1. How to read this chapter

This chapter describes the CAN controllers for the following LPC23XX parts:

- LPC2361/62
- LPC2364/66/68
- LPC2378
- LPC2387
- LPC2388

LPC2365/67 and LPC2377 do **not** include CAN controllers.

2. Basic configuration

The CAN1/2 peripherals are configured using the following registers:

1. Power: In the PCONP register (Table 4–56), set bits PCAN1/2.

   Remark: On reset, the CAN1/2 blocks are disabled (PCAN1/2 = 0).

2. Peripheral clock: In the PCLK_SEL0 register (Table 4–49), select PCLK_CAN1/2 and, for the acceptance filter, PCLK_ACF. PCLK_CAN1/2 and PCLK_ACF must be set to the same value.

   Remark: If CAN baudrates above 100 kbit/s (see Table 12–225) are needed, do not select the IRC as the clock source (see Table 4–35).

3. Wakeup: Use the INTWAKE register (Table 4–55) to enable the CAN controllers to wake up the microcontroller from Power-down mode.
4. Pins: Select CAN1/2 pins and pin modes in registers PINSELn and PINMODEn (see Section 9–5).
5. Interrupts: CAN interrupts are enabled using the CAN1/2IER registers (Table 12–224). Interrupts are enabled in the VIC using the VICIntEnable register (Table 6–76).
6. CAN controller initialization: see CANMOD register (Table 12–220).

3. Introduction

Controller Area Network (CAN) is the definition of a high performance communication protocol for serial data communication. The CAN Controller is designed to provide a full implementation of the CAN-Protocol according to the CAN Specification Version 2.0B. Microcontrollers with this on-chip CAN controller are used to build powerful local networks by supporting distributed real-time control with a very high level of security. The applications are automotive, industrial environments, and high speed networks as well as low cost multiplex wiring. The result is a strongly reduced wiring harness and enhanced diagnostic and supervisory capabilities.

The CAN block is intended to support multiple CAN buses simultaneously, allowing the device to be used as a gateway, switch, or router among a number of CAN buses in various applications.

The CAN module consists of two elements: the controller and the Acceptance Filter. All registers and the RAM are accessed as 32 bit words.

4. Features

4.1 General CAN features

- Compatible with *CAN specification 2.0B, ISO 11898-1*.
- Multi-master architecture with non destructive bit-wise arbitration.
- Bus access priority determined by the message identifier (11-bit or 29-bit).
- Guaranteed latency time for high priority messages.
- Programmable transfer rate (up to 1 Mbit/s).
- Multicast and broadcast message facility.
- Data length from 0 up to 8 bytes.
- Powerful error handling capability.
- Non-return-to-zero (NRZ) coding/decoding with bit stuffing.

4.2 CAN controller features

- 2 CAN controllers and buses.
- Supports 11-bit identifier as well as 29-bit identifier.
- Double Receive Buffer and Triple Transmit Buffer.
- Programmable Error Warning Limit and Error Counters with read/write access.
- Arbitration Lost Capture and Error Code Capture with detailed bit position.
- Single Shot Transmission (no re-transmission).
- Listen Only Mode (no acknowledge, no active error flags).
- Reception of "own" messages (Self Reception Request).

4.3 Acceptance filter features

- Fast hardware implemented search algorithm supporting a large number of CAN identifiers.
- Global Acceptance Filter recognizes 11 and 29 bit Rx Identifiers for all CAN buses.
- Allows definition of explicit and groups for 11-bit and 29-bit CAN identifiers.
- Acceptance Filter can provide FullCAN-style automatic reception for selected Standard Identifiers.

5. Pin description

Table 215. CAN Pin descriptions

Pin Name	Type	Description
RD2/1	Inputs	Serial Inputs. From CAN transceivers.
TD2/1	Outputs	Serial Outputs. To CAN transceivers.

6. CAN controller architecture

The CAN Controller is a complete serial interface with both Transmit and Receive Buffers but without Acceptance Filter. CAN Identifier filtering is done for all CAN channels in a separate block (Acceptance Filter). Except for message buffering and acceptance filtering the functionality is similar to the PeliCAN concept.

The CAN Controller Block includes interfaces to the following blocks:

- APB Interface
- Acceptance Filter
- Vectored Interrupt Controller (VIC)
- CAN Transceiver
- Common Status Registers

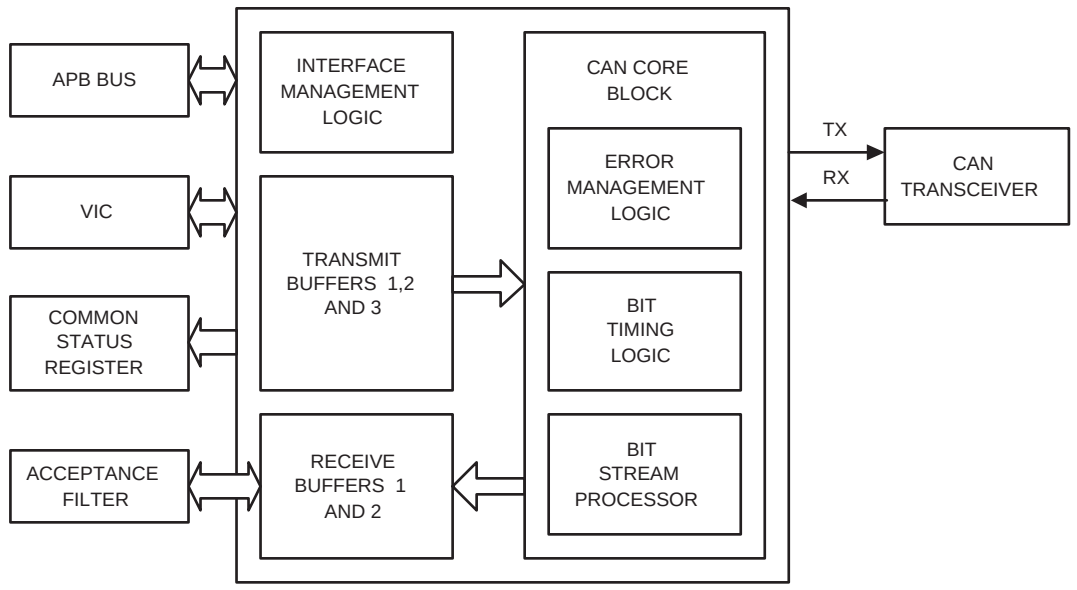

Fig 41. CAN controller block diagram

6.1 APB interface block (AIB)

The APB Interface Block provides access to all CAN Controller registers.

6.2 Interface management logic (IML)

The Interface Management Logic interprets commands from the CPU, controls internal addressing of the CAN Registers and provides interrupts and status information to the CPU.

6.3 Transmit Buffers (TXB)

The TXB represents a Triple Transmit Buffer, which is the interface between the Interface Management Logic (IML) and the Bit Stream Processor (BSP). Each Transmit Buffer is able to store a complete message which can be transmitted over the CAN network. This buffer is written by the CPU and read out by the BSP.

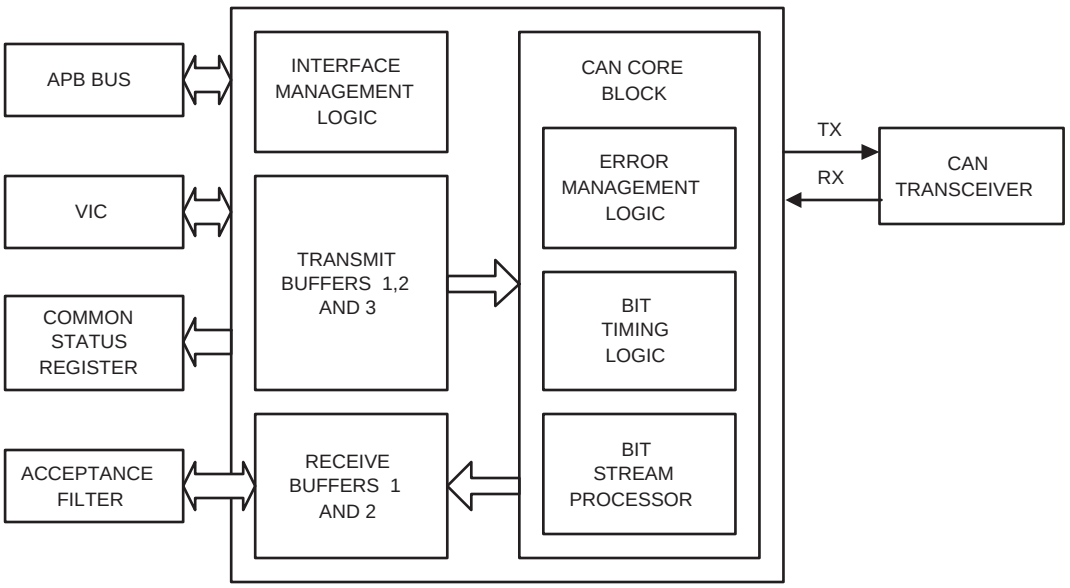

Fig 42. Transmit buffer layout for standard and extended frame format configurations

6.4 Receive Buffer (RXB)

The Receive Buffer (RXB) represents a CPU accessible Double Receive Buffer. It is located between the CAN Controller Core Block and APB Interface Block and stores all received messages from the CAN Bus line. With the help of this Double Receive Buffer concept the CPU is able to process one message while another message is being received.

The global layout of the Receive Buffer is very similar to the Transmit Buffer described earlier. Identifier, Frame Format, Remote Transmission Request bit and Data Length Code have the same meaning as described for the Transmit Buffer. In addition, the Receive Buffer includes an ID Index field (see Section 12–8.9.1 “ID index field”).

The received Data Length Code represents the real transmitted Data Length Code, which may be greater than 8 depending on transmitting CAN node. Nevertheless, the maximum number of received data bytes is 8. This should be taken into account by reading a

message from the Receive Buffer. If there is not enough space for a new message within the Receive Buffer, the CAN Controller generates a Data Overrun condition when this message becomes valid and the acceptance test was positive. A message that is partly written into the Receive Buffer (when the Data Overrun situation occurs) is deleted. This situation is signalled to the CPU via the Status Register and the Data Overrun Interrupt, if enabled.

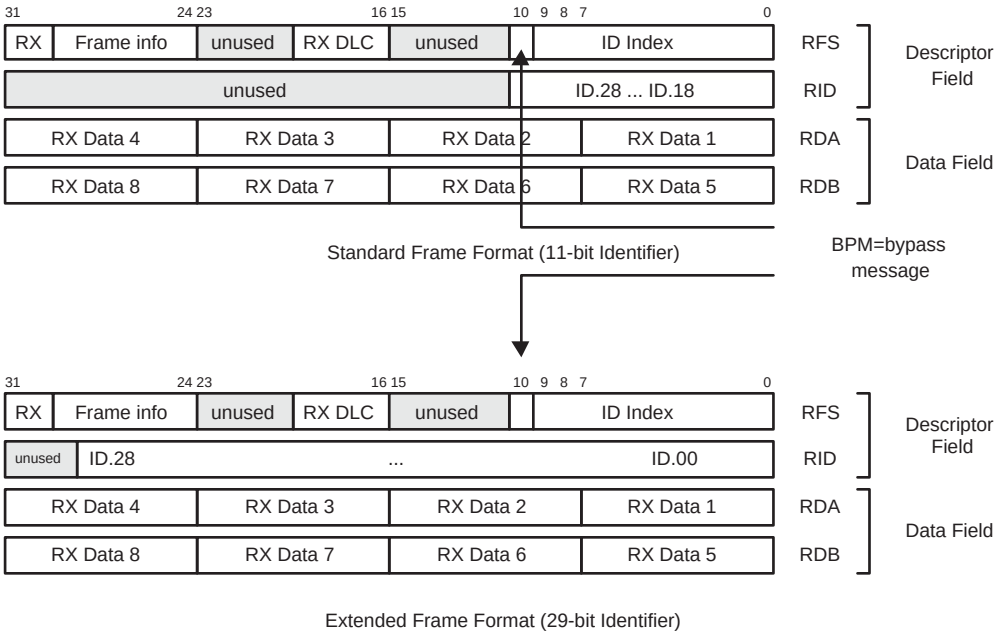

Fig 43. Receive buffer layout for standard and extended frame format configurations

6.5 Error Management Logic (EML)

The EML is responsible for the error confinement. It gets error announcements from the BSP and then informs the BSP and IML about error statistics.

6.6 Bit Timing Logic (BTL)

The Bit Timing Logic monitors the serial CAN Bus line and handles the Bus line related bit timing. It synchronizes to the bit stream on the CAN Bus on a "recessive" to "dominant" Bus line transition at the beginning of a message (hard synchronization) and re-synchronizes on further transitions during the reception of a message (soft synchronization). The BTL also provides programmable time segments to compensate for the propagation delay times and phase shifts (e.g. due to oscillator drifts) and to define the sample point and the number of samples to be taken within a bit time.

6.7 Bit Stream Processor (BSP)

The Bit Stream Processor is a sequencer, controlling the data stream between the Transmit Buffer, Receive Buffers and the CAN Bus. It also performs the error detection, arbitration, stuffing and error handling on the CAN Bus.

6.8 CAN controller self-tests

The CAN controller of the LPC2000 family supports two different options for self-tests:

- Global Self-Test (setting the self reception request bit in normal Operating Mode)
- Local Self-Test (setting the self reception request bit in Self Test Mode)

Both self-tests are using the ‘Self Reception’ feature of the CAN Controller. With the Self Reception Request, the transmitted message is also received and stored in the receive buffer. Therefore the acceptance filter has to be configured accordingly. As soon as the CAN message is transmitted, a transmit and a receive interrupt are generated, if enabled.

Global self test

A Global Self-Test can for example be used to verify the chosen configuration of the CAN Controller in a given CAN system. As shown in Figure 12–44, at least one other CAN node, which is acknowledging each CAN message has to be connected to the CAN bus.

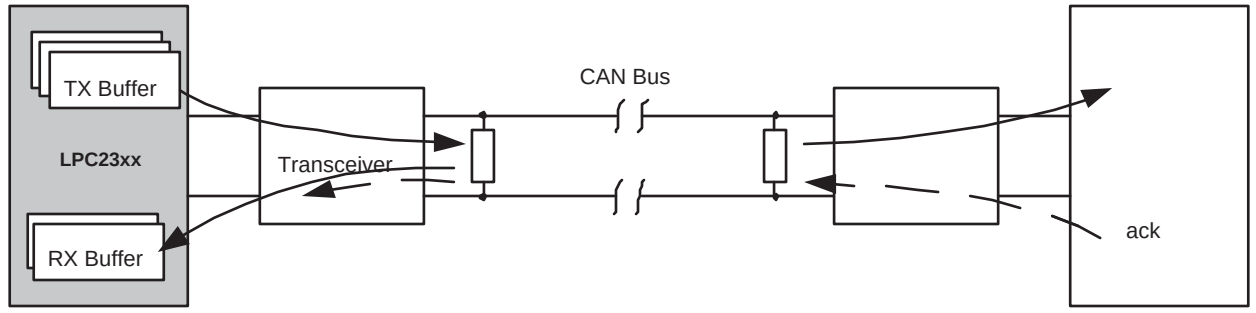

Fig 44. Global Self-Test (high-speed CAN Bus example)

Initiating a Global Self-Test is similar to a normal CAN transmission. In this case the transmission of a CAN message(s) is initiated by setting Self Reception Request bit (SRR) in conjunction with the selected Message Buffer bits (STB3, STB2, STB1) in the CAN Controller Command register (CANCMR).

Local self test

The Local Self-Test perfectly fits for single node tests. In this case an acknowledge from other nodes is not needed. As shown in the Figure below, a CAN transceiver with an appropriate CAN bus termination has to be connected to the LPC. The CAN Controller has to be put into the 'Self Test Mode' by setting the STM bit in the CAN Controller Mode register (CANMOD). Hint: Setting the Self Test Mode bit (STM) is possible only when the CAN Controller is in Reset Mode.

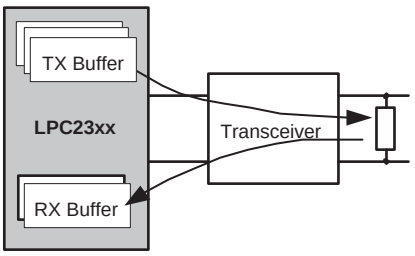

Fig 45. Local Self-Test (high-speed CAN Bus example)

A message transmission is initiated by setting Self Reception Request bit (SRR) in conjunction with the selected Message Buffer(s) (STB3, STB2, STB1).

7. Memory map of the CAN block

The CAN Controllers and Acceptance Filter occupy a number of APB slots, as follows:

Table 216. Memory Map of the CAN Block

Address Range	Used for
0xE003 8000 - 0xE003 87FF	Acceptance Filter RAM.
0xE003 C000 - 0xE003 C017	Acceptance Filter Registers.
0xE004 0000 - 0xE004 000B	Central CAN Registers.
0xE004 4000 - 0xE004 405F	CAN Controller 1 Registers.
0xE004 8000 - 0xE004 805F	CAN Controller 2 Registers.

8. CAN controller registers

CAN block implements the registers shown in Table 12–217 and Table 12–218. More detailed descriptions follow.

