



# M\_TTCAN

Time-triggered Controller Area Network

## User's Manual

## Revision 3.0.2

14.02.2013



**Robert Bosch GmbH**  
Automotive Electronics

## LEGAL NOTICE

© Copyright 2009-2013 by Robert Bosch GmbH and its licensors. All rights reserved.

"Bosch" is a registered trademark of Robert Bosch GmbH.

The content of this document is subject to continuous developments and improvements. All particulars and its use contained in this document are given by BOSCH in good faith.

**NO WARRANTIES:** TO THE MAXIMUM EXTENT PERMITTED BY LAW, NEITHER THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS AND CONTRIBUTORS, NOR ANY PERSON, EITHER EXPRESSLY OR IMPLICITLY, WARRANTS ANY ASPECT OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, INCLUDING ANY OUTPUT OR RESULTS OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO UNLESS AGREED TO IN WRITING. THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO IS BEING PROVIDED "AS IS", WITHOUT ANY WARRANTY OF ANY TYPE OR NATURE, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND ANY WARRANTY THAT THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO IS FREE FROM DEFECTS.

**ASSUMPTION OF RISK:** THE RISK OF ANY AND ALL LOSS, DAMAGE, OR UNSATISFACTORY PERFORMANCE OF THIS SPECIFICATION (RESPECTIVELY THE PRODUCTS MAKING USE OF IT IN PART OR AS A WHOLE), SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO RESTS WITH YOU AS THE USER. TO THE MAXIMUM EXTENT PERMITTED BY LAW, NEITHER THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS AND CONTRIBUTORS, NOR ANY PERSON EITHER EXPRESSLY OR IMPLICITLY, MAKES ANY REPRESENTATION OR WARRANTY REGARDING THE APPROPRIATENESS OF THE USE, OUTPUT, OR RESULTS OF THE USE OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, BEING CURRENT OR OTHERWISE. NOR DO THEY HAVE ANY OBLIGATION TO CORRECT ERRORS, MAKE CHANGES, SUPPORT THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, DISTRIBUTE UPDATES, OR PROVIDE NOTIFICATION OF ANY ERROR OR DEFECT, KNOWN OR UNKNOWN. IF YOU RELY UPON THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, YOU DO SO AT YOUR OWN RISK, AND YOU ASSUME THE RESPONSIBILITY FOR THE RESULTS. SHOULD THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL LOSSES, INCLUDING, BUT NOT LIMITED TO, ANY NECESSARY SERVICING, REPAIR OR CORRECTION OF ANY PROPERTY INVOLVED TO THE MAXIMUM EXTEND PERMITTED BY LAW.

**DISCLAIMER:** IN NO EVENT, UNLESS REQUIRED BY LAW OR AGREED TO IN WRITING, SHALL THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS OR ANY PERSON BE LIABLE FOR ANY LOSS, EXPENSE OR DAMAGE, OF ANY TYPE OR NATURE ARISING OUT OF THE USE OF, OR INABILITY TO USE THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM

RELATED THERETO, INCLUDING, BUT NOT LIMITED TO, CLAIMS, SUITS OR CAUSES OF ACTION INVOLVING ALLEGED INFRINGEMENT OF COPYRIGHTS, PATENTS, TRADEMARKS, TRADE SECRETS, OR UNFAIR COMPETITION.

**INDEMNIFICATION:** TO THE MAXIMUM EXTEND PERMITTED BY LAW YOU AGREE TO INDEMNIFY AND HOLD HARMLESS THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS AND CONTRIBUTORS, AND EMPLOYEES, AND ANY PERSON FROM AND AGAINST ALL CLAIMS, LIABILITIES, LOSSES, CAUSES OF ACTION, DAMAGES, JUDGMENTS, AND EXPENSES, INCLUDING THE REASONABLE COST OF ATTORNEYS' FEES AND COURT COSTS, FOR INJURIES OR DAMAGES TO THE PERSON OR PROPERTY OF THIRD PARTIES, INCLUDING, WITHOUT LIMITATIONS, CONSEQUENTIAL, DIRECT AND INDIRECT DAMAGES AND ANY ECONOMIC LOSSES, THAT ARISE OUT OF OR IN CONNECTION WITH YOUR USE, MODIFICATION, OR DISTRIBUTION OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, ITS OUTPUT, OR ANY ACCOMPANYING DOCUMENTATION.

**GOVERNING LAW:** THE RELATIONSHIP BETWEEN YOU AND ROBERT BOSCH GMBH SHALL BE GOVERNED SOLELY BY THE LAWS OF THE FEDERAL REPUBLIC OF GERMANY. THE STIPULATIONS OF INTERNATIONAL CONVENTIONS REGARDING THE INTERNATIONAL SALE OF GOODS SHALL NOT BE APPLICABLE. THE EXCLUSIVE LEGAL VENUE SHALL BE DUESSELDORF, GERMANY.

MANDATORY LAW SHALL BE UNAFFECTED BY THE FOREGOING PARAGRAPHS.

**INTELLECTUAL PROPERTY OWNERS/COPYRIGHT OWNERS/CONTRIBUTORS:** ROBERT BOSCH GMBH, ROBERT BOSCH PLATZ 1, 70839 GERLINGEN, GERMANY AND ITS LICENSORS.



**SPECIFICATION REVISION HISTORY**

<b>REVISION</b>	<b>DATE</b>	<b>NOTES</b>
0.1	23.02.2009	initial working revision
0.2	01.04.2009	first revised working revision
0.3	24.08.2009	second revised working revision
1.0	27.09.2010	first complete revision
1.01	20.12.2010	minor textual enhancements
1.02	18.02.2011	minor textual enhancements
2.0	12.03.2012	debug on CAN, dedicated Rx Buffers, CAN FD, Extension IF
2.0.1	23.05.2012	Section 3.1.3 CAN FD Operation corrected
3.0	17.10.2012	FIFO overwrite mode, transmit pause, support of CAN FD 64-byte frames
3.0.1	21.12.2012	registers FBTP and TEST updated, minor textual enhancements
3.0.2	14.02.2013	Section 2.4.1 Message RAM Configuration corrected, minor corrections/enhancements

**TRACKING OF MAJOR CHANGES**

## TERMS AND ABBREVIATIONS

This document uses the following terms and abbreviations.

Term	Meaning
BRP	Baud Rate Prescaler
BSP	Bit Stream Processor
BTL	Bit Timing Logic
CAN	Controller Area Network
CAN FD	Controller Area Network with Flexible Data-rate
CRC	Cyclic Redundancy Check
DLC	Data Length Code
ECC	Error Correction Code
ECU	Electronic Control Unit
EML	Error Management Logic
FSE	Frame Synchronization Entity
FSM	Finite State Machine
MRM	Master Reference Mark
MSC	Message Status Count
NTU	Network Time Unit
tq	time quantum
TTCAN	Time-Triggered CAN
TUR	Time Unit Ratio

## CONVENTIONS

The following conventions are used within this User's Manual.

<b>Arial bold</b>	Names of bits and ports
<i>Arial italic</i>	States of bits and ports
Reserved	Configuration values marked as reserved are illegal; they are interpreted generally same as reset values.

**REFERENCES**

This document refers to the following documents:

<b>Ref</b>	<b>Author(s)</b>	<b>Title</b>
[1]	ISO	ISO 11898-1: CAN data link layer and physical signalling
[2]	ISO	ISO 11898-4: Time-triggered communication
[3]	AE/EIN	M_(TT)CAN System Integration Guide
[4]	AE/EIN	M_TTCAN Module Integration Guide
[5]	AE/EIN	CAN FD Protocol Specification



# Table of contents

1	Overview . . . . .	1
1.1	Features . . . . .	1
1.2	Block Diagram . . . . .	2
1.3	Dual Clock Sources . . . . .	3
1.4	Dual Interrupt Lines . . . . .	4
2	Programmer's Model . . . . .	5
2.1	Hardware Reset Description. . . . .	5
2.2	Register Map . . . . .	5
2.3	Registers. . . . .	7
2.3.1	Customer Register . . . . .	7
2.3.2	Core Release Register (CREL) . . . . .	7
2.3.3	Endian Register (ENDN) . . . . .	8
2.3.4	Fast Bit Timing & Prescaler Register (FBTP) . . . . .	8
2.3.5	Test Register (TEST) . . . . .	9
2.3.6	RAM Watchdog (RWD) . . . . .	11
2.3.7	CC Control Register (CCCR) . . . . .	12
2.3.8	Bit Timing & Prescaler Register (BTP) . . . . .	14
2.3.9	Timestamp Counter Configuration (TSCC) . . . . .	15
2.3.10	Timestamp Counter Value (TSCV) . . . . .	15
2.3.11	Timeout Counter Configuration (TOCC) . . . . .	16
2.3.12	Timeout Counter Value (TOCV) . . . . .	16
2.3.13	Error Counter Register (ECR) . . . . .	17
2.3.14	Protocol Status Register (PSR) . . . . .	18
2.3.15	Interrupt Register (IR) . . . . .	20
2.3.16	Interrupt Enable (IE) . . . . .	23
2.3.17	Interrupt Line Select (ILS) . . . . .	25
2.3.18	Interrupt Line Enable (ILE) . . . . .	26
2.3.19	Global Filter Configuration (GFC) . . . . .	27
2.3.20	Standard ID Filter Configuration (SIDFC) . . . . .	28
2.3.21	Extended ID Filter Configuration (XIDFC) . . . . .	28
2.3.22	Extended ID AND Mask (XIDAM) . . . . .	29
2.3.23	High Priority Message Status (HPMS) . . . . .	29
2.3.24	New Data 1 (NDAT1) . . . . .	30
2.3.25	New Data 2 (NDAT2) . . . . .	30
2.3.26	Rx FIFO 0 Configuration (RXF0C) . . . . .	31
2.3.27	Rx FIFO 0 Status (RXF0S) . . . . .	32
2.3.28	Rx FIFO 0 Acknowledge (RXF0A) . . . . .	33
2.3.29	Rx Buffer Configuration (RXBC) . . . . .	33
2.3.30	Rx FIFO 1 Configuration (RXF1C) . . . . .	34
2.3.31	Rx FIFO 1 Status (RXF1S) . . . . .	34
2.3.32	Rx FIFO 1 Acknowledge (RXF1A) . . . . .	35
2.3.33	Rx Buffer / FIFO Element Size Configuration (RXESC) . . . . .	36
2.3.34	Tx Buffer Configuration (TXBC) . . . . .	37
2.3.35	Tx FIFO/Queue Status (TXFQS) . . . . .	38
2.3.36	Tx Buffer Element Size Configuration (TXESC) . . . . .	39
2.3.37	Tx Buffer Request Pending (TXBRP) . . . . .	40
2.3.38	Tx Buffer Add Request (TXBAR) . . . . .	41
2.3.39	Tx Buffer Cancellation Request (TXBCR) . . . . .	41
2.3.40	Tx Buffer Transmission Occurred (TXBTO) . . . . .	42
2.3.41	Tx Buffer Cancellation Finished (TXBCF) . . . . .	42
2.3.42	Tx Buffer Transmission Interrupt Enable (TXBTIE) . . . . .	43
2.3.43	Tx Buffer Cancellation Finished Interrupt Enable (TXBCIE) . . . . .	43
2.3.44	Tx Event FIFO Configuration (TXEFC) . . . . .	44
2.3.45	Tx Event FIFO Status (TXEFS) . . . . .	45
2.3.46	Tx Event FIFO Acknowledge (TXEFA) . . . . .	45

2.3.47	TT Trigger Memory Configuration (TTTMC) . . . . .	46
2.3.48	TT Reference Message Configuration (TTRMC) . . . . .	46
2.3.49	TT Operation Configuration (TTOCF) . . . . .	47
2.3.50	TT Matrix Limits (TTMLM) . . . . .	48
2.3.51	TUR Configuration (TURCF) . . . . .	49
2.3.52	TT Operation Control (TTOCN) . . . . .	50
2.3.53	TT Global Time Preset (TTGTP) . . . . .	52
2.3.54	TT Time Mark (TTTMK) . . . . .	53
2.3.55	TT Interrupt Register (TTIR) . . . . .	54
2.3.56	TT Interrupt Enable (TTIE) . . . . .	56
2.3.57	TT Interrupt Line Select (TTILS) . . . . .	57
2.3.58	TT Operation Status (TTOST) . . . . .	58
2.3.59	TUR Numerator Actual (TURNA) . . . . .	60
2.3.60	TT Local & Global Time (TTLGT) . . . . .	60
2.3.61	TT Cycle Time & Count (TTCTC) . . . . .	61
2.3.62	TT Capture Time (TTCPT) . . . . .	61
2.3.63	TT Cycle Sync Mark (TTCSM) . . . . .	62
2.4	Message RAM . . . . .	63
2.4.1	Message RAM Configuration . . . . .	63
2.4.2	Rx Buffer and FIFO Element . . . . .	64
2.4.3	Tx Buffer Element . . . . .	66
2.4.4	Tx Event FIFO Element . . . . .	68
2.4.5	Standard Message ID Filter Element . . . . .	69
2.4.6	Extended Message ID Filter Element . . . . .	70
2.4.7	Trigger Memory Element . . . . .	72
3	Functional Description . . . . .	75
3.1	Operating Modes . . . . .	75
3.1.1	Software Initialization . . . . .	75
3.1.2	Normal Operation . . . . .	76
3.1.3	CAN FD Operation . . . . .	76
3.1.4	Transceiver Delay Compensation . . . . .	77
3.1.5	Restricted Operation Mode . . . . .	78
3.1.6	Bus Monitoring Mode . . . . .	79
3.1.7	Disabled Automatic Retransmission . . . . .	79
3.1.8	Power Down (Sleep Mode) . . . . .	80
3.1.9	Test Modes . . . . .	80
3.1.10	Application Watchdog . . . . .	81
3.2	Timestamp Generation . . . . .	81
3.3	Timeout Counter . . . . .	81
3.4	Rx Handling . . . . .	82
3.4.1	Acceptance Filtering . . . . .	82
3.4.2	Rx FIFOs . . . . .	86
3.4.3	Dedicated Rx Buffers . . . . .	88
3.4.4	Debug on CAN Support . . . . .	89
3.5	Tx Handling . . . . .	91
3.5.1	Transmit Pause . . . . .	91
3.5.2	Dedicated Tx Buffers . . . . .	91
3.5.3	Tx FIFO . . . . .	92
3.5.4	Tx Queue . . . . .	93
3.5.5	Mixed Dedicated Tx Buffers / Tx FIFO . . . . .	93
3.5.6	Mixed Dedicated Tx Buffers / Tx Queue . . . . .	94
3.5.7	Transmit Cancellation . . . . .	94
3.5.8	Tx Event Handling . . . . .	95
3.6	FIFO Acknowledge Handling . . . . .	95
4	TTCAN Operation . . . . .	97
4.1	Reference Message . . . . .	97
4.1.1	Level 1 . . . . .	97
4.1.2	Level 2 . . . . .	98
4.1.3	Level 0 . . . . .	98

4.2	TTCAN Configuration . . . . .	99
4.2.1	TTCAN Timing . . . . .	99
4.2.2	Message Scheduling . . . . .	100
4.2.3	Trigger Memory . . . . .	101
4.2.4	TTCAN Schedule Initialization . . . . .	105
4.3	TTCAN Gap Control . . . . .	106
4.4	Stop Watch . . . . .	108
4.5	Local Time, Cycle Time, Global Time, and External Clock Synchronization . . . . .	108
4.6	TTCAN Error Level . . . . .	110
4.7	TTCAN Message Handling . . . . .	111
4.7.1	Reference Message . . . . .	111
4.7.2	Message Reception . . . . .	111
4.7.3	Message Transmission . . . . .	112
4.8	TTCAN Interrupt and Error Handling . . . . .	114
4.9	Level 0 . . . . .	115
4.9.1	Synchronizing . . . . .	116
4.9.2	Handling of Error Levels . . . . .	116
4.9.3	Master Slave Relation . . . . .	117
4.10	Asynchronous Serial Communication . . . . .	118
4.11	Synchronization to external Time Schedule . . . . .	119
5	Appendix . . . . .	121
5.1	Register Bit Overview . . . . .	121
5.2	Module Interface . . . . .	127
5.3	Connection to external Message RAM . . . . .	129
5.4	Interface to DMA Controller . . . . .	129



# Chapter 1.

## 1. Overview

The M\_TTCAN module is the new TTCAN Communication Controller IP-module that can be integrated as stand-alone device or as part of an ASIC. It is described in VHDL on RTL level, prepared for synthesis. The M\_TTCAN performs communication according to ISO 11898-1 (identical to Bosch CAN protocol specification 2.0 part A,B) and according to ISO 11898-4 (Time-triggered communication on CAN). The M\_TTCAN provides all features of time-triggered communication specified in ISO 11898-4, including event synchronized time-triggered communication, global system time, and clock drift compensation. In addition the M\_TTCAN supports communication according to CAN FD protocol specification 1.0. The CAN FD option can be used together with event-triggered and time-triggered CAN communication.

The message storage is intended to be a single- or dual-ported Message RAM outside of the module. It is connected to the M\_TTCAN via the Generic Master Interface. Depending on the chosen ASIC integration, multiple M\_TTCAN controllers can share the same Message RAM.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN Core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN Core as well as providing transmit status information. It implements all functions concerning the time schedule and the global system time.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M\_TTCAN can be connected to a wide range of Host CPUs via its 8/16/32-bit Generic Slave Interface. The M\_TTCAN's clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FM-PLL.

### 1.1 Features

- Conform with CAN protocol version 2.0 part A, B and ISO 11898-1, -4
- CAN FD with up to 64 data bytes supported
- TTCAN protocol level 1 and level 2 completely in hardware
- Event synchronized time-triggered communication supported
- CAN Error Logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signalling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Multiple M\_TTCANs may share the same Message RAM
- Programmable loop-back test mode

- Maskable module interrupts
- 8/16/32 bit Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support
- Debug on CAN support

## 1.2 Block Diagram

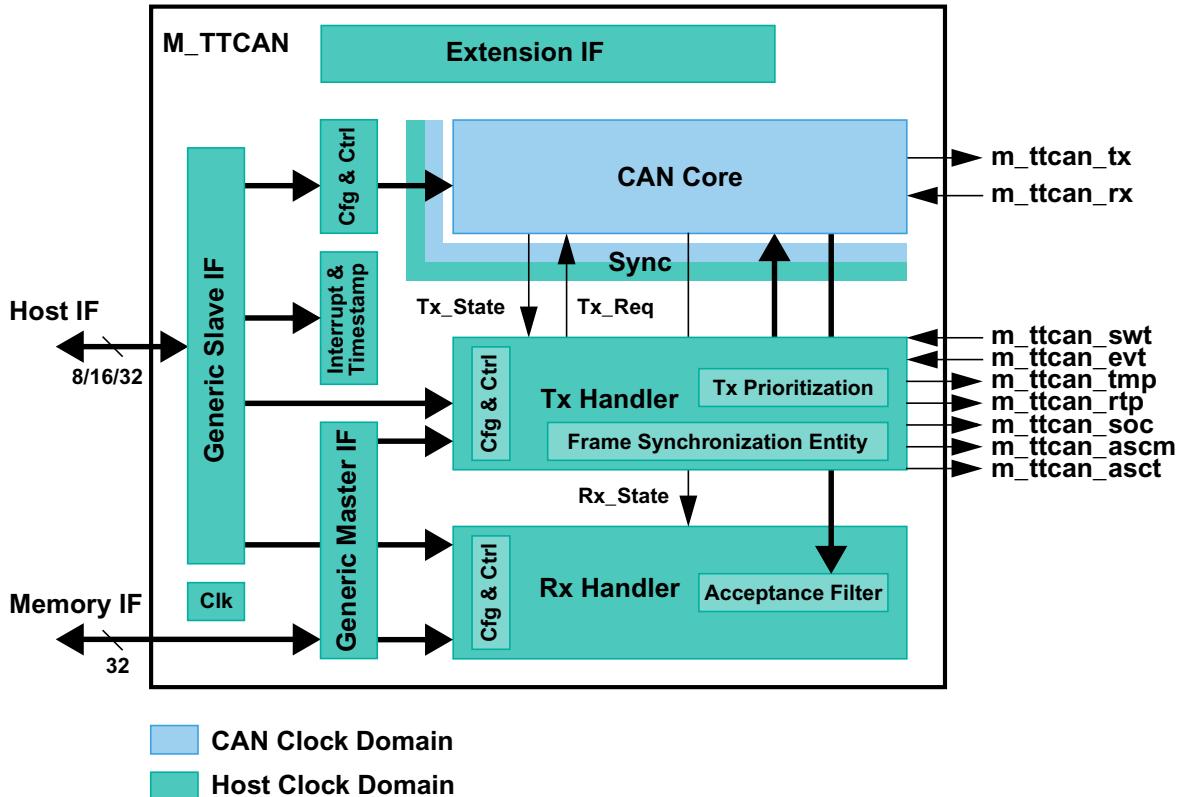


Figure 1 M\_TTCAN Block Diagram

### CAN Core

CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.

### Sync

Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.

### Clk

Synchronizes reset signal to the Host clock domain and to the CAN clock domain.

### Cfg & Ctrl

CAN Core related configuration and control bits.

### **Interrupt & Timestamp**

Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation. An externally generated 16-bit vector may substitute the integrated 16-bit CAN bit time counter for receive and transmit timestamp generation.

### **Tx Handler**

Controls the message transfer from the external Message RAM to the CAN Core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.

The Tx Handler also implements the Frame Synchronization Entity FSE which controls time-triggered communication according to ISO11898-4. It synchronizes itself to the reference messages on the CAN bus, controls cycle time and global time, and handles transmissions according to the predefined message schedule, the system matrix. It also handles the time marks of the system matrix that are linked to the messages in the Message RAM. Stop Watch Trigger, Event Trigger, and Time Mark Interrupt are synchronization interfaces.

### **Rx Handler**

Controls the transfer of received messages from the CAN Core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

### **Generic Slave Interface**

Connects the M\_TTCAN to a customer specific Host CPU. The Generic Slave Interface is capable to connect to an 8/16/32-bit bus to support a wide range of interconnection structures.

### **Generic Master Interface**

Connects the M\_TTCAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16K • 32 bit.

### **Extension Interface**

All flags from the Interrupt Register IR and TT Interrupt Register TTIR as well as selected internal status and control signals are routed to this interface. The interface is intended for connection of the M\_TTCAN to a module-external interrupt unit or to other module-external components. The connection of these signals is optional.

## **1.3 Dual Clock Sources**

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain **m\_ttcn\_hclk**. Due to the high precision clocking requirements of the CAN Core, a separate clock without any modulation has to be provided as **m\_ttcn\_cclk**. The CAN Core should be programmed to have at least 8 clocks per bittime, this is e.g. 1 Mbaud @ **m\_ttcn\_cclk**  $\geq$  8 MHz. Even in case of a very high Host clock frequency, the clock frequency of the CAN Core needs not to be higher than 8 MHz.

Within the M\_TTCAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

**Note:** *In order to achieve a stable function of the M\_TTCAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.*

## 1.4 Dual Interrupt Lines

The module provides two interrupt lines. Interrupts can be routed either to **m\_ttcan\_int0** or to **m\_ttcan\_int1**. By default all interrupts are routed to interrupt line **m\_ttcan\_int0**. By programming **ILE.EINT0** and **ILE.EINT1** the interrupt lines can be enabled or disabled separately.

# Chapter 2.

## 2. Programmer's Model

### 2.1 Hardware Reset Description

After hardware reset, the registers of the M\_TTCAN hold the reset values listed in Table 1. Additionally the *Bus\_Off* state is reset and the output **m\_ttcanc\_tx** is set to recessive (HIGH). The value 0x0001 (**CCCR.INIT** = '1') in the CC Control Register enables software initialization. The M\_TTCAN does not influence the CAN bus until the CPU resets **CCCR.INIT** to '0'.

### 2.2 Register Map

The M\_TTCAN module allocates an address space of 512 bytes. All registers are organized as 32-bit registers. The M\_TTCAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8 bit (byte access), 16 bit (half-word access), or 32 bit (word access). Write access by the Host CPU to registers/bits marked with "P=Protected Write" is possible only with **CCCR.CCE**='1' AND **CCCR.INIT**='1'. There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

ADDRESS	SYMBOL	NAME	PAGE	RESET	ACC
0x000	CREL	Core Release Register	7	rrrd dddd	R
0x004	ENDN	Endian Register	8	8765 4321	R
0x008	CUST	Customer Register	7	t.b.d.	t.b.d.
0x00C	FBTP	Fast Bit Timing & Prescaler Register	8	0000 0A33	RP
0x010	TEST	Test Register	9	0000 0000	RP
0x014	RWD	RAM Watchdog	11	0000 0000	RP
0x018	CCCR	CC Control Register	12	0000 0001	RWPp
0x01C	BTP	Bit Timing & Prescaler Register	14	0000 0A33	RP
0x020	TSCC	Timestamp Counter Configuration	15	0000 0000	RP
0x024	TSCV	Timestamp Counter Value	15	0000 0000	RC
0x028	TOCC	Timeout Counter Configuration	16	FFFF 0000	RP
0x02C	TOCV	Timeout Counter Value	16	0000 FFFF	RC
0x030-03C		reserved (4)		0000 0000	R
0x040	ECR	Error Counter Register	17	0000 0000	RX
0x044	PSR	Protocol Status Register	18	0000 0707	RXS
0x048-04C		reserved (2)		0000 0000	R
0x050	IR	Interrupt Register	20	0000 0000	RW
0x054	IE	Interrupt Enable	23	0000 0000	RW
0x058	ILS	Interrupt Line Select	25	0000 0000	RW
0x05C	ILE	Interrupt Line Enable	26	0000 0000	RW
0x060-07C		reserved (8)		0000 0000	R
0x080	GFC	Global Filter Configuration	27	0000 0000	RP
0x084	SIDFC	Standard ID Filter Configuration	28	0000 0000	RP
0x088	XIDFC	Extended ID Filter Configuration	28	0000 0000	RP
0x08C		reserved (1)		0000 0000	R
0x090	XIDAM	Extended ID AND Mask	29	1FFF FFFF	RP
0x094	HPMS	High Priority Message Status	29	0000 0000	R

Table 1 M\_TTCAN Register Map

ADDRESS	SYMBOL	NAME	PAGE	RESET	ACC
0x098	NDAT1	New Data 1	30	0000 0000	RW
0x09C	NDAT2	New Data 2	30	0000 0000	RW
0x0A0	RXF0C	Rx FIFO 0 Configuration	31	0000 0000	RP
0x0A4	RXF0S	Rx FIFO 0 Status	32	0000 0000	R
0x0A8	RXF0A	Rx FIFO 0 Acknowledge	33	0000 0000	RW
0x0AC	RXBC	Rx Buffer Configuration	33	0000 0000	RP
0x0B0	RXF1C	Rx FIFO 1 Configuration	34	0000 0000	RP
0x0B4	RXF1S	Rx FIFO 1 Status	34	0000 0000	R
0x0B8	RXF1A	Rx FIFO 1 Acknowledge	35	0000 0000	RW
0x0BC	RXESC	Rx Buffer / FIFO Element Size Configuration	36	0000 0000	RP
0x0C0	TXBC	Tx Buffer Configuration	37	0000 0000	RP
0x0C4	TXFQS	Tx FIFO/Queue Status	38	0000 0000	R
0x0C8	TXESC	Tx Buffer Element Size Configuration	39	0000 0000	RP
0x0CC	TXBRP	Tx Buffer Request Pending	40	0000 0000	R
0x0D0	TXBAR	Tx Buffer Add Request	41	0000 0000	RW
0x0D4	TXBCR	Tx Buffer Cancellation Request	41	0000 0000	RW
0x0D8	TXBTO	Tx Buffer Transmission Occurred	42	0000 0000	R
0x0DC	TXBCF	Tx Buffer Cancellation Finished	42	0000 0000	R
0x0E0	TXBTIE	Tx Buffer Transmission Interrupt Enable	43	0000 0000	RW
0x0E4	TXBCIE	Tx Buffer Cancellation Finished Interrupt Enable	43	0000 0000	RW
0x0E8-0EC		reserved (2)		0000 0000	R
0x0F0	TXEFC	Tx Event FIFO Configuration	44	0000 0000	RP
0x0F4	TXEFS	Tx Event FIFO Status	45	0000 0000	R
0x0F8	TXEFA	Tx Event FIFO Acknowledge	45	0000 0000	RW
0x0FC		reserved (1)		0000 0000	R
0x100	TTTMC	TT Trigger Memory Configuration	46	0000 0000	RP
0x104	TTRMC	TT Reference Message Configuration	46	0000 0000	RP
0x108	TTOCF	TT Operation Configuration	47	0001 0000	RP
0x10C	TTMLM	TT Matrix Limits	48	0000 0000	RP
0x110	TURCF	TUR Configuration	49	1000 0000	RP
0x114	TTOCN	TT Operation Control	50	0000 0000	RW
0x118	TTGTP	TT Global Time Preset	52	0000 0000	RW
0x11C	TTTMK	TT Time Mark	53	0000 0000	RW
0x120	TTIR	TT Interrupt Register	54	0000 0000	RW
0x124	TTIE	TT Interrupt Enable	56	0000 0000	RW
0x128	TTILS	TT Interrupt Line Select	57	0000 0000	RW
0x12C	TTOST	TT Operation Status	58	0000 0080	R
0x130	TURNA	TUR Numerator Actual	60	0001 0000	R
0x134	TTLGT	TT Local & Global Time	60	0000 0000	R
0x138	TTCTC	TT Cycle Time & Count	61	003F 0000	R
0x13C	TTCPT	TT Capture Time	61	0000 0000	R
0x140	TTCSM	TT Cycle Sync Mark	62	0000 0000	R
0x144-1FC		reserved (47)		0000 0000	R

R = Read, S = Set on read, X = Reset on read, W = Write, P = Protected write, p = Protected set, C = Clear/preset on write, r = release, d = date

Table 1 M\_TTCAN Register Map

## 2.3 Registers

### 2.3.1 Customer Register

Address 0x08 is reserved for an optional 32 bit customer-specific register. The Customer Register is intended to hold customer-specific configuration, control, and status bits. A description of the functionality is not part of this document.

### 2.3.2 Core Release Register (CREL)

<i>Bits</i>	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x000		REL[3:0]		STEP[3:0]		SUBSTEP[3:0]		YEAR[3:0]								
		R-r		R-r		R-r		R-r								R-d
<i>Bits</i>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		MON[7:0]											DAY[7:0]			R-d
		R-d														

R = Read; -r = release, -d = time stamp, value defined at synthesis by generic parameter

Table 2 Core Release Register (addresses 0x000)

#### Bits 31:28 REL[3:0]: Core Release

One digit, BCD-coded.

#### Bits 27:24 STEP[3:0]: Step of Core Release

One digit, BCD-coded.

#### Bits 23:20 SUBSTEP[3:0]: Sub-step of Core Release

One digit, BCD-coded.

#### Bits 19:16 YEAR[3:0]: Time Stamp Year

One digit, BCD-coded. This field is set by generic parameter on M\_TTCAN synthesis.

#### Bits 15:8 MON[7:0]: Time Stamp Month

Two digits, BCD-coded. This field is set by generic parameter on M\_TTCAN synthesis.

#### Bits 7:0 DAY[7:0]: Time Stamp Day

Two digits, BCD-coded. This field is set by generic parameter on M\_TTCAN synthesis.

<i>Release</i>	<i>Step</i>	<i>SubStep</i>	<i>Year</i>	<i>Month</i>	<i>Day</i>	<i>Name</i>
0	1	0	0	03	10	Revision 0.1.0, Date 2010/03/10

Table 3 Example for Coding of Revisions

### 2.3.3 Endian Register (ENDN)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x004	ETV[31:16]															
	R-0x8765															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ETV[15:0]															
	R-0x4321															

R = Read; -t = test value

Table 4 Endian Register (address 0x004)

#### Bits 31:0 ETV[31:0]: Endianness Test Value

The endianness test value is 0x87654321.

### 2.3.4 Fast Bit Timing & Prescaler Register (FBTP)

This register is only writable if bits **CCCR.CCE** and **CCCR.INIT** are set. The CAN bit time may be programmed in the range of 4 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 **m\_ttcanc\_cclk** periods.  $t_q = (\text{FBRP} + 1) \text{ m_ttcanc_cclk}$  period.

**FTSEG1** is the sum of Prop\_Seg and Phase\_Seg1. **FTSEG2** is Phase\_Seg2.

Therefore the length of the bit time is (programmed values) [**FTSEG1** + **FTSEG2** + 3]  $t_q$  or (functional values) [Sync\_Seg + Prop\_Seg + Phase\_Seg1 + Phase\_Seg2]  $t_q$ .

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0C	res		TDCO[4:0]				TDC		res		FBRP[4:0]					
	R-0		RP-0				RP-0		R-0		RP-0x00					
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res		FTSEG1[3:0]				res		FTSEG2[2:0]		res		FSJW[1:0]		RP-3	
	R-0		RP-0xA				R-0		RP-0x3		R-0		RP-3			

R = Read, P = Protected write; -n = value after reset

Table 5 Fast Bit Timing & Prescaler Register (address 0x0C)

#### Bits 28:24 TDCO[4:0]: Transceiver Delay Compensation Offset

0x00-0x1F Offset value defining the distance between the measured delay from **m\_ttcanc\_tx** to **m\_ttcanc\_rx** and the secondary sample point. Valid values are 0 to 31 **m\_ttcanc\_cclk** periods.

