



M_CAN

Controller Area Network

User's Manual

Revision 2.0.1

12.03.2012



Robert Bosch GmbH
Automotive Electronics

LEGAL NOTICE

© Copyright 2008-2012 by Robert Bosch GmbH and its licensors. All rights reserved.

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

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

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

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

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

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

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

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

MANDATORY LAW SHALL BE UNAFFECTED BY THE FOREGOING PARAGRAPHS.

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

SPECIFICATION REVISION HISTORY

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

TRACKING OF MAJOR CHANGES

TERMS AND ABBREVIATIONS

This document uses the following terms and abbreviations.

Term	Meaning
BRP	Baud Rate Prescaler
BSP	Bit Stream Processor
BTL	Bit Timing Logic
CAN	Controller Area Network
CAN FD	Controller Area Network with Flexible Data-rate
CRC	Cyclic Redundancy Check
DLC	Data Length Code
ECC	Error Correction Code
EML	Error Management Logic
FSM	Finite State Machine

CONVENTIONS

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

Arial bold	Names of bits and ports
<i>Arial italic</i>	States of bits and ports

REFERENCES

This document refers to the following documents:

Ref	Author(s)	Title
[1]	ISO	ISO 11898-1: CAN data link layer and physical signalling
[2]	AE/EIY	M_(TT)CAN System Integration Guide
[3]	AE/EIY	M_CAN Module Integration Guide
[4]	AE/EIY	CAN FD Protocol Specification

Table of contents

1	Overview	1
1.1	Features	1
1.2	Block Diagram	2
1.3	Dual Clock Sources	3
1.4	Dual Interrupt Lines	3
2	Programmer's Model	5
2.1	Hardware Reset Description	5
2.2	Register Map	5
2.3	Registers	7
2.3.1	Customer Register	7
2.3.2	Core Release Register (CREL)	7
2.3.3	Endian Register (ENDN)	8
2.3.4	Fast Bit Timing & Prescaler Register (FBTP)	8
2.3.5	Test Register (TEST)	9
2.3.6	RAM Watchdog (RWD)	10
2.3.7	CC Control Register (CCCR)	11
2.3.8	Bit Timing & Prescaler Register (BTP)	13
2.3.9	Timestamp Counter Configuration (TSCC)	14
2.3.10	Timestamp Counter Value (TSCV)	14
2.3.11	Timeout Counter Configuration (TOCC)	15
2.3.12	Timeout Counter Value (TOCV)	15
2.3.13	Error Counter Register (ECR)	16
2.3.14	Protocol Status Register (PSR)	17
2.3.15	Interrupt Register (IR)	19
2.3.16	Interrupt Enable (IE)	22
2.3.17	Interrupt Line Select (ILS)	24
2.3.18	Interrupt Line Enable (ILE)	25
2.3.19	Global Filter Configuration (GFC)	26
2.3.20	Standard ID Filter Configuration (SIDFC)	27
2.3.21	Extended ID Filter Configuration (XIDFC)	27
2.3.22	Extended ID AND Mask (XIDAM)	28
2.3.23	High Priority Message Status (HPMS)	28
2.3.24	New Data 1 (NDAT1)	29
2.3.25	New Data 2 (NDAT2)	29
2.3.26	Rx FIFO 0 Configuration (RXF0C)	30
2.3.27	Rx FIFO 0 Status (RXF0S)	31
2.3.28	Rx FIFO 0 Acknowledge (RXF0A)	32
2.3.29	Rx Buffer Configuration (RXBC)	32
2.3.30	Rx FIFO 1 Configuration (RXF1C)	33
2.3.31	Rx FIFO 1 Status (RXF1S)	33
2.3.32	Rx FIFO 1 Acknowledge (RXF1A)	34
2.3.33	Tx Buffer Configuration (TXBC)	35
2.3.34	Tx FIFO/Queue Status (TXFQS)	36
2.3.35	Tx Buffer Request Pending (TXBRP)	37
2.3.36	Tx Buffer Add Request (TXBAR)	38
2.3.37	Tx Buffer Cancellation Request (TXBCR)	38
2.3.38	Tx Buffer Transmission Occurred (TXBTO)	39
2.3.39	Tx Buffer Cancellation Finished (TXBCF)	39
2.3.40	Tx Buffer Transmission Interrupt Enable (TXBTIE)	40
2.3.41	Tx Buffer Cancellation Finished Interrupt Enable (TXBCIE)	40
2.3.42	Tx Event FIFO Configuration (TXEFC)	41
2.3.43	Tx Event FIFO Status (TXEFS)	42
2.3.44	Tx Event FIFO Acknowledge (TXEFA)	42

2.4	Message RAM	43
2.4.1	Message RAM Configuration	43
2.4.2	Rx Buffer and FIFO Element	44
2.4.3	Tx Buffer Element	45
2.4.4	Tx Event FIFO Element	47
2.4.5	Standard Message ID Filter Element	48
2.4.6	Extended Message ID Filter Element	49
3	Functional Description	51
3.1	Operating Modes	51
3.1.1	Software Initialization	51
3.1.2	Normal Operation	52
3.1.3	CAN FD Operation	52
3.1.4	Transceiver Delay Compensation	53
3.1.5	Bus Monitoring Mode	54
3.1.6	Disabled Automatic Retransmission	54
3.1.7	Power Down (Sleep Mode)	55
3.1.8	Test Modes	55
3.2	Timestamp Generation	56
3.3	Timeout Counter	56
3.4	Rx Handling	57
3.4.1	Acceptance Filtering	57
3.4.2	Rx FIFOs	61
3.4.3	Dedicated Rx Buffers	61
3.4.4	Debug on CAN Support	62
3.5	Tx Handling	64
3.5.1	Dedicated Tx Buffers	64
3.5.2	Tx FIFO	64
3.5.3	Tx Queue	65
3.5.4	Mixed Dedicated Tx Buffers / Tx FIFO	65
3.5.5	Mixed Dedicated Tx Buffers / Tx Queue	66
3.5.6	Transmit Cancellation	66
3.5.7	Tx Event Handling	67
3.6	FIFO Acknowledge Handling	67
4	Appendix	69
4.1	Register Overview	69
4.2	Module Interface	73
4.3	Connection to external Message RAM	75
4.4	Interface to DMA Controller	75

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 the CAN protocol specification 2.0 part A,B and to CAN FD 1.0. 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 CAN protocol version 2.0 part A, B and ISO 11898-1
- CAN FD with max. 8 data bytes supported
- CAN Error Logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signalling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Multiple M_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

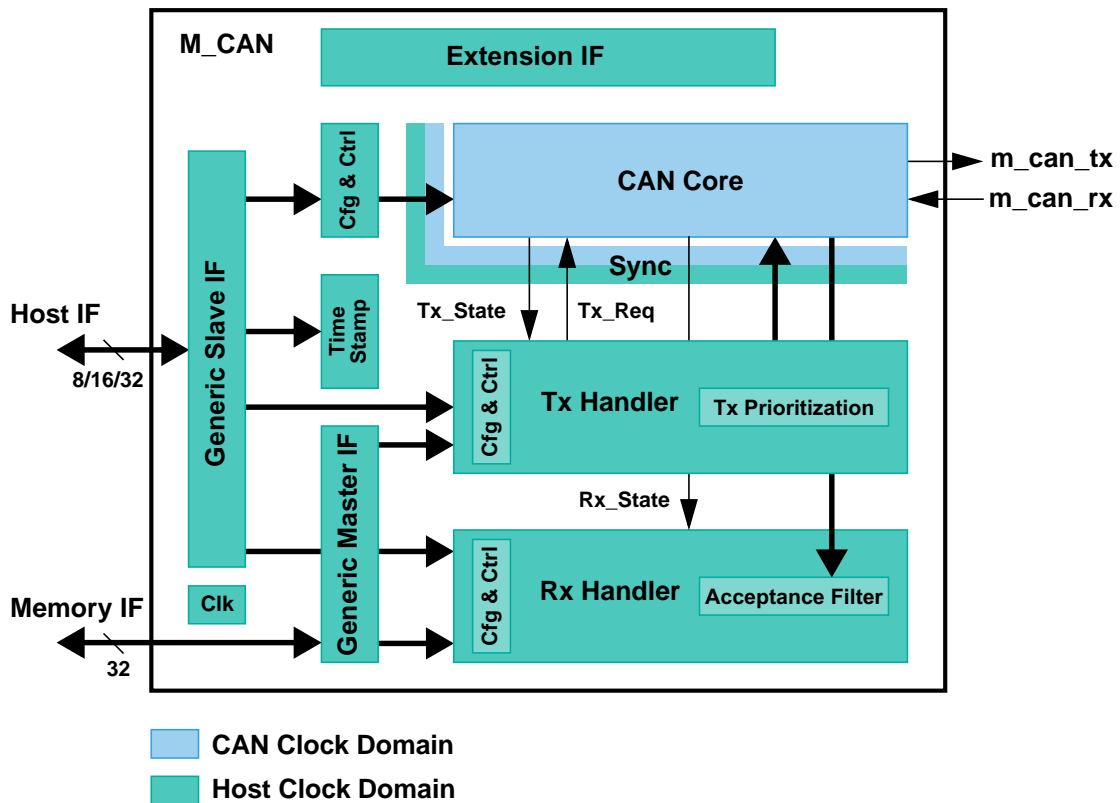


Figure 1 M_CAN Block Diagram

CAN Core

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

Clk

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

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 1K • 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**. The CAN Core should be programmed to have at least 8 clocks per bittime, this is e.g. 1 Mbaud @ **m_can_cclk** \geq 8 MHz. Even in case of a very high Host clock frequency, the clock frequency of the CAN Core needs not to be higher than 8 MHz.

Within the M_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 the spread spectrum clock has to be regarded.*

1.4 Dual Interrupt Lines

The module provides two interrupt lines. Interrupts can be routed either to **m_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	FBTP	Fast 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	RP
0x1C	BTP	Bit Timing & Prescaler Register	13	0000 0A33	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		<i>reserved (4)</i>		0000 0000	R
0x40	ECR	Error Counter Register	16	0000 0000	R
0x44	PSR	Protocol Status Register	17	0000 0707	RS
0x48-4C		<i>reserved (2)</i>		0000 0000	R
0x50	IR	Interrupt Register	19	0000 0000	RW
0x54	IE	Interrupt Enable	22	0000 0000	RW
0x58	ILS	Interrupt Line Select	24	0000 0000	RW
0x5C	ILE	Interrupt Line Enable	25	0000 0000	RW
0x60-7C		<i>reserved (8)</i>		0000 0000	R
0x80	GFC	Global Filter Configuration	26	0000 0000	RP
0x84	SIDFC	Standard ID Filter Configuration	27	0000 0000	RP

Table 1 M_CAN Register Map

ADDRESS	SYMBOL	NAME	PAGE	RESET	ACC
0x88	XIDFC	Extended ID Filter Configuration	27	0000 0000	RP
0x8C		<i>reserved (1)</i>		0000 0000	R
0x90	XIDAM	Extended ID AND Mask	28	1FFF FFFF	RP
0x94	HPMS	High Priority Message Status	28	0000 0000	R
0x98	NDAT1	New Data 1	29	0000 0000	RW
0x9C	NDAT2	New Data 2	29	0000 0000	RW
0xA0	RXF0C	Rx FIFO 0 Configuration	30	0000 0000	RP
0xA4	RXF0S	Rx FIFO 0 Status	31	0000 0000	R
0xA8	RXF0A	Rx FIFO 0 Acknowledge	32	0000 0000	RW
0xAC	RXBC	Rx Buffer Configuration	32	0000 0000	RW
0xB0	RXF1C	Rx FIFO 1 Configuration	33	0000 0000	RP
0xB4	RXF1S	Rx FIFO 1 Status	33	0000 0000	R
0xB8	RXF1A	Rx FIFO 1 Acknowledge	34	0000 0000	RW
0xBC		<i>reserved (1)</i>		0000 0000	R
0xC0	TXBC	Tx Buffer Configuration	35	0000 0000	RP
0xC4	TXFQS	Tx FIFO/Queue Status	36	0000 0000	R
0xC8		<i>reserved (1)</i>		0000 0000	R
0xCC	TXBRP	Tx Buffer Request Pending	37	0000 0000	R
0xD0	TXBAR	Tx Buffer Add Request	38	0000 0000	RW
0xD4	TXBCR	Tx Buffer Cancellation Request	38	0000 0000	RW
0xD8	TXBTO	Tx Buffer Transmission Occurred	39	0000 0000	R
0xDC	TXBCF	Tx Buffer Cancellation Finished	39	0000 0000	R
0xE0	TXBTIE	Tx Buffer Transmission Interrupt Enable	40	0000 0000	RW
0xE4	TXBCIE	Tx Buffer Cancellation Finished Interrupt Enable	40	0000 0000	RW
0xE8-EC		<i>reserved (2)</i>		0000 0000	R
0xF0	TXEFC	Tx Event FIFO Configuration	41	0000 0000	RP
0xF4	TXEFS	Tx Event FIFO Status	42	0000 0000	R
0xF8	TXEFA	Tx Event FIFO Acknowledge	42	0000 0000	RW
0xFC		<i>reserved (1)</i>		0000 0000	R

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

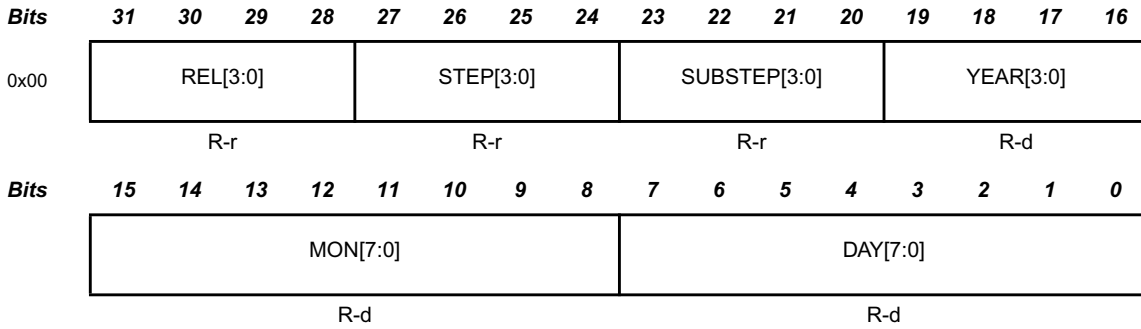
Table 1 M_CAN Register Map

2.3 Registers

2.3.1 Customer Register

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

2.3.2 Core Release Register (CREL)



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.