Table 217. CAN acceptance filter and central CAN registers

Name	Description	Access	Reset Value	Address
AFMR	Acceptance Filter Register	R/W	1	0xE003 C000
SFF_sa	Standard Frame Individual Start Address Register	R/W	0	0xE003 C004
SFF_GRP_sa	Standard Frame Group Start Address Register	R/W	0	0xE003 C008
EFF_sa	Extended Frame Start Address Register	R/W	0	0xE003 C00C
EFF_GRP_sa	Extended Frame Group Start Address Register	R/W	0	0xE003 C010
ENDofTable	End of AF Tables register	R/W	0	0xE003 C014
LUTerrAd	LUT Error Address register	RO	0	0xE003 C018
LUTerr	LUT Error Register	RO	0	0xE003 C01C
FCANIE	FullCAN interrupt enable register	R/W	0	0xE003 C020
FCANIC0	FullCAN interrupt and capture register 0	R/W	0	0xE003 C024
FCANIC1	FullCAN interrupt and capture register 1	R/W	0	0xE003 C028

Table 217. CAN acceptance filter and central CAN registers

Name	Description	Access	Reset Value	Address
CANTxSR	CAN Central Transmit Status Register	RO	0x0003 0300	0xE004 0000
CANRxSR	CAN Central Receive Status Register	RO	0	0xE004 0004
CANMSR	CAN Central Miscellaneous Register	RO	0	0xE004 0008

Table 218. CAN1 and CAN2 controller register map

Generic Name	Description	Access	CAN1 Register Address & Name	CAN2 Register Address & Name
MOD	Controls the operating mode of the CAN Controller.	R/W	CAN1MOD - 0xE004 4000	CAN2MOD - 0xE004 8000
CMR	Command bits that affect the state of the CAN Controller	WO	CAN1CMR - 0xE004 4004	CAN2CMR - 0xE004 8004
GSR	Global Controller Status and Error Counters	RO[1]	CAN1GSR - 0xE004 4008	CAN2GSR - 0xE004 8008
ICR	Interrupt status, Arbitration Lost Capture, Error Code Capture	RO	CAN1ICR - 0xE004 400C	CAN2ICR - 0xE004 800C
IER	Interrupt Enable	R/W	CAN1IER - 0xE004 4010	CAN2IER - 0xE004 8010
BTR	Bus Timing	R/W[2]	CAN1BTR - 0xE004 4014	CAN2BTR - 0xE004 8014
EWL	Error Warning Limit	R/W[2]	CAN1EWL - 0xE004 4018	CAN2EWL - 0xE004 8018
SR	Status Register	RO	CAN1SR - 0xE004 401C	CAN2SR - 0xE004 801C
RFS	Receive frame status	R/W[2]	CAN1RFS - 0xE004 4020	CAN2RFS - 0xE004 8020
RID	Received Identifier	R/W[2]	CAN1RID - 0xE004 4024	CAN2RID - 0xE004 8024
RDA	Received data bytes 1-4	R/W[2]	CAN1RDA - 0xE004 4028	CAN2RDA - 0xE004 8028
RDB	Received data bytes 5-8	R/W[2]	CAN1RDB - 0xE004 402C	CAN2RDB - 0xE004 802C
TFI1	Transmit frame info (Tx Buffer 1)	R/W	CAN1TFI1 - 0xE004 4030	CAN2TFI1 - 0xE004 8030
TID1	Transmit Identifier (Tx Buffer 1)	R/W	CAN1TID1 - 0xE004 4034	CAN2TID1 - 0xE004 8034
TDA1	Transmit data bytes 1-4 (Tx Buffer 1)	R/W	CAN1TDA1 - 0xE004 4038	CAN2TDA1 - 0xE004 8038
TDB1	Transmit data bytes 5-8 (Tx Buffer 1)	R/W	CAN1TDB1- 0xE004 403C CAN2TDB1- 0xE004 803C	CAN1TDB1 - 0xE004 403C CAN2TDB1 - 0xE004 803C
TFI2	Transmit frame info (Tx Buffer 2)	R/W	CAN1TFI2 - 0xE004 4040 CAN2TFI2 - 0xE004 8040	CAN1TFI2 - 0xE004 4040 CAN2TFI2 - 0xE004 8040
TID2	Transmit Identifier (Tx Buffer 2)	R/W	CAN1TID2 - 0xE004 4044 CAN2TID2 - 0xE004 8044	CAN1TID2 - 0xE004 4044 CAN2TID2 - 0xE004 8044
TDA2	Transmit data bytes 1-4 (Tx Buffer 2)	R/W	CAN1TDA2 - 0xE004 4048 CAN2TDA2 - 0xE004 8048	CAN1TDA2 - 0xE004 4048 CAN2TDA2 - 0xE004 8048
TDB2	Transmit data bytes 5-8 (Tx Buffer 2)	R/W	CAN1TDB2 - 0xE004 404C CAN2TDB2 - 0xE004 804C	CAN1TDB2 - 0xE004 404C CAN2TDB2 - 0xE004 804C
TFI3	Transmit frame info (Tx Buffer 3)	R/W	CAN1TFI3 - 0xE004 4050 CAN2TFI3 - 0xE004 8050	CAN1TFI3 - 0xE004 4050 CAN2TFI3 - 0xE004 8050
TID3	Transmit Identifier (Tx Buffer 3)	R/W	CAN1TID3 - 0xE004 4054 CAN2TID3 - 0xE004 8054	CAN1TID3 - 0xE004 4054 CAN2TID3 - 0xE004 8054
TDA3	Transmit data bytes 1-4 (Tx Buffer 3)	R/W	CAN1TDA3 - 0xE004 4058 CAN2TDA3 - 0xE004 8058	CAN1TDA3 - 0xE004 4058 CAN2TDA3 - 0xE004 8058
TDB3	Transmit data bytes 5-8 (Tx Buffer 3)	R/W	CAN1TDB3 - 0xE004 405C CAN2TDB3 - 0xE004 805C	CAN1TDB3 - 0xE004 405C CAN2TDB3 - 0xE004 805C

[1] The error counters can only be written when RM in CANMOD is 1.

[2] These registers can only be written when RM in CANMOD is 1.

The internal registers of each CAN Controller appear to the CPU as on-chip memory mapped peripheral registers. Because the CAN Controller can operate in different modes (Operating/Reset, see also Section 12–8.1 “Mode Register (CAN1MOD - 0xE004 4000, CAN2MOD - 0xE004 8000)”), one has to distinguish between different internal address definitions. Note that write access to some registers is only allowed in Reset Mode.

Table 219. CAN1 and CAN2 controller register map

Generic Name	Operating Mode		Reset Mode	
	Read	Write	Read	Write
MOD	Mode	Mode	Mode	Mode
CMR	0x00	Command	0x00	Command
GSR	Global Status and Error Counters	-	Global Status and Error Counters	Error Counters only
ICR	Interrupt and Capture	-	Interrupt and Capture	-
IER	Interrupt Enable	Interrupt Enable	Interrupt Enable	Interrupt Enable
BTR	Bus Timing	-	Bus Timing	Bus Timing
EWL	Error Warning Limit	-	Error Warning Limit	Error Warning Limit
SR	Status	-	Status	-
RFS	Rx Info and Index	-	Rx Info and Index	Rx Info and Index
RID	Rx Identifier	-	Rx Identifier	Rx Identifier
RDA	Rx Data	-	Rx Data	Rx Data
RDB	Rx Info and Index	-	Rx Info and Index	Rx Info and Index
TFI1	Tx Info1	Tx Info	Tx Info	Tx Info
TID1	Tx Identifier	Tx Identifier	Tx Identifier	Tx Identifier
TDA1	Tx Data	Tx Data	Tx Data	Tx Data
TDB1	Tx Data	Tx Data	Tx Data	Tx Data

In the following register tables, the column “Reset Value” shows how a hardware reset affects each bit or field, while the column “RM Set” indicates how each bit or field is affected if software sets the RM bit, or RM is set because of a Bus-Off condition. Note that while hardware reset sets RM, in this case the setting noted in the “Reset Value” column prevails over that shown in the “RM Set” column, in the few bits where they differ. In both columns, X indicates the bit or field is unchanged.

8.1 Mode Register (CAN1MOD - 0xE004 4000, CAN2MOD - 0xE004 8000)

The contents of the Mode Register are used to change the behavior of the CAN Controller. Bits may be set or reset by the CPU that uses the Mode Register as a read/write memory. Reserved Bits are read as 0 and should be written as 0.

Table 220. Mode register (CAN1MOD - address 0xE004 4000, CAN2MOD - address 0xE004 8000) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RM[1][6]		Reset Mode.	1	1
		0(normal)	The CAN Controller is in the Operating Mode, and certain registers can not be written.		
		1(reset)	CAN operation is disabled, writable registers can be written and the current transmission/reception of a message is aborted.		
1	LOM[3][2][6]		Listen Only Mode.	0	x
		0(normal)	The CAN controller acknowledges a successfully received message on the CAN bus. The error counters are stopped at the current value.		
		1(listen only)	The controller gives no acknowledgment, even if a message is successfully received. Messages cannot be sent, and the controller operates in “error passive” mode. This mode is intended for software bit rate detection and “hot plugging”.		
2	STM[3][6]		Self Test Mode.	0	x
		0(normal)	A transmitted message must be acknowledged to be considered successful.		
		1(self test)	The controller will consider a Tx message successful even if there is no acknowledgment received. In this mode a full node test is possible without any other active node on the bus using the SRR bit in CANxCMR.		
3	TPM[4]		Transmit Priority Mode.	0	x
		0(CAN ID)	The transmit priority for 3 Transmit Buffers depends on the CAN Identifier.		
		1(local prio)	The transmit priority for 3 Transmit Buffers depends on the contents of the Tx Priority register within the Transmit Buffer.		
4	SM[5]		Sleep Mode.	0	0
		0(wake-up)	Normal operation.		
		1(sleep)	The CAN controller enters Sleep Mode if no CAN interrupt is pending and there is no bus activity. See the Sleep Mode description Section 12–9.2 on page 284.		
5	RPM		Receive Polarity Mode.	0	x
		0(low active)	RD input is active Low (dominant bit = 0).		
		1(high active)	RD input is active High (dominant bit = 1) -- reverse polarity.		
6	-	-	Reserved, user software should not write ones to reserved bits.	0	0
7	TM		Test Mode.	0	x
		0(disabled)	Normal operation.		
		1(enabled)	The TD pin will reflect the bit, detected on RD pin, with the next positive edge of the system clock.		

[1] During a Hardware reset or when the Bus Status bit is set '1' (Bus-Off), the Reset Mode bit is set '1' (present). After the Reset Mode bit is set '0' the CAN Controller will wait for:

- one occurrence of Bus-Free signal (11 recessive bits), if the preceding reset has been caused by a Hardware reset or a CPU-initiated reset.

- 128 occurrences of Bus-Free, if the preceding reset has been caused by a CAN Controller initiated Bus-Off, before re-entering the Bus-On mode.

[2] This mode of operation forces the CAN Controller to be error passive. Message Transmission is not possible. The Listen Only Mode can be used e.g. for software driven bit rate detection and "hot plugging".

[3] A write access to the bits MOD.1 and MOD.2 is possible only if the Reset Mode is entered previously.

[4] Transmit Priority Mode is explained in more detail in Section 12–6.3 “Transmit Buffers (TXB)”.

[5] The CAN Controller will enter Sleep Mode, if the Sleep Mode bit is set '1' (sleep), there is no bus activity, and none of the CAN interrupts is pending. Setting of SM with at least one of the previously mentioned exceptions valid will result in a wake-up interrupt. The CAN Controller will wake up if SM is set LOW (wake-up) or there is bus activity. On wake-up, a Wake-up Interrupt is generated. A sleeping CAN Controller which wakes up due to bus activity will not be able to receive this message until it detects 11 consecutive recessive bits (Bus-Free sequence). Note that setting of SM is not possible in Reset Mode. After clearing of Reset Mode, setting of SM is possible only when Bus-Free is detected again.

[6] The LOM and STM bits can only be written if the RM bit is 1 prior to the write operation.

8.2 Command Register (CAN1CMR - 0xE004 x004, CAN2CMR - 0xE004 8004)

Writing to this write-only register initiates an action within the transfer layer of the CAN Controller. Bits not listed should be written as 0. Reading this register yields zeroes.

At least one internal clock cycle is needed for processing between two commands.

Table 221. Command Register (CAN1CMR - address 0xE004 4004, CAN2CMR - address 0xE004 8004) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0[1][2]	TR		Transmission Request.	0	0
		0 (absent)	No transmission request.		
		1 (present)	The message, previously written to the CANxTFI, CANxTID, and optionally the CANxTDA and CANxTDB registers, is queued for transmission from the selected Transmit Buffer. If at two or all three of STB1, STB2 and STB3 bits are selected when TR=1 is written, Transmit Buffer will be selected based on the chosen priority scheme (for details see Section 12–6.3 “Transmit Buffers (TXB)”)		
1[1][3]	AT		Abort Transmission.	0	0
		0 (no action)	Do not abort the transmission.		
		1 (present)	if not already in progress, a pending Transmission Request for the selected Transmit Buffer is cancelled.		
2[4]	RRB		Release Receive Buffer.	0	0
		0 (no action)	Do not release the receive buffer.		
		1 (released)	The information in the Receive Buffer (consisting of CANxRFS, CANxRID, and if applicable the CANxRDA and CANxRDB registers) is released, and becomes eligible for replacement by the next received frame. If the next received frame is not available, writing this command clears the RBS bit in the Status Register(s).		
3[5]	CDO		Clear Data Overrun.	0	0
		0 (no action)	Do not clear the data overrun bit.		
		1 (clear)	The Data Overrun bit in Status Register(s) is cleared.		
4[1][6]	SRR		Self Reception Request.	0	0
		0 (absent)	No self reception request.		
		1 (present)	The message, previously written to the CANxTFS, CANxTID, and optionally the CANxTDA and CANxTDB registers, is queued for transmission from the selected Transmit Buffer and received simultaneously. This differs from the TR bit above in that the receiver is not disabled during the transmission, so that it receives the message if its Identifier is recognized by the Acceptance Filter.		

Table 221. Command Register (CAN1CMR - address 0xE004 4004, CAN2CMR - address 0xE004 8004) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
5	STB1		Select Tx Buffer 1.	0	0
		0 (not selected)	Tx Buffer 1 is not selected for transmission.		
		1 (selected)	Tx Buffer 1 is selected for transmission.		
6	STB2		Select Tx Buffer 2.	0	0
		0 (not selected)	Tx Buffer 2 is not selected for transmission.		
		1 (selected)	Tx Buffer 2 is selected for transmission.		
7	STB3		Select Tx Buffer 3.	0	0
		0 (not selected)	Tx Buffer 3 is not selected for transmission.		
		1 (selected)	Tx Buffer 3 is selected for transmission.		

[1] - Setting the command bits TR and AT simultaneously results in transmitting a message once. No re-transmission will be performed in case of an error or arbitration lost (single shot transmission).
- Setting the command bits SRR and TR simultaneously results in sending the transmit message once using the self-reception feature. No re-transmission will be performed in case of an error or arbitration lost.
- Setting the command bits TR, AT and SRR simultaneously results in transmitting a message once as described for TR and AT. The moment the Transmit Status bit is set within the Status Register, the internal Transmission Request Bit is cleared automatically.
- Setting TR and SRR simultaneously will ignore the set SRR bit.

[2] If the Transmission Request or the Self-Reception Request bit was set '1' in a previous command, it cannot be cancelled by resetting the bits. The requested transmission may only be cancelled by setting the Abort Transmission bit.

[3] The Abort Transmission bit is used when the CPU requires the suspension of the previously requested transmission, e.g. to transmit a more urgent message before. A transmission already in progress is not stopped. In order to see if the original message has been either transmitted successfully or aborted, the Transmission Complete Status bit should be checked. This should be done after the Transmit Buffer Status bit has been set to '1' or a Transmit Interrupt has been generated.

[4] After reading the contents of the Receive Buffer, the CPU can release this memory space by setting the Release Receive Buffer bit '1'. This may result in another message becoming immediately available. If there is no other message available, the Receive Interrupt bit is reset. If the RRB command is given, it will take at least 2 internal clock cycles before a new interrupt is generated.

[5] This command bit is used to clear the Data Overrun condition signalled by the Data Overrun Status bit. As long as the Data Overrun Status bit is set no further Data Overrun Interrupt is generated.

[6] Upon Self Reception Request, a message is transmitted and simultaneously received if the Acceptance Filter is set to the corresponding identifier. A receive and a transmit interrupt will indicate correct self reception (see also Self Test Mode in Section 12–8.1 “Mode Register (CAN1MOD - 0xE004 4000, CAN2MOD - 0xE004 8000)”).

8.3 Global Status Register (CAN1GSR - 0xE004 x008, CAN2GSR - 0xE004 8008)

The content of the Global Status Register reflects the status of the CAN Controller. This register is read-only, except that the Error Counters can be written when the RM bit in the CANMOD register is 1. Bits not listed read as 0 and should be written as 0.

Table 222. Global Status Register (CAN1GSR - address 0xE004 4008, CAN2GSR - address 0xE004 8008) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RBS[1]		Receive Buffer Status.	0	0
		0 (empty)	No message is available.		
		1 (full)	At least one complete message is received by the Double Receive Buffer and available in the CANxRFS, CANxRID, and if applicable the CANxRDA and CANxRDB registers. This bit is cleared by the Release Receive Buffer command in CANxCMR, if no subsequent received message is available.		

Table 222. Global Status Register (CAN1GSR - address 0xE004 4008, CAN2GSR - address 0xE004 8008) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
1	DOS[2]		Data Overrun Status.	0	0
		0 (absent)	No data overrun has occurred since the last Clear Data Overrun command was given/written to CANxCMR (or since Reset).		
		1 (overrun)	A message was lost because the preceding message to this CAN controller was not read and released quickly enough (there was not enough space for a new message in the Double Receive Buffer).		
2	TBS		Transmit Buffer Status.	1	1
		0 (locked)	At least one of the Transmit Buffers is not available for the CPU, i.e. at least one previously queued message for this CAN controller has not yet been sent, and therefore software should not write to the CANxTFI, CANxTID, CANxTDA, nor CANxTDB registers of that (those) Tx buffer(s).		
		1 (released)	All three Transmit Buffers are available for the CPU. No transmit message is pending for this CAN controller (in any of the 3 Tx buffers), and software may write to any of the CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
3	TCS[3]		Transmit Complete Status.	1	x
		0 (incomplete)	At least one requested transmission has not been successfully completed yet.		
		1 (complete)	All requested transmission(s) has (have) been successfully completed.		
4	RS[4]		Receive Status.	1	0
		0 (idle)	The CAN controller is idle.		
		1 (receive)	The CAN controller is receiving a message.		
5	TS[4]		Transmit Status.	1	0
		0 (idle)	The CAN controller is idle.		
		1 (transmit)	The CAN controller is sending a message.		
6	ES[5]		Error Status.	0	0
		0 (ok)	Both error counters are below the Error Warning Limit.		
		1 (error)	One or both of the Transmit and Receive Error Counters has reached the limit set in the Error Warning Limit register.		
7	BS[6]		Bus Status.	0	0
		0 (Bus-On)	The CAN Controller is involved in bus activities		
		1 (Bus-Off)	The CAN controller is currently not involved/prohibited from bus activity because the Transmit Error Counter reached its limiting value of 255.		
15:8	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
23:16	RXERR	-	The current value of the Rx Error Counter (an 8 - bit value).	0	X
31:24	TXERR	-	The current value of the Tx Error Counter (an 8 - bit value).	0	X

[1] After reading all messages and releasing their memory space with the command 'Release Receive Buffer,' this bit is cleared.

[2] If there is not enough space to store the message within the Receive Buffer, that message is dropped and the Data Overrun condition is signalled to the CPU in the moment this message becomes valid. If this message is not completed successfully (e.g. because of an error), no overrun condition is signalled.

[3] The Transmission Complete Status bit is set '0' (incomplete) whenever the Transmission Request bit or the Self Reception Request bit is set '1' at least for one of the three Transmit Buffers. The Transmission Complete Status bit will remain '0' until all messages are transmitted successfully.

[4] If both the Receive Status and the Transmit Status bits are '0' (idle), the CAN-Bus is idle. If both bits are set, the controller is waiting to become idle again. After hardware reset 11 consecutive recessive bits have to be detected until idle status is reached. After Bus-off this will take 128 times of 11 consecutive recessive bits.

[5] Errors detected during reception or transmission will effect the error counters according to the CAN specification. The Error Status bit is set when at least one of the error counters has reached or exceeded the Error Warning Limit. An Error Warning Interrupt is generated, if enabled. The default value of the Error Warning Limit after hardware reset is 96 decimal, see also Section 12–8.7 “Error Warning Limit Register (CAN1EWL - 0xE004 4018, CAN2EWL - 0xE004 8018)”.

[6] Mode bit '1' (present) and an Error Warning Interrupt is generated, if enabled. Afterwards the Transmit Error Counter is set to '127', and the Receive Error Counter is cleared. It will stay in this mode until the CPU clears the Reset Mode bit. Once this is completed the CAN Controller will wait the minimum protocol-defined time (128 occurrences of the Bus-Free signal) counting down the Transmit Error Counter. After that, the Bus Status bit is cleared (Bus-On), the Error Status bit is set '0' (ok), the Error Counters are reset, and an Error Warning Interrupt is generated, if enabled. Reading the TX Error Counter during this time gives information about the status of the Bus-Off recovery.

