- Transmit Handler
- Receive Handler

The user interface of MCMCAN provides the following functionality:

- A configurable Message RAM to store the message to be transmitted or received. The message RAM is shared by all the CAN nodes within a MCMCAN module
- Grouping and signalling of interrupts through Interrupt Compression Unit
- Clock selection and generation through Clock Control Block
- Access protection through BPI
- Timer based transmission of CAN frame and timeouts for reception of CAN frames.

Note: Refer appendix of a product variant for the number of MCMCAN modules, number of M_CAN nodes within a module and Message RAM allocation for individual nodes.

40.3 Functional Description

This section describes the functionality of MCMCAN. The functionality is described as three different sub-sections:

- User Interface
- M_CAN functionality
- TTCAN operation

40.3.1 MCMCAN User Interface

The MCMCAN User Interface describes about the clock generation, grouping of interrupts, external connections to the module and IO configurations.

40.3.1.1 MCMCAN Clockpaths

A general overview of clocks within the MCMCAN module.

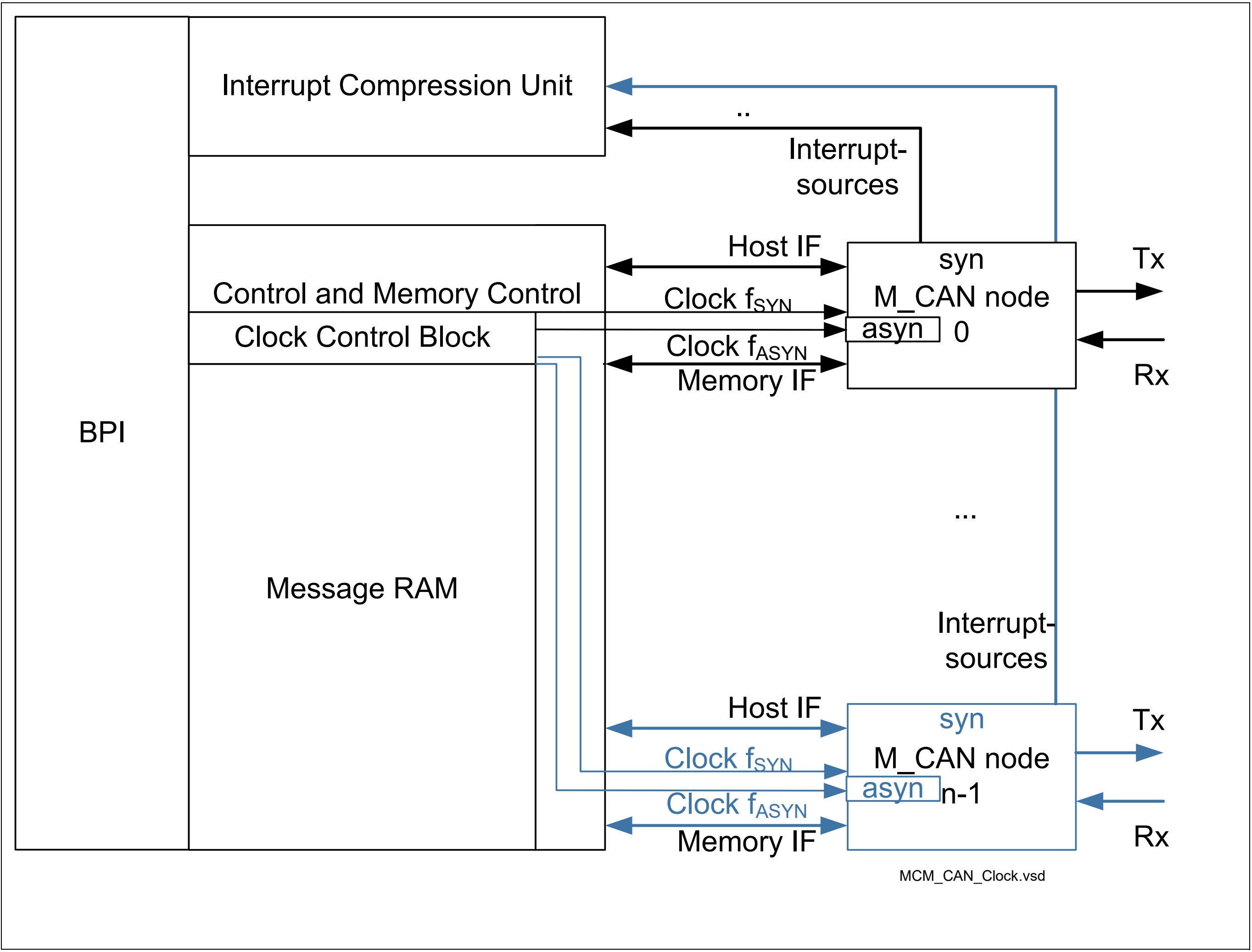


Figure 580 MCMCAN Clock Interconnects with CCU

The MCMCAN module clock inputs are connected to the Clock Control Unit (CCU). The CLC setting supplies the global module registers with its clocks. To supply the M_CAN nodes with the corresponding clocks, the **MCR**.CLKSELi registers have to be set. See **Figure 581** and **Table 354**. The asynchronous clock as well as the synchronous clock of each single M_CAN node can be switched on/off via MCR.CLKSELi register bitfields as shown in **Figure 582** and **Table 354** below.

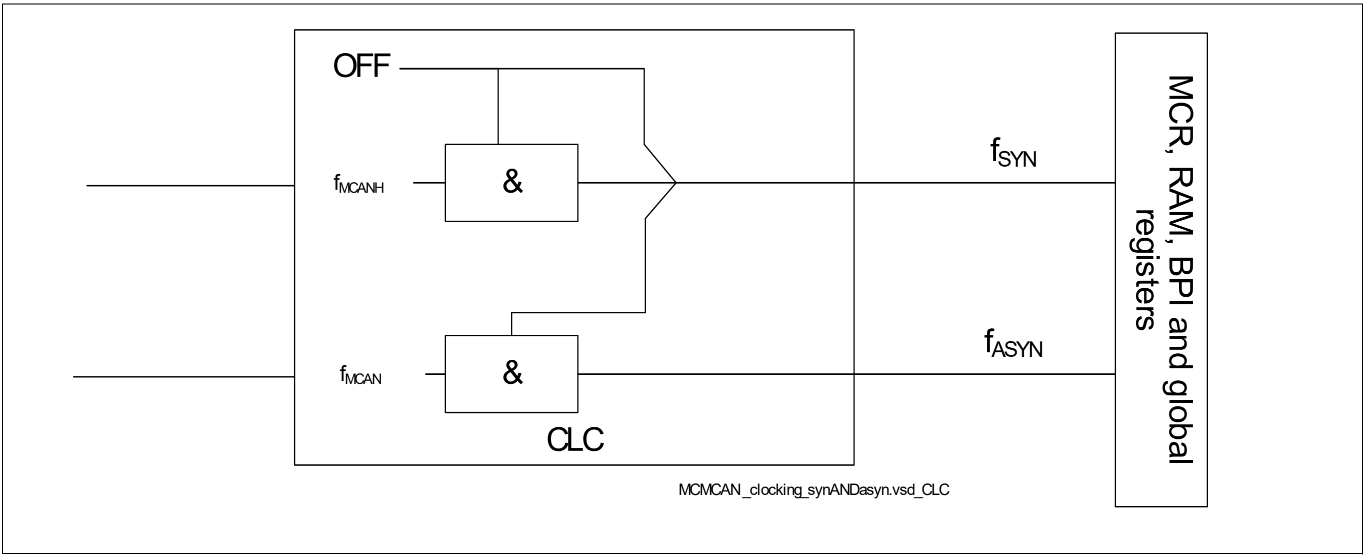

OFF
f_{MCANH}
&
f_{MCAN}
&
CLC
f_{SYN}
f_{ASYN}
MCR, RAM, BPI and global registers
MCMCAN_clocking_synANDasyn.vsd_CLC

Figure 581 MCMCAN Clock Switch via CLC

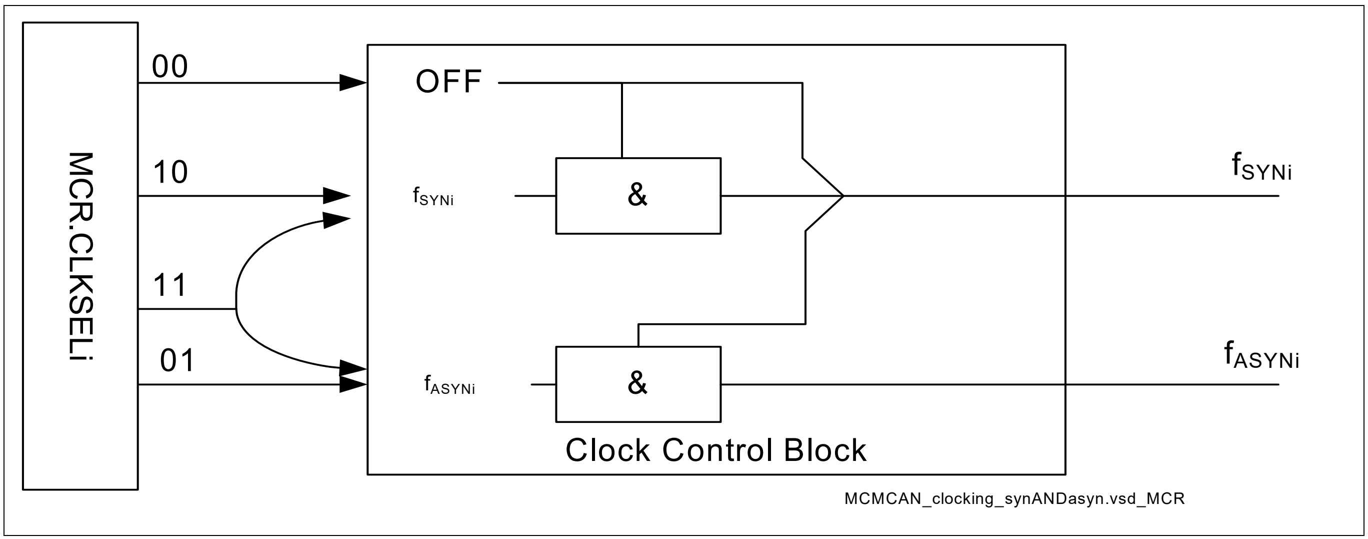


Figure 582 MCMCAN Clock Switch for single nodes

$f_{SYN}$is supplied from f_{MCANH} and f_{ASYN} is supplied from f_{MCAN} from CCU. f_{SYN} is used as the clock source for Register and RAM interface,f_{ASYN} is used to generate the nominal and fast CAN FD baudrates. It is recommended to use f_{ASYN} as 80, 40, 20 MHz from the Peripheral clock or also from f_{OSC}, in order to achieve commonly used nominal and fast CAN FD baudrates. The condition that f_{SYN} >= f_{ASYN} is essential for proper functioning of MCMCAN.

Note: f_{SPB} is used as the clock source for SSH of MCMCAN RAM . Since a synchronization between f_{SPB} and f_{MCANH} happens within SSH, during normal operation of MCMCAN, f_{SPB} should be configured to frequency equal to f_{MCANH}. During Pretended Networking where f_{SPB} is on reduced frequency than f_{MCANH}, the MCMCAN SSH registers should be accessed only after leaving the Pretended Networking Mode. Refer to MTU chapter for additional details.

Table 354 MCMCANClock Interconnects

CAN Clock Inputs	Connected to	Description
f_{MCAN}	CCU	f_{MCAN} of the MCMCAN module is one of the clock inputs of the Clock Control Block, providing the MCMCAN with the clock for the asynchronous clock path f_{ASYN}.
f_{MCANH}	CCU	f_{MCANH} of the MCMCAN module is the clock input of the Clock Control Register Block, providing the main kernel clock f_{SYN}.
f_{SYN}	CLC	f_{MCANH} becomes f_{SYN} within the module
f_{ASYN}	CLC	f_{MCAN} becomes f_{ASYN} within the module
f_{SYNi}	MCR.CLKSELi	f_{SYN} becomes f_{SYNi} clocking the synchronous part of M_CAN node i (Nodes can be switched on/off individually)
f_{ASYNi}	MCR.CLKSELi	f_{ASYN} becomes f_{ASYNi} clocking the asynchronous part of M_CAN node i (Nodes can be switched on/off individually)

40.3.1.1.1 Module Clock Generation

This chapter describes the clock generation and clock destinations.

Clock Selection

The asynchronous clock part of the M_CAN and the rest of the MCMCAN module are separate frequency domains and can be driven by separate independent frequencies. The clocks for the module are chosen within the clock control unit. The asynchronous clock can be chosen in the CCU among the Peripheral PLL or with direct drive from the oscillator.

The purpose of supplying the asynchronous clock part with a direct oscillator clock is to avoid the clock jitter added by the PLL, necessary when the chip is driven by a low cost ceramic resonator instead of by a high precision quartz crystal.

As shown in **Figure 581**, the clock signals for the MCMCAN module are generated and controlled by a clock control unit. This clock control unit is responsible for the enable/disable control, the clock frequency adjustment.

Clock control register

The global registers do include the **CLC** register: Module clock enabled. The module control clock f_{SYN} is used inside the MCMCAN module for control purposes such as clocking of control logic and register operations. The frequency of f_{SYN} is sourced by f_{MCANH} from CCU module. This clock is independent to f_{SPB} and allows M_CAN to continue operation when f_{SPB} is reduced in frequency, thus enabling pretended networking. The clock control register CLC makes it possible to enable/disable f_{SYN} and f_{ASYN} under certain conditions.

40.3.1.2 Interrupt groups

Interrupt grouping is fixed, as shown in figure **Figure 583** and **Figure 584**. For the complete module, 16 interrupt nodes are existing. The interrupt groups can be freely assigned to the nodes by using **GRINT1i (i=0-3)** and **GRINT2i (i=0-3)**.

40.3.1.2.1 Mapping of interrupts

The interrupt assignment from the interrupt register, only forwarding the enabled interrupt sources is as following:

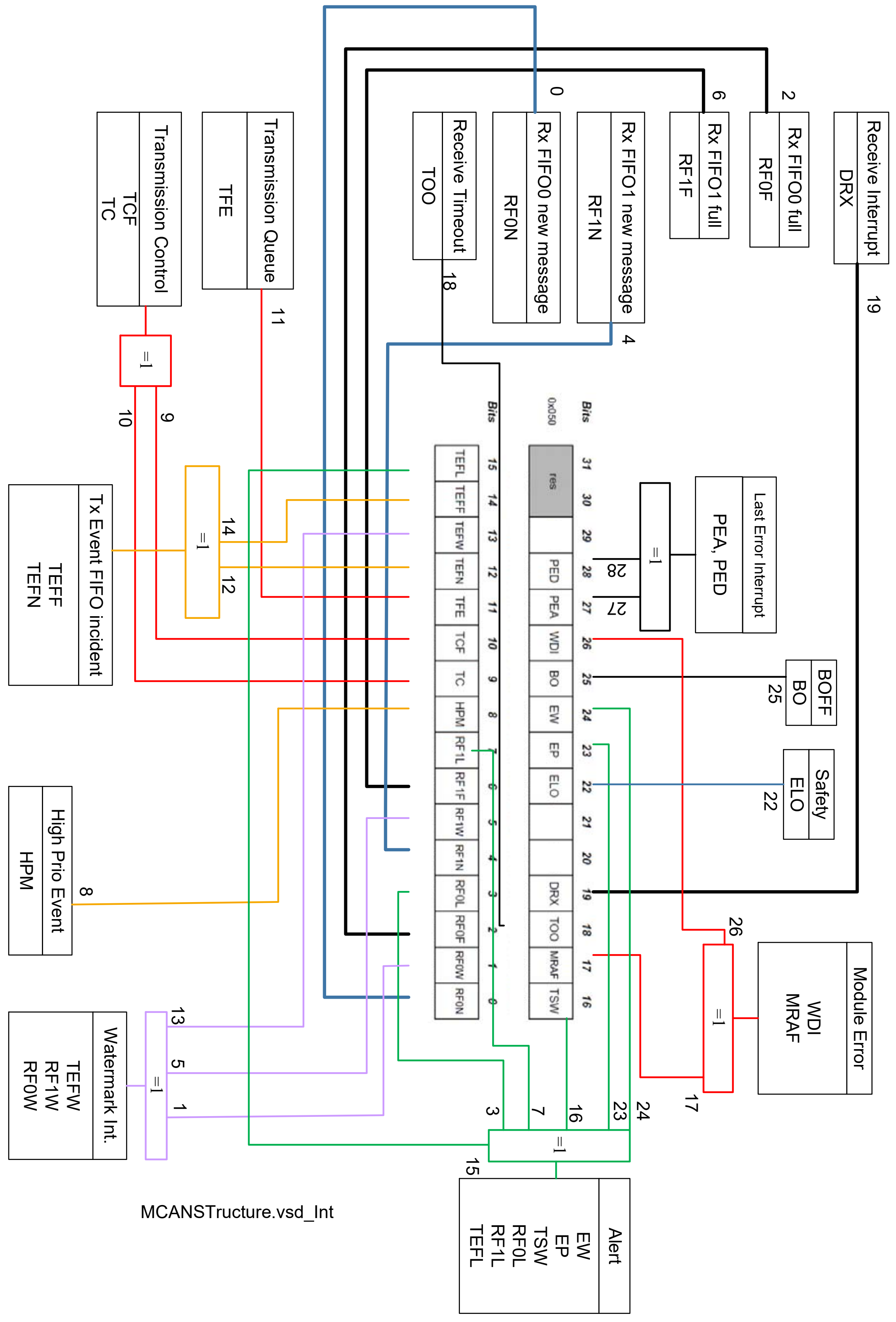


Figure 583 Mapping of interrupts into groups (without TTCAN)

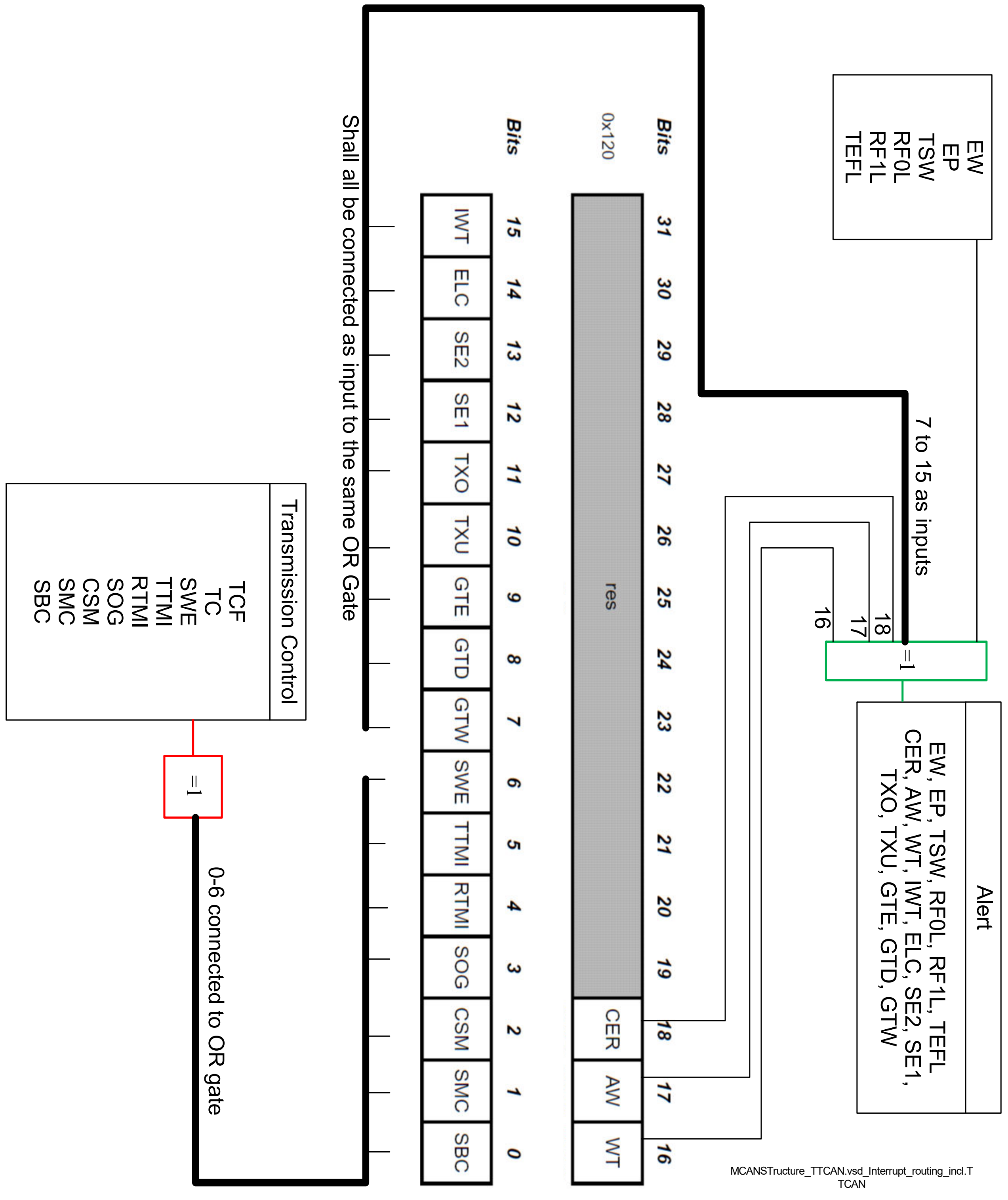


Figure 584 Mapping of TTCAN interrupts into groups

40.3.1.2.2 Signalling interrupts of groups

The groups defined in the previous paragraph are also shown in an **Interrupt Signalling Register i**. 0 means, that no interrupt is pending on the corresponding interrupt, 1 means pending.