#### Bit 23 TDC: Transceiver Delay Compensation

0= Transceiver Delay Compensation disabled

1= Transceiver Delay Compensation enabled

**Bits 20:16 FBRP[4:0]:** Fast Baud Rate Prescaler

0x00-0x1F The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

**Bits 11:8 FTSEG1[3:0]:** Fast time segment before sample point

0x1-0xF Valid values are 1 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

**Bits 6:4 FTSEG2[2:0]:** Fast time segment after sample point

0x0-0x7 Valid values are 0 to 7. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

**Bits 1:0 FSJW[1:0]:** Fast (Re) Synchronization Jump Width

0x0-0x3 Valid values are 0 to 3. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

**Note:** With a CAN clock ( $m_{ttcan\_cclk}$ ) of 8 MHz, the reset value of 0x00000A33 configures the M\_TTCAN for a fast bit rate of 500 kBit/s.

**Note:** The bit rate configured for the CAN FD data phase via FBTP must be higher or equal to the bit rate configured for the arbitration phase via BTP.

**2.3.5 Test Register (TEST)**

Write access to the Test Register has to be enabled by setting bit CCCR.TEST to '1'. All Test Register functions are set to their reset values when bit CCCR.TEST is reset.

Loop Back Mode and software control of pin  $m_{ttcan\_tx}$  are hardware test modes. Programming of  $TX \neq "00"$  may disturb the message transfer on the CAN bus.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x10																

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

R = Read, P = Protected write, -U = undefined; -n = value after reset

**Table 6 Test Register (address 0x10)**

**Bits 13:8 TDCV[5:0]:** Transceiver Delay Compensation Value

0x00-0x3F Position of the secondary sample point, defined by the sum of the measured delay from  $m_{ttcan\_tx}$  to  $m_{ttcan\_rx}$  and FBTP.TDCO. Valid values are 0 to 63  $m_{ttcan\_cclk}$  periods.

**Bit 7 RX:** Receive Pin

Monitors the actual value of pin  $m_{ttcan\_rx}$

- 0= The CAN bus is dominant ( $m_{ttcan\_rx} = '0'$ )
- 1= The CAN bus is recessive ( $m_{ttcan\_rx} = '1'$ )

**Bits 6:5 TX[1:0]: Control of Transmit Pin**

- 00 Reset value, **m\_ttcanc\_tx** controlled by the CAN Core, updated at the end of the CAN bit time
- 01 Sample Point can be monitored at pin **m\_ttcanc\_tx**
- 10 Dominant ('0') level at pin **m\_ttcanc\_tx**
- 11 Recessive ('1') at pin **m\_ttcanc\_tx**

**Bit 4 LBCK: Loop Back Mode**

- 0= Reset value, Loop Back Mode is disabled
- 1= Loop Back Mode is enabled (see Section 3.1.9, *Test Modes*)

**Bit 3 CAT: Check ASC Transmit Control**

Monitors level at output pin **m\_ttcanc\_asct**.

- 0= Output pin **m\_ttcanc\_asct** = '0'
- 1= Output pin **m\_ttcanc\_asct** = '1'

**Bit 2 CAM: Check ASC Multiplexer Control**

Monitors level at output pin **m\_ttcanc\_ascm**.

- 0= Output pin **m\_ttcanc\_ascm** = '0'
- 1= Output pin **m\_ttcanc\_ascm** = '1'

**Bit 1 TAT: Test ASC Transmit Control**

Controls output pin **m\_ttcanc\_asct** in test mode, ORed with the signal from the FSE

- 0= Level at pin **m\_ttcanc\_asct** controlled by FSE
- 1= Level at pin **m\_ttcanc\_asct** = '1'

**Bit 0 TAM: Test ASC Multiplexer Control**

Controls output pin **m\_ttcanc\_ascm** in test mode, ORed with the signal from the FSE

- 0= Level at pin **m\_ttcanc\_ascm** controlled by FSE
- 1= Level at pin **m\_ttcanc\_ascm** = '1'

### 2.3.6 RAM Watchdog (RWD)

The RAM Watchdog monitors the READY output of the Message RAM (**m\_ttcan\_aeim\_ready**). A Message RAM access via the M\_TTCAN's Generic Master Interface (**m\_ttcan\_aeim\_sel** active) starts the Message RAM Watchdog Counter with the value configured by **RWD.WDC**. The counter is reloaded with **RWD.WDC** when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag **IR.WDI** is set. The RAM Watchdog Counter is clocked by the Host clock (**m\_ttcan\_hclk**).

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x14																
																res
																R-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																WDC[7:0]
																R-0
																RP-0

R = Read, W = Write, P = Protected write; -n = value after reset

Table 7     RAM Watchdog (address 0x14)

#### Bits 15:8    WDV[7:0]: Watchdog Value

Actual Message RAM Watchdog Counter Value.

#### Bits 7:0    WDC[7:0]: Watchdog Configuration

Start value of the Message RAM Watchdog Counter. With the reset value of "00" the counter is disabled.

### 2.3.7 CC Control Register (CCCR)

For details about setting and resetting of single bits see Section 3.1.1.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x18										res						
										R-0						

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res	TXP	FDBS	FDO	CMR[1:0]	CME[1:0]	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT		

R = Read, W = Write, P = Protected write, p = Protected set; -n = value after reset

Table 8 CC Control Register (address 0x18)

#### Bit 14 TXP: Transmit Pause

If this bit is set, the M\_TTCAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame (see Section 3.5.1).

- 0= Transmit pause disabled
- 1= Transmit pause enabled

#### Bit 13 FDBS: CAN FD Bit Rate Switching

- 0= This node transmits no frames with bit rate switching
- 1= This node transmits all frames (excl. remote frames) with bit rate switching

#### Bit 12 FDO: CAN FD Operation

- 0= This node transmits all frames in CAN format according to ISO11898-1
- 1= This node transmits all frames (excl. remote frames) in CAN FD format

#### Bits 11:10 CMR[1:0]: CAN Mode Request

A change of the CAN operation mode is requested by writing to this bit field. After change to the requested operation mode the bit field is reset to "00" and the status flags **FDBS** and **FDO** are set accordingly. In case the requested CAN operation mode is not enabled, the value written to **CMR** is retained until it is overwritten by the next mode change request. In case **CME** = "01"/"10"/"11" a change to CAN operation according to ISO 11898-1 is always possible. Default is CAN operation according to ISO11898-1.

- 00 unchanged
- 01 Request CAN FD operation
- 10 Request CAN FD operation with bit rate switching
- 11 Request CAN operation according ISO11898-1

#### Bits 9:8 CME[1:0]: CAN Mode Enable

- 00 CAN operation according to ISO11898-1 enabled
- 01 CAN FD operation enabled
- 10 CAN FD operation with bit rate switching enabled
- 11 CAN FD operation with bit rate switching enabled

**Note:** When **CME** = "00", received frames are strictly interpreted according to ISO11898-1, which leads to the transmission of an error frame when receiving a CAN FD frame. In case **CME** = "01", transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With **CME** = "10"/"11", transmission and reception of long and fast CAN FD frames is enabled.

**Bit 7 TEST:** Test Mode Enable

- 0= Normal operation, register **TEST** holds reset values
- 1= Test Mode, write access to register **TEST** enabled

**Bit 6 DAR:** Disable Automatic Retransmission

- 0= Automatic retransmission of messages not transmitted successfully enabled
- 1= Automatic retransmission disabled

**Bit 5 MON** Bus Monitoring Mode

Bit **MON** can only be set by the Host when both **CCE** and **INIT** are set to '1'. The bit can be reset by the Host at any time.

- 0= Bus Monitoring Mode is disabled
- 1= Bus Monitoring Mode is enabled

**Bit 4 CSR:** Clock Stop Request

- 0= No clock stop is requested
- 1= Clock stop requested. When clock stop is requested, first **INIT** and then **CSA** will be set after all pending transfer requests have been completed and the CAN bus reached *idle*.

**Bit 3 CSA:** Clock Stop Acknowledge

- 0= No clock stop acknowledged
- 1= M\_TTCAN may be set in power down by stopping **m\_ttcanc\_hclk** and **m\_ttcanc\_cclk**

**Bit 2 ASM** Restricted Operation Mode

Bit **ASM** can only be set by the Host when both **CCE** and **INIT** are set to '1'. The bit can be reset by the Host at any time. For a description of the Restricted Operation Mode see Section 3.1.5.

- 0= Normal CAN operation
- 1= Restricted Operation Mode active

**Bit 1 CCE:** Configuration Change Enable

- 0= The CPU has no write access to the protected configuration registers
- 1= The CPU has write access to the protected configuration registers (while **CCCR.INIT** = '1')

**Bit 0 INIT:** Initialization

- 0= Normal Operation
- 1= Initialization is started

**Note:** Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to **INIT** can be read back. Therefore the programmer has to assure that the previous value written to **INIT** has been accepted by reading **INIT** before setting **INIT** to a new value.

### 2.3.8 Bit Timing & Prescaler Register (BTP)

This register is only writable if bits **CCCR.CCE** and **CCCR.INIT** are set. The CAN bit time may be programmed in the range of 4 to 81 time quanta. The CAN time quantum may be programmed in the range of 1 to 1024 **m\_ttcanc\_cclk** periods.  $t_q = (\text{BRP} + 1) \text{ m_ttcanc_cclk}$  period.

**TSEG1** is the sum of Prop\_Seg and Phase\_Seg1. **TSEG2** is Phase\_Seg2.

Therefore the length of the bit time is (programmed values) [**TSEG1 + TSEG2 + 3**]  $t_q$  or (functional values) [Sync\_Seg + Prop\_Seg + Phase\_Seg1 + Phase\_Seg2]  $t_q$ .

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x1C	res															BRP[9:0]
	R-0															RP-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res															SJW[3:0]
	R-0															RP-3

R = Read, P = Protected write; -n = value after reset

Table 9 Bit Timing & Prescaler Register (address 0x1C)

#### Bits 25:16 BRP[9:0]: Baud Rate Prescaler

0x000-0x3FF The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 1023. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

#### Bits 13:8 TSEG1[5:0]: Time segment before sample point

0x01-0x3F Valid values are 1 to 63. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

#### Bits 7:4 TSEG2[3:0]: Time segment after sample point

0x0-0xF Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

#### Bits 3:0 SJW[3:0]: (Re) Synchronization Jump Width

0x0-0xF Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

**Note:** With a CAN clock (**m\_ttcanc\_cclk**) of 8 MHz, the reset value of 0x00000A33 configures the M\_TTCAN for a bit rate of 500 kBit/s.

### 2.3.9 Timestamp Counter Configuration (TSCC)

For a description of the Timestamp Counter see Section 3.2, *Timestamp Generation*.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x020	res														TCP[3:0]	
	R-0														RP-0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res														TSS[1:0]	
	R-0														RP-0	

R = Read, P = Protected write; -n = value after reset

Table 10      *Timestamp Counter Configuration (address 0x020)*

#### Bit 19:16    TCP[3:0]: Timestamp Counter Prescaler

0x0-0xF   Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1...16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

**Note:** *With CAN FD an external counter is required for timestamp generation (TSS = “10”)*

#### Bits 1:0    TSS[1:0]: Timestamp Select

- 00= Timestamp counter value always 0x0000
- 01= Timestamp counter value incremented according to **TCP**
- 10= External timestamp counter value used
- 11= Same as “00”

### 2.3.10 Timestamp Counter Value (TSCV)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x024	res															
	R-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TSC[15:0]															
	RC-0															

R = Read, C = Clear on write; -n = Value after reset

Table 11      *Timestamp Counter Value (address 0x024)*

#### Bit 15:0    TSC[15:0]: Timestamp Counter

The internal/external Timestamp Counter value is captured on start of frame (both Rx and Tx). When **TSCC.TSS** = “01”, the Timestamp Counter is incremented in multiples of CAN bit times [1...16] depending on the configuration of **TSCC.TCP**. A wrap around sets interrupt flag **IR.TSW**. Write access resets the counter to zero. When **TSCC.TSS** = “10”, **TSC** reflects the external Timestamp Counter value. A write access has no impact.

**Note:** *A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.*

### 2.3.11 Timeout Counter Configuration (TOCC)

For a description of the Timeout Counter see Section 3.3.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0x28	TOP[15:0]														
																RP-1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res															TOS[1:0] ETOC
																RP-0 RP-0

R = Read, P = Protected write; -n = value after reset

Table 12 Timeout Counter Configuration (address 0x28)

#### Bit 31:16 TOP[15:0]: Timeout Period

Start value of the Timeout Counter (down-counter). Configures the Timeout Period.

#### Bits 2:1 TOS[1:0]: Timeout Select

When operating in Continuous mode, a write to **TOCV** presets the counter to the value configured by **TOCC.TOP** and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by **TOCC.TOP**. Down-counting is started when the first FIFO element is stored.

00= Continuous operation

01= Timeout controlled by Tx Event FIFO

10= Timeout controlled by Rx FIFO 0

11= Timeout controlled by Rx FIFO 1

#### Bit 0 ETOC: Enable Timeout Counter

0= Timeout Counter disabled

1= Timeout Counter enabled

**Note:** For use of timeout function with CAN FD see Section 3.3.

### 2.3.12 Timeout Counter Value (TOCV)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0x02C	res														
																R-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TOC[15:0]															RC-1

R = Read, C = Clear on write; -n = value after reset

Table 13 Timeout Counter Value (address 0x02C)

#### Bit 15:0 TOC[15:0]: Timeout Counter

The Timeout Counter is decremented in multiples of CAN bit times [1...16] depending on the configuration of **TSCC.TCP**. When decremented to zero, interrupt flag **IR.TOO** is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via **TOCC.TOS**.

### 2.3.13 Error Counter Register (ECR)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0x40	res														CEL[7:0]		
	R-0														X-0		
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	RP		REC[6:0]														TEC[7:0]
	R-0		R-0														R-0

R = Read, X = Reset on read; -n = value after reset

Table 14 Error Counter Register (address 0x40)

#### Bits 23:16 CEL[7:0]: CAN Error Logging

The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to **CEL**. The counter stops at 0xFF; the next increment of **TEC** or **REC** sets interrupt flag **IR.ELO**.

#### Bit 15 RP: Receive Error Passive

- 0= The Receive Error Counter is below the *error passive* level of 128
- 1= The Receive Error Counter has reached the *error passive* level of 128

#### Bits 14:8 REC[6:0]: Receive Error Counter

Actual state of the Receive Error Counter, values between 0 and 127

#### Bits 7:0 TEC[7:0]: Transmit Error Counter

Actual state of the Transmit Error Counter, values between 0 and 255

**Note:** When **CCCR.ASM** is set, the CAN protocol controller does not increment **TEC** and **REC** when a CAN protocol error is detected, but **CEL** is still incremented. This enables monitoring of collisions between CAN frames and ASC frames.

### 2.3.14 Protocol Status Register (PSR)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x44	res															
	R-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res	X-0	X-0	X-0	X-0	FLEC[2:0]		BO	EW	EP	ACT[1:0]		LEC[2:0]			S-111
	R-0	X-0	X-0	X-0	S-111		R-0	R-0	R-0	R-0	R-0		S-111			

R = Read, S = Set on read, X = Reset on read; -n = value after reset

Table 15 Protocol Status Register (address 0x44)

#### Bit 13 REDL: Received a CAN FD Message

This bit is set independent of acceptance filtering.

- 0= Since this bit was reset by the CPU, no CAN FD message has been received
- 1= Message in CAN FD format with **EDL** flag set has been received

#### Bit 12 RBRS: BRS flag of last received CAN FD Message

This bit is set together with **REDL**, independent of acceptance filtering.

- 0= Last received CAN FD message did not have its **BRS** flag set
- 1= Last received CAN FD message had its **BRS** flag set

#### Bit 11 RESI: ESI flag of last received CAN FD Message

This bit is set together with **REDL**, independent of acceptance filtering.

- 0= Last received CAN FD message did not have its **ESI** flag set
- 1= Last received CAN FD message had its **ESI** flag set

#### Bits 10:8 FLEC[2:0]: Fast Last Error Code

Type of last error that occurred in the data phase of a CAN FD format frame with its **BRS** flag set. Coding is the same as for **LEC**. This field will be cleared to zero when a CAN FD format frame with its **BRS** flag set has been transferred (reception or transmission) without error.

#### Bit 7 BO: Bus\_Off Status

- 0= The M\_TTCAN is not Bus\_Off
- 1= The M\_TTCAN is in Bus\_Off state

#### Bit 6 EW: Warning Status

- 0= Both error counters are below the Error\_Warning limit of 96
- 1= At least one of error counter has reached the Error\_Warning limit of 96

#### Bit 5 EP: Error Passive

- 0= The M\_TTCAN is in the Error\_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected
- 1= The M\_TTCAN is in the Error\_Passive state

**Bits 4:3      ACT[1:0]: Activity**

Monitors the module's CAN communication state.

- 00= Synchronizing - node is synchronizing on CAN communication
- 01= Idle - node is neither receiver nor transmitter
- 10= Receiver - node is operating as receiver
- 11= Transmitter - node is operating as transmitter

**Bits 2:0      LEC[2:0]: Last Error Code**

The **LEC** indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error.

- 0= **No Error:** No error occurred since **LEC** has been reset by successful reception or transmission.
- 1= **Stuff Error:** More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
- 2= **Form Error:** A fixed format part of a received frame has the wrong format.
- 3= **AckError:** The message transmitted by the M\_TTCAN was not acknowledged by another node.
- 4= **Bit1Error:** During the transmission of a message (with the exception of the arbitration field), the device wanted to send a *recessive* level (bit of logical value '1'), but the monitored bus value was *dominant*.
- 5= **Bit0Error:** During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a *dominant* level (data or identifier bit logical value '0'), but the monitored bus value was *recessive*. During *Bus\_Off* recovery this status is set each time a sequence of 11 *recessive* bits has been monitored. This enables the CPU to monitor the proceeding of the *Bus\_Off* recovery sequence (indicating the bus is not stuck at *dominant* or continuously disturbed).
- 6= **CRCError:** The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.
- 7= **NoChange:** Any read access to the Protocol Status Register re-initializes the **LEC** to '7'. When the **LEC** shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register.

**Note:** When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.

**Note:** The *Bus\_Off* recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR.INIT. If the device goes *Bus\_Off*, it will set CCCR.INIT of its own accord, stopping all bus activities. Once CCCR.INIT has been cleared by the CPU, the device will then wait for 129 occurrences of *Bus Idle* (129 \* 11 consecutive recessive bits) before resuming normal operation. At the end of the *Bus\_Off* recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to PSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the *Bus\_Off* recovery sequence. ECR.REC is used to count these sequences.

### 2.3.15 Interrupt Register (IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. The configuration of **IE** controls whether an interrupt is generated. The configuration of **ILS** controls on which interrupt line an interrupt is signalled.

<b>Bits</b>	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x50	STE	FOE	ACKE	BE	CRCE	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW
	RW-0															
<b>Bits</b>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
	RW-0															

R = Read, W = Write; -n = value after reset

Table 16      *Interrupt Register (address 0x50)*

**Bit 31      STE:**      Stuff Error

- 0= No Stuff Error detected
- 1= More than 5 equal bits in a sequence occurred

**Bit 30      FOE:**      Format Error

- 0= No Format Error detected
- 1= A fixed format part of a received frame has the wrong format

**Bit 29      ACKE:**      Acknowledge Error

- 0= No Acknowledge Error detected
- 1= A transmitted message was not acknowledged by another node

**Bit 28      BE:**      Bit Error

- 0= No Bit Error detected
- 1= Device wanted to send a *rec / dom* level, but monitored bus level was *dom / rec*

**Bit 27      CRCE:**      CRC Error

- 0= No CRC Error detected
- 1= Received CRC did not match the calculated CRC

**Bit 26      WDI:**      Watchdog Interrupt

- 0= No Message RAM Watchdog event occurred
- 1= Message RAM Watchdog event due to missing READY

**Bit 25      BO:**      Bus\_Off Status

- 0= Bus\_Off status unchanged
- 1= Bus\_Off status changed

**Bit 24      EW:**      Warning Status

- 0= Error\_Warning status unchanged
- 1= Error\_Warning status changed

**Bit 23 EP:** Error Passive

- 0= Error\_Passive status unchanged
- 1= Error\_Passive status changed

**Bit 22 ELO:** Error Logging Overflow

- 0= CAN Error Logging Counter did not overflow
- 1= Overflow of CAN Error Logging Counter occurred

**Bit 21 BEU:** Bit Error Uncorrected

Message RAM bit error detected, uncorrected. Controlled by input signal **m\_ttcan\_aeim\_berr[1]** generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets **CCCR.INIT** to '1'. This is done to avoid transmission of corrupted data.

- 0= No bit error detected when reading from Message RAM
- 1= Bit error detected, uncorrected (e.g. parity logic)

**Bit 20 BEC:** Bit Error Corrected

Message RAM bit error detected and corrected. Controlled by input signal **m\_ttcan\_aeim\_berr[0]** generated by an optional external parity / ECC logic attached to the Message RAM.

- 0= No bit error detected when reading from Message RAM
- 1= Bit error detected and corrected (e.g. ECC)

**Bit 19 DRX:** Message stored to Dedicated Rx Buffer

The flag is set whenever a received message has been stored into a dedicated Rx Buffer.

- 0= No Rx Buffer updated
- 1= At least one received message stored into a Rx Buffer

**Bit 18 TOO:** Timeout Occurred

- 0= No timeout
- 1= Timeout reached

**Bit 17 MRAF:** Message RAM Access Failure

The flag is set, when the Rx Handler

- has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message.
- was not able to write a message to the Message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location.

The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the M\_TTCAN is switched into Restricted Operation Mode (see Section 3.1.5). To leave Restricted Operation Mode, the Host CPU has to reset **CCCR.ASM**.

- 0= No Message RAM access failure occurred
- 1= Message RAM access failure occurred

**Bit 16 TSW:** Timestamp Wraparound

- 0= No timestamp counter wrap-around
- 1= Timestamp counter wrapped around

**Bit 15 TEFL:** Tx Event FIFO Element Lost

- 0= No Tx Event FIFO element lost
- 1= Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero

**Bit 14        TEFF:**    Tx Event FIFO Full

- 0= Tx Event FIFO not full
- 1= Tx Event FIFO full

**Bit 13        TEFW:**    Tx Event FIFO Watermark Reached

- 0= Tx Event FIFO fill level below watermark
- 1= Tx Event FIFO fill level reached watermark

**Bit 12        TEFN:**    Tx Event FIFO New Entry

- 0= Tx Event FIFO unchanged
- 1= Tx Handler wrote Tx Event FIFO element

**Bit 11        TFE:**      Tx FIFO Empty

- 0= Tx FIFO non-empty
- 1= Tx FIFO empty

**Bit 10        TCF:**      Transmission Cancellation Finished

- 0= No transmission cancellation finished
- 1= Transmission cancellation finished

**Bit 9         TC:**        Transmission Completed

- 0= No transmission completed
- 1= Transmission completed

**Bit 8         HPM:**      High Priority Message

- 0= No high priority message received
- 1= High priority message received

**Bit 7         RF1L:**      Rx FIFO 1 Message Lost

- 0= No Rx FIFO 1 message lost
- 1= Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero

**Bit 6         RF1F:**      Rx FIFO 1 Full

- 0= Rx FIFO 1 not full
- 1= Rx FIFO 1 full

**Bit 5         RF1W:**      Rx FIFO 1 Watermark Reached

- 0= Rx FIFO 1 fill level below watermark
- 1= Rx FIFO 1 fill level reached watermark

**Bit 4         RF1N:**      Rx FIFO 1 New Message

- 0= No new message written to Rx FIFO 1
- 1= New message written to Rx FIFO 1

**Bit 3         RF0L:**      Rx FIFO 0 Message Lost

- 0= No Rx FIFO 0 message lost
- 1= Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero

**Bit 2         RF0F:**      Rx FIFO 0 Full

- 0= Rx FIFO 0 not full
- 1= Rx FIFO 0 full

**Bit 1            RF0W:** Rx FIFO 0 Watermark Reached

0= Rx FIFO 0 fill level below watermark

1= Rx FIFO 0 fill level reached watermark

**Bit 0            RF0N:** Rx FIFO 0 New Message

0= No new message written to Rx FIFO 0

1= New message written to Rx FIFO 0

### 2.3.16 Interrupt Enable (IE)

The settings in the Interrupt Enable register determine which status changes in the Interrupt Register will be signalled on an interrupt line.

0= Interrupt disabled

1= Interrupt enabled

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x54	STEE	FOEE	ACKEE	BEE	CRCEE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R = Read, W = Write; -n = value after reset

Table 17      Interrupt Enable (address 0x54)

**Bit 31            STEE:** Stuff Error Interrupt Enable

**Bit 30            FOEE:** Format Error Interrupt Enable

**Bit 29            ACKEE:** Acknowledge Error Interrupt Enable

**Bit 28            BEE:** Bit Error Interrupt Enable

**Bit 27            CRCEE:** CRC Error Interrupt Enable

**Bit 26            WDIE:** Watchdog Interrupt Enable

**Bit 25            BOE:** Bus\_Off Status Interrupt Enable

**Bit 24            EWE:** Warning Status Interrupt Enable

**Bit 23            EPE:** Error Passive Interrupt Enable

**Bit 22            ELOE:** Error Logging Overflow Interrupt Enable

**Bit 21            BEUE:** Bit Error Uncorrected Interrupt Enable

**Bit 20            BECE:** Bit Error Corrected Interrupt Enable

**Bit 19            DRXE:** Message stored to Dedicated Rx Buffer Interrupt Enable

**Bit 18            TOOE:** Timeout Occurred Interrupt Enable

<b>Bit 17</b>	<b>MRAFE:</b> Message RAM Access Failure Interrupt Enable
<b>Bit 16</b>	<b>TSWE:</b> Timestamp Wraparound Interrupt Enable
<b>Bit 15</b>	<b>TEFLE:</b> Tx Event FIFO Event Lost Interrupt Enable
<b>Bit 14</b>	<b>TEFFE:</b> Tx Event FIFO Full Interrupt Enable
<b>Bit 13</b>	<b>TEFWE:</b> Tx Event FIFO Watermark Reached Interrupt Enable
<b>Bit 12</b>	<b>TEFNE:</b> Tx Event FIFO New Entry Interrupt Enable
<b>Bit 11</b>	<b>TFEE:</b> Tx FIFO Empty Interrupt Enable
<b>Bit 10</b>	<b>TCFE:</b> Transmission Cancellation Finished Interrupt Enable
<b>Bit 9</b>	<b>TCE:</b> Transmission Completed Interrupt Enable
<b>Bit 8</b>	<b>HPME:</b> High Priority Message Interrupt Enable
<b>Bit 7</b>	<b>RF1LE:</b> Rx FIFO 1 Message Lost Interrupt Enable
<b>Bit 6</b>	<b>RF1FE:</b> Rx FIFO 1 Full Interrupt Enable
<b>Bit 5</b>	<b>RF1WE:</b> Rx FIFO 1 Watermark Reached Interrupt Enable
<b>Bit 4</b>	<b>RF1NE:</b> Rx FIFO 1 New Message Interrupt Enable
<b>Bit 3</b>	<b>RF0LE:</b> Rx FIFO 0 Message Lost Interrupt Enable
<b>Bit 2</b>	<b>RF0FE:</b> Rx FIFO 0 Full Interrupt Enable
<b>Bit 1</b>	<b>RF0WE:</b> Rx FIFO 0 Watermark Reached Interrupt Enable
<b>Bit 0</b>	<b>RF0NE:</b> Rx FIFO 0 New Message Interrupt Enable

### 2.3.17 Interrupt Line Select (ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via **ILE.EINT0** and **ILE.EINT1**.

0= Interrupt assigned to interrupt line **m\_ttcan\_int0**

1= Interrupt assigned to interrupt line **m\_ttcan\_int1**

<b>Bits</b>	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x58	STEL	FOEL	ACKEL	BEL	CRCEL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

<b>Bits</b>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TEFLL	TEFFL	TEFWL	TEFNLL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RFOLL	RF0FL	RF0WL	RFONL
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R = Read, W = Write; -n = value after reset

Table 18 Interrupt Line Select (address 0x58)

- Bit 31      STEL:** Stuff Error Interrupt Line
- Bit 30      FOEL:** Format Error Interrupt Line
- Bit 29      ACKEL:** Acknowledge Error Interrupt Line
- Bit 28      BEL:** Bit Error Interrupt Line
- Bit 27      CRCEL:** CRC Error Interrupt Line
- Bit 26      WDIL:** Watchdog Interrupt Line
- Bit 25      BOL:** Bus\_Off Status Interrupt Line
- Bit 24      EWL:** Warning Status Interrupt Line
- Bit 23      EPL:** Error Passive Interrupt Line
- Bit 22      ELOL:** Error Logging Overflow Interrupt Line
- Bit 21      BEUL:** Bit Error Uncorrected Interrupt Line
- Bit 20      BECL:** Bit Error Corrected Interrupt Line
- Bit 19      DRXL:** Message stored to Dedicated Rx Buffer Interrupt Line
- Bit 18      TOOL:** Timeout Occurred Interrupt Line
- Bit 17      MRAFL:** Message RAM Access Failure Interrupt Line
- Bit 16      TSWL:** Timestamp Wraparound Interrupt Line
- Bit 15      TEFLL:** Tx Event FIFO Event Lost Interrupt Line
- Bit 14      TEFFL:** Tx Event FIFO Full Interrupt Line

<b>Bit 13</b>	<b>TEFWL:</b> Tx Event FIFO Watermark Reached Interrupt Line
<b>Bit 12</b>	<b>TEFNL:</b> Tx Event FIFO New Entry Interrupt Line
<b>Bit 11</b>	<b>TFEL:</b> Tx FIFO Empty Interrupt Line
<b>Bit 10</b>	<b>TCFL:</b> Transmission Cancellation Finished Interrupt Line
<b>Bit 9</b>	<b>TCL:</b> Transmission Completed Interrupt Line
<b>Bit 8</b>	<b>HPML:</b> High Priority Message Interrupt Line
<b>Bit 7</b>	<b>RF1LL:</b> Rx FIFO 1 Message Lost Interrupt Line
<b>Bit 6</b>	<b>RF1FL:</b> Rx FIFO 1 Full Interrupt Line
<b>Bit 5</b>	<b>RF1WL:</b> Rx FIFO 1 Watermark Reached Interrupt Line
<b>Bit 4</b>	<b>RF1NL:</b> Rx FIFO 1 New Message Interrupt Line
<b>Bit 3</b>	<b>RF0LL:</b> Rx FIFO 0 Message Lost Interrupt Line
<b>Bit 2</b>	<b>RF0FL:</b> Rx FIFO 0 Full Interrupt Line
<b>Bit 1</b>	<b>RF0WL:</b> Rx FIFO 0 Watermark Reached Interrupt Line
<b>Bit 0</b>	<b>RF0NL:</b> Rx FIFO 0 New Message Interrupt Line

### 2.3.18 Interrupt Line Enable (ILE)

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits **EINT0** and **EINT1**.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x05C	res															
R-0																
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
res													<b>EINT1</b>	<b>EINT0</b>	RW-0	RW-0
R-0																

R = Read, W = Write; -n = value after reset

Table 19     Interrupt Line Enable (address 0x05C)

<b>Bit 1</b>	<b>EINT1:</b> Enable Interrupt Line 1
0=	Interrupt line <b>m_ttcanc_int1</b> disabled
1=	Interrupt line <b>m_ttcanc_int1</b> enabled
<b>Bit 0</b>	<b>EINT0:</b> Enable Interrupt Line 0
0=	Interrupt line <b>m_ttcanc_int0</b> disabled
1=	Interrupt line <b>m_ttcanc_int0</b> enabled

### 2.3.19 Global Filter Configuration (GFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in Figure 6 and Figure 7.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x080	res															R-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res								ANFS[1:0]		ANFE[1:0]		RRFS		RRFE	
	R-0								RP-0		RP-0		RP-0		RP-0	

R = Read, P = Protected write; -n = value after reset

Table 20 Global Filter Configuration (address 0x080)

#### Bit 5:4 ANFS[1:0]: Accept Non-matching Frames Standard

Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.

- 00= Accept in Rx FIFO 0
- 01= Accept in Rx FIFO 1
- 10= Reject
- 11= Reject

#### Bit 3:2 ANFE[1:0]: Accept Non-matching Frames Extended

Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.