RX error counter

The RX Error Counter Register, which is part of the Status Register, reflects the current value of the Receive Error Counter. After hardware reset this register is initialized to 0. In Operating Mode this register appears to the CPU as a read only memory. A write access to this register is possible only in Reset Mode. If a Bus Off event occurs, the RX Error Counter is initialized to 0. As long as Bus Off is valid, writing to this register has no effect.The Rx Error Counter is determined as follows:

RX Error Counter = (CANxGSR AND 0x00FF0000) / 0x00010000

Note that a CPU-forced content change of the RX Error Counter is possible only if the Reset Mode was entered previously. An Error Status change (Status Register), an Error Warning or an Error Passive Interrupt forced by the new register content will not occur until the Reset Mode is cancelled again.

TX error counter

The TX Error Counter Register, which is part of the Status Register, reflects the current value of the Transmit Error Counter. In Operating Mode this register appears to the CPU as a read only memory. After hardware reset this register is initialized to ’0’. A write access to this register is possible only in Reset Mode. If a bus-off event occurs, the TX Error Counter is initialized to 127 to count the minimum protocol-defined time (128 occurrences of the Bus-Free signal). Reading the TX Error Counter during this time gives information about the status of the Bus-Off recovery. If Bus Off is active, a write access to TXERR in the range of 0 to 254 clears the Bus Off Flag and the controller will wait for one occurrence of 11 consecutive recessive bits (bus free) after clearing of Reset Mode. The Tx error counter is determined as follows:

TX Error Counter = (CANxGSR AND 0xFF000000) / 0x01000000

Writing 255 to TXERR allows initiation of a CPU-driven Bus Off event. Note that a CPU-forced content change of the TX Error Counter is possible only if the Reset Mode was entered previously. An Error or Bus Status change (Status Register), an Error Warning, or an Error Passive Interrupt forced by the new register content will not occur until the Reset Mode is cancelled again. After leaving the Reset Mode, the new TX Counter content is interpreted and the Bus Off event is performed in the same way as if it was forced by a bus error event. That means, that the Reset Mode is entered again, the TX Error Counter is initialized to 127, the RX Counter is cleared, and all concerned Status and Interrupt Register Bits are set. Clearing of Reset Mode now will perform the protocol

defined Bus Off recovery sequence (waiting for 128 occurrences of the Bus-Free signal). If the Reset Mode is entered again before the end of Bus Off recovery (TXERR>0), Bus Off keeps active and TXERR is frozen.

8.4 Interrupt and Capture Register (CAN1ICR - 0xE004 400C, CAN2ICR - 0xE004 800C)

Bits in this register indicate information about events on the CAN bus. This register is read-only. Bits not listed read as 0 and should be written as 0.

The Interrupt flags of the Interrupt and Capture Register allow the identification of an interrupt source. When one or more bits are set, a CAN interrupt will be indicated to the CPU. After this register is read from the CPU all interrupt bits are reset **except** of the Receive Interrupt bit. The Interrupt Register appears to the CPU as a read only memory.

Bits 1 thru 10 clear when they are read.

Bits 16-23 are captured when a bus error occurs. At the same time, if the BEIE bit in CANIER is 1, the BEI bit in this register is set, and a CAN interrupt can occur.

Bits 24-31 are captured when CAN arbitration is lost. At the same time, if the ALIE bit in CANIER is 1, the ALI bit in this register is set, and a CAN interrupt can occur. Once either of these bytes is captured, its value will remain the same until it is read, at which time it is released to capture a new value.

The clearing of bits 1 to 10 and the releasing of bits 16-23 and 24-31 all occur on any read from CANxICR, regardless of whether part or all of the register is read. This means that software should always read CANxICR as a word, and process and deal with all bits of the register as appropriate for the application.

Table 223. Interrupt and Capture Register (CAN1ICR - address 0xE004 400C, CAN2ICR - address 0xE004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RI[1]	0 (reset) 1 (set)	Receive Interrupt. This bit is set whenever the RBS bit in CANxSR and the RIE bit in CANxIER are both 1, indicating that a new message was received and stored in the Receive Buffer.	0	0
1	TI1	0 (reset) 1 (set)	Transmit Interrupt 1. This bit is set when the TBS1 bit in CANxSR goes from 0 to 1 (whenever a message out of TXB1 was successfully transmitted or aborted), indicating that Transmit buffer 1 is available, and the TIE1 bit in CANxIER is 1.	0	0
2	EI	0 (reset) 1 (set)	Error Warning Interrupt. This bit is set on every change (set or clear) of either the Error Status or Bus Status bit in CANxSR and the EIE bit bit is set within the Interrupt Enable Register at the time of the change.	0	X
3	DOI	0 (reset) 1 (set)	Data Overrun Interrupt. This bit is set when the DOS bit in CANxSR goes from 0 to 1 and the DOIE bit in CANxIER is 1.	0	0
4	WUI[2]	0 (reset) 1 (set)	Wake-Up Interrupt. This bit is set if the CAN controller is sleeping and bus activity is detected and the WUIE bit in CANxIER is 1.	0	0

Table 223. Interrupt and Capture Register (CAN1ICR - address 0xE004 400C, CAN2ICR - address 0xE004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
5	EPI	0 (reset) 1 (set)	Error Passive Interrupt. This bit is set if the EPIE bit in CANxIER is 1, and the CAN controller switches between Error Passive and Error Active mode in either direction. This is the case when the CAN Controller has reached the Error Passive Status (at least one error counter exceeds the CAN protocol defined level of 127) or if the CAN Controller is in Error Passive Status and enters the Error Active Status again.	0	0
6	ALI	0 (reset) 1 (set)	Arbitration Lost Interrupt. This bit is set if the ALIE bit in CANxIER is 1, and the CAN controller loses arbitration while attempting to transmit. In this case the CAN node becomes a receiver.	0	0
7	BEI	0 (reset) 1 (set)	Bus Error Interrupt -- this bit is set if the BEIE bit in CANxIER is 1, and the CAN controller detects an error on the bus.	0	X
8	IDI	0 (reset) 1 (set)	ID Ready Interrupt -- this bit is set if the IDIE bit in CANxIER is 1, and a CAN Identifier has been received (a message was successfully transmitted or aborted). This bit is set whenever a message was successfully transmitted or aborted and the IDIE bit is set in the IER reg.	0	0
9	TI2	0 (reset) 1 (set)	Transmit Interrupt 2. This bit is set when the TBS2 bit in CANxSR goes from 0 to 1 (whenever a message out of TXB2 was successfully transmitted or aborted), indicating that Transmit buffer 2 is available, and the TIE2 bit in CANxIER is 1.	0	0
10	TI3	0 (reset) 1 (set)	Transmit Interrupt 3. This bit is set when the TBS3 bit in CANxSR goes from 0 to 1 (whenever a message out of TXB3 was successfully transmitted or aborted), indicating that Transmit buffer 3 is available, and the TIE3 bit in CANxIER is 1.	0	0
15:11	-	-	Reserved, user software should not write ones to reserved bits.	0	0

Table 223. Interrupt and Capture Register (CAN1ICR - address 0xE004 400C, CAN2ICR - address 0xE004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
20:16	ERRBIT 4:0[3]		Error Code Capture: when the CAN controller detects a bus error, the location of the error within the frame is captured in this field. The value reflects an internal state variable, and as a result is not very linear:	0	X
		00011	Start of Frame		
		00010	ID28 ... ID21		
		00110	ID20 ... ID18		
		00100	SRTR Bit		
		00101	IDE bit		
		00111	ID17 ... 13		
		01111	ID12 ... ID5		
		01110	ID4 ... ID0		
		01100	RTR Bit		
		01101	Reserved Bit 1		
		01001	Reserved Bit 0		
		01011	Data Length Code		
		01010	Data Field		
		01000	CRC Sequence		
		11000	CRC Delimiter		
		11001	Acknowledge Slot		
		11011	Acknowledge Delimiter		
		11010	End of Frame		
		10010	Intermission		
		10001	Active Error Flag		
		10110	Passive Error Flag		
		10011	Tolerate Dominant Bits		
		10111	Error Delimiter		
		11100	Overload flag		
21	ERRDIR		When the CAN controller detects a bus error, the direction of the current bit is captured in this bit.	0	X
		0	Error occurred during transmitting.		
		1	Error occurred during receiving.		
23:22	ERRC1:0		When the CAN controller detects a bus error, the type of error is captured in this field:	0	X
		00	Bit error		
		01	Form error		
		10	Stuff error		
		11	Other error		

Table 223. Interrupt and Capture Register (CAN1ICR - address 0xE004 400C, CAN2ICR - address 0xE004 800C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
31:24	ALCBIT[4]	-	Each time arbitration is lost while trying to send on the CAN, the bit number within the frame is captured into this field. After the content of ALCBIT is read, the ALI bit is cleared and a new Arbitration Lost interrupt can occur.	0	X
		00	arbitration lost in the first bit (MS) of identifier		
		...	a		
		11	arbitration lost in SRTS bit (RTR bit for standard frame messages)		
		12	arbitration lost in IDE bit		
		13	arbitration lost in 12th bit of identifier (extended frame only)		
		...			
		30	arbitration lost in last bit of identifier (extended frame only)		
		31	arbitration lost in RTR bit (extended frame only)		

[1] The Receive Interrupt Bit is not cleared upon a read access to the Interrupt Register. Giving the Command “Release Receive Buffer” will clear RI temporarily. If there is another message available within the Receive Buffer after the release command, RI is set again. Otherwise RI remains cleared.

[2] A Wake-Up Interrupt is also generated if the CPU tries to set the Sleep bit while the CAN controller is involved in bus activities or a CAN Interrupt is pending. The WUI flag can also get asserted when the according enable bit WUIE is not set. In this case a Wake-Up Interrupt does not get asserted.

[3] Whenever a bus error occurs, the corresponding bus error interrupt is forced, if enabled. At the same time, the current position of the Bit Stream Processor is captured into the Error Code Capture Register. The content within this register is fixed until the user software has read out its content once. From now on, the capture mechanism is activated again, i.e. reading the CANxICR enables another Bus Error Interrupt.

[4] On arbitration lost, the corresponding arbitration lost interrupt is forced, if enabled. At that time, the current bit position of the Bit Stream Processor is captured into the Arbitration Lost Capture Register. The content within this register is fixed until the user application has read out its contents once. From now on, the capture mechanism is activated again.

8.5 Interrupt Enable Register (CAN1IER - 0xE004 4010, CAN2IER - 0xE004 8010)

This read/write register controls whether various events on the CAN controller will result in an interrupt or not. Bits 10:0 in this register correspond 1-to-1 with bits 10:0 in the CANxICR register. If a bit in the CANxIER register is 0 the corresponding interrupt is disabled; if a bit in the CANxIER register is 1 the corresponding source is enabled to trigger an interrupt.

Table 224. Interrupt Enable Register (CAN1IER - address 0xE004 4010, CAN2IER - address 0xE004 8010) bit description

Bit	Symbol	Function	Reset Value	RM Set
0	RIE	Receiver Interrupt Enable. When the Receive Buffer Status is 'full', the CAN Controller requests the respective interrupt.	0	X
1	TIE1	Transmit Interrupt Enable for Buffer1. When a message has been successfully transmitted out of TXB1 or Transmit Buffer 1 is accessible again (e.g. after an Abort Transmission command), the CAN Controller requests the respective interrupt.	0	X
2	EIE	Error Warning Interrupt Enable. If the Error or Bus Status change (see Status Register), the CAN Controller requests the respective interrupt.	0	X
3	DOIE	Data Overrun Interrupt Enable. If the Data Overrun Status bit is set (see Status Register), the CAN Controller requests the respective interrupt.	0	X
4	WUIE	Wake-Up Interrupt Enable. If the sleeping CAN controller wakes up, the respective interrupt is requested.	0	X
5	EPIE	Error Passive Interrupt Enable. If the error status of the CAN Controller changes from error active to error passive or vice versa, the respective interrupt is requested.	0	X
6	ALIE	Arbitration Lost Interrupt Enable. If the CAN Controller has lost arbitration, the respective interrupt is requested.	0	X
7	BEIE	Bus Error Interrupt Enable. If a bus error has been detected, the CAN Controller requests the respective interrupt.	0	X
8	IDIE	ID Ready Interrupt Enable. When a CAN identifier has been received, the CAN Controller requests the respective interrupt.	0	X
9	TIE2	Transmit Interrupt Enable for Buffer2. When a message has been successfully transmitted out of TXB2 or Transmit Buffer 2 is accessible again (e.g. after an Abort Transmission command), the CAN Controller requests the respective interrupt.	0	X
10	TIE3	Transmit Interrupt Enable for Buffer3. When a message has been successfully transmitted out of TXB3 or Transmit Buffer 3 is accessible again (e.g. after an Abort Transmission command), the CAN Controller requests the respective interrupt.	0	X
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

8.6 Bus Timing Register (CAN1BTR - 0xE004 4014, CAN2BTR - 0xE004 8014)

This register controls how various CAN timings are derived from the APB clock. It defines the values of the Baud Rate Prescaler (BRP) and the Synchronization Jump Width (SJW). Furthermore, it defines the length of the bit period, the location of the sample point and the number of samples to be taken at each sample point. It can be read at any time but can only be written if the RM bit in CANmod is 1.

Table 225. Bus Timing Register (CAN1BTR - address 0xE004 4014, CAN2BTR - address 0xE004 8014) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
9:0	BRP		Baud Rate Prescaler. The APB clock is divided by (this value plus one) to produce the CAN clock.	0	X
13:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
15:14	SJW		The Synchronization Jump Width is (this value plus one) CAN clocks.	0	X
19:16	TESG1		The delay from the nominal Sync point to the sample point is (this value plus one) CAN clocks.	1100	X
22:20	TESG2		The delay from the sample point to the next nominal sync point is (this value plus one) CAN clocks. The nominal CAN bit time is (this value plus the value in TSEG1 plus 3) CAN clocks.	001	X
23	SAM		Sampling		
		0	The bus is sampled once (recommended for high speed buses)	0	X
		1	The bus is sampled 3 times (recommended for low to medium speed buses to filter spikes on the bus-line)		
31:24	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

Baud rate prescaler

The period of the CAN system clock t_{SCL} is programmable and determines the individual bit timing. The CAN system clock t_{SCL} is calculated using the following equation:

(1)

$$t_{SCL} = t_{CANsuppliedCLK} \times (BRP + 1)$$

Synchronization jump width

To compensate for phase shifts between clock oscillators of different bus controllers, any bus controller must re-synchronize on any relevant signal edge of the current transmission. The synchronization jump width t_{SJW} defines the maximum number of clock cycles a certain bit period may be shortened or lengthened by one re-synchronization:

(2)

$$t_{SJW} = t_{SCL} \times (SJW + 1)$$

Time segment 1 and time segment 2

Time segments TSEG1 and TSEG2 determine the number of clock cycles per bit period and the location of the sample point:

(3)

$$t_{SYNCSEG} = t_{SCL}$$

(4)

$$t_{TSEG1} = t_{SCL} \times (TSEG1 + 1)$$

(5)

$$t_{TSEG2} = t_{SCL} \times (TSEG2 + 1)$$

8.7 Error Warning Limit Register (CAN1EWL - 0xE004 4018, CAN2EWL - 0xE004 8018)

This register sets a limit on Tx or Rx errors at which an interrupt can occur. It can be read at any time but can only be written if the RM bit in CANmod is 1. The default value (after hardware reset) is 96.

Table 226. Error Warning Limit register (CAN1EWL - address 0xE004 4018, CAN2EWL - address 0xE004 8018) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	EWL	During CAN operation, this value is compared to both the Tx and Rx Error Counters. If either of these counter matches this value, the Error Status (ES) bit in CANSR is set.	9610 = 0x60	X

Note that a content change of the Error Warning Limit Register is possible only if the Reset Mode was entered previously. An Error Status change (Status Register) and an Error Warning Interrupt forced by the new register content will not occur until the Reset Mode is cancelled again.

8.8 Status Register (CAN1SR - 0xE004 401C, CAN2SR - 0xE004 801C)

This register contains three status bytes in which the bits not related to transmission are identical to the corresponding bits in the Global Status Register, while those relating to transmission reflect the status of each of the 3 Tx Buffers.

Table 227. Status Register (CAN1SR - address 0xE004 401C, CAN2SR - address 0xE004 801C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
0	RBS		Receive Buffer Status. This bit is identical to the RBS bit in the CANxGSR.	0	0
1	DOS		Data Overrun Status. This bit is identical to the DOS bit in the CANxGSR.	0	0
2	TBS1[1]		Transmit Buffer Status 1.	1	1
		0(locked)	Software cannot access the Tx Buffer 1 nor write to the corresponding CANxTFI, CANxTID, CANxTDA, and CANxTDB registers because a message is either waiting for transmission or is in transmitting process.		
		1(released)	Software may write a message into the Transmit Buffer 1 and its CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
3	TCS1[2]		Transmission Complete Status.	1	x
		0(incomplete)	The previously requested transmission for Tx Buffer 1 is not complete.		
		1(complete)	The previously requested transmission for Tx Buffer 1 has been successfully completed.		
4	RS		Receive Status. This bit is identical to the RS bit in the GSR.	1	0

Table 227. Status Register (CAN1SR - address 0xE004 401C, CAN2SR - address 0xE004 801C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
5	TS1		Transmit Status 1.	1	0
		0(idle)	There is no transmission from Tx Buffer 1.		
		1(transmit)	The CAN Controller is transmitting a message from Tx Buffer 1.		
6	ES		Error Status. This bit is identical to the ES bit in the CANxGSR.	0	0
7	BS		Bus Status. This bit is identical to the BS bit in the CANxGSR.	0	0
8	RBS		Receive Buffer Status. This bit is identical to the RBS bit in the CANxGSR.	0	0
9	DOS		Data Overrun Status. This bit is identical to the DOS bit in the CANxGSR.	0	0
10	TBS2[1]		Transmit Buffer Status 2.	1	1
		0(locked)	Software cannot access the Tx Buffer 2 nor write to the corresponding CANxTFI, CANxTID, CANxTDA, and CANxTDB registers because a message is either waiting for transmission or is in transmitting process.		
		1(released)	Software may write a message into the Transmit Buffer 2 and its CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
11	TCS2[2]		Transmission Complete Status.	1	x
		0(incomplete)	The previously requested transmission for Tx Buffer 2 is not complete.		
		1(complete)	The previously requested transmission for Tx Buffer 2 has been successfully completed.		
12	RS		Receive Status. This bit is identical to the RS bit in the GSR.	1	0
13	TS2		Transmit Status 2.	1	0
		0(idle)	There is no transmission from Tx Buffer 2.		
		1(transmit)	The CAN Controller is transmitting a message from Tx Buffer 2.		
14	ES		Error Status. This bit is identical to the ES bit in the CANxGSR.	0	0
15	BS		Bus Status. This bit is identical to the BS bit in the CANxGSR.	0	0
16	RBS		Receive Buffer Status. This bit is identical to the RBS bit in the CANxGSR.	0	0
17	DOS		Data Overrun Status. This bit is identical to the DOS bit in the CANxGSR.	0	0
18	TBS3[1]		Transmit Buffer Status 3.	1	1
		0(locked)	Software cannot access the Tx Buffer 3 nor write to the corresponding CANxTFI, CANxTID, CANxTDA, and CANxTDB registers because a message is either waiting for transmission or is in transmitting process.		
		1(released)	Software may write a message into the Transmit Buffer 3 and its CANxTFI, CANxTID, CANxTDA, and CANxTDB registers.		
19	TCS3[2]		Transmission Complete Status.	1	x
		0(incomplete)	The previously requested transmission for Tx Buffer 3 is not complete.		
		1(complete)	The previously requested transmission for Tx Buffer 3 has been successfully completed.		
20	RS		Receive Status. This bit is identical to the RS bit in the GSR.	1	0
21	TS3		Transmit Status 3.	1	0
		0(idle)	There is no transmission from Tx Buffer 3.		
		1(transmit)	The CAN Controller is transmitting a message from Tx Buffer 3.		