40.3.1.2.3 Connections to Interrupt Router Inputs

The interrupt output line INT_O0-15 of MCMCAN is connected to the Interrupt Router module, see **Table 355**.

Table 355 Interrupt Router Inputs

Interrupt Router Input	Connected to CAN Interrupt Output
SRC_CANzINT0	INT_O0 MCMCAN
SRC_CANzINT1	INT_O1 MCMCAN
SRC_CANzINT2	INT_O2 MCMCAN
SRC_CANzINT3	INT_O3 MCMCAN
SRC_CANzINT4	INT_O4 MCMCAN
SRC_CANzINT5	INT_O5 MCMCAN
SRC_CANzINT6	INT_O6 MCMCAN
SRC_CANzINT7	INT_O7 MCMCAN
SRC_CANzINT8	INT_O8 MCMCAN
SRC_CANzINT9	INT_O9 MCMCAN
SRC_CANzINT10	INT_O10 MCMCAN
SRC_CANzINT11	INT_O11 MCMCAN
SRC_CANzINT12	INT_O12 MCMCAN
SRC_CANzINT13	INT_O13 MCMCAN
SRC_CANzINT14	INT_O14 MCMCAN
SRC_CANzINT15	INT_O15 MCMCAN

Interrupt Control

The general interrupt structure is shown in **Figure 585**. The interrupt event can trigger the interrupt generation. The interrupt pulse is generated based on the interrupt flag in the interrupt (status) register (**IRi (i=0-3)** and **TTIR0**) and the interrupt enable bit in Interrupt Enable register (**IEi (i=0-3)**and **TTIE0**). It is purely based on AND logic between the interrupt flags and interrupt enable bit fields. The interrupt flag can be reset by software by writing a '1' to the CANn_IRi bit.

If enabled by the related interrupt enable bit in the corresponding interrupt enable register (**IEi (i=0-3)**, **NTRTRi (i=0-3)**.TEIE and **TTIE0**), an interrupt pulse can be generated at one of the 16 interrupt output lines INT_On of the MCMCAN module using **GRINT1i (i=0-3)** and **GRINT2i (i=0-3)**. If more than one interrupt source is connected to the same interrupt node (in **GRINT1i (i=0-3)** and **GRINT2i (i=0-3)**), the requests are combined to one common line.

The interrupt groups are only ORing the interrupts of the corresponding CAN nodes. The interrupt request has to be reset in the node.

Note: Enabling an interrupt in Interrupt Enable register when the corresponding flag is already set in Interrupt Register will also generate an interrupt pulse. Hence it is recommended to clear the interrupt flags before enabling the corresponding interrupts.

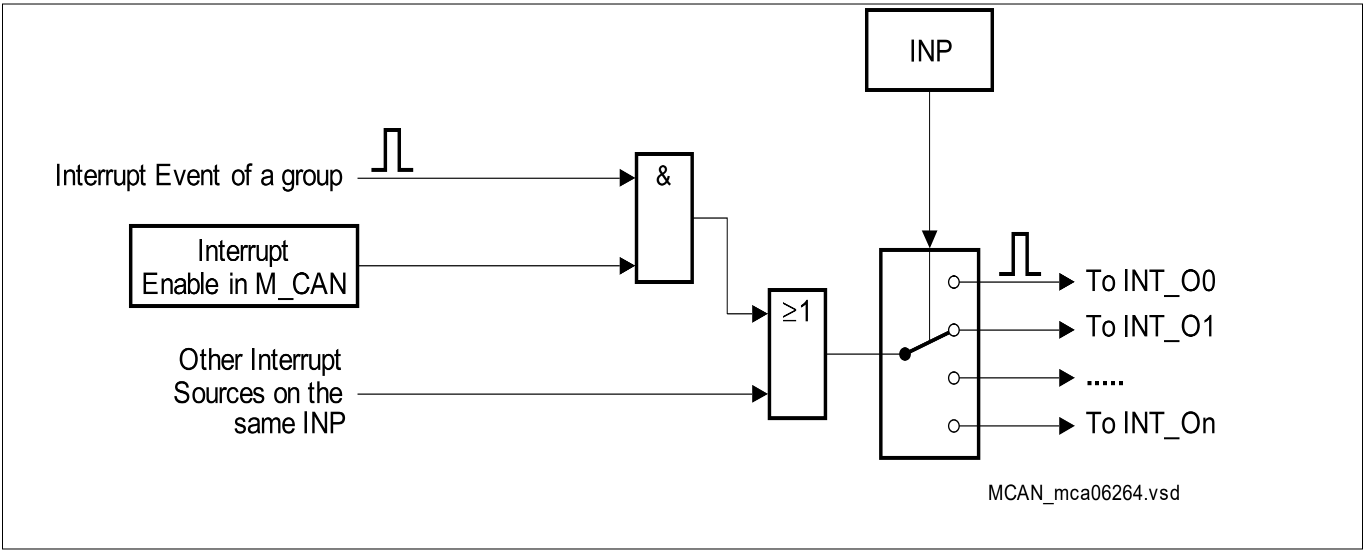

INP
Interrupt Event of a group
&
Interrupt Enable in M_CAN
≥1
Other Interrupt Sources on the same INP
To INT_O0
To INT_O1
.....
To INT_On
MCAN_mca06264.vsd

Figure 585 General Interrupt Structure

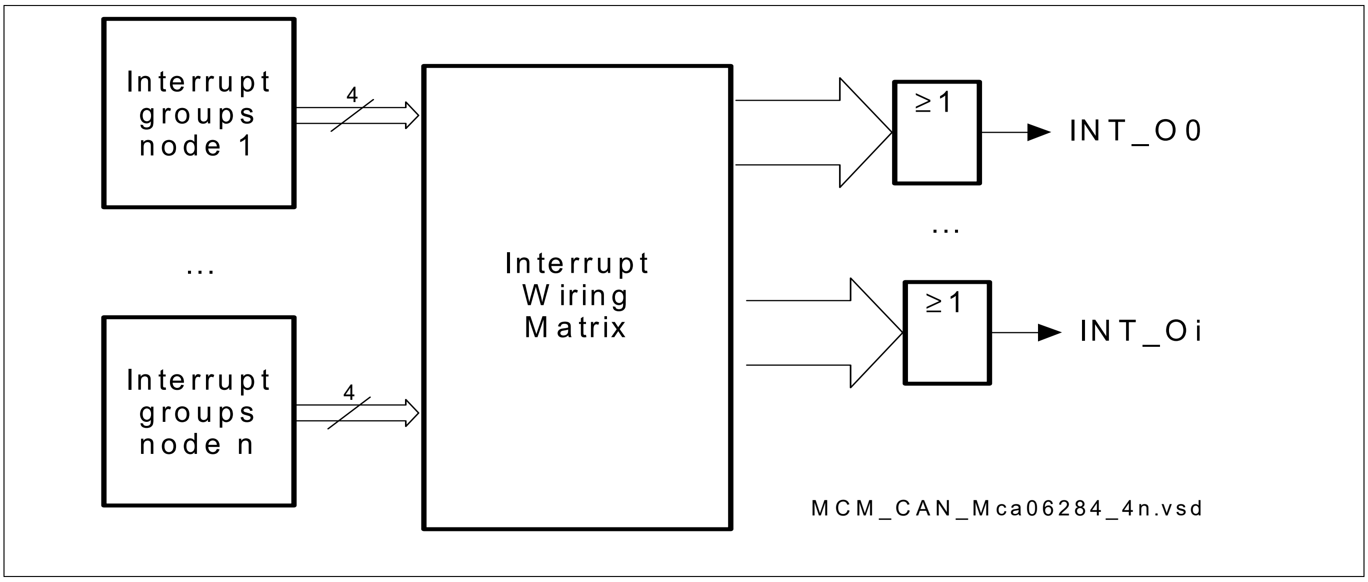


Figure 586 Interrupt compression Unit

40.3.1.3 Port, I/O Line Control and Interconnects

The interconnections between the MCMCAN module and the port I/O lines are controlled in the port logic. Additionally to the port input selection, the following port control operations must be executed:

- Input/output function selection (IOCR registers)
- Pad driver characteristics selection for the outputs (PDR registers)
- This chapter describes all connections, which are in relation

40.3.1.3.1 Input/Output Function Selection in Ports

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the MCMCAN module are controlled by the port input/output control registers, which are described in the port chapter.

Table 356 shows the corresponding pins, sorted by node. Even though the table is pair wise, it is possible to select a different pairing for RXD and TXD. In addition the RXSEL value to be programmed for the Receive Pins is part of this table. For more information on RXSEL please see **Node Receive Input Selection**.

Note: It is recommended to use "Fast Pads with Strong driver, medium/sharp edge" configuration when CAN-FD is used."

Table 356 MCMCAN I/O Control Selection and Setup

Node	RXD	Receive Port	NPCRx.RXSEL	Transmit Port
CAN00	CAN00_RXDA	P2.1	000_B	P2.0
	CAN00_RXDB	P20.7	001_B	P20.8
	CAN00_RXDC	P12.0	010_B	P12.1
	CAN00_RXDD	P33.12	011_B	P33.13
	CAN00_RXDE	P33.7	100_B	P33.8
	CAN00_RXDF	P1.8	101_B	P1.13
	CAN00_RXDG	P34.2	110_B	P34.1
	CAN00_RXDH	P2.14	111_B	P2.13
CAN01	CAN01_RXDA	P15.3	000_B	P15.2
	CAN01_RXDB	P14.1	001_B	P14.0
	CAN01_RXDC	P1.4	010_B	P1.3
	CAN01_RXDD	P33.10	011_B	P33.9
	CAN01_RXDE	P2.10	100_B	P2.9
	CAN01_RXDF	not used	101_B	
	CAN01_RXDG	not used	110_B	
	CAN01_RXDH	not used	111_B	
CAN02	CAN02_RXDA	P15.1	000_B	P15.0
	CAN02_RXDB	P2.3	001_B	P2.2
	CAN02_RXDC	P32.6	010_B	P32.5
	CAN02_RXDD	P14.8	011_B	P14.10
	CAN02_RXDE	P10.2	100_B	P10.3
	CAN02_RXDF	not used	101_B	
	CAN02_RXDG	not used	110_B	
	CAN02_RXDH	not used	111_B	
CAN03	CAN03_RXDA	P0.3	000_B	P0.2
	CAN03_RXDB	P32.2	001_B	P32.3
	CAN03_RXDC	P20.0	010_B	P20.3
	CAN03_RXDD	P11.10	011_B	P11.12
	CAN03_RXDE	P20.9	100_B	P20.10
	CAN03_RXDF	P1.0	101_B	P1.2
	CAN03_RXDG	not used	110_B	
	CAN03_RXDH	not used	111_B	

Table 356 MCMCAN I/O Control Selection and Setup (cont'd)

Node	RXD	Receive Port	NPCRx.RXSEL	Transmit Port
CAN10	CAN10_RXDA	P0.1	000_B	P0.0
	CAN10_RXDB	P14.7	001_B	P14.9
	CAN10_RXDC	P23.0	010_B	P23.1
	CAN10_RXDD	P13.1	011_B	P13.0
	CAN10_RXDE	not used	100_B	
	CAN10_RXDF	not used	101_B	
	CAN10_RXDG	not used	110_B	
	CAN10_RXDH	not used	111_B	
CAN11	CAN11_RXDA	P2.4	000_B	P2.5
	CAN11_RXDB	P0.5	001_B	P0.4
	CAN11_RXDC	P23.7	010_B	P23.6
	CAN11_RXDD	P11.7	011_B	P11.0
	CAN11_RXDE	not used	100_B	
	CAN11_RXDF	not used	101_B	
	CAN11_RXDG	not used	110_B	
	CAN11_RXDH	not used	111_B	
CAN12	CAN12_RXDA	P20.6	000_B	P20.7
	CAN12_RXDB	P10.8	001_B	P10.7
	CAN12_RXDC	P23.3	010_B	P23.2
	CAN12_RXDD	P11.8	011_B	P11.1
	CAN12_RXDE	not used	100_B	
	CAN12_RXDF	not used	101_B	
	CAN12_RXDG	not used	110_B	
	CAN12_RXDH	not used	111_B	
CAN13	CAN13_RXDA	P14.7	000_B	P14.6
	CAN13_RXDB	P33.5	001_B	P33.4
	CAN13_RXDC	P22.5	010_B	P22.4
	CAN13_RXDD	P11.13	011_B	P11.4
	CAN13_RXDE	not used	100_B	
	CAN13_RXDF	not used	101_B	
	CAN13_RXDG	not used	110_B	
	CAN13_RXDH	not used	111_B	
CAN20	CAN20_RXDA	P10.5	000_B	P10.6
	CAN20_RXDB	P10.8	001_B	P10.7
	CAN20_RXDC	P34.2	010_B	P34.1
	CAN20_RXDD	P2.14	011_B	P2.13
	CAN20_RXDE	P1.8	100_B	P1.13
	CAN20_RXDF	P11.14	101_B	P11.5

Table 356 MCMCAN I/O Control Selection and Setup (cont'd)

Node	RXD	Receive Port	NPCRx.RXSEL	Transmit Port
	CAN20_RXDG	not used	110_B	
	CAN20_RXDH	not used	111_B	
CAN21	CAN21_RXDA	P0.3	000_B	P0.2
	CAN21_RXDB	P13.12	001_B	P13.9
	CAN21_RXDC	P20.0	010_B	P20.3
	CAN21_RXDD	P32.2	011_B	P32.3
	CAN21_RXDE	P1.0	100_B	P1.2
	CAN21_RXDF	P22.7	101_B	P22.6
	CAN21_RXDG	not used	110_B	
	CAN21_RXDH	not used	111_B	
CAN22	CAN22_RXDA	P33.13	000_B	P33.12
	CAN22_RXDB	P32.7	001_B	P32.6
	CAN22_RXDC	P23.6	010_B	P23.5
	CAN22_RXDD	P14.14	011_B	P14.13
	CAN22_RXDE	P22.9	100_B	P22.8
	CAN22_RXDF	not used	101_B	
	CAN22_RXDG	not used	110_B	
	CAN22_RXDH	not used	111_B	
CAN23	CAN23_RXDA	P14.10	000_B	P14.9
	CAN23_RXDB	P23.3	001_B	P23.2
	CAN23_RXDC	P14.15	010_B	P14.14
	CAN23_RXDD	P13.5	011_B	P13.4
	CAN23_RXDE	P22.11	100_B	P22.10
	CAN23_RXDF	not used	101_B	
	CAN23_RXDG	not used	110_B	
	CAN23_RXDH	not used	111_B	

Node Receive Input Selection

Additionally to the I/O control selection, as defined in the port chapter, the selection of a CAN node's receive input line requires that bit field RXSEL in its node port control register **NPCRi (i=0-3)** must be set according to **Table 356**. Values for **NPCRi (i=0-3)**.RXSEL other than those of the table mentioned above will result in a recessive receive input for node x. As a hint for the RXSEL values, a receive pin ending with A results in 0x0, B in 0x1 until H resulting in the value of 0x7.

40.3.1.3.2 Trigger inputs

The following paragraph describes all interconnections, which are relevant for triggering events.

40.3.1.3.3 External CAN Time Trigger Inputs

The external CAN time trigger inputs ECTT[8:1] of MCMCANcan be used as a transmit trigger for a reference message. In the AURIX™ TC3xx Platform, these input lines are connected as shown in **Table 357**.

Table 357 External CAN Time Trigger Inputs

Multiplexer Input	Connected to	From / to Module
ECTT1	P2.4 / ECTT1	Port 2
ECTT2	P2.5 / ECTT2	Port 2
ECTT3	Output IOUT2	External Request Unit (SCU)
ECTT4	Output IOUT3	External Request Unit (SCU)
ECTT5	eray.eray_tint0_o	Timer Service Request 0 (E-Ray)
ECTT6	eray.eray_tint1_o	Timer Service Request 1 (E-Ray)
ECTT7	gtm_0.sx_gtm2mcan_mcan0.0	GTM TTCAN Trigger 0
ECTT8	gtm_0.sx_gtm2mcan_mcan0.1	GTM TTCAN Trigger 1

40.3.1.3.4 CAN Transmit Trigger Inputs

The CAN transmit trigger inputs of MCMCAN are used to trigger for transmission of a message. In the AURIX™ TC3xx Platform, these input lines are connected as shown in **Table 358**. **Table 358** is CAN0 only, for CAN1 and following the mcan0 has to be exchanged to mcan1 and following.

TC39xA only: The cross connections from GTM are identical for CAN1 and CAN2.

Table 358 CAN Transmit Trigger Inputs

Receive Input	Connected to	From / to Module
CAN Node 0		
STM_N0_TRIG	STM0.SR0_INT	System Timer Module (STM)
GTM_N0_TRIG	gtm_0.sx_gtm2mcan_mcan0.0	General Timer Module (GTM)
…		
…	…	…
CAN Node 3		

Table 358 CAN Transmit Trigger Inputs (cont'd)

Receive Input	Connected to	From / to Module
STM_N3_TRIG	STM0.SR0_INT	System Timer Module (STM)
GTM_N3_TRIG	gtm_0.sx_gtm2mcan_mcan0.3	General Timer Module (GTM)

40.3.1.3.5 TT Capture Time Trigger Input

The CAN local time register capture input of MCMCAN is used to trigger the capture of the TT Cycle Time & Count (TTCTC) to the TT Capture Time (TTCPT). In the AURIX™ TC3xx Platform, the input lines are connected as shown in **Table 359** and can be selected using **TTCR0**.TTCTSS.

Table 359 TT Capture Time Register Trigger Input

Receive Input	Connected to	From / to Module
TTCPT_TRIG1	STM0.SR0_INT	System Timer Module (STM)
TTCPT_TRIG2	STM1.SR0_INT	System Timer Module (STM)
TTCPT_TRIG3	STM2.SR0_INT	System Timer Module (STM)
TTCPT_TRIG4	Output IOUT4	External Request Unit (SCU)

Transmit Trigger for a Reference Message

The transmission of a reference message can be triggered by a time mark or by an external event. If it is triggered by a time mark, the "Next is Gap" bit (TTOCN0.NIG for receiving **TTOST0**.WFE) of the previous reference message has been 0. If it is triggered by an external event, the "Next is Gap" bit of the previous reference message has been 1.

The software can set bit **TTOCN0**.NIG in order to initiate the synchronization of a reference message to an event. This event can be an edge at an external input or a software action (write bit **TTOCN0**.FGP = 1, software trigger event).

The transmit trigger generation logic for the reference message is shown in **Figure 587**. The edge detection for ECTTx contains a synchronization stage.

The trigger event can be selected by bit field **TTCR0**.ETESEL (external trigger event selection). Only trigger events are taken into account for the transfer of a reference message that have been detected after the start of the current basic cycle.