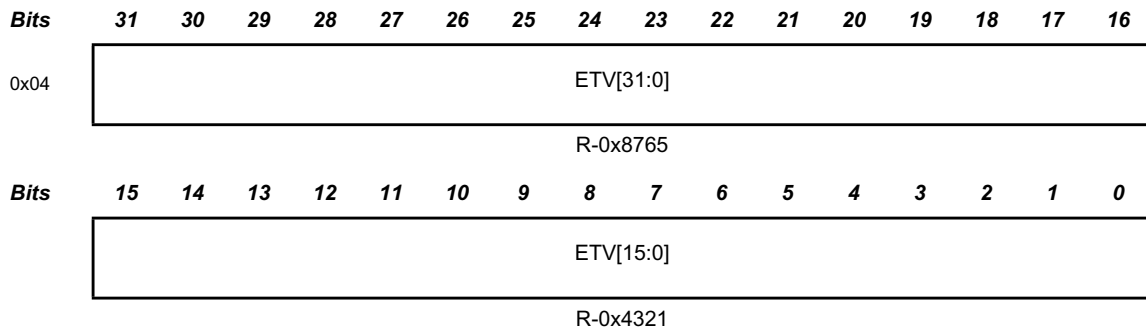
Bits 7:0 DAY[7:0]: Time Stamp Day

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)



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.

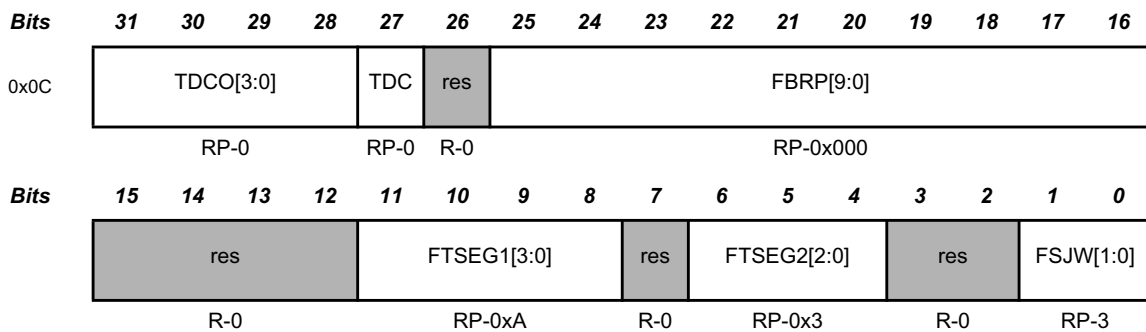
2.3.4 Fast Bit Timing & Prescaler Register (FBTP)

This register is only writable if bits **CCCR.CCE** and **CCCR.INIT** are set. The CAN bit time may be programmed in the range of 4 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to 1024 **m_can_cclk** periods. $t_q = (FBRP + 1) m_can_cclk$ period.

FTSEG1 is the sum of Prop_Seg and Phase_Seg1. **FTSEG2** is Phase_Seg2.

Therefore the length of the bit time is (programmed values) **[FTSEG1 + FTSEG2 + 3]** t_q or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q .

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



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

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

Bits 31:28 TDCO[3:0]: Transceiver Delay Compensation Offset

0x0-0xF 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 15 **m_can_cclk** periods.

Bit 27 TDC: Transceiver Delay Compensation

0= Transceiver Delay Compensation disabled

1= Transceiver Delay Compensation enabled

Bits 25:16 FBRP[9:0]: Fast Baud Rate Prescaler

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

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

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

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

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

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

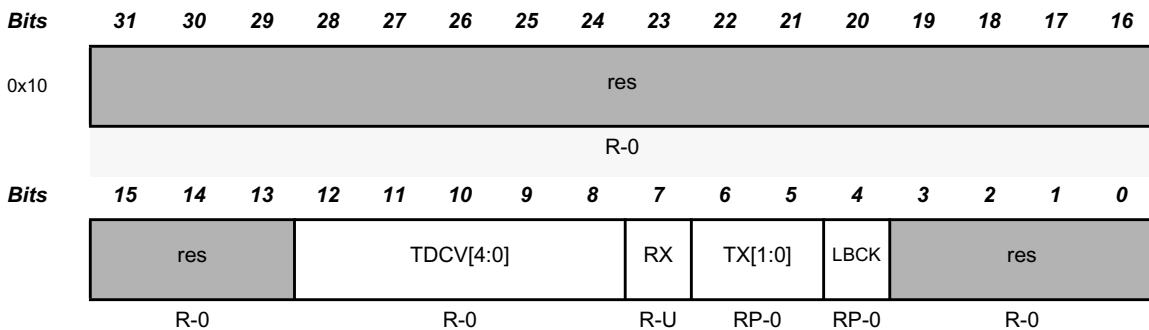
0x0-0x3 Valid values are 0 to 3. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a CAN clock (*m_can_cclk*) of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a fast bit rate of 500 kBit/s.

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.



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

Table 6 Test Register (address 0x10)

Bits 12:8 TDCV[4:0]: Transceiver Delay Compensation Value

0x00-0x1F Position of the secondary sample point, defined by the sum of the measured delay from *m_can_tx* to *m_can_rx* and **FBTP.TDCO**. Valid values are 0 to 31 *m_can_cclk* periods.

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')

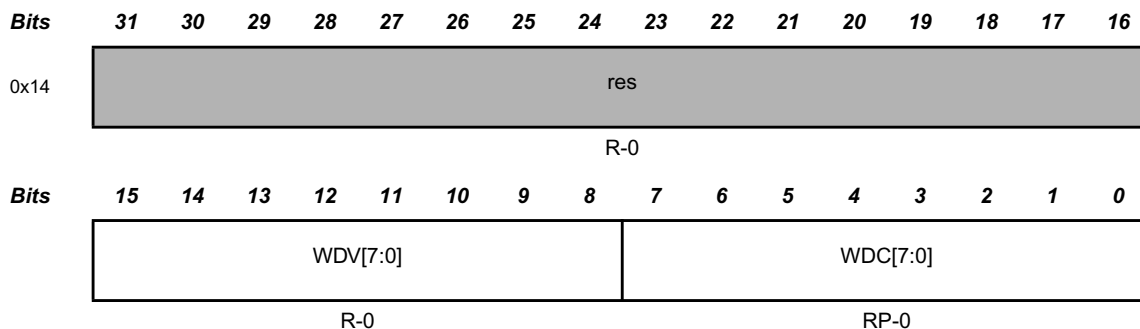
Bits 6:5 TX[1:0]: Control of Transmit Pin

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 3.1.8)

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**).



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

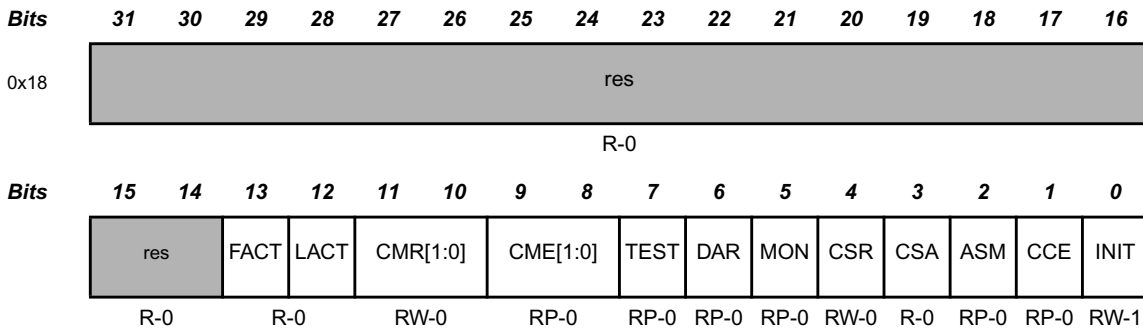
Table 7 RAM Watchdog (address 0x14)

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.



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

Table 8 CC Control Register (address 0x18)

Bit 13 FACT: Fast Frame Mode Active
 0= Format of transmitted frames defined by **LACT**
 1= Frames transmitted in CAN FD Fast format

Bit 12 LACT: Long Frame Mode Active
 0= Frames transmitted in Standard CAN format
 1= Frames transmitted in CAN FD Long format

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

A change of the CAN operation mode is requested by writing to this bit field. After change to the requested operation mode the bit field is reset to “00” and the status flags **FACT** and **LACT** are set accordingly. In case the requested CAN operation mode is not enabled, the value written to **CMR** is retained until it is overwritten by the next mode change request. Default is normal CAN operation.

- 00 unchanged
- 01 Long Frame Mode request
- 10 Long + Fast Frame Mode request
- 11 Normal CAN operation request

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

- 00 Normal CAN operation according to ISO11898-1
- 01 Long Frame Mode enabled
- 10 Long + Fast Frame Mode enabled
- 11 Long + Fast Frame Mode enabled

Note: In case **CME** = “01” transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With **CME** = “10”/“11” transmission and reception of long and fast CAN FD frames is enabled.

Bit 7 TEST: Test Mode Enable
 0= Normal operation, register **TEST** holds reset values
 1= Test Mode, write access to register **TEST** enabled

Bit 6 DAR: Disable Automatic Retransmission
 0= Automatic retransmission of messages not transmitted successfully enabled
 1= Automatic retransmission disabled

Bit 5 MON Bus Monitoring Mode

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

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

Bit 4 CSR: Clock Stop Request

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

Bit 3 CSA: Clock Stop Acknowledge

- 0= No clock stop acknowledged
- 1= M_CAN may be set in power down by stopping **m_can_hclk** and **m_can_cclk**

Bit 2 ASM Restricted Operation Mode

The Restricted Operation Mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame. In the optional Restricted Operation Mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. 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.

If the M_CAN is connected to a Clock Calibration on CAN unit, **ASM** is controlled by input **m_can_cok**. In case **m_can_cok** switches to '0', bit **ASM** is set. When **m_can_cok** switches back to '1', bit **ASM** returns to the previously written value. The state of **ASM** is the written value while input **m_can_cok** is at '1'. The input is hardwired to '1' when there is no Clock Calibration on CAN unit connected.

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

Bit 1 CCE: Configuration Change Enable

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

Bit 0 INIT: Initialization

- 0= Normal Operation
- 1= Initialization is started

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

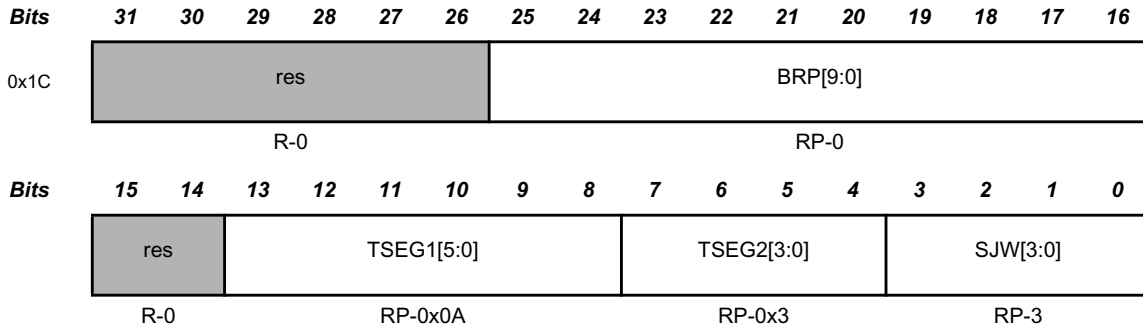
2.3.8 Bit Timing & Prescaler Register (BTP)

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

TSEG1 is the sum of Prop_Seg and Phase_Seg1. **TSEG2** is Phase_Seg2.

Therefore the length of the bit time is (programmed values) $[TSEG1 + TSEG2 + 3] t_q$
or (functional values) $[Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q$.

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



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

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

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

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

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

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

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

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

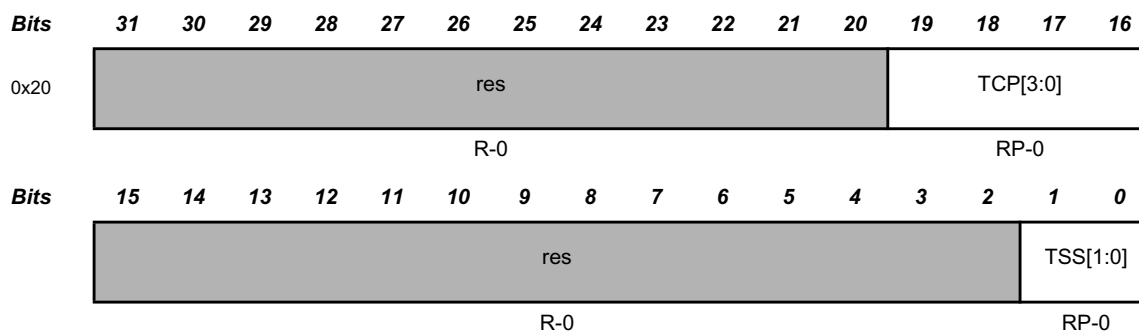
Bits 3:0 SJW[3:0]: (Re) Synchronization Jump Width

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

Note: With a CAN clock (**m_can_cclk**) of 8 MHz, the reset value of 0x0000A33 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*.



