errors (Bit Error, Stuff Error, Form Error, CRC Error, and Acknowledgement Error).

8.1.5 Status Buffer

The status buffer stores the header bits of the received messages. This information is needed on-the-fly for the protocol handling and is later stored into the RBUF.

8.2 CAN-CTRL Core Wrapper (can_ctrl)

8.2.1 Host Interface

The host interface includes control and status register bits. This includes the register map (Table 3-5) except for the RBUF, TBUF and ACF.

8.2.2 Receive Buffer (RBUF)

RBUF contains the memory for the receive buffer. The number of RBUF slots is configurable before synthesis by a generic parameter.

8.2.3 Transmit Buffer (TBUF)

The TBUF consists of the Primary Transmit Buffer PTB and the Secondary Transmit Buffer STB. The PTB has a higher priority than the STB. A PTB transmission will suspend an STB transmission and is executed first. The CAN protocol rules require that an STB transmission that is written to the CAN bus must not be interrupted. Therefore, the PTB transmission will start afterwards (after the interframe space).

The number of the STB slots is configurable before synthesis by a generic parameter.

8.2.4 Acceptance Filters (ACF)

The ACF consists of the control logic for acceptance filtering and the filter memory blocks.

The number of the acceptance filters ACF is configurable before synthesis by a generic parameter from 1 to 16 filters.

8.3 Buffer Memories

The CAN-CTRL core uses memories for the RBUF, PTB, STB and ACF. The HDL source code includes a configurable memory model used by all of these components. This memory model supports FPGA Block RAM or Distributed RAM which can be selected using synthesis parameters (chap. 0) individually for each component.

For ASIC synthesis both supported memory models will synthesize to flipflops which is quite area-consuming and inefficient for bigger memory sizes. For this purpose it is suggested to replace the HDL source file containing these memory models (memory.v(hd)) by another one making an instance of the preferred RAM IP core.

9. CAN Multicore

The CAN multicore is an optional feature which can be purchased additionally.

The delivery package may include a container for multiple CAN-CTRL instances: the CAN multicore. This container includes a generic parameter which defines the number of CAN-CTRL instances inside of the container. All CAN-CTRL instances are connected to the same host controller data bus but provide individual interrupt signals.

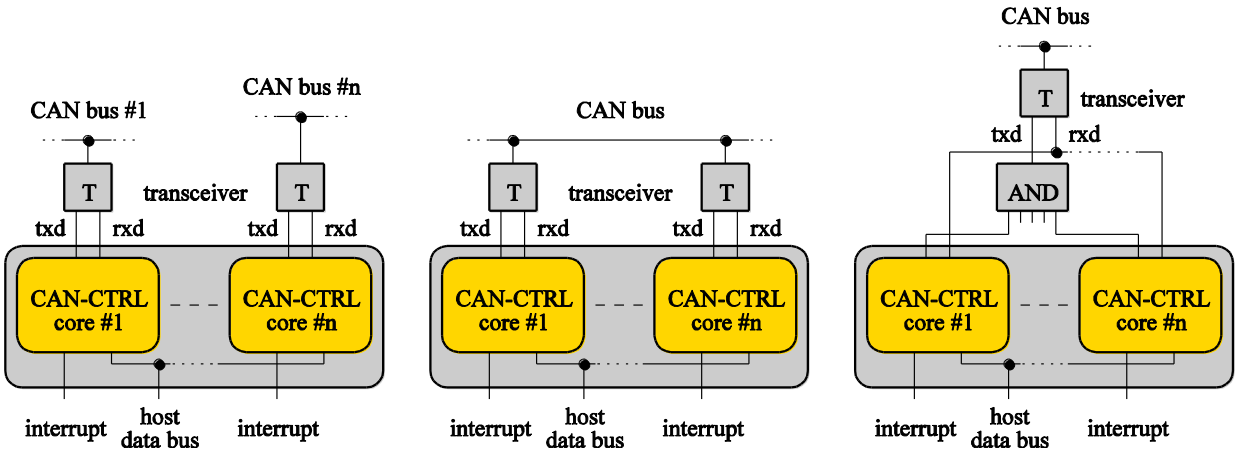

Figure 9-1 CAN Multicore – Connection Options to the CAN Bus

As can be seen in Figure 9-1 each CAN-CTRL instance can be connected to a separate CAN bus. As an alternative some or all of the CAN-CTRL instances can be connected to the same CAN bus. Then it is possible to use individual CAN bus transceivers or to connect the txd and rxd signals as shown in the right example in Figure 9-1.

At runtime all CAN-CTRL instances are fully independent, but for synthesis all of them share the same configuration settings in the <can_package_synparam.v(hd)>. Therefore all instances have the same hardware features if they are synthesized together. If it is required to have individual hardware configurations of the CAN-CTRL instances then it is recommended to do individual synthesis for each instance and connect them in a similar way as the CAN multicore container does it.

10. Support

Every effort has been made to ensure that this core functions correctly. If a problem is encountered please contact CAST:

CAST, Inc.
11 Stonewall Court
Woodcliff Lake, New Jersey 07677 USA

Technical Support Hotline: +1-201-391-8300 ext. 2

Fax: +1 (845) 818-3767
E-mail: support@cast-inc.com
URL: www.cast-inc.com