BOSCH

M_CAN

Controller Area Network

User's Manual

Revision 3.2.1.2

20.02.2017

Robert Bosch GmbH
Automotive Electronics

LEGAL NOTICE

SPECIFICATION REVISION HISTORY

REVISION	***DATE***	***NOTES***
0.1	18.01.2008	initial working revision
0.2	12.02.2008	first revised working revision
0.3	10.03.2008	second revised working revision
0.4	18.04.2008	third revised working revision
0.5	22.07.2008	fourth revised working revision
0.6	24.10.2008	fifth revised working revision
0.7	18.12.2008	sixth revised working revision
1.0	25.03.2009	first complete revision
1.1	25.06.2009	Tx Handler functionality updated
1.2	20.08.2009	register TXBSC removed, RAM Watchdog added
1.21	10.02.2010	address 0x08 reserved for customer defined register
1.22	26.11.2010	minor textual enhancements
1.23	18.02.2011	typos corrected
2.0	27.10.2011	debug on CAN, dedicated Rx Buffers, CAN FD, Extension IF
2.0.1	12.03.2012	minor corrections, interface signals to Clock Calibration on CAN unit updated
2.0.2	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	26.11.2012	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

REVISION	***DATE***	***NOTES***
3.1.0	22.07.2014	Register FBTP renamed to DBTP and restructured • **TDCO** moved to new register TDCR • increased configuration range for data bit timing Register TEST restructured • **TDCV** moved to register PSR Register CCCR restructured • FDBS and FDO removed • new control bit **EFBI** replaces status flag **FDBS** • new control bit **PXHD** replaces status flag **FDO** • **CMR** removed, transmit format configured in Tx Buffer element • **CME** replaced by **FDOE** and **BRSE** Register BTP renamed to NBTP and restructured • **BRP** renamed to **NBRP**, range reduced • **TSEG1** renamed to **NTSEG1**, range expanded • **TSEG2** renamed to **NTSEG2**, range expanded • **SJW** renamed to **NSJW**, range expanded Register PSR updated • **TDCV** moved from register TEST, range increased • status flag **PXE** added • **FLEC** renamed to **DLEC** Register TDCR added • **TDCO** moved from register DBTP, range expanded • new configuration **TDCF** field Register IR updated • interrupt flags **STE**, **FOE**, **ACKE**, **BE**, **CRCE** replaced by **ARA**, **PED**, **PEA** Register IE updated • interrupt enable bits **STEE**, **FOEE**, **ACKEE**, **BEE**, **CRCEE** replaced by **ARAE**, **PEDE**, **PEAE** Register ILS updated • interrupt line select bits **STEL**, **FOEL**, **ACKEL**, **BEL**, **CRCEL** replaced by **ARAL**, **PEDL**, **PEAL** Rx buffer and FIFO element updated • bit **EDL** renamed to **FDF** Tx buffer element updated • transmission of bit **ESI** recessive configurable • selection of Classic/FD format transmission via flag **FDF** • configuration of bit rate switching via **BRS** Section 3.1.3 CAN FD Operation updated Section 3.1.4 Transmitter Delay Compensation updated Minor amendments and textual enhancements
3.1.5	14.10.2014	Bit **NISO** added to register CCCR
3.2.0	07.11.2014	Table 61: description of m_can_dis_mord updated Baud Rate replaced by Bit Rate Note about Message RAM initialization added
3.2.1	16.03.2015	minor textual enhancements and corrections
3.2.1.1	24.03.2016	References to ISO 11898-1 updated, range of **NBTP.NTSEG2** updated to fix erratum #16.
3.2.1.2	20.02.2017	Description of **DBTP.DBRP** in Section 2.3.4 enhanced.

TRACKING OF MAJOR CHANGES

TERMS AND ABBREVIATIONS

This document uses the following terms and abbreviations.

Term	Meaning
BRP	Bit 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
FSM	Finite State Machine
mtq	minimum time quantum = CAN clock period (**m_can_cclk**)
SSP	Secondary Sample Point
TDC	Transmitter Delay Compensation
tq	time quantum
TSEG1	Time Segment before Sample Point
TSEG2	Time Segment after Sample Point
TTCAN	Time-Triggered CAN

CONVENTIONS

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

Arial bold	Names of bits and ports
Arial italic	States of bits and ports

REFERENCES

This document refers to the following documents:

Ref	Author(s)	Title
[1]	ISO	ISO 11898-1:2015: CAN data link layer and physical signalling
[2]	AE/PJ-SCI	M_(TT)CAN System Integration Guide
[3]	AE/PJ-SCI	M_CAN Module Integration Guide

Table of contents

Chapter 1.

1. Overview

The M_CAN module is the new CAN 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_CAN performs communication according to ISO 11898-1:2015. Additional transceiver hardware is required for connection to the physical layer.

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

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_CAN can be connected to a wide range of Host CPUs via its 8/16/32-bit Generic Slave Interface. The M_CAN'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 ISO 11898-1:2015
- CAN FD with up to 64 data bytes supported
- CAN Error Logging
- AUTOSAR support
- SAE J1939 support
- 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_CANs 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

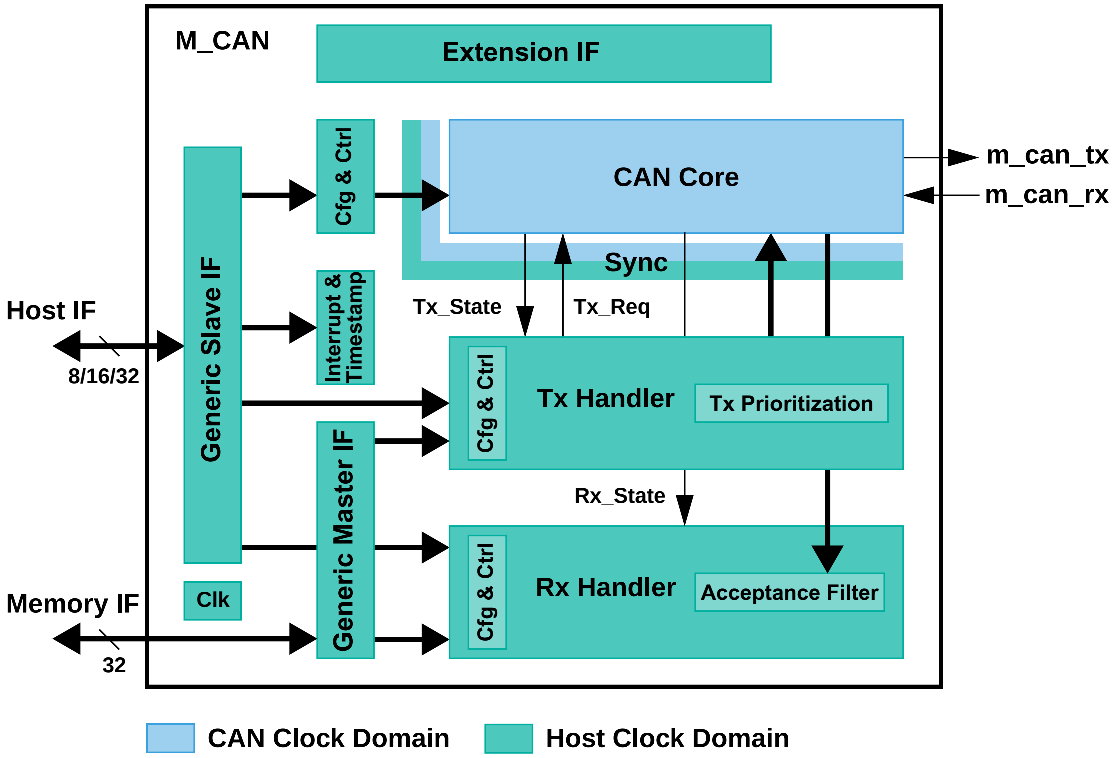


Figure 1 M_CAN Block Diagram

CAN Core

CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1:2015 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.

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_CAN 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_CAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16K • 32 bit. A single M_CAN can use at most 4.25K • 32 bit.

Extension Interface

All flags from the Interrupt Register IR as well as selected internal status and control signals are routed to this interface. The interface is intended for connection of the M_CAN 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_can_hclk**. Due to the high precision clocking requirements of the CAN Core, a separate clock without any modulation has to be provided as **m_can_cclk**.

Within the M_CAN 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_CAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of a 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_can_int0** or to **m_can_int1**. By default all interrupts are routed to interrupt line **m_can_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_CAN hold the reset values listed in Table 1. Additionally the *Bus_Off* state is reset and the output **m_can_tx** is set to *recessive* (HIGH). The value 0x0001 (**CCCR.INIT** = '1') in the CC Control Register enables software initialization. The M_CAN does not influence the CAN bus until the CPU resets **CCCR.INIT** to '0'.