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

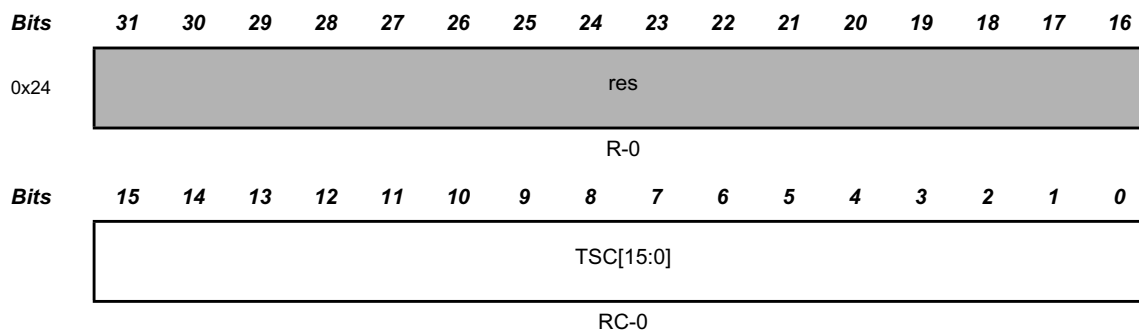
00= Timestamp counter value always 0x0000

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)



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

Table 11 Timestamp Counter Value (address 0x24)

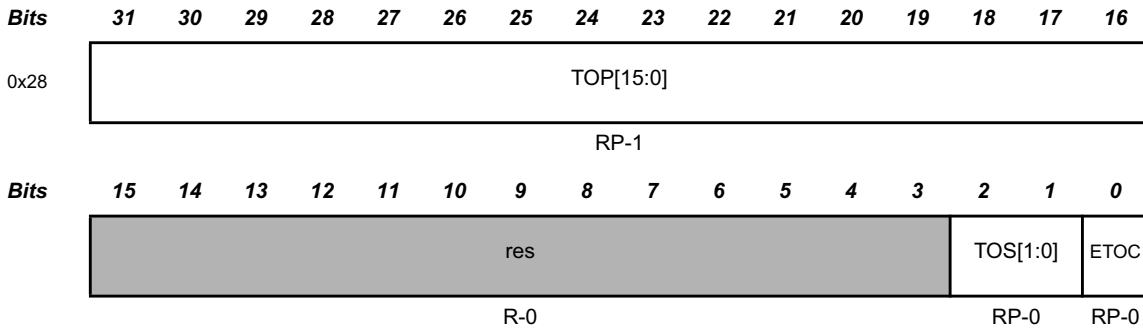
Bit 15:0 TSC[15:0]: Timestamp Counter

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*.



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.

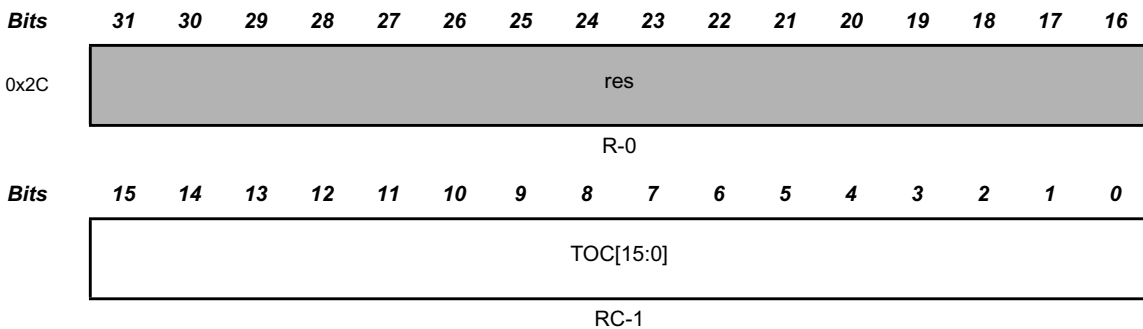
- 00= Continuous operation
- 01= Timeout controlled by Tx Event FIFO
- 10= Timeout controlled by Rx FIFO 0
- 11= Timeout controlled by Rx FIFO 1

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)



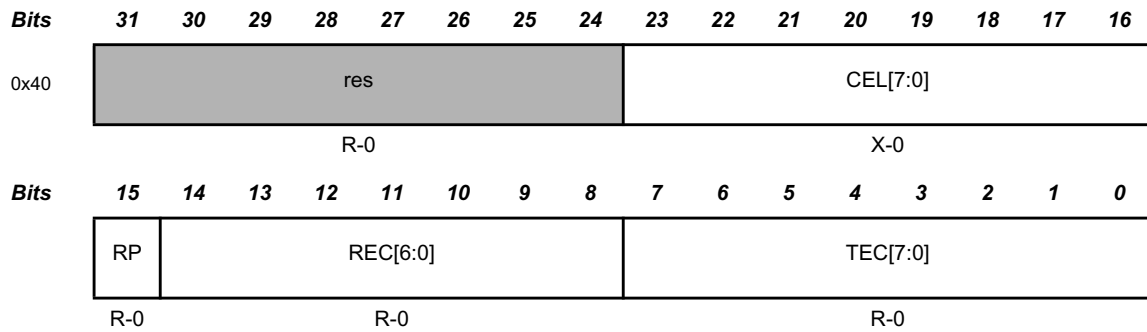
R = Read, C = Clear on write; -n = value after reset

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)



R = 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

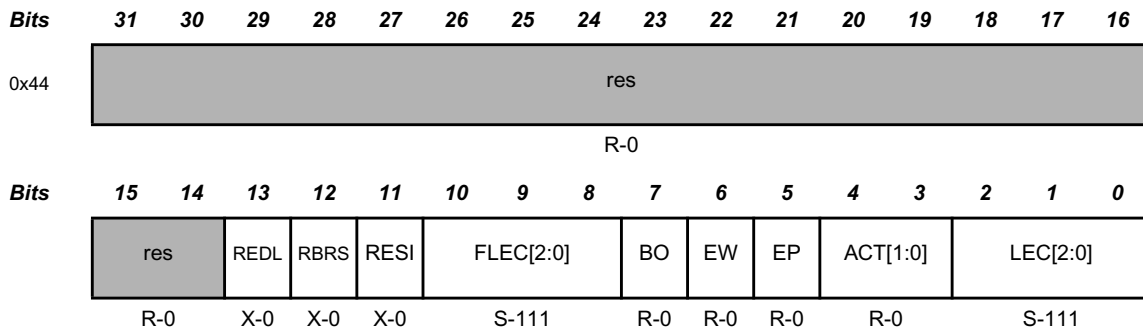
Actual state of the Receive Error Counter, values between 0 and 127

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)



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

Table 15 Protocol Status Register (address 0x44)

Bit 13 REDL: Received a CAN FD Message

This bit is set independent of acceptance filtering.

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

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

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

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

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

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

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

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

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

Bit 7 BO: Bus_Off Status

- 0= The M_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

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 FLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.*

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

2.3.15 Interrupt Register (IR)

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

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x50	STE	FOE	ACKE	BE	CRCE	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	UMD	TSW
	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
	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 16 Interrupt Register (address 0x50)

- Bit 31 STE:** Stuff Error
 0= No Stuff Error detected
 1= More than 5 equal bits in a sequence occurred
- Bit 30 FOE:** Format Error
 0= No Format Error detected
 1= A fixed format part of a received frame has the wrong format
- Bit 29 ACKE:** Acknowledge Error
 0= No Acknowledge Error detected
 1= A transmitted message was not acknowledged by another node
- Bit 28 BE:** Bit Error
 0= No Bit Error detected
 1= Device wanted to send a *rec / dom* level, but monitored bus level was *dom / rec*
- Bit 27 CRCE:** CRC Error
 0= No CRC Error detected
 1= Received CRC did not match the calculated CRC
- Bit 26 WDI:** Watchdog Interrupt
 0= No Message RAM Watchdog event occurred
 1= Message RAM Watchdog event due to missing READY
- Bit 25 BO:** Bus_Off Status
 0= Bus_Off status unchanged
 1= Bus_Off status changed
- Bit 24 EW:** Warning Status
 0= Error_Warning status unchanged
 1= Error_Warning status changed

- Bit 23 EP:** Error Passive
 0= Error_Passive status unchanged
 1= Error_Passive status changed
- Bit 22 ELO:** Error Logging Overflow
 0= CAN Error Logging Counter did not overflow
 1= Overflow of CAN Error Logging Counter occurred
- Bit 21 BEU:** Bit Error Uncorrected
 Message RAM bit error detected, uncorrected. Controlled by input signal **m_ttcan_aeim_berr[1]** generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets **CCCR.INIT** to '1'. This is done to avoid transmission of corrupted data.
 0= No bit error detected when reading from Message RAM
 1= Bit error detected, uncorrected (e.g. parity logic)
- Bit 20 BEC:** Bit Error Corrected
 Message RAM bit error detected and corrected. Controlled by input signal **m_ttcan_aeim_berr[0]** generated by an optional external parity / ECC logic attached to the Message RAM.
 0= No bit error detected when reading from Message RAM
 1= Bit error detected and corrected (e.g. ECC)
- Bit 19 DRX:** Message stored to Dedicated Rx Buffer
 The flag is set whenever a received message has been stored into a dedicated Rx Buffer.
 0= No Rx Buffer updated
 1= At least one received message stored into a Rx Buffer
- Bit 18 TOO:** Timeout Occurred
 0= No timeout
 1= Timeout reached
- Bit 17 UMD:** Unprocessed Message Discarded
 When a new message is received while the acceptance filtering process for the previously received message has not yet completed, this message is discarded.
 0= No unprocessed message discarded
 1= Unprocessed message discarded
- Bit 16 TSW:** Timestamp Wraparound
 0= No timestamp counter wrap-around
 1= Timestamp counter wrapped around
- Bit 15 TEFL:** Tx Event FIFO Element Lost
 0= No Tx Event FIFO element lost
 1= Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
- Bit 14 TEFF:** Tx Event FIFO Full
 0= Tx Event FIFO not full
 1= Tx Event FIFO full
- Bit 13 TEFW:** Tx Event FIFO Watermark Reached
 0= Tx Event FIFO fill level below watermark
 1= Tx Event FIFO fill level reached watermark

- Bit 12** **TEFN:** Tx Event FIFO New Entry
 0= Tx Event FIFO unchanged
 1= Tx Handler wrote Tx Event FIFO element
- Bit 11** **TFE:** Tx FIFO Empty
 0= Tx FIFO non-empty
 1= Tx FIFO empty
- Bit 10** **TCF:** Transmission Cancellation Finished
 0= No transmission cancellation finished
 1= Transmission cancellation finished
- Bit 9** **TC:** Transmission Completed
 0= No transmission completed
 1= Transmission completed
- Bit 8** **HPM:** High Priority Message
 0= No high priority message received
 1= High priority message received
- Bit 7** **RF1L:** Rx FIFO 1 Message Lost
 0= No Rx FIFO 1 message lost
 1= Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
- Bit 6** **RF1F:** Rx FIFO 1 Full
 0= Rx FIFO 1 not full
 1= Rx FIFO 1 full
- Bit 5** **RF1W:** Rx FIFO 1 Watermark Reached
 0= Rx FIFO 1 fill level below watermark
 1= Rx FIFO 1 fill level reached watermark
- Bit 4** **RF1N:** Rx FIFO 1 New Message
 0= No new message written to Rx FIFO 1
 1= New message written to Rx FIFO 1
- Bit 3** **RF0L:** Rx FIFO 0 Message Lost
 0= No Rx FIFO 0 message lost
 1= Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
- Bit 2** **RF0F:** Rx FIFO 0 Full
 0= Rx FIFO 0 not full
 1= Rx FIFO 0 full
- Bit 1** **RF0W:** Rx FIFO 0 Watermark Reached
 0= Rx FIFO 0 fill level below watermark
 1= Rx FIFO 0 fill level reached watermark
- Bit 0** **RF0N:** Rx FIFO 0 New Message
 0= No new message written to Rx FIFO 0
 1= New message written to Rx FIFO 0

2.3.16 Interrupt Enable (IE)

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

- 0= Interrupt disabled
- 1= Interrupt enabled

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x54	STEE	FOEE	ACKEE	BEE	CRCEE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	UMDE	TSWE
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

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

Table 17 Interrupt Enable (address 0x54)

- Bit 31** **STEE:** Stuff Error Interrupt Enable
- Bit 30** **FOEE:** Format Error Interrupt Enable
- Bit 29** **ACKEE:** Acknowledge Error Interrupt Enable
- Bit 28** **BEE:** Bit Error Interrupt Enable
- Bit 27** **CRCEE:** CRC Error Interrupt Enable
- Bit 26** **WDIE:** Watchdog Interrupt Enable
- Bit 25** **BOE:** Bus_Off Status Interrupt Enable
- Bit 24** **EWE:** Warning Status Interrupt Enable
- Bit 23** **EPE:** Error Passive Interrupt Enable
- Bit 22** **ELOE:** Error Logging Overflow Interrupt Enable
- Bit 21** **BEUE:** Bit Error Uncorrected Interrupt Enable
- Bit 20** **BECE:** Bit Error Corrected Interrupt Enable
- Bit 19** **DRXE:** Message stored to Dedicated Rx Buffer Interrupt Enable
- Bit 18** **TOOE:** Timeout Occurred Interrupt Enable
- Bit 17** **UMDE:** Unprocessed Message Discarded 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