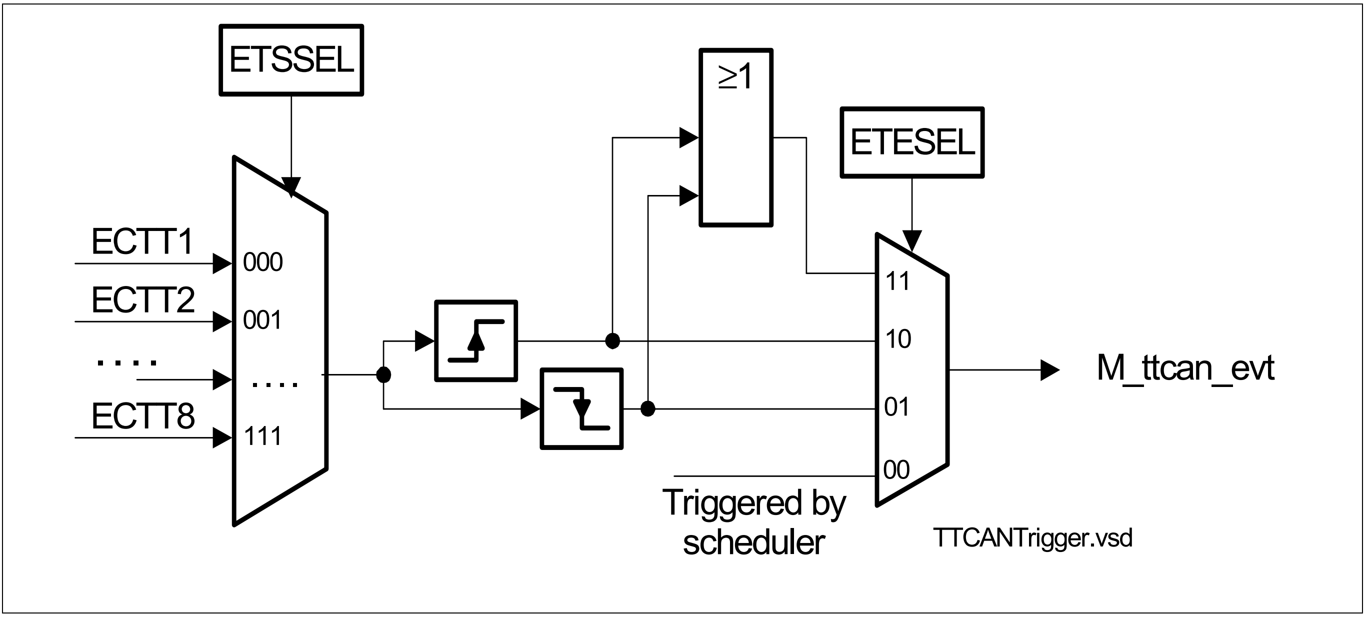


Figure 587 External Trigger Generation and Transmit Trigger for the Reference Message

40.3.1.3.6 Signals to other modules

GTM: Interrupt 12 to 15 are connected to GTM inputs.

Connections to General Timer Module (GTM) Inputs

The interrupt output line INT_O12-15 of MCMCAN is connected to the General Timer Module, see **Table 360**.

Table 360 General Timer Module Inputs

GTM Input	Connected to CAN Interrupt Output
Timer input (gtm.tim_0_muxin_1_i(13))	INT_O12 (mcan0.int_req_o(12))
Timer input (gtm.tim_1_muxin_1_i(13))	
Timer input (gtm.tim_2_muxin_1_i(13))	
Timer input (gtm.tim_0_muxin_2_i(13))	INT_O13 (mcan0.int_req_o(13))
Timer input (gtm.tim_1_muxin_2_i(13))	
Timer input (gtm.tim_2_muxin_2_i(13))	
Timer input (gtm.tim_0_muxin_3_i(13))	INT_O14 (mcan0.int_req_o(14))
Timer input (gtm.tim_1_muxin_3_i(13))	
Timer input (gtm.tim_2_muxin_3_i(13))	
Timer input (gtm.tim_0_muxin_4_i(13))	INT_O15 (mcan0.int_req_o(15))
Timer input (gtm.tim_1_muxin_4_i(13))	
Timer input (gtm.tim_2_muxin_4_i(13))	

Table 361 TIM0 Mapping

CH1SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[12]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH2SEL	**Pad / Input**	**Name**	Packages/[Products]

Table 361 TIM0 Mapping (cont'd)

1101_B	can0_int[13]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH3SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[14]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH4SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[15]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH5SEL	**Pad / Input**	**Name**	Packages/[Products]

Table 362 TIM1 Mapping

CH1SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[12]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH2SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[13]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH3SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[14]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH4SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[15]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]

Table 363 TIM2 Mapping

CH1SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[12]	CAN	QFP144,176,BGA292,516/ [TC39x,38x,37x,36x]
CH2SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[13]	CAN	QFP144,176,BGA292,516/ [TC39x,38x,37x,36x]
CH3SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[14]	CAN	QFP144,176,BGA292,516/ [TC39x,38x,37x,36x]
CH4SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[15]	CAN	QFP144,176,BGA292,516/ [TC39x,38x,37x,36x]

Table 364 TIM3 Mapping

CH1SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[12]	CAN	QFP176,BGA292,BGA516/[TC39x/38x/37x]
CH2SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[13]	CAN	QFP176,BGA292,BGA516/[TC39x/38x/37x]
CH3SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[14]	CAN	QFP176,BGA292,BGA516/[TC39x/38x/37x]
CH4SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[15]	CAN	

40.3.1.4 Connecting the module to the outside world

The MCMCAN module can be connected per node either to a port configuring the **NPCRi (i=0-3)**.RXSEL for the input, or can be connected to an module internal CAN bus.

40.3.1.4.1 Module internal Loop-Back Mode

The MCMCAN module provides a Module internal Loop-Back Mode to enable an in-system test of the MCMCAN module as well as the development of CAN driver software without access to an external CAN bus.

The loop-back feature consists of an internal CAN bus (inside the MCMCAN module) and a bus select switch for each CAN node. With the switch, each CAN node can be connected either to the internal CAN bus (Internal Loop-Back Mode activated) or the external CAN bus, respectively to transmit and receive pins (normal operation). The CAN bus that is not currently selected is driven recessive; this means the transmit pin is held at 1, and the receive pin is ignored by the CAN nodes that are in Loop-Back Mode.

The Internal Loop-Back Mode is selected for CAN node x by setting the Node x Port Control Register bit **NPCRi (i=0-3)**.LBM. All CAN nodes that are in Loop-Back Mode may communicate together via the internal CAN bus without affecting the normal operation of the other CAN nodes that are not in Loop-Back Mode.

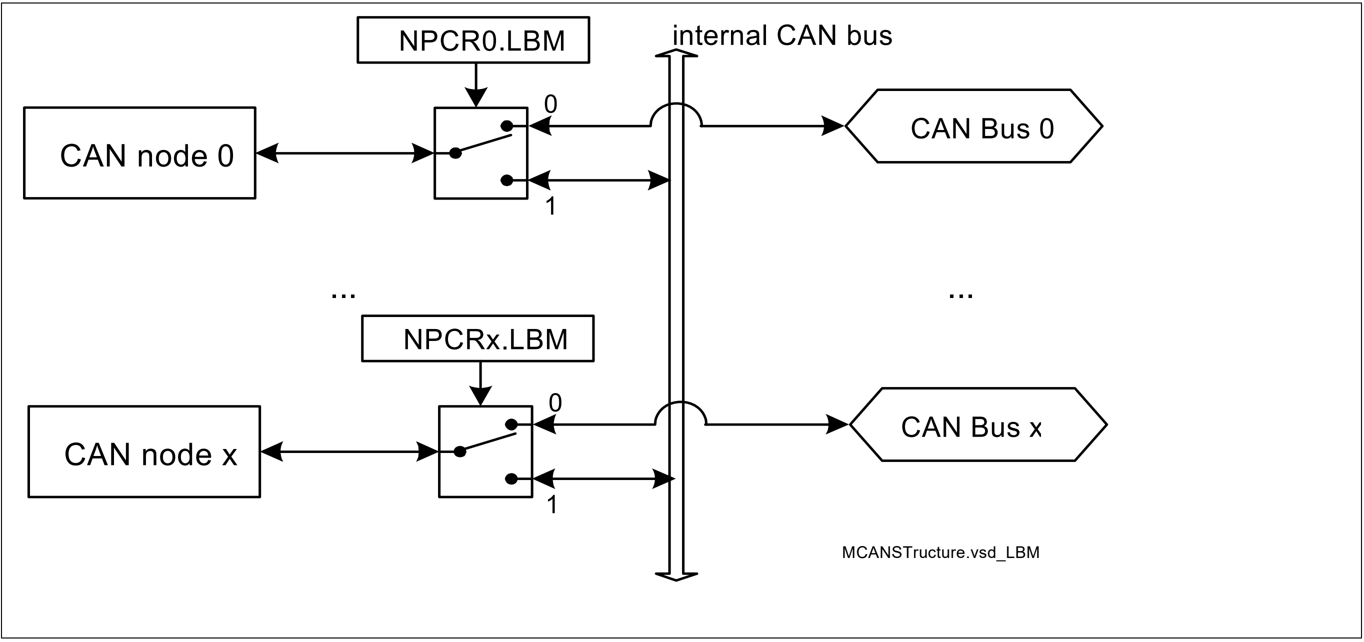


Figure 588 Module Internal Loop-Back Mode

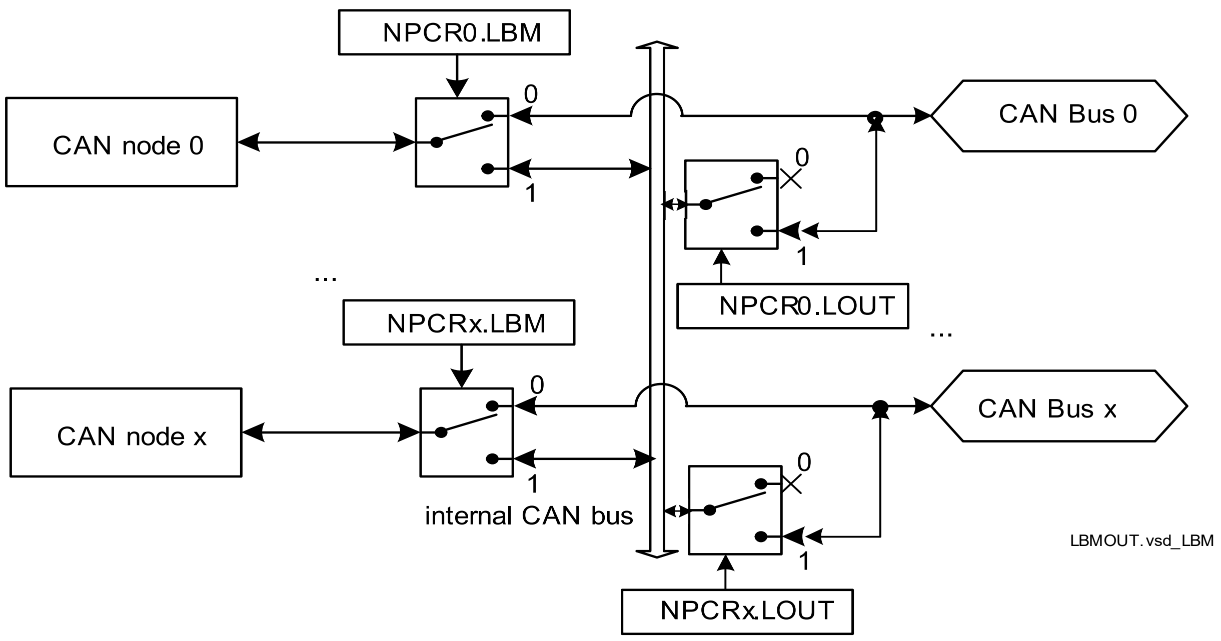


Figure 589 Module Loop-Back Mode Out

40.3.1.4.2 Module Loop Back Mode Out (B-step feature)

Setting **NPCRi (i=0-3)**.LOUT bit will send the signals from the internal loop back bus to the corresponding pins of the CAN node and vice versa. Therefore a receive and transmit between internal loop back bus and a bus system outside is possible. When **NPCRi (i=0-3)**.LOUT is set to one, the internal loop back bus and the external bus will be communicating with each other. The table below explains the behaviour of the CAN module for different combinations **NPCRi (i=0-3)**.LOUT and **NPCRi (i=0-3)**.LBM.

Table 365 Combinations of Internal Loop Back Mode and Loop Back Mode Out

NPCR.LOUT	NPCR.LBM	Description
0	0	Normal CAN operation. Internal and loop back out mode deactivated
0	1	Internal Loop Back mode
1	0	Loop Back Mode Out
1	1	Reserved

40.3.1.5 Fixing the address protection for the CAN nodes

Inside the BPI part, the address protection is defined. Here the nodes become protectable, as well as the control area. With the nodes a certain RAM area is assigned, therefore a start address and an end address per node is necessary. An overlap of the protected areas, is not found by the hardware. **STARTADRi (i=0-3)** and **ENDADRi (i=0-3)** registers, give the possibility to define a range within CAN RAM, belonging to a node, which can only be written by certain masters. The mechanism is identical to ACCEN registers. The RAM will only be write and not read protected.

40.3.1.6 Node Timing Functions

A CAN node offers the following timing functions:

- A receive time-out mode that can detect the reception within message buffers. The timeout expires, when no message has been received.
- Without CPU involvement, a selectable message buffers can be transmitted periodically, triggered by a timer, a System Timer (STM) or General Timer Module (GTM).

The clocking options for the node timer is controlled by Node Timer Clock Control Register, **Node i Timer Clock Control Register**, the node timing functions for Receive Timeout Mode is controlled by Node x Timer Receive Timeout Register, **Node i Timer Receive Timeout Register** and for Transmit Trigger Mode is controlled by Node x Timer A/B/C Transmit Trigger Registers, **Node i Timer A Transmit Trigger Register**, **Node i Timer B Transmit Trigger Register**, **Node i Timer C Transmit Trigger Register**. Links are only for CAN0, but the feature is available for all CAN modules. In general, the timers will not run, without setting the START bit and not without setting the RELOAD value, to a different value than 0. Whatever comes last, will start the timer.

Modes with Timer Usage

A CAN node timer is driven by the synchronous clock or by the corresponding clock sources, divided by a prescaler selected via bit field TPSC in the corresponding timer control register. The timers are enabled by writing to the RELOAD bits in the relevant node timer registers; then it decrements from its initial value. The further behavior depends on the selected timer mode:

- **Receive Timeout Mode:**The receive timeout function is valid for the receive buffers. A receive time-out check is enabled, which may be a received frame or remote data frame. If any of the message buffers receives before the timer is 0, the timer will be reloaded. When the timer reaches 0, it will stop. Bit TE in **NTRTRi (i=0-3)** register will be set. With bit **NTRTRi (i=0-3)**.TEIE = 1, an interrupt will be generated.
- **Transmit Trigger Mode:**When the timer reaches 0, the TXRQ of the corresponding message buffer is selected.

Transmit Trigger by System Timer or General Timer Module

For Node i Timer the trigger for transmission of a message can also be set by a System Timer (STM) trigger event or General Timer Module (GTM) trigger event. See **Figure 590**.
Bit **NTCCRi (i=0-3)**.TRIGSRC in the timer clock control register enables this feature and the timer is started once values are written to the RELOAD bits of the relevant **NTATTRi (i=0-3)**, **NTBTTRi (i=0-3)**, or **NTCTTRi (i=0-3)** registers. In transmit trigger mode, when a trigger event occurs (STM or GTM), the node timer will be decremented per trigger event timing prescaled by (TPSC+1) till it reaches zero where the transmit request is set for the corresponding transmit message buffer.

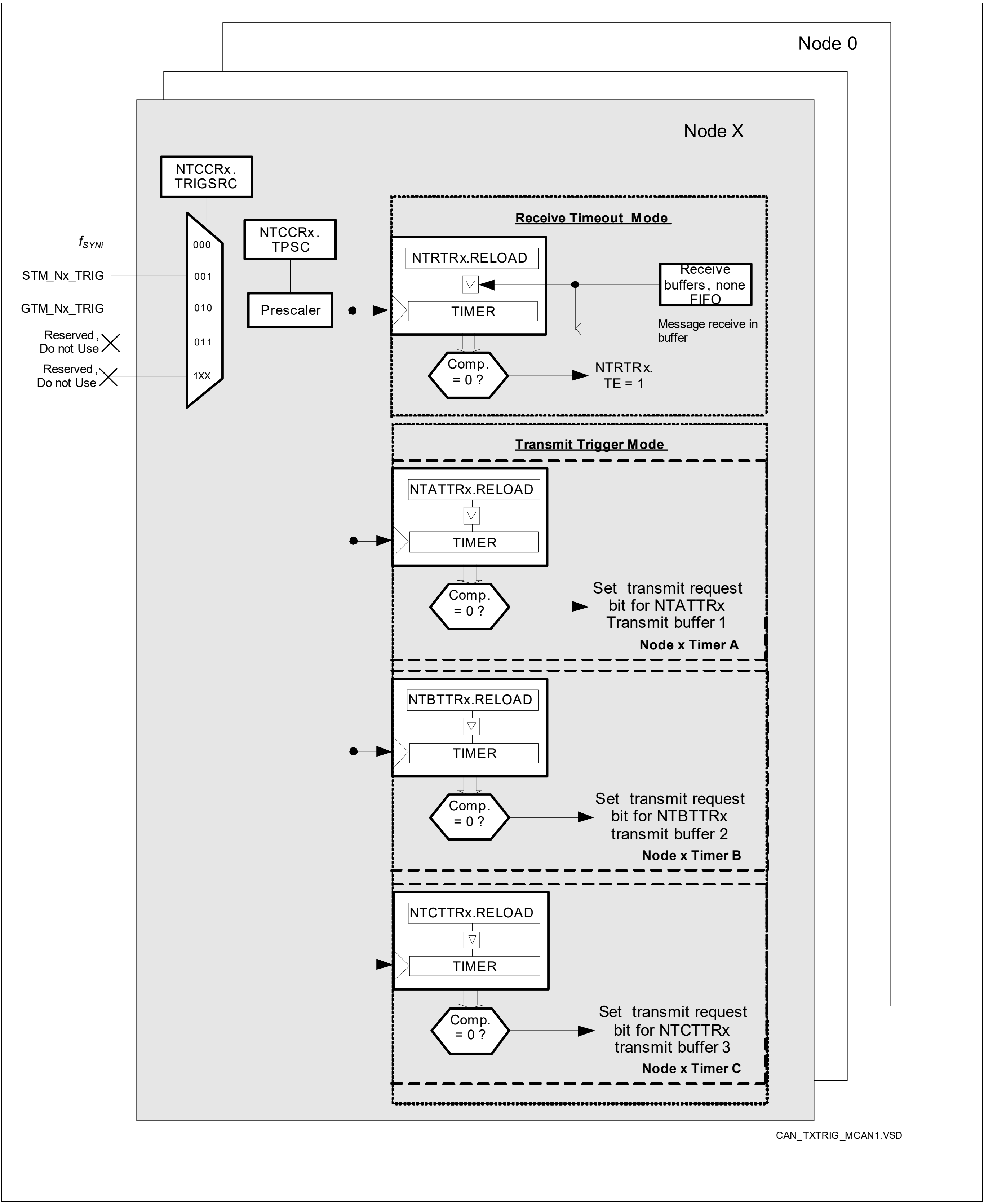


Figure 590 CAN Node Timing Modes

40.3.1.6.1 Automatic transferring of messages

NTATTRi (i=0-3), **NTBTTRi (i=0-3)**, **NTCTTRi (i=0-3)** registers enable the M_CAN IP to trigger messages on timer events. The timer per module have one clock source. The counting events can take place on an STM interrupt, a GTM interrupt or by f_{SYNi}. These timers can be used to enable Pretended Networking, or to have hardware support for a gateway functionality. This functionality needs a receive interrupt or a watermark interrupt within a FIFO, triggering a DMA transfer. The transmit objects will be automatically triggered to transfer the messages below, via timer underflow. As the transmit triggers are fixed to message RAM buffers 1-3, this feature only works if these buffers are available.

40.3.1.7 Destructive Debug Entry

On destructive debug entry ie., when DMU_SP_PROCONHSMCFG.DESTDBG = 11b and DMU_HF_PROCONDBG.EDM = 11b, the CAN transmit data output (TXD) is tied to “Recessive state” (logic ‘1b’). This blocks any further CAN transmission when AURIX™ TC3xx Platform has entered the Destructive Debug state.

40.3.2 M_CAN Functional Description

40.3.2.1 Operating Modes

40.3.2.1.1 Software Initialization

Software initialization is started by setting bit CCCRi.INIT, either by software or by a hardware reset, or by going Bus_Off. While **CCCRi (i=0-3)**.INIT is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output TXD is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting **CCCRi (i=0-3)**.INIT does not change any configuration register. Resetting **CCCRi (i=0-3)**.INIT finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_CAN configuration registers is only enabled when both bits **CCCRi (i=0-3)**.INIT and **CCCRi (i=0-3)**.CCE are set (protected write).

CCCRi (i=0-3).CCE can only be set/reset while **CCCRi (i=0-3)**.INIT = ‘1’. **CCCRi (i=0-3)**.CCE is automatically reset, when CCCRi.INIT is cleared.