2.2 Register Map

The M_CAN module allocates an address space of 256 bytes. All registers are organized as 32-bit registers. The M_CAN 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
0x00	CREL	Core Release Register	7	rrrd dddd	R
0x04	ENDN	Endian Register	8	8765 4321	R
0x08	CUST	Customer Register	7	t.b.d.	t.b.d.
0x0C	DBTP	Data Bit Timing & Prescaler Register	8	0000 0A33	RP
0x10	TEST	Test Register	9	0000 0000	RP
0x14	RWD	RAM Watchdog	10	0000 0000	RP
0x18	CCCR	CC Control Register	11	0000 0001	RWPp
0x1C	NBTP	Nominal Bit Timing & Prescaler Register	13	0600 0A03	RP
0x20	TSCC	Timestamp Counter Configuration	14	0000 0000	RP
0x24	TSCV	Timestamp Counter Value	14	0000 0000	RC
0x28	TOCC	Timeout Counter Configuration	15	FFFF 0000	RP
0x2C	TOCV	Timeout Counter Value	15	0000 FFFF	RC
0x30-3C		*reserved (4)*		0000 0000	R
0x40	ECR	Error Counter Register	16	0000 0000	RX
0x44	PSR	Protocol Status Register	17	0000 0707	RXS
0x48	TDCR	Transmitter Delay Compensation Register	19	0000 0000	RP
0x4C		*reserved (1)*		0000 0000	R
0x50	IR	Interrupt Register	20	0000 0000	RW
0x54	IE	Interrupt Enable	23	0000 0000	RW
0x58	ILS	Interrupt Line Select	25	0000 0000	RW
0x5C	ILE	Interrupt Line Enable	26	0000 0000	RW
0x60-7C		*reserved (8)*		0000 0000	R
0x80	GFC	Global Filter Configuration	27	0000 0000	RP
0x84	SIDFC	Standard ID Filter Configuration	28	0000 0000	RP

Table 1 M_CAN Register Map

ADDRESS	SYMBOL	NAME	PAGE	RESET	ACC
0x88	XIDFC	Extended ID Filter Configuration	28	0000 0000	RP
0x8C		*reserved (1)*		0000 0000	R
0x90	XIDAM	Extended ID AND Mask	29	1FFF FFFF	RP
0x94	HPMS	High Priority Message Status	29	0000 0000	R
0x98	NDAT1	New Data 1	30	0000 0000	RW
0x9C	NDAT2	New Data 2	30	0000 0000	RW
0xA0	RXF0C	Rx FIFO 0 Configuration	31	0000 0000	RP
0xA4	RXF0S	Rx FIFO 0 Status	32	0000 0000	R
0xA8	RXF0A	Rx FIFO 0 Acknowledge	33	0000 0000	RW
0xAC	RXBC	Rx Buffer Configuration	33	0000 0000	RP
0xB0	RXF1C	Rx FIFO 1 Configuration	34	0000 0000	RP
0xB4	RXF1S	Rx FIFO 1 Status	34	0000 0000	R
0xB8	RXF1A	Rx FIFO 1 Acknowledge	35	0000 0000	RW
0xBC	RXESC	Rx Buffer / FIFO Element Size Configuration	36	0000 0000	RP
0xC0	TXBC	Tx Buffer Configuration	37	0000 0000	RP
0xC4	TXFQS	Tx FIFO/Queue Status	38	0000 0000	R
0xC8	TXESC	Tx Buffer Element Size Configuration	39	0000 0000	RP
0xCC	TXBRP	Tx Buffer Request Pending	40	0000 0000	R
0xD0	TXBAR	Tx Buffer Add Request	41	0000 0000	RW
0xD4	TXBCR	Tx Buffer Cancellation Request	41	0000 0000	RW
0xD8	TXBTO	Tx Buffer Transmission Occurred	42	0000 0000	R
0xDC	TXBCF	Tx Buffer Cancellation Finished	42	0000 0000	R
0xE0	TXBTIE	Tx Buffer Transmission Interrupt Enable	43	0000 0000	RW
0xE4	TXBCIE	Tx Buffer Cancellation Finished Interrupt Enable	43	0000 0000	RW
0xE8-EC		*reserved (2)*		0000 0000	R
0xF0	TXEFC	Tx Event FIFO Configuration	44	0000 0000	RP
0xF4	TXEFS	Tx Event FIFO Status	45	0000 0000	R
0xF8	TXEFA	Tx Event FIFO Acknowledge	45	0000 0000	RW
0xFC		*reserved (1)*		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_CAN Register Map

2.2.1 Access to reserved Register Addresses

In case the application software wants to access one of the reserved addresses in the M_CAN register map (read or write access), interrupt flag **IR.ARA** is set, and if enable the interrupt is signalled via the assigned interrupt line (**m_can_int0** or **m_can_int1**).

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)*

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x00	REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
	R-r				R-r				R-r				R-d			

Bits	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 0x00)

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_CAN synthesis.

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

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

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

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

Release	***Step***	***SubStep***	***Year***	***Month***	***Day***	***Name***
0	2	0	9	03	26	Revision 0.2.0, Date 2009/03/26

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
0x04	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 0x04)

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

The endianness test value is 0x87654321.

2.3.4 Data Bit Timing & Prescaler Register (DBTP)

This register is only writable if bits **CCCR.CCE** and **CCCR.INIT** are set. The CAN bit time may be programed in the range of 4 to 49 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 **m_can_cclk** periods. t_q = (**DBRP** + 1) mtq.

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. **DTSEG2** is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [**DTSEG1** + **DTSEG2** + 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								TDC	res		DBRP[4:0]				
	R-0x0								RP-0	R-0x0		RP-0x0				

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res			DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
	R-0x0			RP-0xA					RP-0x3				RP-0x3			

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

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

Bit 23 TDC: Transmitter Delay Compensation

0= Transmitter Delay Compensation disabled

1= Transmitter Delay Compensation enabled

Bits 20:16 DBRP[4:0]: Data Bit 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 Bit Rate Prescaler are 0 to 31. When **TDC** = ‘1’, the range is limited to 0,1. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Bits 12:8 **DTSEG1[4:0]:** Data time segment before sample point

0x00-0x1F Valid values are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Bits 7:4 **DTSEG2[3:0]:** Data 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 **DSJW[3:0]:** Data (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_can_cclk) of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a data phase bit rate of 500 kBit/s.

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

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_can_tx** are hardware test modes. Programming of **TX** ≠ “00” may disturb the message transfer on the CAN bus.

Bits	31–16
0x10	res
	R-0x0

Bits	15–8	7	6:5	4	3–0
	res	RX	TX[1:0]	LBCK	res
	R-0x0	R-U	RP-0x0	RP-0	R-0x0

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

Table 6 Test Register (address 0x10)

Bit 7 **RX:** Receive Pin

Monitors the actual value of pin **m_can_rx**

0= The CAN bus is dominant (**m_can_rx** = ‘0’)

1= The CAN bus is recessive (**m_can_rx** = ‘1’)

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

00 Reset value, **m_can_tx** controlled by the CAN Core, updated at the end of the CAN bit time

01 Sample Point can be monitored at pin **m_can_tx**

10 Dominant (‘0’) level at pin **m_can_tx**

11 Recessive (‘1’) at pin **m_can_tx**

Bit 4 **LBCK:** Loop Back Mode

0= Reset value, Loop Back Mode is disabled

1= Loop Back Mode is enabled (see Chapter 3.1.9)

2.3.6 RAM Watchdog (RWD)

The RAM Watchdog monitors the READY output of the Message RAM (**m_can_aeim_ready**). A Message RAM access via the M_CAN's Generic Master Interface (**m_can_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_can_hclk**).

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x14	res															
	R-0x0															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WDV[7:0]								WDC[7:0]							
	R-0x0								RP-0x0							

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

Table 7 RAM Watchdog (address 0x14)

Bits 7:0 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-0x0															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NISO	TXP	EFBI	PXHD	res		BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
	RP-0	RP-0	RP-0	RP-0	R-0x0		RP-0	RP-0	Rp-0	RP-0	Rp-0	RW-0	R-0	Rp-0	RP-0	RW-1

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

Table 8 CC Control Register (address 0x18)

Bit 15 NISO: Non ISO Operation

If this bit is set, the M_CAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.

0= CAN FD frame format according to ISO 11898-1:2015

1= CAN FD frame format according to Bosch CAN FD Specification V1.0

Note: When the generic parameter iso_only_g is set to '1' in hardware synthesis, this bit becomes reserved and is read as '0'. The M_CAN always operates with the CAN FD frame format according to ISO 11898-1:2015.

Bit 14 TXP: Transmit Pause

If this bit is set, the M_CAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame (see Section 3.5).

0= Transmit pause disabled

1= Transmit pause enabled

Bit 13 EFBI: Edge Filtering during Bus Integration

0= Edge filtering disabled

1= Two consecutive dominant tq required to detect an edge for hard synchronization

Bit 12 PXHD: Protocol Exception Handling Disable

0= Protocol exception handling enabled

1= Protocol exception handling disabled

Note: When protocol exception handling is disabled, the M_CAN will transmit an error frame when it detects a protocol exception condition.

Bit 9 BRSE: Bit Rate Switch Enable

0= Bit rate switching for transmissions disabled

1= Bit rate switching for transmissions enabled

Note: When CAN FD operation is disabled FDOE = '0', BRSE is not evaluated.

Bit 8 FDOE: FD Operation Enable

0= FD operation disabled

1= FD operation 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_CAN may be set in power down by stopping **m_can_hclk** and **m_can_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 Nominal Bit Timing & Prescaler Register (NBTP)

This register is only writable if bits **CCCR.CCE** and **CCCR.INIT** are set. The CAN bit time may be programed in the range of 4 to 385 time quanta. The CAN time quantum may be programmed in the range of 1 to 512 **m_can_cclk** periods. t_q = (**NBRP** + 1) mtq.

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. **NTSEG2** is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [**NTSEG1** + **NTSEG2** + 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	NSJW[6:0]							NBRP[8:0]								
	RP-0x3							RP-0x0								
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	NTSEG1[7:0]								res	NTSEG2[6:0]						
	RP-0xA								R-0	RP-0x3						

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

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

Bits 31:25 NSJW[6:0]: Nominal (Re)Synchronization Jump Width

0x00-0x7F Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Bits 24:16 NBRP[8:0]: Nominal Bit Rate Prescaler

0x000-0x1FF 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 Bit Rate Prescaler are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Bits 15:8 NTSEG1[7:0]: Nominal Time segment before sample point

0x01-0xFF Valid values are 1 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Bits 6:0 NTSEG2[6:0]: Nominal Time segment after sample point

0x01-0x7F Valid values are 1 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Note: With a CAN clock (m_can_cclk) of 8 MHz, the reset value of 0x06000A03 configures the M_CAN 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
0x20	res												TCP[3:0]			
	R-0x0												RP-0x0			
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	res														TSS[1:0]	
	R-0x0														RP-0x0	

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

Table 10 Timestamp Counter Configuration (address 0x20)

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
0x24	res															
	R-0x0															
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	TSC[15:0]															
	RC-0x0															

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

Table 11 Timestamp Counter Value (address 0x24)

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, *Timeout Counter*.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x28	TOP[15:0]															
	RP-0xFFFF															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res													TOS[1:0]		ETOC
	R-0x0													RP-0x0		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
0x2C	res															
	R-0x0															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TOC[15:0]															
	RC-0xFFFF															

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

Table 13 *Timeout Counter Value (address 0x2C)*

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-0x0								X-0x0							
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-0x0							R-0x0							

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.