2.3.17 Interrupt Line Select (ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines.

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	STEL	FOEL	ACKEL	BEL	CRCEL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	UMDL	TSWL
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
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

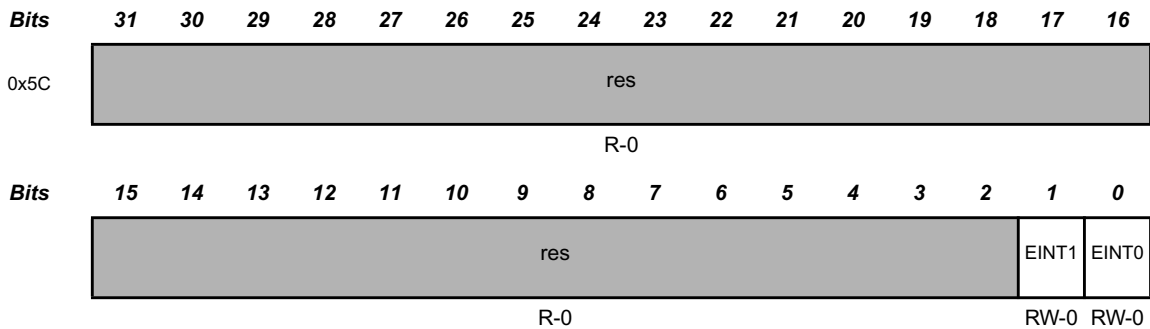
Table 18 Interrupt Line Select (address 0x58)

Bit 31	STEL:	Stuff Error Interrupt Line
Bit 30	FOEL:	Format Error Interrupt Line
Bit 29	ACKEL:	Acknowledge Error Interrupt Line
Bit 28	BEL:	Bit Error Interrupt Line
Bit 27	CRCEL:	CRC Error Interrupt Line
Bit 26	WDIL:	Watchdog Interrupt Line
Bit 25	BOL:	Bus_Off Status Interrupt Line
Bit 24	EWL:	Warning Status Interrupt Line
Bit 23	EPL:	Error Passive Interrupt Line
Bit 22	ELOL:	Error Logging Overflow Interrupt Line
Bit 21	BEUL:	Bit Error Uncorrected Interrupt Line
Bit 20	BECL:	Bit Error Corrected Interrupt Line
Bit 19	DRXL:	Message stored to Dedicated Rx Buffer Interrupt Line
Bit 18	TOOL:	Timeout Occurred Interrupt Line
Bit 17	UMDL:	Unprocessed Message Discarded 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

2.3.18 Interrupt Line Enable (ILE)

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



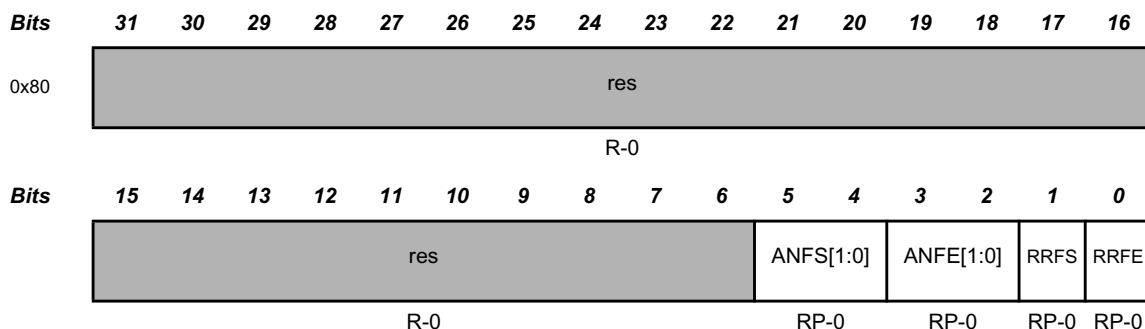
R = Read, W = Write; -n = value after reset

Table 19 *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

2.3.19 Global Filter Configuration (GFC)

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



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

Table 20 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

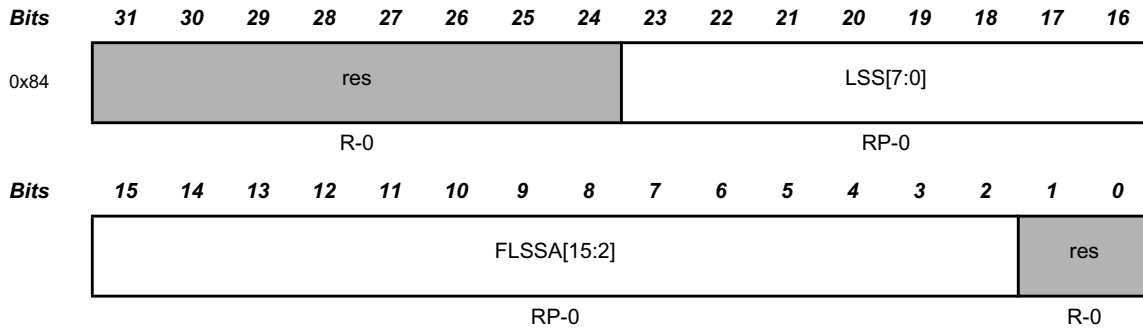
- 0= Filter remote frames with 11-bit standard IDs
- 1= Reject all remote frames with 11-bit standard IDs

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

2.3.20 Standard ID Filter Configuration (SIDFC)

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



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

Table 21 Standard ID Filter Configuration (address 0x84)

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

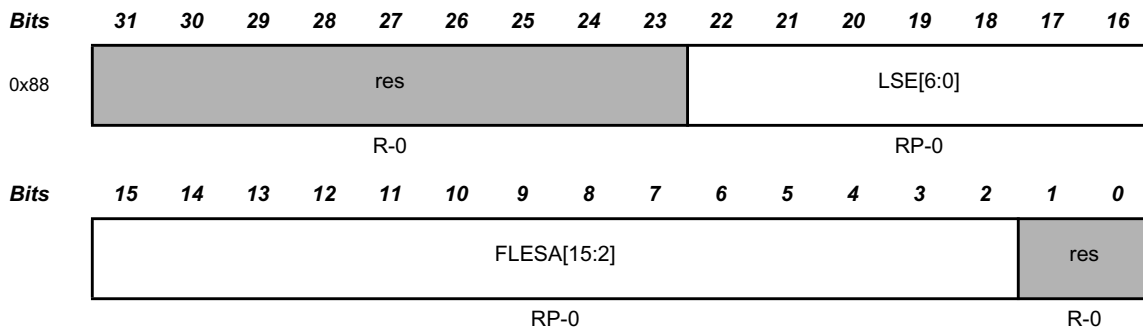
- 0= No standard Message ID filter
- 1-128= Number of standard Message ID filter elements
- >128= Values greater than 128 are interpreted as 128

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).

2.3.21 Extended ID Filter Configuration (XIDFC)

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



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

Table 22 Extended ID Filter Configuration (address 0x88)

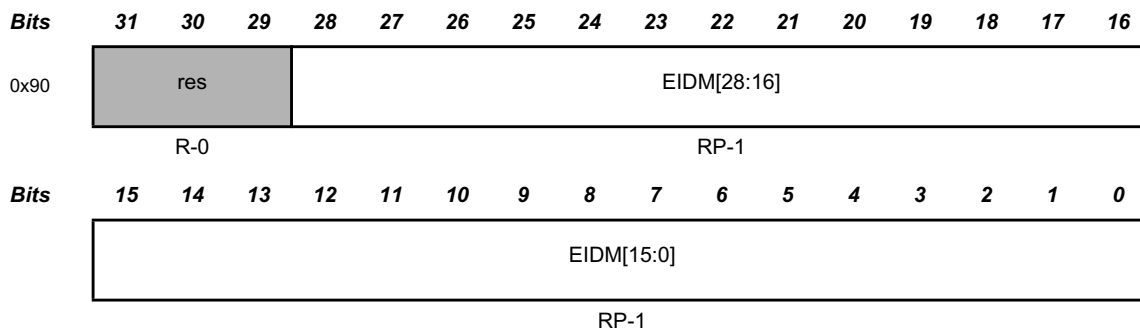
Bit 22:16 LSE[6:0]: List Size Extended

- 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).

2.3.22 Extended ID AND Mask (XIDAM)



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

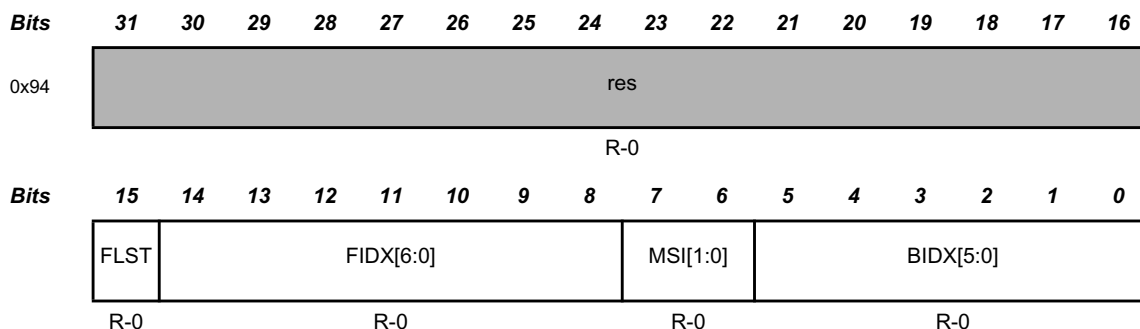
Table 23 Extended ID AND Mask (address 0x90)

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.

2.3.23 High Priority Message Status (HPMS)

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



R = Read; -n = Value after reset

Table 24 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 overrun

10= Message stored in FIFO 0

11= Message stored in FIFO 1

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

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

2.3.24 New Data 1 (NDAT1)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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 25 New Data 1 (address 0x98)

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

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

- 0= Rx Buffer not updated
- 1= Rx Buffer updated from new message

2.3.25 New Data 2 (NDAT2)

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

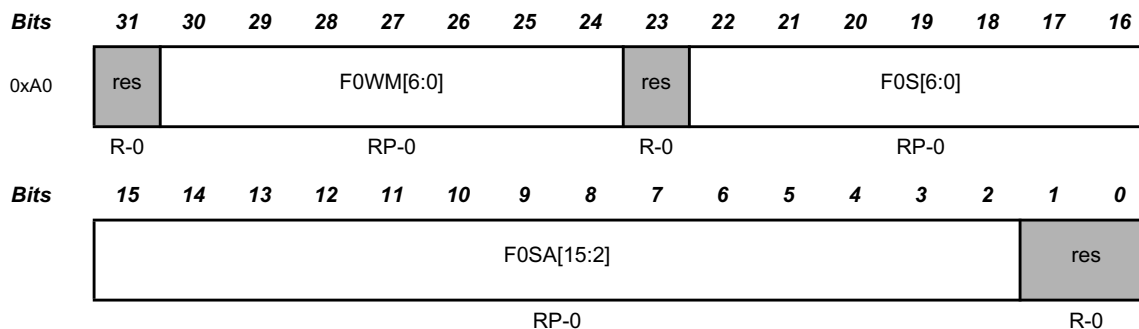
Table 26 New Data 2 (address 0x9C)

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

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

- 0= Rx Buffer not updated
- 1= Rx Buffer updated from new message

2.3.26 Rx FIFO 0 Configuration (RXF0C)

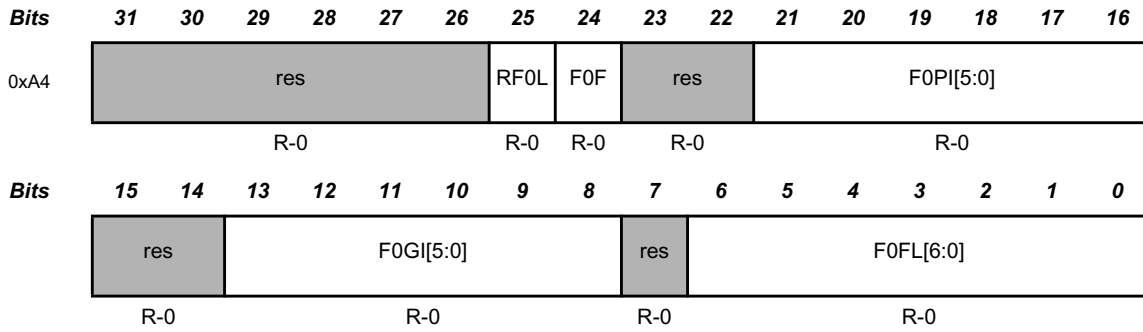


R = Read, P = Protected write; -n = Value after reset

Table 27 Rx FIFO 0 Configuration (address 0xA0)

- Bit 30:24 F0WM[6:0]:** Rx FIFO 0 Watermark
 0= Watermark interrupt disabled
 1-64= Level for Rx FIFO 0 watermark interrupt (**IR.RF0W**)
 >64= Watermark interrupt disabled
- Bit 22:16 F0S[6:0]:** Rx FIFO 0 Size
 0= No Rx FIFO 0
 1-64= Number of Rx FIFO 0 elements
 >64= Values greater than 64 are interpreted as 64
 The Rx FIFO 0 elements are indexed from 0 to **F0S**-1
- Bit 15:2 F0SA[15:2]:** Rx FIFO 0 Start Address
 Start address of Rx FIFO 0 in Message RAM (32-bit word address, see Figure 2).