The following registers are reset when **CCCRi (i=0-3)**.CCE is set

- **HPMSi (i=0-3)** - High Priority Message Status
- **RXF0Si (i=0-3)** - Rx FIFO 0 Status
- **RXF1Si (i=0-3)** - Rx FIFO 1 Status
- **TXFQSi (i=0-3)** - Tx FIFO/Queue Status
- **TXBRPi (i=0-3)** - Tx Buffer Request Pending
- **TXBTOi (i=0-3)** - Tx Buffer Transmission Occurred
- **TXBCFi (i=0-3)** - Tx Buffer Cancellation Finished
- **TXEFSi (i=0-3)** - Tx Event FIFO Status
- **TTOST0** - TT Operation Status
- **TTLGT0** - TT Local & Global Time, only Global Time TTLGT0.GT is reset
- **TTCTC0** - TT Cycle Time & Count
- **TTCSM0** - TT Cycle Sync Mark

The Timeout Counter value **TOCVi (i=0-3)**.TOC is preset to the value configured by **TOCCi (i=0-3)**.TOP when **CCCRi (i=0-3)**.CCE is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while **CCCRi (i=0-3)**.CCE = ‘1’.

The following registers are only writeable while **CCCRi (i=0-3)**.CCE = ‘0’

- **TXBARi (i=0-3)** - Tx Buffer Add Request
- **TXBCRi (i=0-3)** - Tx Buffer Cancellation Request

CCCRi (i=0-3).TEST and **CCCRi (i=0-3)**.MON can only be set by the Host while **CCCRi (i=0-3)**.INIT = ‘1’ and **CCCRi (i=0-3)**.CCE = ‘1’. Both bits may be reset at any time. **CCCRi (i=0-3)**.DAR can only be set/reset while **CCCRi (i=0-3)**.INIT = ‘1’ and **CCCRi (i=0-3)**.CCE = ‘1’.

40.3.2.1.2 Normal Operation

The M_CAN's default operating mode after hardware reset is event-driven CAN communication without time triggers (TTOCF0.OM = "00"). It is required that both **CCCRi (i=0-3)**.INIT and **CCCRi (i=0-3)**.CCE are set before the TT Operation Mode can be changed.

Once the M_CAN is initialized and **CCCRi (i=0-3)**.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.

40.3.2.1.3 CAN FD Operation

There are two variants in the CAN FD frame transmission, first the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame. The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = recessive signifies a CAN FD frame, FDF = dominant signifies a Classical CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. The coding of res = recessive is reserved for future expansion of the protocol. In case the M_CAN receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event by setting bit **PSRi (i=0-3)**.PXE. When Protocol Exception Handling is enabled (**CCCRi (i=0-3)**.PXHD = '0'), this causes the operation state to change from Receiver (**PSRi (i=0-3)**.ACT = "10") to Integrating (**PSRi (i=0-3)**.ACT = "00") at the next sample point. In case Protocol Exception Handling is disabled (**CCCRi (i=0-3)**.PXHD = '1'), the M_CAN will treat a recessive res bit as an form error and will respond with an error frame.CAN FD operation is enabled by programming **CCCRi (i=0-3)**.FDOE. In case **CCCRi (i=0-3)**.FDOE = '1', transmission and reception of CAN FD frames is enabled. Transmission and reception of Classical CAN frames is always possible. Whether a CAN FD frame or a Classical CAN frame is transmitted can be configured via bit FDF in the respective Tx Buffer element. With **CCCRi (i=0-3)**.FDOE = '0', received frames are interpreted as Classical CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx Buffer element is set. CCCRi.FDOE and CCCRi.BRSE can only be changed while CCCRi.INIT and CCCRi.CCE are both set.With CCCRi.FDOE = '0', the setting of bits FDF and BRS is ignored and frames are transmitted in Classical CAN format. With **CCCRi (i=0-3)**.FDOE = '1' and **CCCRi (i=0-3)**.BRSE = '0', only bit FDF of a Tx Buffer element is evaluated. With **CCCRi (i=0-3)**.FDOE = '1' and **CCCRi (i=0-3)**.BRSE = '1', transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting Classical CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN Partial Networking have to be transmitted in Classical CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to Classical CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to **Table 366** below.

Table 366 Coding of DLC in CAN FD

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

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Nominal Bit Timing & Prescaler Register NBTPi. In the following CAN FD data phase, the data phase bit timing is used as defined by the Data Bit Timing & Prescaler Register **DBTPi (i=0-3)**. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency (asynchronous clock). Example: with a CAN clock frequency of 20 MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s. (As 4 tq is quite tight, please check if this works in your environment.)

Note: MCMCAN supports upto 5 Mbit/s considering the physical medium CAN-FD timing requirements from ISO 11898-2:2016. At f_{MCAN} = 80MHz, CAN-FD data phase bit rate can be extended to 8 Mbit/s, while the bit asymmetry effect from CAN-FD transceiver, physical layer network topology etc., has to be considered by the user.

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.

40.3.2.1.4 Transmitter Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin TXD the protocol controller receives the transmitted data from its local CAN transceiver via pin RXD. The received data is delayed by the CAN transmitter delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transmitter delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transmitter delay.

Description

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

For the transmitter delay compensation the following boundary conditions have to be considered:

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

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit **DBTPi (i=0-3)**.TDC.The received bit is compared against the transmitted bit at the Secondary Sample Point. The Secondary Sample Point position is defined as the sum of the measured delay from the M_CAN's transmit output TX through the transceiver to the receive input RX plus the transmitter delay compensation offset as configured by **TDCRi (i=0-3)**.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq.PSRi.TDCV shows the actual transmitter delay compensation value. **PSRi (i=0-3)**.TDCV is cleared when CCCRi.INIT is set and is updated at each transmission of an FD frame while **DBTPi (i=0-3)**.TDC is set.

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

- The sum of the measured delay from TX to RX and the configured transmitter delay compensation offset **TDCRi (i=0-3)**.TDCO has to be less than 6 bit times in the data phase.The sum of the measured delay from RX to RX and the configured transmitter delay compensation offset TDCRi.TDCO has to be less or equal 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transmitter delay compensation.The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the Secondary Sample Points.

Transmitter Delay Compensation Measurement

If transmitter delay compensation is enabled by programming **DBTPi (i=0-3)**.TDC = '1', the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input RX of the transmitter. The resolution of this measurement is one mtq.

Figure 591 below describes how the transmitter loop delay is measured.

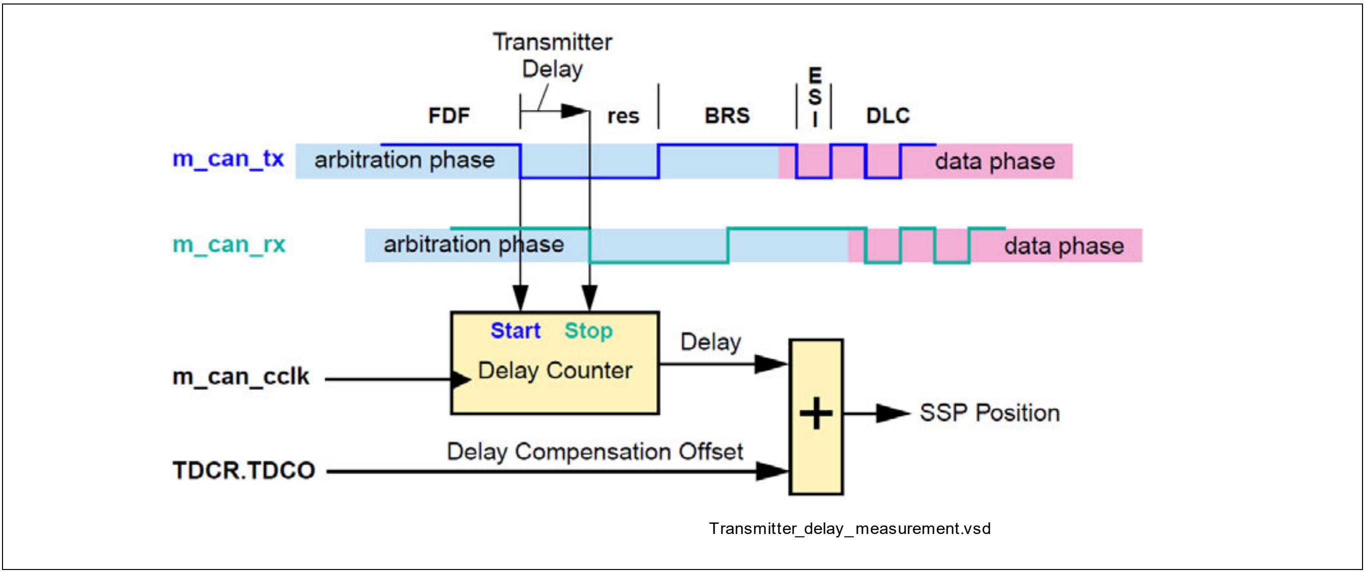


Figure 591 Transmitter delay measurement

To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received "res" bit, resulting in a to early Secondary Sample Point position, the use of a transmitter delay compensation filter window can be enabled by programming **TDCRi (i=0-3)**.TDCF.This defines a minimum value for the Secondary Sample Point position. Dominant edges on RX, that would result in an earlier Secondary Sample Point position are ignored for transmitter delay measurement. The measurement is stopped when the Secondary Sample Point position is at least **TDCRi (i=0-3)**.TDCF AND RX is low.

40.3.2.1.5 Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle to resynchronize itself to the CAN communication. The error counters (**ECRi (i=0-3)**.REC and **ECRi (i=0-3)**.TEC) are frozen while Error Logging (**ECRi (i=0-3)**.CEL) is active. The Host can set the M_CAN into Restricted Operation mode by setting bit CCCRi.ASM. The bit can only be set by the Host when both CCCRi.CCE and CCCRi.INIT are set to '1'. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset **CCCRi (i=0-3)**.ASM.

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

40.3.2.1.6 Bus Monitoring Mode

The M_CAN is set in Bus Monitoring Mode by programming **CCCRi (i=0-3)**.MON to one or when error level S3 (**TTOST0**.EL = "11") is entered. 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 592** shows the connection of signals TXD and RXD to the M_CAN in Bus Monitoring Mode.

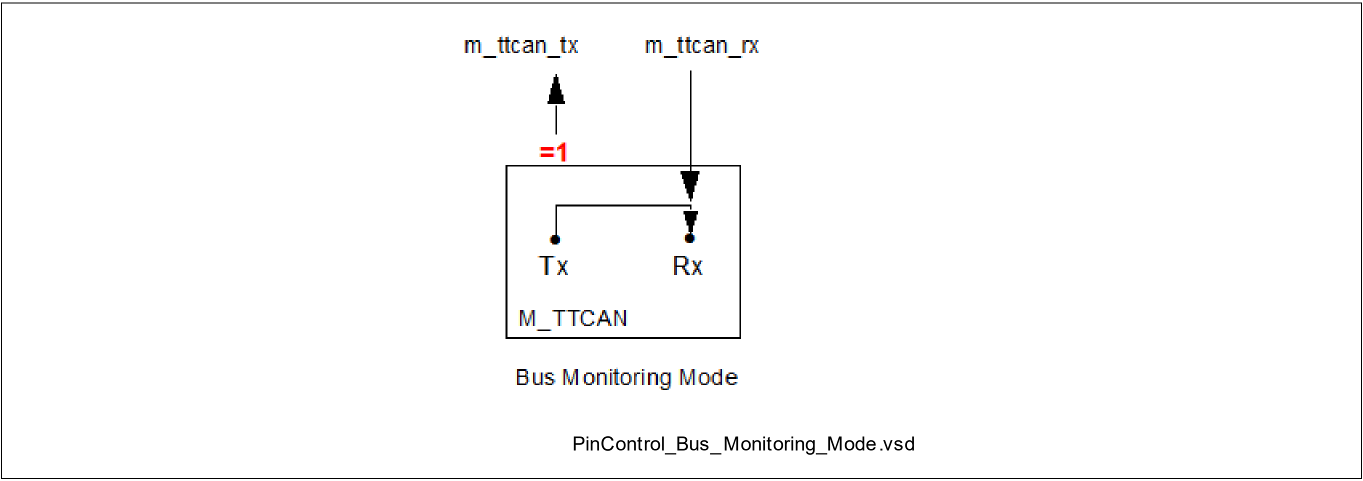


Figure 592 Pin Control in Bus Monitoring Mode

40.3.2.1.7 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 **CCCRi (i=0-3)**.DAR.

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 **TXBRPi (i=0-3)**.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 **TXBTOi (i=0-3)**.TOx set
  - Corresponding Tx Buffer Cancellation Finished bit **TXBCFi (i=0-3)**.CFx not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx Buffer Transmission Occurred bit **TXBTOi (i=0-3)**.TOx set
  - Corresponding Tx Buffer Cancellation Finished bit **TXBCFi (i=0-3)**.CFx set
- Arbitration lost or frame transmission disturbed:
  - Corresponding Tx Buffer Transmission Occurred bit **TXBTOi (i=0-3)**.TOx not set
  - Corresponding Tx Buffer Cancellation Finished bit **TXBCFi (i=0-3)**.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).

40.3.2.1.8 Power Down (Sleep Mode)

The M_CAN can be set into power down mode controlled by input signal clock stop request or via CC Control Register **CCCRi (i=0-3)**.CSR. As long as the clock stop request signal is active, bit **CCCRi (i=0-3)**.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 **CCCRi (i=0-3)**.INIT to one to prevent any further CAN transfers. Now the M_CAN acknowledges that it is ready for power down by setting **CCCRi (i=0-3)**.CSA to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to **CCCRi (i=0-3)**.INIT will have no effect. Now the module clocks may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting CC Control Register flag **CCCRi (i=0-3)**.CSR. The M_CAN will acknowledge this by resetting **CCCRi (i=0-3)**.CSA. Afterwards, the application can restart CAN communication by resetting bit **CCCRi (i=0-3)**.INIT.

40.3.2.1.9 Test Modes

To enable write access to register TEST (see **Test Register i**), bit **CCCRi (i=0-3)**.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 TXD by programming **TESTi (i=0-3)**.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 RXD can be read from **TESTi (i=0-3)**.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 **TESTi (i=0-3)**.TX until the new configuration is visible at output pin TXD. This applies also when reading input pin RXD via **TESTi (i=0-3)**.RX.

Note: Test modes should be used for production tests or self test only. The software control for pin TXD interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External Loop Back Mode

The M_CAN can be set in External Loop Back Mode by programming **TESTi (i=0-3)**.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 593** shows the connection of TXD and RXD 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 RXD input pin is disregarded by the M_CAN. The transmitted messages can be monitored at the TXD pin.

Internal Loop Back Mode

Internal Loop Back Mode is entered by programming bits **TESTi (i=0-3)**.LBCK and **CCCRi (i=0-3)**.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 TXD and RXD. In this mode pin RXD is disconnected from the M_CAN and pin TXD is held recessive. **Figure 593** shows the connection of TXD and RXD to the M_CAN in case of Internal Loop Back Mode.

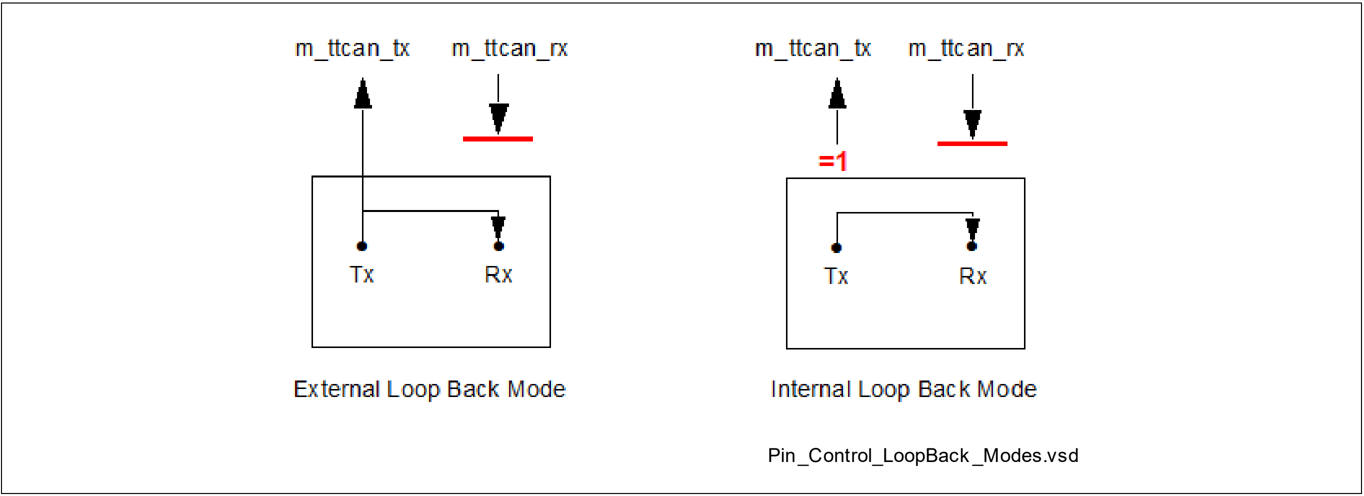


Figure 593 Pin Control in Loop Back Modes

40.3.2.1.10 Application Watchdog

The application watchdog is served by reading register **TTOST0**. When the application watchdog is not served in time, bit **TTOST0**.AWE is set, all TTCAN communication is stopped, and the M_CAN is set into Bus Monitoring Mode.

The TT Application Watchdog can be disabled by programming the Application Watchdog Limit **TTOCF0**.AWL to 0x00. The TT Application Watchdog should not be disabled in a TTCAN application program.

40.3.2.2 Timestamp Generation

For timestamp generation the M_CAN supplies a 16-bit wrap-around counter. A prescaler **TSCCi (i=0-3)**.TCP can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via **TSCVi (i=0-3)**.TSC. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag **IRi (i=0-3)**.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 **TSCCi (i=0-3)**.TSS an external 16-bit timestamp can be used. The external timer can be controlled using the **NTCCRi (i=0-3)** register. The external timer can be started by setting **NTCCRi (i=0-3)**.STSTART. It can be reset to zero by setting **NTCCRi (i=0-3)**.STRESET. A write of '1b' to **NTCCRi (i=0-3)**.STSTART is effective only when **NTCCRi (i=0-3)**.STRESET is '0b' ie., the external timer cannot be started when **NTCCRi (i=0-3)**.STRESET is '1b'.

40.3.2.3 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 **TSCCi (i=0-3)**.TCP as the Timestamp Counter. The Timeout Counter is configured via register **TOCCi (i=0-3)**. The actual counter value can be read from **TOCVi (i=0-3)**.TOC.

The Timeout Counter can only be started while **CCCRi (i=0-3)**.INIT = '0'. It is stopped when **CCCRi (i=0-3)**.INIT = '1', e.g. when the M_CAN enters Bus_Off state.

The operation mode is selected by **TOCCi (i=0-3)**.TOS. When operating in Continuous Mode, the counter starts when CCCRi.INIT is reset. A write to **TOCVi (i=0-3)** presets the counter to the value configured by **TOCCi (i=0-3)**.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 **TOCCi (i=0-3)**.TOP. Down-counting is started when the first FIFO element is stored. Writing to **TOCVi (i=0-3)** has no effect.

When the counter reaches zero, interrupt flag **IRi (i=0-3)**.TOO is set. In Continuous Mode, the counter is immediately restarted at **TOCCi (i=0-3)**.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.

40.3.2.4 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.

40.3.2.4.1 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 **GFCi (i=0-3)**
- Standard ID Filter Configuration **SIDFCi (i=0-3)**
- Extended ID Filter Configuration **XIDFCi (i=0-3)**
- Extended ID AND Mask **XIDAMi (i=0-3)**

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag **IRi (i=0-3)**.HPM
- Set High Priority Message interrupt flag **IRi (i=0-3)**.HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32-bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

Rx Buffer
New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see **PSRi (i=0-3)**.LEC respectively **PSRi (i=0-3)**.FLEC.

Rx FIFO
Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see **PSRi (i=0-3)**.LEC respectively **PSRi (i=0-3)**.FLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in **Rx FIFO Overwrite Mode** have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

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

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.

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.

Standard Message ID Filtering

Figure 594 below shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in **Section 40.4.6.5**.

Controlled by the Global Filter Configuration **GFCi (i=0-3)** 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.