Table 227. Status Register (CAN1SR - address 0xE004 401C, CAN2SR - address 0xE004 801C) bit description

Bit	Symbol	Value	Function	Reset Value	RM Set
22	ES		Error Status. This bit is identical to the ES bit in the CANxGSR.	0	0
23	BS		Bus Status. This bit is identical to the BS bit in the CANxGSR.	0	0
31:24	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

[1] If the CPU tries to write to this Transmit Buffer when the Transmit Buffer Status bit is '0' (locked), the written byte is not accepted and is lost without this being signalled.

[2] The Transmission Complete Status bit is set '0' (incomplete) whenever the Transmission Request bit or the Self Reception Request bit is set '1' for this TX buffer. The Transmission Complete Status bit remains '0' until a message is transmitted successfully.

8.9 Receive Frame Status Register (CAN1RFS - 0xE004 4020, CAN2RFS - 0xE004 8020)

This register defines the characteristics of the current received message. It is read-only in normal operation but can be written for testing purposes if the RM bit in CANxMOD is 1.

Table 228. Receive Frame Status register (CAN1RFS - address 0xE004 4020, CAN2RFS - address 0xE004 8020) bit description

Bit	Symbol	Function	Reset Value	RM Set
9:0	ID Index	If the BP bit (below) is 0, this value is the zero-based number of the Lookup Table RAM entry at which the Acceptance Filter matched the received Identifier. Disabled entries in the Standard tables are included in this numbering, but will not be matched. See Section 12–18 “Examples of acceptance filter tables and ID index values” on page 308 for examples of ID Index values.	0	X
10	BP	If this bit is 1, the current message was received in AF Bypass mode, and the ID Index field (above) is meaningless.	0	X
15:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
19:16	DLC	The field contains the Data Length Code (DLC) field of the current received message. When RTR = 0, this is related to the number of data bytes available in the CANRDA and CANRDB registers as follows: 0000-0111 = 0 to 7 bytes1000-1111 = 8 bytes With RTR = 1, this value indicates the number of data bytes requested to be sent back, with the same encoding.	0	X
29:20	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	
30	RTR	This bit contains the Remote Transmission Request bit of the current received message. 0 indicates a Data Frame, in which (if DLC is non-zero) data can be read from the CANRDA and possibly the CANRDB registers. 1 indicates a Remote frame, in which case the DLC value identifies the number of data bytes requested to be sent using the same Identifier.	0	X
31	FF	A 0 in this bit indicates that the current received message included an 11 bit Identifier, while a 1 indicates a 29 bit Identifier. This affects the contents of the CANid register described below.	0	X

8.9.1 ID index field

The ID Index is a 10-bit field in the Info Register that contains the table position of the ID Look-up Table if the currently received message was accepted. The software can use this index to simplify message transfers from the Receive Buffer into the Shared Message Memory. Whenever bit 10 (BP) of the ID Index in the CANRFS register is 1, the current CAN message was received in acceptance filter bypass mode.

8.10 Receive Identifier Register (CAN1RID - 0xE004 4024, CAN2RID - 0xE004 8024)

This register contains the Identifier field of the current received message. It is read-only in normal operation but can be written for testing purposes if the RM bit in CANmod is 1. It has two different formats depending on the FF bit in CANRFS. See Table 12–217 for details on specific CAN channel register address.

Table 229. Receive Identifier Register (CAN1RID - address 0xE004 4024, CAN2RID - address 0xE004 8024) bit description

Bit	Symbol	Function	Reset Value	RM Set
10:0	ID	The 11 bit Identifier field of the current received message. In CAN 2.0A, these bits are called ID10-0, while in CAN 2.0B they're called ID29-18.	0	X
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

Table 230. RX Identifier register when FF = 1

Bit	Symbol	Function	Reset Value	RM Set
28:0	ID	The 29 bit Identifier field of the current received message. In CAN 2.0B these bits are called ID29-0.	0	X
31:29	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

8.11 Receive Data Register A (CAN1RDA - 0xE004 4028, CAN2RDA - 0xE004 8028)

This register contains the first 1-4 Data bytes of the current received message. It is read-only in normal operation, but can be written for testing purposes if the RM bit in CANMOD is 1. See Table 12–217 for details on specific CAN channel register address.

Table 231. Receive Data register A (CAN1RDA - address 0xE004 4028, CAN2RDA - address 0xE004 8028) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 1	If the DLC field in CANRFS Š 0001, this contains the first Data byte of the current received message.	0	X

Table 231. Receive Data register A (CAN1RDA - address 0xE004 4028, CAN2RDA - address 0xE004 8028) bit description

Bit	Symbol	Function	Reset Value	RM Set
15:8	Data 2	If the DLC field in CANRFS Š 0010, this contains the first Data byte of the current received message.	0	X
23:16	Data 3	If the DLC field in CANRFS Š 0011, this contains the first Data byte of the current received message.	0	X
31:24	Data 4	If the DLC field in CANRFS Š 0100, this contains the first Data byte of the current received message.	0	X

8.12 Receive Data Register B (CAN1RDB - 0xE004 402C, CAN2RDB - 0xE004 802C)

This register contains the 5th through 8th Data bytes of the current received message. It is read-only in normal operation, but can be written for testing purposes if the RM bit in CANMOD is 1. See Table 12–217 for details on specific CAN channel register address.

Table 232. Receive Data register B (CAN1RDB - address 0xE004 402C, CAN2RDB - address 0xE004 802C) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 5	If the DLC field in CANRFS Š 0101, this contains the first Data byte of the current received message.	0	X
15:8	Data 6	If the DLC field in CANRFS Š 0110, this contains the first Data byte of the current received message.	0	X
23:16	Data 7	If the DLC field in CANRFS Š 0111, this contains the first Data byte of the current received message.	0	X
31:24	Data 8	If the DLC field in CANRFS Š 1000, this contains the first Data byte of the current received message.	0	X

8.13 Transmit Frame Information Register (CAN1TFI[1/2/3] - 0xE004 40[30/40/50], CAN2TFI[1/2/3] - 0xE004 80[30/40/50])

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the format of the next transmit message for that Tx buffer. Bits not listed read as 0 and should be written as 0.

The values for the reserved bits of the CANxTFI register in the Transmit Buffer should be set to the values expected in the Receive Buffer for an easy comparison, when using the Self Reception facility (self test), otherwise they are not defined.

The CAN Controller consist of three Transmit Buffers. Each of them has a length of 4 words and is able to store one complete CAN message as shown in Figure 12–42.

The buffer layout is subdivided into Descriptor and Data Field where the first word of the Descriptor Field includes the TX Frame Info that describes the Frame Format, the Data Length and whether it is a Remote or Data Frame. In addition, a TX Priority register allows the definition of a certain priority for each transmit message. Depending on the chosen Frame Format, an 11-bit identifier for Standard Frame Format (SFF) or an 29-bit identifier for Extended Frame Format (EFF) follows. Note that unused bits in the TID field have to be defined as 0. The Data Field in TDA and TDB contains up to eight data bytes.

Table 233. Transmit Frame Information Register (CAN1TFI[1/2/3] - address 0xE004 40[30/40/50], CAN2TFI[1/2/3] - 0xE004 80[30/40/50]) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	PRIO	If the TPM (Transmit Priority Mode) bit in the CANxMOD register is set to 1, enabled Tx Buffers contend for the right to send their messages based on this field. The buffer with the lowest TX Priority value wins the prioritization and is sent first.		x
15:8	-	Reserved.	0	
19:16	DLC	Data Length Code. This value is sent in the DLC field of the next transmit message. In addition, if RTR = 0, this value controls the number of Data bytes sent in the next transmit message, from the CANxTDA and CANxTDB registers: 0000-0111 = 0-7 bytes 1xxx = 8 bytes	0	X
29:20	-	Reserved.	0	
30	RTR	This value is sent in the RTR bit of the next transmit message. If this bit is 0, the number of data bytes called out by the DLC field are sent from the CANxTDA and CANxTDB registers. If this bit is 1, a Remote Frame is sent, containing a request for that number of bytes.	0	X
31	FF	If this bit is 0, the next transmit message will be sent with an 11 bit Identifier (standard frame format), while if it's 1, the message will be sent with a 29 bit Identifier (extended frame format).	0	X

Automatic transmit priority detection

To allow uninterrupted streams of transmit messages, the CAN Controller provides Automatic Transmit Priority Detection for all Transmit Buffers. Depending on the selected Transmit Priority Mode, internal prioritization is based on the CAN Identifier or a user defined "local priority". If more than one message is enabled for transmission (TR=1) the internal transmit message queue is organized such as that the transmit buffer with the lowest CAN Identifier (TID) or the lowest "local priority" (TX Priority) wins the prioritization and is sent first. The result of the internal scheduling process is taken into account short before a new CAN message is sent on the bus. This is also true after the occurrence of a transmission error and right before a re-transmission.

Tx DLC

The number of bytes in the Data Field of a message is coded with the Data Length Code (DLC). At the start of a Remote Frame transmission the DLC is not considered due to the RTR bit being '1 ' (remote). This forces the number of transmitted/received data bytes to be 0. Nevertheless, the DLC must be specified correctly to avoid bus errors, if two CAN Controllers start a Remote Frame transmission with the same identifier simultaneously. For reasons of compatibility no DLC > 8 should be used. If a value greater than 8 is selected, 8 bytes are transmitted in the data frame with the Data Length Code specified in DLC. The range of the Data Byte Count is 0 to 8 bytes and is coded as follows:

$$DataByteCount = DLC \quad (6)$$

8.14 Transmit Identifier Register (CAN1TID[1/2/3] - 0xE004 40[34/44/54], CAN2TID[1/2/3] - 0xE004 80[34/44/54])

When the corresponding TBS bit in CANxSR is 1, software can write to one of these registers to define the Identifier field of the next transmit message. Bits not listed read as 0 and should be written as 0. The register assumes two different formats depending on the FF bit in CANTFI.

In Standard Frame Format messages, the CAN Identifier consists of 11 bits (ID.28 to ID.18), and in Extended Frame Format messages, the CAN identifier consists of 29 bits (ID.28 to ID.0). ID.28 is the most significant bit, and it is transmitted first on the bus during the arbitration process. The Identifier acts as the message's name, used in a receiver for acceptance filtering, and also determines the bus access priority during the arbitration process.

Table 234. Transfer Identifier Register (CAN1TID[1/2/3] - address 0xE004 40[34/44/54], CAN2TID[1/2/3] - address 0xE004 80[34/44/54]) bit description

Bit	Symbol	Function	Reset Value	RM Set
10:0	ID	The 11 bit Identifier to be sent in the next transmit message.	0	X
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

Table 235. Transfer Identifier register when FF = 1

Bit	Symbol	Function	Reset Value	RM Set
28:0	ID	The 29 bit Identifier to be sent in the next transmit message.	0	X
31:29	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA	

8.15 Transmit Data Register A (CAN1TDA[1/2/3] - 0xE004 40[38/48/58], CAN2TDA[1/2/3] - 0xE004 80[38/48/58])

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the first 1 - 4 data bytes of the next transmit message. The Data Length Code defines the number of transferred data bytes. The first bit transmitted is the most significant bit of TX Data Byte 1.

Table 236. Transmit Data Register A (CAN1TDA[1/2/3] - address 0xE004 40[38/48/58], CAN2TDA[1/2/3] - address 0xE004 80[38/48/58]) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 1	If RTR = 0 and DLC Š 0001 in the corresponding CANxTFI, this byte is sent as the first Data byte of the next transmit message.	0	X
15;8	Data 2	If RTR = 0 and DLC Š 0010 in the corresponding CANxTFI, this byte is sent as the 2nd Data byte of the next transmit message.	0	X
23:16	Data 3	If RTR = 0 and DLC Š 0011 in the corresponding CANxTFI, this byte is sent as the 3rd Data byte of the next transmit message.	0	X
31:24	Data 4	If RTR = 0 and DLC Š 0100 in the corresponding CANxTFI, this byte is sent as the 4th Data byte of the next transmit message.	0	X

8.16 Transmit Data Register B (CAN1TDB[1/2/3] - 0xE004 40[3C/4C/5C], CAN2TDB[1/2/3] - 0xE004 80[3C/4C/5C])

When the corresponding TBS bit in CANSR is 1, software can write to one of these registers to define the 5th through 8th data bytes of the next transmit message. The Data Length Code defines the number of transferred data bytes. The first bit transmitted is the most significant bit of TX Data Byte 1.

Table 237. Transmit Data Register B (CAN1TDB[1/2/3] - address 0xE004 40[3C/4C/5C], CAN2TDB[1/2/3] - address 0xE004 80[3C/4C/5C]) bit description

Bit	Symbol	Function	Reset Value	RM Set
7:0	Data 5	If RTR = 0 and DLC Š 0101 in the corresponding CANTFI, this byte is sent as the 5th Data byte of the next transmit message.	0	X
15;8	Data 6	If RTR = 0 and DLC Š 0110 in the corresponding CANTFI, this byte is sent as the 6th Data byte of the next transmit message.	0	X
23:16	Data 7	If RTR = 0 and DLC Š 0111 in the corresponding CANTFI, this byte is sent as the 7th Data byte of the next transmit message.	0	X
31:24	Data 8	If RTR = 0 and DLC Š 1000 in the corresponding CANTFI, this byte is sent as the 8th Data byte of the next transmit message.	0	X

9. CAN controller operation

9.1 Error handling

The CAN Controllers count and handle transmit and receive errors as specified in CAN Spec 2.0B. The Transmit and Receive Error Counters are incriminated for each detected error and are decremented when operation is error-free. If the Transmit Error counter contains 255 and another error occurs, the CAN Controller is forced into a state called Bus-Off. In this state, the following register bits are set: BS in CANxSR, BEI and EI in CANxIR if these are enabled, and RM in CANxMOD. RM resets and disables much of the CAN Controller. Also at this time the Transmit Error Counter is set to 127 and the Receive Error Counter is cleared. Software must next clear the RM bit. Thereafter the Transmit Error Counter will count down 128 occurrences of the Bus Free condition (11 consecutive recessive bits). Software can monitor this countdown by reading the Tx Error Counter. When this countdown is complete, the CAN Controller clears BS and ES in CANxSR, and sets EI in CANxSR if EIE in IER is 1.

The Tx and Rx error counters can be written if RM in CANxMOD is 1. Writing 255 to the Tx Error Counter forces the CAN Controller to Bus-Off state. If Bus-Off (BS in CANxSR) is 1, writing any value 0 through 254 to the Tx Error Counter clears Bus-Off. When software clears RM in CANxMOD thereafter, only one Bus Free condition (11 consecutive recessive bits) is needed before operation resumes.

9.2 Sleep mode

The CAN Controller will enter sleep mode if the SM bit in the CAN Mode register is 1, no CAN interrupt is pending, and there is no activity on the CAN bus. Software can only set SM when RM in the CAN Mode register is 0; it can also set the WUIE bit in the CAN Interrupt Enable register to enable an interrupt on any wake-up condition.

The CAN Controller wakes up (and sets WUI in the CAN Interrupt register if WUIE in the CAN Interrupt Enable register is 1) in response to a) a dominant bit on the CAN bus, or b) software clearing SM in the CAN Mode register. A sleeping CAN Controller, that wakes up in response to bus activity, is not able to receive an initial message, until after it detects Bus_Free (11 consecutive recessive bits). If an interrupt is pending or the CAN bus is active when software sets SM, the wake-up is immediate.

9.3 Interrupts

Each CAN Controller produces interrupt requests for Receive, Transmit, and “other status”. The Transmit interrupt is the OR of the Transmit interrupts from the three Tx Buffers. The Receive, Transmit, and “other status” interrupts from all of the CAN controllers and the Acceptance Filter LUTerr condition are ORed into one VIC channel (see Table 6–86).

9.4 Transmit priority

If the TPM bit in the CANxMOD register is 0, multiple enabled Tx Buffers contend for the right to send their messages based on the value of their CAN Identifier (TID). If TPM is 1, they contend based on the PRIO fields in bits 7:0 of their CANxTFS registers. In both cases the smallest binary value has priority. If two (or three) transmit-enabled buffers have the same smallest value, the lowest-numbered buffer sends first.

The CAN controller selects among multiple enabled Tx Buffers dynamically, just before it sends each message.

10. Centralized CAN registers

For easy and fast access, all CAN Controller Status bits from each CAN Controller Status register are bundled together. Each defined byte of the following registers contains one particular status bit from each of the CAN controllers, in its LS bits.

All Status registers are “read-only” and allow byte, half word and word access.

10.1 Central Transmit Status Register (CANTxSR - 0xE004 0000)

Table 238. Central Transit Status Register (CANTxSR - address 0xE004 0000) bit description

Bit	Symbol	Description	Reset Value
0	TS1	When 1, the CAN controller 1 is sending a message (same as TS in the).	0
1	TS2	When 1, the CAN controller 2 is sending a message (same as TS in the CAN2GSR)	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	TBS1	When 1, all 3 Tx Buffers of the CAN1 controller are available to the CPU (same as TBS in CAN1GSR).	1
9	TBS2	When 1, all 3 Tx Buffers of the CAN2 controller are available to the CPU (same as TBS in CAN2GSR).	1
15:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 238. Central Transit Status Register (CANTxSR - address 0xE004 0000) bit description

Bit	Symbol	Description	Reset Value
16	TCS1	When 1, all requested transmissions have been completed successfully by the CAN1 controller (same as TCS in CAN1GSR).	1
17:16	TCS2	When 1, all requested transmissions have been completed successfully by the CAN2 controller (same as TCS in CAN2GSR).	1
31:18	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.2 Central Receive Status Register (CANRxSR - 0xE004 0004)

Table 239. Central Receive Status Register (CANRxSR - address 0xE004 0004) bit description

Bit	Symbol	Description	Reset Value
0	RS1	When 1, CAN1 is receiving a message (same as RS in CAN1GSR).	0
1	RS2	When 1, CAN2 is receiving a message (same as RS in CAN2GSR).	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	RB1	When 1, a received message is available in the CAN1 controller (same as RBS in CAN1GSR).	0
9	RB2	When 1, a received message is available in the CAN2 controller (same as RBS in CAN2GSR).	0
15:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
16	DOS1	When 1, a message was lost because the preceding message to CAN1 controller was not read out quickly enough (same as DOS in CAN1GSR).	0
17:16	DOS2	When 1, a message was lost because the preceding message to CAN2 controller was not read out quickly enough (same as DOS in CAN2GSR).	0
31:18	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.3 Central Miscellaneous Status Register (CANMSR - 0xE004 0008)

Table 240. Central Miscellaneous Status Register (CANMSR - address 0xE004 0008) bit description

Bit	Symbol	Description	Reset Value
0	E1	When 1, one or both of the CAN1 Tx and Rx Error Counters has reached the limit set in the CAN1EWL register (same as ES in CAN1GSR)	0
1	E2	When 1, one or both of the CAN2 Tx and Rx Error Counters has reached the limit set in the CAN2EWL register (same as ES in CAN2GSR)	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 240. Central Miscellaneous Status Register (CANMSR - address 0xE004 0008) bit description

Bit	Symbol	Description	Reset Value
8	BS1	When 1, the CAN controller is currently not involved/prohibited from bus activity (same as BS in CAN1GSR).	0
9	BS2	When 1, the CAN controller is currently not involved/prohibited from bus activity (same as BS in CAN1GSR).	0
31:10	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

11. Global acceptance filter

This block provides lookup for received Identifiers (called Acceptance Filtering in CAN terminology) for all the CAN Controllers. It includes a 512 ¥ 32 (2 kB) RAM in which software maintains one to five tables of Identifiers. This RAM can contain up to 1024 Standard Identifiers or 512 Extended Identifiers, or a mixture of both types.