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									TDCV[6:0]						
	R-0x0									R-0x0						

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res	PXE	RFDF	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	R-0	X-0	X-0	X-0	X-0	S-0x7			R-0	R-0	R-0	R-0x0		S-0x7		

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

Table 15 Protocol Status Register (address 0x44)

Bits 22:16 TDCV[6:0]: Transmitter Delay Compensation Value

0x00-0x7F Position of the secondary sample point, defined by the sum of the measured delay from **m_can_tx** to **m_can_rx** and **TDCR.TDCO**. The SSP position is, in the data phase, the number of mtq between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.

Bit 14 PXE: Protocol Exception Event

0= No protocol exception event occurred since last read access

1= Protocol exception event occurred

Bit 13 RFDF: 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 **FDF** flag set has been received

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

This bit is set together with **RFDF**, 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 **RFDF**, 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 DLEC[2:0]: Data Phase 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_CAN is not Bus_Off

1= The M_CAN 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_CAN 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_CAN 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

Note: ACT is set to "00" by a Protocol Exception Event.

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_CAN 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 DLEC 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 ISO 11898-1:2015) 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 Transmitter Delay Compensation Register (TDCR)

Table 16 Transmitter Delay Compensation Register (address 0x048)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x048	res															
	R-0x0															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res	TDCO[6:0]							res	TDCF[6:0]						
	R-0	RP-0x0							R-0	RP-0x0						

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

Bits 14:8 **TDCO[6:0]:** Transmitter Delay Compensation Offset

0x00-0x7F Offset value defining the distance between the measured delay from **m_can_tx** to **m_can_rx** and the secondary sample point. Valid values are 0 to 127 mtq.

Bits 6:0 **TDCF[6:0]:** Transmitter Delay Compensation Filter Window Length

0x00-0x7F Defines the minimum value for the SSP position, dominant edges on **m_can_rx** that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when **TDCF** is configured to a value greater than **TDCO**. Valid values are 0 to 127 mtq.

2.3.16 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.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x50	res		ARA	PED	PEA	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW
	R-0x0		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**
	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
	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 Register (address 0x50)

Bit 29 ARA: Access to Reserved Address

0= No access to reserved address occurred
1= Access to reserved address occurred

Bit 28 PED: Protocol Error in Data Phase (Data Bit Time is used)

0= No protocol error in data phase
1= Protocol error in data phase detected (**PSR.DLEC** ≠ 0,7)

Bit 27 PEA: Protocol Error in Arbitration Phase (Nominal Bit Time is used)

0= No protocol error in arbitration phase
1= Protocol error in arbitration phase detected (**PSR.LEC** ≠ 0,7)

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_can_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_can_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 an 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_CAN 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

2.3.17 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	res		ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
	R-0x0		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 18 Interrupt Enable (address 0x54)

Bit 29 **ARAE:** Access to Reserved Address Enable

Bit 28 **PEDE:** Protocol Error in Data Phase Enable

Bit 27 **PEAE:** Protocol Error in Arbitration Phase 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

Bit 17 **MRAFE:** Message RAM Access Failure Interrupt Enable

Bit 16 **TSWE:** Timestamp Wraparound Interrupt Enable

Bit 15 **TEFLE:** Tx Event FIFO Event Lost Interrupt Enable

Bit 14 **TEFFE:** Tx Event FIFO Full Interrupt Enable

Bit 13 **TEFWE:** Tx Event FIFO Watermark Reached Interrupt Enable

Bit 12 **TEFNE:** Tx Event FIFO New Entry Interrupt Enable

Bit 11	**TFEE:**	Tx FIFO Empty Interrupt Enable
Bit 10	**TCFE:**	Transmission Cancellation Finished Interrupt Enable
Bit 9	**TCE:**	Transmission Completed Interrupt Enable
Bit 8	**HPME:**	High Priority Message Interrupt Enable
Bit 7	**RF1LE:**	Rx FIFO 1 Message Lost Interrupt Enable
Bit 6	**RF1FE:**	Rx FIFO 1 Full Interrupt Enable
Bit 5	**RF1WE:**	Rx FIFO 1 Watermark Reached Interrupt Enable
Bit 4	**RF1NE:**	Rx FIFO 1 New Message Interrupt Enable
Bit 3	**RF0LE:**	Rx FIFO 0 Message Lost Interrupt Enable
Bit 2	**RF0FE:**	Rx FIFO 0 Full Interrupt Enable
Bit 1	**RF0WE:**	Rx FIFO 0 Watermark Reached Interrupt Enable
Bit 0	**RF0NE:**	Rx FIFO 0 New Message Interrupt Enable

2.3.18 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_can_int0**

1= Interrupt assigned to interrupt line **m_can_int1**

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x58	res		ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL
	R-0x0		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
	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
	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 19 Interrupt Line Select (address 0x58)

Bit 29	**ARAL:**	Access to Reserved Address Line
Bit 28	**PEDL:**	Protocol Error in Data Phase Line
Bit 27	**PEAL:**	Protocol Error in Arbitration Phase 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
Bit 13	**TEFWL:**	Tx Event FIFO Watermark Reached Interrupt Line
Bit 12	**TEFNL:**	Tx Event FIFO New Entry Interrupt Line

Bit 11 **TFEL:** Tx FIFO Empty Interrupt Line

Bit 10 **TCFL:** Transmission Cancellation Finished Interrupt Line

Bit 9 **TCL:** Transmission Completed Interrupt Line

Bit 8 **HPML:** High Priority Message Interrupt Line

Bit 7 **RF1LL:** Rx FIFO 1 Message Lost Interrupt Line

Bit 6 **RF1FL:** Rx FIFO 1 Full Interrupt Line

Bit 5 **RF1WL:** Rx FIFO 1 Watermark Reached Interrupt Line

Bit 4 **RF1NL:** Rx FIFO 1 New Message Interrupt Line

Bit 3 **RF0LL:** Rx FIFO 0 Message Lost Interrupt Line

Bit 2 **RF0FL:** Rx FIFO 0 Full Interrupt Line

Bit 1 **RF0WL:** Rx FIFO 0 Watermark Reached Interrupt Line

Bit 0 **RF0NL:** Rx FIFO 0 New Message Interrupt Line

2.3.19 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	31–16		
0x5C	res (R-0x0)		
Bits	**15–2**	**1**	**0**
	res (R-0x0)	EINT1 (RW-0)	EINT0 (RW-0)

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

Table 20 Interrupt Line Select (address 0x5C)

Bit 1 **EINT1:** Enable Interrupt Line 1

0= Interrupt line **m_can_int1** disabled

1= Interrupt line **m_can_int1** enabled

Bit 0 **EINT0:** Enable Interrupt Line 0

0= Interrupt line **m_can_int0** disabled

1= Interrupt line **m_can_int0** enabled

2.3.20 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
0x80	res															
	R-0x0															

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-0x0										RP-0x0		RP-0x0		RP-0	RP-0

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

Table 21 Global Filter Configuration (address 0x80)

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

2.3.21 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
0x84	res								LSS[7:0]							
	R-0x0								RP-0x0							
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	FLSSA[15:2]														res	
	RP-0x0														R-0x0	

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

Table 22 Standard ID Filter Configuration (address 0x84)

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).

2.3.22 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
0x88	res									LSE[6:0]						
	R-0x0									RP-0x0						
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	FLESA[15:2]														res	
	RP-0x0														R-0x0	

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

Table 23 Extended ID Filter Configuration (address 0x88)

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).

2.3.23 Extended ID AND Mask (XIDAM)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x90	res			EIDM[28:16]												
	R-0x0			RP-0x1FFF												
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	EIDM[15:0]															
	RP-0xFFFF															

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

Table 24 Extended ID AND Mask (address 0x90)

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.

2.3.24 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
0x94	res															
	R-0x0															
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-0x0							R-0x0		R-0x0					

R = Read; -n = Value after reset

Table 25 High Priority Message Status (address 0x94)

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’.

2.3.25 New Data 1 (NDAT1)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x98	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
	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**
	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
	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; -n = value after reset

Table 26 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

2.3.26 New Data 2 (NDAT2)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x9C	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
	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**
	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
	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; -n = value after reset

Table 27 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

2.3.27 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-0x0							R-0	RP-0x0						

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	F0SA[15:2]														res	
	RP-0x0														R-0x0	

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

Table 28 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).

2.3.28 Rx FIFO 0 Status (RXF0S)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xA4	res						RF0L	F0F	res		F0PI[5:0]					
	R-0x0						R-0	R-0	R-0x0		R-0x0x0					

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res		F0GI[5:0]						res	F0FL[6:0]						
	R-0x0		R-0x0						R-0	R-0x0						

R = Read; -n = value after reset

Table 29 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.

2.3.29 Rx FIFO 0 Acknowledge (RXF0A)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xA8	res															
	R-0x0															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res										F0AI[5:0]					
	R-0x0										RW-0x0					

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

Table 30 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**.

2.3.30 Rx Buffer Configuration (RXBC)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xAC	res															
	R-0x0															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RBSA[15:2]														res	
	RP-0x0														R-0x0	

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

Table 31 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.