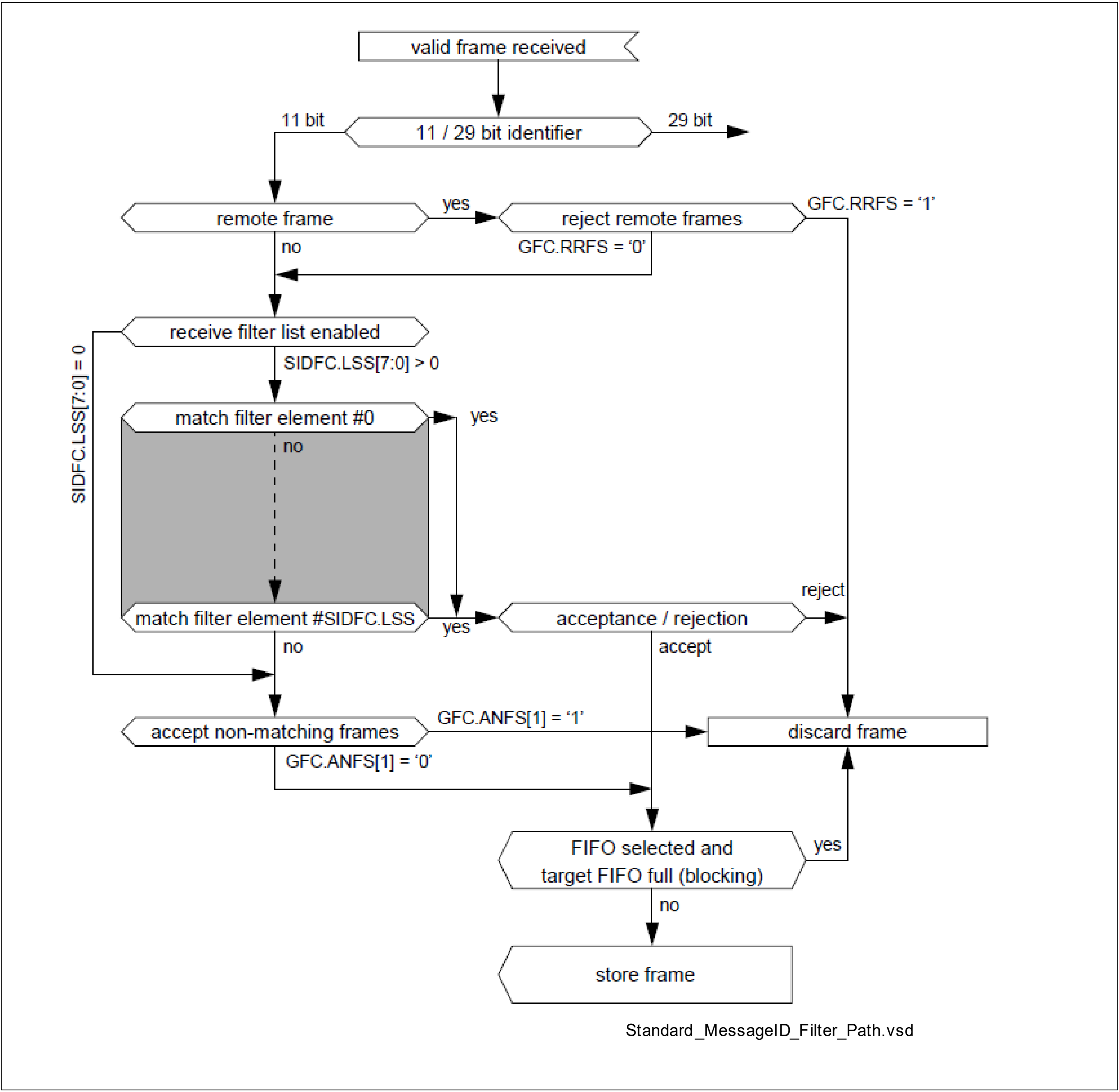


Figure 594 Standard Message ID Filter Path

Extended Message ID Filtering

Figure 595 below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in **Section 40.4.6.6**.

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.

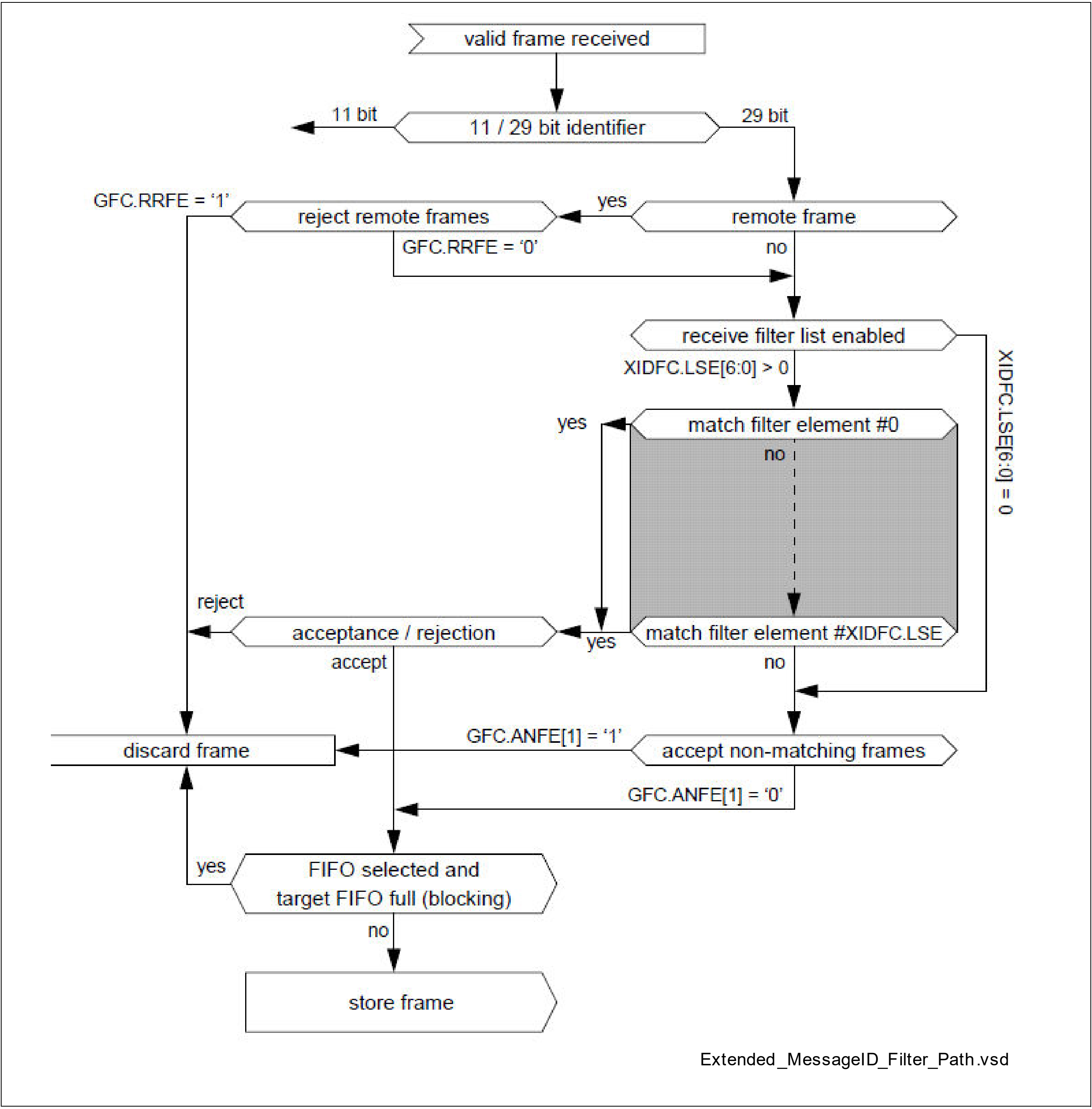


Figure 595 Extended Message ID Filter Path

40.3.2.4.2 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 **RXF0Ci (i=0-3)** and **RXF1Ci (i=0-3)**.

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 40.3.2.4.1**. The Rx FIFO element is described in **Section 40.4.6.2**.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC.FnWM, interrupt flag IRi.RFnW is set. When the Rx FIFO Put Index reaches the Rx

FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS.FnF. In addition interrupt flag **IRi (i=0-3)**.RFnF is set.

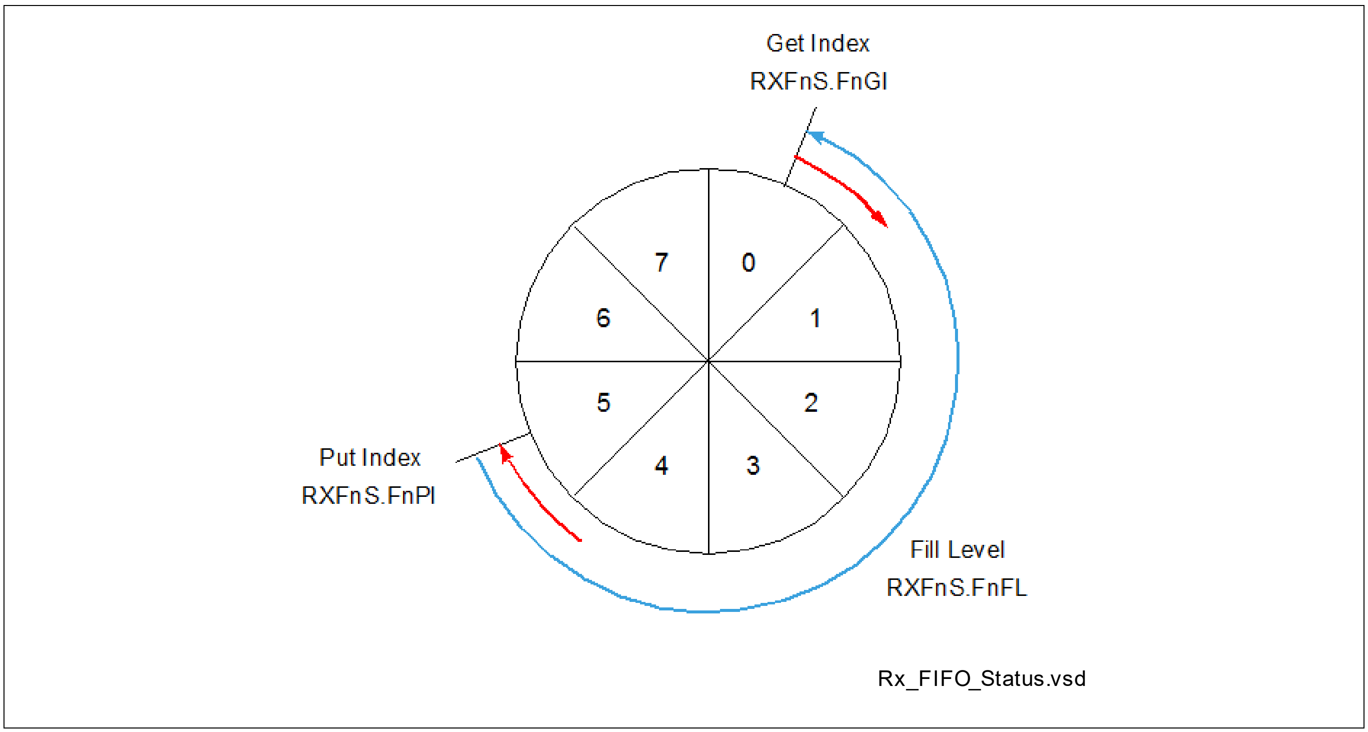


Figure 596 Rx FIFO Status

When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI • FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

Table 367 Rx Buffer / FIFO Element Size

RXESCi.RBDS[2:0] RXESCi.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = ‘0’. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by RXFnS.FnF = ‘1’. In addition interrupt flag IRi.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS.RFnL = ‘1’. In addition interrupt flag IRi.RFnL is set.

Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = ‘1’.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signalled by RXFnS.FnF = ‘1’, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. **Figure 597** shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

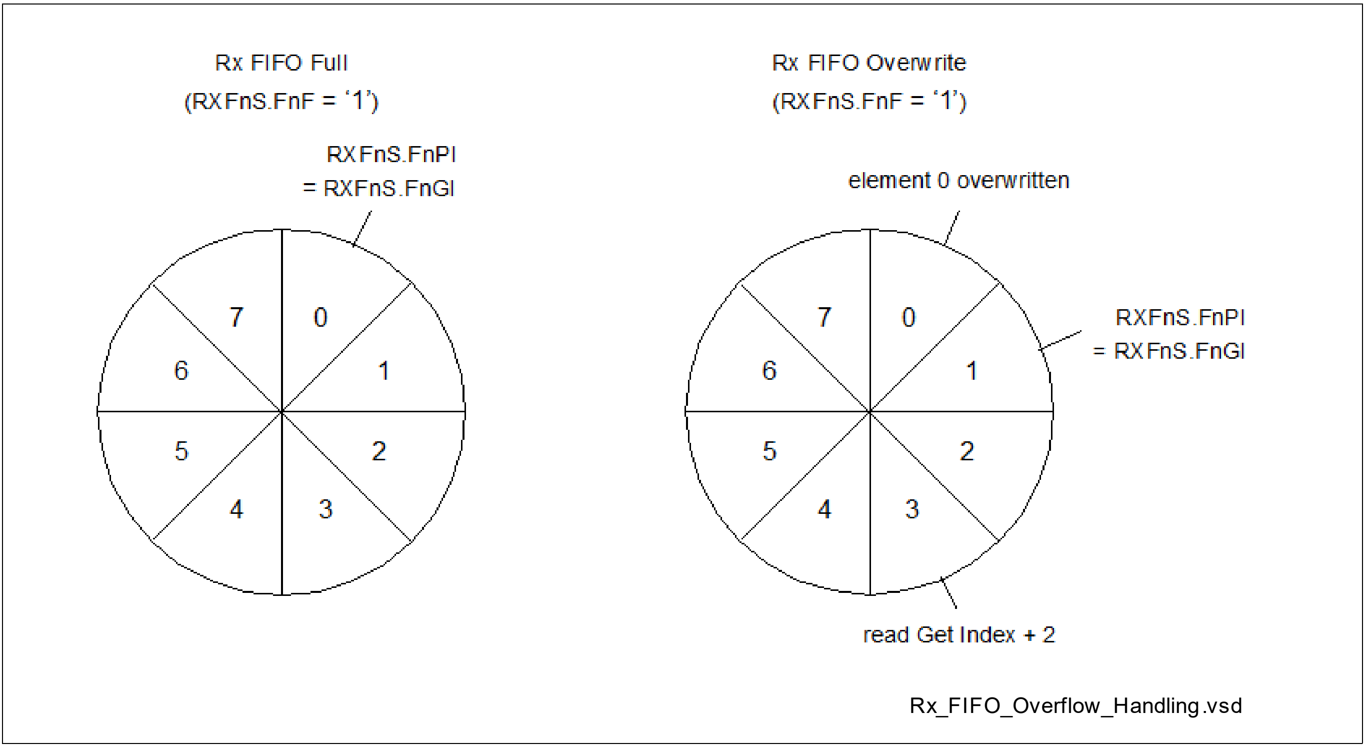


Figure 597 Rx FIFO Overflow Handling

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = ‘0’).

40.3.2.4.3 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 **RXBCi (i=0-3)**.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 40.4.6.5** and **Section 40.4.6.6**).

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 IRi.DRX (Message stored in dedicated Rx Buffer) in the interrupt register is set.

Table 368 **Example Filter Configuration for Rx Buffers**

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

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register **NDAT1i (i=0-3)**,**NDAT2i (i=0-3)** 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.

Rx Buffer Handling

- Reset interrupt flag IRi.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

40.3.2.5 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 Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The CAN mode for transmission (Classic CAN or CAN FD) can be configured separately for each Tx Buffer element. The Tx Buffer element is described in **Section 40.4.6.3**. **Table** below describes the possible configurations for frame transmission.

Table 369 **Possible Configuration for Frame Transmission**

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

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

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register **TXBRPi (i=0-3)** is updated, or when a transmission has been started.

40.3.2.5.1 Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit **CCCRi (i=0-3)**.TXP. If the bit is set, the M_CAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (**CCCRi (i=0-3)**.TXP = '0').

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

40.3.2.5.2 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an "Add Request" via **TXBARi (i=0-3)**.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (see **Chapter 40.3.2.5.3**). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0…31) • Element Size to the Tx Buffer Start Address **TXBCi (i=0-3)**.TBSA.

Table 370 Tx Buffer / FIFO / Queue Element Size

TXESCi.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

40.3.2.5.3 Tx FIFO

Tx FIFO operation is configured by programming **TXBCi (i=0-3)**.TFQM to ‘0’. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index **TXFQSi (i=0-3)**.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 **TXFQSi (i=0-3)**.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 **TXFQSi (i=0-3)**.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 (**TXFQSi (i=0-3)**.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 **TXBARi (i=0-3)**. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see **Table 370**). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index **TXFQSi (i=0-3)**.TFQPI (0…31) • Element Size to the Tx Buffer Start Address **TXBCi (i=0-3)**.TBSA.

40.3.2.5.4 Tx Queue

Tx Queue operation is configured by programming **TXBCi (i=0-3)**.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 **TXFQSi (i=0-3)**.TFQPI. An "Add Request" cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (**TXFQSi (i=0-3)**.TFQF = “1”), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see **Table 370**). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index **TXFQSi (i=0-3)**.TFQPI (0…31) • Element Size to the Tx Buffer Start Address **TXBCi (i=0-3)**.TBSA.

40.3.2.5.5 Mixed Dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by TXBCi.NDTB. The number of Tx Buffers assigned to the Tx FIFO is configured by **TXBCi (i=0-3)**.TFQSi. In case **TXBCi (i=0-3)**.TFQS is programmed to 0x0, only dedicated Tx Buffers are used.

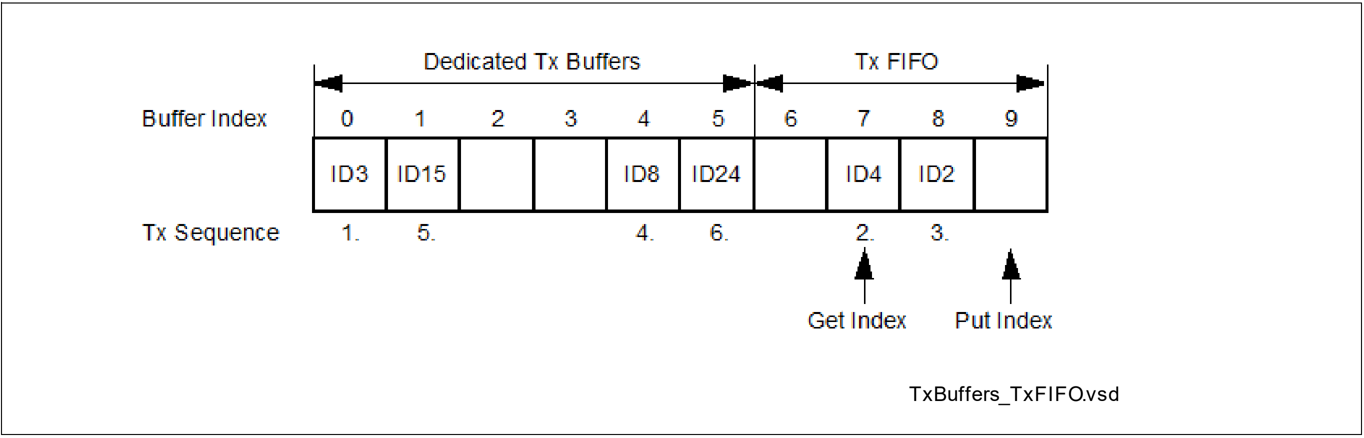


Figure 598 Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by **TXFQSi (i=0-3)**.TFGI)
- Buffer with lowest Message ID gets highest priority and is transmitted next

40.3.2.5.6 Mixed Dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by **TXBCi (i=0-3)**.NDTB. The number of Tx Queue Buffers is configured by **TXBCi (i=0-3)**.TFQSi. In case **TXBCi (i=0-3)**.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

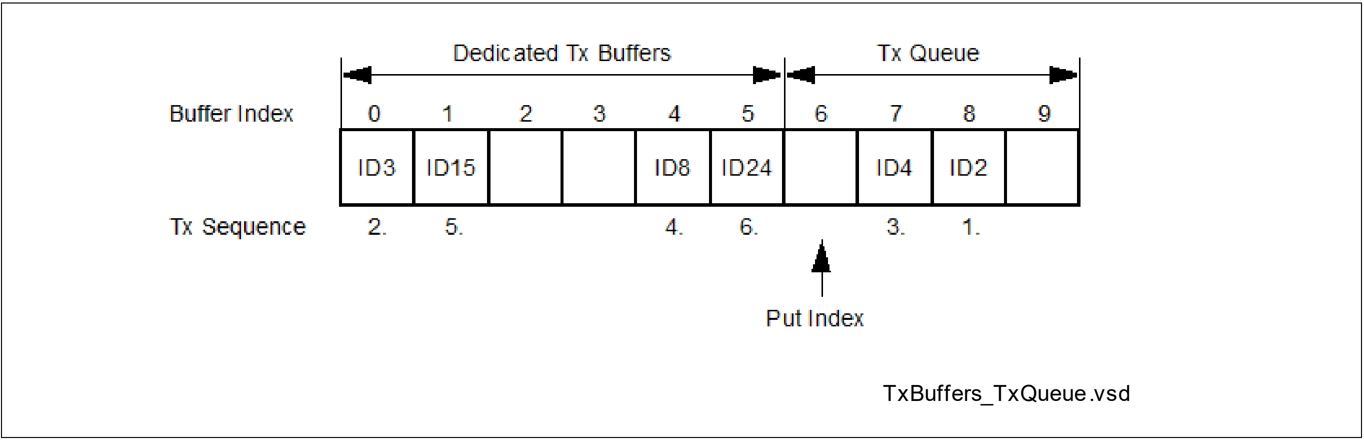


Figure 599 Example of mixed Configuration Dedicated Tx Buffers / Tx Queue

Tx prioritization:

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

40.3.2.5.7 Transmit Cancellation

The M_CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a Dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register **TXBCRi (i=0-3)**. 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.