12. Acceptance filter modes

The Acceptance Filter can be put into different modes by setting the according AccOff, AccBP, and eFCAN bits in the Acceptance Filter Mode Register (Section 12–15.1 “Acceptance Filter Mode Register (AFMR - 0xE003 C000)”). During each mode the access to the Configuration Register and the ID Look-up table is handled differently.

Table 241. Acceptance filter modes and access control

Acceptance filter mode	Bit AccOff	Bit AccBP	Acceptance filter state	ID Look-up table RAM[1]	Acceptance filter config. registers	CAN controller message receive
Off Mode	1	0	reset & halted	r/w access from CPU	r/w access from CPU	no messages accepted
Bypass Mode	X	1	reset & halted	r/w access from CPU	r/w access from CPU	all messages accepted
Operating Mode and FullCAN Mode	0	0	running	read only from CPU[2]	access from Acceptance filter only	hardware acceptance filtering

[1] The whole ID Look-up Table RAM is only word accessible.

[2] During the Operating Mode of the Acceptance Filter the Look-up Table can be accessed only to disable or enable Messages.

A write access to all section configuration registers is only possible during the Acceptance Filter Off and Bypass Mode. Read access is allowed in all Acceptance Filter Modes.

12.1 Acceptance filter Off mode

The Acceptance Filter Off Mode is typically used during initialization. During this mode an unconditional access to all registers and to the Look-up Table RAM is possible. With the Acceptance Filter Off Mode, CAN messages are not accepted and therefore not stored in the Receive Buffers of active CAN Controllers.

12.2 Acceptance filter Bypass mode

The Acceptance Filter Bypass Mode can be used for example to change the acceptance filter configuration during a running system, e.g. change of identifiers in the ID-Look-up Table memory. During this re-configuration, software acceptance filtering has to be used.

It is recommended to use the ID ready Interrupt (ID Index) and the Receive Interrupt (RI). In this mode all CAN message are accepted and stored in the Receive Buffers of active CAN Controllers.

12.3 Acceptance filter Operating mode

The Acceptance Filter is in Operating Mode when neither the AccOff nor the AccBP in the Configuration Register is set and the eFCAN = 0.

12.4 FullCAN mode

The Acceptance Filter is in Operating Mode when neither the AccOff nor the AccBP in the Configuration Register is set and the eFCAN = 1. More details on FullCAN mode are available in Section 12–17 “FullCAN mode”.

13. Sections of the ID look-up table RAM

Four 12-bit section configuration registers (SFF_sa, SFF_GRP_sa, EFF_sa, EFF_GRP_sa) are used to define the boundaries of the different identifier sections in the ID-Look-up Table Memory. The fifth 12-bit section configuration register, the End of Table address register (ENDofTable) is used to define the end of all identifier sections. The End of Table address is also used to assign the start address of the section where FullCAN Message Objects, if enabled are stored.

Table 242. Section configuration register settings

ID-Look up Table Section	Register	Value	Section status
FullCAN (Standard Frame Format) Identifier Section	SFF_sa	= 0x000	disabled
		> 0x000	enabled
Explicit Standard Frame Format Identifier Section	SFF_GRP_sa	= SFF_sa	disabled
		> SFF_sa	enabled
Group of Standard Frame Format Identifier Section	EFF_sa	= SFF_GRP_sa	disabled
		> SFF_GRP_sa	enabled
Explicit Extended Frame Format Identifier Section	EFF_GRP_sa	= EFF_sa	disabled
		> EFF_sa	enabled
Group of Extended Frame Format Identifier Section	ENDofTable	= EFF_GRP_sa	disabled
		> EFF_GRP_sa	enabled

14. ID look-up table RAM

The Whole ID Look-up Table RAM is only word accessible. A write access is only possible during the Acceptance Filter Off or Bypass Mode. Read access is allowed in all Acceptance Filter Modes.

If Standard (11 bit) Identifiers are used in the application, at least one of 3 tables in Acceptance Filter RAM must not be empty. If the optional “fullCAN mode” is enabled, the first table contains Standard identifiers for which reception is to be handled in this mode. The next table contains individual Standard Identifiers and the third contains ranges of Standard Identifiers, for which messages are to be received via the CAN Controllers. The tables of fullCAN and individual Standard Identifiers must be arranged in ascending numerical order, one per halfword, two per word. Since each CAN bus has its own address map, each entry also contains the number of the CAN Controller (001-010) to which it applies.

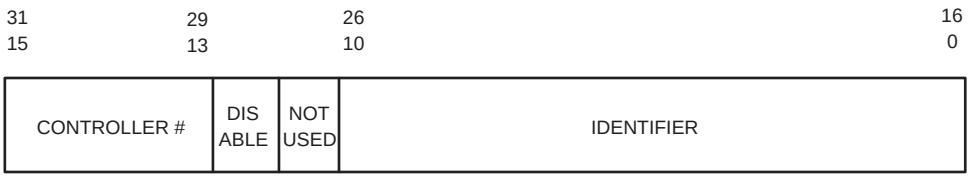

Fig 46. Entry in FullCAN and individual standard identifier tables

The table of Standard Identifier Ranges contains paired upper and lower (inclusive) bounds, one pair per word. These must also be arranged in ascending numerical order.

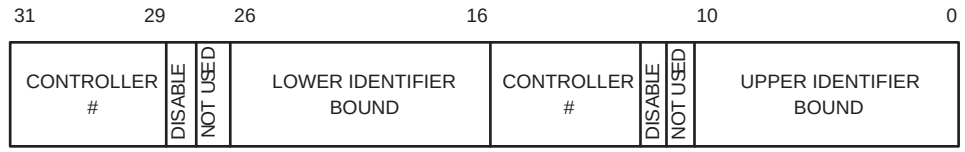

Fig 47. Entry in standard identifier range table

The disable bits in Standard entries provide a means to turn response, to particular CAN Identifiers or ranges of Identifiers, on and off dynamically. When the Acceptance Filter function is enabled, only the disable bits in Acceptance Filter RAM can be changed by software. Response to a range of Standard addresses can be enabled by writing 32 zero bits to its word in RAM, and turned off by writing 32 one bits (0xFFFF FFFF) to its word in RAM. Only the disable bits are actually changed. Disabled entries must maintain the ascending sequence of Identifiers.

If Extended (29 bit) Identifiers are used in the application, at least one of the other two tables in Acceptance Filter RAM must not be empty, one for individual Extended Identifiers and one for ranges of Extended Identifiers. The table of individual Extended Identifiers must be arranged in ascending numerical order.

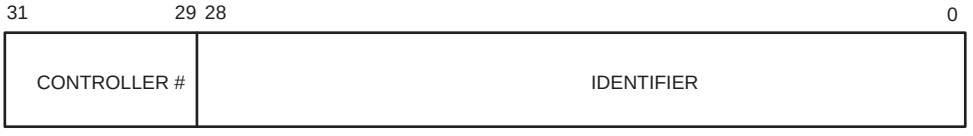

Fig 48. Entry in either extended identifier table

The table of ranges of Extended Identifiers must contain an even number of entries, of the same form as in the individual Extended Identifier table. Like the Individual Extended table, the Extended Range must be arranged in ascending numerical order. The first and second (3rd and 4th …) entries in the table are implicitly paired as an inclusive range of Extended addresses, such that any received address that falls in the inclusive range is received (accepted). Software must maintain the table to consist of such word pairs.

There is no facility to receive messages to Extended identifiers using the fullCAN method.

Five address registers point to the boundaries between the tables in Acceptance Filter RAM: fullCAN Standard addresses, Standard Individual addresses, Standard address ranges, Extended Individual addresses, and Extended address ranges. These tables must be consecutive in memory. The start of each of the latter four tables is implicitly the end of the preceding table. The end of the Extended range table is given in an End of Tables register. If the start address of a table equals the start of the next table or the End Of Tables register, that table is empty.

When the Receive side of a CAN controller has received a complete Identifier, it signals the Acceptance Filter of this fact. The Acceptance Filter responds to this signal, and reads the Controller number, the size of the Identifier, and the Identifier itself from the Controller. It then proceeds to search its RAM to determine whether the message should be received or ignored.

If fullCAN mode is enabled and the CAN controller signals that the current message contains a Standard identifier, the Acceptance Filter first searches the table of identifiers for which reception is to be done in fullCAN mode. Otherwise, or if the AF doesn't find a match in the fullCAN table, it searches its individual Identifier table for the size of Identifier signalled by the CAN controller. If it finds an equal match, the AF signals the CAN controller to retain the message, and provides it with an ID Index value to store in its Receive Frame Status register.

If the Acceptance Filter does not find a match in the appropriate individual Identifier table, it then searches the Identifier Range table for the size of Identifier signalled by the CAN controller. If the AF finds a match to a range in the table, it similarly signals the CAN controller to retain the message, and provides it with an ID Index value to store in its Receive Frame Status register. If the Acceptance Filter does not find a match in either the individual or Range table for the size of Identifier received, it signals the CAN controller to discard/ignore the received message.

15. Acceptance filter registers

15.1 Acceptance Filter Mode Register (AFMR - 0xE003 C000)

The AccBP and AccOff bits of the acceptance filter mode register are used for putting the acceptance filter into the Bypass and Off mode. The eFCAN bit of the mode register can be used to activate a FullCAN mode enhancement for received 11-bit CAN ID messages.

Table 243. Acceptance Filter Mode Register (AFMR - address 0xE003 C000) bit description

Bit	Symbol	Value	Description	Reset Value
0	AccOff[2]	1	if AccBP is 0, the Acceptance Filter is not operational. All Rx messages on all CAN buses are ignored.	1
1	AccBP[1]	1	All Rx messages are accepted on enabled CAN controllers. Software must set this bit before modifying the contents of any of the registers described below, and before modifying the contents of Lookup Table RAM in any way other than setting or clearing Disable bits in Standard Identifier entries. When both this bit and AccOff are 0, the Acceptance filter operates to screen received CAN Identifiers.	0
2	eFCAN[3]	0	Software must read all messages for all enabled IDs on all enabled CAN buses, from the receiving CAN controllers.	0
		1	The Acceptance Filter itself will take care of receiving and storing messages for selected Standard ID values on selected CAN buses. See Section 12–17 “FullCAN mode” on page 297.	
31:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Acceptance Filter Bypass Mode (AccBP): By setting the AccBP bit in the Acceptance Filter Mode Register, the Acceptance filter is put into the Acceptance Filter Bypass mode. During bypass mode, the internal state machine of the Acceptance Filter is reset and halted. All received CAN messages are accepted, and acceptance filtering can be done by software.

[2] Acceptance Filter Off mode (AccOff): After power-upon hardware reset, the Acceptance filter will be in Off mode, the AccOff bit in the Acceptance filter Mode register 0 will be set to 1. The internal state machine of the acceptance filter is reset and halted. If not in Off mode, setting the AccOff bit, either by hardware or by software, will force the acceptance filter into Off mode.

[3] FullCan Mode Enhancements: A FullCan mode for received CAN messages can be enabled by setting the eFCAN bit in the acceptance filter mode register.

15.2 Section configuration registers

The 10 bit section configuration registers are used for the ID look-up table RAM to indicate the boundaries of the different sections for explicit and group of CAN identifiers for 11 bit CAN and 29 bit CAN identifiers, respectively. The 10 bit wide section configuration registers allow the use of a 512x32 (2 kB) look-up table RAM. The whole ID Look-up Table RAM is only word accessible. All five section configuration registers contain APB addresses for the acceptance filter RAM and do not include the APB base address. A write access to all section configuration registers is only possible during the Acceptance filter off and Bypass modes. Read access is allowed in all acceptance filter modes.

15.3 Standard Frame Individual Start Address Register (SFF_sa - 0xE003 C004)

Table 244. Standard Frame Individual Start Address Register (SFF_sa - address 0xE003 C004) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
10:2	SFF_sa[1]	The start address of the table of individual Standard Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the SFF_GRP_sa register described below. For compatibility with possible future devices, write zeroes in bits 31:11 and 1:0 of this register. If the eFCAN bit in the AFMR is 1, this value also indicates the size of the table of Standard IDs which the Acceptance Filter will search and (if found) automatically store received messages in Acceptance Filter RAM.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

15.4 Standard Frame Group Start Address Register (SFF_GRP_sa - 0xE003 C008)

Table 245. Standard Frame Group Start Address Register (SFF_GRP_sa - address 0xE003 C008) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:2	SFF_GRP_sa[1]	The start address of the table of grouped Standard Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the EFF_sa register described below. The largest value that should be written to this register is 0x800, when only the Standard Individual table is used, and the last word (address 0x7FC) in AF Lookup Table RAM is used. For compatibility with possible future devices, please write zeroes in bits 31:12 and 1:0 of this register.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

15.5 Extended Frame Start Address Register (EFF_sa - 0xE003 C00C)

Table 246. Extended Frame Start Address Register (EFF_sa - address 0xE003 C00C) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
10:2	EFF_sa[1]	The start address of the table of individual Extended Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the EFF_GRP_sa register described below. The largest value that should be written to this register is 0x800, when both Extended Tables are empty and the last word (address 0x7FC) in AF Lookup Table RAM is used. For compatibility with possible future devices, please write zeroes in bits 31:11 and 1:0 of this register.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

15.6 Extended Frame Group Start Address Register (EFF_GRP_sa - 0xE003 C010)

Table 247. Extended Frame Group Start Address Register (EFF_GRP_sa - address 0xE003 C010) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:2	Eff_GRP_sa[1]	The start address of the table of grouped Extended Identifiers in AF Lookup RAM. If the table is empty, write the same value in this register and the ENDofTable register described below. The largest value that should be written to this register is 0x800, when this table is empty and the last word (address 0x7FC) in AF Lookup Table RAM is used. For compatibility with possible future devices, please write zeroes in bits 31:12 and 1:0 of this register.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

15.7 End of AF Tables Register (ENDofTable - 0xE003 C014)

Table 248. End of AF Tables Register (ENDofTable - address 0xE003 C014) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
11:2	EndofTable [1]	The address above the last active address in the last active AF table. For compatibility with possible future devices, please write zeroes in bits 31:12 and 1:0 of this register. If the eFCAN bit in the AFMR is 0, the largest value that should be written to this register is 0x800, which allows the last word (address 0x7FC) in AF Lookup Table RAM to be used. If the eFCAN bit in the AFMR is 1, this value marks the start of the area of Acceptance Filter RAM, into which the Acceptance Filter will automatically receive messages for selected IDs on selected CAN buses. In this case, the maximum value that should be written to this register is 0x800 minus 6 times the value in SFF_sa. This allows 12 bytes of message storage between this address and the end of Acceptance Filter RAM, for each Standard ID that is specified between the start of Acceptance Filter RAM, and the next active AF table.	0
31:12	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] Write access to the look-up table section configuration registers are possible only during the Acceptance filter bypass mode or the Acceptance filter off mode.

15.8 Status registers

The look-up table error status registers, the error addresses, and the flag register provide information if a programming error in the look-up table RAM during the ID screening was encountered. The look-up table error address and flag register have only read access. If an error is detected, the LUTerror flag is set, and the LUTerrorAddr register provides the information under which address during an ID screening an error in the look-up table was encountered. Any read of the LUTerrorAddr Filter block can be used for a look-up table interrupt.

15.9 LUT Error Address Register (LUTerrAd - 0xE003 C018)

Table 249. LUT Error Address Register (LUTerrAd - address 0xE003 C018) bit description

Bit	Symbol	Description	Reset Value
1:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
10:2	LUTerrAd	It the LUT Error bit (below) is 1, this read-only field contains the address in AF Lookup Table RAM, at which the Acceptance Filter encountered an error in the content of the tables.	0
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

15.10 LUT Error Register (LUTerr - 0xE003 C01C)

Table 250. LUT Error Register (LUTerr - address 0xE003 C01C) bit description

Bit	Symbol	Description	Reset Value
0	LUTerr	This read-only bit is set to 1 if the Acceptance Filter encounters an error in the content of the tables in AF RAM. It is cleared when software reads the LUTerrAd register. This condition is ORed with the “other CAN” interrupts from the CAN controllers, to produce the request for a VIC interrupt channel.	0
31:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

15.11 Global FullCANInterrupt Enable register (FCANIE - 0xE003 C020)

A write access to the Global FullCAN Interrupt Enable register is only possible when the Acceptance Filter is in the off mode.

Table 251. Global FullCAN Enable register (FCANIE - address 0xE003 C020) bit description

Bit	Symbol	Description	Reset Value
0	FCANIE	Global FullCAN Interrupt Enable. When 1, this interrupt is enabled.	0
31:1	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

15.12 FullCAN Interrupt and Capture registers (FCANIC0 - 0xE003 C024 and FCANIC1 - 0xE003 C028)

For detailed description on these two registers, see Section 12–17.2 “FullCAN interrupts”.

Table 252. FullCAN Interrupt and Capture register 0 (FCANIC0 - address 0xE003 C024) bit description

Bit	Symbol	Description	Reset Value
0	IntPnd0	FullCan Interrupt Pending bit 0.	0
...	IntPndx (0<x<31)	FullCan Interrupt Pending bit x.	0
31	IntPnd31	FullCan Interrupt Pending bit 31.	0

Table 253. FullCAN Interrupt and Capture register 1 (FCANIC1 - address 0xE003 C028) bit description

Bit	Symbol	Description	Reset Value
0	IntPnd32	FullCan Interrupt Pending bit 32.	0
...	IntPndx (32<x<63)	FullCan Interrupt Pending bit x.	0
31	IntPnd63	FullCan Interrupt Pending bit 63.	0

16. Configuration and search algorithm

The CAN Identifier Look-up Table Memory can contain explicit identifiers and groups of CAN identifiers for Standard and Extended CAN Frame Formats. They are organized as a sorted list or table with an increasing order of the Source CAN Channel (SCC) together with CAN Identifier in each section.