- 00= Accept in Rx FIFO 0
- 01= Accept in Rx FIFO 1
- 10= Reject
- 11= Reject

#### Bit 1 RRFS: Reject Remote Frames Standard

- 0= Filter remote frames with 11-bit standard IDs
- 1= Reject all remote frames with 11-bit standard IDs

#### Bit 0 RRFE: Reject Remote Frames Extended

- 0= Filter remote frames with 29-bit extended IDs
- 1= Reject all remote frames with 29-bit extended IDs

### 2.3.20 Standard ID Filter Configuration (SIDFC)

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages as described in Figure 6.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x084	res															LSS[7:0]
	R-0															RP-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FLSSA[15:2]															res
	RP-0															R-0

R = Read, P = Protected write; -n = value after reset

Table 21 Standard ID Filter Configuration (address 0x084)

#### Bit 23:16 LSS[7:0]: List Size Standard

- 0= No standard Message ID filter
- 1-128= Number of standard Message ID filter elements
- >128= Values greater than 128 are interpreted as 128

#### Bit 15:2 FLSSA[15:2]: Filter List Standard Start Address

Start address of standard Message ID filter list (32-bit word address, see Figure 2).

### 2.3.21 Extended ID Filter Configuration (XIDFC)

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in Figure 7.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x088	res															LSE[6:0]
	R-0															RP-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FLESA[15:2]															res
	RP-0															R-0

R = Read, P = Protected write; -n = value after reset

Table 22 Extended ID Filter Configuration (address 0x088)

#### Bit 22:16 LSE[6:0]: List Size Extended

- 0= No extended Message ID filter
- 1-64= Number of extended Message ID filter elements
- >64= Values greater than 64 are interpreted as 64

#### Bit 15:2 FLESA[15:2]: Filter List Extended Start Address

Start address of extended Message ID filter list (32-bit word address, see Figure 2).

### 2.3.22 Extended ID AND Mask (XIDAM)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0x090	res		EIDM[28:16]														
	R-0															RP-1	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	EIDM[15:0]															RP-1	

R = Read, P = Protected write; -n = value after reset

Table 23 Extended ID AND Mask (address 0x090)

#### Bit 28:0 EIDM[28:0]: Extended ID Mask

For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active.

### 2.3.23 High Priority Message Status (HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x094	res															RP-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FLST		FIDX[6:0]						MSI[1:0]		BIDX[5:0]					
	R-0				R-0				R-0				R-0			

R = Read; -n = Value after reset

Table 24 High Priority Message Status (address 0x094)

#### Bit 15 FLST: Filter List

Indicates the filter list of the matching filter element.

0= Standard Filter List

1= Extended Filter List

#### Bit 14:8 FIDX[6:0]: Filter Index

Index of matching filter element. Range is 0 to **SIDFC.LSS** - 1 resp. **XIDFC.LSE** - 1.

#### Bit 7:6 MSI[1:0]: Message Storage Indicator

00= No FIFO selected

01= FIFO message lost

10= Message stored in FIFO 0

11= Message stored in FIFO 1

#### Bit 5:0 BIDX[5:0]: Buffer Index

Index of Rx FIFO element to which the message was stored. Only valid when **MSI[1]** = '1'.

### 2.3.24 New Data 1 (NDAT1)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
0x98	RW-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
	RW-0															

R = Read; -n = value after reset

Table 25 New Data 1 (address 0x98)

#### Bit 31:0 ND[31:0]: New Data

The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.

0= Rx Buffer not updated

1= Rx Buffer updated from new message

### 2.3.25 New Data 2 (NDAT2)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
0x9C	RW-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
	RW-0															

R = Read; -n = value after reset

Table 26 New Data 2 (address 0x9C)

#### Bit 31:0 ND[63:32]: New Data

The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.

0= Rx Buffer not updated

1= Rx Buffer updated from new message

### 2.3.26 Rx FIFO 0 Configuration (RXF0C)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xA0	F0OM				F0WM[6:0]				res				F0S[6:0]			
	RP-0				RP-0				R-0				RP-0			
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									F0SA[15:2]				res			
									RP-0				R-0			

R = Read, P = Protected write; -n = Value after reset

Table 27 Rx FIFO 0 Configuration (address 0xA0)

#### Bit 31 F0OM: FIFO 0 Operation Mode

FIFO 0 can be operated in blocking or in overwrite mode (see Section 3.4.2).

- 0= FIFO 0 blocking mode
- 1= FIFO 0 overwrite mode

#### Bit 30:24 F0WM[6:0]: Rx FIFO 0 Watermark

- 0= Watermark interrupt disabled
- 1-64= Level for Rx FIFO 0 watermark interrupt (**IR.RF0W**)
- >64= Watermark interrupt disabled

#### Bit 22:16 F0S[6:0]: Rx FIFO 0 Size

- 0= No Rx FIFO 0
  - 1-64= Number of Rx FIFO 0 elements
  - >64= Values greater than 64 are interpreted as 64
- The Rx FIFO 0 elements are indexed from 0 to **F0S-1**

#### Bit 15:2 F0SA[15:2]: Rx FIFO 0 Start Address

Start address of Rx FIFO 0 in Message RAM (32-bit word address, see Figure 2).

### 2.3.27 Rx FIFO 0 Status (RXF0S)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0A4							RF0L	F0F		res						F0PI[5:0]
					R-0		R-0	R-0		R-0						R-0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										res						F0GI[5:0]
										R-0						R-0

R = Read; -n = value after reset

Table 28 Rx FIFO 0 Status (address 0xA4)

#### Bit 25 RF0L: Rx FIFO 0 Message Lost

This bit is a copy of interrupt flag **IR.RF0L**. When **IR.RF0L** is reset, this bit is also reset.

0= No Rx FIFO 0 message lost

1= Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero

**Note:** Overwriting the oldest message when RXF0C.F0OM = '1' will not set this flag.

#### Bit 24 F0F: Rx FIFO 0 Full

0= Rx FIFO 0 not full

1= Rx FIFO 0 full

#### Bit 21:16 F0PI[5:0]: Rx FIFO 0 Put Index

Rx FIFO 0 write index pointer, range 0 to 63.

#### Bit 13:8 F0GI[5:0]: Rx FIFO 0 Get Index

Rx FIFO 0 read index pointer, range 0 to 63.

#### Bit 6:0 F0FL[6:0]: Rx FIFO 0 Fill Level

Number of elements stored in Rx FIFO 0, range 0 to 64.

### 2.3.28 Rx FIFO 0 Acknowledge (RXF0A)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0A8	res															
	R-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res															F0AI[5:0]
	R-0															RW-0

R = Read, W = Write; -n = value after reset

Table 29 Rx FIFO 0 Acknowledge (address 0xA8)

#### Bit 5:0 F0AI[5:0]: Rx FIFO 0 Acknowledge Index

After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to **F0AI**. This will set the Rx FIFO 0 Get Index **RXF0S.F0GI** to **F0AI** + 1 and update the FIFO 0 Fill Level **RXF0S.F0FL**.

### 2.3.29 Rx Buffer Configuration (RXBC)

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

R = Read, P = Protected write; -n = value after reset

Table 30 Rx Buffer Configuration (address 0xAC)

#### Bit 15:2 RBSA[15:2]: Rx Buffer Start Address

Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C.

### 2.3.30 Rx FIFO 1 Configuration (RXF1C)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0B0	F1OM	F1WM[6:0]						res	F1S[6:0]							
	RP-0	RP-0						R-0	RP-0							
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	F1SA[15:2]															res
	RP-0															R-0

R = Read, P = Protected write; -n = value after reset

Table 31 Rx FIFO 1 Configuration (address 0xB00)

#### Bit 31 F1OM: FIFO 1 Operation Mode

FIFO 1 can be operated in blocking or in overwrite mode (see Section 3.4.2).

0= FIFO 1 blocking mode

1= FIFO 1 overwrite mode

#### Bit 30:24 F1WM[6:0]: Rx FIFO 1 Watermark

0= Watermark interrupt disabled

1-64= Level for Rx FIFO 1 watermark interrupt (IR.RF1W)

>64= Watermark interrupt disabled

#### Bit 22:16 F1S[6:0]: Rx FIFO 1 Size

0= No Rx FIFO 1

1-64= Number of Rx FIFO 1 elements

>64= Values greater than 64 are interpreted as 64

The Rx FIFO 1 elements are indexed from 0 to F1S - 1

#### Bit 15:2 F1SA[15:2]: Rx FIFO 1 Start Address

Start address of Rx FIFO 1 in Message RAM (32-bit word address, see Figure 2).

### 2.3.31 Rx FIFO 1 Status (RXF1S)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0B4	DMS[1:0]	res				RF1L	F1F	res	F1PI[5:0]							
	R-0	R-0				R-0	R-0	R-0	R-0							
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res	F1GI[5:0]						res	F1FL[6:0]							
	R-0	R-0						R-0	R-0							

R = Read; -n = value after reset

Table 32 Rx FIFO 1 Status (address 0xB4)

#### Bits 31:30 DMS[1:0]: Debug Message Status

00= Idle state, wait for reception of debug messages, DMA request is cleared

01= Debug message A received

10= Debug messages A, B received

11= Debug messages A, B, C received, DMA request is set

**Bit 25      RF1L:** Rx FIFO 1 Message Lost

This bit is a copy of interrupt flag **IR.RF1L**. When **IR.RF1L** is reset, this bit is also reset.

0= No Rx FIFO 1 message lost

1= Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero

**Note:** Overwriting the oldest message when **RXF1C.F1OM = '1'** will not set this flag.

**Bit 24      F1F:** Rx FIFO 1 Full

0= Rx FIFO 1 not full

1= Rx FIFO 1 full

**Bit 21:16    F1PI[5:0]:** Rx FIFO 1 Put Index

Rx FIFO 1 write index pointer, range 0 to 63.

**Bit 13:8    F1GI[5:0]:** Rx FIFO 1 Get Index

Rx FIFO 1 read index pointer, range 0 to 63.

**Bit 6:0      F1FL[6:0]:** Rx FIFO 1 Fill Level

Number of elements stored in Rx FIFO 1, range 0 to 64.

**2.3.32 Rx FIFO 1 Acknowledge (RXF1A)**

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0B8																
	res															
	R-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res								F1AI[5:0]							
	R-0								RW-0							

R = Read, W = Write; -n = value after reset

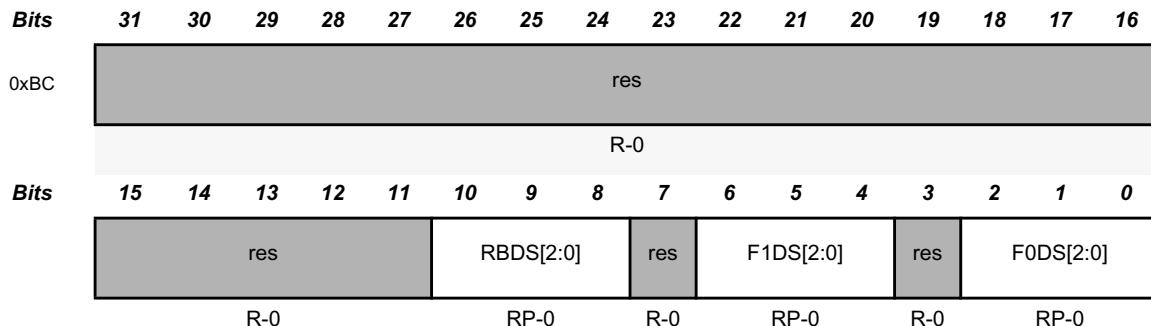
**Table 33 Rx FIFO 1 Acknowledge (address 0xB8)**

**Bit 5:0      F1AI[5:0]:** Rx FIFO 1 Acknowledge Index

After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to **F1AI**. This will set the Rx FIFO 1 Get Index **RXF1S.F1GI** to **F1AI + 1** and update the FIFO 1 Fill Level **RXF1S.F1FL**.

### 2.3.33 Rx Buffer / FIFO Element Size Configuration (RXESC)

Configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.



R = Read, P = Protected write, -U = undefined; -n = value after reset

Table 34 Rx Buffer / FIFO Element Size Configuration (address 0xBC)

#### Bits 10:8 RBDS[2:0]: Rx Buffer Data Field Size

- 000= 8 byte data field
- 001= 12 byte data field
- 010= 16 byte data field
- 011= 20 byte data field
- 100= 24 byte data field
- 101= 32 byte data field
- 110= 48 byte data field
- 111= 64 byte data field

#### Bits 6:4 F1DS[2:0]: Rx FIFO 1 Data Field Size

- 000= 8 byte data field
- 001= 12 byte data field
- 010= 16 byte data field
- 011= 20 byte data field
- 100= 24 byte data field
- 101= 32 byte data field
- 110= 48 byte data field
- 111= 64 byte data field

#### Bits 2:0 F0DS[2:0]: Rx FIFO 0 Data Field Size

- 000= 8 byte data field
- 001= 12 byte data field
- 010= 16 byte data field
- 011= 20 byte data field
- 100= 24 byte data field
- 101= 32 byte data field
- 110= 48 byte data field
- 111= 64 byte data field

**Note:** In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer or Rx FIFO, only the number of bytes as configured by RXESC are stored to the Rx Buffer resp. Rx FIFO element. The rest of the frame's data field is ignored.

### 2.3.34 Tx Buffer Configuration (TXBC)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0C0	res	TFQM			TFQS[5:0]				res			NDTB[5:0]				
	R-0	RP-0			RP-0				R-0			RP-0				
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TBSA[15:2]				res				
									RP-0			R-0				

R = Read, P = Protected write; -n = value after reset

Table 35 Tx Buffer Configuration (address 0x0C0)

**Bit 30 TFQM:** Tx FIFO/Queue Mode

- 0= Tx FIFO operation
- 1= Tx Queue operation

**Bit 29:24 TFQS[5:0]:** Transmit FIFO/Queue Size

- 0= No Tx FIFO/Queue
- 1-32= Number of Tx Buffers used for Tx FIFO/Queue
- >32= Values greater than 32 are interpreted as 32

**Bit 21:16 NDTB[5:0]:** Number of Dedicated Transmit Buffers

- 0= No Dedicated Tx Buffers
- 1-32= Number of Dedicated Tx Buffers
- >32= Values greater than 32 are interpreted as 32

**Bit 15:2 TBSA[15:2]:** Tx Buffers Start Address

Start address of Tx Buffers section in Message RAM (32-bit word address, see Figure 2).

**Note:** Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

### 2.3.35 Tx FIFO/Queue Status (TXFQS)

The Tx FIFO/Queue status is related to the pending Tx requests listed in register **TXBRP**. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (**TXBRP** not yet updated).

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xC4												TFQF		TFQPI[4:0]		

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			res			TFGI[4:0]			res			TFFL[5:0]				

R = Read; -n = value after reset

Table 36 Tx FIFO/Queue Status (address 0xC4)

**Bit 21 TFQF:** Tx FIFO/Queue Full

0= Tx FIFO/Queue not full

1= Tx FIFO/Queue full

**Bit 20:16 TFQPI[4:0]:** Tx FIFO/Queue Put Index

Tx FIFO/Queue write index pointer, range 0 to 31.

**Bit 12:8 TFGI[4:0]:** Tx FIFO Get Index

Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (**TXBC.TFQM = '1'**).

**Bit 5:0 TFFL[5:0]:** Tx FIFO Free Level

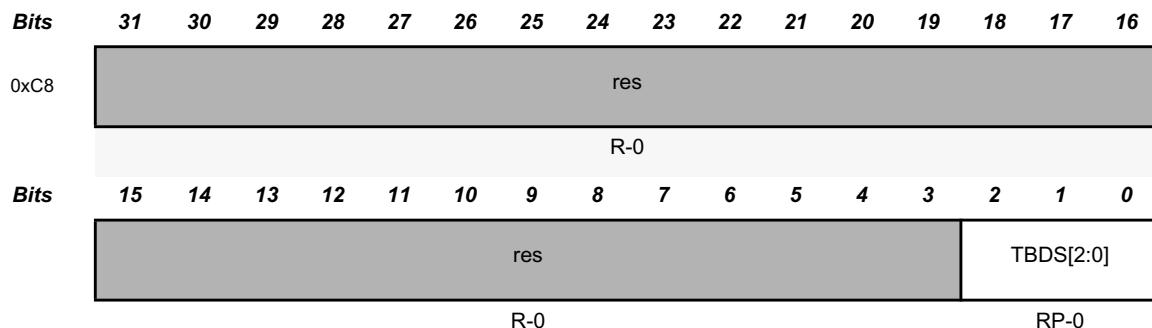
Number of consecutive free Tx FIFO elements starting from **TFGI**, range 0 to 32. Read as zero when Tx Queue operation is configured (**TXBC.TFQM = '1'**).

**Note:** In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers.

**Example:** For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

### 2.3.36 Tx Buffer Element Size Configuration (TXESC)

Configures the number of data bytes belonging to a Tx Buffer element. Data field sizes > 8 bytes are intended for CAN FD operation only.



R = Read, P = Protected write, -U = undefined; -n = value after reset

Table 37 Tx Buffer Element Size Configuration (address 0xC8)

#### Bits 2:0 TBDS[2:0]: Tx Buffer Data Field Size

- 000= 8 byte data field
- 001= 12 byte data field
- 010= 16 byte data field
- 011= 20 byte data field
- 100= 24 byte data field
- 101= 32 byte data field
- 110= 48 byte data field
- 111= 64 byte data field

**Note:** In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size TXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as “0xCC” (padding bytes).

### 2.3.37 Tx Buffer Request Pending (TXBRP)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0CC	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
	R-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
	R-0															

R = Read; -n = value after reset

Table 38 Tx Buffer Request Pending (address 0x0CC)

#### Bit 31:0 TRP[31:0]: Transmission Request Pending

Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register **TXBAR**. The bits are reset after a requested transmission has completed or has been cancelled via register **TXBCR**.

**TXBRP** bits are set only for those Tx Buffers configured via **TXBC**. After a **TXBRP** bit has been set, a Tx scan (see Section 3.5) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID).

A cancellation request resets the corresponding transmission request pending bit of register **TXBRP**. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding **TXBRP** bit has been reset.

After a cancellation has been requested, a finished cancellation is signalled via **TXBCF**

- after successful transmission together with the corresponding **TXBTO** bit
- when the transmission has not yet been started at the point of cancellation
- when the transmission has been aborted due to lost arbitration
- when an error occurred during frame transmission

In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding **TXBCF** bit is set for all unsuccessful transmissions.

0= No transmission request pending

1= Transmission request pending

**Note:** *TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding TXBRP bit is reset.*

### 2.3.38 Tx Buffer Add Request (TXBAR)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0D0	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
	RW-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
	RW-0															

R = Read, W = Write; -n = value after reset

Table 39 Tx Buffer Add Request (address 0x0D0)

#### Bit 31:0 AR[31:0]: Add Request

Each Tx Buffer has its own Add Request bit. Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to **TXBAR**. **TXBAR** bits are set only for those Tx Buffers configured via **TXBC**. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.

0= No transmission request added

1= Transmission requested added

**Note:** If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored.

### 2.3.39 Tx Buffer Cancellation Request (TXBCR)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0D4	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16
	RW-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
	RW-0															

R = Read, W = Write; -n = value after reset

Table 40 Tx Buffer Cancellation Request (address 0x0D4)

#### Bit 31:0 CR[31:0]: Cancellation Request

Each Tx Buffer has its own Cancellation Request bit. Writing a '1' will set the corresponding Cancellation Request bit; writing a '0' has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to **TXBCR**. **TXBCR** bits are set only for those Tx Buffers configured via **TXBC**. The bits remain set until the corresponding bit of **TXBRP** is reset.

0= No cancellation pending

1= Cancellation pending

### 2.3.40 Tx Buffer Transmission Occurred (TXBTO)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16
	R-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
	R-0															

R = Read; -n = value after reset

Table 41 Tx Buffer Transmission Occurred (address 0x0D8)

#### Bit 31:0 TO[31:0]: Transmission Occurred

Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding **TXBRP** bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register **TXBAR**.

0= No transmission occurred

1= Transmission occurred

### 2.3.41 Tx Buffer Cancellation Finished (TXBCF)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
	R-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
	R-0															

R = Read; -n = value after reset

Table 42 Transmit Buffer Cancellation Finished (address 0x0DC)

#### Bit 31:0 CF[31:0]: Cancellation Finished

Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding **TXBRP** bit is cleared after a cancellation was requested via **TXBCR**. In case the corresponding **TXBRP** bit was not set at the point of cancellation, **CF** is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register **TXBAR**.

0= No transmit buffer cancellation

1= Transmit buffer cancellation finished

### 2.3.42 Tx Buffer Transmission Interrupt Enable (TXBTIE)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0E0	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16
	RW-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
	RW-0															

R = Read, W = Write; -n = value after reset

Table 43 Tx Buffer Transmission Interrupt Enable (address 0x0E0)

#### Bit 31:0 TIE[31:0]: Transmission Interrupt Enable

Each Tx Buffer has its own Transmission Interrupt Enable bit.

- 0= Transmission interrupt disabled
- 1= Transmission interrupt enable

### 2.3.43 Tx Buffer Cancellation Finished Interrupt Enable (TXBCIE)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0E4	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16
	RW-0															
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
	RW-0															

R = Read, W = Write; -n = value after reset

Table 44 Tx Buffer Cancellation Finished Interrupt Enable (address 0x0E4)

#### Bit 31:0 CFIE[31:0]: Cancellation Finished Interrupt Enable

Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit.

- 0= Cancellation finished interrupt disabled
- 1= Cancellation finished interrupt enabled

### 2.3.44 Tx Event FIFO Configuration (TXEFC)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0F0	res					EFWM[5:0]		res				EFS[5:0]				
	R-0					RP-0				R-0			RP-0			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									EFSA[15:2]				res			
													RP-0			R-0

R = Read, P = Protected write; -n = value after reset

Table 45 Tx Event FIFO Configuration (address 0x0F0)

**Bit 29:24 EFWM[5:0]: Event FIFO Watermark**

- 0= Watermark interrupt disabled
- 1-32= Level for Tx Event FIFO watermark interrupt (**IR.TEFW**)
- >32= Watermark interrupt disabled

**Bit 21:16 EFS[5:0]: Event FIFO Size**

- 0= Tx Event FIFO disabled
  - 1-32= Number of Tx Event FIFO elements
  - >32= Values greater than 32 are interpreted as 32
- The Tx Event FIFO elements are indexed from 0 to **EFS-1**

**Bit 15:2 EFSA[15:2]: Event FIFO Start Address**

Start address of Tx Event FIFO in Message RAM (32-bit word address, see Figure 2).

### 2.3.45 Tx Event FIFO Status (TXEFS)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0F4							TEFL	EFF								EFPI[4:0]
					R-0		R-0	R-0			R-0					R-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							EFGI[4:0]			res						EFFL[5:0]
					R-0		R-0			R-0						R-0

R = Read; -n = value after reset

Table 46 Tx Event FIFO Status (address 0x0F4)

#### Bit 25 TEFL: Tx Event FIFO Element Lost

This bit is a copy of interrupt flag **IR.TEFL**. When **IR.TEFL** is reset, this bit is also reset.

0= No Tx Event FIFO element lost

1= Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.

#### Bit 24 EFF: Event FIFO Full

0= Tx Event FIFO not full

1= Tx Event FIFO full

#### Bit 20:16 EFPI[4:0]: Event FIFO Put Index

Tx Event FIFO write index pointer, range 0 to 31.

#### Bit 12:8 EFGI[4:0]: Event FIFO Get Index

Tx Event FIFO read index pointer, range 0 to 31.

#### Bit 5:0 EFFL[5:0]: Event FIFO Fill Level

Number of elements stored in Tx Event FIFO, range 0 to 32.

### 2.3.46 Tx Event FIFO Acknowledge (TXEFA)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0F8																res
																R-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																EFAI[4:0]
																RW-0

R = Read, W = Write; -n = value after reset

Table 47 Tx Event FIFO Acknowledge (address 0x0F8)

#### Bit 4:0 EFAI[4:0]: Event FIFO Acknowledge Index

After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to **EFAI**. This will set the Tx Event FIFO Get Index **TXEFS.EFGI** to **EFAI** + 1 and update the FIFO 0 Fill Level **TXEFS.EFFL**.

### 2.3.47 TT Trigger Memory Configuration (TTTMC)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x100	res														TME[6:0]	
	R-0														RP-0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TMSA[15:2]														res	
	RP-0														R-0	

R = Read, P = Protected write; -n = value after reset

Table 48 TT Trigger Memory Configuration (address 0x100)

#### Bit 22:16 TME[6:0]: Trigger Memory Elements

- 0= No Trigger Memory
- 1-64= Number of Trigger Memory elements
- >64= Values greater than 64 are interpreted as 64

#### Bit 15:2 TMSA[15:2]: Trigger Memory Start Address

Start address of Trigger Memory in Message RAM (32-bit word address, see Figure 2).

### 2.3.48 TT Reference Message Configuration (TTRMC)

For details about handling of reference messages see Section 4.7.1.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x104	RMPS			XTD		res		RID[28:16]								
	RP-0			RP-0		R-0		RP-0								
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RID[15:0]														RP-0	

R = Read, P = Protected write; -n = value after reset

Table 49 TT Reference Message Configuration (address 0x104)

#### Bit 31 RMPS: Reference Message Payload Select

Ignored in case of time slaves.

- 0= Reference message has no additional payload
- 1= The following elements are taken from Tx Buffer 0:  
Message Marker **MM**, Event FIFO Control **EFC**, Data Length Code **DLC**, Data Bytes **DB**  
(Level 1: bytes 2-8, Level 0,2: bytes 5-8)

#### Bit 30 XTD: Extended Identifier

- 0= 11-bit standard identifier
- 1= 29-bit extended identifier

#### Bit 28:0 RID[28:0]: Reference Identifier

Identifier transmitted with reference message and used for reference message filtering. Standard or extended reference identifier depending on bit **XTD**. A standard identifier has to be written to **ID[28:18]**.

### 2.3.49 TT Operation Configuration (TTOCF)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x108	res				EVTP	ECC	EGTF						AWL[7:0]			
	R-0				RP-0	RP-0	RP-0						RP-0x01			
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EECS				IRTO[6:0]				LDSSDL[2:0]		TM	GEN	res		OM[1:0]	
	RP-0				RP-0				RP-0		RP-0	RP-0	R-0		RP-0	

R = Read, P = Protected write; -n = value after reset

Table 50 TT Operation Configuration (address 0x108)

**Bit 26 EVTP:** Event Trigger Polarity

- 0= Rising edge trigger
- 1= Falling edge trigger

**Bit 25 ECC:** Enable Clock Calibration

- 0= Automatic clock calibration in TTCAN Level 0,2 is disabled
- 1= Automatic clock calibration in TTCAN Level 0,2 is enabled

**Bit 24 EGTF:** Enable Global Time Filtering

- 0= Global time filtering in TTCAN Level 0,2 is disabled
- 1= Global time filtering in TTCAN Level 0,2 is enabled

**Bit 23:16 AWL[7:0]: Application Watchdog Limit**

The application watchdog can be disabled by programming **AWL** to 0x00.

0x00-FF Maximum time after which the application has to serve the application watchdog.  
The application watchdog is incremented once each 256 NTUs.

**Bit 15 EECS:** Enable External Clock Synchronization

If enabled, TUR configuration (**TURCF.NCL** only) may be updated during TTCAN operation.

- 0= External clock synchronization in TTCAN Level 0,2 disabled
- 1= External clock synchronization in TTCAN Level 0,2 enabled

**Bit 14:8 IRTO[6:0]: Initial Reference Trigger Offset**

0x00-7F Positive offset, range from 0 to 127

**Bit 7:5 LDSSDL[2:0]: LD of Synchronization Deviation Limit**

The Synchronization Deviation Limit **SDL** is configured by its dual logarithm **LDSSDL** with  $\text{SDL} = 2^{(\text{LDSSDL} + 5)}$ . It should not exceed the clock tolerance given by the CAN bit timing configuration.

0x0-7 LD of Synchronization Deviation Limit (**SDL**  $\leq$  32...4096)

**Bit 4 TM:** Time Master

- 0= Time Master function disabled
- 1= Potential Time Master

**Bit 3 GEN:** Gap Enable

- 0= Strictly time-triggered operation
- 1= External event-synchronized time-triggered operation

**Bit 1:0 OM[1:0]: Operation Mode**

- 00= Event-driven CAN communication, default
- 01= TTCAN level 1
- 10= TTCAN level 2
- 11= TTCAN level 0

**2.3.50 TT Matrix Limits (TTMLM)**

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0x10C	res		ENTT[11:0]														
	R-0															RP-0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	res		TXEW[3:0]			CSS[1:0]			CCM[5:0]						RP-0		
	R-0		RP-0			RP-0			RP-0						RP-0		

R = Read, P = Protected write; -n = value after reset

Table 51 TT Matrix Limits (address 0x10C)

**Bit 27:16 ENTT[11:0]: Expected Number of Tx Triggers**

0x000-FFF Expected number of Tx Triggers in one Matrix Cycle

**Bit 11:8 TXEW[3:0]: Tx Enable Window**

0x0-F Length of Tx enable window, 1-16 NTU cycles

**Bit 7:6 CSS[1:0]: Cycle Start Synchronization**

Enables sync pulse output at pin **m\_ttcanc\_soc**.

- 00= No sync pulse
- 01= Sync pulse at start of basic cycle
- 10= Sync pulse at start of matrix cycle
- 11= Reserved

**Bit 5:0 CCM[5:0]: Cycle Count Max**

- |        |                                  |
|--------|----------------------------------|
| 0x00   | 1 Basic Cycle per Matrix Cycle   |
| 0x01   | 2 Basic Cycles per Matrix Cycle  |
| 0x03   | 4 Basic Cycles per Matrix Cycle  |
| 0x07   | 8 Basic Cycles per Matrix Cycle  |
| 0x0F   | 16 Basic Cycles per Matrix Cycle |
| 0x1F   | 32 Basic Cycles per Matrix Cycle |
| 0x3F   | 64 Basic Cycles per Matrix Cycle |
| others | Reserved                         |

**Note:** ISO 11898-4, Section 5.2.1 requires, that only the listed cycle count values are configured.

Other values are possible but may lead to inconsistent matrix cycles.

### 2.3.51 TUR Configuration (TURCF)

The length of the NTU is given by:  $NTU = CAN\ Clock\ Period \bullet NC/DC$

**NC** is an 18-bit value. Its high part, **NCH[17:16]** is hard wired to 0b01. Therefore the range of **NC** is 0x10000...0x1FFF. The value configured by **NCL** is the initial value for **TURNA.NAV[15:0]**. **DC** is set to 0x1000 by hardware reset and it may not be written to 0x0000.

Level 1:  $NC \geq 4 \bullet DC$  and  $NTU = CAN\ bit\ time$

Level 0,2:  $NC \geq 8 \bullet DC$

The actual value of TUR may be changed by the clock drift compensation function of TTCAN Level 0 and Level 2 in order to adjust the node's local view of the NTU to the time master's view of the NTU. **DC** will not be changed by the automatic drift compensation, **TURNA.NAV** may be adjusted around **NC** in the range of the Synchronisation Deviation Limit given by **TTOCF.LDSL**. **NC** and **DC** should be programmed to the largest suitable values in order to allow the best computational accuracy for the drift compensation process.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x110	ELT	res														
	RP-0	R-0														RP-0x1000
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																NCL[15:0]
																RP-0

R = Read, P = Protected write; -n = value after reset

Table 52 TUR Configuration (address 0x110)

#### Bit 31 ELT: Enable Local Time

- 0= Local time is stopped, default
- 1= Local time is enabled

**Note:** Local time is started by setting **ELT**. It remains active until **ELT** is reset or until the next hardware reset. **TURCF.DC** is locked when **TURCF.ELT** = '1'. If **ELT** is written to '0', the readable value will stay at '1' until the new value has been synchronized into the CAN clock domain. During this time write access to the other bits of the register remains locked.

#### Bit 29:16 DC[13:0]: Denominator Configuration

- 0x0000 Illegal value
- 0x0001-3FFF Denominator Configuration

#### Bit 15:0 NCL[15:0]: Numerator Configuration Low

Write access to the TUR Numerator Configuration Low is only possible during configuration with **TURCF.ELT** = '0' or if **TTOCF.EECS** (external clock synchronization enabled) is set. When a new value for **NCL** is written outside TT Configuration Mode, the new value takes effect when **TTOST.WECS** is cleared to '0'. **NCL** is locked **TTOST.WECS** is '1'.

0x0000-FFFF Numerator Configuration Low

**Note:** If  $NC < 7 \bullet DC$  in TTCAN Level 1, then it is required that subsequent time marks in the Trigger Memory must differ by at least 2 NTU.

### 2.3.52 TT Operation Control (TTOCN)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x114														res		
														R-0		

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LCKC	res	ESCN	NIG	TMG	FGP	GCS	TTIE	TMC[1:0]	RTIE	SWS[1:0]	SWP	ECS	SGT		

R = Read, P = Protected write; -n = value after reset

Table 53 TT Operation Control (address 0x114)

#### Bit 15 LCKC: TT Operation Control Register Locked

Set by a write access to register TTOCN. Reset when the updated configuration has been synchronized into the CAN clock domain.

- 0= Write access to TTOCN enabled
- 1= Write access to TTOCN locked

#### Bit 13 ESCN: External Synchronization Control

If enabled the M\_TTCAN synchronizes its cycle time phase to an external event signalled by a rising edge at pin **m\_ttcanc\_evt** (see Section 4.11).