40.3.2.5.8 Tx Event Handling

To support Tx event handling the M_CAN has implemented a Tx Event FIFO. After the M_CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in **Section 40.4.6.4**.

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

When a Tx Event FIFO full condition is signalled by **IRi (i=0-3)**.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 **IRi (i=0-3)**.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 **TXEFCi (i=0-3)**.EFWM, interrupt flag IRi.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index **TXEFSi (i=0-3)**.EFGI has to be added to the Tx Event FIFO start address **TXEFCi (i=0-3)**.EFSA.

40.3.2.6 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 **Rx FIFO 0 Acknowledge i**, **Rx FIFO 1 Acknowledge i**, and **Tx Event FIFO Acknowledge i**). 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.

40.3.2.7 OCDS Suspend Behaviour Support

In case of Hard Suspend, the clock is switched off to the MCMCAN module, and any ongoing transmission or reception event will be suspended immediately.

In case of Soft Suspend, the clock is switched off to the MCMCAN module after completion of ongoing transmission and reception events. The CCCR.INIT bit will be set during the Soft Suspend. Hence, when leaving the Soft Suspend, the debugger has to ensure clearing of the CCCR.INIT bit, inorder to continue normal CAN operation.

40.3.3 TTCAN Operation

40.3.3.1 Reference Message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the Time Master (sender of the reference message).

For Level 1 the data length must be at least one; for Level 0,2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to 8 potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference message is configured via register TTRMC.

The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted Master_Ref_Mark is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (Next_is_Gap bit) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters.

The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

40.3.3.1.1 Level 1

Level 1 operation is configured via TTOCF0.OM = "01" and TTOCF0.GEN. External clock synchronization is not available in Level 1.

The information related to the reference message is stored in the first data byte as shown in **Table 371** below. Cycle_Count is optional.

Table 371 First byte of Level 1 reference message

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
First Byte	Next_is_Gap	0	Cycle_Count[5:0]					

40.3.3.1.2 Level 2

Level 2 operation is configured via TTOCF0.OM = “10” and TTOCF0.GEN.

The information related to the reference message is stored in the first four data bytes as shown in **Table 372** below. Cycle_Count and the lower four bits of NTU_Res are optional. The M_CAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

Table 372 First four bytes of Level 2 reference message

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
First Byte	Next_is_Gap	0	Cycle_Count[5:0]					
Second Byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third Byte	Master_Ref_Mark[7:0]							
Fourth Byte	Master_Ref_Mark[15:8]							

40.3.3.1.3 Level 0

Level 0 operation is configured via TTOCF0.OM = “11”. External event-synchronized time-triggered operation is not available in Level 0.

The information related to the reference message is stored in the first four data bytes as shown in **Table 372** below. In Level 0 Next_is_Gap is always zero. Cycle_Count and the lower four bits of NTU_Res are optional. The M_CAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

Table 373 First four bytes of Level 0 reference message

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
First Byte	Next_is_Gap	0	Cycle_Count[5:0]					
Second Byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third Byte	Master_Ref_Mark[7:0]							
Fourth Byte	Master_Ref_Mark[15:8]							

40.3.3.2 TTCAN Configuration

40.3.3.2.1 TTCAN Timing

The Network Time Unit NTU is the unit in which all times are measured. The NTU is a constant of the whole network and is defined a priori by the network system designer. In TTCAN Level 1 the NTU is the nominal CAN bit time. In TTCAN Level 0 and Level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable.

In TTCAN Level 0 and Level 2 the length of the NTU is defined by the Time Unit Ratio TUR. The TUR is in principle a non-integer number and given by the formula TUR = TURNA0.NAV/TURCF0.DC. The length of the NTU is given by the formula NTU = CAN Clock Period • TUR.

The TUR Numerator Configuration NC is an 18-bit number, TURCF0.NCL[15:0] can be programmed in the range 0x0000-0xFFFF. TURCF0.NCH[17:16] is hard wired to 0b01. When the number 0xnnnn is written to TURCF0.NCL[15:0], TURNA0.NAV starts with the value 0x10000 + 0x0nnnn = 0x1nnnn. The TUR Denominator Configuration TURCF0.DC is a 14-bit number. TURCF0.DC may be programmed in the range 0x0001 - 0x3FFF, 0x0000 is an illegal value.

In Level 1, NC must be ≥ 4 • TURCF0.DC. In Level 0,2 NC must be ≥ 8 • TURCF0.DC to allow the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets TURCF0.DC to 0x1000 and TURCF0.NCL to 0x10000, resulting in an NTU consisting of 16 CAN clock periods. Local time and application watchdog are not started before either the CCCRi.INIT is reset, or TURCF0.ELT is set. TURCF0.ELT may not be set before the NTU is configured. Setting TURCF0.ELT to ‘1’ also locks the write access to register TURCF0.

At startup TURNA0.NAV is updated from NC (= TURCF0.NCL + 0x10000) when TURCF0.ELT is set. In TTCAN Level 1 there is no drift compensation. TURNA0.NAV does not change during operation, it always equals NC.

In TTCAN Level 0 and Level 2 there are two possibilities for TURNA0.NAV to change. When operating as time slave or backup time master, and when TTOCF0.ECC is set, TURNA0.NAV is updated automatically to the value calculated from the monitored global time speed, as long as the M_CAN is in synchronization state In_Schedule or In_Gap. When it loses synchronization it returns to NC. When operating as the actual time master, and when TTOCF0.EECS is set, the Host may update TURCF0.NCL. When the Host sets TTOCN0.ECS, TURNA0.NAV will be updated from the new value of NC at the next reference message. The status flag TTOST0.WECS as is set when TTOCN0.ECS is set and is cleared when TURNA0.NAV is updated. TURCF0.NCL is write locked while TTOST0.WECS is set.

In TTCAN Level 0 and Level 2 the clock calibration process adapts TURNA0.NAV in the range of the Synchronization Deviation Limit SDL of NC ± $2^{(\text{TTOCF0.LDSDL}+5)}$. TURCF0.NCL should be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of TURNA0.NAV.

The synchronization deviation SD is the difference between NC and TURNA0.NAV (SD = |NC - TURNA0.NAV|). It is limited by the Synchronization Deviation Limit SDL, which is configured by its dual logarithm TTOCF0.LDSDL (SDL = $2^{(\text{TTOCF0.LDSDL}+5)}$) and should not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new Basic Cycle. When the calculated TURNA0.NAV deviates by more than SDL from NC, or if the Disc_Bit in the reference message is set, the drift compensation is suspended and TTIR0.GTE is set and TTOSC.QCS is reset, or in case of the Disc_Bit = ‘1’, TTIR0.GTD is set.

TUR configuration examples are shown in **Table 374** below.

Table 374 TUR Configuration Examples

TUR	8	10	24	50	510	125000	32.5	100/12	529/17
NC	0x1FFF8	0x1FFFE	0x1FFF8	0x1FFEA	0x1FFFE	0x1E848	0x1FFE0	0x19000	0x10880
TURCF0.DC	0x3FFF	0x3333	0x1555	0x0A3D	0x0101	0x0001	0x0FC0	0x3000	0x0880

TTOCN0.ECS schedules NC for activation by the next reference message. TTOCN0.SGT schedules TTGTP0.TP for activation by the next reference message. Setting of TTOCN0.ECS and TTOCN0.SGT requires TTOCF0.EECS to be set (external clock synchronization enabled) while the M_CAN is actual time master.

The M_CAN module provides an application watchdog to verify the function of the application program. The Host has to serve this watchdog regularly, else all CAN bus activity is stopped. The Application Watchdog Limit TTOCF0.AWL specifies the number of NTUs between two times the watchdog has to be served. The maximum number of NTUs is 256. The Application Watchdog is served by reading register TTOST0. TTOST0.AWE indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, interrupt flag TTIR0.AW is set. For software development, the application watchdog may be disabled by programming TTOCF0.AWL to 0x00 (see also **Chapter 40.3.2.1.10**).

40.3.3.2.2 Message Scheduling

TTOCF0.TM controls whether the M_CAN operates as a potential time master or as a time slave. If it is a potential time master, the three LSBs of the reference message's identifier TTRMC.RID define the master priority, 0 giving the highest and 7 giving the lowest priority. There may not be two nodes in the network using the same master priority. TTRMC.RID is used for recognition of reference messages. TTRMC.RMPS is not relevant for time slaves.

The Initial Reference Trigger Offset TTOCF0.IRTO is a 7-bit-value that defines (in NTUs) how long a backup time master waits before it starts the transmission of a reference message when a reference message is expected but the bus remains idle. The recommended value for TTOCF0.IRTO is the master priority multiplied with a factor depending on the expected clock drift between the potential time masters in the network. The sequential order of the backup time masters, when one of them starts the reference message in case the current time master fails, should correspond to their master priority, even with maximum clock drift.

TTOCF0.OM decides whether the node operates in TTCAN Level 0, Level 1, or Level 2. In one network, all potential time masters have to operate on the same level. Time slaves may operate on Level 1 in a Level 2 network, but not vice versa. The configuration of the TTCAN operation mode via TTOCF0.OM is the last step in the setup. With TTOCF0.OM = "00" (event-driven CAN communication), the M_CAN operates according to ISO 11898-1, without time triggers. With TTOCF0.OM = "01" (Level 1), the M_CAN operates according to ISO 11898-4, but without the possibility to synchronize the basic cycles to external events, the Next_is_Gap bit in the reference message is ignored. With TTOCF0.OM = "10" (Level 2), the M_CAN operates according to ISO 11898-4, including the event-synchronized start of a basic cycle. With TTOCF0.OM = "11" (Level 0), the M_CAN operates as event-driven CAN but maintains a calibrated global time base as in Level 2.

TTOCF0.EECS enables the external clock synchronisation, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in Level 0,2 only) the global clock phase to an external reference.

TTMLM.ENTT in the TT Matrix Limits register specifies the number of expected Tx_Triggers in the system matrix. This is the sum of Tx_Triggers for exclusive, single arbitrating and merged arbitrating windows, excluding the Tx_Ref_Triggers. Note that this is usually not the number of Tx_Trigger memory elements; the number of basic cycles in the system matrix and the trigger's repeat factors have to be taken into account. An inaccurate configuration of TTMLM.ENTT will result in either a Tx Count Underflow (TTIR0.TXU = '1' and TTOST0.EL = "01", severity 1) or in a Tx Count Overflow (TTIR0.TXO = '1' and TTOST0.EL = "10", severity 2).

Note: *In case the first reference message seen by a node does not have Cycle_Count zero, this node may finish its first matrix cycle with its Tx count resulting in a Tx Count Underflow condition. As long as a node is in state Synchronizing its Tx_Triggers will not lead to transmissions.*

TTMLM.CCM specifies the number of the last basic cycle in the system matrix. The counting of basic cycles starts at 0. In a system matrix consisting of 8 basic cycles TTMLM.CCM would be 7. TTMLM.CCM is ignored by time slaves, a receiver of a reference message considers the received cycle count as the valid cycle count for the actual basic cycle.

TTMLM.TXEW specifies the length of the Tx enable window in NTUs. The Tx enable window is that period of time at the beginning of a time window where a transmission may be started. If the sample point of the first bit of a transmit message is not inside the Tx enable window because of e.g. a slight overlap from the previous time window's message, the transmission cannot be started in that time window at all. TTMLM.TXEW has to be chosen with respect to the network's synchronisation quality and with respect to the relation between the length of the time windows and the length of the messages.

40.3.3.2.3 Trigger Memory

The trigger memory is part of the external Message RAM to which the M_CAN is connected via its Generic Master Interface (see **Figure 605**). It stores up to 64 trigger elements. A trigger memory element consists of Time Mark TM, Cycle Code CC, Trigger Type TYPE, Filter Type FTYPE, Message Number MNR, Message Status Count MSC, Time Mark Event Internal TMIN andTime Mark Event External TMEX.

The time mark defines at which cycle time a trigger becomes active. The triggers in the trigger memory have to be sorted by their time marks. The trigger element with the lowest time mark is written to the first trigger memory word. Message number and cycle code are ignored for triggers of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, and End_of_List.

When the cycle time reaches the time mark of the actual trigger, the TTCAN node switches to the next trigger and starts to read the following trigger from the trigger memory. In case of a transmit trigger, the Tx Handler starts to read the message from the Message RAM as soon as the TTCAN node switches to its trigger. The RAM access speed defines the minimum time step between a transmit trigger and its preceding trigger, the Tx Handler has to be able to prepare the transmission before the transmit trigger's time mark is reached. The RAM access speed also limits the number of non-matching (with regard to their cycle code) triggers between two matching triggers, the next matching trigger must be read before its time mark is reached. If the reference message is n NTU long, a trigger with a time mark < n will never become active and will be treated as a configuration error.

Starting point of the cycle time is the sample point of the reference message's start of frame bit. The next reference message is requested when cycle time reaches the Tx_Ref_Trigger's time mark. The M_CAN reacts on the transmission request at the next sample point. A new Sync_Mark is captured at the start of frame bit, but the cycle time is incremented until the reference message is successfully transmitted (or received) and the Sync_Mark is taken as the new Ref_Mark. At that point in time, cycle time is restarted. As a consequence, cycle time can never (with the exception of initialisation) be seen at a value < n, with n being the length of the reference message measured in NTU.

Length of a basic cycle: Tx_Ref_Trigger's time mark + 1 NTU + 1 CAN bit time

The trigger list will be different for all nodes in the TTCAN network. Each node knows only the Tx_Triggers for its own transmit messages, the Rx_Triggers for those receive messages that are processed by this node, and the triggers concerning the reference messages.

Trigger Types

Tx_Ref_Trigger (TYPE = "0000") and Tx_Ref_Trigger_Gap (TYPE = "0001") cause the transmission of a reference message by a time master. A configuration error (TTOST0.EL = "11", severity 3) is detected when a time slave encounters a Tx_Ref_Trigger(_Gap) in its trigger memory. Tx_Ref_Trigger_Gap is only used in external event-synchronised time-triggered operation mode. In that mode, Tx_Ref_Trigger is ignored when the M_CAN synchronisation state is In_Gap (TTOST0.SYS = "10").

Tx_Trigger_Single (TYPE = "0010"), Tx_Trigger_Continous (TYPE = "0011"), Tx_Trigger_Arbitration (TYPE = "0100"), and Tx_Trigger_Merged (TYPE = "0101") cause the start of a transmission. They define the start of a time window.

Tx_Trigger_Single starts a single transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit is reset.

Tx_Trigger_Continous starts a transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit remains set, and the message buffer is transmitted again in the next matching time window.

Tx_Trigger_Arbitration starts an arbitrating time window, Tx_Trigger_Merged a merged arbitrating time window. The last Tx_Trigger of a merged arbitrating time window must be of type Tx_Trigger_Arbitration. A Configuration Error (TTOST0.EL = "11", severity 3) is detected when a trigger of type Tx_Trigger_Merged is followed by any other

Tx_Trigger than one of type Tx_Trigger_Merged or Tx_Trigger_Arbitration. Several Tx_Triggers may be defined for the same Tx message buffer. Depending on their cycle code, they may be ignored in some basic cycles. The cycle code has to be considered when the expected number of Tx_Triggers (TTMLM.ENTT) is calculated.

Watch_Trigger (TYPE = “0110”) and Watch_Trigger_Gap (TYPE = “0111”) check for missing reference messages. They are used by both time masters and time slaves. Watch_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, a Watch_Trigger is ignored when the M_CAN synchronisation state is In_Gap (TTOST0.SYS = “10”).

Rx_Trigger (TYPE = “1000”) is used to check for the reception of periodic messages in exclusive time windows. Rx_Triggers are not active until state In_Schedule or In_Gap is reached. The time mark of an Rx_Trigger shall be placed after the end of that message’s transmission, independent of time window boundaries. Depending on their cycle code, Rx_Triggers may be ignored in some basic cycles. At the time mark of the Rx_Trigger, it is checked whether the last received message before this time mark and after start of cycle or previous Rx_Trigger had matched the acceptance filter element referenced by MNR. Accepted messages are stored in one of the two receive FIFOs, according to the acceptance filtering, independent of the Rx_Trigger. Acceptance filter elements which are referenced by Rx_Triggers should be placed at the beginning of the filter list to ensure that the filtering is finished before the Rx_Trigger’s time mark is reached.

Time_Base_Trigger (TYPE = “1001”) are used to generate internal/external events depending on the configuration of ASC, TMIN, and TMEX.

End_of_List (TYPE = “1010…1111”) is an illegal trigger type, a configuration error (TTOST0.EL = “11”, severity 3) is detected when an End_of_List trigger is encountered in the trigger memory before the Watch_Trigger or Watch_Trigger_Gap.

Restrictions for the Node’s Trigger List

There may not be two triggers that are active at the same cycle time and cycle count, but triggers that are active in different basic cycles (different cycle code) may share the same time mark.

Rx_Triggers and Time_Base_Triggers may not be placed inside the Tx enable windows of Tx_Trigger_Single/Continuous/Arbitration, but they may be placed after Tx_Trigger_Merged.

Triggers that are placed after the Watch_Trigger (or the Watch_Trigger_Gap when TTOST0.SYS = “10”) will never become active. The watch triggers themselves will not become active when the reference messages are transmitted on time.

All unused trigger memory words (after the Watch_Trigger or after the Watch_Trigger_Gap when TTOST0.SYS = “10”) must be set to trigger type End_of_List.

A typical trigger list for a potential time master will begin with a number of Tx_Triggers and Rx_Triggers followed by the Tx_Ref_Trigger and the Watch_Trigger. For networks with external event- synchronized time-triggered communication, this is followed by the Tx_Ref_Trigger_Gap and the Watch_Trigger_Gap. The trigger list for a time slave will be the same but without the Tx_Ref_Trigger and the Tx_Ref_Trigger_Gap.

At the beginning of each basic cycle, that is at each reception or transmission of a reference message, the trigger list is processed starting with the first trigger memory element. The FSE looks for the first trigger with a cycle code that matches the current cycle count. The FSE waits until cycle time reaches the trigger’s time mark and activates the trigger. Afterwards the FSE looks for the next trigger in the list with a cycle code that matches the current cycle count.

Special consideration is needed for the time around Tx_Ref_Trigger and Tx_Ref_Trigger_Gap. In a time master competing for master ship, the effective time mark of a Tx_Ref_Trigger may be decremented in order to be the first node to start a reference message. In backup time masters the effective time mark of a Tx_Ref_Trigger or Tx_Ref_Trigger_Gap is the sum of its configured time mark and the Reference Trigger Offset TTOCF0.IRTO. In case error level 2 is reached (TTOST0.EL = “10”), the effective time mark is the sum of its time mark and 0x127. No other trigger elements should be placed in this range otherwise it may happen, that the time marks appear out of order and are flagged as a configuration error. Trigger elements which are coming after Tx_Ref_Trigger may never become active as long as the reference messages come in time.

There are interdependencies between the following parameters:

- Host clock frequency
- Speed and waiting time for Trigger RAM accesses
- Length of the acceptance filter list
- Number of trigger elements
- Complexity of cycle code filtering in the trigger elements
- Offset between time marks of the trigger elements

Example for Trigger Handling

The example below shows how the trigger list is derived from a node's system matrix. Assumed node A is first time master and has knowledge of the section of the system matrix shown in **Table 375** below.