2.3.27 Rx FIFO 0 Status (RXF0S)

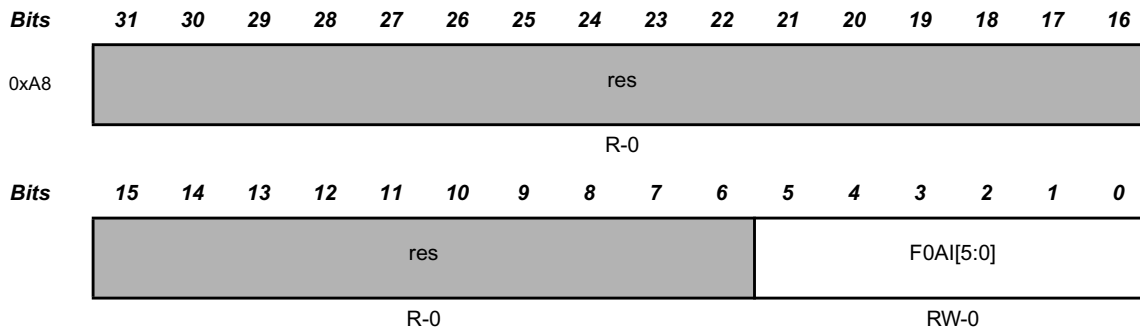


R = Read; -n = value after reset

Table 28 Rx FIFO 0 Status (address 0xA4)

- Bit 25 RF0L:** Rx FIFO 0 Message Lost
 This bit is a copy of interrupt flag **IR.RF0L**. When **IR.RF0L** is reset, this bit is also reset.
 0= No Rx FIFO 0 message lost
 1= Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
- 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. For internal use only.
- Bit 13:8 F0GI[5:0]:** Rx FIFO 0 Get Index
 Rx FIFO 0 read index pointer, range 0 to 63.
- Bit 6:0 F0FL[6:0]:** Rx FIFO 0 Fill Level
 Number of elements stored in Rx FIFO 0, range 0 to 64.

2.3.28 Rx FIFO 0 Acknowledge (RXF0A)



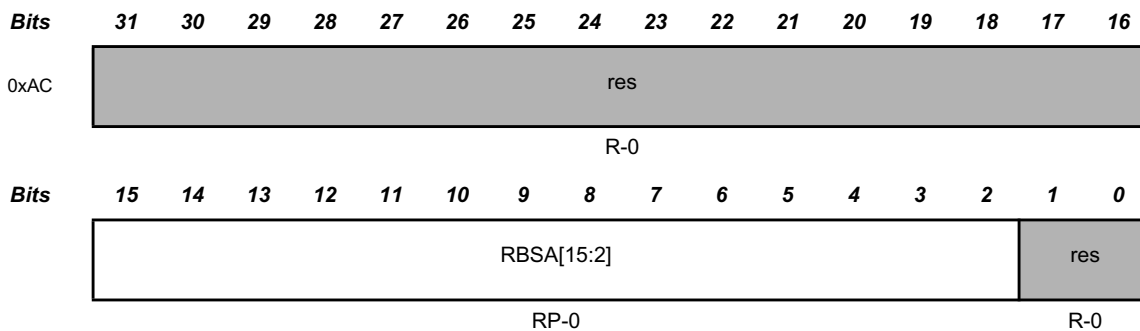
R = Read, W = Write; -n = value after reset

Table 29 Rx FIFO 0 Acknowledge (address 0xA8)

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

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

2.3.29 Rx Buffer Configuration (RXBC)



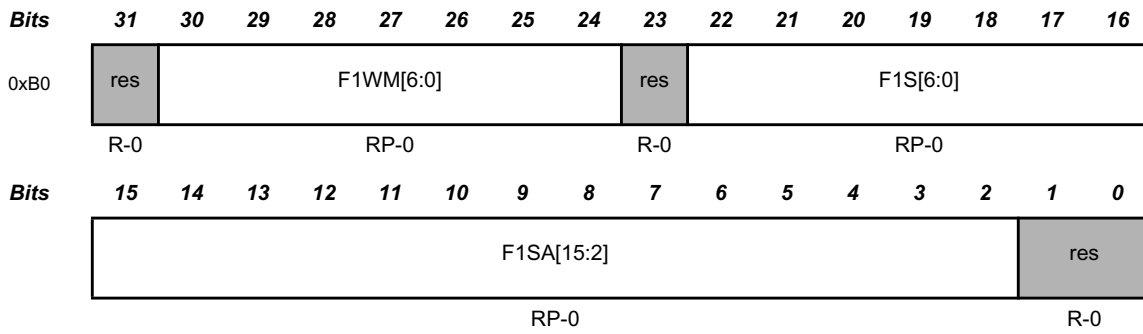
R = Read, P = Protected write; -n = value after reset

Table 30 Rx Buffer Configuration (address 0xAC)

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

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

2.3.30 Rx FIFO 1 Configuration (RXF1C)



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

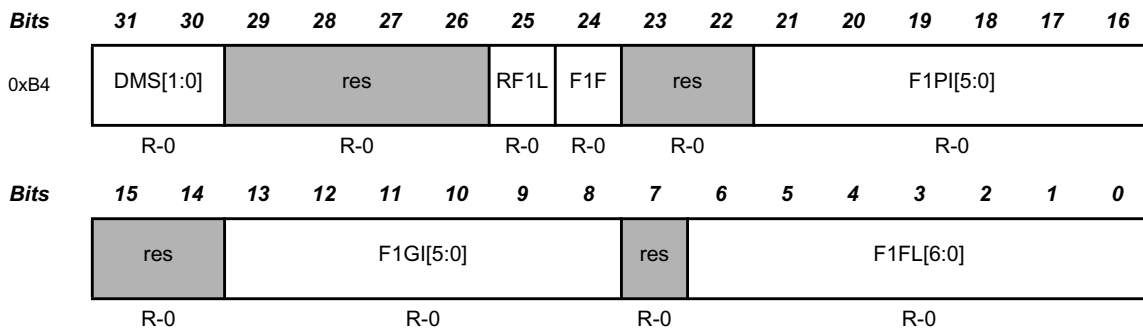
Table 31 Rx FIFO 1 Configuration (address 0xB0)

- Bit 30:24 F1WM[6:0]:** Rx FIFO 1 Watermark
 - 0= Watermark interrupt disabled
 - 1-64= Level for Rx FIFO 1 watermark interrupt (**IR.RF1W**)
 - >64= Watermark interrupt disabled

- Bit 22:16 F1S[6:0]:** Rx FIFO 1 Size
 - 0= No Rx FIFO 1
 - 1-64= Number of Rx FIFO 1 elements
 - >64= Values greater than 64 are interpreted as 64
 The Rx FIFO 1 elements are indexed from 0 to **F1S** - 1

- Bit 15:2 F1SA[15:2]:** Rx FIFO 1 Start Address
 - Start address of Rx FIFO 1 in Message RAM (32-bit word address, see Figure 2).

2.3.31 Rx FIFO 1 Status (RXF1S)



R = Read; -n = value after reset

Table 32 Rx FIFO 1 Status (address 0xB4)

- Bits 31:30 DMS[1:0]:** Debug Message Status
 - 00= Idle state, wait for reception of debug messages, DMA request is cleared
 - 01= Debug message A received
 - 10= Debug messages A, B received
 - 11= Debug messages A, B, C received, DMA request is set

- Bit 25 RF1L:** Rx FIFO 1 Message Lost
 - This bit is a copy of interrupt flag **IR.RF1L**. When **IR.RF1L** is reset, this bit is also reset.
 - 0= No Rx FIFO 1 message lost
 - 1= Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero

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. For internal use only.

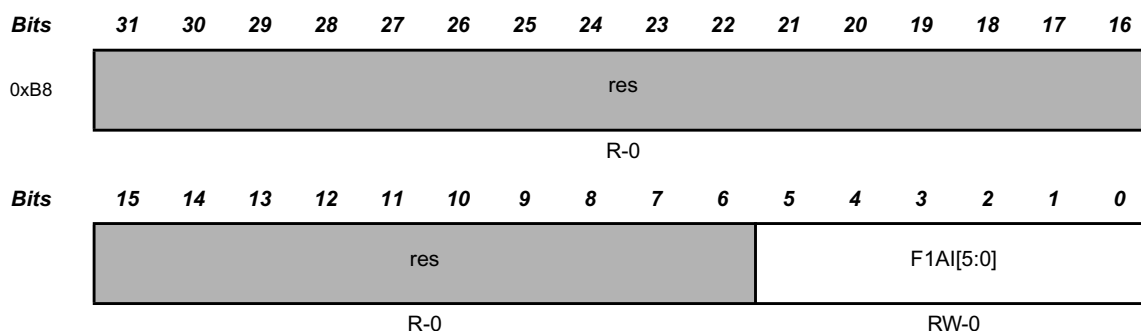
Bit 13:8 F1GI[5:0]: Rx FIFO 1 Get Index

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

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

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

2.3.32 Rx FIFO 1 Acknowledge (RXF1A)



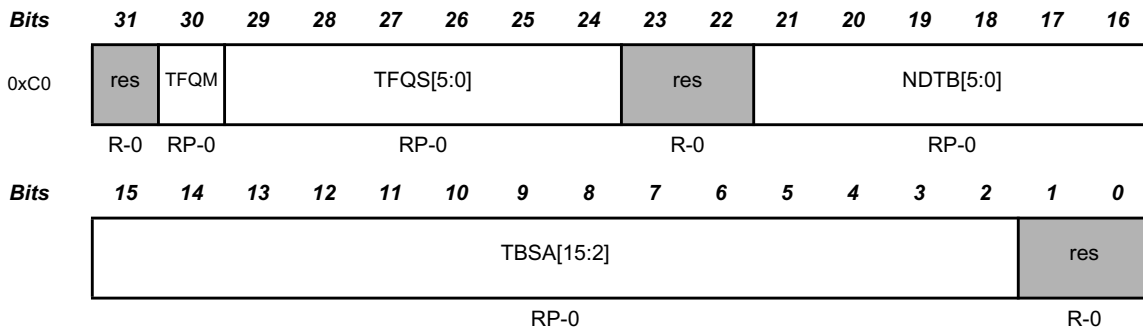
R = Read, W = Write; -n = value after reset

Table 33 Rx FIFO 1 Acknowledge (address 0xB8)

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

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

2.3.33 Tx Buffer Configuration (TXBC)



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

Table 34 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

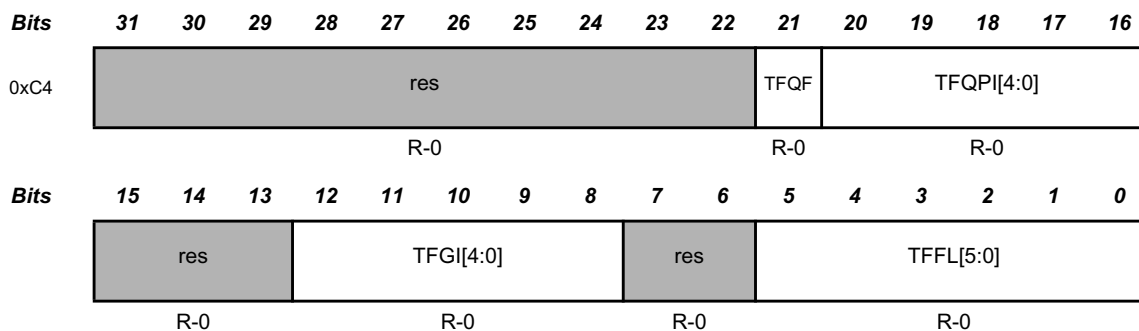
Bit 15:2 TBSA[15:2]: Tx Buffers Start Address

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.

2.3.34 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).



R = Read; -n = value after reset

Table 35 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. For internal use only. 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.

2.3.35 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 36 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.*

2.3.36 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 37 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.

2.3.37 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 38 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

2.3.38 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 39 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

2.3.39 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 40 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

2.3.40 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 41 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

2.3.41 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 42 Tx Buffer Cancellation Finished Interrupt Enable (address 0xE4)

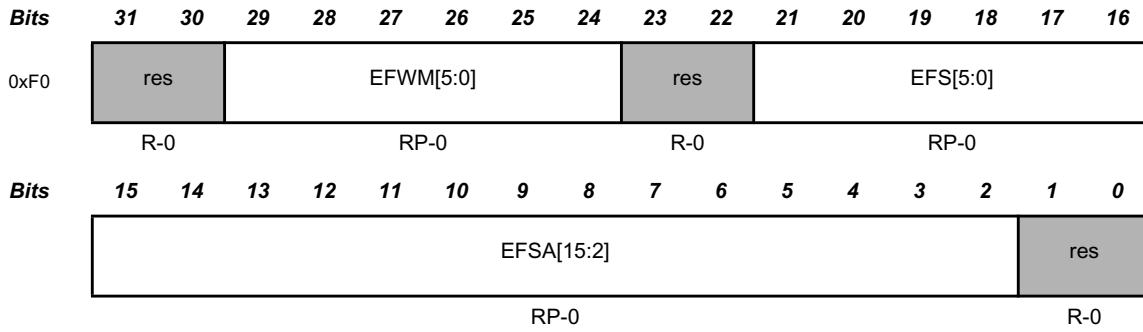
Bit 31:0 CFIE[31:0]: Cancellation Finished Interrupt Enable

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

0= Cancellation finished interrupt disabled

1= Cancellation finished interrupt enabled

2.3.42 Tx Event FIFO Configuration (TXEFC)



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

Table 43 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

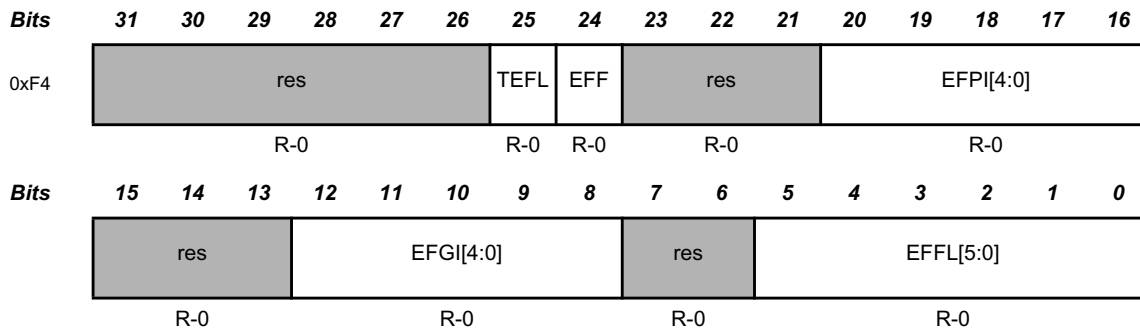
- Bit 21:16 EFS[5:0]:** Event FIFO Size
 - 0= Tx Event FIFO disabled
 - 1-32= Number of Tx Event FIFO elements
 - >32= Values greater than 32 are interpreted as 32

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).