- 0= External synchronization disabled
- 1= External synchronization enabled

#### Bit 12 NIG: Next is Gap

This bit can only be set when the M\_TTCAN is the actual Time Master and when it is configured for external event-synchronized time-triggered operation (**TTOCF.GEN** = '1')

- 0= No action, reset by reception of any reference message
- 1= Transmit next reference message with **Next\_is\_Gap** = '1'

#### Bit 11 TMG: Time Mark Gap

- 0= Reset by each reference message
- 1= Next reference message started when Register Time Mark interrupt **TTIR.RTMI** is activated

#### Bit 10 FGP: Finish Gap

Set by the CPU, reset by each reference message

- 0= No reference message requested
- 1= Application requested start of reference message

#### Bit 9 GCS: Gap Control Select

- 0= Gap control independent from **m\_ttcanc\_evt**
- 1= Gap control by input pin **m\_ttcanc\_evt**

#### Bit 8 TTIE: Trigger Time Mark Interrupt Pulse Enable

External time mark events are configured by trigger memory element **TMEX** (see Section 2.4.7). A trigger time mark interrupt pulse is generated when the trigger memory element becomes active, and the M\_TTCAN is in synchronization state **In\_Schedule** or **In\_Gap**.

- 0= Trigger Time Mark Interrupt output **m\_ttcanc\_tmp** disabled
- 1= Trigger Time Mark Interrupt output **m\_ttcanc\_tmp** enabled

**Bit 7:6      TMC[1:0]: Register Time Mark Compare**

- 00= No Register Time Mark Interrupt generated
- 01= Register Time Mark Interrupt if Time Mark = cycle time
- 10= Register Time Mark Interrupt if Time Mark = local time
- 11= Register Time Mark Interrupt if Time Mark = global time

**Note:** When changing the time mark reference (cycle, local, global time), it is recommended to first write TMC = "00", then reconfigure TTTMK, and finally set TMC to the intended time reference.

**Bit 5      RTIE: Register Time Mark Interrupt Pulse Enable**

Register time mark interrupts are configured by register **TTTMK**. A register time mark interrupt pulse with the length of one **m\_ttcanc\_clk** period is generated when the time referenced by **TTOCN.TMC** (cycle, local, or global) equals **TTTMK.TM**, independent of the synchronization state.

- 0= Register Time Mark Interrupt output **m\_ttcanc\_rtp** disabled
- 1= Register Time Mark Interrupt output **m\_ttcanc\_rtp** enabled

**Bit 4:3      SWS[1:0]: Stop Watch Source**

- 00= Stop Watch disabled
- 01= Actual value of cycle time is copied to **TTCPT.SWV**
- 10= Actual value of local time is copied to **TTCPT.SWV**
- 11= Actual value of global time is copied to **TTCPT.SWV**

**Bit 2      SWP: Stop Watch Polarity**

- 0= Rising edge trigger
- 1= Falling edge trigger

**Bit 1      ECS: External Clock Synchronization**

Writing a '1' to **ECS** sets **TTOST.WECS** if the node is the actual Time Master. **ECS** is reset after one Host clock period. The external clock synchronization takes effect at the start of the next basic cycle.

**Bit 0      SGT: Set Global time**

Writing a '1' to **SGT** sets **TTOST.WGDT** if the node is the actual Time Master. **SGT** is reset after one Host clock period. The global time preset takes effect when the node transmits the next reference message with the **Master\_Ref\_Mark** modified by the preset value written to **TTGTP**.

### 2.3.53 TT Global Time Preset (TTGTP)

If **TTOST.WGDT** is set, the next reference message will be transmitted with the **Master\_Ref\_Mark** modified by the preset value and with **Disc\_Bit** = '1', presetting the global time in all nodes simultaneously.

**TP** is reset to 0x0000 each time a reference message with **Disc\_Bit** = '1' becomes valid or if the node is not the current Time Master. **TP** is locked while **TTOST.WGTD** = '1' after setting **TTOCN.SGT** until the reference message with **Disc\_Bit** = '1' becomes valid or until the node is no longer the current Time Master.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x118																
CTP[15:0]																
																RW-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TP[15:0]																
																RW-0

R = Read, P = Protected write; -n = value after reset

Table 54 TT Global Time Preset (address 0x118)

#### Bit 31:16 CTP[15:0]: Cycle Time Target Phase

CTP is write-protected while **TTOCN.ESCN** or **TTOST.SPL** are set (see Section 4.11).

0x0000-FFFF Defines target value of cycle time when a rising edge of **m\_ttcanc\_evt** is expected

#### Bit 15:0 TP[15:0]: Time Preset

TP is write-protected while **TTOST.WGTD** is set.

0x0000-7FFF Next Master Reference Mark = Master Reference Mark + **TP**

0x8000 reserved

0x8001-FFFF Next Master Reference Mark = Master Reference Mark - (0x10000 - **TP**)

### 2.3.54 TT Time Mark (TTTMK)

A time mark interrupt (**TTIR.RTMI = '1'**) is generated when the time base indicated by **TTOCN.TMC** (cycle time, local time, or global time) has the same value as **TM**.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x11C	LCKM															
	R-0															RW-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																TM[15:0]
																RW-0

R = Read, P = Protected write; -n = value after reset

Table 55 TT Time Mark (address 0x11C)

#### Bit 31 LCKM: TT Time Mark Register Locked

Always set by a write access to registers TTOCN. Set by write access to register TTTMK when **TTOCN.TMC ≠ "00"**. Reset when the registers have been synchronized into the CAN clock domain.

- 0= Write access to TTTMK enabled
- 1= Write access to TTTMK locked

#### Bit 22:16 TICC[6:0]: Time Mark Cycle Code

Cycle count for which the time mark is valid.

- 0b0000000x valid for all cycles
- 0b0000001c valid every second cycle at cycle count mod2 = c
- 0b00001cc valid every fourth cycle at cycle count mod4 = cc
- 0b0001ccc valid every eighth cycle at cycle count mod8 = ccc
- 0b001cccc valid every sixteenth cycle at cycle count mod16 = cccc
- 0b01cccccc valid every thirty-second cycle at cycle count mod32 = ccccc
- 0b1ccccccc valid every sixty-fourth cycle at cycle count mod64 = ccccc

#### Bit 15:0 TM[15:0]: Time Mark

0x0000-FFFF Time Mark

**Note:** When using byte access to register TTTMK it is recommended to first disable the time mark compare function (TTOCN.TMC = "00") to avoid compares on inconsistent register values.

### 2.3.55 TT Interrupt Register (TTIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x120														CER	AW	WT
									R-0					RW-0	RW-0	RW-0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IWT	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
	RW-0															

R = Read, W = Write; -n = value after reset

Table 56 TT Interrupt Register (address 0x120)

**Bit 18 CER:** Configuration Error

Trigger out of order.

- 0= No error found in trigger list
- 1= Error found in trigger list

**Bit 17 AW:** Application Watchdog

- 0= Application watchdog served in time
- 1= Application watchdog not served in time

**Bit 16 WT:** Watch Trigger

- 0= No missing reference message
- 1= Missing reference message (Level 0: cycle time 0xFF00)

**Bit 15 IWT:** Initialization Watch Trigger

The initialization is restarted by resetting **IWT**.

- 0= No missing reference message during system startup
- 1= No system startup due to missing reference message

**Bit 14 ELC:** Error Level Changed

Not set when error level changed during initialization.

- 0= No change in error level
- 1= Error level changed

**Bit 13 SE2:** Scheduling Error 2

- 0= No scheduling error 2
- 1= Scheduling error 2 occurred

**Bit 12 SE1:** Scheduling Error 1

- 0= No scheduling error 1
- 1= Scheduling error 1 occurred

**Bit 11 TXO:** Tx Count Overflow

- 0= Number of Tx Trigger as expected
- 1= More Tx trigger than expected in one matrix cycle

**Bit 10 TXU:** Tx Count Underflow

- 0= Number of Tx Trigger as expected
- 1= Less Tx trigger than expected in one matrix cycle

**Bit 9 GTE:** Global Time Error

Synchronization deviation SD exceeds limit specified by **TTOCF.LDSL**, TTCAN Level 0,2 only.

- 0= Synchronization deviation within limit
- 1= Synchronization deviation exceeded limit

**Bit 8 GTD:** Global Time Discontinuity

- 0= No discontinuity of global time
- 1= Discontinuity of global time

**Bit 7 GTW:** Global Time Wrap

- 0= No global time wrap occurred
- 1= Global time wrap from 0xFFFF to 0x0000 occurred

**Bit 6 SWE:** Stop Watch Event

- 0= No rising/falling edge at stop watch trigger pin **m\_ttcanc\_swt** detected
- 1= Rising/falling edge at stop watch trigger pin **m\_ttcanc\_swt** detected

**Bit 5 TTMi:** Trigger Time Mark Event Internal

Internal time mark events are configured by trigger memory element **TMIN** (see Section 2.4.7). Set when the trigger memory element becomes active, and the M\_TTCAN is in synchronization state In\_Gap or In\_Schedule.

- 0= Time mark not reached
- 1= Time mark reached (Level 0: cycle time **TTOCF.IRTO** • 0x200)

**Bit 4 RTMI:** Register Time Mark Interrupt

Set when time referenced by **TTOCN.TMC** (cycle, local, or global) equals **TTTMK.TM**, independent of the synchronization state.

- 0= Time mark not reached
- 1= Time mark reached

**Bit 3 SOG:** Start of Gap

- 0= No reference message seen with Next\_is\_Gap bit set
- 1= Reference message with Next\_is\_Gap bit set becomes valid

**Bit 2 CSM:** Change of Synchronization Mode

- 0= No change in master to slave relation or schedule synchronization
- 1= Master to slave relation or schedule synchronization changed, also set when **TTOST.SPL** is reset

**Bit 1 SMC:** Start of Matrix Cycle

- 0= No Matrix Cycle started since bit has been reset
- 1= Matrix Cycle started

**Bit 0 SBC:** Start of Basic Cycle

- 0= No Basic Cycle started since bit has been reset
- 1= Basic Cycle started

### 2.3.56 TT Interrupt Enable (TTIE)

The settings in the TT Interrupt Enable register determine which status changes in the TT Interrupt Register will result in an interrupt.

- 0= TT interrupt disabled
- 1= TT interrupt enabled

<b>Bits</b>	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x124														CERE	AWE	WTE
														R-0	RW-0	RW-0

<b>Bits</b>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0										

R = Read, W = Write; -n = value after reset

Table 57 TT Interrupt Enable (address 0x124)

<b>Bit 18</b>	<b>CERE:</b>	Configuration Error Interrupt Enable
<b>Bit 17</b>	<b>AWE:</b>	Application Watchdog Interrupt Enable
<b>Bit 16</b>	<b>WTE:</b>	Watch Trigger Interrupt Enable
<b>Bit 15</b>	<b>IWTE:</b>	Initialization Watch Trigger Interrupt Enable
<b>Bit 14</b>	<b>ELCE:</b>	Change Error Level Interrupt Enable
<b>Bit 13</b>	<b>SE2E:</b>	Scheduling Error 2 Interrupt Enable
<b>Bit 12</b>	<b>SE1E:</b>	Scheduling Error 1 Interrupt Enable
<b>Bit 11</b>	<b>TXOE:</b>	Tx Count Overflow Interrupt Enable
<b>Bit 10</b>	<b>TXUE:</b>	Tx Count Underflow Interrupt Enable
<b>Bit 9</b>	<b>GTEE:</b>	Global Time Error Interrupt Enable
<b>Bit 8</b>	<b>GTDE:</b>	Global Time Discontinuity Interrupt Enable
<b>Bit 7</b>	<b>GTWE:</b>	Global Time Wrap Interrupt Enable
<b>Bit 6</b>	<b>SWEE:</b>	Stop Watch Event Interrupt Enable
<b>Bit 5</b>	<b>TTMIE:</b>	Trigger Time Mark Event Internal Enable
<b>Bit 4</b>	<b>RTMIE:</b>	Register Time Mark Interrupt Enable
<b>Bit 3</b>	<b>SOGE:</b>	Start of Gap Interrupt Enable
<b>Bit 2</b>	<b>CSME:</b>	Change of Synchronization Mode Interrupt Enable
<b>Bit 1</b>	<b>SMCE:</b>	Start of Matrix Cycle Interrupt Enable
<b>Bit 0</b>	<b>SBCE:</b>	Start of Basic Cycle Interrupt Enable

### 2.3.57 TT Interrupt Line Select (TTILS)

The TT Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the TT Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via **ILE.EINT0** and **ILE.EINT1**.

0= TT interrupt assigned to interrupt line **m\_ttcan\_int0**

1= TT interrupt assigned to interrupt line **m\_ttcan\_int1**

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x128	res													CERL	AWL	WTL
	R-0													RW-0	RW-0	RW-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IWTL	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	GTWL	SWEL	TTMIL	RTMIL	SOGL	CSML	SMCL	SBCL
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0										

R = Read, W = Write; -n = value after reset

Table 58 TT Interrupt Line Select (address 0x128)

<b>Bit 18</b>	<b>CERL:</b>	Configuration Error Interrupt Line
<b>Bit 17</b>	<b>AWL:</b>	Application Watchdog Interrupt Line
<b>Bit 16</b>	<b>WTL:</b>	Watch Trigger Interrupt Line
<b>Bit 15</b>	<b>IWTL:</b>	Initialization Watch Trigger Interrupt Line
<b>Bit 14</b>	<b>ELCL:</b>	Change Error Level Interrupt Line
<b>Bit 13</b>	<b>SE2L:</b>	Scheduling Error 2 Interrupt Line
<b>Bit 12</b>	<b>SE1L:</b>	Scheduling Error 1 Interrupt Line
<b>Bit 11</b>	<b>TXOL:</b>	Tx Count Overflow Interrupt Line
<b>Bit 10</b>	<b>TXUL:</b>	Tx Count Underflow Interrupt Line
<b>Bit 9</b>	<b>GTEL:</b>	Global Time Error Interrupt Line
<b>Bit 8</b>	<b>GTDL:</b>	Global Time Discontinuity Interrupt Line
<b>Bit 7</b>	<b>GTWL:</b>	Global Time Wrap Interrupt Line
<b>Bit 6</b>	<b>SWEL:</b>	Stop Watch Event Interrupt Line
<b>Bit 5</b>	<b>TTMIL:</b>	Trigger Time Mark Event Internal Line
<b>Bit 4</b>	<b>RTMIL:</b>	Register Time Mark Interrupt Line
<b>Bit 3</b>	<b>SOGL:</b>	Start of Gap Interrupt Line
<b>Bit 2</b>	<b>CSML:</b>	Change of Synchronization Mode Interrupt Line
<b>Bit 1</b>	<b>SMCL:</b>	Start of Matrix Cycle Interrupt Line
<b>Bit 0</b>	<b>SBCL:</b>	Start of Basic Cycle Interrupt Line

### 2.3.58 TT Operation Status (TTOST)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x12C	SPL	WECS	AWE	WFE	GSI		TMP[2:0]		GFI	WGTD			res			
	R-0	R-0	R-0	R-0	R-0		R-0		R-0	R-0			R-0			
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							RTO[7:0]		QCS	QGTP	SYS[1:0]		MS[1:0]		EL[1:0]	
							R-0		R-1	R-0	R-0		R-0		R-0	

R = Read, P = Protected write; -n = value after reset

Table 59 TT Operation Status (address 0x12C)

#### Bit 31 SPL: Schedule Phase Lock

The bit is valid only when external synchronization is enabled (**TTOCN.ESCN** = '1'). In this case it signals that the difference between cycle time configured by **TTGTP.CTP** and the cycle time at the rising edge at pin **m\_ttcanc\_evt** is less or equal 9 NTU (see Section 4.11).

0= Phase outside range

1= Phase inside range

#### Bit 30 WECS: Wait for External Clock Synchronization

0= No external clock synchronization pending

1= Node waits for external clock synchronization to take effect. The bit is reset at the start of the next basic cycle.

#### Bit 29 AWE: Application Watchdog Event

The application watchdog is served by reading TTOST. When the watchdog is not served in time, bit **AWE** is set, all TTCAN communication is stopped, and the M\_TTCAN is set into Bus Monitoring Mode.

0= Application Watchdog served in time

1= Failed to serve Application Watchdog in time

#### Bit 28 WFE: Wait for Event

0= No Gap announced, reset by a reference message with **Next\_is\_Gap** = '0'

1= Reference message with **Next\_is\_Gap** = '1' received

#### Bit 27 GSI: Gap Started Indicator

0= No Gap in schedule, reset by each reference message and for all time slaves

1= Gap time after Basic Cycle has started

#### Bit 26:24 TMP[2:0]: Time Master Priority

0x0-7 Priority of actual Time Master

#### Bit 23 GFI: Gap Finished Indicator

Set when the CPU writes **TTOCN.FGP**, or by a time mark interrupt if **TMG** = '1', or via input pin **m\_ttcanc\_evt** if **TTOCN.GCS** = '1'. Not set by Ref\_Trigger\_Gap or when Gap is finished by another node sending a reference message.

0= Reset at the end of each reference message

1= Gap finished by M\_TTCAN

**Bit 22        WGTD:** Wait for Global Time Discontinuity

- 0= No global time preset pending
- 1= Node waits for the global time preset to take effect. The bit is reset when the node has transmitted a reference message with **Disc\_Bit** = '1' or after it received a reference message.

**Bit 15:8      RTO[7:0]:** Reference Trigger Offset

The Reference Trigger Offset value is a signed integer with a range from -127 (0x81) to 127 (0x7F). There is no notification when the lower limit of -127 is reached. In case the M\_TTCAN becomes Time Master (**MS[1:0]** = "11"), the reset of **RTO** is delayed due to synchronization between Host and CAN clock domain. For time slaves the value configured by **TTOCF.IRTO** is read.

0x00-FF      Actual Reference Trigger offset value

**Bit 7        QCS:** Quality of Clock Speed

Only relevant in TTCAN Level 0 and Level 2, otherwise fixed to '1'.

- 0= Local clock speed not synchronized to Time Master clock speed
- 1= Synchronization Deviation ≤ **SDL**

**Bit 6        QGTP:** Quality of Global Time Phase

Only relevant in TTCAN Level 0 and Level 2, otherwise fixed to '0'.

- 0= Global time not valid
- 1= Global time in phase with Time Master

**Bit 5:4      SYS[1:0]:** Synchronization State

- 00= Out of Synchronization
- 01= Synchronizing to TTCAN communication
- 10= Schedule suspended by Gap (In\_Gap)
- 11= Synchronized to schedule (In\_Schedule)

**Bit 3:2      MS[1:0]:** Master State

- 00= Master\_Off, no master properties relevant
- 01= Operating as Time Slave
- 10= Operating as Backup Time Master
- 11= Operating as current Time Master

**Bit 1:0      EL[1:0]:** Error Level

- 00= Severity 0 - No Error
- 01= Severity 1 - Warning
- 10= Severity 2 - Error
- 11= Severity 3 - Severe Error

### 2.3.59 TUR Numerator Actual (TURNA)

There is no drift compensation in TTCAN Level 1 (**NAV = NC**). In TTCAN Level 0 and Level 2, the drift between the node's local clock and the time master's local clock is calculated. The drift is compensated when the Synchronisation Deviation (difference between **NC** and the calculated **NAV**) is not more than  $2^{(\text{TTOCF.LDSL} + 5)}$ . With  $\text{TTOCF.LDSL} \leq 7$ , this results in a maximum range for **NAV** of  
 $(\text{NC} - 0x1000) \leq \text{NAV} \leq (\text{NC} + 0x1000)$ .

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x130	res															NAV[17:16]
	R-0															R-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NAV[15:0]															R-0

R = Read, P = Protected write; -n = value after reset

Table 60 TUR Numerator Actual (address 0x130)

#### Bit 17:0 NAV[17:0]: Numerator Actual Value

- $\leq 0x0EFF$  Illegal value
- $0x0F000-20FFF$  Actual numerator value
- $\geq 0x21000$  Illegal value

### 2.3.60 TT Local & Global Time (TTLGT)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x134	GT[15:0]															R-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LT[15:0]															R-0

R = Read, P = Protected write; -n = value after reset

Table 61 TT Local & Global Time (address 0x134)

#### Bit 31:16 GT[15:0]: Global Time

- Non-fractional part of the sum of the node's local time and its local offset (see Section 4.5).
- $0x0000-FFFF$  Global time value of TTCAN network

#### Bit 15:0 LT[15:0]: Local Time

- Non-fractional part of local time, incremented once each local NTU (see Section 4.5).
- $0x0000-FFFF$  Local time value of TTCAN node

### 2.3.61 TT Cycle Time & Count (TTCTC)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x138	res														CC[5:0]	
	R-0														R-0x3F	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CT[15:0]														R-0	

R = Read, P = Protected write; -n = value after reset

Table 62 TT Cycle Time & Count (address 0x138)

**Bit 21:16 CC[5:0]:** Cycle Count

0x00-3F Number of actual Basic Cycle in the System Matrix

**Bit 15:0 CT[15:0]:** Cycle Time

Non-fractional part of the difference of the node's local time and Ref\_Mark (see Section 4.5).

0x0000-FFFF Cycle time value of TTCAN Basic Cycle

### 2.3.62 TT Capture Time (TTCPT)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x13C	SWV[15:0]														R-0	
	R-0														R-0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res														CCV[5:0]	

R = Read, P = Protected write; -n = value after reset

Table 63 TT Capture Time (address 0x13C)

**Bit 31:16 SWV[15:0]:** Stop Watch Value

On a rising/falling edge (as configured via **TTOCN.SWP**) at the Stop Watch Trigger pin **m\_ttcanc\_swt**, when **TTOCN.SWS** is ≠ "00" and **TTIR.SWE** is '0', the actual time value as selected by **TTOCN.SWS** (cycle, local, global) is copied to **SWV** and **TTIR.SWE** will be set to '1'. Capturing of the next stop watch value is enabled by resetting **TTIR.SWE**.

0x0000-FFFF Captured Stop Watch value

**Bit 5:0 CCV[5:0]:** Cycle Count Value

Cycle count value captured together with **SWV**.

0x00-3F Captured cycle count value

### 2.3.63 TT Cycle Sync Mark (TTCSM)

<i>Bits</i>	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x140																
	res															R-0
<i>Bits</i>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CSM[15:0]															R-0

R = Read, P = Protected write; -n = value after reset

Table 64 TT Cycle Sync Mark (address 0x140)

#### Bit 15:0 CSM[15:0]: Cycle Sync Mark

The Cycle Sync Mark is measured in cycle time. It is updated when the reference message becomes valid and retains its value until the next reference message becomes valid.

0x0000-FFFF Captured cycle time

## 2.4 Message RAM

For storage of Rx/Tx messages and for storage of the filter configuration a single- or dual-ported Message RAM has to be connected to the M\_TTCAN module.

### 2.4.1 Message RAM Configuration

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M\_TTCAN module can be configured to allocate up to 4480 words in the Message RAM. It is not necessary to configure each of the sections listed in Figure 2, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via **RXESC.F0DS**, **RXESC.F1DS**, **RXESC.RBDS**, and **TXESC.TBDS**.

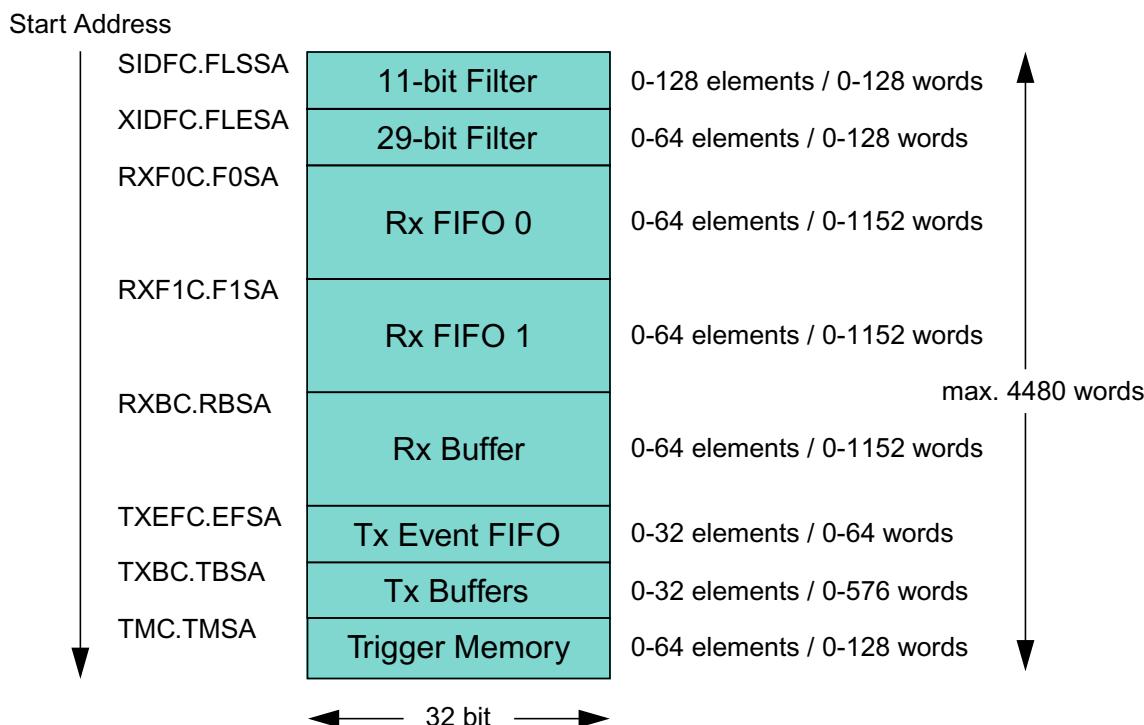


Figure 2 Message RAM Configuration

When the M\_TTCAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

**Note:** *The M\_TTCAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.*

## 2.4.2 Rx Buffer and FIFO Element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in Table 65 below. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register RXESC.

	31	24   23	16   15	8   7	0
R0	ESI XTD RTR	ID[28:0]			
R1	ANMF	FIDX[6:0]	res EDL BRS	DLC[3:0]	RXTS[15:0]
R2	DB3[7:0]	DB2[7:0]		DB1[7:0]	DB0[7:0]
R3	DB7[7:0]	DB6[7:0]		DB5[7:0]	DB4[7:0]
...	...	...		...	...
Rn	DBm[7:0]	DBm-1[7:0]		DBm-2[7:0]	DBm-3[7:0]

Table 65 Rx Buffer and FIFO Element

### R0 Bit 31 ESI: Error State Indicator

- 0= Transmitting node is error active
- 1= Transmitting node is error passive

### R0 Bit 30 XTD: Extended Identifier

Signals to the Host whether the received frame has a standard or extended identifier.

- 0= 11-bit standard identifier
- 1= 29-bit extended identifier

### R0 Bit 29 RTR: Remote Transmission Request

Signals to the Host whether the received frame is a data frame or a remote frame.

- 0= Received frame is a data frame
- 1= Received frame is a remote frame

**Note:** There are no remote frames in CAN FD format. In case a CAN FD frame was received (EDL = 1), bit RTR reflects the state of the reserved bit r1.

### R0 Bits 28:0 ID[28:0]: Identifier

Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

### R1 Bit 31 ANMF: Accepted Non-matching Frame

Acceptance of non-matching frames may be enabled via GFC.ANFS and GFC.ANFE.

- 0= Received frame matching filter index FIDX
- 1= Received frame did not match any Rx filter element

**R1 Bits 30:24 FIDX[6:0]: Filter Index**

0-127=Index of matching Rx acceptance filter element (invalid if **ANMF** = '1').  
Range is 0 to **SIDFC.LSS** - 1 resp. **XIDFC.LSE** - 1.

**R1 Bit 21 EDL: Extended Data Length**

- 0= Standard frame format
- 1= CAN FD frame format (new DLC-coding and CRC)

**R1 Bit 20 BRS: Bit Rate Switch**

- 0= Frame received without bit rate switching
- 1= Frame received with bit rate switching

**R1 Bits 19:16 DLC[3:0]: Data Length Code**

- 0-8= CAN + CAN FD: received frame has 0-8 data bytes
- 9-15= CAN: received frame has 8 data bytes
- 9-15= CAN FD: received frame has 12/16/20/24/32/48/64 data bytes

**R1 Bits 15:0 RXTS[15:0]: Rx Timestamp**

Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler **TSCC.TCP**.

**R2 Bits 31:24 DB3[7:0]: Data Byte 3****R2 Bits 23:16 DB2[7:0]: Data Byte 2****R2 Bits 15:8 DB1[7:0]: Data Byte 1****R2 Bits 7:0 DB0[7:0]: Data Byte 0****R3 Bits 31:24 DB7[7:0]: Data Byte 7****R3 Bits 23:16 DB6[7:0]: Data Byte 6****R3 Bits 15:8 DB5[7:0]: Data Byte 5****R3 Bits 7:0 DB4[7:0]: Data Byte 4**

...      ...      ...

**Rn Bits 31:24 DBm[7:0]: Data Byte m****Rn Bits 23:16 DBm-1[7:0]: Data Byte m-1****Rn Bits 15:8 DBm-2[7:0]: Data Byte m-2****Rn Bits 7:0 DBm-3[7:0]: Data Byte m-3**

**Note:** Depending on the configuration of the element size (**RXESC**), between two and sixteen 32-bit words ( $Rn = 3 \dots 17$ ) are used for storage of a CAN message's data field.

### 2.4.3 Tx Buffer Element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration **TXBC.TFQS** and **TXBC.NDTB**. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register TXESC.

	31	24	23	16	15	8	7	0
T0	res	XTD	RTR	ID[28:0]				
T1	MM[7:0]			EFC	res	DLC[3:0]	res	
T2	DB3[7:0]			DB2[7:0]		DB1[7:0]	DB0[7:0]	
T3	DB7[7:0]			DB6[7:0]		DB5[7:0]	DB4[7:0]	
...	...			...		...	...	
Tn	DBm[7:0]			DBm-1[7:0]		DBm-2[7:0]	DBm-3[7:0]	

Table 66 Tx Buffer Element

**T0 Bit 30 XTD:** Extended Identifier

- 0= 11-bit standard identifier
- 1= 29-bit extended identifier

**T0 Bit 29 RTR:** Remote Transmission Request

- 0= Transmit data frame
- 1= Transmit remote frame

**Note:** When RTR = 1, the M\_TTCAN transmits a remote frame according to ISO11898-1, even if CCCR.CME enables the transmission in CAN FD format.

**T0 Bits 28:0 ID[28:0]:** Identifier

Standard or extended identifier depending on bit **XTD**. A standard identifier has to be written to **ID[28:18]**.

**T1 Bits 31:24MM[7:0]:** Message Marker

Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.

**T1 Bit 23 EFC:** Event FIFO Control

- 0= Don't store Tx events
- 1= Store Tx events

**T1 Bits 19:16DLC[3:0]:** Data Length Code

0-8= CAN + CAN FD: transmit frame has 0-8 data bytes

9-15= CAN: transmit frame has 8 data bytes

9-15= CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes

**T2 Bits 31:24 DB3[7:0]:** Data Byte 3

**T2 Bits 23:16 DB2[7:0]:** Data Byte 2

**T2 Bits 15:8 DB1[7:0]:** Data Byte 1

**T2 Bits 7:0 DB0[7:0]:** Data Byte 0

**T3 Bits 31:24 DB7[7:0]:** Data Byte 7

**T3 Bits 23:16 DB6[7:0]:** Data Byte 6

**T3 Bits 15:8 DB5[7:0]:** Data Byte 5

**T3 Bits 7:0 DB4[7:0]:** Data Byte 4

...

...

...

**Tn Bits 31:24 DBm[7:0]:** Data Byte m

**Tn Bits 23:16 DBm-1[7:0]:** Data Byte m-1

**Tn Bits 15:8 DBm-2[7:0]:** Data Byte m-2

**Tn Bits 7:0 DBm-3[7:0]:** Data Byte m-3

*Note: Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 3 .. 17) are used for storage of a CAN message's data field.*

#### 2.4.4 Tx Event FIFO Element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register **TXEFS**.

	31	24   23			16   15		8   7	0
E0	ES 	XTD 	RTR 	ID[28:0]				
E1	MM[7:0]			ET[1:0]	EDL 	BRS 	DLC[3:0]	TXTS[15:0]

Table 67 Tx Event FIFO Element

**E0 Bit 31 ESI:** Error State Indicator

- 0= Transmitting node is error active
- 1= Transmitting node is error passive

**E0 Bit 30 XTD:** Extended Identifier

- 0= 11-bit standard identifier
- 1= 29-bit extended identifier

**E0 Bit 29 RTR:** Remote Transmission Request

- 0= Data frame transmitted
- 1= Remote frame transmitted

**E0 Bits 28:0 ID[28:0]:** Identifier

Standard or extended identifier depending on bit **XTD**. A standard identifier is stored into **ID[28:18]**.

**E1 Bits 31:24 MM[7:0]:** Message Marker

Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.