2.3.31 Rx FIFO 1 Configuration (RXF1C)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xB0	F1OM	F1WM[6:0]							res	F1S[6:0]						
	RP-0	RP-0x0							R-0	RP-0x0						

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	F1SA[15:2]														res	
	RP-0x0														R-0x0	

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

Table 32 Rx FIFO 1 Configuration (address 0xB0)

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).

2.3.32 Rx FIFO 1 Status (RXF1S)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xB4	DMS[1:0]		res				RF1L	F1F	res		F1PI[5:0]					
	R-0x0		R-0x0				R-0	R-0	R-0x0		R-0x0					

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-0x0		R-0x0						R-0	R-0x0						

R = Read; -n = value after reset

Table 33 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.

2.3.33 Rx FIFO 1 Acknowledge (RXF1A)

Bits	***31***	***30***	***29***	***28***	***27***	***26***	***25***	***24***	***23***	***22***	***21***	***20***	***19***	***18***	***17***	***16***
0xB8	res															
	R-0x0															

Bits	***15***	***14***	***13***	***12***	***11***	***10***	***9***	***8***	***7***	***6***	***5***	***4***	***3***	***2***	***1***	***0***
	res										F1AI[5:0]					
	R-0x0										RW-0x0					

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

Table 34 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**.

2.3.34 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.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xBC	res															
	R-0x0															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res					RBDS[2:0]			res	F1DS[2:0]			res	F0DS[2:0]		
	R-0x0					RP-0x0			R-0	RP-0x0			R-0	RP-0x0		

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

Table 35 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.*

2.3.35 Tx Buffer Configuration (TXBC)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xC0	res	TFQM	TFQS[5:0]						res		NDTB[5:0]					
	R-0	RP-0	RP-0x0						R-0x0		RP-0x0					
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	TBSA[15:2]														res	
	RP-0x0														R-0x0	

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

Table 36 Tx Buffer Configuration (address 0xC0)

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.

2.3.36 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	res										TFQF	TFQPI[4:0]				
	R-0x0										R-0	R-0x0				

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-0x0			R-0x0					R-0x0		R-0x0					

R = Read; -n = value after reset

Table 37 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.

2.3.37 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.

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xC8	res															
	R-0x0															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	res													TBDS[2:0]		
	R-0x0													RP-0x0		

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

Table 38 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).

BOSCH

2.3.38 Tx Buffer Request Pending (TXBRP)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xCC	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
	R-0	R-0	R-0	R-0	R-0	R-0	R-0	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**
	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R = Read; -n = value after reset

Table 39 Tx Buffer Request Pending (address 0xCC)

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, *Tx Handling*) 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.

2.3.39 Tx Buffer Add Request (TXBAR)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xD0	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
	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**
	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
	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 40 Tx Buffer Add Request (address 0xD0)

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.

2.3.40 Tx Buffer Cancellation Request (TXBCR)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xD4	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16
	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**
	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
	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 41 Tx Buffer Cancellation Request (address 0xD4)

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

2.3.41 Tx Buffer Transmission Occurred (TXBTO)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xD8	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16
	R-0	R-0	R-0	R-0	R-0	R-0	R-0	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**
	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R = Read; -n = value after reset

Table 42 Tx Buffer Transmission Occurred (address 0xD8)

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

2.3.42 Tx Buffer Cancellation Finished (TXBCF)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xDC	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
	R-0	R-0	R-0	R-0	R-0	R-0	R-0	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**
	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R = Read; -n = value after reset

Table 43 Transmit Buffer Cancellation Finished (address 0xDC)

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

2.3.43 Tx Buffer Transmission Interrupt Enable (TXBTIE)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xE0	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16
	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**
	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
	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 44 Tx Buffer Transmission Interrupt Enable (address 0xE0)

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

2.3.44 Tx Buffer Cancellation Finished Interrupt Enable (TXBCIE)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xE4	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16
	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**
	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
	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 45 Tx Buffer Cancellation Finished Interrupt Enable (address 0xE4)

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

2.3.45 Tx Event FIFO Configuration (TXEFC)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xF0	res		EFWM[5:0]						res		EFS[5:0]					
	R-0x0		RP-0x0						R-0x0		RP-0x0					

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EFSA[15:2]														res	
	RP-0x0														R-0x0	

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

Table 46 Tx Event FIFO Configuration (address 0xF0)

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).

2.3.46 Tx Event FIFO Status (TXEFS)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xF4	res						TEFL	EFF	res			EFPI[4:0]				
	R-0x0						R-0	R-0	R-0x0			R-0x0				
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	res			EFGI[4:0]					res		EFFL[5:0]					
	R-0x0			R-0x0					R-0x0		R-0x0					

R = Read; -n = value after reset

Table 47 Tx Event FIFO Status (address 0xF4)

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.

2.3.47 Tx Event FIFO Acknowledge (TXEFA)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0xF8	res															
	R-0x0															
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	res											EFAI[4:0]				
	R-0x0											RW-0x0				

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

Table 48 Tx Event FIFO Acknowledge (address 0xF8)

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 Event FIFO Fill Level **TXEFS.EFFL**.

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_CAN module.

Note: In case the Message RAM is equipped with parity or ECC functionality, it is recommended to initialize the Message RAM after hardware reset by writing e.g. 0x00000000 to each Message RAM word to create valid parity/ECC checksums. This avoids that reading from uninitialized Message RAM sections will activate interrupt IR.BEC (Bit Error Corrected) or IR.BEU (Bit Error Uncorrected).

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_CAN module can be configured to allocate up to 4352 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**.

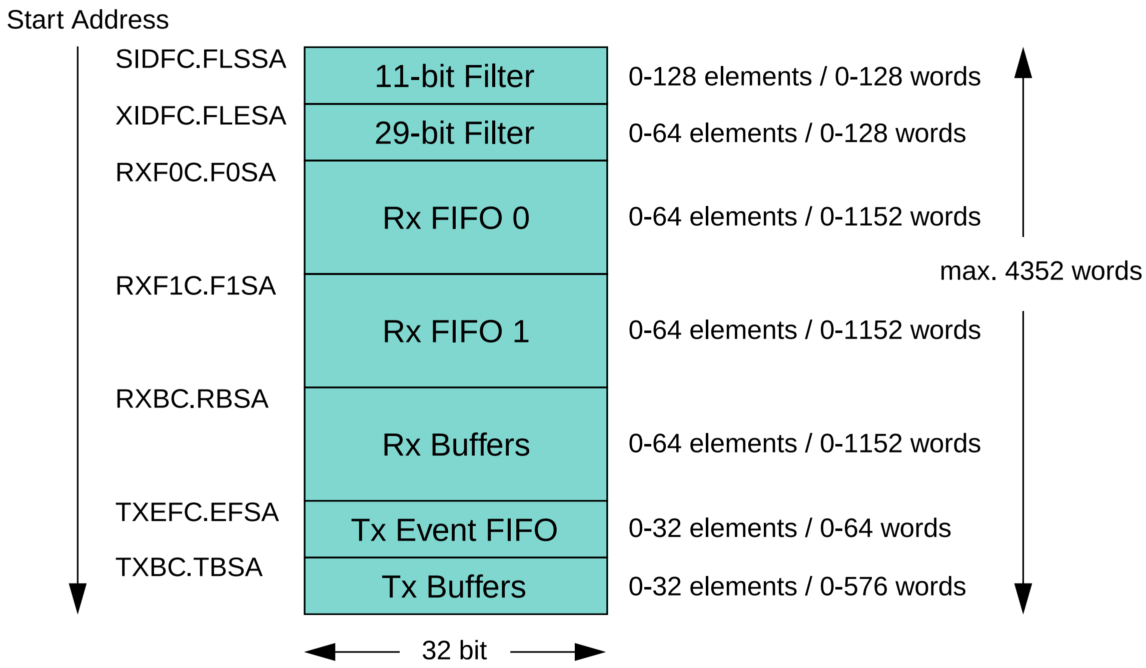


Figure 2 Message RAM Configuration

When the M_CAN 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_CAN 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 49 below. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register RXESC.

<table>
<tr><th></th><th colspan="4">31 … 24</th><th colspan="4">23 … 16</th><th>15 … 8</th><th>7 … 0</th></tr>
<tr><td>R0</td><td>ESI</td><td>XTD</td><td>RTR</td><td colspan="7">ID[28:0]</td></tr>
<tr><td>R1</td><td>ANMF</td><td colspan="3">FIDX[6:0]</td><td>res</td><td>FDF</td><td>BRS</td><td>DLC[3:0]</td><td colspan="2">RXTS[15:0]</td></tr>
<tr><td>R2</td><td colspan="4">DB3[7:0]</td><td colspan="4">DB2[7:0]</td><td>DB1[7:0]</td><td>DB0[7:0]</td></tr>
<tr><td>R3</td><td colspan="4">DB7[7:0]</td><td colspan="4">DB6[7:0]</td><td>DB5[7:0]</td><td>DB4[7:0]</td></tr>
<tr><td>...</td><td colspan="4">...</td><td colspan="4">...</td><td>...</td><td>...</td></tr>
<tr><td>Rn</td><td colspan="4">DBm[7:0]</td><td colspan="4">DBm-1[7:0]</td><td>DBm-2[7:0]</td><td>DBm-3[7:0]</td></tr>
</table>

Table 49 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 CAN FD frames (FDF = 1'), the dominant RRS (Remote Request Substitution) bit replaces bit RTR (Remote Transmission Request).

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 FDF: FD Format

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 ..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.