2.3.43 Tx Event FIFO Status (TXEFS)

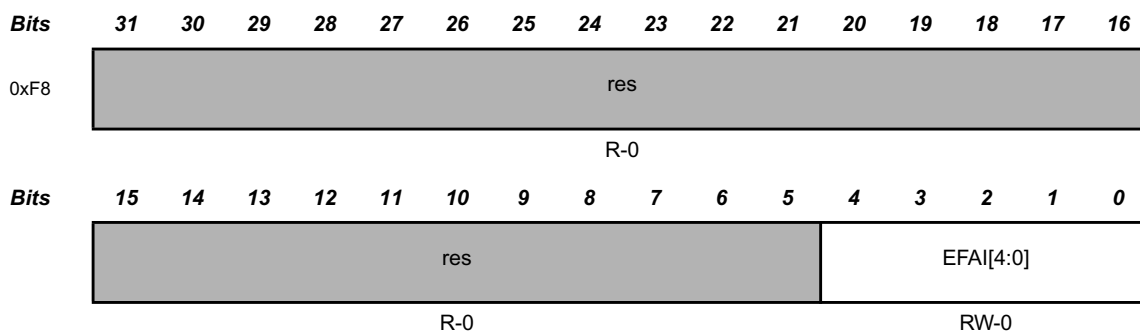


R = Read; -n = value after reset

Table 44 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. For internal use only.
- 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.

2.3.44 Tx Event FIFO Acknowledge (TXEFA)



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

Table 45 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 FIFO 0 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.

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 1216 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.

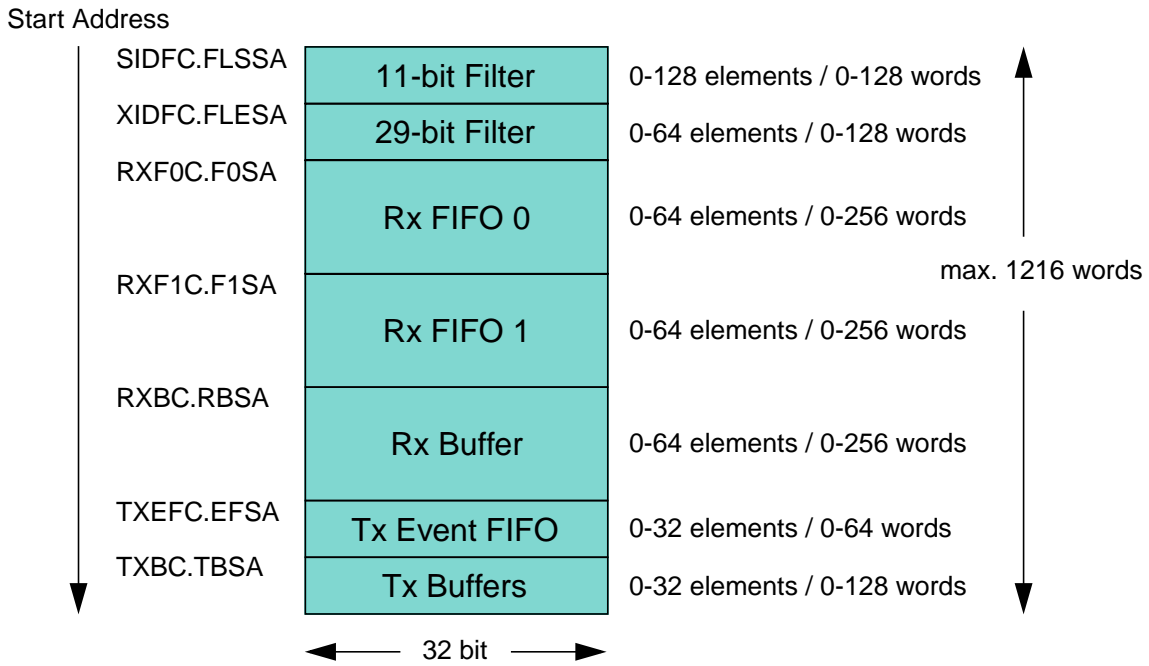


Figure 2 Message RAM Configuration

When the M_CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configured start addresses are 32-bit word addresses.

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 46 below.

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

Table 46 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

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

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

R1 Bit 31 ANMF: Accepted Non-matching Frame

- Acceptance of non-matching frames may be enabled via **GFC.ANFS** and **GFC.ANFE**.
- 0= Received frame matching filter index **FIDX**
 - 1= Received frame did not match any Rx filter element

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

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

R1 Bit 21 EDL: Extended Data Length

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

R1 Bit 20 BRS: Bit Rate Switch

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

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

- 0-8= Received frame has 0-8 data bytes
- 9-15= Received frame has 8 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

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**.

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

Table 47 Tx Buffer Element

T0 Bit 30 XTD: Extended Identifier

0= 11-bit standard identifier

1= 29-bit extended identifier

T0 Bit 29 RTR: Remote Transmission Request

0= Transmit data frame

1= Transmit remote frame

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

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

T1 Bits 31:24 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 Bits 19:16 DLC[3:0]: Data Length Code

0-8= Transmit frame with 0-8 data bytes

9-15= Transmit frame with 8 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

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**.

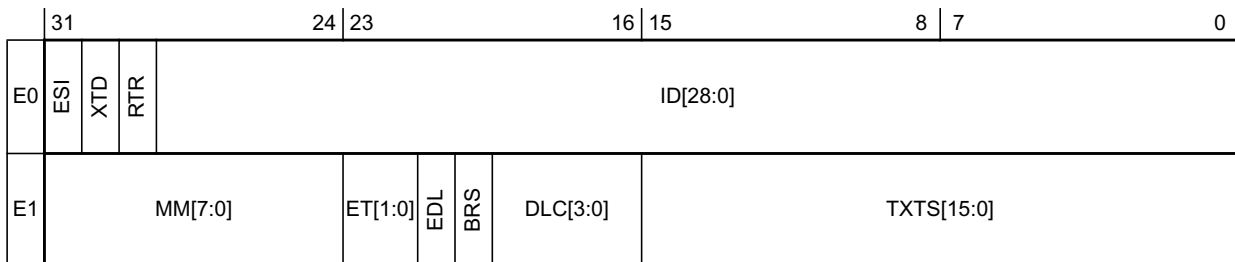


Table 48 Tx Event FIFO Element

E0 Bit 31 ESI: Error State Indicator

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

E0 Bit 30 XTD: Extended Identifier

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

E0 Bit 29 RTR: Remote Transmission Request

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

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

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

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

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

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

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

E1 Bit 21 EDL: Extended Data Length

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

E1 Bit 20 BRS: Bit Rate Switch

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

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

- 0-8= Frame with 0-8 data bytes transmitted
- 9-15=Frame with 8 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).

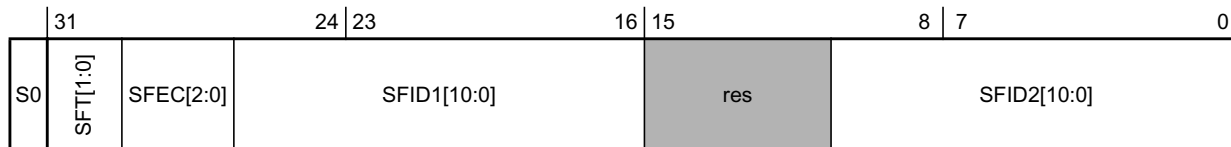


Table 49 *Standard Message ID Filter Element*

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

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

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

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

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

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

First ID of standard ID filter element.

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

Bits 10:0 **SFID2[10:0]:** Standard Filter ID 2

This bit field has a different meaning depending on the configuration of **SFEC**:

- 1) **SFEC** = "001"... "110" Second ID of standard ID filter element
- 2) **SFEC** = "111" Filter for Rx Buffers or for debug messages

SFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.

- 00= Store message into an Rx Buffer
- 01= Debug Message A
- 10= Debug Message B
- 11= Debug Message C

SFID2[8:6] is used to control the filter event pins **m_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).

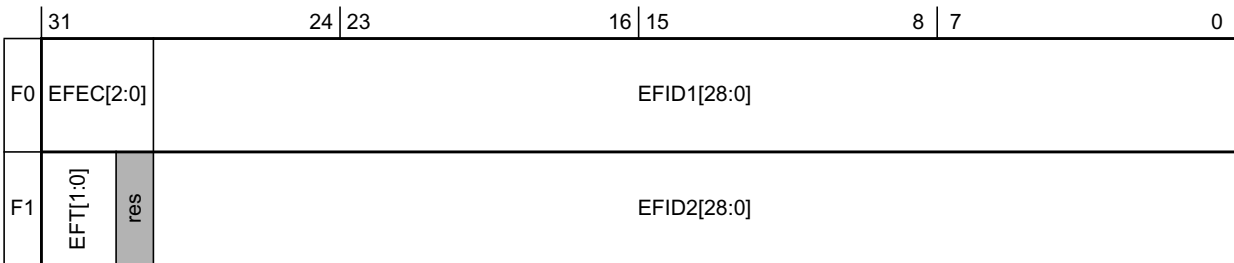


Table 50 Extended Message ID Filter Element

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 **EF1ID** to **EF2ID** (**EF2ID** ≥ **EF1ID**)

01= Dual ID filter for **EF1ID** or **EF2ID**

10= Classic filter: **EF1ID** = filter, **EF2ID** = mask

11= Range filter from **EF1ID** to **EF2ID** (**EF2ID** ≥ **EF1ID**), XIDAM mask not applied

F1 Bits 28:0 EFID2[28:0]: Extended Filter ID 2

This bit field has a different meaning depending on the configuration of **EFEC**:

1) **EFEC** = "001"...110" Second ID of extended ID filter element

2) **EFEC** = "111" Filter for Rx Buffers or for debug messages

EFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.

00= Store message into an Rx Buffer

01= Debug Message A

10= Debug Message B

11= Debug Message C

EFID2[8:6] is used to control the filter event pins **m_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 (\equiv *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'.

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 stages in the CAN FD protocol, first the Long Frame Mode where the data field of a CAN frame may be longer than 8 bytes. The second stage is the Fast Frame Mode where control field, data field, and CRC field of a CAN Frame are transmitted with a higher bit rate than the beginning and the end of the frame. Fast Frame Mode can only be used in combination with Long Frame Mode.

The Long Frame Mode is enabled by setting bit **CCCR.LONG** in the CC Control Register. The Fast Frame Mode is enabled by setting bit **CCCR.FAST** in the CC Control Register. Both bits can only be set while **CCCR.INIT** and **CCCR.CCE** are both set.

When the initialization is left (**CCCR.INIT** set to '0'), both Long Frame Mode and Fast Frame Mode are inactive, they need to be activated by writing '1' to **CCCR.SETL** and to **CCCR.SETF**. The bits **CCCR.LACT** and **CCCR.FACT** show whether the modes are active. Activation (and de-activation by **CCCR.CLRL** and **CCCR.CLRF**) will occur next time the CAN protocol controller FSM reaches idle phase between CAN frames. Upon this event bits **CCCR.SETL**, **CCCR.CLRL**, **CCCR.SETF**, or **CCCR.CLRF** will be cleared automatically. Fast Frame Mode can only be activated when Long Frame Mode is active. The bits **CCCR.LACT** and **CCCR.FACT** should not be changed while there are pending transmission requests.

The configuration bit **CCCR.LONG** decides how received frames are interpreted. If **CCCR.LONG** is set, the reserved bit in CAN frames with 11-bit identifiers and the first reserved bit in CAN frames with 29-bit identifiers will be decoded as **EDL** bit. **EDL = recessive** signifies a CAN FD long frame, **EDL = dominant** signifies a standard CAN frame.

In a CAN FD long frame, the two bits following **EDL**, **r0** and **BRS**, decide whether this CAN FD long frame is also a CAN FD fast frame. A CAN FD fast frame is signified by **r0 = dominant** and **BRS = recessive**. The coding of **r0 = recessive** is reserved for future expansion of the protocol.

The configuration bits **CCCR.LACT** and **CCCR.FACT** decide the format of transmitted frames. When **CCCR.LACT** is set, frames will be transmitted in CAN FD long format with **EDL = recessive**. When both **CCCR.LACT** and **CCCR.FACT** are set, frames will be transmitted in CAN FD Fast format with both **EDL** and **BRS = recessive**.

In the CAN FD long 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 51 below.

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

Table 51 Coding of DLC in CAN FD

In CAN FD fast frames, the bit timing will be switched inside the frame, after the **BRS** (Bit Rate Switch) bit, if this bit is *recessive*. Before the **BRS** bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Bit Timing & Prescaler Register **BTP**. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the Fast Bit Timing & Prescaler

Register **FBTP**. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit **ESI** (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, **ESI** is transmitted *recessive*, else it is transmitted *dominant*. In CAN FD remote frames the **ESI** bit is always transmitted dominant, independent of the transmitter's error state. The data length code of CAN FD remote frames is transmitted as *zero*.