Table 375 System Matrix Node A

Cycle Count	Time Mark1	Time Mark2	Time Mark3	Time Mark4	Time Mark5	Time Mark6	Time Mark7
0	Tx7					TxRef	Error
1	Rx3		Tx2, Tx4			TxRef	Error
2						TxRef	Error
3	Tx7		Rx5			TxRef	Error
4	Tx7			Rx6		TxRef	Error

The cycle count starts with 0 and runs until 0, 1, 3, 7, 15, 31, 63 (the number of basic cycles in the system matrix is 1, 2, 4, 8, 16, 32, 64). The maximum cycle count is configured by TTMLM.CCM. The Cycle Code CC is composed of repeat factor (= value of most significant '1') and the number of the first basic cycle in the system matrix (= bit field after most significant '1').

Example:with a cycle code of 0b0010011 (repeat factor: 16, first basic cycle: 3) and a maximum cycle count of TTMLM.CCM = "0x3F"matches occur at cycle counts 3, 19, 35, 51

A trigger element consists of Time Mark TM, Cycle Code CC, Trigger Type TYPE, and Message Number MNR. For transmission MNR references the Tx Buffer number (0…31). For reception MNR references the number of the filter element (0…127) that matched during acceptance filtering. Depending on the configuration of the Filter Type FTYPE, the 11-bit or 29-bit message ID filter list is referenced.

In addition a trigger element can be configured for Asynchronous Serial Communication ASC, generation of Time Mark Event Internal TMIN, and Time Mark Event External TMEX. The Message Status Count MSC holds the counter value (0…7) for scheduling errors for periodic messages in exclusive time windows at the point in time when the time mark of the trigger element became active.

Table 376 Trigger List Node A

Trigger	Time Mark TM[15:0]	Cycle Code CC[6:0]	Trigger Type TYPE[3:0]	Mess. No. MNR[6:0]
0	Mark1	0b0000100	Tx_Trigger_Single	7
1	Mark1	0b1000000	Rx_Trigger	3
2	Mark1	0b1000011	Tx_Trigger_Single	7
3	Mark3	0b1000001	Tx_Trigger_Merged	2

Table 376 Trigger List Node A (cont'd)

Trigger	Time Mark TM[15:0]	Cycle Code CC[6:0]	Trigger Type TYPE[3:0]	Mess. No. MNR[6:0]
4	Mark3	0b1000011	Rx_Trigger	5
5	Mark4	0b1000001	Tx_Trigger_Arbitration	4
6	Mark4	0b1000100	Rx_Trigger	6
7	Mark6	n.a.	Tx_Ref_Trigger	0 (Ref)
8	Mark7	n.a.	Watch_Trigger	n.a.
9	n.a.	n.a.	End_of_List	n.a.

Tx_Trigger_Single, Tx_Trigger_Continous, Tx_Trigger_Merged, Tx_Trigger_Arbitration, Rx_Trigger, and Time_Base_Trigger are only valid for the specified cycle code. For all other trigger types the cycle code is ignored.

The FSE starts the basic cycle with scanning the trigger list starting from zero until a trigger with time mark > cycle time and with its Cycle Code CC matching the actual cycle count is reached, or a trigger of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, or Watch_Trigger_Gap is encountered.

When the cycle time reached the Time Mark TM, the action defined by Trigger Type TYPE and Message Number MNR is started. There is an error in the configuration when End_of_List is reached.

At Mark6 the reference message (always TxRef) is transmitted. After transmission of the reference message the FSE returns to the beginning of the trigger list. When the Watch Trigger at Mark7 is reached, the node was not able to transmit the reference message; error treatment is started.

Detection of Configuration Errors

A configuration error is signalled via TTOST0.EL = "11" (severity 3) when:

The FSE comes to a trigger in the list with a cycle code that matches the current cycle count but with a time mark that is less than the cycle time.

The previous active trigger was a Tx_Trigger_Merged and the FSE comes to a trigger in the list with a cycle code that matches the current cycle count but that is neither a Tx_Trigger_Merged nor a Tx_Trigger_Arbitration nor a Time_Base_Trigger nor an Rx_Trigger.

The FSE of a node with TTOCF0.TM='0' (time slave) encounters a Tx_Ref_Trigger or a Tx_Ref_Trigger_Gap.

Any time mark placed inside the Tx enable window (defined by TTMLM.TXEW) of a Tx_Trigger with a matching cycle code.

A time mark is placed near the time mark of a Tx_Ref_Trigger and the Reference Trigger Offset TTOST0.RTO causes a reversal of their sequential order measured in cycle time.

40.3.3.2.4 TTCAN Schedule Initialization

The synchronisation to the M_CAN's message schedule starts when CCCRi.INIT is reset. The M_CAN can operate strictly time-triggered (TTOCF0.GEN = '0') or external event-synchronized time-triggered (TTOCF0.GEN = '1'). All nodes start with cycle time zero at the beginning of their trigger list with TTOST0.SYS = "00" (out of synchronisation), no transmission is enabled with the exception of the reference message. Nodes in external event-synchronised time-triggered operation mode will ignore Tx_Ref_Trigger and Watch_Trigger and will use instead Tx_Ref_Trigger_Gap and Watch_Trigger_Gap until the first reference message decides whether a Gap is active.

Time Slaves

After configuration, a time slave will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message before reaching the Watch_Triggers. When it reaches Init_Watch_Trigger, interrupt flag TTIR0.IWT is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). The first received reference message will restart the FSE and the cycle time.

Note: *Init_Watch_Trigger is not part of the trigger list. It is implemented as an internal counter which counts up to 0xFFFF = maximum cycle time.*

When a time slave has received any message but the reference message before reaching the Watch_Triggers, it will assume a severe error (TTOST0.EL = "11", severity 3), set interrupt flag TTIR0.WT, switch off its CAN bus output, and enter the bus monitoring mode (CCCRi.MON set to '1'). In the bus monitoring mode it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

Note: *To leave the severe error state, the Host has to set CCCRi.INIT = '1'. After reset of CCCRi.INIT, the node restarts TTCAN communication.*

When no error is encountered during synchronisation, the first reference message sets TTOST0.SYS = "01" (Synchronizing), the second sets the TTCAN synchronization state (depending on its Next_is_Gap bit) to TTOST0.SYS = "11" (In_Schedule) or TTOST0.SYS = "10" (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

Potential Time Masters

After configuration, a potential time master will start the transmission of a reference message when it reaches its Tx_Ref_Trigger (or its Tx_Ref_Trigger_Gap when in external event-synchronized time-triggered operation). It will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message or transmit the reference message successfully before reaching the Watch_Triggers (assumed reason: all other nodes still in reset or configuration, giving no acknowledge). When it reaches Init_Watch_Trigger, the attempted transmission is aborted, interrupt flag TTIR0.IWT is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). Resetting TTIR0.IWT will re-enable the transmission of reference messages until next time the Init_Watch_Trigger condition is met, or another CAN message is received. The FSE will be restarted by the reception of a reference message.

When a potential time master reaches the Watch_Triggers after it has received any message but the reference message, it will assume a severe error (TTOST0.EL = "11", severity 3), set interrupt flag TTIR0.WT, switch off its CAN bus output, and enter the bus monitoring mode (CCCRi.MON set to '1'). In bus monitoring mode, it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

When no error is detected during initialization, the first reference message sets TTOST0.SYS = "01" (synchronizing), the second sets the TTCAN synchronization state (depending on its Next_is_Gap bit) to TTOST0.SYS = "11" (In_Schedule) or TTOST0.SYS = "10" (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

A potential time master is current time master (TTOST0.MS = "11") when it was the transmitter of the last reference message, else it is backup time master (TTOST0.MS = "10").

When all potential time masters have finished configuration, the node with the highest time master priority in the network will become the current time master.

40.3.3.3 TTCAN Gap Control

All functions related to Gap control apply only when the M_CAN is operated in external event- synchronized time-triggered mode (TTOCF0.GEN = ‘1’). In this operation mode the TTCAN message schedule may be interrupted by inserting Gaps between the basic cycles of the system matrix. All nodes connected to the CAN network have to be configured for external event- synchronized time-triggered operation.

During a Gap, all transmissions are stopped and the CAN bus remains idle. A Gap is finished when the next reference message starts a new basic cycle. A Gap starts at the end of a basic cycle that itself was started by a reference message with bit Next_is_Gap = ‘1’ e.g. Gaps are initiated by the current time master.

The current time master has two options to initiate a Gap. A Gap can be initiated under software control when the application program writes TTOCN0.NIG = ‘1’. The Next_is_Gap bit will be transmitted as ‘1’ with the next reference message. A Gap can also be initiated under hardware control when the application program enables the event trigger input by writing TTOCN0.GCS = ‘1’. When a reference message is started and TTOCN0.GCS is set, a HIGH level at the event trigger will set Next_is_Gap = ‘1’.

As soon as that reference message is completed, the TTOST0.WFE bit will announce the Gap to the time master as well as to the time slaves. The current basic cycle will continue until its last time window. The time after the last time window is the Gap time.

For the actual time master and the potential time masters, TTOST0.GSI will be set when the last basic cycle has finished and the Gap time starts. In nodes that are time slaves, bit TTOST0.GSI will remain at ‘0’.

When a potential time master is in synchronization state In_Gap (TTOST0.SYS = “10”), it has four options to intentionally finish a Gap:

Under software control by writing TTOCN0.FGP = ‘1’.

Under hardware control (TTOCN0.GCS = ‘1’) an edge from HIGH to LOW at the event-trigger sets TTOCN0.FGP and restarts the schedule.

The third option is a time-triggered restart. When TTOCN0.TMG = ‘1’, the next register time mark interrupt (TTIR0.RTMI = ‘1’) will set TTOCN0.FGP and start the reference message.

Finally any potential time master will finish a Gap when it reaches its Tx_Ref_Trigger_Gap, assuming that the event to synchronize on did not occur in time.

Neither of these options can cause a basic cycle to be interrupted with a reference message.

Setting of TTOCN0.FGP after the Gap time has started will start the transmission of a reference message immediately and will thereby synchronize the message schedule. When TTOCN0.FGP is set before the Gap time has started (while the basic cycle is still in progress), the next reference message is started at the end of the basic cycle, at the Tx_Ref_Trigger – there will be no Gap time in the message schedule.

In strictly time-triggered operation, bit Next_is_Gap = ‘1’ in the reference message will be ignored, as well as the event-trigger and the bits TTOCN0.NIG, TTOCN0.FGP, and TTOCN0.TMG.

40.3.3.4 Stop Watch

The stop watch function enables capturing of M_CAN internal time values (local time, cycle time, or global time) triggered by an external event.

To enable the stop watch function, the application program first has to define local time, cycle time, or global time as stop watch source via TTOCN0.SWS. When TTOCN0.SWS is ≠ “00” and TT Interrupt Register flag TTIR0.SWE is ‘0’, the actual value of the time selected by TTOCN0.SWS will be copied into TTCPT.SWV on the next rising/falling edge (as configured via TTOCN0.SWP) on stop watch trigger. This will set interrupt flag TTIR0.SWE. After the application program has read TTCPT.SWV, it may enable the next stop watch event by resetting TTIR0.SWE to ‘0’.

40.3.3.5 Local Time, Cycle Time, Global Time, and External Clock Synchronization

There are two possible levels in time-triggered CAN: Level 1 and Level 2. Level 1 only provides time-triggered operation using cycle time. Level 2 additionally provides increased synchronisation quality, global time and external clock synchronisation. In both levels, all timing features are based on a local time base - the local time.

The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU is represented by a 3-bit counter which can be regarded as a fractional part (three binary digits) of the local time. Generally, the 3-bit NTU counter is incremented 8 times each NTU. If the length of the NTU is shorter than 8 CAN clock periods (as may be configured in Level 1, or as a result of clock calibration in Level 2), the length of the NTU fraction is adapted, and the NTU counter is incremented only 4 times each NTU.

Figure 600 describes the synchronisation of the cycle time and global time, performed in the same manner by all TTCAN nodes, including the time master. Any message received or transmitted invokes a capture of the local time taken at the message's frame synchronisation event. This frame synchronisation event occurs at the sample point of each Start of Frame (SoF) bit and causes the local time to be stored as Sync_Mark. Sync_Marks and Ref_Marks are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref_Mark is updated from the Sync_Mark. The difference between Ref_Mark and Sync_Mark is the Cycle Sync Mark (Cycle Sync Mark = Sync_Mark - Ref_Mark) stored in register TTCSM. The most significant 16 bits of the difference between Ref_Mark and the actual value of the local time is the cycle time (Cycle Time = Local Time - Ref_Mark).

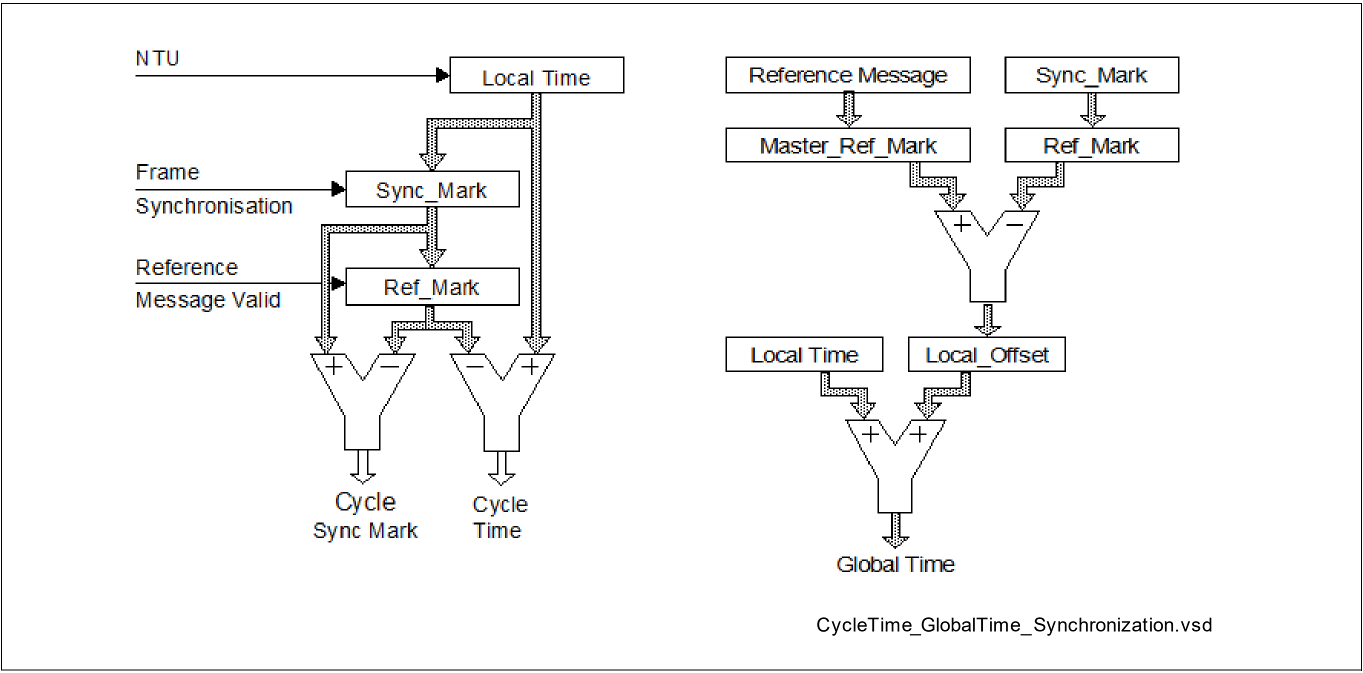


Figure 600 Cycle Time and Global Time Synchronization

The cycle time that can be read from TTCTC0.CT is the difference of the node's local time and Ref_Mark, both synchronized into the Host clock domain and truncated to 16 bits.

The global time exists for TTCAN Level 0 and Level 2 only, in Level 1 it is invalid. The node's view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master will use its own local time as global time. The time master establishes its own local time as global time by transmitting its own Ref_Marks as Master_Ref_Marks in the reference message (bytes 3, 4). The global time that can be read from TTLGT0.GT is the sum of the node's local time and its local offset, both synchronized into the Host clock domain and truncated to 16 bits. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local Ref_Mark with the received Master_Ref_Mark (see **Figure 600**). The node's view of the global time is local time + local offset. In a potential time master that has never received another time master's reference message, Local_Offset will be zero. When a node becomes the current time master after first having received other reference messages, Local_Offset will be frozen at its last value. In the time receiving nodes, Local_Offset may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signalled by Disc_Bit in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register TTLGT is smoothed by a low-pass filtering to have a continuous monotonic value.

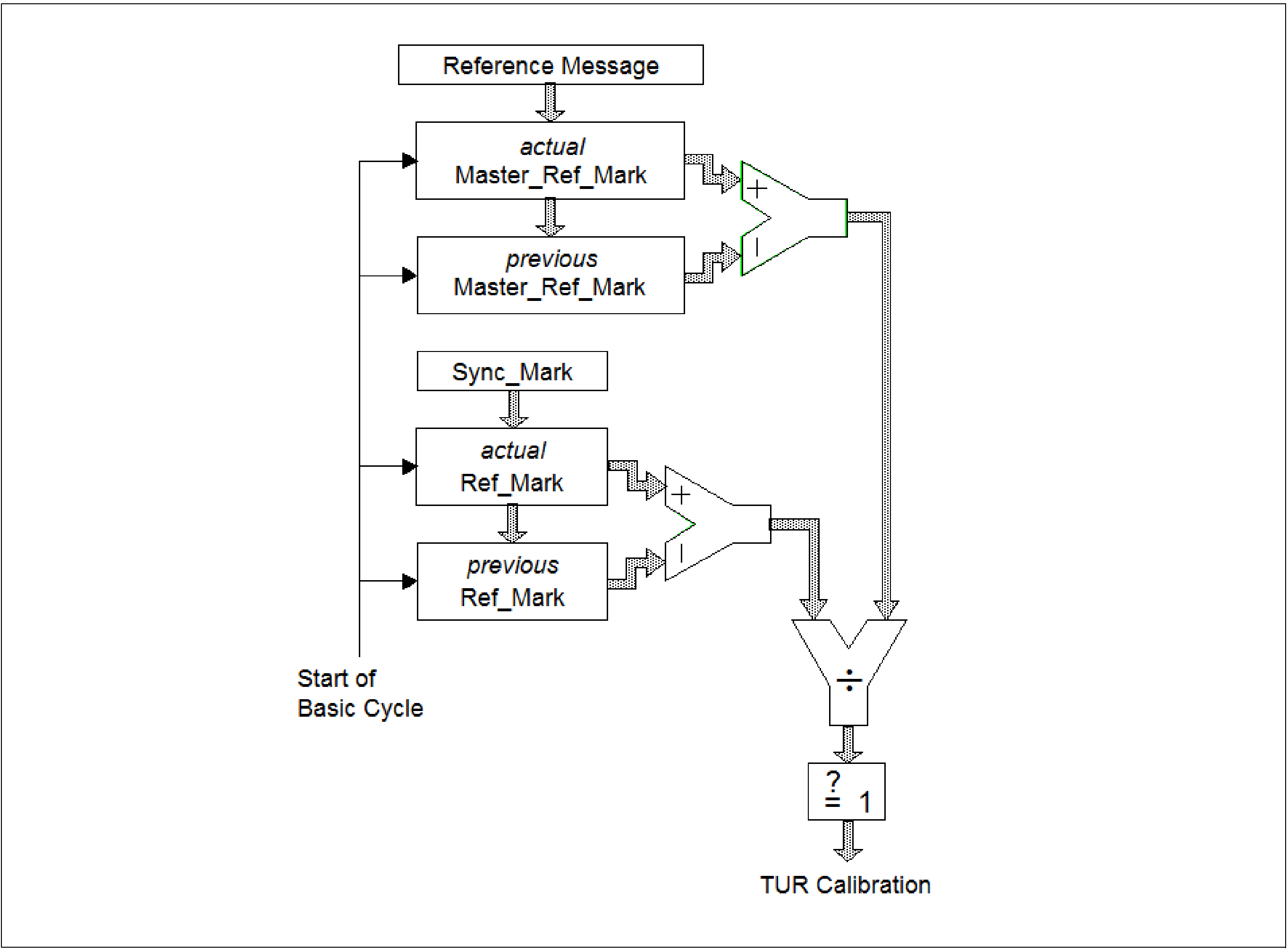


Figure 601 TTCAN Level 0 and Level 2 Drift Compensation

Figure 601 describes how in TTCAN Level 0,2 each time receiving node compensates the drift between its own local clock and the time master's clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the Disc_Bit in the reference message is not set, a new value for TURNA0.NAV is calculated. If the Synchronisation Deviation SD = |NC - TURNA0.NAV| ≤ SDL (Synchronisation Deviation Limit), the new value for TURNA0.NAV takes effect. Else the automatic drift compensation is suspended.