SCC value equals CAN_controller - 1, i.e., SCC = 0 matches CAN1 and SCC = 1 matches CAN2.

Every CAN identifier is linked to an ID Index number. In case of a CAN Identifier match, the matching ID Index is stored in the Identifier Index of the Frame Status Register (CANRFS) of the according CAN Controller.

16.1 Acceptance filter search algorithm

The identifier screening process of the acceptance filter starts in the following order:

1. FullCAN (Standard Frame Format) Identifier Section
2. Explicit Standard Frame Format Identifier Section
3. Group of Standard Frame Format Identifier Section
4. Explicit Extended Frame Format Identifier Section
5. Group of Extended Frame Format Identifier Section

Note: Only activated sections will take part in the screening process.

In cases where equal message identifiers of same frame format are defined in more than one section, the first match will end the screening process for this identifier.

For example, if the same Source CAN Channel in conjunction with the identifier is defined in the FullCAN, the Explicit Standard Frame Format and the Group of Standard Frame Format Identifier Sections, the screening will already be finished with the match in the FullCAN section.

In the example of Figure 12–49, Identifiers with their Source CAN Channel have been defined in the FullCAN, Explicit and Group of Standard Frame Format Identifier Sections. This example corresponds to a LPC2290 compatible part that would have 6 CAN controllers.

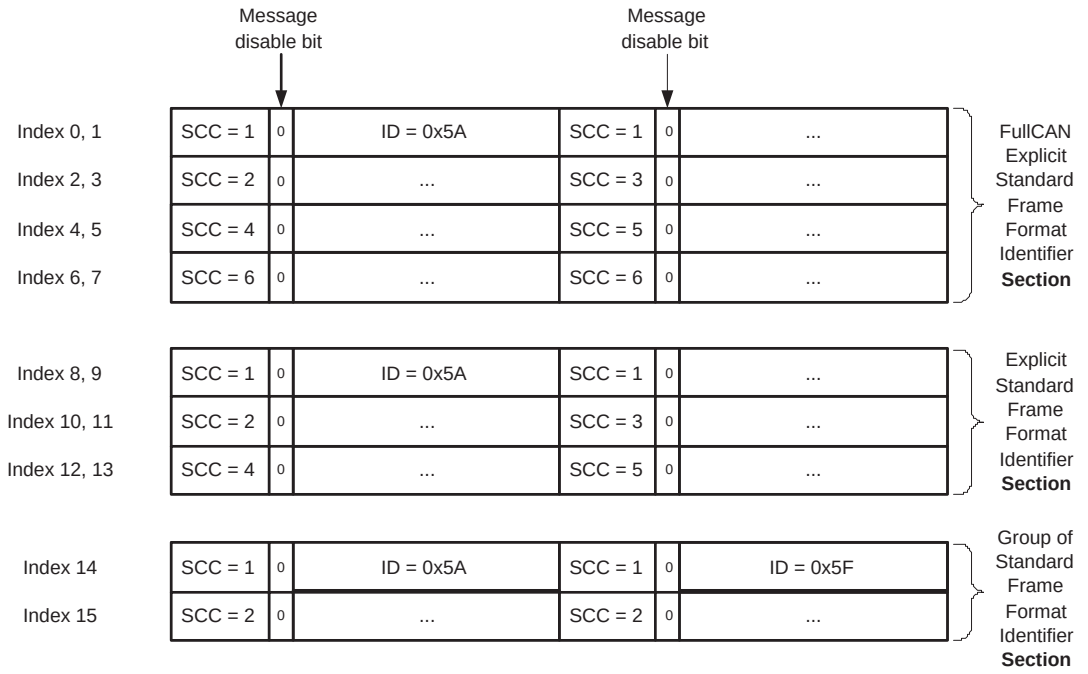

Fig 49. ID Look-up table example explaining the search algorithm

The identifier 0x5A of the CAN Controller 1 with the Source CAN Channel SCC = 1, is defined in all three sections. With this configuration incoming CAN messages on CAN Controller 1 with a 0x5A identifier will find a match in the FullCAN section.

It is possible to disable the '0x5A identifier' in the FullCAN section. With that, the screening process would be finished with the match in the Explicit Identifier Section.

The first group in the Group Identifier Section has been defined in that way, that incoming CAN messages with identifiers of 0x5A up to 0x5F are accepted on CAN Controller 1 with the Source CAN Channel SCC = 1. As stated above, the identifier 0x5A would find a match already in the FullCAN or in the Explicit Identifier section if enabled. The rest of the defined identifiers of this group (0x5B to 0x5F) will find a match in this Group Identifier Section.

This way the user can switch dynamically between different filter modes for same identifiers.

17. FullCAN mode

The FullCAN mode is based on capabilities provided by the CAN Gateway module used in the LPC2000 family of products. This block uses the Acceptance Filter to provide filtering for both CAN channels.

The concept of the CAN Gateway block is mainly based on a BasicCAN functionality. This concept fits perfectly in systems where a gateway is used to transfer messages or message data between different CAN channels. A BasicCAN device is generating a

receive interrupt whenever a CAN message is accepted and received. Software has to move the received message out of the receive buffer from the according CAN controller into the user RAM.

To cover dashboard like applications where the controller typically receives data from several CAN channels for further processing, the CAN Gateway block was extended by a so-called FullCAN receive function. This additional feature uses an internal message handler to move received FullCAN messages from the receive buffer of the according CAN controller into the FullCAN message object data space of Look-up Table RAM.

When fullCAN mode is enabled, the Acceptance Filter itself takes care of receiving and storing messages for selected Standard ID values on selected CAN buses, in the style of “FullCAN” controllers.

In order to set this bit and use this mode, two other conditions must be met with respect to the contents of Acceptance Filter RAM and the pointers into it:

- The Standard Frame Individual Start Address Register (SFF_sa) must be greater than or equal to the number of IDs for which automatic receive storage is to be done, times two. SFF_sa must be rounded up to a multiple of 4 if necessary.
- The EndOfTable register must be less than or equal to 0x800 minus 6 times the SFF_sa value, to allow 12 bytes of message storage for each ID for which automatic receive storage will be done.

When these conditions are met and eFCAN is set:

- The area between the start of Acceptance Filter RAM and the SFF_sa address, is used for a table of individual Standard IDs and CAN Controller/bus identification, sorted in ascending order and in the same format as in the Individual Standard ID table (see Figure 12–46 “Entry in FullCAN and individual standard identifier tables” on page 289). Entries can be marked as “disabled” as in the other Standard tables. If there are an odd number of “FullCAN” ID’s, at least one entry in this table must be so marked.
- The first (SFF_sa)/2 IDindex values are assigned to these automatically-stored ID’s. That is, IDindex values stored in the Rx Frame Status Register, for IDs not handled in this way, are increased by (SFF_sa)/2 compared to the values they would have when eFCAN is 0.
- When a Standard ID is received, the Acceptance Filter searches this table before the Standard Individual and Group tables.
- When a message is received for a controller and ID in this table, the Acceptance filter reads the received message out of the CAN controller and stores it in Acceptance Filter RAM, starting at (EndOfTable) + its IDindex*12.
- The format of such messages is shown in Table 12–254.

17.1 FullCAN message layout

Table 254. Format of automatically stored Rx messages

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FF	RTR	0000				SEM [1:0]		0000				DLC				00000						ID.28 ... ID.18									
+4	Rx Data 4								Rx Data 3								Rx Data 2								Rx Data 1							
+8	Rx Data 8								Rx Data 7								Rx Data 6								Rx Data 5							

The FF, RTR, and DLC fields are as described in Table 12–228.

Since the FullCAN message object section of the Look-up table RAM can be accessed both by the Acceptance Filter and the CPU, there is a method for insuring that no CPU reads from FullCAN message object occurs while the Acceptance Filter hardware is writing to that object.

For this purpose the Acceptance Filter uses a 3-state semaphore, encoded with the two semaphore bits SEM1 and SEM0 (see Table 12–254 “Format of automatically stored Rx messages”) for each message object. This mechanism provides the CPU with information about the current state of the Acceptance Filter activity in the FullCAN message object section.

The semaphore operates in the following manner:

Table 255. FullCAN semaphore operation

SEM1	SEM0	activity
0	1	Acceptance Filter is updating the content
1	1	Acceptance Filter has finished updating the content
0	0	CPU is in process of reading from the Acceptance Filter

Prior to writing the first data byte into a message object, the Acceptance Filter will write the FrameInfo byte into the according buffer location with SEM[1:0] = 01.

After having written the last data byte into the message object, the Acceptance Filter will update the semaphore bits by setting SEM[1:0] = 11.

Before reading a message object, the CPU should read SEM[1:0] to determine the current state of the Acceptance Filter activity therein. If SEM[1:0] = 01, then the Acceptance Filter is currently active in this message object. If SEM[1:0] = 11, then the message object is available to be read.

Before the CPU begins reading from the message object, it should clear SEM[1:0] = 00.

When the CPU is finished reading, it can check SEM[1:0] again. At the time of this final check, if SEM[1:0] = 01 or 11, then the Acceptance Filter has updated the message object during the time when the CPU reads were taking place, and the CPU should discard the data. If, on the other hand, SEM[1:0] = 00 as expected, then valid data has been successfully read by the CPU.

Figure 12–50 shows how software should use the SEM field to ensure that all three words read from the message are all from the same received message.

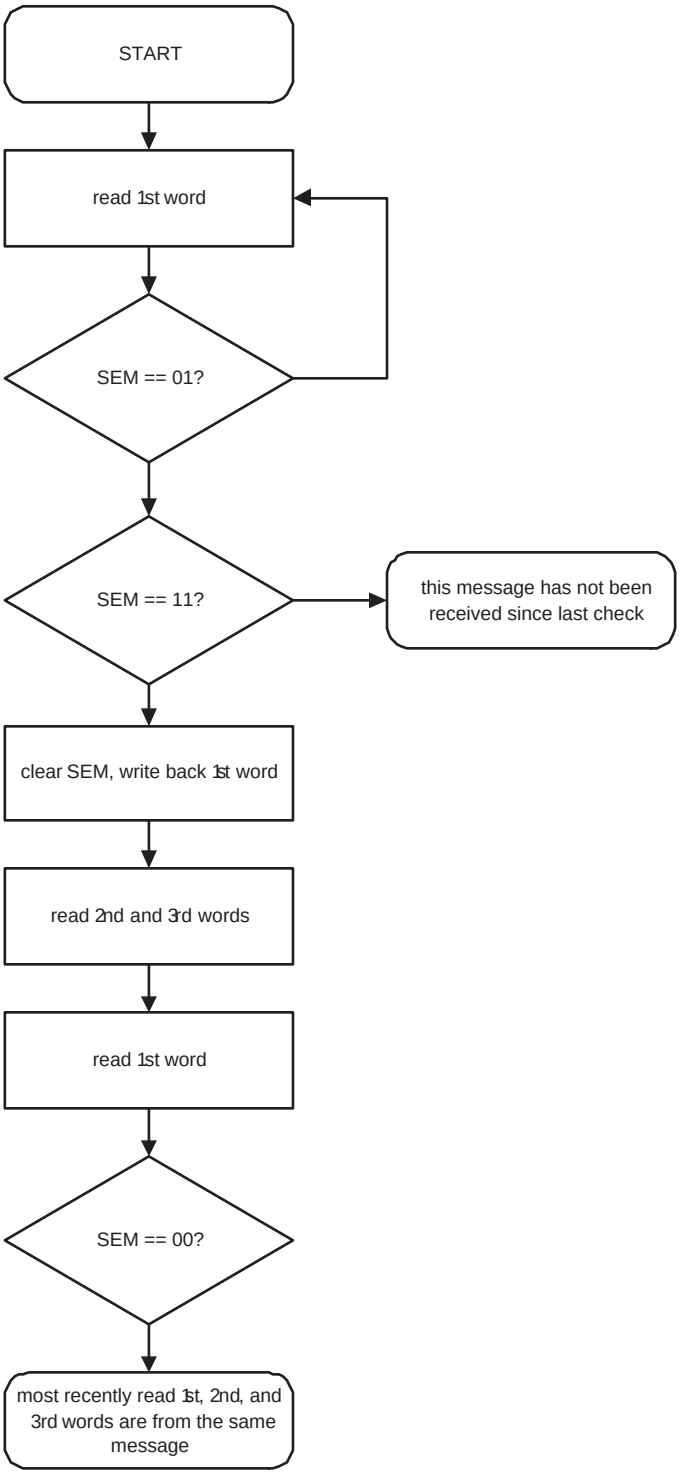

Fig 50. Semaphore procedure for reading an auto-stored message

17.2 FullCAN interrupts

The CAN Gateway Block contains a 2 kB ID Look-up Table RAM. With this size a maximum number of 146 FullCAN objects can be defined if the whole Look-up Table RAM is used for FullCAN objects only. Only the first 64 FullCAN objects can be configured to participate in the interrupt scheme. It is still possible to define more than 64 FullCAN objects. The only difference is, that the remaining FullCAN objects will not provide a FullCAN interrupt.

The FullCAN Interrupt Register-set contains interrupt flags (IntPndx) for (pending) FullCAN receive interrupts. As soon as a FullCAN message is received, the according interrupt bit (IntPndx) in the FCAN Interrupt Register gets asserted. In case that the Global FullCAN Interrupt Enable bit is set, the FullCAN Receive Interrupt is passed to the Vectored Interrupt Controller.

Application Software has to solve the following:

1. Index/Object number calculation based on the bit position in the FCANIC Interrupt Register for more than one pending interrupt.
2. Interrupt priority handling if more than one FullCAN receive interrupt is pending.

The software that covers the interrupt priority handling has to assign a receive interrupt priority to every FullCAN object. If more than one interrupt is pending, then the software has to decide, which received FullCAN object has to be served next.

To each FullCAN object a new FullCAN Interrupt Enable bit (FCANIntxEn) is added, so that it is possible to enable or disable FullCAN interrupts for each object individually. The new Message Lost flag (MsgLstx) is introduced to indicate whether more than one FullCAN message has been received since last time this message object was read by the CPU. The Interrupt Enable and the Message Lost bits reside in the existing Look-up Table RAM.

17.2.1 FullCAN message interrupt enable bit

In Figure 12–51 8 FullCAN Identifiers with their Source CAN Channel are defined in the FullCAN, Section. The new introduced FullCAN Message Interrupt enable bit can be used to enable for each FullCAN message an Interrupt.

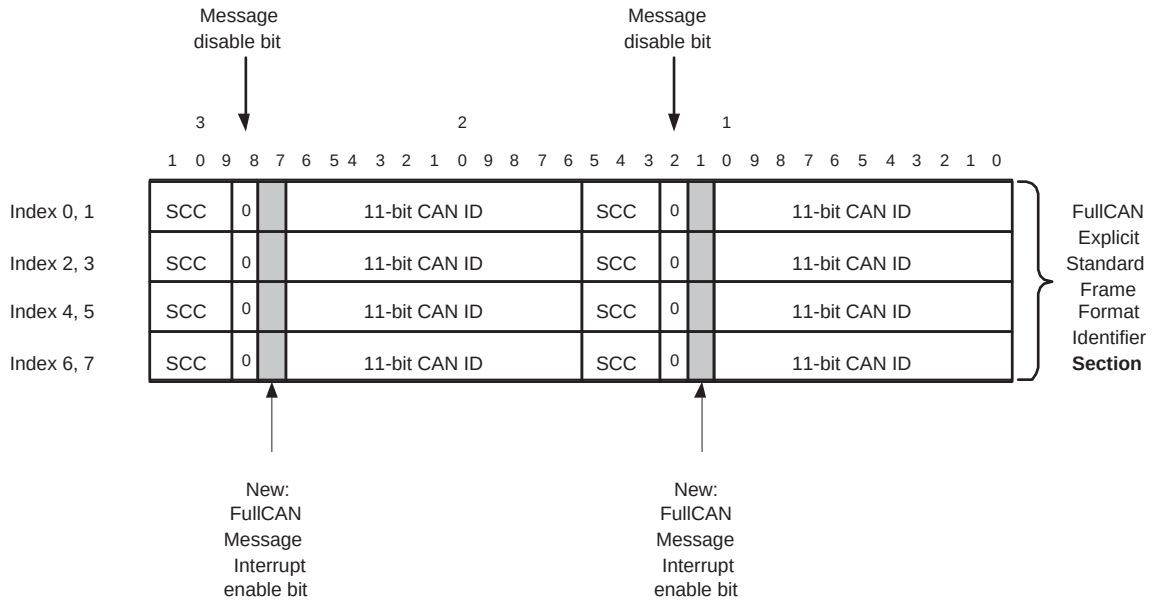

Fig 51. FullCAN section example of the ID look-up table

17.2.2 Message lost bit and CAN channel number

Figure 12–52 is the detailed layout structure of one FullCAN message stored in the FullCAN message object section of the Look-up Table.

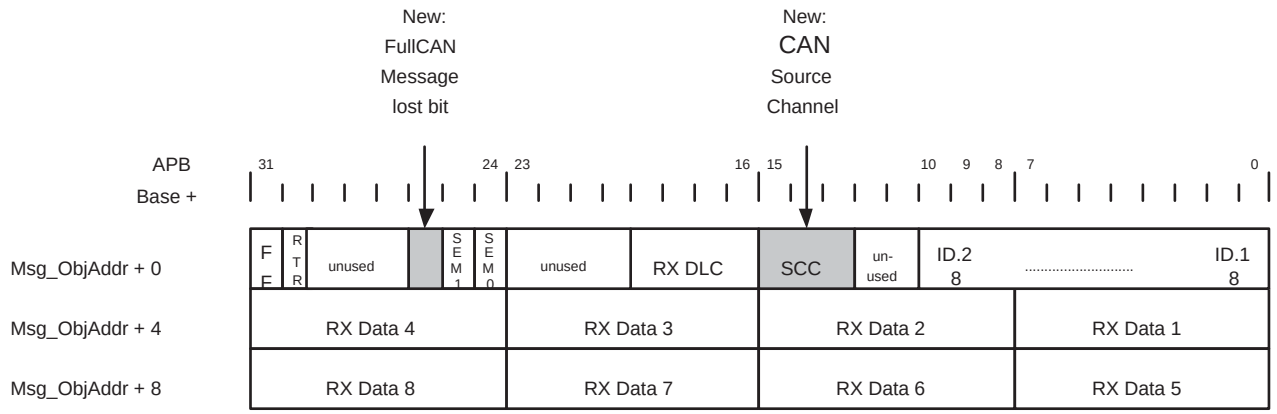

Fig 52. FullCAN message object layout

The new message lost bit (MsgLst) is introduced to indicate whether more than one FullCAN message has been received since last time this message object was read. For more information the CAN Source Channel (SCC) of the received FullCAN message is added to Message Object.

17.2.3 Setting the interrupt pending bits (IntPnd 63 to 0)

The interrupt pending bit (IntPndx) gets asserted in case of an accepted FullCAN message and if the interrupt of the according FullCAN Object is enabled (enable bit FCANIntxEn) is set).

During the **last write access** from the data storage of a FullCAN message object the interrupt pending bit of a FullCAN object (IntPndx) gets asserted.

17.2.4 Clearing the interrupt pending bits (IntPnd 63 to 0)

Each of the FullCAN Interrupt Pending requests gets cleared when the semaphore bits of a message object are cleared by Software (ARM CPU).