In case a M_CAN Tx Buffer is configured for CAN FD transmission with DLC > 8, the first 8 bytes are transmitted as configured in the Tx Buffer while the remaining part of the data field is padded with 0xCC. When the M_CAN receives a CAN FD frame with DLC > 8, the first 8 bytes of that frame are stored into the matching Rx Buffer or Rx FIFO. The remaining bytes are discarded.

3.1.4 Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin **m_can_tx** the protocol controller receives the transmitted data from its local CAN transceiver via pin **m_can_rx**. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

3.1.4.1 Description

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver. Figure 3 below describes how the transceiver loop delay is measured.

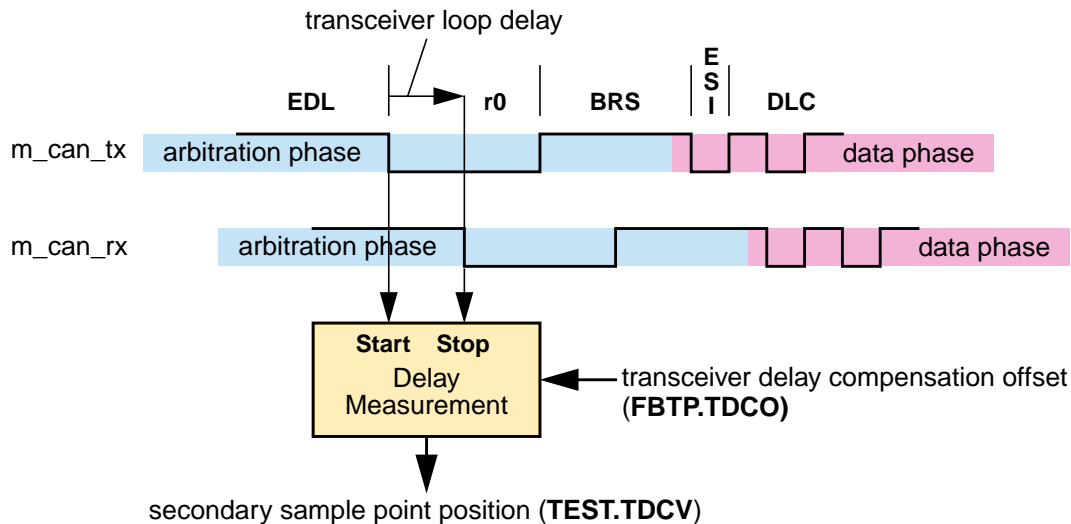


Figure 3 Transceiver delay measurement

Within each CAN FD frame, the transmitter measures the delay between the data transmitted at pin **m_can_tx** and the data received at pin **m_can_rx**, starting with the falling edge of bit **EDL**. The delay is measured in **m_can_cclk** periods.

A secondary sample point is calculated by adding a configurable transceiver delay compensation offset (**FBTP.TDCO**) to the measured transceiver delay. The transceiver delay compensation offset is used to adjust the secondary sample point inside the bit time (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of time quanta **tq**.

To check for bit errors during the data phase, the delayed transmit data is compared against the received data at the secondary sample point. If a bit error is detected at the secondary sample point, the transmitter will react to this bit error at the next following sample point. During arbitration phase the delay compensation is always disabled.

The maximum delay which can be compensated by the M_CAN's delay compensation during the data phase is three bit times.

3.1.4.2 Configuration and Status

Compensation for the transceiver loop delay by the M_CAN is enabled via **FBTP.TDC**. The transceiver delay compensation offset is configured via **FBTP.TDCO**. The actual delay compensation value applied by the M_CAN's protocol engine can be read from **TEST.TDCV**.

3.1.5 Bus Monitoring Mode

The M_CAN is set in Bus Monitoring Mode by programming **CCCR.MON** to *one*. In Bus Monitoring Mode (see ISO11898-1, 10.12 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.

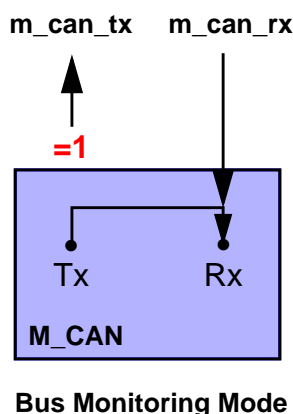


Figure 4 Pin Control in Bus Monitoring Mode

3.1.6 Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 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, chapter 9.2, the automatic retransmission may be disabled via **CCCR.DAR**.

3.1.6.1 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).

3.1.7 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**.

3.1.8 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.

3.1.8.1 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.

3.1.8.2 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.

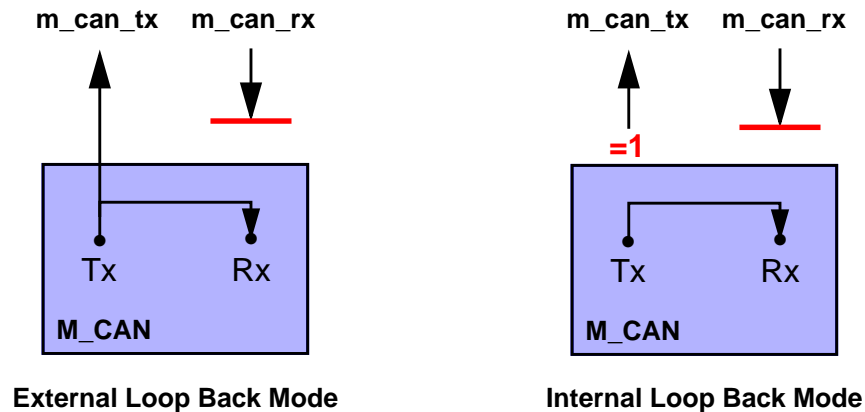


Figure 5 Pin Control in Loop Back Modes

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 baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.*

3.4 Rx Handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

3.4.1 Acceptance Filtering

The M_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
- 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

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.

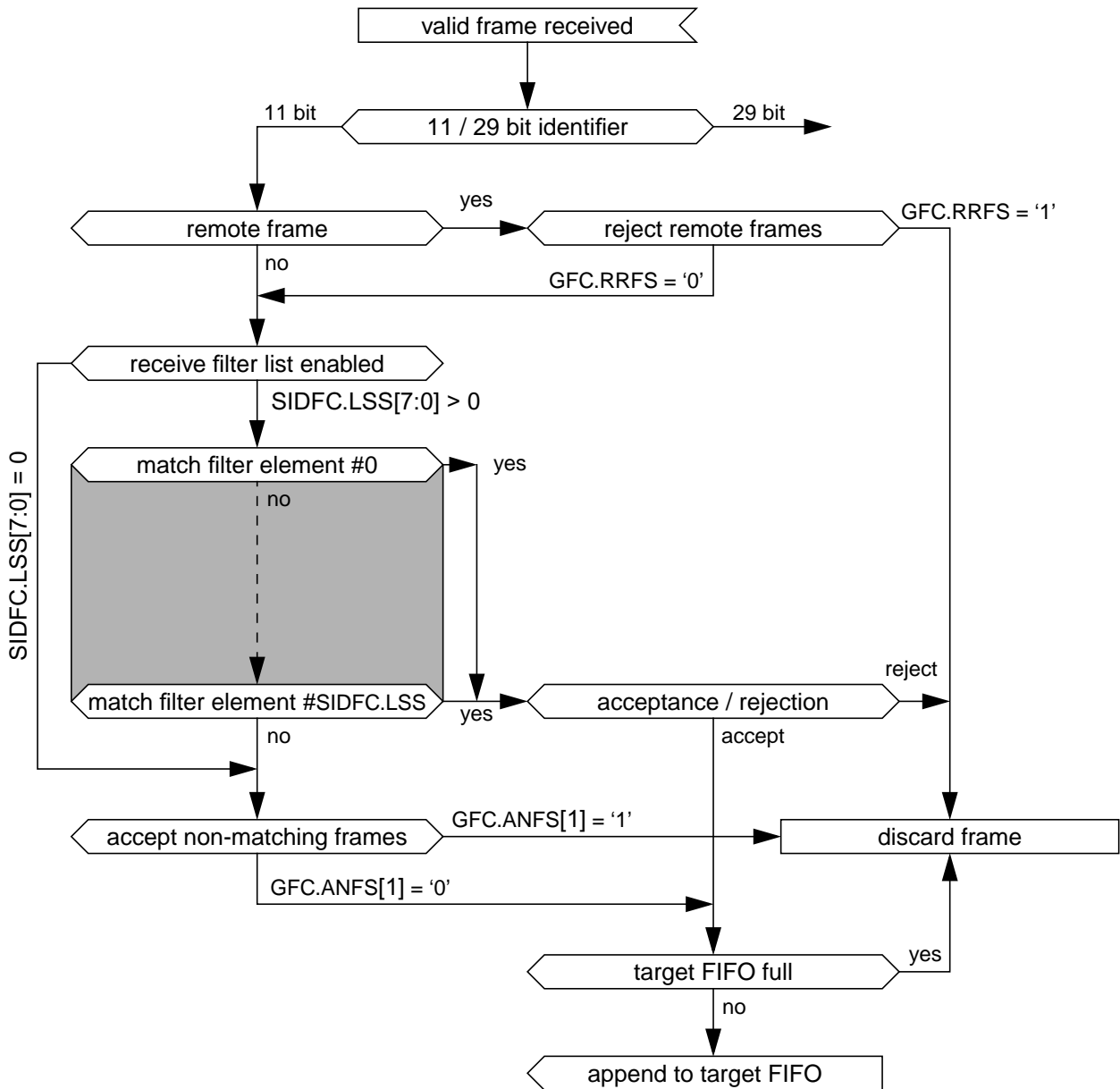


Figure 6 Standard Message ID Filter Path

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.

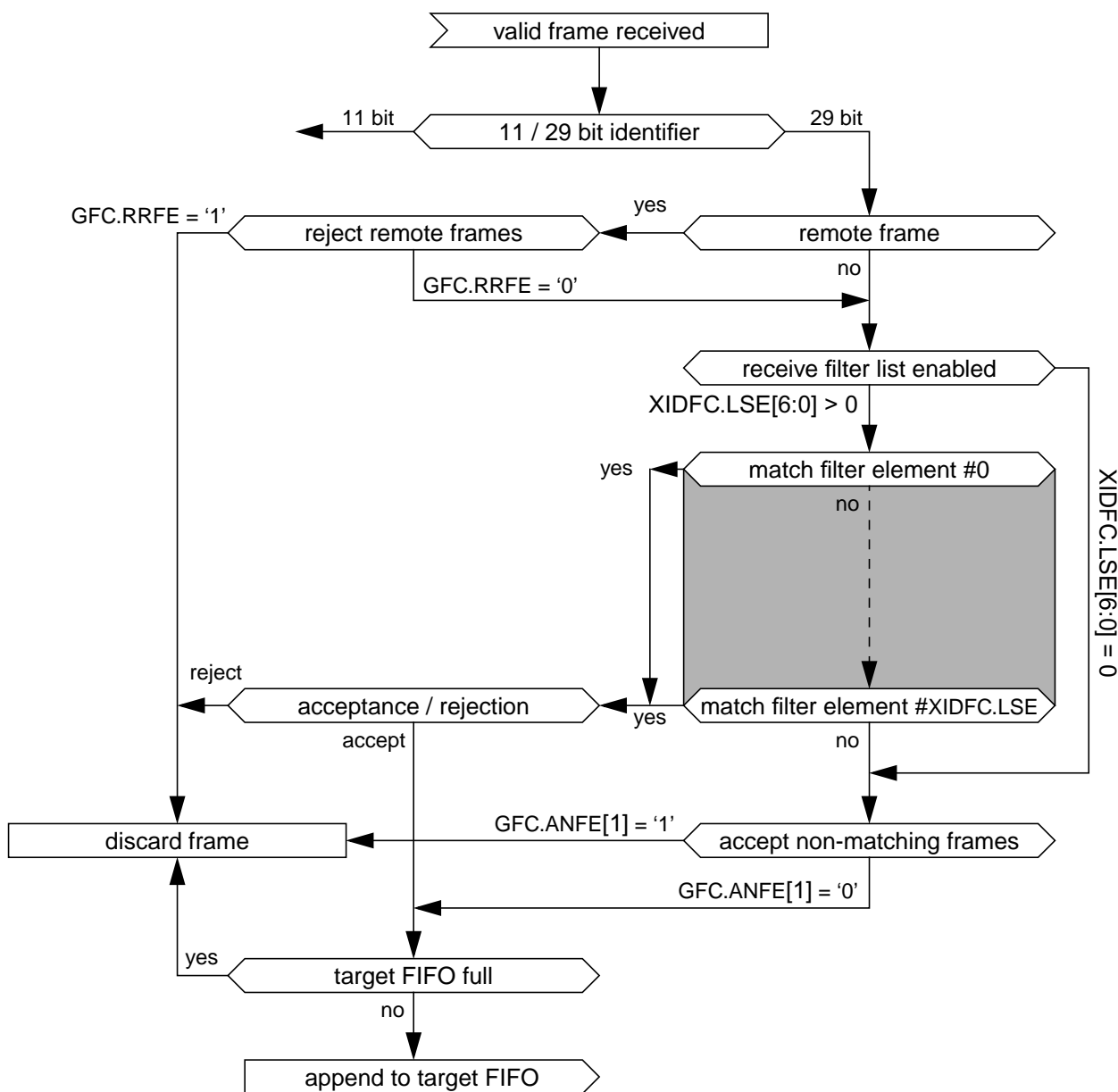


Figure 7 Extended Message ID Filter Path

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.