**E1 Bit 23:22 ET[1:0]:** Event Type

- 00= Reserved
- 01= Tx event
- 10= Transmission in spite of cancellation (always set for transmissions in DAR mode)
- 11= Reserved

**E1 Bit 21 EDL:** Extended Data Length

- 0= Standard frame format
- 1= CAN FD frame format (new DLC-coding and CRC)

**E1 Bit 20 BRS:** Bit Rate Switch

- 0= Frame received without bit rate switching
- 1= Frame received with bit rate switching

**E1 Bits 19:16 DLC[3:0]:** Data Length Code

- 0-8= CAN + CAN FD: frame with 0-8 data bytes transmitted
- 9-15= CAN: frame with 8 data bytes transmitted
- 9-15= CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted

**E1 Bits 15:0 TXTS[15:0]: Tx Timestamp**

Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler **TSCC.TCP**.

**2.4.5 Standard Message ID Filter Element**

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address **SIDFC.FLSSA** plus the index of the filter element (0...127).

	31	24   23	16   15	8   7	0
S0	SFT[1:0]	SFEC[2:0]	SFID1[10:0]	res	SFID2[10:0]

Table 68 Standard Message ID Filter Element

**Bits 31:30 SFT[1:0]: Standard Filter Type**

- 00= Range filter from **SF1ID** to **SF2ID** (**SF2ID**  $\geq$  **SF1ID**)
- 01= Dual ID filter for **SF1ID** or **SF2ID**
- 10= Classic filter: **SF1ID** = filter, **SF2ID** = mask
- 11= Reserved

**Bit 29:27 SFEC[2:0]: Standard Filter Element Configuration**

All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If **SFEC** = "100", "101", or "110" a match sets interrupt flag **IR.HPM** and, if enabled, an interrupt is generated. In this case register **HPMS** is updated with the status of the priority match.

- 000= Disable filter element
- 001= Store in Rx FIFO 0 if filter matches
- 010= Store in Rx FIFO 1 if filter matches
- 011= Reject ID if filter matches
- 100= Set priority if filter matches
- 101= Set priority and store in FIFO 0 if filter matches
- 110= Set priority and store in FIFO 1 if filter matches
- 111= Store into Rx Buffer or as debug message, configuration of **SFT[1:0]** ignored

**Bits 26:16 SFID1[10:0]: Standard Filter ID 1**

First ID of standard ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.

**Bits 10:0 SFID2[10:0]: Standard Filter ID 2**

This bit field has a different meaning depending on the configuration of **SFEC**:

- 1) **SFEC** = "001"..."110" Second ID of standard ID filter element
- 2) **SFEC** = "111" Filter for Rx Buffers or for debug messages

**SFID2[10:9]** decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.

- 00= Store message into an Rx Buffer
- 01= Debug Message A
- 10= Debug Message B
- 11= Debug Message C

**SFID2[8:6]** is used to control the filter event pins **m\_ttcan\_fe[2:0]** at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one **m\_ttcan\_hclk** period in case the filter matches.

**SFID2[5:0]** defines the offset to the Rx Buffer Start Address **RXBC.RBSA** for storage of a matching message.

#### 2.4.6 Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address **XIDFC.FLESA** plus two times the index of the filter element (0...63).

	31	24   23	16   15	8   7	0
F0	EFEC[2:0]			EFID1[28:0]	
F1	EFT[1:0]	res		EFID2[28:0]	

Table 69      *Extended Message ID Filter Element*

##### F0 Bit 31:29      EFEC[2:0]: Extended Filter Element Configuration

All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If **EFEC** = "100", "101", or "110" a match sets interrupt flag **IR.HPM** and, if enabled, an interrupt is generated. In this case register **HPMS** is updated with the status of the priority match.

- 000= Disable filter element
- 001= Store in Rx FIFO 0 if filter matches
- 010= Store in Rx FIFO 1 if filter matches
- 011= Reject ID if filter matches
- 100= Set priority if filter matches
- 101= Set priority and store in FIFO 0 if filter matches
- 110= Set priority and store in FIFO 1 if filter matches
- 111= Store into Rx Buffer or as debug message, configuration of **EFT[1:0]** ignored

##### F0 Bits 28:0      EFID1[28:0]: Extended Filter ID 1

First ID of extended ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only **XIDAM** masking mechanism (see Section 3.4.1.5) is used.

**F1 Bits 31:30      EFT[1:0]:** Extended Filter Type

- 00= Range filter from **EF1ID** to **EF2ID** (**EF2ID**  $\geq$  **EF1ID**)
- 01= Dual ID filter for **EF1ID** or **EF2ID**
- 10= Classic filter: **EF1ID** = filter, **EF2ID** = mask
- 11= Range filter from **EF1ID** to **EF2ID** (**EF2ID**  $\geq$  **EF1ID**), XIDAM mask not applied

**F1 Bits 28:0      EFID2[28:0]:** Extended Filter ID 2

This bit field has a different meaning depending on the configuration of **EFEC**:

- 1) **EFEC** = "001"..."110"    Second ID of extended ID filter element
- 2) **EFEC** = "111"                Filter for Rx Buffers or for debug messages

**EFID2[10:9]** decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.

- 00= Store message into an Rx Buffer
- 01= Debug Message A
- 10= Debug Message B
- 11= Debug Message C

**EFID2[8:6]** is used to control the filter event pins **m\_ttcan\_fe[2:0]** at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one **m\_ttcan\_hclk** period in case the filter matches.

**EFID2[5:0]** defines the offset to the Rx Buffer Start Address **RXBC.RBSA** for storage of a matching message.

#### 2.4.7 Trigger Memory Element

Up to 64 trigger memory elements can be configured. When accessing a Trigger Memory element, its address is the Trigger Memory Start Address **TTTMC.TMSA** plus the index of the trigger memory element (0...63).

	31	24	23	16	15	8	7	0	
T0	TM[15:0]			res	CC[6:0]	ASC[1:0]	TMIN	TMEX	TYPE[3:0]
T1	res	FTYPE	MNR[6:0]	res				MSC[2:0]	

Table 70 Trigger Memory Element

##### T0 Bit 31:16 TM[15:0]: Time Mark

Cycle time for which the trigger becomes active.

##### T0 Bit 14:8 CC[6:0]: Cycle Code

Cycle count for which the trigger is valid. Ignored for trigger types Tx\_Ref\_Trigger, Tx\_Ref\_Trigger\_Gap, Watch\_Trigger, Watch\_Trigger\_Gap, End\_of\_List.

0b000000x valid for all cycles

0b000001c valid every 2nd cycle at cycle count mod2 = c

0b00001cc valid every 4th cycle at cycle count mod4 = cc

0b0001ccc valid every 8th cycle at cycle count mod8 = ccc

0b001cccc valid every 16th cycle at cycle count mod16 = cccc

0b01cccc valid every 32nd cycle at cycle count mod32 = ccccc

0b1cccccc valid every 64th cycle at cycle count mod64 = ccccccc

##### T0 Bit 7:6 ASC[1:0]: Asynchronous Serial Communication

00= No ASC operation

01= Reserved, do not use

10= Node is ASC receiver

11= Node is ASC transmitter

##### T0 Bit 5 TMIN: Time Mark Event Internal

0= No action

1= TTIR.TTMI is set when trigger memory element becomes active

##### T0 Bit 4 TMEX: Time Mark Event External

0= No action

1= Pulse at output **m\_ttcanc\_tmp** with the length of one **m\_ttcanc\_cclk** period is generated when the time mark of the trigger memory element becomes active and **TTOCN.TTMIE** = '1'

**T0 Bit 3:0 TYPE[3:0]: Trigger Type**

- 0000=Tx\_Ref\_Trigger - valid when not in Gap
- 0001=Tx\_Ref\_Trigger\_Gap - valid when in Gap
- 0010=Tx\_Trigger\_Single - starts a single transmission in an exclusive time window
- 0011=Tx\_Trigger\_Continous - starts continuous transmission in an exclusive time window
- 0100=Tx\_Trigger\_Arbitration - starts a transmission in an arbitrating time window
- 0101=Tx\_Trigger\_Merged - starts a merged arbitration window
- 0110=Watch\_Trigger - valid when not in Gap
- 0111=Watch\_Trigger\_Gap - valid when in Gap
- 1000=Rx\_Trigger - check for reception
- 1001=Time\_Base\_Trigger - only control **TMIN**, **TMEX**, and **ASC**
- 1010...1111=End\_of\_List - illegal type, causes config error

**Note:** For ASC operation (ASC = "10", "11") only trigger types Rx\_Trigger and Time\_Base\_Trigger should be used.

**T1 Bit 23 FTYPE: Filter Type**

- 0= 11-bit standard message ID
- 1= 29-bit extended message ID

**T1 Bit 22:16 MNR[6:0]: Message Number**

Transmission: Trigger is valid for configured Tx Buffer number. Valid values are 0 to 31.

Reception: Trigger is valid for standard / extended message ID filter element number. Valid values are 0 to 63 resp. 0 to 127.

**T1 Bits 2:0 MSC[2:0]: Message Status Count**

Counts scheduling errors for periodic messages in exclusive time windows. It has no function for arbitrating messages and in event-driven CAN communication (ISO11898-1).

0-7= Actual status

**Note:** The trigger memory elements have to be written when the M\_TTCAN is in INIT state. Write access to the trigger memory elements outside INIT state is not allowed.

*There is an exception for TMIN and TMEX when they are defined as part of a trigger memory element of TYPE Tx\_Ref\_Trigger. In this case they become active at the time mark modified by the actual Reference Trigger Offset (TTOST.RTO).*



# Chapter 3.

## 3. Functional Description

### 3.1 Operating Modes

#### 3.1.1 Software Initialization

Software initialization is started by setting bit **CCCR.INIT**, either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going *Bus\_Off*. While **CCCR.INIT** is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output **m\_ttcanc\_tx** is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting **CCCR.INIT** does not change any configuration register. Resetting **CCCR.INIT** finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits ( $\equiv$  *Bus\_Idle*) before it can take part in bus activities and start the message transfer.

Access to the M\_TTCAN configuration registers is only enabled when both bits **CCCR.INIT** and **CCCR.CCE** are set (protected write).

**CCCR.CCE** can only be set/reset while **CCCR.INIT** = '1'. **CCCR.CCE** is automatically reset when **CCCR.INIT** is reset.

The following registers are reset when **CCCR.CCE** is set

- **HPMS** - High Priority Message Status
- **RXF0S** - Rx FIFO 0 Status
- **RXF1S** - Rx FIFO 1 Status
- **TXFQS** - Tx FIFO/Queue Status
- **TXBRP** - Tx Buffer Request Pending
- **TXBTO** - Tx Buffer Transmission Occurred
- **TXBCF** - Tx Buffer Cancellation Finished
- **TXEFS** - Tx Event FIFO Status
- **TTOST** - TT Operation Status
- **TTLGT** - TT Local & Global Time, only Global Time **TTLGT.GT** is reset
- **TTCTC** - TT Cycle Time & Count
- **TTCSM** - TT Cycle Sync Mark

The Timeout Counter value **TOCV.TOC** is preset to the value configured by **TOCC.TOP** when **CCCR.CCE** is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while **CCCR.CCE** = '1'.

The following registers are only writeable while **CCCR.CCE** = '0'

- **TXBAR** - Tx Buffer Add Request
- **TXBCR** - Tx Buffer Cancellation Request

**CCCR.TEST** and **CCCR.MON** can only be set by the Host while **CCCR.INIT** = '1' and **CCCR.CCE** = '1'. Both bits may be reset at any time. **CCCR.DAR** can only be set/reset while **CCCR.INIT** = '1' and **CCCR.CCE** = '1'.

### 3.1.2 Normal Operation

The M\_TTCAN's default operating mode after hardware reset is event-driven CAN communication without time triggers (**TTOCF.OM** = "00"). It is required that both **CCCR.INIT** and **CCCR.CCE** are set before the TT Operation Mode can be changed.

Once the M\_TTCAN is initialized and **CCCR.INIT** is reset to zero, the M\_TTCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

### 3.1.3 CAN FD Operation

There are two variants in the CAN FD frame format, first the CAN FD frame without bit rate switching where the data field of a CAN frame may be longer than 8 bytes. The second variant is the CAN FD frame where control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame.

The CAN operation mode is enabled by programming **CCCR.CME**. In case **CCCR.CME** = "01" transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With **CCCR.CME** = "10"/"11" transmission and reception of long and fast CAN FD frames is enabled. **CCCR.CME** can only be changed while **CCCR.INIT** and **CCCR.CCE** are both set.

When initialization is left (**CCCR.INIT** set to '0'), the CAN FD protocol option is inactive, it has to be requested by writing to **CCCR.CMR**.

A mode change requested by writing to **CCCR.CMR** will be executed next time the CAN protocol controller FSM reaches idle phase between CAN frames. Upon this event **CCCR.CMR** is reset to "00" and the status flags **CCCR.FDBS** and **CCCR.FDO** are set accordingly. In case the requested CAN operation mode is not enabled, the value written to **CCCR.CMR** is retained until it is overwritten by the next mode change request. Default is CAN operation according to ISO11898-1.

It is not necessary to change the CAN operation mode after system startup. A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting according to ISO11898-1 until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in silent mode until programming has completed. Then all nodes switch back to CAN communication according ISO11898-1.

When **CCCR.CME** ≠ "00", received CAN FD frames are interpreted according to the CAN FD Protocol Specification. The reserved bit in CAN frames with 11-bit identifiers and the first reserved bit in CAN frames with 29-bit identifiers will be decoded as **EDL** bit. **EDL** = *recessive* signifies a CAN FD frame, **EDL** = *dominant* signifies a standard CAN frame. In a CAN FD frame, the two bits following **EDL**, **r0** and **BRS**, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by **r0** = *dominant* and **BRS** = *recessive*. The coding of **r0** = *recessive* is reserved for future expansion of the protocol.

Reception of CAN frames according to ISO 11898-1 is possible in all CAN operation modes.

The status bits **CCCR.FDO** and **CCCR.FDBS** indicate the format of transmitted frames. When **CCCR.FDO** is set, frames will be transmitted in CAN FD format with **EDL** = *recessive*. When both **CCCR.FDO** and **CCCR.FDBS** are set, frames will be transmitted in CAN FD format with bit rate switching and both bits **EDL** and **BRS** = *recessive*.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to Table 71 below.

DLC	9	10	11	12	13	14	15
Number of Data Bytes	12	16	20	24	32	48	64

Table 71 Coding of DLC in CAN FD

In CAN FD frames, the bit timing will be switched inside the frame, after the **BRS** (Bit Rate Switch) bit, if this bit is *recessive*. Before the **BRS** bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Bit Timing & Prescaler Register **BTP**. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the Fast Bit Timing & Prescaler Register **FBTP**. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency (**m\_ttcanc\_cclk**). Example: with a CAN clock frequency of 20MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit **ESI** (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, **ESI** is transmitted *recessive*, else it is transmitted *dominant*.

### 3.1.4 Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin **m\_ttcanc\_tx** the protocol controller receives the transmitted data from its local CAN transceiver via pin **m\_ttcanc\_rx**. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transceiver loop delay, the delay compensation is introduced. Without transceiver delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

#### 3.1.4.1 Description

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver. Figure 3 below describes how the transceiver loop delay is measured.

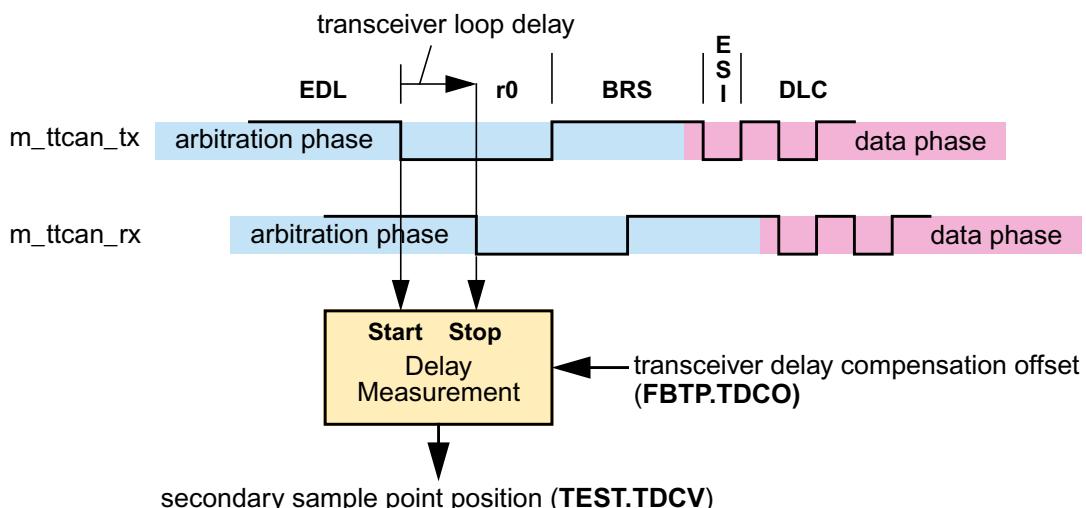


Figure 3 Transceiver delay measurement

Within each CAN FD frame, the transmitter measures the delay between the data transmitted at pin **m\_ttcn\_tx** and the data received at pin **m\_ttcn\_rx**. The measurement is done once, at the falling edge of bit **EDL** to bit **r0**. The delay is measured in **m\_ttcn\_cclk** periods.

A secondary sample point position is calculated by adding a configurable transceiver delay compensation offset **FBTP.TDCO** to the measured transceiver delay. This transceiver delay compensation value **TEST.TDCV** is the sum of the measured transceiver delay and the transceiver delay compensation offset. The transceiver delay compensation offset is chosen to adjust the secondary sample point inside the bit time (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of time quanta  $t_q$ .

To check for bit errors during the data phase, the delayed transmit data is compared against the received data at the secondary sample point. If a bit error is detected at the secondary sample point, the transmitter will react to this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

For the transceiver delay compensation the following boundary conditions have to be considered:

- The sum of the measured delay from **m\_ttcn\_tx** to **m\_ttcn\_rx** and the configured transceiver delay compensation offset **FBTP.TDCO** has to be less than 3 bit times in the data phase.
- The sum of the measured delay from **m\_ttcn\_tx** to **m\_ttcn\_rx** and the configured transceiver delay compensation offset **FBTP.TDCO** has to be less or equal 63 **m\_ttcn\_cclk** periods. In case this sum exceeds 63 **m\_ttcn\_cclk** periods, the maximum value of 63 **m\_ttcn\_cclk** periods is used for transceiver delay compensation.

The actual delay compensation value is monitored by reading **TEST.TDCV**.

### 3.1.4.2 Configuration and Status

Compensation for the transceiver loop delay by the M\_TTCAN is enabled via **FBTP.TDC**. The transceiver delay compensation offset is configured via **FBTP.TDCO**. The actual delay compensation value applied by the M\_TTCAN's protocol engine can be read from **TEST.TDCV**.

### 3.1.5 Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. The Host can set the M\_TTCAN into Restricted Operation mode by setting bit **CCCR.ASM**. The bit can only be set by the Host when both **CCCR.CCE** and **CCCR.INIT** are set to '1'. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset **CCCR.ASM**.

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

When the M\_TTCAN is configured for Asynchronous Serial Communication (see Section 4.10), the Host has to set **CCCR.ASM** during initialization to start with an ASC window. The bit is reset at the end of the first ASC window after the M\_TTCAN has finished initializing. During time-triggered operation **CCCR.ASM** is set by the M\_TTCAN at the beginning of an ASC window (trigger memory element with **T0.ASC** = "10", "11"). It is reset by each trigger memory element with **T0.ASC** = "00".

If the M\_TTCAN is connected to a Clock Calibration on CAN unit, **CCCR.ASM** is controlled by input **m\_ttcn\_cok**. In case **m\_ttcn\_cok** switches to '0', bit **CCCR.ASM** is set. When **m\_ttcn\_cok** switches back to '1', bit **CCCR.ASM** returns to the previously written value. The state of **CCCR.ASM** is the written value while input **m\_ttcn\_cok** is at '1'. The input is hardwired to '1' when there is no Clock Calibration on CAN unit connected.

### 3.1.6 Bus Monitoring Mode

The M\_TTCAN is set in Bus Monitoring Mode by programming **CCCR.MON** to *one or* when error level S3 (**TTOST.EL** = "11") is entered. In Bus Monitoring Mode (see ISO11898-1, 10.12 Bus monitoring), the M\_TTCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the M\_TTCAN is required to send a *dominant* bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M\_TTCAN monitors this *dominant* bit, although the CAN bus may remain in *recessive* state. In Bus Monitoring Mode register **TXBRP** is held in reset state.

The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of *dominant* bits. Figure 5 shows the connection of signals **m\_ttcanc\_tx** and **m\_ttcanc\_rx** to the M\_TTCAN in Bus Monitoring Mode.

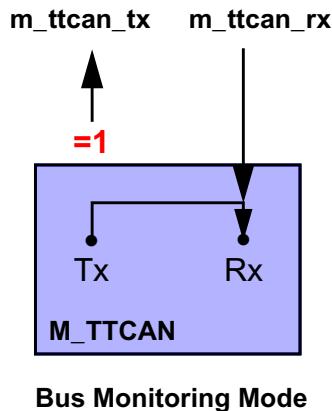


Figure 4 Pin Control in Bus Monitoring Mode

### 3.1.7 Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the M\_TTCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via **CCCR.DAR**.

#### 3.1.7.1 Frame Transmission in DAR Mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit **TXBRP.TRPx** is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:  
Corresponding Tx Buffer Transmission Occurred bit **TXBTO.TOx** set  
Corresponding Tx Buffer Cancellation Finished bit **TXBCF.CFx** not set
- Successful transmission in spite of cancellation:  
Corresponding Tx Buffer Transmission Occurred bit **TXBTO.TOx** set  
Corresponding Tx Buffer Cancellation Finished bit **TXBCF.CFx** set
- Arbitration lost or frame transmission disturbed:  
Corresponding Tx Buffer Transmission Occurred bit **TXBTO.TOx** not set  
Corresponding Tx Buffer Cancellation Finished bit **TXBCF.CFx** set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type **ET** = "10" (transmission in spite of cancellation).

### 3.1.8 Power Down (Sleep Mode)

The M\_TTCAN can be set into power down mode controlled by input signal **m\_ttcanc\_clkstop\_req** or via CC Control Register **CCCR.CSR**. As long as the clock stop request signal **m\_ttcanc\_clkstop\_req** is active, bit **CCCR.CSR** is read as *one*.

When all pending transmission requests have completed, the M\_TTCAN waits until bus idle state is detected. Then the M\_TTCAN sets then **CCCR.INIT** to *one* to prevent any further CAN transfers. Now the M\_TTCAN acknowledges that it is ready for power down by setting output signal **m\_ttcanc\_clkstop\_ack** to *one* and **CCCR.CSA** to *one*. In this state, before the clocks are switched off, further register accesses can be made. A write access to **CCCR.INIT** will have no effect. Now the module clock inputs **m\_ttcanc\_hclk** and **m\_ttcanc\_cclk** may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting signal **m\_ttcanc\_clkstop\_req** resp. CC Control Register flag **CCCR.CSR**. The M\_TTCAN will acknowledge this by resetting output signal **m\_ttcanc\_clkstop\_ack** and resetting **CCCR.CSA**. Afterwards, the application can restart CAN communication by resetting bit **CCCR.INIT**.

### 3.1.9 Test Modes

To enable write access to register **TEST** (see Section 2.3.5), bit **CCCR.TEST** has to be set to *one*. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin **m\_ttcanc\_tx** by programming **TEST.TX**. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M\_TTCAN's bit timing and it can drive constant dominant or recessive values. The actual value at pin **m\_ttcanc\_rx** can be read from **TEST.RX**. Both functions can be used to check the CAN bus' physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to **TEST.TX** until the new configuration is visible at output pin **m\_ttcanc\_tx**. This applies also when reading input pin **m\_ttcanc\_rx** via **TEST.RX**.

**Note:** *Test modes should be used for production tests or self test only. The software control for pin m\_ttcanc\_tx interferes with all CAN protocol functions. It is not recommended to use test modes for application.*

#### 3.1.9.1 External Loop Back Mode

The M\_TTCAN can be set in External Loop Back Mode by programming **TEST.LBCK** to *one*. In Loop Back Mode, the M\_TTCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx Buffer or an Rx FIFO. Figure 5 shows the connection of signals **m\_ttcanc\_tx** and **m\_ttcanc\_rx** to the M\_TTCAN in External Loop Back Mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M\_TTCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode. In this mode the M\_TTCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the **m\_ttcanc\_rx** input pin is disregarded by the M\_TTCAN. The transmitted messages can be monitored at the **m\_ttcanc\_tx** pin.

#### 3.1.9.2 Internal Loop Back Mode

Internal Loop Back Mode is entered by programming bits **TEST.LBCK** and **CCCR.MON** to *one*. This mode can be used for a “Hot Selftest”, meaning the M\_TTCAN can be tested without affecting a running CAN system connected to the pins **m\_ttcanc\_tx** and **m\_ttcanc\_rx**. In this mode pin **m\_ttcanc\_rx** is disconnected from the M\_TTCAN and pin **m\_ttcanc\_tx** is held recessive. Figure 5 shows the connection of **m\_ttcanc\_tx** and **m\_ttcanc\_rx** to the M\_TTCAN in case of Internal Loop Back Mode.

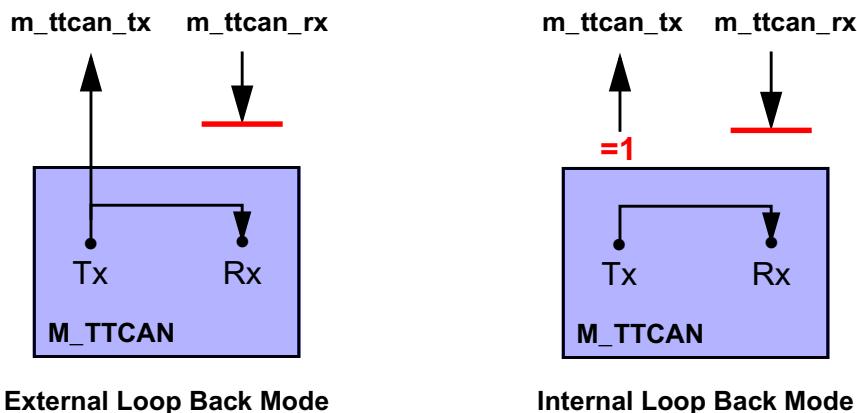


Figure 5 Pin Control in Loop Back Modes

### 3.1.10 Application Watchdog

The application watchdog is served by reading register **TTOST**. When the application watchdog is not served in time, bit **TTOST.AWE** is set, all TTCAN communication is stopped, and the M\_TTCAN is set into Bus Monitoring Mode.

The TT Application Watchdog can be disabled by programming the Application Watchdog Limit **TTOCF.AWL** to 0x00. The TT Application Watchdog should not be disabled in a TTCAN application program.

## 3.2 Timestamp Generation

For timestamp generation the M\_TTCAN supplies a 16-bit wrap-around counter. A prescaler **TSCC.TCP** can be configured to clock the counter in multiples of CAN bit times (1...16). The counter is readable via **TSCV.TCV**. A write access to register **TSCV** resets the counter to zero. When the timestamp counter wraps around interrupt flag **IR.TSW** is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer / Rx FIFO (**RXTS[15:0]**) or Tx Event FIFO (**TXTS[15:0]**) element.

By programming bit **TSCC.TSS** an external 16-bit timestamp can be used.

## 3.3 Timeout Counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M\_TTCAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by **TSCC.TCP** as the Timestamp Counter. The Timeout Counter is configured via register **TOCC**. The actual counter value can be read from **TOCV.TOC**.

The Timeout Counter can only be started while **CCCR.INIT** = '0'. It is stopped when **CCCR.INIT** = '1', e.g. when the M\_TTCAN enters *Bus\_Off* state.

The operation mode is selected by **TOCC.TOS**. When operating in Continuous Mode, the counter starts when **CCCR.INIT** is reset. A write to **TOCV** presets the counter to the value configured by **TOCC.TOP** and continues down-counting.

When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by **TOCC.TOP**. Down-counting is started when the first FIFO element is stored. Writing to **TOCV** has no effect.

When the counter reaches zero, interrupt flag **IR.TOO** is set. In Continuous Mode, the counter is immediately restarted at **TOCC.TOP**.

**Note:** The clock signal for the Timeout Counter is derived from the CAN Core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN Core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

## 3.4 Rx Handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

### 3.4.1 Acceptance Filtering

The M\_TTCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration **GFC**
- Standard ID Filter Configuration **SIDFC**
- Extended ID Filter Configuration **XIDFC**
- Extended ID AND Mask **XIDAM**

Depending on the configuration of the filter element (**SFEC/EFEC**) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag **IR.HPM**
- Set High Priority Message interrupt flag **IR.HPM** and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

Rx Buffer

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see **PSR.LEC** respectively **PSR.FLEC**.

### Rx FIFO

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see **PSR.LEC** respectively **PSR.FLEC**. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in Section 3.4.2.2 have to be considered.

**Note:** When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

#### 3.4.1.1 Range Filter

The filter matches for all received frames with Message IDs in the range defined by **SF1ID/SF2ID** resp. **EF1ID/EF2ID**.

There are two possibilities when range filtering is used together with extended frames:

**EFT = “00”:** The Message ID of received frames is ANDed with the Extended ID AND Mask (**XIDAM**) before the range filter is applied

**EFT = “11”:** The Extended ID AND Mask (**XIDAM**) is not used for range filtering

#### 3.4.1.2 Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with **SF1ID = SF2ID** resp. **EF1ID = EF2ID**.

#### 3.4.1.3 Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering **SF1ID/EF1ID** is used as Message ID filter, while **SF2ID/EF2ID** is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

### 3.4.1.4 Standard Message ID Filtering

Figure 6 below shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in Section 2.4.5.

Controlled by the Global Filter Configuration **GFC** and the Standard ID Filter Configuration **SIDFC** Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

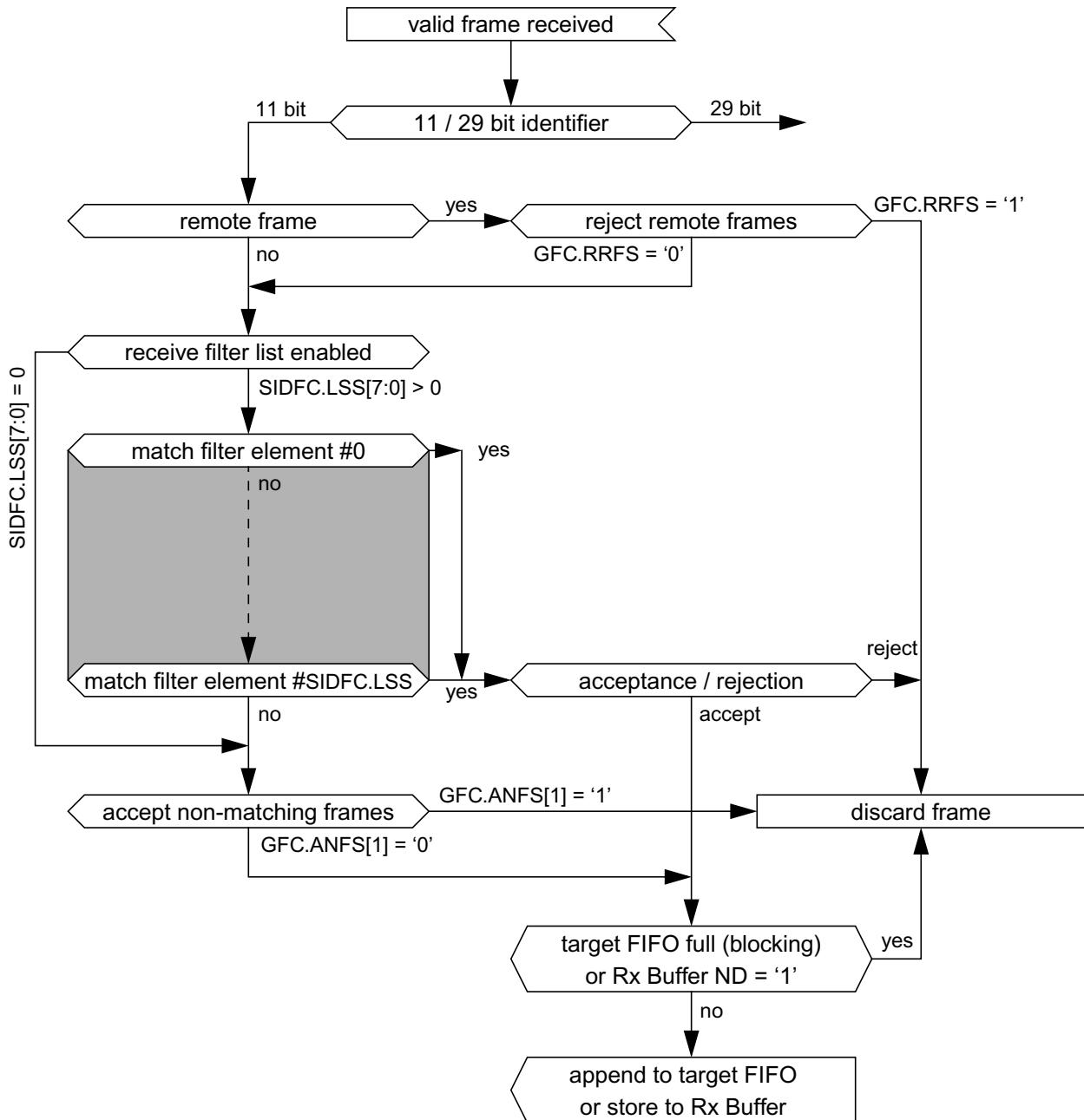


Figure 6 Standard Message ID Filter Path

### 3.4.1.5 Extended Message ID Filtering

Figure 7 below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in Section 2.4.6.