<table>
<tr><th></th><th colspan="4">31 … 24</th><th colspan="5">23 … 16</th><th>15 … 8</th><th>7 … 0</th></tr>
<tr><td>T0</td><td>ESI</td><td>XTD</td><td>RTR</td><td colspan="8">ID[28:0]</td></tr>
<tr><td>T1</td><td colspan="4">MM[7:0]</td><td>EFC</td><td>res</td><td>FDF</td><td>BRS</td><td>DLC[3:0]</td><td colspan="2">res</td></tr>
<tr><td>T2</td><td colspan="4">DB3[7:0]</td><td colspan="5">DB2[7:0]</td><td>DB1[7:0]</td><td>DB0[7:0]</td></tr>
<tr><td>T3</td><td colspan="4">DB7[7:0]</td><td colspan="5">DB6[7:0]</td><td>DB5[7:0]</td><td>DB4[7:0]</td></tr>
<tr><td>...</td><td colspan="4">...</td><td colspan="5">...</td><td>...</td><td>...</td></tr>
<tr><td>Tn</td><td colspan="4">DBm[7:0]</td><td colspan="5">DBm-1[7:0]</td><td>DBm-2[7:0]</td><td>DBm-3[7:0]</td></tr>
</table>

Table 50 Tx Buffer Element

T0 Bit 31 ESI: Error State Indicator

0= ESI bit in CAN FD format depends only on error passive flag

1= ESI bit in CAN FD format transmitted recessive

Note: The ESI bit of the transmit buffer is or'ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive

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_CAN transmits a remote frame according to ISO 11898-1:2015, even if CCCR.FDOE 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]**.

BOSCH

T1 Bits 31:24 MM[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 Bit 21 FDF: FD Format

0= Frame transmitted in Classic CAN format
1= Frame transmitted in CAN FD format

T1 Bit 20 BRS: Bit Rat Switching

0= CAN FD frames transmitted without bit rate switching
1= CAN FD frames transmitted with bit rate switching

Note: Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled CCCR.FDOE = 1'. Bit BRS is only evaluated when in addition CCCR.BRSE = '1'.

T1 Bits 19:16 DLC[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**.

<table>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>24 | 23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>16 | 15</td><td>8 | 7</td><td>0</td></tr>
<tr><td>E0</td><td>ESI</td><td>XTD</td><td>RTR</td><td colspan="9">ID[28:0]</td></tr>
<tr><td>E1</td><td colspan="4">MM[7:0]</td><td colspan="2">ET[1:0]</td><td>FDF</td><td>BRS</td><td colspan="2">DLC[3:0]</td><td colspan="2">TXTS[15:0]</td></tr>
</table>

Table 51 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 FDF: FD Format

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

E1 Bit 20 BRS: Bit Rate Switch

0= Frame transmitted without bit rate switching
1= Frame transmitted 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 52 Standard Message ID Filter Element

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

00= Range filter from **SFID1** to **SFID2** (**SFID2** ≥ **SFID**1)

01= Dual ID filter for **SFID1** or **SFID2**

10= Classic filter: **SFID1** = filter, **SFID2** = mask

11= Filter element disabled

Note: With SFT = “11” the filter element is disabled and the acceptance filtering continues (same behaviour as with SFEC = “000”)

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_can_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_can_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 53 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, *Extended Message ID Filtering*) is used.

F1 Bits 31:30 **EFT[1:0]:** Extended Filter Type

00= Range filter from **EFID1** to **EFID2** (**EFID2** ≥ **EFID1**)
01= Dual ID filter for **EFID1** or **EFID2**
10= Classic filter: **EFID1** = filter, **EFID2** = mask
11= Range filter from **EFID1** to **EFID2** (**EFID2** ≥ **EFID1**), 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_can_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_can_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.

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_can_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 (≡ *Bus_Idle*) before it can take part in bus activities and start the message transfer.

Access to the M_CAN 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

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’.

Note: In case the Message RAM is equipped with parity or ECC functionality, it is recommended to initialize the Message RAM after hardware reset by writing e.g. 0x00000000 to each Message RAM word to create valid parity/ECC checksums. This avoids that reading from uninitialized Message RAM sections will activate interrupt IR.BEC (Bit Error Corrected) or IR.BEU (Bit Error Uncorrected).

3.1.2 Normal Operation

Once the M_CAN is initialized and **CCCR.INIT** is reset to *zero*, the M_CAN 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 transmission, first the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as **FDF** bit. **FDF** = *recessive* signifies a CAN FD frame, **FDF** = *dominant* signifies a Classic CAN frame. In a CAN FD frame, the two bits following **FDF**, **res** and **BRS**, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by **res** = *dominant* and **BRS** = *recessive*. The coding of **res** = *recessive* is reserved for future expansion of the protocol. In case the M_CAN receives a frame with **FDF** = *recessive* and **res** = *recessive*, it will signal a Protocol Exception Event by setting bit **PSR.PXE**. When Protocol Exception Handling is enabled (**CCCR.PXHD** = ‘0’), this causes the operation state to change from Receiver (**PSR.ACT** = “10”) to Integrating (**PSR.ACT** = “00”) at the next sample point. In case Protocol Exception Handling is disabled (**CCCR.PXHD** = ‘1’), the M_CAN will treat a *recessive* **res** bit as an form error and will respond with an error frame.

CAN FD operation is enabled by programming **CCCR.FDOE**. In case **CCCR.FDOE** = ‘1’, transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via bit **FDF** in the respective Tx Buffer element. With **CCCR.FDOE** = ‘0’, received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit **FDF** of a Tx Buffer element is set. **CCCR.FDOE** and **CCCR.BRSE** can only be changed while **CCCR.INIT** and **CCCR.CCE** are both set.

With **CCCR.FDOE** = ‘0’, the setting of bits **FDF** and **BRS** is ignored and frames are transmitted in Classic CAN format. With **CCCR.FDOE** = ‘1’ and **CCCR.BRSE** = ‘0’, only bit FDF of a Tx Buffer element is evaluated. With **CCCR.FDOE** = ‘1’ and **CCCR.BRSE** = ‘1’, transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits **FDF** and **BRS** set are transmitted in CAN FD format with bit rate switching.

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 Classic CAN messages 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.
- Wake-up messages in CAN Partial Networking have to be transmitted in Classic CAN format.
- 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 Classic CAN communication.

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 54 below.

DLC	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Number of Data Bytes	12	16	20	24	32	48	64

Table 54 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 nominal CAN bit timing is used as defined by the Nominal Bit Timing & Prescaler Register **NBTP**. In the following CAN FD data phase, the data phase bit timing is used as defined by the Data Bit Timing & Prescaler Register **DBTP**. The bit timing is switched back from the data phase 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_can_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 and CAN FD with bit rate switching, 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 Transmitter 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_can_tx** the M_CAN receives the transmitted data from its local CAN transceiver via pin **m_can_rx**. The received data is delayed by the transmitter 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 transmitter delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transmitter delay.

3.1.4.1 Description

The M_CAN's protocol unit has implemented a delay compensation mechanism to compensate the transmitter delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in ISO 11898-1:2015. It is enabled by setting bit **DBTP.TDC**.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the M_CAN's transmit output **m_can_tx** through the transceiver to the receive input **m_can_rx** plus the transmitter delay compensation offset as configured by **TDCR.TDCO**. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (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 mtq.

PSR.TDCV shows the actual transmitter delay compensation value. **PSR.TDCV** is cleared when **CCCR.INIT** is set and is updated at each transmission of an FD frame while **DBTP.TDC** is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the M_CAN:

- The sum of the measured delay from **m_can_tx** to **m_can_rx** and the configured transmitter delay compensation offset **TDCR.TDCO** has to be less than 6 bit times in the data phase.
- The sum of the measured delay from **m_can_tx** to **m_can_rx** and the configured transmitter delay compensation offset **TDCR.TDCO** has to be less or equal 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the SSPs

3.1.4.2 Transmitter Delay Compensation Measurement

If transmitter delay compensation is enabled by programming **DBTP.TDC** = ‘1’, the measurement is started within each transmitted CAN FD frame at the falling edge of bit **FDF** to bit **res**. The measurement is stopped when this edge is seen at the receive input **m_can_rx** of the transmitter. The resolution of this measurement is one mtq.

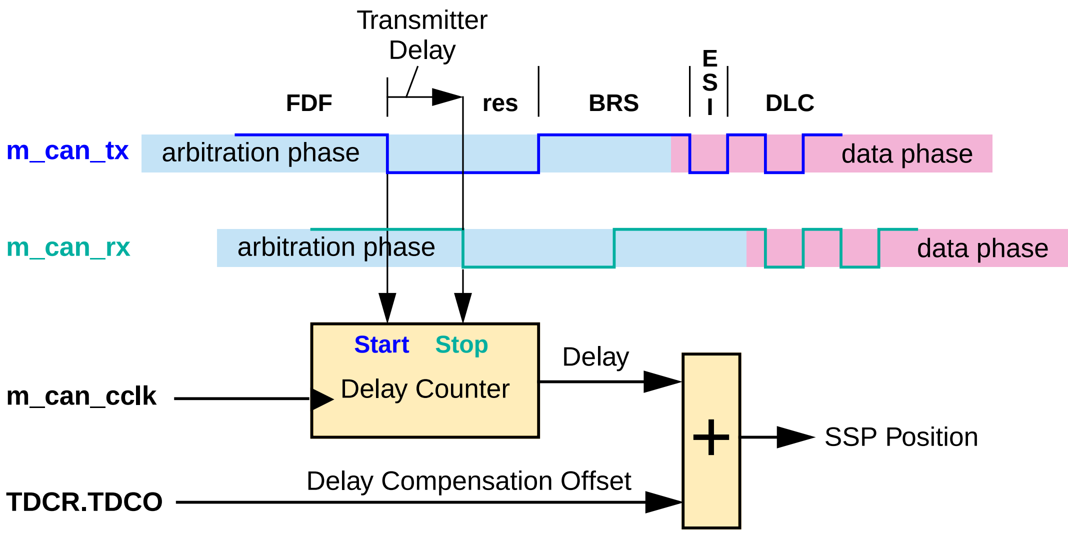


Figure 3 Transmitter Delay Measurement

To avoid that a dominant glitch inside the received **FDF** bit ends the delay compensation measurement before the falling edge of the received **res** bit, resulting in a to early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming **TDCR.TDCF**. This defines a minimum value for the SSP position. Dominant edges on **m_can_rx**, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least **TDCR.TDCF** AND **m_can_rx** is low.