When an Rx FIFO full condition is signalled by **IR.RFnF**, 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. In case a message is received while the corresponding Rx FIFO is full, this message is discarded and interrupt flag **IR.RFnL** is set.

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 reading from an Rx FIFO, four times the Rx FIFO Get Index **RXFnS.FnGI** has to be added to the corresponding Rx FIFO start address **RXFnC.FnSA**.

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.

<i>Filter Element</i>	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 52 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 2.2 and Section 2.3). 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**.

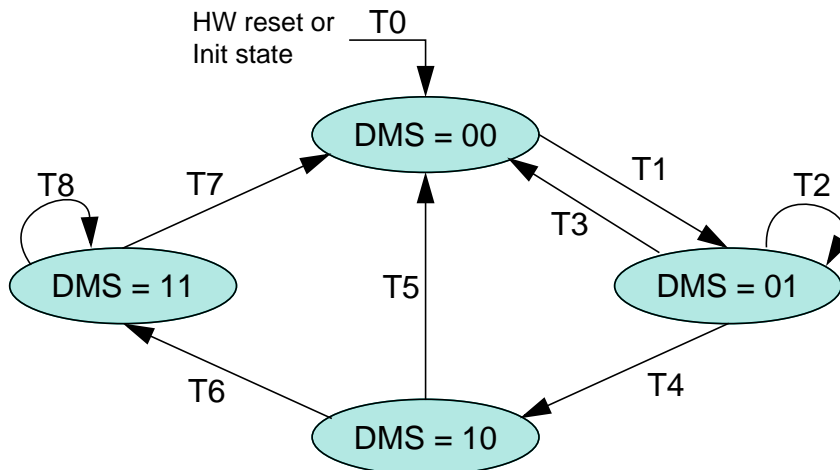
<i>Filter Element</i>	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

Table 53 Example Filter Configuration for Debug Messages

3.4.4.2 Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signalled via **RXF1S.DMS**.



T0: reset m_cam_dma_req output, enable reception of debug messages A, B, and C

T1: reception of debug message A

T2: reception of debug message A

T3: reception of debug message C

T4: reception of debug message B

T5: reception of debug messages A, B

T6: reception of debug message C

T7: DMA transfer completed

T8: reception of debug message A,B,C (message rejected)

Debug Message Handling State Machine

3.5 Tx Handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in Section 2.4.3.

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.

Note: *AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation*

3.5.1 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 four 32-bit words in the Message RAM. Therefore the start address of a Dedicated Tx Buffer in the Message RAM is calculated by adding four times the transmit buffer index (0...31) to the Tx Buffer Start Address **TXBC.TBSA**.

3.5.2 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 four 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding four times the Put Index **TXFQS.TFQPI** (0...31) to the Tx Buffer Start Address **TXBC.TBSA**.

3.5.3 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 four 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding four times the Tx Queue Put Index **TXFQS.TFQPI** (0...31) to the Tx Buffer Start Address **TXBC.TBSA**.

3.5.4 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.

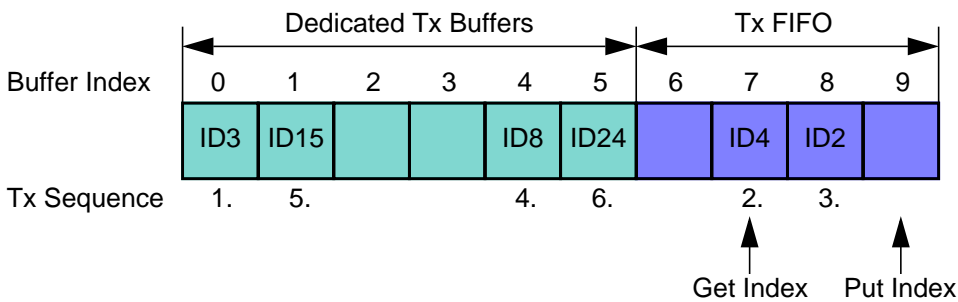


Figure 8 Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO

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

3.5.5 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.

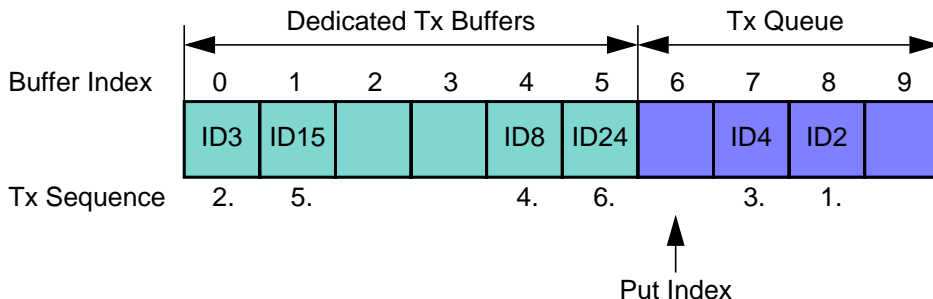


Figure 9 Example of mixed Configuration Dedicated Tx Buffers / Tx Queue

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

3.5.6 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.*

3.5.7 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.

When a Tx Event FIFO full condition is signalled by **IR.TEFF**, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag **IR.TEFL** is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by **TXEFC.EFWM**, interrupt flag **IR.TEFW** is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index **TXEFS.EFGI** has to be added to the Tx Event FIFO start address **TXEFC.EFSA**.

3.6 FIFO Acknowledge Handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see Section 2.3.28, Section 2.3.32, and Section 2.3.44). 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.*

Chapter 4.

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_ddd								
0x04	ETV[31:0]																															ENDN 8765_4321	
0x08	CUST[31:0]																															CUST t.b.d.	
0x0C	TDCO[3:0]			TDC			FBRP[9:0]									FTSEG1[3:0]			FTSEG2[2:0]			FSJW[1:0]			FBTP 0000_0320								
0x10																	TDCV[4:0]				RX	TX[1:0]	LBC					TEST 0000_0080					
0x14																	WDV[7:0]				WDC[7:0]							RWD 0000_0000					
0x18																	FACT	LACT	CMR[1:0]	CME[1:0]	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT	CCCR 0000_0001				
0x1C	BRP[9:0]									TSEG1[5:0]					TSEG2[3:0]			SJW[3:0]			BTP 0000_0320												
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 54 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																																	TOCV 0000_FFFF	
0x40																																		ECR 0000_0000
0x44																																		PSR 0000_0707
0x50	STE	FOE	ACKE	BE	CRCE	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	UMD	TSW	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N	IR 0000_0000	
0x54	STEE	FOEE	ACKEE	BEE	CRCEE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	UMDE	TSWE	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE	IE 0000_0000	
0x58	STEL	FOEL	ACKEL	BEL	CRCEL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECLE	DRXL	TOOL	UMDL	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
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	

Table 54 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
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	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	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					
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						
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

Table 54 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
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			EFP[4:0]					EFGI[4:0]							EFFL[5:0]					TXEFS 0000_0000						
0xF8																												EFAI[4:0]					TXEFA 0000_0000

Table 54 M_CAN Register Overview

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.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)
m_can_rxfid	OUT	CCLK	receive fast data

Table 55 M_CAN Module Interface

PORT	DIR	DOMAIN	DESCRIPTION
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
m_can_mrx	OUT	CCLK	message received
m_can_calf	OUT	CCLK	calibration field
m_can_aff	OUT	HCLK	acceptance filtering finished

Table 55 M_CAN Module Interface

Note: Signals *m_can_cok*, *m_can_spt*, *m_can_mrx*, *m_can_calf*, *m_can_aff* 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'.

4.3 Connection to external Message RAM

The M_CAN is prepared for connection to a module-external single-ported or dual-ported 32-bit Message RAM via its Generic Master Interface. Depending on the system requirements additional logic for Message RAM arbitration, initialization, parity checking, or ECC has to be attached to the Message RAM. Further information with respect to Message RAM connection can be found in [2] and [3].

4.4 Interface to DMA Controller

When all three debug messages A, B, C have been received in the correct order, M_CAN output signal **m_can_dma_req** is activated to trigger a DMA transfer. The RAM words holding debug messages A, B, C will not be changed by the M_CAN while **m_can_dma_req** is active.

After the transfer of the received messages has completed the DMA unit activates input signal **m_can_dma_ack**. This resets **m_can_dma_req**. The debug message handling state machine enters idle state (DMS = "00") and waits for reception of the next debug messages (see Figure 1, *Debug Message Handling State Machine*).

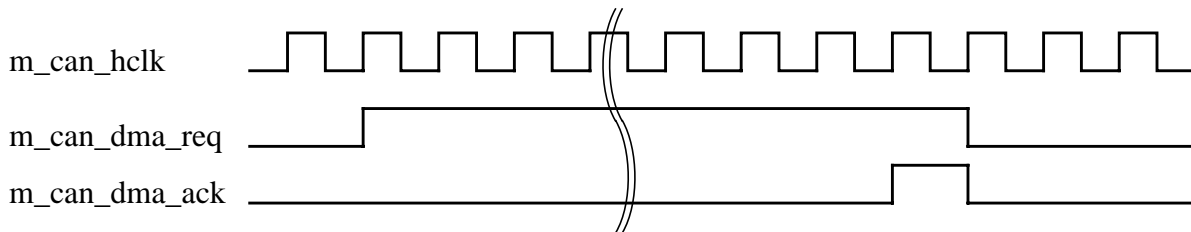


Figure 10 Timing of DMA Interface Signals

Note: If the DMA unit activates input signal **m_can_dma_ack** before the DMA transfer has completed, the Rx Buffer elements holding debug messages A, B, C are unlocked and may be overwritten by received debug messages.

List of Figures

Figure 1	M_CAN Block Diagram.....	2
Figure 2	Message RAM Configuration.....	43
Figure 3	Transceiver delay measurement.....	53
Figure 4	Pin Control in Bus Monitoring Mode.....	54
Figure 5	Pin Control in Loop Back Modes.....	56
Figure 6	Standard Message ID Filter Path.....	59
Figure 7	Extended Message ID Filter Path.....	60
Figure 8	Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO.....	65
Figure 9	Example of mixed Configuration Dedicated Tx Buffers / Tx Queue.....	66
Figure 10	Timing of DMA Interface Signals.....	75

List of Tables

Table 1	M_CAN Register Map.....	5
Table 2	Core Release Register (addresses 0x00).....	7
Table 3	Example for Coding of Revisions.....	7
Table 4	Endian Register (address 0x04).....	8
Table 5	Fast Bit Timing & Prescaler Register (address 0x0C).....	8
Table 6	Test Register (address 0x10).....	9
Table 7	RAM Watchdog (address 0x14).....	10
Table 8	CC Control Register (address 0x18).....	11
Table 9	Bit Timing & Prescaler Register (address 0x1C).....	13
Table 10	Timestamp Counter Configuration (address 0x20).....	14
Table 11	Timestamp Counter Value (address 0x24).....	14
Table 12	Timeout Counter Configuration (address 0x28).....	15
Table 13	Timeout Counter Value (address 0x2C).....	15
Table 14	Error Counter Register (address 0x40).....	16
Table 15	Protocol Status Register (address 0x44).....	17
Table 16	Interrupt Register (address 0x50).....	19
Table 17	Interrupt Enable (address 0x54).....	22
Table 18	Interrupt Line Select (address 0x58).....	24
Table 19	Interrupt Line Select (address 0x5C).....	25
Table 20	Global Filter Configuration (address 0x80).....	26
Table 21	Standard ID Filter Configuration (address 0x84).....	27
Table 22	Extended ID Filter Configuration (address 0x88).....	27
Table 23	Extended ID AND Mask (address 0x90).....	28
Table 24	High Priority Message Status (address 0x94).....	28
Table 25	New Data 1 (address 0x98).....	29
Table 26	New Data 2 (address 0x9C).....	29
Table 27	Rx FIFO 0 Configuration (address 0xA0).....	30
Table 28	Rx FIFO 0 Status (address 0xA4).....	31
Table 29	Rx FIFO 0 Acknowledge (address 0xA8).....	32
Table 30	Rx Buffer Configuration (address 0xAC).....	32
Table 31	Rx FIFO 1 Configuration (address 0xB0).....	33
Table 32	Rx FIFO 1 Status (address 0xB4).....	33
Table 33	Rx FIFO 1 Acknowledge (address 0xB8).....	34
Table 34	Tx Buffer Configuration (address 0xC0).....	35
Table 35	Tx FIFO/Queue Status (address 0xC4).....	36
Table 36	Tx Buffer Request Pending (address 0xCC).....	37
Table 37	Tx Buffer Add Request (address 0xD0).....	38
Table 38	Tx Buffer Cancellation Request (address 0xD4).....	38
Table 39	Tx Buffer Transmission Occurred (address 0xD8).....	39
Table 40	Transmit Buffer Cancellation Finished (address 0xDC).....	39
Table 41	Tx Buffer Transmission Interrupt Enable (address 0xE0).....	40
Table 42	Tx Buffer Cancellation Finished Interrupt Enable (address 0xE4).....	40
Table 43	Tx Event FIFO Configuration (address 0xF0).....	41
Table 44	Tx Event FIFO Status (address 0xF4).....	42
Table 45	Tx Event FIFO Acknowledge (address 0xF8).....	42
Table 46	Rx Buffer and FIFO Element.....	44
Table 47	Tx Buffer Element.....	45
Table 48	Tx Event FIFO Element.....	47
Table 49	Standard Message ID Filter Element.....	48
Table 50	Extended Message ID Filter Element.....	49
Table 51	Coding of DLC in CAN FD.....	52

Table 52	Example Filter Configuration for Rx Buffers.....	61
Table 53	Example Filter Configuration for Debug Messages	62
Table 54	M_CAN Register Overview.....	69
Table 55	M_CAN Module Interface	73