Controlled by the Global Filter Configuration **GFC** and the Extended ID Filter Configuration **XIDFC** Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The Extended ID AND Mask **XIDAM** is ANDed with the received identifier before the filter list is executed.

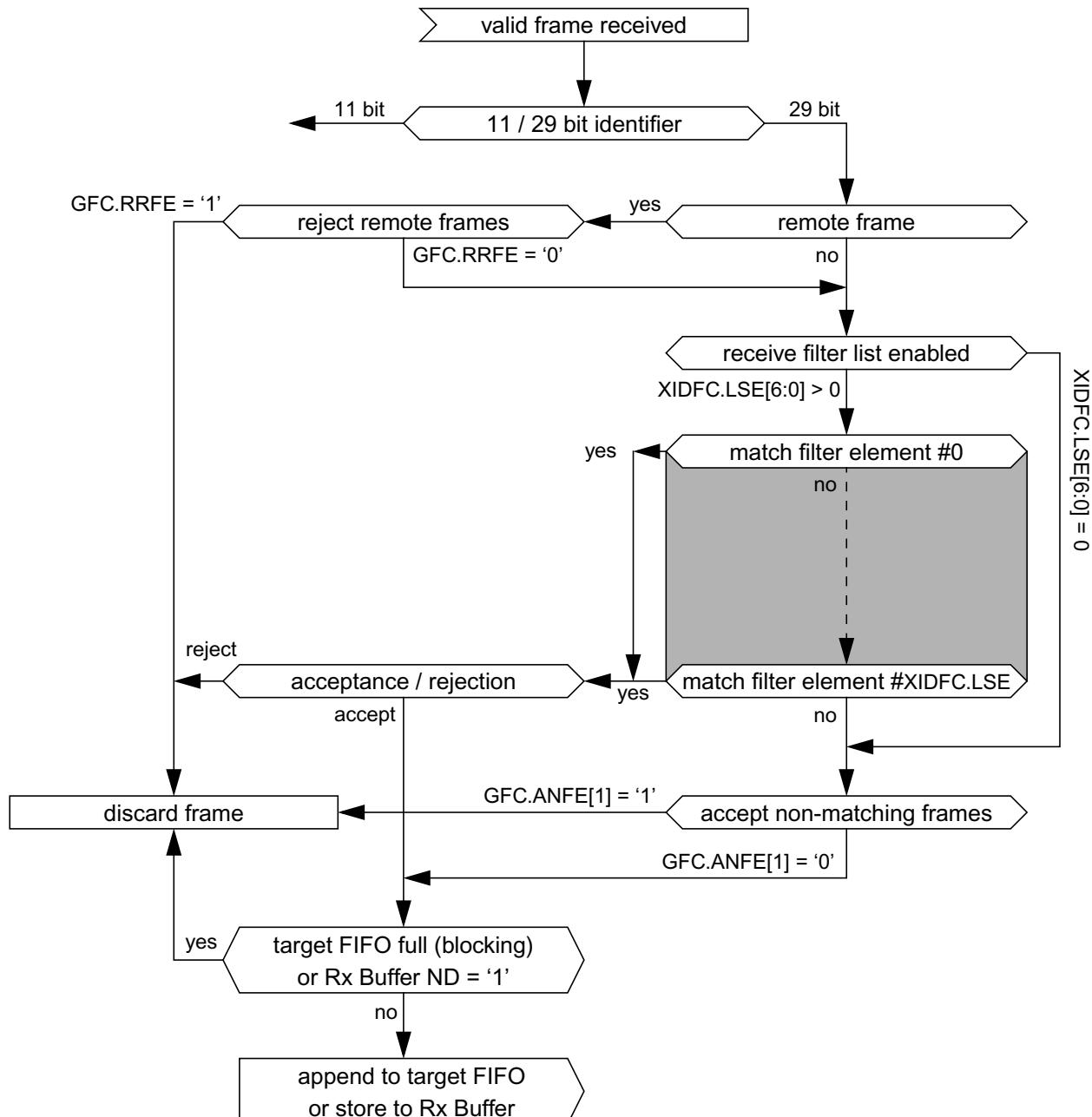


Figure 7 Extended Message ID Filter Path

### 3.4.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers **RXF0C** and **RXF1C**.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see Section 3.4.1. The Rx FIFO element is described in Section 2.4.2.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by **RXFnC.FnWM**, interrupt flag **IR.RFnW** is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by **RXFnS.FnF**. In addition interrupt flag **IR.RFnF** is set.

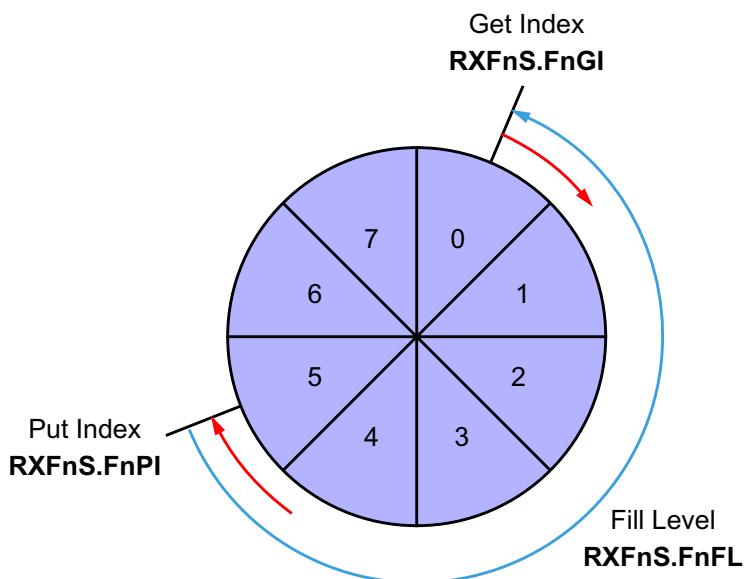


Figure 8 Rx FIFO Status

When reading from an Rx FIFO, Rx FIFO Get Index **RXFnS.FnGI** • FIFO Element Size has to be added to the corresponding Rx FIFO start address **RXFnC.FnSA**.

<b>RXESC.RBDS[2:0]</b> <b>RXESC.FnDS[2:0]</b>	<b>Data Field [bytes]</b>	<b>FIFO Element Size [RAM words]</b>
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

Table 72 Rx Buffer / FIFO Element Size

### 3.4.2.1 Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by **RXFnC.FnOM = '0'**. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (**RXFnS.FnPI = RXFnS.FnGI**), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by **RXFnS.FnF = '1'**. In addition interrupt flag **IR.RFnF** is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by **RXFnS.RFnL = '1'**. In addition interrupt flag **IR.RFnL** is set.

### 3.4.2.2 Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by **RXFnC.FnOM = '1'**.

When an Rx FIFO full condition (**RXFnS.FnPI = RXFnS.FnGI**) is signalled by **RXFnS.FnF = '1'**, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. Figure 9 shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

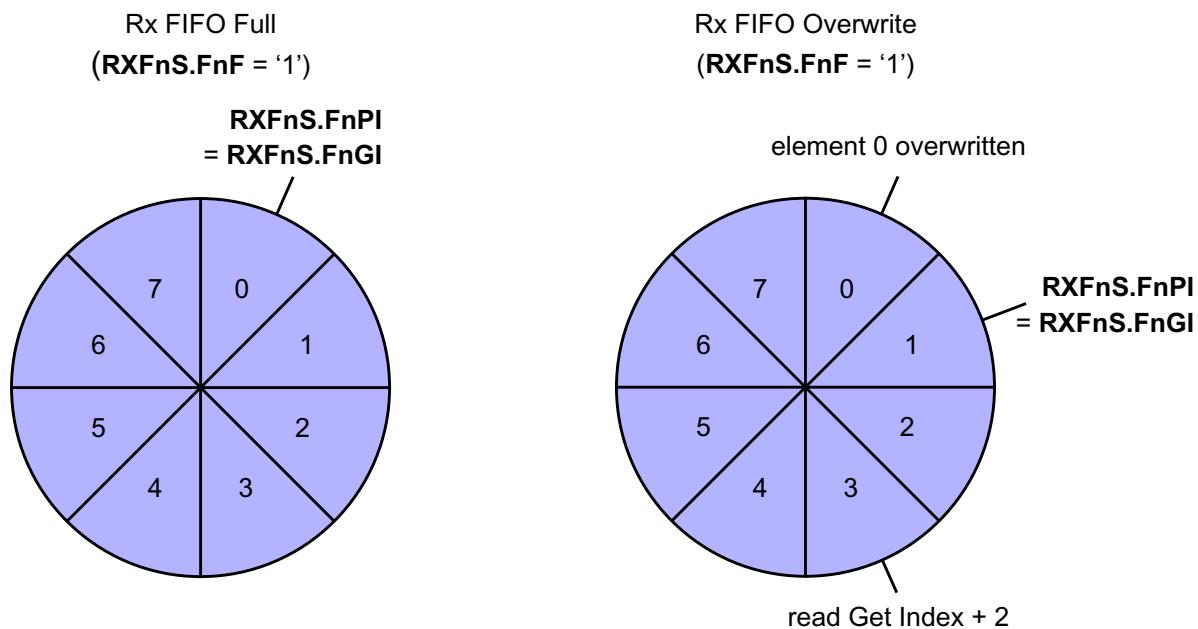


Figure 9 Rx FIFO Overflow Handling

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index **RXFnA.FnA**. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (**RXFnS.FnF = '0'**).

### 3.4.3 Dedicated Rx Buffers

The M\_TTCAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via **RXBC.RBSA**.

For each Rx Buffer a Standard or Extended Message ID Filter Element with **SFEC / EFEC = "111"** and **SFID2 / EFID2[10:9] = "00"** has to be configured (see Section 2.4.5 and Section 2.4.6).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag **IR.DRX** (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

<b>Filter Element</b>	<b>SFID1[10:0] EFID1[28:0]</b>	<b>SFID2[10:9] EFID2[10:9]</b>	<b>SFID2[5:0] EFID2[5:0]</b>
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

Table 73 Example Filter Configuration for Rx Buffers

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a '1' to the respective bit position.

While an Rx Buffer's New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

#### 3.4.3.1 Rx Buffer Handling

- Reset interrupt flag **IR.DRX**
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

### 3.4.4 Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (e.g. #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element (see M\_TTCAN User's Manual section 2.4.2).

Advantage: Fixed start address for the DMA transfers (relative to **RXBC.RBSA**), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with **SFEC / EFEC = "111"** have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by **SFID2 / EFID2[5:0]**.

After message C has been stored, the DMA request output **m\_ttcanc\_dma\_req** is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the M\_TTCAN while **m\_ttcanc\_dma\_req** is activated. The behaviour is similar to that of an Rx Buffers with its New Data flag set.

After the DMA has completed the DMA unit sets **m\_ttcanc\_dma\_ack**. This resets **m\_ttcanc\_dma\_req**. Now the M\_TTCAN is prepared to receive the next set of debug messages.

#### 3.4.4.1 Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages **SFEC / EFEC** has to be programmed to "111". In this case fields **SFID1 / SFID2** and **EFID1 / EFID2** have a different meaning (see Section 2.4.5 and Section 2.4.6). While **SFID2 / EFID2[10:9]** controls the debug message handling state machine, **SFID2 / EFID2[5:0]** controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor **IR.DRX** are set. The reception of debug messages can be monitored via **RXF1S.DMS**.

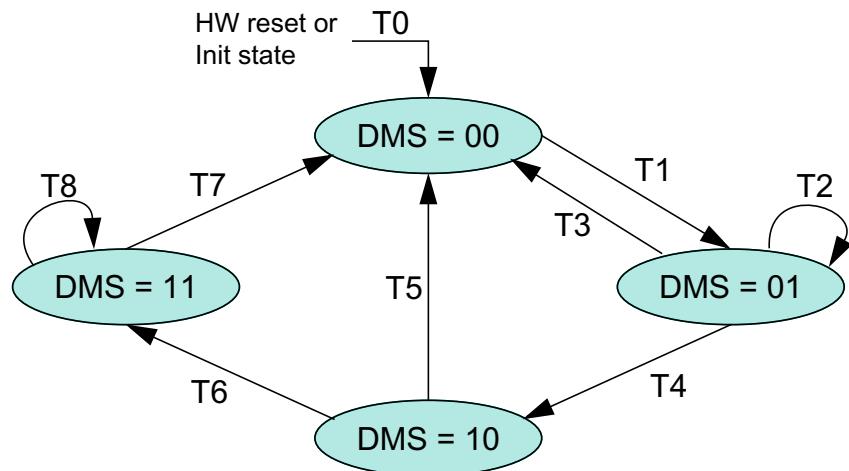
<b>Filter Element</b>	<b>SFID1[10:0] EFID1[28:0]</b>	<b>SFID2[10:9] EFID2[10:9]</b>	<b>SFID2[5:0] EFID2[5:0]</b>
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

Table 74 Example Filter Configuration for Debug Messages

#### 3.4.4.2 Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signalled via **RXF1S.DMS**.



- T0: reset m\_cam\_dma\_req output, enable reception of debug messages A, B, and C
- T1: reception of debug message A
- T2: reception of debug message A
- T3: reception of debug message C
- T4: reception of debug message B
- T5: reception of debug messages A, B,
- T6: reception of debug message C
- T7: DMA transfer completed
- T8: reception of debug message A,B,C (message rejected)

*Figure 10 Debug Message Handling State Machine*

### 3.5 Tx Handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in Section 2.4.3.

**Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation**

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register **TXBRP** is updated, or when a transmission has been started.

#### 3.5.1 Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit **CCCR.TXP**. If the bit is set, the M\_TTCAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (**CCCR.TXP = '0'**).

This feature loosens up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

#### 3.5.2 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via **TXBAR.ARn**. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (see Table 75). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0...31) • Element Size to the Tx Buffer Start Address **TXBC.TBSA**.

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

Table 75 Tx Buffer / FIFO / Queue Element Size

### 3.5.3 Tx FIFO

Tx FIFO operation is configured by programming **TXBC.TFQM** to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index **TXFQS.TFGI**. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M\_TTCAN calculates the Tx FIFO Free Level **TXFQS.TFFL** as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index **TXFQS.TFQPI**. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (**TXFQS.TFQF** = '1') is signalled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the **TXBAR** bit related to the Tx Buffer referenced by the Tx FIFO's Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via **TXBAR**. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see Table 75). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index **TXFQS.TFQPI** (0...31) • Element Size to the Tx Buffer Start Address **TXBC.TBSA**.

### 3.5.4 Tx Queue

Tx Queue operation is configured by programming **TXBC.TFQM** to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx Buffer referenced by the Put Index **TXFQS.TFQPI**. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (**TXFQS.TFQF** = '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register **TXBRP** instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see Table 75). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index **TXFQS.TFQPI** (0...31) • Element Size to the Tx Buffer Start Address **TXBC.TBSA**.

### 3.5.5 Mixed Dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by **TXBC.NDTB**. The number of Tx Buffers assigned to the Tx FIFO is configured by **TXBC.TFQS**. In case **TXBC.TFQS** is programmed to zero, only Dedicated Tx Buffers are used.

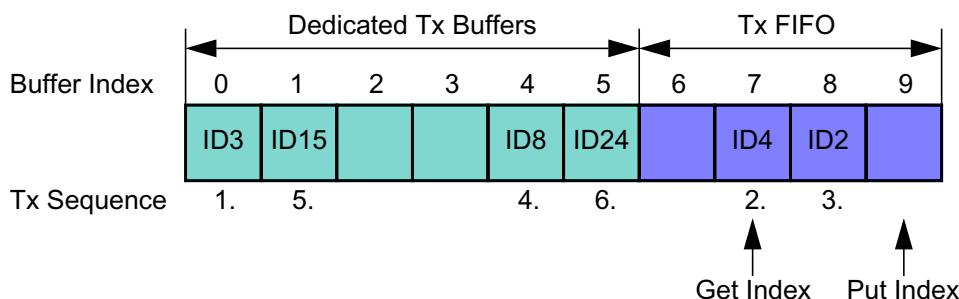


Figure 11 Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by **TXFS.TFGI**)
- Buffer with lowest Message ID gets highest priority and is transmitted next

### 3.5.6 Mixed Dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by **TXBC.NDTB**. The number of Tx Queue Buffers is configured by **TXBC.TFQS**. In case **TXBC.TFQS** is programmed to zero, only Dedicated Tx Buffers are used.

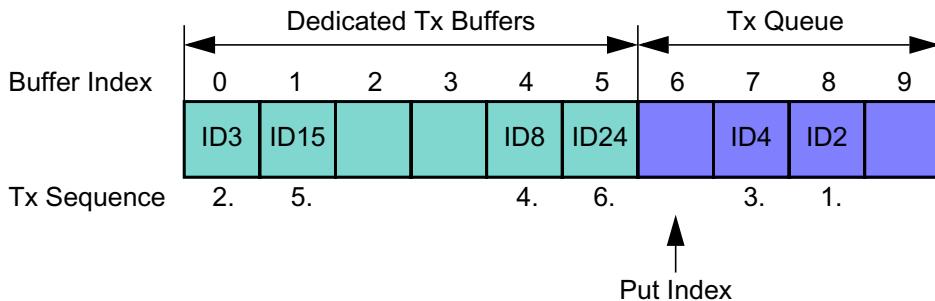


Figure 12 Example of mixed Configuration Dedicated Tx Buffers / Tx Queue

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

### 3.5.7 Transmit Cancellation

The M\_TTCAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a Dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register **TXBCR**. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signalled by setting the corresponding bit of register **TXBCF** to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding **TXBRP** bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding **TXBTO** and **TXBCF** bits are set. If the transmission was not successful, it is not repeated and only the corresponding **TXBCF** bit is set.

**Note:** *In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.*

### 3.5.8 Tx Event Handling

To support Tx event handling the M\_TTCAN has implemented a Tx Event FIFO. After the M\_TTCAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in Section 2.4.4.

When a Tx Event FIFO full condition is signalled by **IR.TEFF**, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag **IR.TEFL** is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by **TXEFC.EFWM**, interrupt flag **IR.TEFW** is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index **TXEFS.EFGI** has to be added to the Tx Event FIFO start address **TXEFC.EFSA**.

## 3.6 FIFO Acknowledge Handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see Section 2.3.28, Section 2.3.32, and Section 2.3.46). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.

When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO's Get Index.

Due to the fact that the CPU has free access to the M\_TTCAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

**Note:** *The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M\_TTCAN does not check for erroneous values.*



# Chapter 4.

## 4. TTCAN Operation

### 4.1 Reference Message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the Time Master (sender of the reference message).

For Level 1 the data length must be at least one; for Level 0,2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to 8 potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference message is configured via register TTRMC.

The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted **Master\_Ref\_Mark** is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (**Next\_is\_Gap** bit) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters.

The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

#### 4.1.1 Level 1

Level 1 operation is configured via **TTOCF.OM** = "01" and **TTOCF.GEN**. External clock synchronization is not available in Level 1.

The information related to the reference message is stored in the first data byte as shown in Table 76 below. **Cycle\_Count** is optional.

Bits	7	6	5	4	3	2	1	0
First Byte	Next_is_Gap	res			Cycle_Count[5:0]			

Table 76 First byte of Level 1 reference message

#### 4.1.2 Level 2

Level 2 operation is configured via **TTOCF.OM** = “10” and **TTOCF.GEN**.

The information related to the reference message is stored in the first four data bytes as shown in Table 77 below. **Cycle\_Count** and the lower four bits of **NTU\_Res** are optional. The M\_TTCAN does not evaluate **NTU\_Res[3:0]** from received reference messages, it always transmits these bits as zero.

<b>Bits</b>	7	6	5	4	3	2	1	0			
<b>First Byte</b>	Next_is_Gap	res	Cycle_Count[5:0]								
<b>Second Byte</b>	NTU_Res[6:4]			NTU_Res[3:0]			Disc_Bit				
<b>Third Byte</b>	Master_Ref_Mark[7:0]										
<b>Fourth Byte</b>	Master_Ref_Mark[15:8]										

Table 77 First four bytes of Level 2 reference message

#### 4.1.3 Level 0

Level 0 operation is configured via **TTOCF.OM** = “11”. External event-synchronized time-triggered operation is not available in Level 0.

The information related to the reference message is stored in the first four data bytes as shown in Table 77 below. In Level 0 **Next\_is\_Gap** is always zero. **Cycle\_Count** and the lower four bits of **NTU\_Res** are optional. The M\_TTCAN does not evaluate **NTU\_Res[3:0]** from received reference messages, it always transmits these bits as zero.

<b>Bits</b>	7	6	5	4	3	2	1	0			
<b>First Byte</b>	Next_is_Gap	res	Cycle_Count[5:0]								
<b>Second Byte</b>	NTU_Res[6:4]			NTU_Res[3:0]			Disc_Bit				
<b>Third Byte</b>	Master_Ref_Mark[7:0]										
<b>Fourth Byte</b>	Master_Ref_Mark[15:8]										

Table 78 First four bytes of Level 0 reference message

## 4.2 TTCAN Configuration

### 4.2.1 TTCAN Timing

The Network Time Unit NTU is the unit in which all times are measured. The NTU is a constant of the whole network and is defined a priori by the network system designer. In TTCAN Level 1 the NTU is the nominal CAN bit time. In TTCAN Level 0 and Level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable.

In TTCAN Level 0 and Level 2 the length of the NTU is defined by the Time Unit Ratio TUR. The TUR is in principle a non-integer number and given by the formula  $TUR = \text{TURCF.NAV} / \text{TURCF.DC}$ . The length of the NTU is given by the formula  $\text{NTU} = \text{CAN Clock Period} \cdot \text{TUR}$ .

The TUR Numerator Configuration **NC** is an 18-bit number, **TURCF.NCL[15:0]** can be programmed in the range 0x0000-0xFFFF. **TURCF.NCH[17:16]** is hard wired to 0b01. When the number 0xnnnn is written to **TURCF.NCL[15:0]**, **TURNA.NAV** starts with the value  $0x10000 + 0x0nnnn = 0x1nnnn$ . The TUR Denominator Configuration **TURCF.DC** is a 14-bit number. **TURCF.DC** may be programmed in the range 0x0001 - 0x3FFF, 0x0000 is an illegal value.

In Level 1, **NC** must be  $\geq 4 \cdot \text{TURCF.DC}$ . In Level 0,2 **NC** must be  $\geq 8 \cdot \text{TURCF.DC}$  to allow the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets **TURCF.DC** to 0x1000 and **TURCF.NCL** to 0x10000, resulting in an NTU consisting of 16 CAN clock periods. Local time and application watchdog are not started before either the **CCCR.INIT** is reset, or **TURCF.ELT** is set. **TURCF.ELT** may not be set before the NTU is configured. Setting **TURCF.ELT** to '1' also locks the write access to register **TURCF**.

At startup **TURNA.NAV** is updated from **NC** ( $= \text{TURCF.NCL} + 0x10000$ ) when **TURCF.ELT** is set. In TTCAN Level 1 there is no drift compensation. **TURNA.NAV** does not change during operation, it always equals **NC**.

In TTCAN Level 0 and Level 2 there are two possibilities for **TURNA.NAV** to change. When operating as time slave or backup time master, and when **TTOCF.ECC** is set, **TURNA.NAV** is updated automatically to the value calculated from the monitored global time speed, as long as the M\_TTCAN is in synchronization state **In\_Schedule** or **In\_Gap**. When it loses synchronization it returns to **NC**. When operating as the actual time master, and when **TTOCF.EECS** is set, the Host may update **TURCF.NCL**. When the Host sets **TTOCN.ECS**, **TURNA.NAV** will be updated from the new value of **NC** at the next reference message. The status flag **TTOST.WECS** is set when **TTOCN.ECS** is set and is cleared when **TURNA.NAV** is updated. **TURCF.NCL** is write locked while **TTOST.WECS** is set.

In TTCAN Level 0 and Level 2 the clock calibration process adapts **TURNA.NAV** in the range of the Synchronization Deviation Limit SDL of  $\text{NC} \pm 2^{(\text{TTOCF.LDSL}+5)}$ . **TURCF.NCL** should be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of **TURNA.NAV**.

The synchronization deviation SD is the difference between **NC** and **TURNA.NAV** ( $SD = |\text{NC} - \text{TURNA.NAV}|$ ). It is limited by the Synchronization Deviation Limit SDL, which is configured by its dual logarithm **TTOCF.LDSL** ( $SDL = 2^{(\text{TTOCF.LDSL}+5)}$ ) and should not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new Basic Cycle. When the calculated **TURNA.NAV** deviates by more than SDL from **NC**, or if the **Disc\_Bit** in the reference message is set, the drift compensation is suspended and **TTIR.GTE** is set and **TTOSC.QCS** is reset, or in case of the **Disc\_Bit** = '1', **TTIR.GTD** is set.

TUR configuration examples are shown in Table 79 below.

<b>TUR</b>	<b>8</b>	<b>10</b>	<b>24</b>	<b>50</b>	<b>510</b>	<b>125000</b>	<b>32.5</b>	<b>100/12</b>	<b>529/17</b>
<b>NC</b>	0x1FFF8	0x1FFE	0x1FFF8	0x1FFEA	0x1FFE	0x1E848	0x1FFE0	0x19000	0x10880
<b>TURCF.DC</b>	0x3FFF	0x3333	0x1555	0x0A3D	0x0101	0x0001	0x0FC0	0x3000	0x0880

Table 79    TUR Configuration Examples

**TTOCN.ECS** schedules **NC** for activation by the next reference message. **TTOCN.SGT** schedules **TTGTP.TP** for activation by the next reference message. Setting of **TTOCN.ECS** and **TTOCN.SGT** requires **TTOCF.EECS** to be set (external clock synchronization enabled) while the M\_TTCAN is actual time master.

The M\_TTCAN module provides an application watchdog to verify the function of the application program. The Host has to serve this watchdog regularly, else all CAN bus activity is stopped. The Application Watchdog Limit **TTOCF.AWL** specifies the number of NTUs between two times the watchdog has to be served. The maximum number of NTUs is 256. The Application Watchdog is served by reading register TTOST. **TTOST.AWE** indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, interrupt flag **TTIR.AW** is set. For software development, the application watchdog may be disabled by programming **TTOCF.AWL** to 0x00 (see also Section 3.1.10).

#### 4.2.1.1 Timing of Interface Signals

The timing events which cause a pulse at output **m\_ttcanc\_tmp** and **m\_ttcanc\_rtp** are generated in the CAN clock domain. There is a clock domain crossing delay to be considered before the same event is visible in the Host clock domain (**TTIR.TTMI** resp. **TTIR.RTMI** set). The signals can be connected e.g. to the timing input(s) of another TTCAN node (**m\_ttcanc\_swt** / **m\_ttcanc\_evt**), in order to automatically synchronize two TTCAN networks.

Output **m\_ttcanc\_soc** gets active whenever a reference message is completed (either transmitted or received). The output is controlled in the Host clock domain.

#### 4.2.2 Message Scheduling

**TTOCF.TM** controls whether the M\_TTCAN operates as a potential time master or as a time slave. If it is a potential time master, the three LSBs of the reference message's identifier **TTRMC.RID** define the master priority, 0 giving the highest and 7 giving the lowest priority. There may not be two nodes in the network using the same master priority. **TTRMC.RID** is used for recognition of reference messages. **TTRMC.RMPS** is not relevant for time slaves.

The Initial Reference Trigger Offset **TTOCF.IRTO** is a 7-bit-value that defines (in NTUs) how long a backup time master waits before it starts the transmission of a reference message when a reference message is expected but the bus remains idle. The recommended value for **TTOCF.IRTO** is the master priority multiplied with a factor depending on the expected clock drift between the potential time masters in the network. The sequential order of the backup time masters, when one of them starts the reference message in case the current time master fails, should correspond to their master priority, even with maximum clock drift.

**TTOCF.OM** decides whether the node operates in TTCAN Level 0, Level 1, or Level 2. In one network, all potential time masters have to operate on the same level. Time slaves may operate on Level 1 in a Level 2 network, but not vice versa. The configuration of the TTCAN operation mode via **TTOCF.OM** is the last step in the setup. With **TTOCF.OM** = "00" (event-driven CAN communication), the M\_TTCAN operates according to ISO 11898-1, without time triggers. With **TTOCF.OM** = "01" (Level 1), the M\_TTCAN operates according to ISO 11898-4, but without the possibility to synchronize the basic cycles to external events, the **Next\_is\_Gap** bit in the reference message is ignored. With **TTOCF.OM** = "10" (Level 2), the M\_TTCAN operates according to ISO 11898-4, including the event-synchronized start of a basic cycle. With **TTOCF.OM** = "11" (Level 0), the M\_TTCAN operates as event-driven CAN but maintains a calibrated global time base as in Level 2.

**TTOCF.EECS** enables the external clock synchronisation, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in Level 0,2 only) the global clock phase to an external reference.

**TTMLM.ENTT** in the TT Matrix Limits register specifies the number of expected Tx\_Triggers in the system matrix. This is the sum of Tx\_Triggers for exclusive, single arbitrating and merged arbitrating windows, excluding the Tx\_Ref\_Triggers. Note that this is usually not the number of Tx\_Trigger memory elements; the number of basic cycles in the system matrix and the trigger's repeat factors have to be taken into account. An inaccurate configuration of **TTMLM.ENTT** will result in either a Tx Count Underflow (**TTIR.TXU** = '1' and **TTOST.EL** = "01", severity 1) or in a Tx Count Overflow (**TTIR.TXO** = '1' and **TTOST.EL** = "10", severity 2).

**Note:** *In case the first reference message seen by a node does not have Cycle\_Count zero, this node may finish its first matrix cycle with its Tx count resulting in a Tx Count Underflow condition. As long as a node is in state Synchronizing its Tx\_Triggers will not lead to transmissions.*

**TTMLM.CCM** specifies the number of the last basic cycle in the system matrix. The counting of basic cycles starts at 0. In a system matrix consisting of 8 basic cycles **TTMLM.CCM** would be 7. **TTMLM.CCM** is ignored by time slaves, a receiver of a reference message considers the received cycle count as the valid cycle count for the actual basic cycle.

**TTMLM.TXEW** specifies the length of the Tx enable window in NTUs. The Tx enable window is that period of time at the beginning of a time window where a transmission may be started. If the sample point of the first bit of a transmit message is not inside the Tx enable window because of e.g. a slight overlap from the previous time window's message, the transmission cannot be started in that time window at all. **TTMLM.TXEW** has to be chosen with respect to the network's synchronisation quality and with respect to the relation between the length of the time windows and the length of the messages.

#### 4.2.3 Trigger Memory

The trigger memory is part of the external Message RAM to which the M\_TTCAN is connected via its Generic Master Interface (see Figure 2). It stores up to 64 trigger elements. A trigger memory element consists of Time Mark **TM**, Cycle Code **CC**, Trigger Type **TYPE**, Filter Type **FTYPE**, Message Number **MNR**, Message Status Count **MSC**, Time Mark Event Internal **TMIN**, Time Mark Event External **TMEX**, and Asynchronous Serial Communication **ASC** (see Section 2.4.7).

The time mark defines at which cycle time a trigger becomes active. The triggers in the trigger memory have to be sorted by their time marks. The trigger element with the lowest time mark is written to the first trigger memory word. Message number and cycle code are ignored for triggers of type Tx\_Ref\_Trigger, Tx\_Ref\_Trigger\_Gap, Watch\_Trigger, Watch\_Trigger\_Gap, and End\_of\_List.

When the cycle time reaches the time mark of the actual trigger, the FSE switches to the next trigger and starts to read the following trigger from the trigger memory. In case of a transmit trigger, the Tx Handler starts to read the message from the Message RAM as soon as the FSE switches to its trigger. The RAM access speed defines the minimum time step between a transmit trigger and its preceding trigger, the Tx Handler has to be able to prepare the transmission before the transmit trigger's time mark is reached. The RAM access speed also limits the number of non-matching (with regard to their cycle code) triggers between two matching triggers, the next matching trigger must be read before its time mark is reached. If the reference message is n NTU long, a trigger with a time mark < n will never become active and will be treated as a configuration error.

Starting point of the cycle time is the sample point of the reference message's start of frame bit. The next reference message is requested when cycle time reaches the Tx\_Ref\_Trigger's time mark. The M\_TTCAN reacts on the transmission request at the next sample point. A new Sync\_Mark is captured at the start of frame bit, but the cycle time is incremented until the reference message is successfully transmitted (or received) and the Sync\_Mark is taken as the new Ref\_Mark. At that point in time, cycle time is restarted. As a consequence, cycle time can never (with the exception of initialisation) be seen at a value < n, with n being the length of the reference message measured in NTU.