17.2.5 Setting the message lost bit of a FullCAN message object (MsgLost 63 to 0)

The Message Lost bit of a FullCAN message object gets asserted in case of an accepted FullCAN message and when the FullCAN Interrupt of the same object is asserted already.

During the **first write access** from the data storage of a FullCAN message object the Message Lost bit of a FullCAN object (MsgLostx) gets asserted if the interrupt pending bit is set already.

17.2.6 Clearing the message lost bit of a FullCAN message object (MsgLost 63 to 0)

The Message Lost bit of a FullCAN message object gets cleared when the FullCAN Interrupt of the same object is not asserted.

During the **first write access** from the data storage of a FullCAN message object the Message Lost bit of a FullCAN object (MsgLostx) gets cleared if the interrupt pending bit is not set.

17.3 Set and clear mechanism of the FullCAN interrupt

Special precaution is needed for the built-in set and clear mechanism of the FullCAN Interrupts. The following text illustrates how the already existing Semaphore Bits (see Section 12–17.1 “FullCAN message layout” for more details) and how the new introduced features (IntPndx, MsgLstx) will behave.

17.3.1 Scenario 1: Normal case, no message lost

Figure 12–53 below shows a typical “normal” scenario in which an accepted FullCAN message is stored in the FullCAN Message Object Section. After storage the message is read out by Software (ARM CPU).

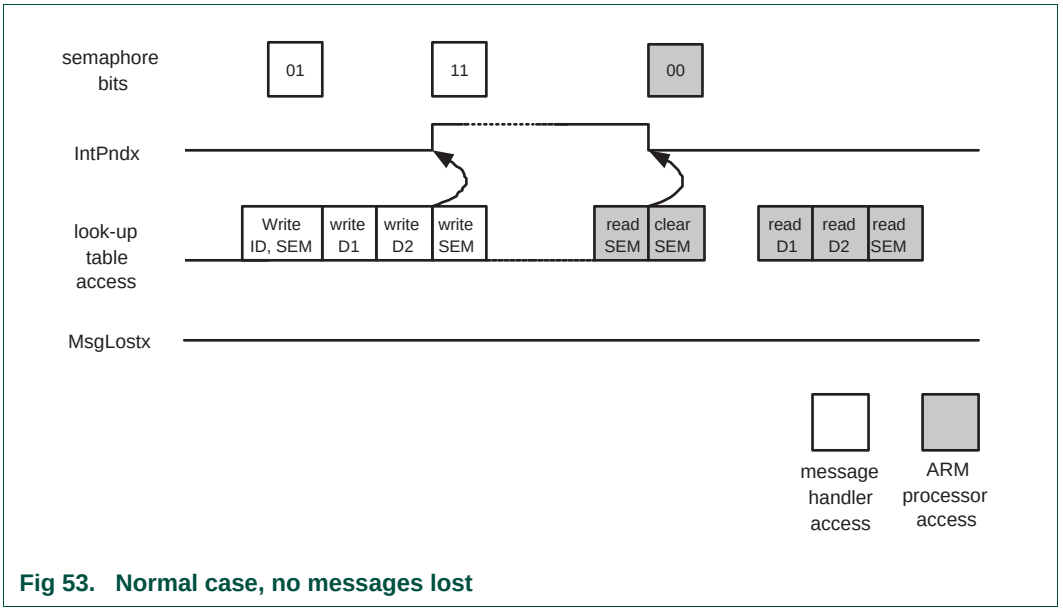

Fig 53. Normal case, no messages lost

17.3.2 Scenario 2: Message lost

In this scenario a first FullCAN Message is stored and read out by Software (1st Object write and read). In a second course a second message is stored (2nd Object write) but not read out before a third message gets stored (3rd Object write). Since the FullCAN Interrupt of that Object (IntPndx) is already asserted, the Message Lost Signal gets asserted.

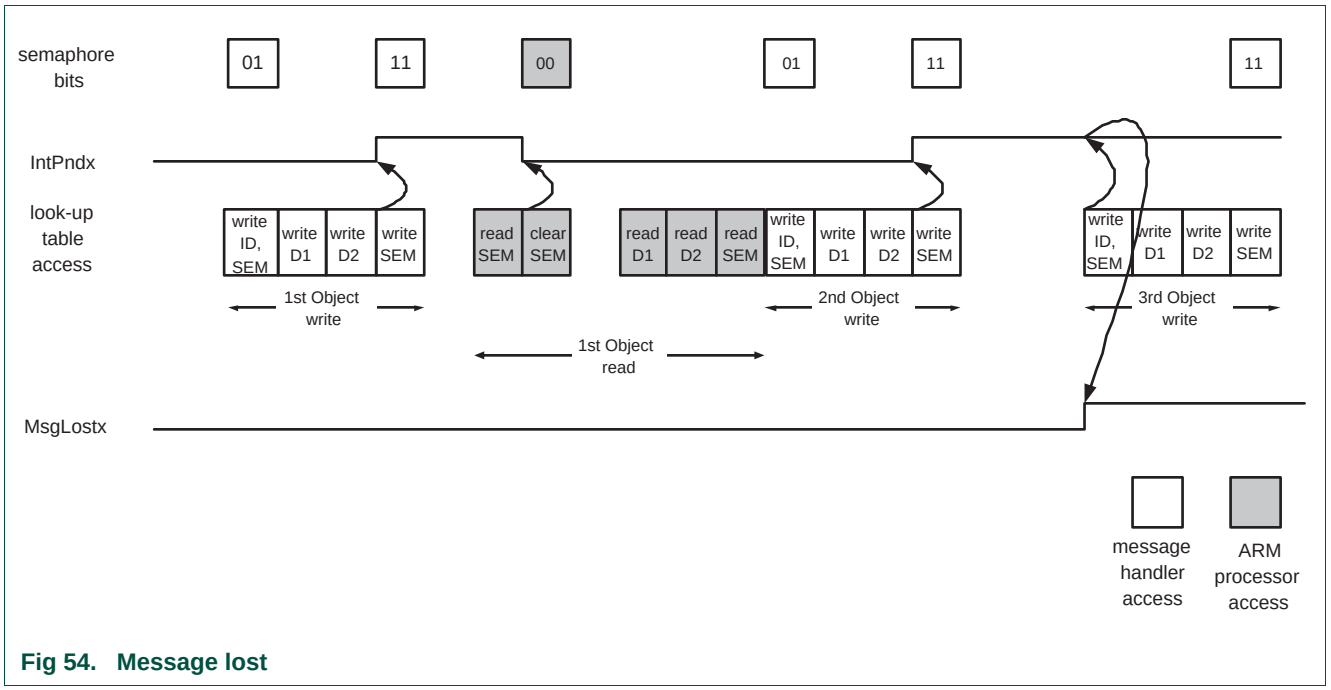

Fig 54. Message lost

17.3.3 Scenario 3: Message gets overwritten indicated by Semaphore bits

This scenario is a special case in which the lost message is indicated by the existing semaphore bits. The scenario is entered, if during a Software read of a message object another new message gets stored by the message handler. In this case, the FullCAN Interrupt bit gets set for a second time with the 2nd Object write.

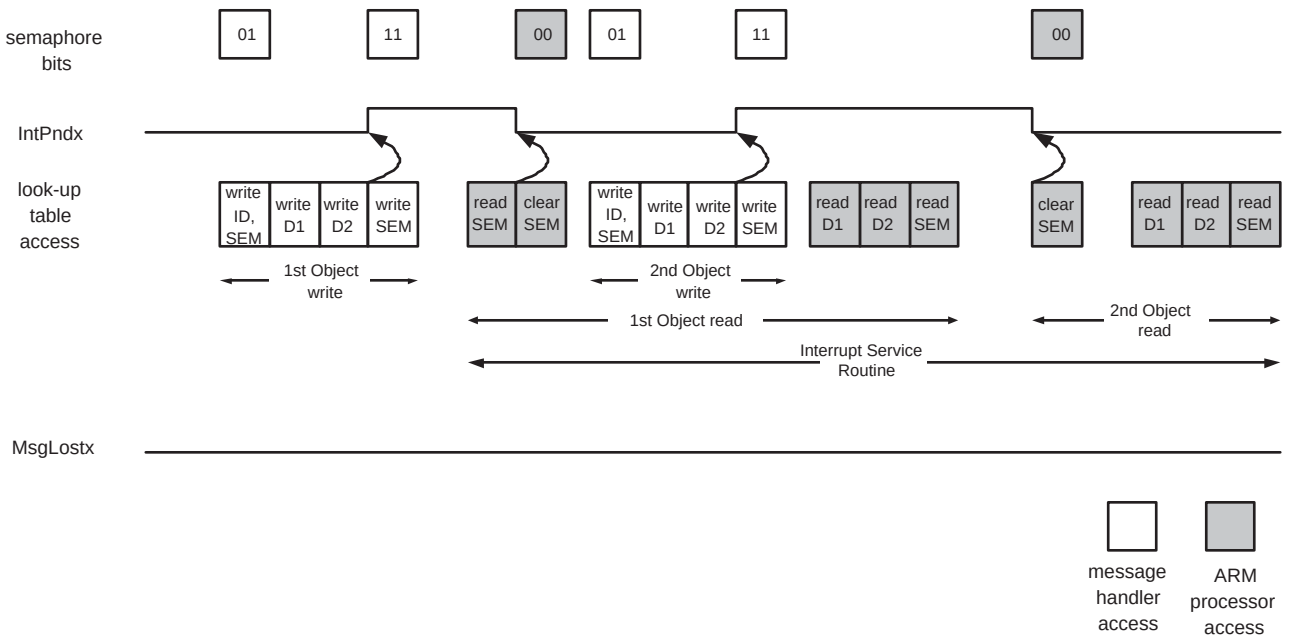

Fig 55. Message gets overwritten

17.3.4 Scenario 3.1: Message gets overwritten indicated by Semaphore bits and Message Lost

This scenario is a sub-case to Scenario 3 in which the lost message is indicated by the existing semaphore bits and by Message Lost.

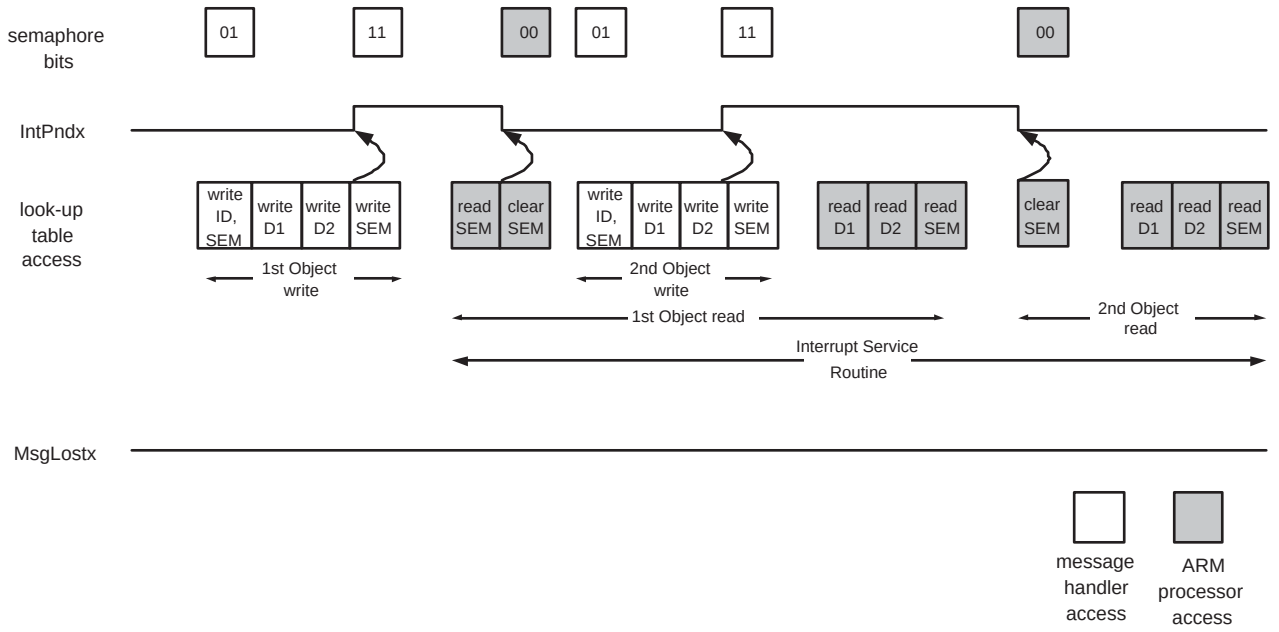

Fig 56. Message overwritten indicated by semaphore bits and message lost

17.3.5 Scenario 3.2: Message gets overwritten indicated by Message Lost

This scenario is a sub-case to Scenario 3 in which the lost message is indicated by Message Lost.

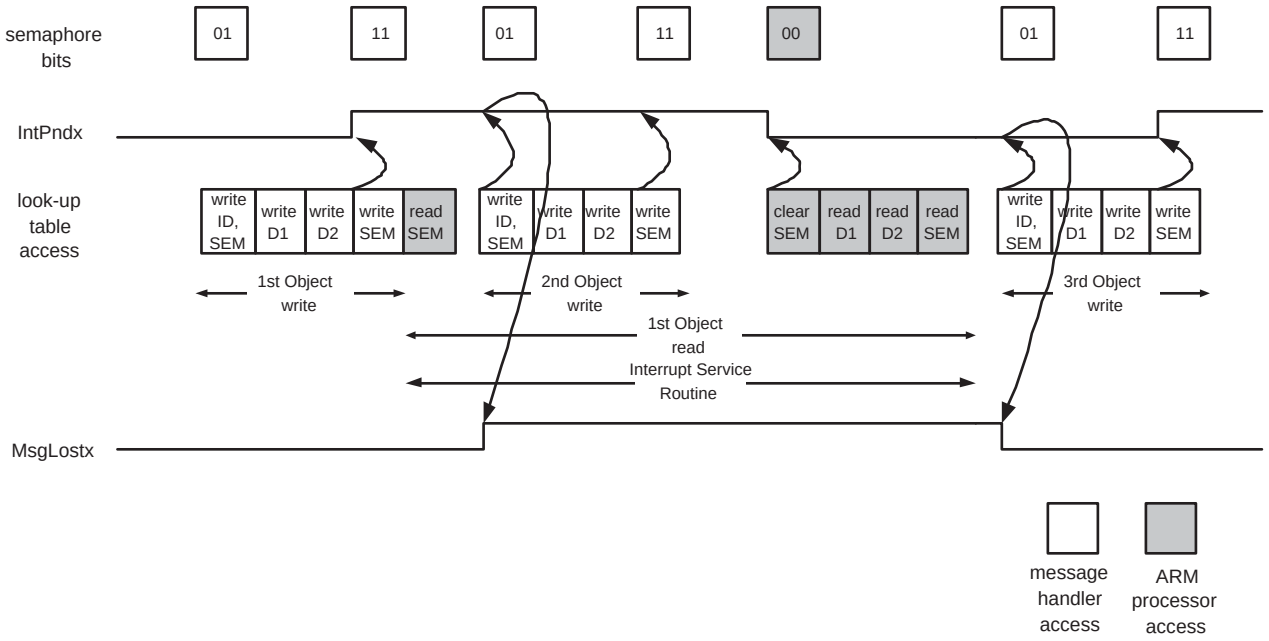

Fig 57. Message overwritten indicated by message lost

17.3.6 Scenario 4: Clearing Message Lost bit

This scenario is a special case in which the lost message bit of an object gets set during an overwrite of a none read message object (2nd Object write). The subsequent read out of that object by Software (1st Object read) clears the pending Interrupt. The 3rd Object write clears the Message Lost bit. Every “write ID, SEM” clears Message Lost bit if no pending Interrupt of that object is set.

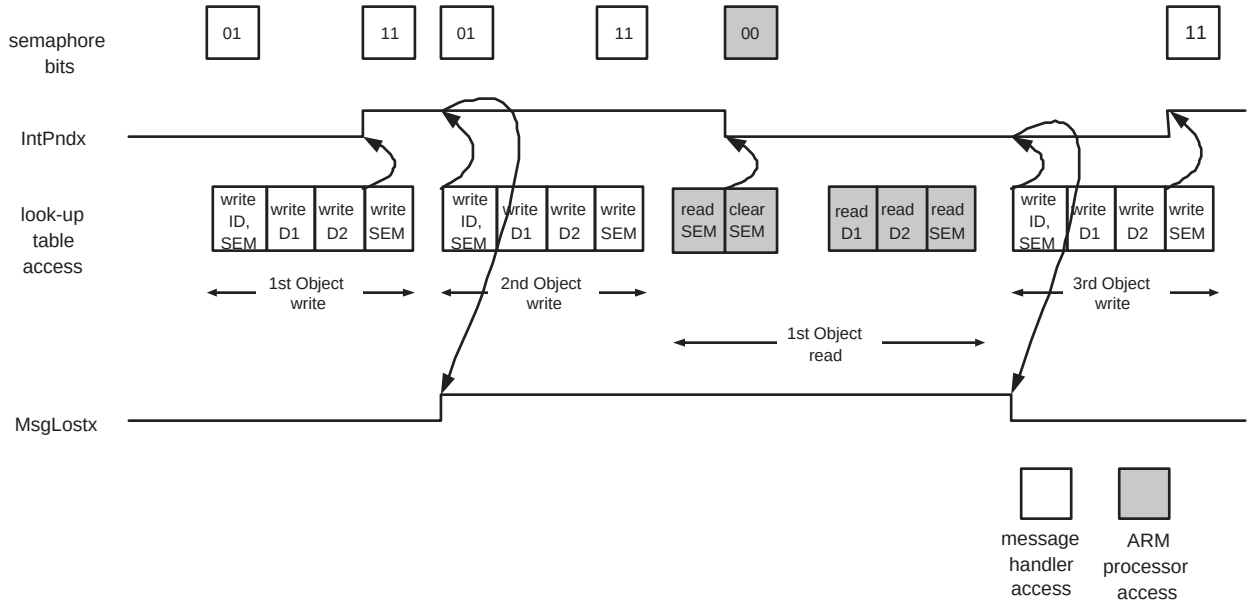

Fig 58. Clearing message lost

18. Examples of acceptance filter tables and ID index values

18.1 Example 1: only one section is used

```
SFF_sa     <    ENDofTable      OR
SFF_GRP_sa <    ENDofTable      OR
EFF_sa     <    ENDofTable      OR
EFF_GRP_sa <    ENDofTable
```

The start address of a section is lower than the end address of all programmed CAN identifiers.

18.2 Example 2: all sections are used

```
SFF_sa     <    SFF_GRP_sa      AND
SFF_GRP_sa <    EFF_sa          AND
EFF_sa     <    EFF_GRP_sa      AND
EFF_GRP_sa <    ENDofTable
```

In cases of a section not being used, the start address has to be set onto the value of the next section start address.

18.3 Example 3: more than one but not all sections are used

If the SFF group is not used, the start address of the SFF Group Section (SFF_GRP_sa register) has to be set to the same value of the next section start address, in this case the start address of the Explicit SFF Section (SFF_sa register).