In TTCAN Level 0 and Level 2, TTOST0.QCS indicates whether the automatic drift compensation is active or suspended. In TTCAN Level 1, TTOST0.QCS is always '1'.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit TTOCF0.EECS.

The stop watch function (see **Section 40.3.3.4**) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated Numerator Configuration Low to TURCF0.NCL (TURCF0.DC cannot be updated during operation). The new value takes effect by writing TTOCN0.ECS to '1'.

The global time phase is adjusted by first writing the phase offset into the TT Global Time Preset register TTGTP0. The new value takes effect by writing TTOCN0.SGT to '1'. The first reference message transmitted after the global time phase adjustment will have the Disc_Bit set to '1'.

TTOST0.QGTP shows whether the node's global time is in phase with the time master's global time. TTOST0.QGTP is permanently '0' in TTCAN Level 1 and when the Synchronisation Deviation Limit is exceeded in TTCAN Level 0,2 (TTOST0.QCS = '0'). It is temporarily '0' while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a Disc_Bit = '1' or when TTOST0.QCS='0'.

40.3.3.6 TTCAN Error Level

The ISO 11898-4 specifies four levels of error severity:

S0 - No Error

S1 - Warning
Only notification of application, reaction application-specific.

S2 Error
Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (i.e. no data or remote frames may be started). Potential time masters still transmit reference messages with the Reference Trigger Offset TTOST0.RTO set to the maximum value of 127.

S3 - Severe Error
Notification of application. All CAN bus operations are stopped, i.e. transmission of dominant bits is not allowed, and CCCRi.MON is set. The S3 error condition remains active until the application updates the configuration (set CCCRi.CCE).

If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by TTIR0.ELC. The error level is monitored by TTOST0.EL.

The M_CAN signals the following error conditions as required by ISO 11898-4:

Config_Error (S3)

Sets Error Level TTOST0.EL to "11" when a merged arbitrating time window is not properly closed or when there is a Tx_Trigger with a time mark beyond the Tx_Ref_Trigger.

Watch_Trigger_Reached (S3)

Sets Error Level TTOST0.EL to "11" when a watch trigger was reached because the reference message is missing.

Application_Watchdog (S3)

Sets Error Level TTOST0.EL to "11" when the application failed to serve the application watchdog. The application watchdog is configured via TTOCF0.AWL. It is served by reading register TTOST0. When the watchdog is not served in time, bit TTOST0.AWE and interrupt flag TTIR0.AW are set, all TTCAN communication is stopped, and the M_CAN is set into bus monitoring mode (CCCRi.MON set to '1').

CAN_Bus_Off (S3)

Entering CAN_Bus_Off state sets error level TTOST0.EL to "11". CAN_Bus_Off state is signalled by PSRi.BO = '1' and CCCRi.INIT = '1'.

Scheduling_Error_2 (S2)

Sets Error Level TTOST0.EL to "10" if the MSC of one Tx_Trigger has reached 7. In addition interrupt flag TTIR0.SE2 is set. The Error Level TTOST0.EL is reset to "00" at the beginning of a matrix cycle when no Tx_Trigger has an MSC of 7 in the preceding matrix cycle.

Tx_Overflow (S2)

Sets Error Level TTOST0.EL to "10" when the Tx count is equal or higher than the expected number of Tx_Triggers TTMLM.ENTT and a Tx_Trigger event occurs. In addition interrupt flag TTIR0.TXO is set. The Error Level TTOST0.EL is reset to "00" when the Tx count is no more than TTMLM.ENTT at the start of a new matrix cycle.

Scheduling_Error_1 (S1)

Sets Error Level TTOST0.EL to "01" if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than 2, or if one of the MSCs of an exclusive Rx_Trigger has reached 7. In addition interrupt flag TTIR0.SE1 is set. If within one matrix cycle none of these conditions is valid, the Error Level TTOST0.EL is reset to "00".

Tx_Underflow (S1)

Sets Error Level TTOST0.EL to "01" when the Tx count is less than the expected number of Tx_Triggers TTMLM.ENTT at the start of a new matrix cycle. In addition interrupt flag TTIR0.TXU is set. The Error Level TTOST0.EL is reset to "00" when the Tx count is at least TTMLM.ENTT at the start of a new matrix cycle.

40.3.3.7 TTCAN Message Handling

40.3.3.7.1 Reference Message

For potential time masters the identifier of the reference message is configured via TTRMC.RID. No dedicated Tx Buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte (TTCAN Level 1) resp. the first four data bytes (TTCAN Level 0 and Level 2) will be provided by the FSE.

In case the reference message Payload Select TTRMC.RMPS is set, the rest of the reference message's payload (Level 1: bytes 2-8, Level 0,2: bytes 5-6) is taken from Tx Buffer 0. In this case the data length DLC code from message buffer 0 is used.

Table 377 Number of Data Bytes transmitted with a reference messages

TTRMC.RMPS	TXBRPi.TRP0	Level 0	Level 1	Level 2
0	0	4	1	4
0	1	4	1	4
1	0	4	1	4
1	1	4 + MB0	1 + MB0	4 + MB0

To send additional payload with the reference message in Level 1 a DLC > 1 has to be configured, for Level 0,2 a DLC > 4 is required. In addition the transmission request pending bit TXBRPi.TRP0 of message buffer 0 must be set (see **Table 377**). In case bit TXBRPi.TRP0 is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the Reference Identifier TTRMC.RID is used.

40.3.3.7.2 Message Reception

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (see **Chapter 40.3.2.4**).

The Message Status Count MSC is part of the corresponding trigger memory element and has to be initialized to zero during configuration. It is updated while the M_CAN is in synchronization states In_Gap or In_Schedule. The update happens at the message's Rx_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the MSC is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the MSC is incremented by one. At each Rx_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx_Trigger should be set to a value where it is ensured that reception and acceptance filtering for the targeted message has completed. This has to take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx_Trigger for the reference message.

40.3.3.7.3 Message Transmission

For time-triggered message transmission the M_CAN supplies 32 dedicated Tx buffers (see **Chapter 40.3.2.5.2**). A Tx FIFO or Tx queue is not available when the M_CAN is configured for time-triggered operation (TTOCF0.OM = “01” or “10”).

Each Tx_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle.

The application program has to update the data regularly and on time, synchronized to the cycle time. The Host CPU is responsible that no partially updated messages are transmitted. To assure this the Host has to proceed in the following way:

Tx_Trigger_Single / Tx_Trigger_Merged / Tx_Trigger_Arbitration

- Check whether the previous transmission has completed by reading TXBTO
- Update the Tx buffer’s configuration and/or payload
- Issue an "Add Request" to set the Tx Buffer Request Pending bit

Tx_Trigger_Continous

- Issue a Cancellation Request to reset the Tx Buffer Request Pending bit
- Check whether the cancellation has finished by reading TXBCF
- Update Tx buffer’s configuration and/or payload
- Issue an "Add Request" to set the Tx Buffer Request Pending bit

The message’s MSC stored with the corresponding Tx_Trigger provides information on the success of the transmission.

The MSC is incremented by one when the transmission could not be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and could not be completed successfully. The MSC is decremented by one when the message was transmitted successfully or when the message could have been started within its transmit enable window but was not started because transmission was disabled (M_CAN in Error Level S2 or Host has disabled this particular message).

The Tx buffers may be managed dynamically, i.e. several messages with different identifiers may share the same Tx buffer element. In this case the Host has to assure that no transmission request is pending for the Tx buffer element to be reconfigured by checking TXBRPi.

If a Tx buffer with pending transmission request should be updated, the Host first has to issue a cancellation request and check whether the cancellation has completed by reading TXBCF before it starts updating.

The Tx Handler will transfer a message from the Message RAM to its intermediate output buffer at the trigger element which becomes active immediately before the Tx_Trigger element which defines the beginning of the transmit window. During and after the transfer time the transmit message may not be updated and its TXBRP bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx_Trigger. This may be e.g. a Time_Base_Trigger which need not cause any other action. The difference in time marks between the Tx_Trigger and the preceding trigger has to be large enough to guarantee that the Tx Handler can read four words from the Message RAM even at high RAM access load from other modules.

Transmission in Exclusive Time Windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Single or Tx_Trigger_Continous. There is no arbitration on the bus with messages from other nodes. The MSC is updated according the result of the transmission attempt. After successful transmission started by a Tx_Trigger_Single the respective Tx Buffer Request Pending bit is reset. After successful transmission started by a Tx_Trigger_Continous the respective Tx Buffer Request Pending remains set. When the transmission was not successful due to disturbances, it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in Arbitrating Time Windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Arbitration. Several nodes may start to transmit at the same time. In this case the message has to arbitrate with the messages from other nodes. The MSC is not updated. When the transmission was not successful (lost arbitration or disturbance), it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in Merged Arbitrating Time Windows

The purpose of a merged arbitrating time window is to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a single node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission will not be started if the corresponding Transmission Request Pending flag was reset by a successful Tx cancellation.

In single transmit windows, the Tx Handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmissions will be attempted in the sequence defined by the trigger list. If the time mark of a fourth message is read before the first is transmitted (or cancelled by the Host), the fourth request will be ignored.

The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window will be started in the order of their Tx_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if there is competing bus traffic. This has to be considered for the configuration of the trigger list. Time_Base_Triggers may be placed between consecutive Tx_Triggers to define the time until the data of the corresponding Tx Buffer needs to be updated.

40.3.3.8 TTCAN Interrupt and Error Handling

The TT Interrupt Register TTIR consists of four segments. Each interrupt can be enabled separately by the corresponding bit in the TT Interrupt Enable register TTIE. The flags remain set until the Host clears them. A flag is cleared by writing a “1” to the corresponding bit position.

The first segment consists of flags CER, AW, WT, and IWT. Each flag indicates a severe error condition where the CAN communication is stopped. With the exception of IWT, these error conditions require a re-configuration of the M_CAN module before the communication can be restarted.

The second segment consists of flags ELC, SE1, SE2, TXO, TXU, and GTE. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, e.g. by disturbances on the CAN bus, they will be handled by the TTCAN protocol’s failure handling and do not require intervention by the application program.

The third segment consists of flags GTD, GTW, SWE, TTMI, and RTMI. The first two flags are controlled by global time events (Level 0,2 only) that require a reaction by the application program. With a Stop Watch Event triggered by a rising/falling edge the internal time values are captured. The Trigger Time Mark Interrupt notifies the application that a specific Time_Base_Trigger is reached. The Register Time Mark Interrupt signals that the time referenced by TTOCN0.TMC (Cycle, Local, or Global) equals time mark TTTMK.TM. It can also be used to finish a Gap.

The fourth segment consists of flags SOG, CSM, SMC, and SBC. These flags provide a means to synchronize the application program to the communication schedule.

40.3.3.9 Level 0

TTCAN Level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in TTCAN Level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1.

Level 0 operation is configured via TTOCF0.OM = “11”. In this mode the M_CAN operates in event-driven CAN communication, there is no fixed schedule, the configuration of TTOCF0.GEN is ignored. External event-synchronized operation is not available in Level 0. A synchronized time base is maintained by transmission of reference messages.

In Level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (TTIR0.TTMI) is set when the cycle time has reached TTOCF0.IRTO * 0x200, it reminds the Host to set a transmission request for message buffer 0. The Watch_Trigger interrupt flag (TTIR0.WT) is set when the cycle time has reached 0xFF00. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks.

Register time mark interrupts (TTIR0.RTMI) are also possible.

The reference message is configured as for Level 2 operation. Received reference messages are recognized by the identifier configured in register TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the Host sets a transmission request for message buffer 0, there is no reference trigger offset.

Level 0 operation is configured via:

- TTRMC
- TTOCF except EVTP, AWL, GEN
- TTMLM except ENTT, TXEW
- TURCF

Level 0 operation is controlled via:

- TTOCN except NIG, TMG, FGP, GCS, TTMIE
- TTGTP
- TTTMK
- TTIR excluding bits CER, AW, IWT SE2, SE1, TXO, TXU, SOG (no function)
- TTIR the following bits have changed function
  - TTMI not defined by trigger memory - activated at cycle time TTOCF0.IRTO * 0x200
  - WT not defined by trigger memory - activated at cycle time 0xFF00

Level 0 operation is signalled via:

- TTOST excluding bits AWE, WFE, GSI, GFI, RTO (no function)

40.3.3.9.1 Synchronizing

Figure 602 below describes the states and state transitions in TTCAN Level 0 operation. Level 0 has no In_Gap state.

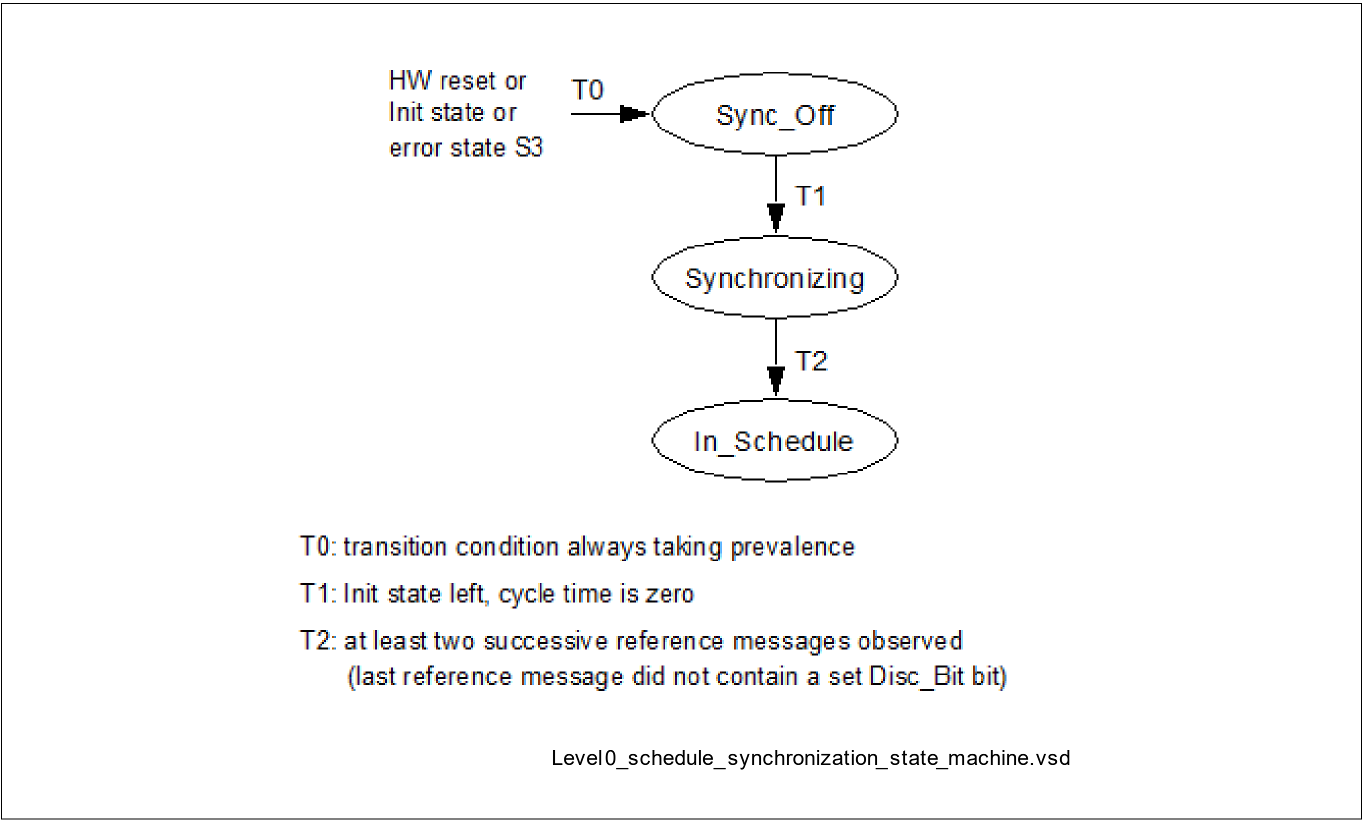

HW reset or
Init state or
error state S3
T0
Sync_Off
T1
Synchronizing
T2
In_Schedule
T0: transition condition always taking prevalence
T1: Init state left, cycle time is zero
T2: at least two successive reference messages observed
(last reference message did not contain a set Disc_Bit bit)
Level0_schedule_synchronization_state_machine.vsd

Figure 602 Level 0 schedule synchronization state machine

40.3.3.9.2 Handling of Error Levels

During Level 0 operation only the following error conditions may occur:

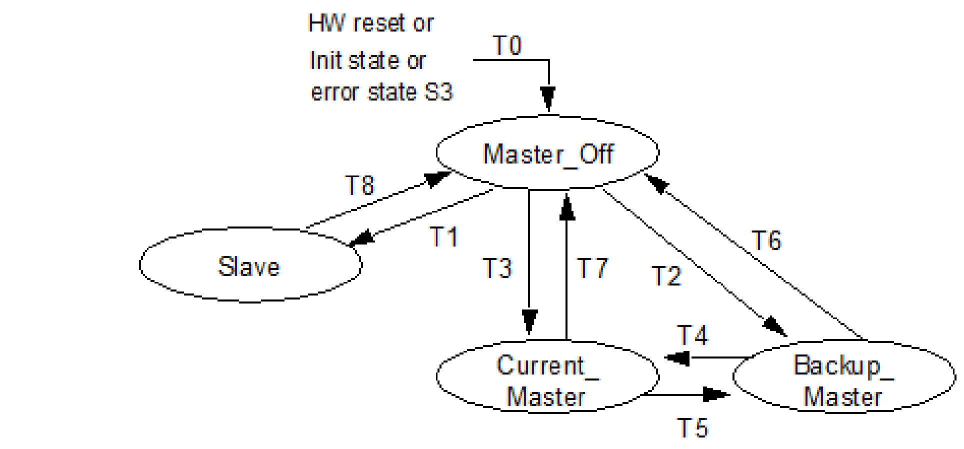
- Watch_Trigger_Reached (S3), reached cycle time $FF00_H$
- CAN_Bus_Off (S3)

Since no S1 and S2 error are possible, the error level can only switch between S0 (No Error) and S3 (Severe Error). In TTCAN Level 0 an S3 error is handled differently. When error level S3 is reached, both TTOST0.SYS and TTOST0.MS are reset, and interrupt flags TTIR0.GTE and TTIR0.GTD are set.

When error level S3 (TTOST0.EL = “11”) is entered, bus monitoring mode is, contrary to TTCAN Level 1 and Level 2, not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.

40.3.3.9.3 Master Slave Relation

Figure 603 below describes the master slave relation in TTCAN Level 0. In case of an S3 error the M_CAN returns to state Master_Off.



T0: transition condition always taking prevalence

T1: reference message observed when not potential time master

T2: reference message observed with master priority ≠ own master priority, error state ≠ S3

T3: reference message observed with master priority = own master priority, error state ≠ S3

T4: reference message observed with own master priority

T5: reference message observed with master priority higher than own master priority

T6: error state S3

T7: error state S3

T8: error state S3

Level0_master_to_slave_relation.vsd

Figure 603 Level 0 master to slave relation

40.3.3.10 Synchronization to external Time Schedule

This feature can be used to synchronize the phase of the M_CAN's schedule to an external schedule (e.g. that of a second TTCAN network or FlexRay network). It is applicable only when the M_CAN is current time master (TTOST0.MS = "11").

External synchronization is controlled by event trigger input. If bit TTOCN0.ESCN is set, a rising edge at the event trigger the M_CAN compares its actual cycle time with the target phase value configured by TTGTP0.CTP.

Before setting TTOCN0.ESCN the Host has to adapt the phases of the two time schedules e.g. by using the TTCAN gap control (see **Section 40.3.3.3**). When the Host sets TTOCN0.ESCN, TTOST0.SPL is set.

If the difference between the cycle time and theTTIR0.target phase value TTGTP0.CTP at the rising edge at the event trigger is greater than 9 NTU, the phase lock bit TTOST0.SPL is reset, and interrupt flag TTIR0.CSM is set. TTOST0.SPL is also reset (and TTIR0.CSM is set), when another node becomes time master.

If both TTOST0.SPL and TTOCN0.ESCN are set, and if the difference between the cycle time and the target phase value TTGTP0.CTP at the rising edge at the event trigger is less or equal 9 NTU, the phase lock bit TTOST0.SPL remains set, and the measured difference is used as reference trigger offset value to adjust the phase at the next transmitted reference message.