Length of a basic cycle: Tx\_Ref\_Trigger's time mark+1 NTU+1 CAN bit time

The trigger list will be different for all nodes in the TTCAN network. Each node knows only the Tx\_Triggers for its own transmit messages, the Rx\_Triggers for those receive messages that are processed by this node, and the triggers concerning the reference messages.

#### 4.2.3.1 Trigger Types

Tx\_Ref\_Trigger (**TYPE** = "0000") and Tx\_Ref\_Trigger\_Gap (**TYPE** = "0001") cause the transmission of a reference message by a time master. A configuration error (**TTOST.EL** = "11", severity 3) is detected when a time slave encounters a Tx\_Ref\_Trigger(\_Gap) in its trigger memory. Tx\_Ref\_Trigger\_Gap is only used in external event-synchronised time-triggered operation mode. In that mode, Tx\_Ref\_Trigger is ignored when the M\_TTCAN synchronisation state is In\_Gap (**TTOST.SYS** = "10").

Tx\_Trigger\_Single (**TYPE** = "0010"), Tx\_Trigger\_Continous (**TYPE** = "0011"), Tx\_Trigger\_Arbitration (**TYPE** = "0100"), and Tx\_Trigger\_Merged (**TYPE** = "0101") cause the start of a transmission. They define the start of a time window.

Tx\_Trigger\_Single starts a single transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit is reset.

Tx\_Trigger\_Continous starts a transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit remains set, and the message buffer is transmitted again in the next matching time window.

Tx\_Trigger\_Arbitration starts an arbitrating time window, Tx\_Trigger\_Merged a merged arbitrating time window. The last Tx\_Trigger of a merged arbitrating time window must be of type Tx\_Trigger\_Arbitration. A Configuration Error (**TTOST.EL** = "11", severity 3) is detected when a trigger of type Tx\_Trigger\_Merged is followed by any other Tx\_Trigger than one of type Tx\_Trigger\_Merged or Tx\_Trigger\_Arbitration. Several Tx\_Triggers may be defined for the same Tx message buffer. Depending on their cycle code, they may be ignored in some basic cycles. The cycle code has to be considered when the expected number of Tx\_Triggers (**TTMLM.ENNT**) is calculated.

Watch\_Trigger (**TYPE** = "0110") and Watch\_Trigger\_Gap (**TYPE** = "0111") check for missing reference messages. They are used by both time masters and time slaves. Watch\_Trigger\_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, a Watch\_Trigger is ignored when the M\_TTCAN synchronisation state is In\_Gap (**TTOST.SYS** = "10").

Rx\_Trigger (**TYPE** = "1000") is used to check for the reception of periodic messages in exclusive time windows. Rx\_Triggers are not active until state In\_Schedule or In\_Gap is reached. The time mark of an Rx\_Trigger shall be placed after the end of that message's transmission, independent of time window boundaries. Depending on their cycle code, Rx\_Triggers may be ignored in some basic cycles. At the time mark of the Rx\_Trigger, it is checked whether the last received message before this time mark and after start of cycle or previous Rx\_Trigger had matched the acceptance filter element referenced by **MNR**. Accepted messages are stored in one of the two receive FIFOs, according to the acceptance filtering, independent of the Rx\_Trigger. Acceptance filter elements which are referenced by Rx\_Triggers should be placed at the beginning of the filter list to ensure that the filtering is finished before the Rx\_Trigger's time mark is reached.

Time\_Base\_Trigger (**TYPE** = "1001") are used to generate internal/external events depending on the configuration of **ASC**, **TMIN**, and **TMEX**.

End\_of\_List (**TYPE** = "1010...1111") is an illegal trigger type, a configuration error (**TTOST.EL** = "11", severity 3) is detected when an End\_of\_List trigger is encountered in the trigger memory before the Watch\_Trigger or Watch\_Trigger\_Gap.

#### 4.2.3.2 Restrictions for the Node's Trigger List

There may not be two triggers that are active at the same cycle time and cycle count, but triggers that are active in different basic cycles (different cycle code) may share the same time mark.

Rx\_Triggers and Time\_Base\_Triggers may not be placed inside the Tx enable windows of Tx\_Trigger\_Single/Continuous/Arbitration, but they may be placed after Tx\_Trigger\_Merged.

Triggers that are placed after the Watch\_Trigger (or the Watch\_Trigger\_Gap when **TTOST.SYS** = "10") will never become active. The watch triggers themselves will not become active when the reference messages are transmitted on time.

All unused trigger memory words (after the Watch\_Trigger or after the Watch\_Trigger\_Gap when **TTOST.SYS** = "10") must be set to trigger type End\_of\_List.

A typical trigger list for a potential time master will begin with a number of Tx\_Triggers and Rx\_Triggers followed by the Tx\_Ref\_Trigger and the Watch\_Trigger. For networks with external event-synchronized time-triggered communication, this is followed by the Tx\_Ref\_Trigger\_Gap and the Watch\_Trigger\_Gap. The trigger list for a time slave will be the same but without the Tx\_Ref\_Trigger and the Tx\_Ref\_Trigger\_Gap.

At the beginning of each basic cycle, that is at each reception or transmission of a reference message, the trigger list is processed starting with the first trigger memory element. The FSE looks for the first trigger with a cycle code that matches the current cycle count. The FSE waits until cycle time reaches the trigger's time mark and activates the trigger. Afterwards the FSE looks for the next trigger in the list with a cycle code that matches the current cycle count.

Special consideration is needed for the time around Tx\_Ref\_Trigger and Tx\_Ref\_Trigger\_Gap. In a time master competing for master ship, the effective time mark of a Tx\_Ref\_Trigger may be decremented in order to be the first node to start a reference message. In backup time masters the effective time mark of a Tx\_Ref\_Trigger or Tx\_Ref\_Trigger\_Gap is the sum of its configured time mark and the Reference Trigger Offset **TTOCF.IRTO**. In case error level 2 is reached (**TTOST.EL** = "10"), the effective time mark is the sum of its time mark and 0x127. No other trigger elements should be placed in this range otherwise it may happen, that the time marks appear out of order and are flagged as a configuration error. Trigger elements which are coming after Tx\_Ref\_Trigger may never become active as long as the reference messages come in time.

There are interdependencies between the following parameters:

- Host clock frequency
- Speed and waiting time for Trigger RAM accesses
- Length of the acceptance filter list
- Number of trigger elements
- Complexity of cycle code filtering in the trigger elements
- Offset between time marks of the trigger elements

#### 4.2.3.3 Example for Trigger Handling

The example below shows how the trigger list is derived from a node's system matrix. Assumed node A is first time master and has knowledge of the section of the system matrix shown in Table 76 below.

Cycle Count	Time Mark1	Time Mark2	Time Mark3	Time Mark4	Time Mark5	Time Mark6	Time Mark7
0	Tx7					TxRef	Error
1	Rx3			Tx2, Tx4		TxRef	Error
2						TxRef	Error
3	Tx7			Rx5		TxRef	Error
4	Tx7				Rx6	TxRef	Error

Table 80 System Matrix Node A

The cycle count starts with 0 and runs until 0, 1, 3, 7, 15, 31, 63 (the number of basic cycles in the system matrix is 1, 2, 4, 8, 16, 32, 64). The maximum cycle count is configured by **TTMLM.CCM**. The Cycle Code **CC** is composed of repeat factor (= value of most significant '1') and the number of the first basic cycle in the system matrix (= bit field after most significant '1').

Example: with a cycle code of 0b0010011 (repeat factor: 16, first basic cycle: 3) and a maximum cycle count of **TTMLM.CCM** = "0x3F" matches occur at cycle counts 3, 19, 35, 51

A trigger element consists of Time Mark **TM**, Cycle Code **CC**, Trigger Type **TYPE**, and Message Number **MNR**. For transmission **MNR** references the Tx Buffer number (0..31). For reception **MNR** references the number of the filter element (0..127) that matched during acceptance filtering. Depending on the configuration of the Filter Type **FTYPE**, the 11-bit or 29-bit message ID filter list is referenced.

In addition a trigger element can be configured for Asynchronous Serial Communication **ASC**, generation of Time Mark Event Internal **TMIN**, and Time Mark Event External **TMEX**. The Message Status Count **MSC** holds the counter value (0..7) for scheduling errors for periodic messages in exclusive time windows at the point in time when the time mark of the trigger element became active.

Trigger	Time Mark TM[15:0]	Cycle Code CC[6:0]	Trigger Type TYPE[3:0]	Mess. No. MNR[6:0]
0	Mark1	0b0000100	Tx_Trigger_Single	7
1	Mark1	0b1000000	Rx_Trigger	3
2	Mark1	0b1000011	Tx_Trigger_Single	7
3	Mark3	0b1000001	Tx_Trigger_Merged	2
4	Mark3	0b1000011	Rx_Trigger	5
5	Mark4	0b1000001	Tx_Trigger_Arbitration	4
6	Mark4	0b1000100	Rx_Trigger	6
7	Mark6	n.a.	Tx_Ref_Trigger	0 (Ref)
8	Mark7	n.a.	Watch_Trigger	n.a.
9	n.a.	n.a.	End_of_List	n.a.

Table 81 Trigger List Node A

Tx\_Trigger\_Single, Tx\_Trigger\_Continous, Tx\_Trigger\_Merged, Tx\_Trigger\_Arbitration, Rx\_Trigger, and Time\_Base\_Trigger are only valid for the specified cycle code. For all other trigger types the cycle code is ignored.

The FSE starts the basic cycle with scanning the trigger list starting from zero until a trigger with time mark > cycle time and with its Cycle Code **CC** matching the actual cycle count is reached, or a trigger of type Tx\_Ref\_Trigger, Tx\_Ref\_Trigger\_Gap, Watch\_Trigger, or Watch\_Trigger\_Gap is encountered.

When the cycle time reached the Time Mark **TM**, the action defined by Trigger Type **TYPE** and Message Number **MNR** is started. There is an error in the configuration when **End\_of\_List** is reached.

At Mark6 the reference message (always TxRef) is transmitted. After transmission of the reference message the FSE returns to the beginning of the trigger list. When the Watch Trigger at Mark7 is reached, the node was not able to transmit the reference message; error treatment is started.

#### 4.2.3.4 Detection of Configuration Errors

A configuration error is signalled via **TTOST.EL** = "11" (severity 3) when:

The FSE comes to a trigger in the list with a cycle code that matches the current cycle count but with a time mark that is less than the cycle time.

The previous active trigger was a Tx\_Trigger\_Merged and the FSE comes to a trigger in the list with a cycle code that matches the current cycle count but that is neither a Tx\_Trigger\_Merged nor a Tx\_Trigger\_Arbitration nor a Time\_Base\_Trigger nor an Rx\_Trigger.

The FSE of a node with **TTOCF.TM**='0' (time slave) encounters a Tx\_Ref\_Trigger or a Tx\_Ref\_Trigger\_Gap.

Any time mark placed inside the Tx enable window (defined by **TTMLM.TXEW**) of a Tx\_Trigger with a matching cycle code.

A time mark is placed near the time mark of a Tx\_Ref\_Trigger and the Reference Trigger Offset **TTOST.RTO** causes a reversal of their sequential order measured in cycle time.

#### 4.2.4 TTCAN Schedule Initialization

The synchronisation to the M\_TTCAN's message schedule starts when **CCCR.INIT** is reset. The M\_TTCAN can operate strictly time-triggered (**TTOCF.GEN** = '0') or external event-synchronized time-triggered (**TTOCF.GEN** = '1'). All nodes start with cycle time zero at the beginning of their trigger list with **TTOST.SYS** = "00" (out of synchronisation), no transmission is enabled with the exception of the reference message. Nodes in external event-synchronised time-triggered operation mode will ignore Tx\_Ref\_Trigger and Watch\_Trigger and will use instead Tx\_Ref\_Trigger\_Gap and Watch\_Trigger\_Gap until the first reference message decides whether a Gap is active.

##### 4.2.4.1 Time Slaves

After configuration, a time slave will ignore its Watch\_Trigger and Watch\_Trigger\_Gap when it did not receive any message before reaching the Watch\_Triggers. When it reaches Init\_Watch\_Trigger, interrupt flag **TTIR.IWT** is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). The first received reference message will restart the FSE and the cycle time.

**Note:** *Init\_Watch\_Trigger is not part of the trigger list. It is implemented as an internal counter which counts up to 0xFFFF = maximum cycle time.*

When a time slave has received any message but the reference message before reaching the Watch\_Triggers, it will assume a fatal error (**TTOST.EL** = "11", severity 3), set interrupt flag **TTIR.WT**, switch off its CAN bus output, and enter the bus monitoring mode (**CCCR.MON** set to '1'). In the bus monitoring mode it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

**Note:** *To leave the fatal error state, the Host has to set CCCR.INIT = '1'. After reset of CCCR.INIT, the node restarts TTCAN communication.*

When no error is encountered during synchronisation, the first reference message sets **TTOST.SYS** = "01" (Synchronizing), the second sets the TTCAN synchronization state (depending on its **Next\_is\_Gap** bit) to **TTOST.SYS** = "11" (In\_Schedule) or **TTOST.SYS** = "10" (In\_Gap), enabling all Tx\_Triggers and Rx\_Triggers.

#### 4.2.4.2 Potential Time Masters

After configuration, a potential time master will start the transmission of a reference message when it reaches its Tx\_Ref\_Trigger (or its Tx\_Ref\_Trigger\_Gap when in external event-synchronized time-triggered operation). It will ignore its Watch\_Trigger and Watch\_Trigger\_Gap when it did not receive any message or transmit the reference message successfully before reaching the Watch\_Triggers (assumed reason: all other nodes still in reset or configuration, giving no acknowledge). When it reaches Init\_Watch\_Trigger, the attempted transmission is aborted, interrupt flag **TTIR.IWT** is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). Resetting **TTIR.IWT** will re-enable the transmission of reference messages until next time the Init\_Watch\_Trigger condition is met, or another CAN message is received. The FSE will be restarted by the reception of a reference message.

When a potential time master reaches the Watch\_Triggers after it has received any message but the reference message, it will assume a fatal error (**TTOST.EL** = "11", severity 3), set interrupt flag **TTIR.WT**, switch off its CAN bus output, and enter the bus monitoring mode (**CCCR.MON** set to '1'). In bus monitoring mode, it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

When no error is detected during initialization, the first reference message sets **TTOST.SYS** = "01" (synchronizing), the second sets the TTCAN synchronization state (depending on its **Next\_is\_Gap** bit) to **TTOST.SYS** = "11" (In\_Schedule) or **TTOST.SYS** = "10" (In\_Gap), enabling all Tx\_Triggers and Rx\_Triggers.

A potential time master is current time master (**TTOST.MS** = "11") when it was the transmitter of the last reference message, else it is backup time master (**TTOST.MS** = "10").

When all potential time masters have finished configuration, the node with the highest time master priority in the network will become the current time master.

### 4.3 TTCAN Gap Control

All functions related to Gap control apply only when the M\_TTCAN is operated in external event-synchronized time-triggered mode (**TTOCF.GEN** = '1'). In this operation mode the TTCAN message schedule may be interrupted by inserting Gaps between the basic cycles of the system matrix. All nodes connected to the CAN network have to be configured for external event-synchronized time-triggered operation.

During a Gap, all transmissions are stopped and the CAN bus remains idle. A Gap is finished when the next reference message starts a new basic cycle. A Gap starts at the end of a basic cycle that itself was started by a reference message with bit **Next\_is\_Gap** = '1' e.g. Gaps are initiated by the current time master.

The current time master has two options to initiate a Gap. A Gap can be initiated under software control when the application program writes **TTOCN.NIG** = '1'. The **Next\_is\_Gap** bit will be transmitted as '1' with the next reference message. A Gap can also be initiated under hardware control when the application program enables the event trigger input pin **m\_ttcanc\_evt** by writing **TTOCN.GCS** = '1'. When a reference message is started and **TTOCN.GCS** is set, a HIGH level at pin **m\_ttcanc\_evt** will set **Next\_is\_Gap** = '1'.

As soon as that reference message is completed, the **TTOST.WFE** bit will announce the Gap to the time master as well as to the time slaves. The current basic cycle will continue until its last time window. The time after the last time window is the Gap time.

For the actual time master and the potential time masters, **TTOST.GSI** will be set when the last basic cycle has finished and the Gap time starts. In nodes that are time slaves, bit **TTOST.GSI** will remain at '0'.

When a potential time master is in synchronization state **In\_Gap** (**TTOST.SYS** = "10"), it has four options to intentionally finish a Gap:

Under software control by writing **TTOCN.FGP** = '1'.

Under hardware control (**TTOCN.GCS** = '1') an edge from HIGH to LOW at the event-trigger input pin **m\_ttcn\_evt** sets **TTOCN.FGP** and restarts the schedule.

The third option is a time-triggered restart. When **TTOCN.TMG** = '1', the next register time mark interrupt (**TTIR.RTMI** = '1') will set **TTOCN.FGP** and start the reference message.

Finally any potential time master will finish a Gap when it reaches its **Tx\_Ref\_Trigger\_Gap**, assuming that the event to synchronize on did not occur in time.

Neither of these options can cause a basic cycle to be interrupted with a reference message.

Setting of **TTOCN.FGP** after the Gap time has started will start the transmission of a reference message immediately and will thereby synchronize the message schedule. When **TTOCN.FGP** is set before the Gap time has started (while the basic cycle is still in progress), the next reference message is started at the end of the basic cycle, at the **Tx\_Ref\_Trigger** – there will be no Gap time in the message schedule.

In strictly time-triggered operation, bit **Next\_is\_Gap** = '1' in the reference message will be ignored, as well as the event-trigger input pin **m\_ttcn\_evt** and the bits **TTOCN.NIG**, **TTOCN.FGP**, and **TTOCN.TMG**.

#### 4.4 Stop Watch

The stop watch function enables capturing of M\_TTCAN internal time values (local time, cycle time, or global time) triggered by an external event.

To enable the stop watch function, the application program first has to define local time, cycle time, or global time as stop watch source via **TTOCN.SWS**. When **TTOCN.SWS** is ≠ “00” and TT Interrupt Register flag **TTIR.SWE** is ‘0’, the actual value of the time selected by **TTOCN.SWS** will be copied into **TTCPT.SWV** on the next rising/falling edge (as configured via **TTOCN.SWP**) on pin **m\_ttcanc\_swt**. This will set interrupt flag **TTIR.SWE**. After the application program has read **TTCPT.SWV**, it may enable the next stop watch event by resetting **TTIR.SWE** to ‘0’.

#### 4.5 Local Time, Cycle Time, Global Time, and External Clock Synchronization

There are two possible levels in time-triggered CAN: Level 1 and Level 2. Level 1 only provides time-triggered operation using cycle time. Level 2 additionally provides increased synchronisation quality, global time and external clock synchronisation. In both levels, all timing features are based on a local time base - the local time.

The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU is represented by a 3-bit counter which can be regarded as a fractional part (three binary digits) of the local time. Generally, the 3-bit NTU counter is incremented 8 times each NTU. If the length of the NTU is shorter than 8 CAN clock periods (as may be configured in Level 1, or as a result of clock calibration in Level 2), the length of the NTU fraction is adapted, and the NTU counter is incremented only 4 times each NTU.

Figure 13 describes the synchronisation of the cycle time and global time, performed in the same manner by all TTCAN nodes, including the time master. Any message received or transmitted invokes a capture of the local time taken at the message's frame synchronisation event. This frame synchronisation event occurs at the sample point of each Start of Frame (SoF) bit and causes the local time to be stored as Sync\_Mark. Sync\_Marks and Ref\_Marks are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref\_Mark is updated from the Sync\_Mark. The difference between Ref\_Mark and Sync\_Mark is the Cycle Sync Mark (Cycle Sync Mark = Sync\_Mark - Ref\_Mark) stored in register **TTCSM**. The most significant 16 bits of the difference between Ref\_Mark and the actual value of the local time is the cycle time (Cycle Time = Local Time - Ref\_Mark).

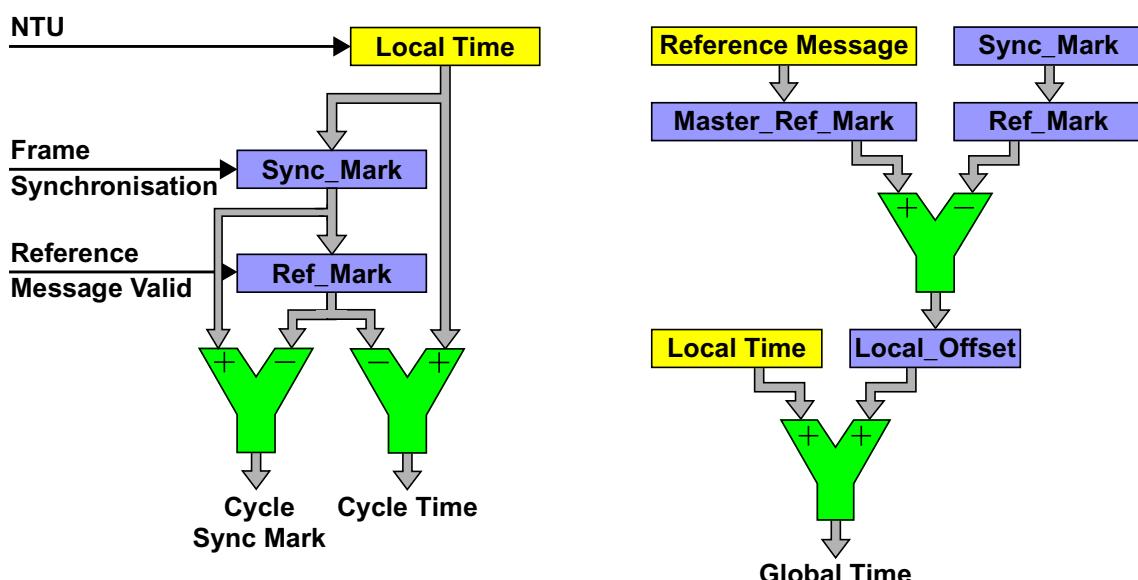


Figure 13 Cycle Time and Global Time Synchronization

The cycle time that can be read from **TTCTC.CT** is the difference of the node's local time and **Ref\_Mark**, both synchronized into the Host clock domain and truncated to 16 bit.

The global time exists for TTCAN Level 0 and Level 2 only, in Level 1 it is invalid. The node's view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master will use its own local time as global time. The time master establishes its own local time as global time by transmitting its own **Ref\_Marks** as **Master\_Ref\_Marks** in the reference message (bytes 3,4). The global time that can be read from **TTLGT.GT** is the sum of the node's local time and its local offset, both synchronized into the Host clock domain and truncated to 16 bit. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local **Ref\_Mark** with the received **Master\_Ref\_Mark** (see Figure 13). The node's view of the global time is local time + local offset. In a potential time master that has never received another time master's reference message, **Local\_Offset** will be zero. When a node becomes the current time master after first having received other reference messages, **Local\_Offset** will be frozen at its last value. In the time receiving nodes, **Local\_Offset** may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signalled by **Disc\_Bit** in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register **TTLGT** is smoothed by a low-pass filtering to have a continuous monotonic value.

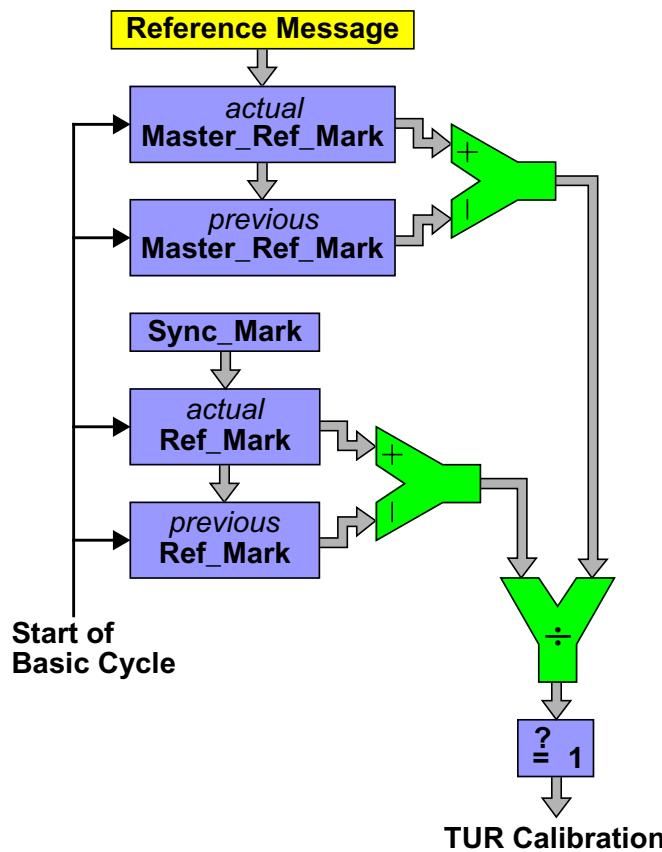


Figure 14     TTCAN Level 0 and Level 2 Drift Compensation

Figure 14 describes how in TTCAN Level 0,2 each time receiving node compensates the drift between its own local clock and the time master's clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the **Disc\_Bit** in the reference message is not set, a new value for **TURNA.NAV** is calculated. If the Synchronisation Deviation SD =  $|NC - TURNA.NAV| \leq SDL$  (Synchronisation Deviation Limit), the new value for **TURNA.NAV** takes effect. Else the automatic drift compensation is suspended.

In TTCAN Level 0 and Level 2, **TTOST.QCS** indicates whether the automatic drift compensation is active or suspended. In TTCAN Level 1, **TTOST.QCS** is always '1'.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit **TTOCF.EECS**.

The stop watch function (see Section 4.4) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated Numerator Configuration Low to **TURCF.NCL** (**TURCF.DC** cannot be updated during operation). The new value takes effect by writing **TTOCN.ECS** to '1'.

The global time phase is adjusted by first writing the phase offset into the TT Global Time Preset register **TTGTP**. The new value takes effect by writing **TTOCN.SGT** to '1'. The first reference message transmitted after the global time phase adjustment will have the **Disc\_Bit** set to '1'.

**TTOST.QGTP** shows whether the node's global time is in phase with the time master's global time. **TTOST.QGTP** is permanently '0' in TTCAN Level 1 and when the Synchronisation Deviation Limit is exceeded in TTCAN Level 0,2 (**TTOST.QCS** = '0'). It is temporarily '0' while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a **Disc\_Bit** = '1' or when **TTOST.QCS** = '0'.

## 4.6 TTCAN Error Level

The ISO 11898-4 specifies four levels of error severity:

S0 - No Error

S1 - Warning

Only notification of application, reaction application-specific.

S2 Error

Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (i.e. no data or remote frames may be started). Potential time masters still transmit reference messages with the Reference Trigger Offset **TTOST.RTO** set to the maximum value of 127.

S3 - Severe Error

Notification of application. All CAN bus operations are stopped, i.e. transmission of dominant bits is not allowed, and **CCCR.MON** is set. The S3 error condition remains active until the application updates the configuration (set **CCCR.CCE**).

If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by **TTIR.ELC**. The error level is monitored by **TTOST.EL**.

The M\_TTCAN signals the following error conditions as required by ISO 11898-4:

**Config\_Error (S3)**

Sets Error Level **TTOST.EL** to "11" when a merged arbitrating time window is not properly closed or when there is a Tx\_Trigger with a time mark beyond the Tx\_Ref\_Trigger.

**Watch\_Trigger\_Reached (S3)**

Sets Error Level **TTOST.EL** to "11" when a watch trigger was reached because the reference message is missing.

**Application\_Watchdog (S3)**

Sets Error Level **TTOST.EL** to "11" when the application failed to serve the application watchdog. The application watchdog is configured via **TTOCF.AWL**. It is served by reading register TTOST. When the watchdog is not served in time, bit **TTOST.AWE** and interrupt flag **TTIR.AW** are set, all TTCAN communication is stopped, and the M\_TTCAN is set into bus monitoring mode (**CCCR.MON** set to '1').

**CAN\_Bus\_Off (S3)**

Entering **CAN\_Bus\_Off** state sets error level **TTOST.EL** to "11". **CAN\_Bus\_Off** state is signalled by **PSR.BO** = '1' and **CCCR.INIT** = '1'.

**Scheduling\_Error\_2 (S2)**

Sets Error Level **TTOST.EL** to “10” if the MSC of one Tx\_Trigger has reached 7. In addition interrupt flag **TTIR.SE2** is set. The Error Level **TTOST.EL** is reset to “00” at the beginning of a matrix cycle when no Tx\_Trigger has an MSC of 7 in the preceding matrix cycle.

**Tx\_Overflow (S2)**

Sets Error Level **TTOST.EL** to “10” when the Tx count is equal or higher than the expected number of Tx\_Triggers **TTMLM.ENTT** and a Tx\_Trigger event occurs. In addition interrupt flag **TTIR.TXO** is set. The Error Level **TTOST.EL** is reset to “00” when the Tx count is no more than **TTMLM.ENTT** at the start of a new matrix cycle.

**Scheduling\_Error\_1 (S1)**

Sets Error Level **TTOST.EL** to “01” if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than 2, or if one of the MSCs of an exclusive Rx\_Trigger has reached 7. In addition interrupt flag **TTIR.SE1** is set. If within one matrix cycle none of these conditions is valid, the Error Level **TTOST.EL** is reset to “00”.

**Tx\_Underflow (S1)**

Sets Error Level **TTOST.EL** to “01” when the Tx count is less than the expected number of Tx\_Triggers **TTMLM.ENTT** at the start of a new matrix cycle. In addition interrupt flag **TTIR.TXU** is set. The Error Level **TTOST.EL** is reset to “00” when the Tx count is at least **TTMLM.ENTT** at the start of a new matrix cycle.

## 4.7 TTCAN Message Handling

### 4.7.1 Reference Message

For potential time masters the identifier of the reference message is configured via **TTRMC.RID**. No dedicated Tx Buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte (TTCAN Level 1) resp. the first four data bytes (TTCAN Level 0 and Level 2) will be provided by the FSE.

In case the reference message Payload Select **TTRMC.RMPS** is set, the rest of the reference message's payload (Level 1: bytes 2-8, Level 0,2: bytes 5-6) is taken from Tx Buffer 0. In this case the data length **DLC** code from message buffer 0 is used.

<b>TTRMC.RMPS</b>	<b>TXBRP.TRP0</b>	<b>Level 0</b>	<b>Level 1</b>	<b>Level 2</b>
0	0	4	1	4
0	1	4	1	4
1	0	4	1	4
1	1	4 + MB0	1 + MB0	4 + MB0

Table 82 Number of Data Bytes transmitted with a reference messages

To send additional payload with the reference message in Level 1 a **DLC** > 1 has to be configured, for Level 0,2 a **DLC** > 4 is required. In addition the transmission request pending bit **TXBRP.TRP0** of message buffer 0 must be set (see Table 82). In case bit **TXBRP.TRP0** is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the Reference Identifier **TTRMC.RID** is used.

### 4.7.2 Message Reception

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (see Section 3.4).

The Message Status Count **MSC** is part of the corresponding trigger memory element and has to be initialized to zero during configuration. It is updated while the M\_TTCAN is in synchronization states In\_Gap or In\_Schedule. The update happens at the message's Rx\_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the **MSC** is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the **MSC** is incremented by one. At each Rx\_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx\_Trigger should be set to a value where it is ensured that reception and acceptance filtering for the targeted message has completed. This has to take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx\_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx\_Trigger for the reference message.

#### 4.7.3 Message Transmission

For time-triggered message transmission the M\_TTCAN supplies 32 dedicated Tx buffers (see Section 3.5.2). A Tx FIFO or Tx queue is not available when the M\_TTCAN is configured for time-triggered operation (**TTOCF.OM** = "01" or "10").

Each Tx\_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx\_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle.

The application program has to update the data regularly and on time, synchronized to the cycle time. The Host CPU is responsible that no partially updated messages are transmitted. To assure this the Host has to proceed in the following way:

Tx\_Trigger\_Single / Tx\_Trigger\_Merged / Tx\_Trigger\_Arbitration

- Check whether the previous transmission has completed by reading **TXBTO**
- Update the Tx buffer's configuration and/or payload
- Issue an Add Request to set the Tx Buffer Request Pending bit

Tx\_Trigger\_Continous

- Issue a Cancellation Request to reset the Tx Buffer Request Pending bit
- Check whether the cancellation has finished by reading **TXBCF**
- Update Tx buffer's configuration and/or payload
- Issue an Add Request to set the Tx Buffer Request Pending bit

The message's **MSC** stored with the corresponding Tx\_Trigger provides information on the success of the transmission.

The **MSC** is incremented by one when the transmission could not be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and could not be completed successfully. The **MSC** is decremented by one when the message was transmitted successfully or when the message could have been started within its transmit enable window but was not started because transmission was disabled (M\_TTCAN in Error Level S2 or Host has disabled this particular message).

The Tx buffers may be managed dynamically, i.e. several messages with different identifiers may share the same Tx buffer element. In this case the Host has to assure that no transmission request is pending for the Tx buffer element to be reconfigured by checking **TXBRP**.

If a Tx buffer with pending transmission request should be updated, the Host first has to issue a cancellation request and check whether the cancellation has completed by reading **TXBCF** before it starts updating.