In cases where explicit identifiers as well as groups of the identifiers are programmed, a CAN identifier search has to start in the explicit identifier section first. If no match is found, it continues the search in the group of identifier section. By this order it can be guaranteed that in case where an explicit identifier match is found, the succeeding software can directly proceed on this certain message whereas in case of a group of identifier match the succeeding software needs more steps to identify the message.

18.4 Configuration example 4

Suppose that the five Acceptance Filter address registers contain the values shown in the third column below. In this case each table contains the decimal number of words and entries shown in the next two columns, and the ID Index field of the CANRFS register can return the decimal values shown in the column ID Indexes for CAN messages whose Identifiers match the entries in that table.

Table 256. Example of Acceptance Filter Tables and ID index Values

Table	Register	Value	# Words	# Entire	ID Indexes
Standard Individual	SFF_sa	0x040	810	1610	0-1510
Standard Group	SFF_GRP_sa	0x060	410	410	16-1910
Extended Individual	EFF_sa	0x070	810	1610	20-5510
Extended Group	EFF_GRP_sa	0x100	810	1610	56-5710
	ENDofTable	0x110			

18.5 Configuration example 5

Figure 12–59 below is a more detailed and graphic example of the address registers, table layout, and ID Index values. It shows:

- A Standard Individual table starting at the start of Acceptance Filter RAM and containing 26 Identifiers, followed by:
- A Standard Group table containing 12 ranges of Identifiers, followed by:
- An Extended Individual table containing 3 Identifiers, followed by:
- An Extended Group table containing 2 ranges of Identifiers.

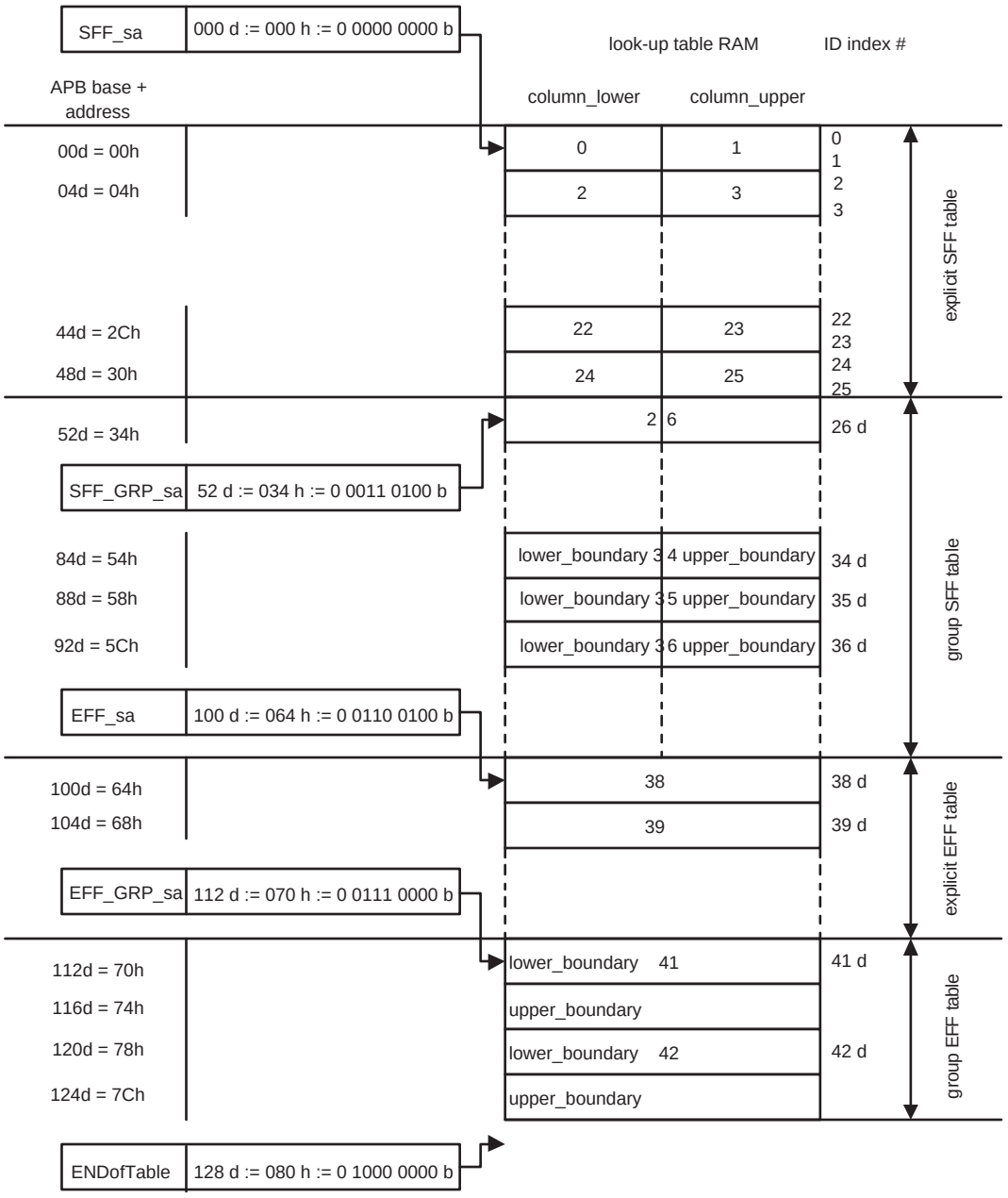

Fig 59. Detailed example of acceptance filter tables and ID index values

18.6 Configuration example 6

The Table below shows which sections and therefore which types of CAN identifiers are used and activated. The ID-Look-up Table configuration of this example is shown in Figure 12–60.

Table 257. Used ID-Look-up Table sections

ID-Look-up Table Section	Status
FullCAN	not activated
Explicit Standard Frame Format	activated
Group of Standard Frame Format	activated
Explicit Extended Frame Format	activated
Group of Extended Frame Format	activated

Explicit standard frame format identifier section (11-bit CAN ID):

The start address of the Explicit Standard Frame Format section is defined in the SFF_sa register with the value of 0x00. The end of this section is defined in the SFF_GRP_sa register. In the Explicit Standard Frame Format section of the ID Look-up Table two CAN Identifiers with their Source CAN Channels (SCC) share one 32-bit word. Not used or disabled CAN Identifiers can be marked by setting the message disable bit.

Group of standard frame format identifier section (11-bit CAN ID):

The start address of the Group of Standard Frame Format section is defined with the SFF_GRP_sa register with the value of 0x10. The end of this section is defined with the EFF_sa register. In the Group of Standard Frame Format section two CAN Identifiers with the same Source CAN Channel (SCC) share one 32-bit word and represent a range of CAN Identifiers to be accepted. Bit 31 down to 16 represents the lower boundary and bit 15 down to 0 represents the upper boundary of the range of CAN Identifiers. All Identifiers within this range (including the boundary identifiers) will be accepted. A whole group can be disabled and not used by the acceptance filter by setting the message disable bit in the upper and lower boundary identifier. To provide memory space for four Groups of Standard Frame Format identifiers, the EFF_sa register value is set to 0x20. The identifier group with the Index 9 of this section is not used and therefore disabled.

Explicit extended frame format identifier section (29-bit CAN ID, Figure 12–60)

The start address of the Explicit Extended Frame Format section is defined with the EFF_sa register with the value of 0x20. The end of this section is defined with the EFF_GRP_sa register. In the explicit Extended Frame Format section only one CAN Identifier with its Source CAN Channel (SCC) is programmed per address line. To provide memory space for four Explicit Extended Frame Format identifiers, the EFF_GRP_sa register value is set to 0x30.

Group of extended frame format identifier section (29-bit CAN ID, Figure 12–60)

The start address of the Group of Extended Frame Format is defined with the EFF_GRP_sa register with the value of 0x30. The end of this section is defined with the End of Table address register (ENDofTable). In the Group of Extended Frame Format section the boundaries are programmed with a pair of address lines; the first is the lower boundary, the second the upper boundary. To provide memory space for two Groups of Extended Frame Format Identifiers, the ENDofTable register value is set to 0x40.

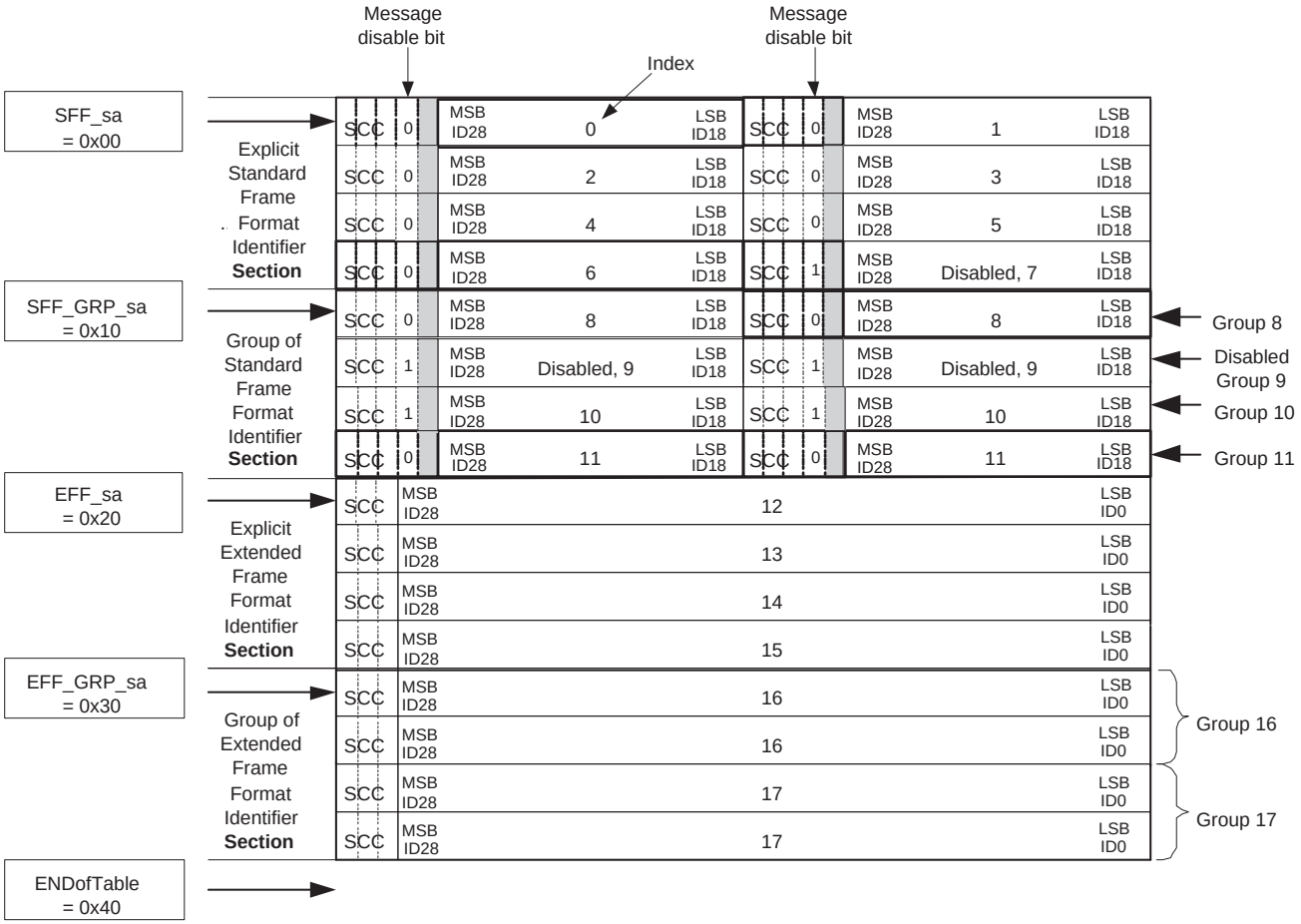

Fig 60. ID Look-up table configuration example (no FullCAN)

18.7 Configuration example 7

The Table below shows which sections and therefore which types of CAN identifiers are used and activated. The ID-Look-up Table configuration of this example is shown in Figure 12–61.

This example uses a typical configuration in which FullCAN as well as Explicit Standard Frame Format messages are defined. As described in Section 12–16.1 “Acceptance filter search algorithm”, acceptance filtering takes place in a certain order. With the enabled FullCAN section, the identifier screening process of the acceptance filter starts always in the FullCAN section first, before it continues with the rest of enabled sections.e disabled.

Table 258. Used ID-Look-up Table sections

ID-Look-up Table Section	Status
FullCAN	activated and enabled
Explicit Standard Frame Format	activated
Group of Standard Frame Format	not activated
Explicit Extended Frame Format	not activated
Group of Extended Frame Format	not activated

FullCAN explicit standard frame format identifier section (11-bit CAN ID)

The start address of the FullCAN Explicit Standard Frame Format Identifier section is (automatically) set to 0x00. The end of this section is defined in the SFF_sa register. In the FullCAN ID section only identifiers of FullCAN Object are stored for acceptance filtering. In this section two CAN Identifiers with their Source CAN Channels (SCC) share one 32-bit word. Not used or disabled CAN Identifiers can be marked by setting the message disable bit. The FullCAN Object data for each defined identifier can be found in the FullCAN Message Object section. In case of an identifier match during the acceptance filter process, the received FullCAN message object data is moved from the Receive Buffer of the appropriate CAN Controller into the FullCAN Message Object section. To provide memory space for eight FullCAN, Explicit Standard Frame Format identifiers, the SFF_sa register value is set to 0x10. The identifier with the Index 1 of this section is not used and therefore disabled.

Explicit standard frame format identifier section (11-bit CAN ID)

The start address of the Explicit Standard Frame Format section is defined in the SFF_sa register with the value of 0x10. The end of this section is defined in the End of Table address register (ENDofTable). In the explicit Standard Frame Format section of the ID Look-up Table two CAN Identifiers with their Source CAN Channel (SCC) share one 32-bit word. Not used or disabled CAN Identifiers can be marked by setting the message disable bit. To provide memory space for eight Explicit Standard Frame Format identifiers, the ENDofTable register value is set to 0x20.

FullCAN message object data section

The start address of the FullCAN Message Object Data section is defined with the ENDofTable register. The number of enabled FullCAN identifiers is limited to the available memory space in the FullCAN Message Object Data section. Each defined FullCAN Message needs three address lines for the Message Data in the FullCAN Message Object Data section. The FullCAN Message Object section is organized in that way, that each Index number of the FullCAN Identifier section corresponds to a Message Object Number in the FullCAN Message Object section.

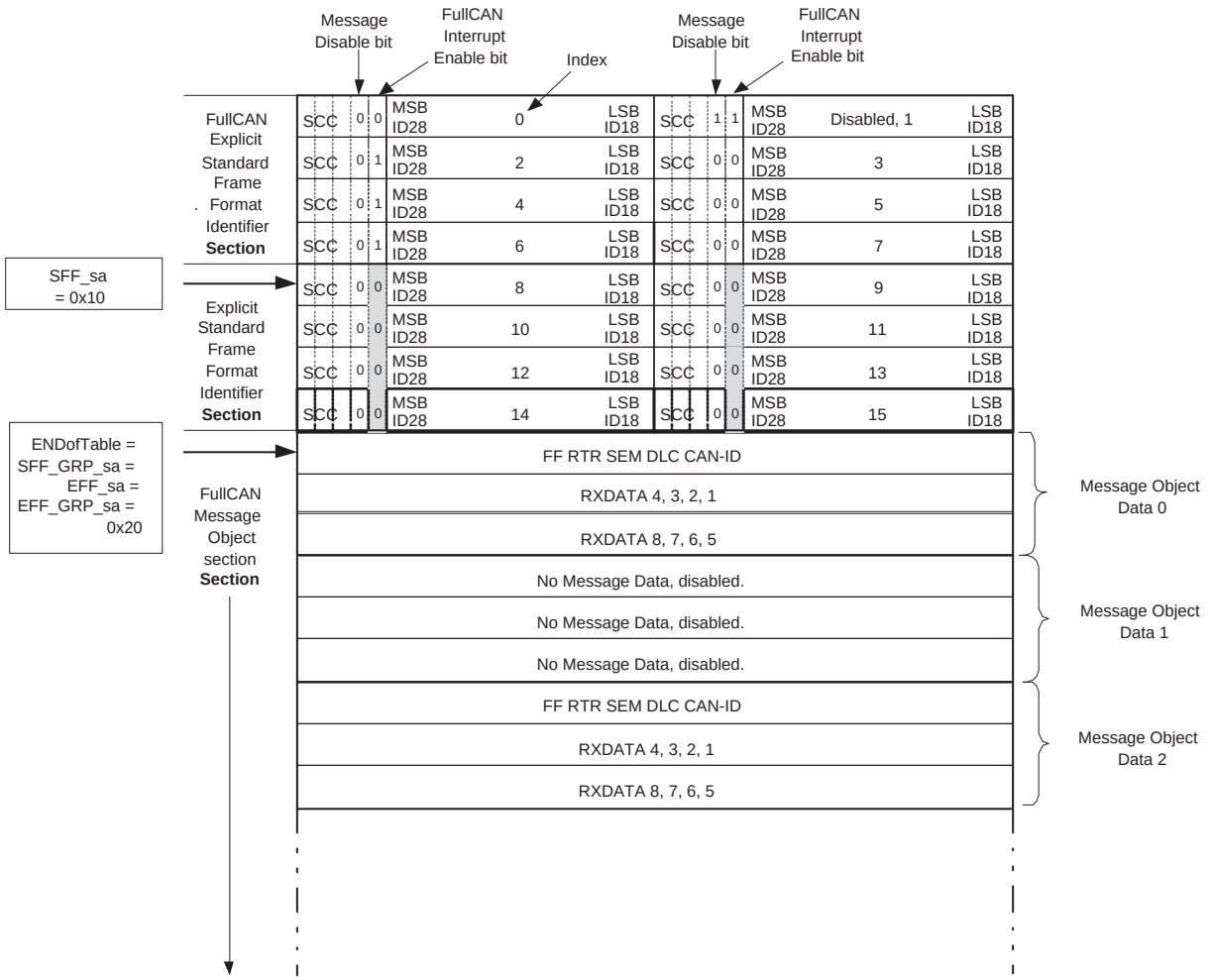

Fig 61. ID Look-up table configuration example (FullCAN activated and enabled)

18.8 Look-up table programming guidelines

All identifier sections of the ID Look-up Table have to be programmed in such a way, that each active section is organized as a sorted list or table with an increasing order of the Source CAN Channel (SCC) together with CAN Identifier in each section.

SCC value equals CAN_controller - 1, i.e., SCC = 0 matches CAN1 and SCC = 1 matches CAN2.

In cases, where a syntax error in the ID Look-up Table is encountered, the Look-up Table address of the incorrect line is made available in the Look-up Table Error Address Register (LUTerrAd).

The reporting process in the Look-up Table Error Address Register (LUTerrAd) is a “run-time” process. Only those address lines with syntax error are reported, which were passed through the acceptance filtering process.

The following general rules for programming the Look-up Table apply:

- Each section has to be organized as a sorted list or table with an increasing order of the Source CAN Channel (SCC) in conjunction with the CAN Identifier (there is no exception for disabled identifiers).
- The upper and lower bound in a Group of Identifiers definition has to be from the same Source CAN Channel.
- To disable a Group of Identifiers the message disable bit has to be set for both, the upper and lower bound.