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 (**ECR.REC**, **ECR.TEC**) are frozen while Error Logging (**ECR.CEL**) is active. The Host can set the M_CAN 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.

If the M_CAN is connected to a Clock Calibration on CAN unit, **CCCR.ASM** is controlled by input **m_can_cok**. In case **m_can_cok** switches to ‘0’, bit **CCCR.ASM** is set. When **m_can_cok** switches back to ‘1’, bit **CCCR.ASM** returns to the previously written value. When there is no Clock Calibration on CAN unit connected input **m_can_cok** is hardwired to ‘1’.

Note: The Restricted Operation Mode must not be combined with the Loop Back Mode (internal or external).

3.1.6 Bus Monitoring Mode

The M_CAN is set in Bus Monitoring Mode by programming **CCCR.MON** to *one.* In Bus Monitoring Mode (see ISO 11898-1:2015, 10.14 Bus monitoring), the M_CAN 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_CAN is required to send a *dominant* bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_CAN 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_can_tx** and **m_can_rx** to the M_CAN in Bus Monitoring Mode.

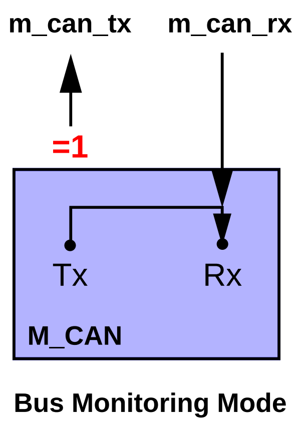


Figure 4 Pin Control in Bus Monitoring Mode

3.1.7 Disabled Automatic Retransmission

According to the CAN Specification (see ISO 11898-1:2015, 8.3.4 Recovery Management), the M_CAN 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 11898-1:2015, 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_CAN can be set into power down mode controlled by input signal **m_can_clkstop_req** or via CC Control Register **CCCR.CSR**. As long as the clock stop request signal **m_can_clkstop_req** is active, bit **CCCR.CSR** is read as *one*.

When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the M_CAN sets then **CCCR.INIT** to *one* to prevent any further CAN transfers. Now the M_CAN acknowledges that it is ready for power down by setting output signal **m_can_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_can_hclk** and **m_can_cclk** may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting signal **m_can_clkstop_req** resp. CC Control Register flag **CCCR.CSR**. The M_CAN will acknowledge this by resetting output signal **m_can_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_can_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_CAN's bit timing and it can drive constant dominant or recessive values. The actual value at pin **m_can_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_can_tx**. This applies also when reading input pin **m_can_rx** via **TEST.RX**.

Note: Test modes should be used for production tests or self test only. The software control for pin m_can_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_CAN can be set in External Loop Back Mode by programming **TEST.LBCK** to *one*. In Loop Back Mode, the M_CAN 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_can_tx** and **m_can_rx** to the M_CAN in External Loop Back Mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode. In this mode the M_CAN performs an internal feedback from its Tx output to its Rx input. The actual value of the **m_can_rx** input pin is disregarded by the M_CAN. The transmitted messages can be monitored at the **m_can_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_CAN can be tested without affecting a running CAN system connected to the pins **m_can_tx** and **m_can_rx**. In this mode pin **m_can_rx** is disconnected from the M_CAN and pin **m_can_tx** is held *recessive*. Figure 5 shows the connection of **m_can_tx** and **m_can_rx** to the M_CAN in case of Internal Loop Back Mode.

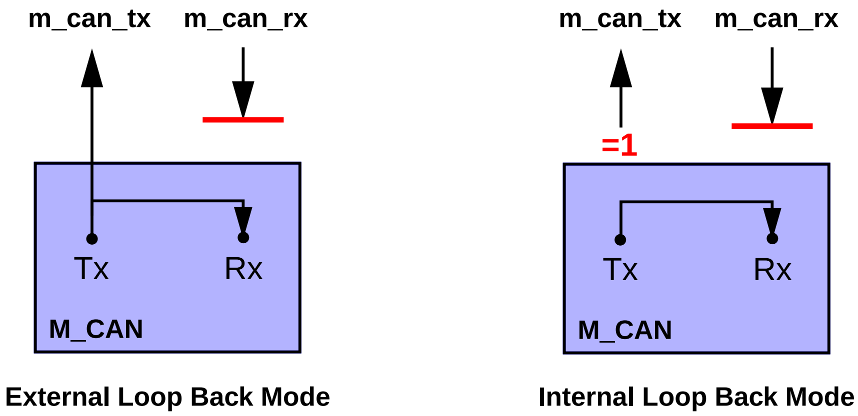


Figure 5 Pin Control in Loop Back Modes

3.2 Timestamp Generation

For timestamp generation the M_CAN 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.TSC**. 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_CAN 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_CAN 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 bit 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_CAN 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.DLEC**.

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.DLEC**. 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.

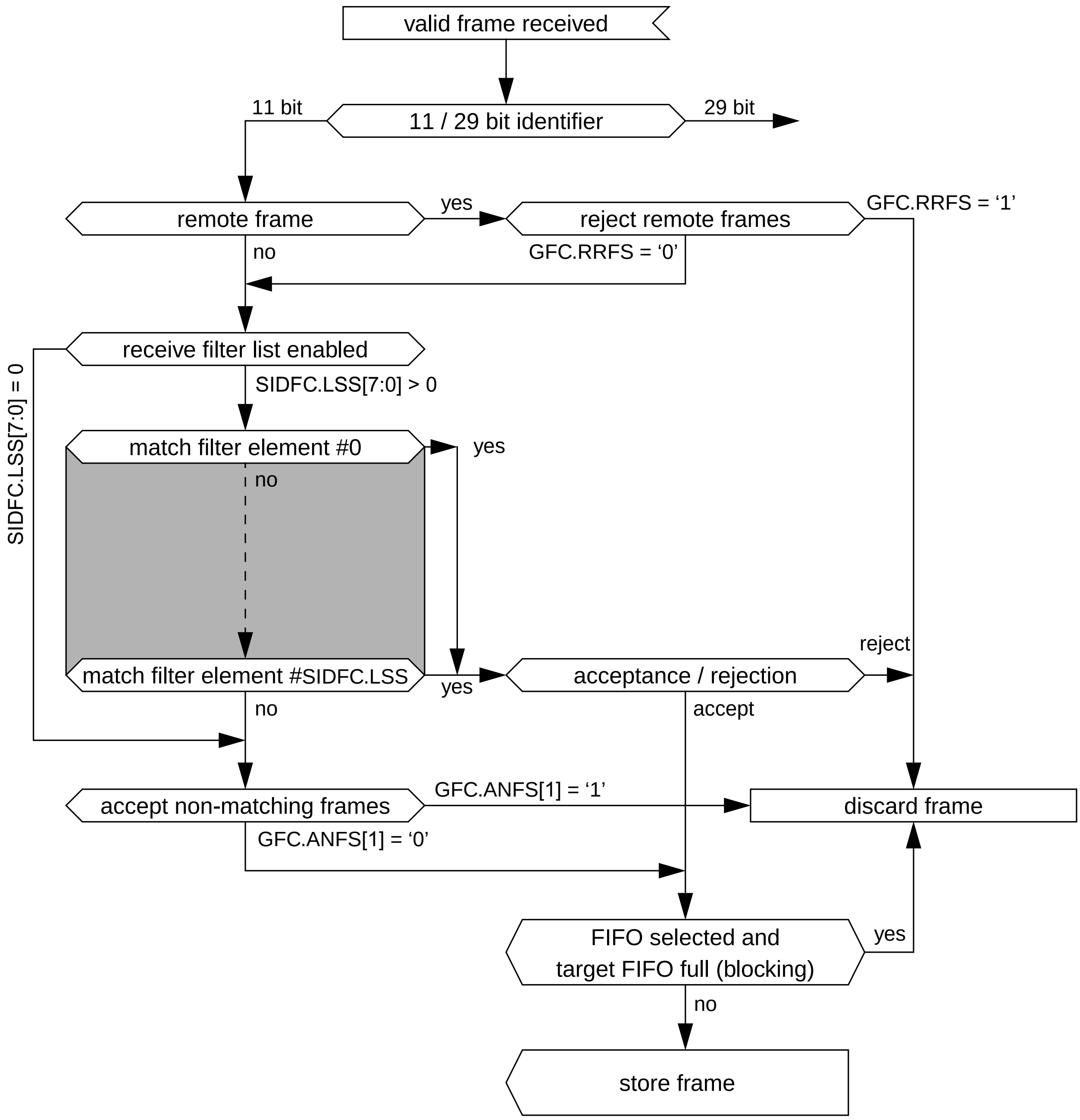


Figure 6 Standard Message ID Filter Path

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.