The Tx Handler will transfer a message from the Message RAM to its intermediate output buffer at the trigger element which becomes active immediately before the Tx\_Trigger element which defines the beginning of the transmit window. During and after the transfer time the transmit message may not be updated and its **TXBRP** bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx\_Trigger. This may be e.g. a Time\_Base\_Trigger which need not cause any other action. The difference in time marks between the Tx\_Trigger and the preceding trigger has to be large enough to guarantee that the Tx Handler can read four words from the Message RAM even at high RAM access load from other modules.

#### **4.7.3.1 Transmission in Exclusive Time Windows**

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx\_Trigger\_Single or Tx\_Trigger\_Continous. There is no arbitration on the bus with messages from other nodes. The **MSC** is updated according the result of the transmission attempt. After successful transmission started by a Tx\_Trigger\_Single the respective Tx Buffer Request Pending bit is reset. After successful transmission started by a Tx\_Trigger\_Continous the respective Tx Buffer Request Pending remains set. When the transmission was not successful due to disturbances, it will be repeated next time (one of) its Tx\_Trigger(s) become(s) active.

#### **4.7.3.2 Transmission in Arbitrating Time Windows**

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx\_Trigger\_Arbitration. Several nodes may start to transmit at the same time. In this case the message has to arbitrate with the messages from other nodes. The **MSC** is not updated. When the transmission was not successful (lost arbitration or disturbance), it will be repeated next time (one of) its Tx\_Trigger(s) become(s) active.

#### **4.7.3.3 Transmission in Merged Arbitrating Time Windows**

The purpose of a merged arbitrating time window is to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a single node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission will not be started if the corresponding Transmission Request Pending flag was reset by a successful Tx cancellation.

In single transmit windows, the Tx Handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmissions will be attempted in the sequence defined by the trigger list. If the time mark of a fourth message is read before the first is transmitted (or cancelled by the Host), the fourth request will be ignored.

The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window will be started in the order of their Tx\_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if there is competing bus traffic. This has to be considered for the configuration of the trigger list. Time\_Base\_Triggers may be placed between consecutive Tx\_Triggers to define the time until the data of the corresponding Tx Buffer needs to be updated.

## 4.8 TTCAN Interrupt and Error Handling

The TT Interrupt Register **TTIR** consists of four segments. Each interrupt can be enabled separately by the corresponding bit in the TT Interrupt Enable register **TTIE**. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position.

The first segment consists of flags **CER**, **AW**, **WT**, and **IWT**. Each flag indicates a fatal error condition where the CAN communication is stopped. With the exception of **IWT**, these error conditions require a re-configuration of the M\_TTCAN module before the communication can be restarted.

The second segment consists of flags **ELC**, **SE1**, **SE2**, **TXO**, **TXU**, and **GTE**. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, e.g. by disturbances on the CAN bus, they will be handled by the TTCAN protocol's failure handling and do not require intervention by the application program.

The third segment consists of flags **GTD**, **GTW**, **SWE**, **TTMI**, and **RTMI**. The first two flags are controlled by global time events (Level 0,2 only) that require a reaction by the application program. With a Stop Watch Event triggered by a rising/falling edge on pin **m\_ttcanc\_swt** internal time values are captured. The Trigger Time Mark Interrupt notifies the application that a specific Time\_Base\_Trigger is reached. The Register Time Mark Interrupt signals that the time referenced by **TTOCN.TMC** (Cycle, Local, or Global) equals time mark **TTTMK.TM**. It can also be used to finish a Gap.

The fourth segment consists of flags **SOG**, **CSM**, **SMC**, and **SBC**. These flags provide a means to synchronize the application program to the communication schedule.

## 4.9 Level 0

TTCAN Level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in TTCAN Level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1.

Level 0 operation is configured via **TTOCF.OM** = “11”. In this mode the M\_TTCAN operates in event-driven CAN communication, there is no fixed schedule, the configuration of **TTOCF.GEN** is ignored. External event-synchronized operation is not available in Level 0. A synchronized time base is maintained by transmission of reference messages.

In Level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (**TTIR.TTMI**) is set when the cycle time has reached **TTOCF.IRTO** • 0x200, it reminds the Host to set a transmission request for message buffer 0. The Watch\_Trigger interrupt flag (**TTIR.WT**) is set when the cycle time has reached 0xFF00. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks.

Register time mark interrupts (**TTIR.RTMI**) are also possible.

The reference message is configured as for Level 2 operation. Received reference messages are recognized by the identifier configured in register TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the Host sets a transmission request for message buffer 0, there is no reference trigger offset.

Level 0 operation is configured via:

- **TTRMC**
- **TTOCF** except **EVTP, AWL, GEN**
- **TTMLM** except **ENTT, TXEW**
- **TURCF**

Level 0 operation is controlled via:

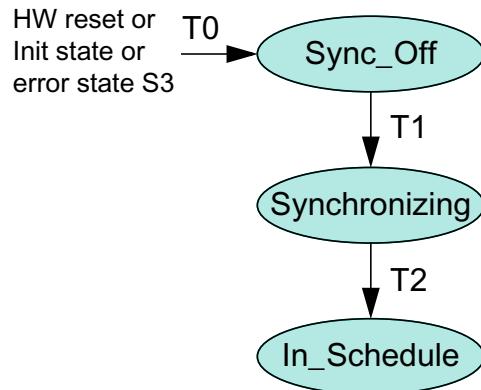
- **TTOCN** except **NIG, TMG, FGP, GCS, TTMIE**
- **TTGTP**
- **TTTMK**
- **TTIR** excluding bits **CER, AW, IWT SE2, SE1, TXO, TXU, SOG** (no function)
- **TTIR** the following bits have changed function
  - **TTMI** not defined by trigger memory - activated at cycle time **TTOCF.IRTO** • 0x200
  - **WT** not defined by trigger memory - activated at cycle time 0xFF00

Level 0 operation is signalled via:

- **TTOST** excluding bits **AWE, WFE, GSI, GFI, RTO** (no function)

#### 4.9.1 Synchronizing

Figure 15 below describes the states and state transitions in TTCAN Level 0 operation. Level 0 has no In\_Gap state.



T0: transition condition always taking prevalence

T1: Init state left, cycle time is zero

T2: at least two successive reference messages observed  
(last reference message did not contain a set Disc\_Bit bit)

Figure 15 Level 0 schedule synchronization state machine

#### 4.9.2 Handling of Error Levels

During Level 0 operation only the following error conditions may occur:

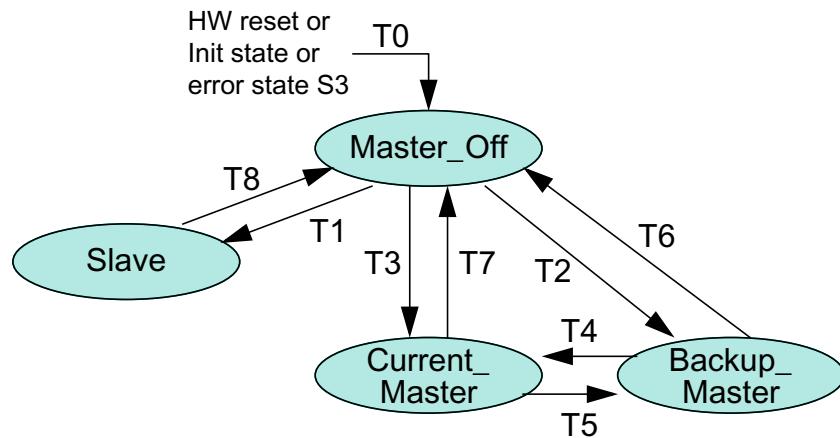
- Watch\_Trigger\_Reached (S3), reached cycle time 0xFF00
- CAN\_Bus\_Off (S3)

Since no S1 and S2 error are possible, the error level can only switch between S0 (No Error) and S3 (Severe Error). In TTCAN Level 0 an S3 error is handled differently. When error level S3 is reached, both **TTOST.SYS** and **TTOST.MS** are reset, and interrupt flags **TTIR.GTE** and **TTIR.GTD** are set.

When error level S3 (**TTOST.EL** = “11”) is entered, bus monitoring mode is, contrary to TTCAN Level 1 and Level 2, not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.

#### 4.9.3 Master Slave Relation

Figure 16 below describes the master slave relation in TTCAN Level 0. In case of an S3 error the M\_TTCAN returns to state Master\_Off.



T0: transition condition always taking prevalence

T1: reference message observed when not potential time master

T2: reference message observed with master priority ≠ own master priority, error state ≠ S3

T3: reference message observed with master priority = own master priority, error state ≠ S3

T4: reference message observed with own master priority

T5: reference message observed with master priority higher than own master priority

T6: error state S3

T7: error state S3

T8: error state S3

*Figure 16 Level 0 master to slave relation*

## 4.10 Asynchronous Serial Communication

When configured for TTCAN Level 1 or Level 2 operation, the M\_TTCAN time base can be used to switch access to the CAN bus for predefined time windows between M\_TTCAN and an ASC module (see Figure 17).

When an exclusive time window is assigned to the ASC module, the multiplexer connects the ASC module to the CAN transceiver. ASC transmission is free of CAN requirements for arbitration and fault confinement and therefore higher bit rates and effective payloads are possible.

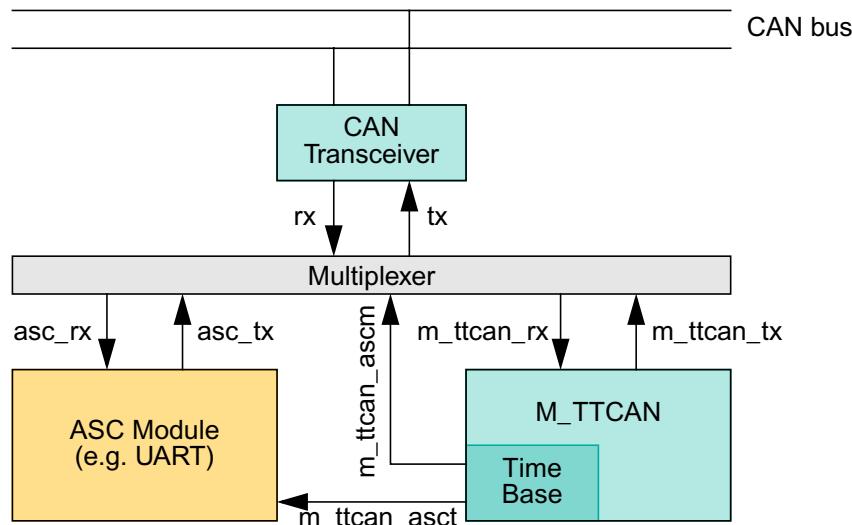


Figure 17 Asynchronous Serial Communication at CAN

Asynchronous serial operation is configured for each trigger element separately via **T0.ASC**. The M\_TTCAN's time base controls access to the CAN transceiver for M\_TTCAN or ASC module via output signal **m\_ttcn\_ascm**. Output signal **m\_ttcn\_asct** controls whether the ASC module is transmitter or receiver. For ASC transmission only one node in the network must be configured as transmitter while all other nodes are receivers.

With **T0.ASC** = "00" the ASC module is disconnected from the CAN bus (**m\_ttcn\_ascm** = '0', **m\_ttcn\_asct** = '0'). When **T0.ASC** = "01" the ASC module is receiver (**m\_ttcn\_ascm** = '1', **m\_ttcn\_asct** = '0'). When **T0.ASC** = "10" the ASC module is transmitter (**m\_ttcn\_ascm** = '1', **m\_ttcn\_asct** = '1').

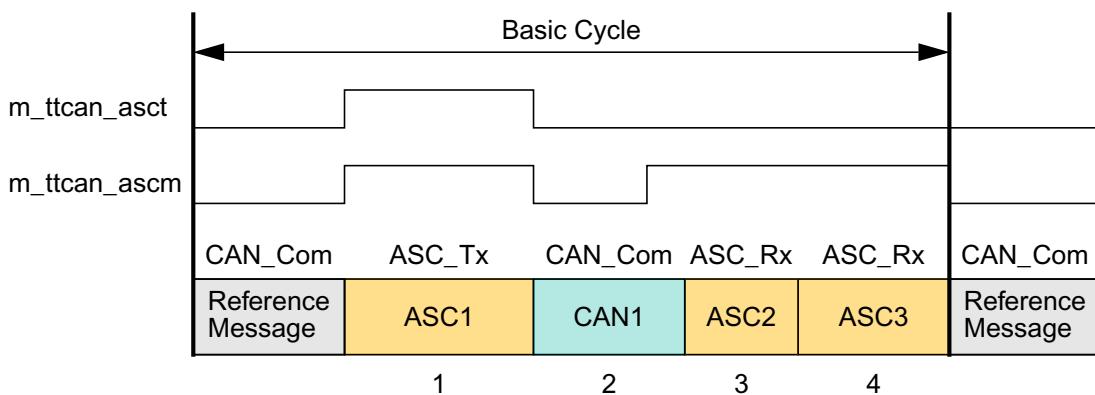


Figure 18 ASC@CAN operation

There are separate time windows for ASC transmission and reception. For each ASC time window there is one predefined transmitter, all other nodes are configured as receivers.

To start with an ASC window, the Host has to set **CCCR.ASM** during initialization. The bit is reset at the end of the first ASC window after the M\_TTCAN has finished initializing. During time-triggered operation **CCCR.ASM** is set by the M\_TTCAN at the beginning of an ASC window (trigger memory element with **T0.ASC** = "10", "11"). It is reset by each trigger memory element with **T0.ASC** = "00".

When **CCCR.ASM** is set, the CAN protocol controller

- sends no error/overload frames
- does not increment **ECR.REC** and **ECR.TEC**
- waits for Bus\_Idle (11 recessive bits) after it has detected an error or overload condition
- acknowledges valid frames
- may start the transmission of a frame after it has detected Bus\_Idle (11 recessive bits)

ASC mode operation is only entered when the M\_TTCAN is synchronization state In\_Schedule or In\_Gap (**m\_ttcn\_ascm** = '0' and **m\_ttcn\_asct** = '0' when not synchronized).

#### 4.11 Synchronization to external Time Schedule

This feature can be used to synchronize the phase of the M\_TTCAN's schedule to an external schedule (e.g. that of a second TTCAN network or FlexRay network). It is applicable only when the M\_TTCAN is current time master (**TTOST.MS** = "11").

External synchronization is controlled by event trigger input pin **m\_ttcn\_evt**. If bit **TTOCN.ESCN** is set, a rising edge at pin **m\_ttcn\_evt** the M\_TTCAN compares its actual cycle time with the target phase value configured by **TTGTP.CTP**.

Before setting **TTOCN.ESCN** the Host has to adapt the phases of the two time schedules e.g. by using the TTCAN gap control (see Section 4.3). When the Host sets **TTOCN.ESCN**, **TTOST.SPL** is set.

If the difference between the cycle time and the target phase value **TTGTP.CTP** at the rising edge at pin **m\_ttcn\_evt** is greater than 9 NTU, the phase lock bit **TTOST.SPL** is reset, and interrupt flag **TTIR.CSM** is set. **TTOST.SPL** is also reset (and **TTIR.CSM** is set), when another node becomes time master.

If both **TTOST.SPL** and **TTOCN.ESCN** are set, and if the difference between the cycle time and the target phase value **TTGTP.CTP** at the rising edge at pin **m\_ttcn\_evt** is less or equal 9 NTU, the phase lock bit **TTOST.SPL** remains set, and the measured difference is used as reference trigger offset value to adjust the phase at the next transmitted reference message.

**Note:** *The rising edge detection at pin m\_ttcn\_evt is enabled with the start of each basic cycle.*

*The first rising edge triggers the compare of the actual cycle time with TTGTP.CTP. All further edges until the beginning of the next basic cycle are ignored.*



# Chapter 5.

## 5. Appendix

### 5.1 Register Bit Overview

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Symbol Reset Value
0x000	REL[4:0]		STEP[4:0]		SUBSTEP [4:0]		YEAR[4:0]					MON[7:0]																			CREL rrrd_dddd		
0x004																															ENDN 8765_4321		
0x008																															CUST t.b.d.		
0x0C					TDCO[4:0]				TDC				FBRP[4:0]																			FBTP 0000_0A33	
0x010																							TDCV[5:0]		RX	TX[1:0]	FTSEG2[2:0]					TEST 0000_0000	
0x014																							WDV[7:0]			WDC[7:0]					RWD 0000_0000		
0x018																								TXP	FDBS	FDO	CMR[1:0]	CME[1:0]	TEST 0000_0001				
0x01C													BRP[9:0]												TSEG2[3:0]			SJW[3:0]				BTP 0000_0A33	
0x020																							TCP[3:0]								TSCC 0000_0000		
0x024																															TSCV 0000_0000		
0x028													TOP[15:0]																		TOCC FFFF_0000		

Table 83 M\_TTCAN Register Overview

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Symbol Reset Value
0x02C																															TOCV 0000_FFFF		
0x040																															ECR 0000_0000		
0x44																															PSR 0000_0707		
0x50																															IR 0000_0000		
0x54																															ILS 0000_0000		
0x58																															ILE 0000_0000		
0x05C																															SIDFC 0000_0000		
0x080																															XIDFC 0000_0000		
0x084																															EIDAM 1FFF_FFFF		
0x088																															HPMS 0000_0000		
0x090																															NDAT1 0000_0000		
0x094																															GFC 0000_0000		
0x98																																	

Table 83 M\_TTCAN Register Overview

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Symbol Reset Value
0x9C			ND63																												NDAT2 0000_0000		
0xA0	F0OM			F0WM[6:0]												F0S[6:0]															RXF0C 0000_0000		
0x0A4																	F0PI[5:0]														RXF0S 0000_0000		
0x0A8																														RXF0A 0000_0000			
0xAC																														RXBC 0000_0000			
0xB0	F1OM			F1WM[6:0]												F1S[6:0]														RXF1C 0000_0000			
0xB4		DMS[1:0]															F1PI[5:0]														RXF1S 0000_0000		
0x0B8																														RXF1A 0000_0000			
0xBC																														RXESC 0000_0000			
0x0C0	TFQM				TFQS[5:0]												NDTB[5:0]														TXBC 0000_0000		
0x0C4		TFQF															TFQPI[4:0]														TXFQS 0000_0000		
0xC8																		TFGI[4:0]														TXESC 0000_0000	
0xCC	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0	TXBRP 0000_0000

Table 83 M\_TTCAN Register Overview

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Symbol Reset Value
0x0D0																																TXBAR 0000_0000	
0x0D4																																TXBCR 0000_0000	
0x0D8																																TXBTO 0000_0000	
0x0DC																																TXBCF 0000_0000	
0x0E0																																TXBTIE 0000_0000	
0x0E4																																TXBCIE 0000_0000	
0x0F0																																TXEFC 0000_0000	
0x0F4																																TXEFS 0000_0000	
0x0F8																																TXEFA 0000_0000	
0x100																																TTTMC 0000_0000	
0x104	RMPS	XTD																														TTRMC 0000_0000	
0x108	EVTP	ECC	EGTF																													TTMC 0000_0000	
0x10C																																TTOCF 0001_0000	

**Table 83 M\_TTCAN Register Overview**

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Symbol Reset Value		
0x110	ELT																														TURCF 1000_0000				
0x114																															TTOCN 0000_0000				
0x118																															TTGTP 0000_0000				
0x11C	LCKM																														TTTMK 0000_0000				
0x120																															TTIR 0000_0000				
0x124																															TTIE 0000_0000				
0x128																															TTILS 0000_0000				
0x12C	SPL		WECS	AWE	WFE	GSI												CERL	CERE	AVL	WTL	IWTL	TXO	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	QCS	GTWL	RTMIL	SWS[1:0]	TTOST 0000_0080
0x130																															TURNA 0000_0000				
0x134																															TTLGT 0000_0000				
0x138																															TTCTC 003F_0000				

Table 83 M\_TTCAN Register Overview

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Symbol Reset Value
0x13C	SWV[15:0]																								CCV[5:0]	TTCPT 0000_0000							
0x140																															TTCSTM 0000_0000		

Table 83 M\_TTCAN Register Overview

## 5.2 Module Interface

The M\_TTCAN module's toplevel entity has the ports listed in the table below. More details on how to connect the M\_TTCAN to a customer-specific design can be found in [3] and [4].

PORT	DIR	DOMAIN	DESCRIPTION
Clocks and Reset			
m_ttcn_hclk	IN	HCLK	Host clock
m_ttcn_cclk	IN	CCLK	CAN clock
m_ttcn_reset	IN	ASYNC	module reset
Physical Layer Interface			
m_ttcn_rx	IN	ASYNC	CAN receive input
m_ttcn_tx	OUT	CCLK	CAN transmit output
Generic Slave Interface			
m_ttcn_aei_sel	IN	HCLK	module select
m_ttcn_aei_w1r0	IN	HCLK	module write
m_ttcn_aei_byteen[3:0]	IN	HCLK	module byte enable
m_ttcn_aei_addr[8:2]	IN	HCLK	module address bus
m_ttcn_aei_wdata[31:0]	IN	HCLK	module data bus input
m_ttcn_aei_ready	OUT	HCLK	module ready
m_ttcn_aei_rdata[31:0]	OUT	HCLK	module data bus output
Generic Master Interface			
m_ttcn_aeim_ready	IN	HCLK	memory ready
m_ttcn_aeim_rdata[31:0]	IN	HCLK	memory data bus input
m_ttcn_aeim_berr[1:0]	IN	HCLK	Message RAM bit error
m_ttcn_aeim_sel	OUT	HCLK	memory select
m_ttcn_aeim_w1r0	OUT	HCLK	memory write
m_ttcn_aeim_addr[15:2]	OUT	HCLK	memory address bus, 32-bit word address
m_ttcn_aeim_wdata[31:0]	OUT	HCLK	memory data bus output
Miscellaneous			
m_ttcn_ext_ts[15:0]	IN	HCLK	external timestamp vector
m_ttcn_clkstop_req	IN	HCLK	clock stop request
m_ttcn_scanmode	IN	ASYNC	scan mode enable
m_ttcn_dis_mord	IN	HCLK	disable modification on read <b>(ECR.CEL, PSR.LEC)</b>
m_ttcn_swt	IN	ASYNC	stop watch trigger
m_ttcn_evt	IN	ASYNC	event trigger
m_ttcn_int0	OUT	HCLK	interrupt 0
m_ttcn_int1	OUT	HCLK	interrupt 1
m_ttcn_clkstop_ack	OUT	HCLK	clock stop acknowledge
m_ttcn_soc	OUT	HCLK	start of cycle
m_ttcn_tmpr	OUT	CCLK	trigger time mark interrupt pulse
m_ttcn_rtp	OUT	CCLK	register time mark interrupt pulse
m_ttcn_ascm	OUT	HCLK	ASC multiplexer control
m_ttcn_asct	OUT	HCLK	ASC transmit control

Table 84 M\_TTCAN Module Interface

PORT	DIR	DOMAIN	DESCRIPTION
DMA Interface			
m_ttcn_dma_ack	IN	HCLK	DMA acknowledge
m_ttcn_dma_req	OUT	HCLK	DMA request
Extension Interface			
m_ttcn_cok	IN	HCLK	calibration OK, has to be hardwired to '1' in case no Clock Calibration on CAN unit is connected
m_ttcn_ir[31:0]	OUT	HCLK	Interrupt Register flags
m_ttcn_ttir[18:0]	OUT	HCLK	TT Interrupt Register flags
m_ttcn_txbrp[31:0]	OUT	HCLK	Tx Buffer Request Pending ( <b>TXBRP</b> )
m_ttcn_rxfd	OUT	CCLK	receive fast data
m_ttcn_txfd	OUT	CCLK	transmit fast data
m_ttcn_fe[2:0]	OUT	HCLK	filter events 0..2
m_ttcn_cce	OUT	HCLK	Configuration Change Enable ( <b>CCCR.CCE</b> )
m_ttcn_spt	OUT	CCLK	sample point
m_ttcn_mrxd	OUT	CCLK	message received
m_ttcn_calf	OUT	CCLK	calibration field
m_ttcn_aff	OUT	HCLK	acceptance filtering finished

Table 84 M\_TTCAN Module Interface

**Note:** Signals m\_ttcn\_cok, m\_ttcn\_spt, m\_ttcn\_mrxd, m\_ttcn\_calf, m\_ttcn\_aff are interfacing to an optional Clock Calibration on CAN unit. In case the M\_TTCAN is used without Clock Calibration on CAN unit, input m\_ttcn\_cok has to be hardwired to '1'.

### 5.3 Connection to external Message RAM

The M\_TTCAN is prepared for connection to a module-external single-ported or dual-ported 32-bit Message RAM via its Generic Master Interface. Depending on the system requirements additional logic for Message RAM arbitration, initialization, parity checking, or ECC has to be attached to the Message RAM. Further information with respect to Message RAM connection can be found in [3] and [4].

### 5.4 Interface to DMA Controller

When all three debug messages A, B, C have been received in the correct order, M\_TTCAN output signal **m\_ttcanc\_dma\_req** is activated to trigger a DMA transfer. The RAM words holding debug messages A, B, C will not be changed by the M\_TTCAN while **m\_ttcanc\_dma\_req** is active.

After the transfer of the received messages has completed the DMA unit activates input signal **m\_ttcanc\_dma\_ack**. This resets **m\_ttcanc\_dma\_req**. The debug message handling state machine enters idle state (DMS = "00") and waits for reception of the next debug messages (see Figure 10, *Debug Message Handling State Machine*).

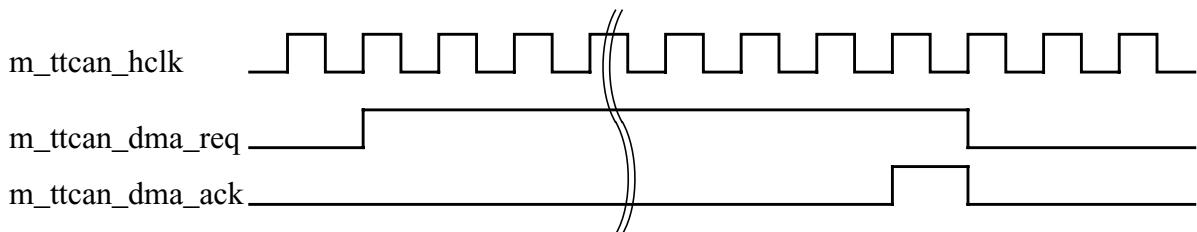


Figure 19 Timing of DMA Interface Signals

**Note:** If the DMA unit activates input signal **m\_ttcanc\_dma\_ack** before the DMA transfer has completed, the Rx Buffer elements holding debug messages A, B, C are unlocked and may be overwritten by received debug messages.



# List of Figures

Figure 1	M_TTCAN Block Diagram .....	2
Figure 2	Message RAM Configuration.....	63
Figure 3	Transceiver delay measurement .....	77
Figure 4	Pin Control in Bus Monitoring Mode .....	79
Figure 5	Pin Control in Loop Back Modes .....	81
Figure 6	Standard Message ID Filter Path.....	84
Figure 7	Extended Message ID Filter Path .....	85
Figure 8	Rx FIFO Status .....	86
Figure 9	Rx FIFO Overflow Handling .....	87
Figure 10	Debug Message Handling State Machine .....	90
Figure 11	Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO .....	93
Figure 12	Example of mixed Configuration Dedicated Tx Buffers / Tx Queue.....	94
Figure 13	Cycle Time and Global Time Synchronization.....	108
Figure 14	TTCAN Level 0 and Level 2 Drift Compensation .....	109
Figure 15	Level 0 schedule synchronization state machine.....	116
Figure 16	Level 0 master to slave relation .....	117
Figure 17	Asynchronous Serial Communication at CAN .....	118
Figure 18	ASC@CAN operation .....	118
Figure 19	Timing of DMA Interface Signals .....	129

# List of Tables

Table 1	M_TTCAN Register Map .....	5
Table 2	Core Release Register (addresses 0x000) .....	7
Table 3	Example for Coding of Revisions.....	7
Table 4	Endian Register (address 0x004) .....	8
Table 5	Fast Bit Timing & Prescaler Register (address 0x0C) .....	8
Table 6	Test Register (address 0x10).....	9
Table 7	RAM Watchdog (address 0x14).....	11
Table 8	CC Control Register (address 0x18) .....	12
Table 9	Bit Timing & Prescaler Register (address 0x1C) .....	14
Table 10	Timestamp Counter Configuration (address 0x020).....	15
Table 11	Timestamp Counter Value (address 0x024) .....	15
Table 12	Timeout Counter Configuration (address 0x28) .....	16
Table 13	Timeout Counter Value (address 0x02C) .....	16
Table 14	Error Counter Register (address 0x40) .....	17
Table 15	Protocol Status Register (address 0x44).....	18
Table 16	Interrupt Register (address 0x50).....	20
Table 17	Interrupt Enable (address 0x54) .....	23
Table 18	Interrupt Line Select (address 0x58) .....	25
Table 19	Interrupt Line Enable (address 0x05C).....	26
Table 20	Global Filter Configuration (address 0x080).....	27
Table 21	Standard ID Filter Configuration (address 0x084) .....	28
Table 22	Extended ID Filter Configuration (address 0x088) .....	28
Table 23	Extended ID AND Mask (address 0x090).....	29
Table 24	High Priority Message Status (address 0x094) .....	29
Table 25	New Data 1 (address 0x98).....	30
Table 26	New Data 2 (address 0x9C) .....	30
Table 27	Rx FIFO 0 Configuration (address 0xA0) .....	31
Table 28	Rx FIFO 0 Status (address 0x0A4) .....	32
Table 29	Rx FIFO 0 Acknowledge (address 0x0A8) .....	33
Table 30	Rx Buffer Configuration (address 0xAC) .....	33
Table 31	Rx FIFO 1 Configuration (address 0xB0) .....	34
Table 32	Rx FIFO 1 Status (address 0x0B4) .....	34
Table 33	Rx FIFO 1 Acknowledge (address 0x0B8) .....	35
Table 34	Rx Buffer / FIFO Element Size Configuration (address 0xBC) .....	36
Table 35	Tx Buffer Configuration (address 0xC0) .....	37
Table 36	Tx FIFO/Queue Status (address 0xC4).....	38
Table 37	Tx Buffer Element Size Configuration (address 0xC8) .....	39
Table 38	Tx Buffer Request Pending (address 0x0CC).....	40
Table 39	Tx Buffer Add Request (address 0x0D0).....	41
Table 40	Tx Buffer Cancellation Request (address 0x0D4) .....	41
Table 41	Tx Buffer Transmission Occurred (address 0x0D8).....	42
Table 42	Transmit Buffer Cancellation Finished (address 0x0DC) .....	42
Table 43	Tx Buffer Transmission Interrupt Enable (address 0xE0) .....	43
Table 44	Tx Buffer Cancellation Finished Interrupt Enable (address 0xE4) .....	43
Table 45	Tx Event FIFO Configuration (address 0xF0).....	44
Table 46	Tx Event FIFO Status (address 0xF4) .....	45
Table 47	Tx Event FIFO Acknowledge (address 0xF8).....	45
Table 48	TT Trigger Memory Configuration (address 0x100).....	46
Table 49	TT Reference Message Configuration (address 0x104).....	46
Table 50	TT Operation Configuration (address 0x108) .....	47
Table 51	TT Matrix Limits (address 0x10C) .....	48

Table 52	TUR Configuration (address 0x110) .....	49
Table 53	TT Operation Control (address 0x114) .....	50
Table 54	TT Global Time Preset (address 0x118).....	52
Table 55	TT Time Mark (address 0x11C).....	53
Table 56	TT Interrupt Register (address 0x120) .....	54
Table 57	TT Interrupt Enable (address 0x124).....	56
Table 58	TT Interrupt Line Select (address 0x128).....	57
Table 59	TT Operation Status (address 0x12C).....	58
Table 60	TUR Numerator Actual (address 0x130) .....	60
Table 61	TT Local & Global Time (address 0x134).....	60
Table 62	TT Cycle Time & Count (address 0x138) .....	61
Table 63	TT Capture Time (address 0x13C) .....	61
Table 64	TT Cycle Sync Mark (address 0x140) .....	62
Table 65	Rx Buffer and FIFO Element .....	64
Table 66	Tx Buffer Element.....	66
Table 67	Tx Event FIFO Element .....	68
Table 68	Standard Message ID Filter Element.....	69
Table 69	Extended Message ID Filter Element .....	70
Table 70	Trigger Memory Element .....	72
Table 71	Coding of DLC in CAN FD .....	77
Table 72	Rx Buffer / FIFO Element Size .....	86
Table 73	Example Filter Configuration for Rx Buffers.....	88
Table 74	Example Filter Configuration for Debug Messages .....	89
Table 75	Tx Buffer / FIFO / Queue Element Size .....	92
Table 76	First byte of Level 1 reference message .....	97
Table 77	First four bytes of Level 2 reference message .....	98
Table 78	First four bytes of Level 0 reference message .....	98
Table 79	TUR Configuration Examples .....	100
Table 80	System Matrix Node A .....	104
Table 81	Trigger List Node A .....	104
Table 82	Number of Data Bytes transmitted with a reference messages.....	111
Table 83	M_TTCAN Register Overview .....	121
Table 84	M_TTCAN Module Interface .....	127