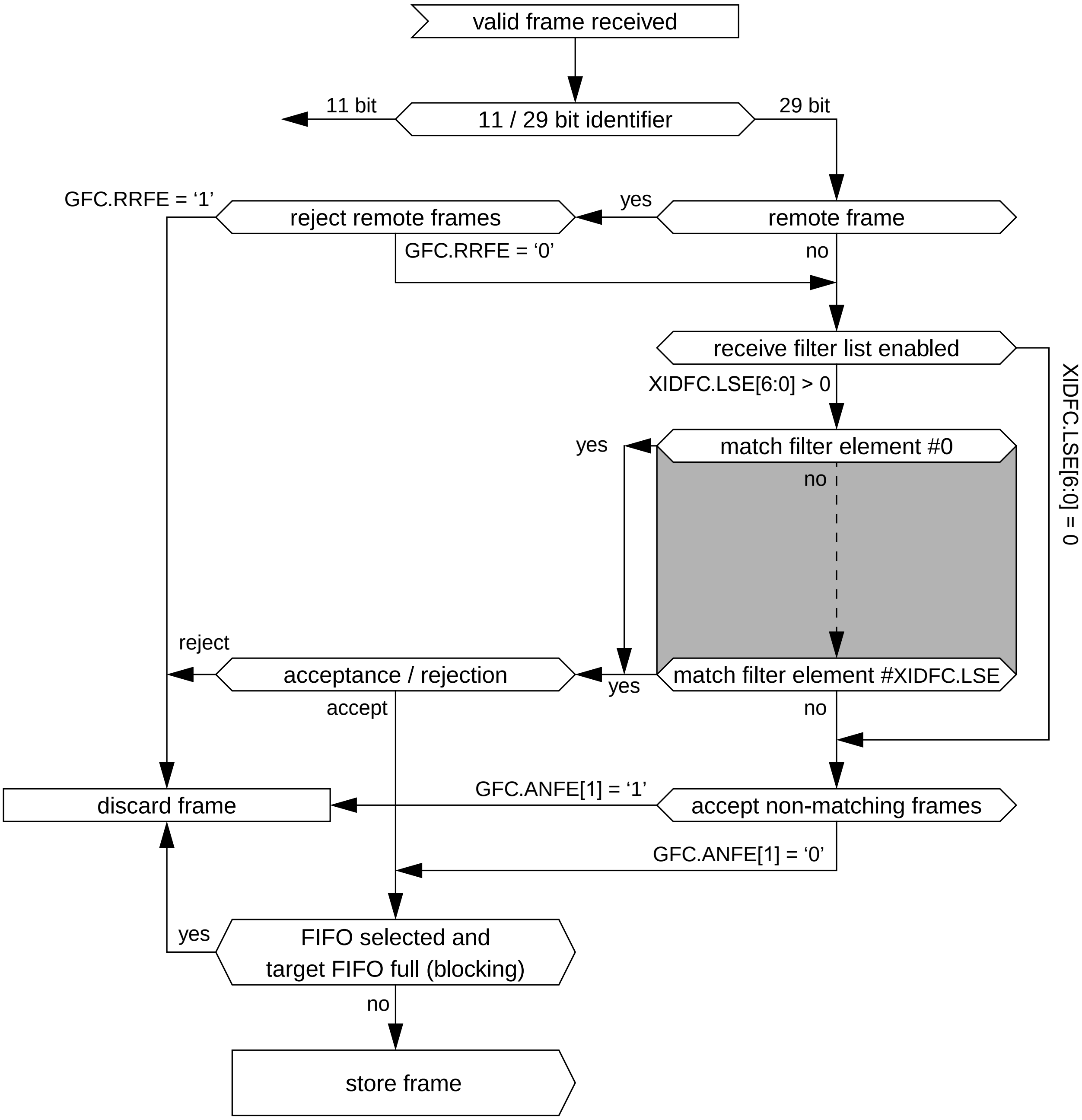


Figure 7 Extended Message ID Filter Path

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.

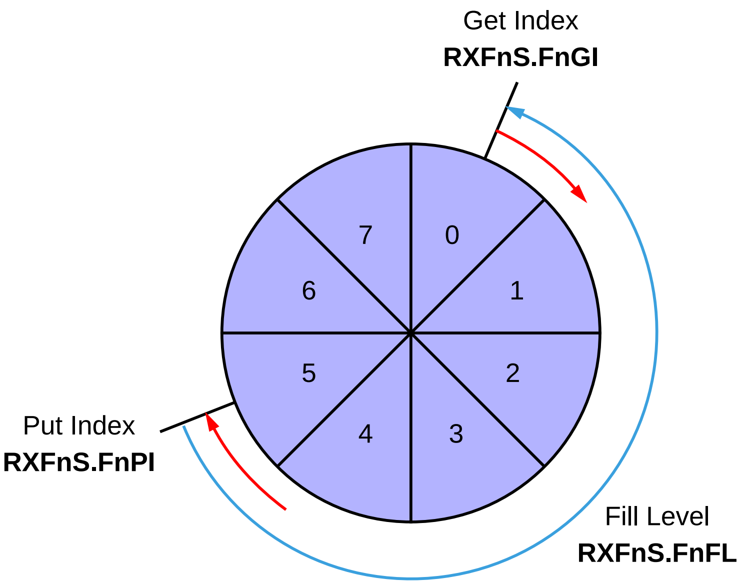


Figure 8 Rx FIFO Status

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**.

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO 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 55 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.

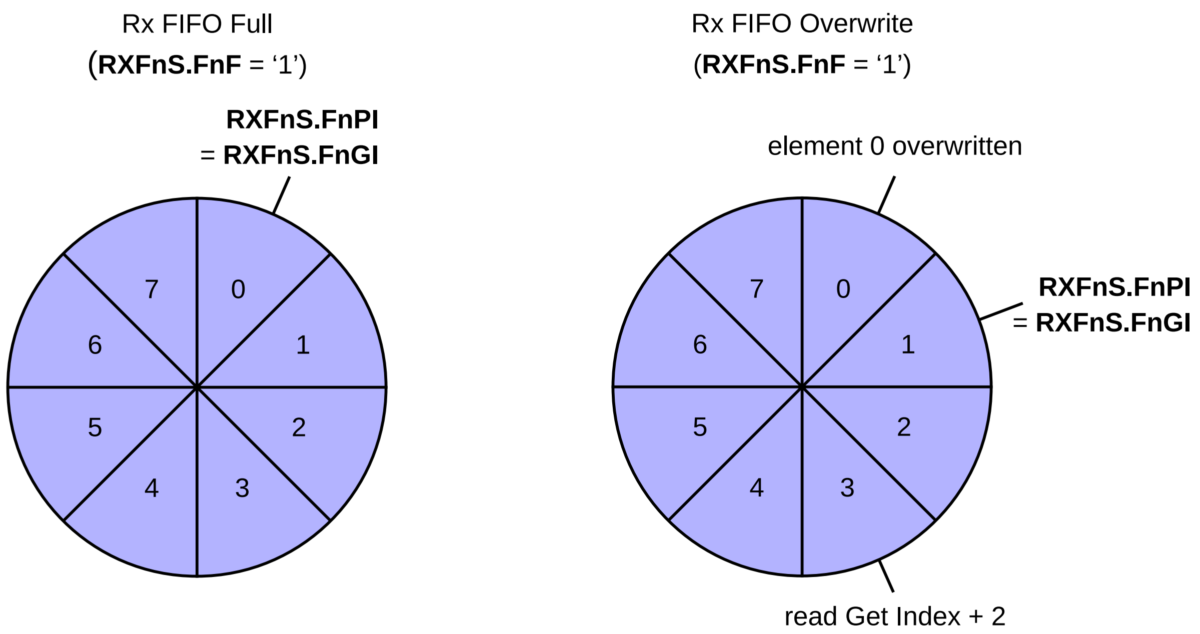


Figure 9 Rx FIFO Overflow Handling

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_CAN 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.

Filter Element	**SFID1[10:0] EFID1[28:0]**	**SFID2[10:9] EFID2[10:9]**	**SFID2[5:0] EFID2[5:0]**
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

Table 56 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_CAN 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_can_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_CAN while **m_can_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_can_dma_ack**. This resets **m_can_dma_req**. Now the M_CAN 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**.

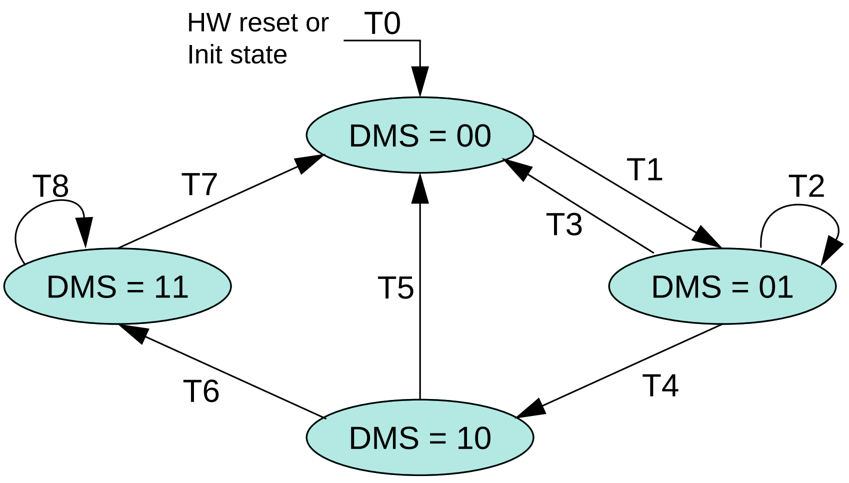
Filter Element	**SFID1[10:0] EFID1[28:0]**	**SFID2[10:9] EFID2[10:9]**	**SFID2[5:0] EFID2[5:0]**
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

Table 57 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_can_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 CAN mode for transmission (Classic CAN or CAN FD) can be configured separately for each Tx Buffer element. The Tx Buffer element is described in Section 2.4.3. Table 58 below describes the possible configurations for frame transmission.

CCCR		Tx Buffer Element		Frame Transmission
BRSE	FDOE	FDF	BRS	
ignored	0	ignored	ignored	Classic CAN
0	1	0	ignored	Classic CAN
0	1	1	ignored	FD without bit rate switching
1	1	0	ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Table 58 Possible Configurations for Frame Transmission

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_CAN 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 looses 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 59). 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 59 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_CAN 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 59). 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 59). 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.

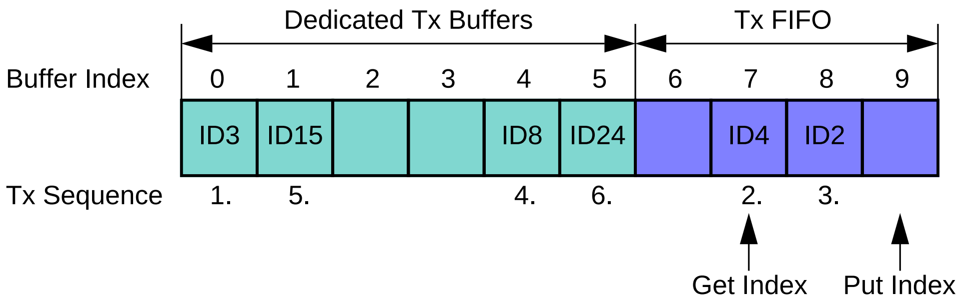


Figure 11 Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO

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.

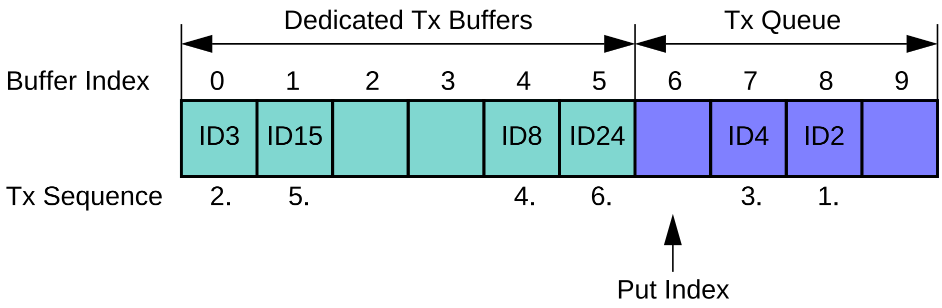


Figure 12 Example of mixed Configuration Dedicated Tx Buffers / Tx Queue

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_CAN 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_CAN has implemented a Tx Event FIFO. After the M_CAN 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.

The purpose of the Tx Event FIFO is to decouple handling transmit status information from transmit message handling i.e. a Tx Buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx Event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx Buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx Buffer before overwriting that Tx Buffer.

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 2.3.29, Section 2.3.33, and Section 2.3.47). 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_CAN'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_CAN does not check for erroneous values.

4. Appendix

4.1 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
0x00	RE14.10.20 11L[4:0]				STEP[4:0]				SUBSTEP [4:0]				YEAR[4:0]				MON[7:0]								DAY[7:0]								CREL rrrd_dddd
0x04	ETV[31:0]																																ENDN 8765_4321
0x08	CUST[31:0]																																CUST t.b.d.
0x0C									TDC			DBRP[4:0]								DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]				DBTP 0000_0A33
0x10																									RX	TX[1:0]		LBCK					TEST 0000_0000
0x14																	WDV[7:0]								WDC[7:0]								RWD 0000_0000
0x18																	NISO	TXP	EFBI	PXHD			BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT	CCCR 0000_0001
0x1C	NSJW[6:0]							NBRP[8:0]									NTSEG1[7:0]									NTSEG2[6:0]							NBTP 0000_0A33
0x20													TCP[3:0]																		TSS[1:0]		TSCC 0000_0000
0x24																	TSC[15:0]																TSCV 0000_0000
0x28	TOP[15:0]																													TOS[1:0]		ETOC	TOCC FFFF_0000

Table 60 M_CAN 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
0x2C																	TOC[15:0]																TOCV 0000_FFFF
0x40									CEL[7:0]								RP	REC[6:0]							TEC[7:0]								ECR 0000_0000
0x44										TDCV[6:0]								PXE	RFDF	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]			PSR 0000_0707
0x48																		TDCO[6:0]								TDCF[6:0]							TDCR 0000_0000
0x50			ARA	PED	PEA	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N	IR 0000_0000
0x54			ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE	IE 0000_0000
0x58			ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL	ILS 0000_0000
0x5C																															EINT1	EINT0	ILE 0000_0000
0x80																											ANFS[1:0]		ANFE[1:0]		RRFS	RRFE	GFC 0000_0000
0x84									LSS[7:0]								FLSSA[15:2]																SIDFC 0000_0000
0x88										LSE[6:0]							FLESA[15:2]																XIDFC 0000_0000
0x90				EIDM[28:0]																													XIDAM 1FFF_FFFF
0x94																	FLST	FIDX[6:0]							MSI[1:0]		BIDX[5:0]						HPMS 0000_0000

Table 60 M_CAN 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
0x98	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0	NDAT1 0000_0000
0x9C	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32	NDAT2 0000_0000
0xA0	F0OM	F0WM[6:0]								F0S[6:0]							F0SA[15:2]																RXF0C 0000_0000
0xA4							RF0L	F0F			F0PI[5:0]								F0GI[5:0]							F0FL[6:0]							RXF0S 0000_0000
0xA8																											F0AI[5:0]						RXF0A 0000_0000
0xAC																	RBSA[15:2]																RXBC 0000_0000
0xB0	F1OM	F1WM[6:0]								F1S[6:0]							F1SA[15:2]																RXF1C 0000_0000
0xB4	DMS[1:0]						RF1L	F1F			F1PI[5:0]								F1GI[5:0]							F1FL[6:0]							RXF1S 0000_0000
0xB8																											F1AI[5:0]						RXF1A 0000_0000
0xBC																						RBDS[2:0]				F1DS[2:0]				F0DS[2:0]			RXESC 0000_0000
0xC0		TFQM	TFQS[5:0]								NDTB[5:0]						TBSA[15:2]																TXBC 0000_0000
0xC4											TFQF	TFQPI[4:0]								TFGI[4:0]							TFFL[5:0]						TXFQS 0000_0000
0xC8																														TBDS[2:0]			TXESC 0000_0000

Table 60 M_CAN Register Overview

BOSCH

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
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
0xD0	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0	TXBAR 0000_0000
0xD4	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0	TXBCR 0000_0000
0xD8	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0	TXBTO 0000_0000
0xDC	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0	TXBCF 0000_0000
0xE0	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0	TXBTIE 0000_0000
0xE4	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0	TXBCIE 0000_0000
0xF0			EFWM[5:0]								EFS[5:0]						EFSA[15:2]																TXEFC 0000_0000
0xF4							TEFL	EFF				EFPI[4:0]								EFGI[4:0]							EFFL[5:0]						TXEFS 0000_0000
0xF8																												EFAI[4:0]					TXEFA 0000_0000

Table 60 M_CAN Register Overview

BOSCH

4.2 Module Interface

The M_CAN module's toplevel entity has the ports listed in the table below. More details on how to connect the M_CAN to a customer-specific design can be found in [2] and [3].

PORT	DIR	DOMAIN	DESCRIPTION
Clocks and Reset			
m_can_hclk	IN	HCLK	Host clock
m_can_cclk	IN	CCLK	CAN clock
m_can_reset	IN	ASYNC	module reset
Physical Layer Interface			
m_can_rx	IN	ASYNC	CAN receive input
m_can_tx	OUT	CCLK	CAN transmit output
Generic Slave Interface			
m_can_aei_sel	IN	HCLK	module select
m_can_aei_w1r0	IN	HCLK	module write
m_can_aei_byteen[3:0]	IN	HCLK	module byte enable
m_can_aei_addr[8:2]	IN	HCLK	module address bus
m_can_aei_wdata[31:0]	IN	HCLK	module data bus input
m_can_aei_ready	OUT	HCLK	module ready
m_can_aei_rdata[31:0]	OUT	HCLK	module data bus output
Generic Master Interface			
m_can_aeim_ready	IN	HCLK	memory ready
m_can_aeim_rdata[31:0]	IN	HCLK	memory data bus input
m_can_aeim_berr[1:0]	IN	HCLK	Message RAM bit error
m_can_aeim_sel	OUT	HCLK	memory select
m_can_aeim_w1r0	OUT	HCLK	memory write
m_can_aeim_addr[15:2]	OUT	HCLK	memory address bus, 32-bit word address
m_can_aeim_wdata[31:0]	OUT	HCLK	memory data bus output
Miscellaneous			
m_can_ext_ts[15:0]	IN	HCLK	external timestamp vector
m_can_clkstop_req	IN	HCLK	clock stop request
m_can_scanmode	IN	ASYNC	scan mode enable
m_can_dis_mord	IN	HCLK	disable modification on read (**ECR.CEL**, **PSR.PXE**, **PSR.RFDF**, **PSR.RBRS**, **PSR.RESI**, **PSR.DLEC**, **PSR.LEC**)
m_can_int0	OUT	HCLK	interrupt 0
m_can_int1	OUT	HCLK	interrupt 1
m_can_clkstop_ack	OUT	HCLK	clock stop acknowledge
DMA Interface			
m_can_dma_ack	IN	HCLK	DMA acknowledge
m_can_dma_req	OUT	HCLK	DMA request
Extension Interface			
m_can_cok	IN	HCLK	calibration OK, has to be hardwired to '1' in case no Clock Calibration on CAN unit is connected
m_can_ir[31:0]	OUT	HCLK	Interrupt Register flags
m_can_txbrp[31:0]	OUT	HCLK	Tx Buffer Request Pending (**TXBRP**)

Table 61 M_CAN Module Interface

PORT	DIR	DOMAIN	DESCRIPTION
m_can_rxfd	OUT	CCLK	receive fast data
m_can_txfd	OUT	CCLK	transmit fast data
m_can_fe[2:0]	OUT	HCLK	filter events 0..2
m_can_cce	OUT	HCLK	Configuration Change Enable (**CCCR.CCE**)
m_can_spt	OUT	CCLK	sample point delayed by one m_can_cclk period
m_can_mrx	OUT	CCLK	message received
m_can_calf	OUT	CCLK	calibration field
m_can_aff	OUT	HCLK	acceptance filtering finished

Table 61 M_CAN Module Interface

Note: Signals m_can_cok, m_can_spt, m_can_mrx, m_can_calf, m_can_aff, and one of the filter event outputs m_can_fe are interfacing to an optional Clock Calibration on CAN unit. In case the M_CAN is used without Clock Calibration on CAN unit, input m_can_cok has to be hardwired to '1'.

List of Figures

List of Tables