

# Chapter 1

## Freescle's Scalable Controller Area Network (S12MSCANV2)

### 1.1 Introduction

Freescle's scalable controller area network (S12MSCANV2) definition is based on the MSCAN12 definition, which is the specific implementation of the MSCAN concept targeted for the M68HC12 microcontroller family.

The module is a communication controller implementing the CAN 2.0A/B protocol as defined in the Bosch specification dated September 1991. For users to fully understand the MSCAN specification, it is recommended that the Bosch specification be read first to familiarize the reader with the terms and concepts contained within this document.

Though not exclusively intended for automotive applications, CAN protocol is designed to meet the specific requirements of a vehicle serial data bus: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth.

MSCAN uses an advanced buffer arrangement resulting in predictable real-time behavior and simplified application software.

#### 1.1.1 Glossary

ACK: Acknowledge of CAN message

CAN: Controller Area Network

CRC: Cyclic Redundancy Code

EOF: End of Frame

FIFO: First-In-First-Out Memory

IFS: Inter-Frame Sequence

SOF: Start of Frame

CPU bus: CPU related read/write data bus

CAN bus: CAN protocol related serial bus

oscillator clock: Direct clock from external oscillator

bus clock: CPU bus related clock

CAN clock: CAN protocol related clock

## 1.1.2 Block Diagram

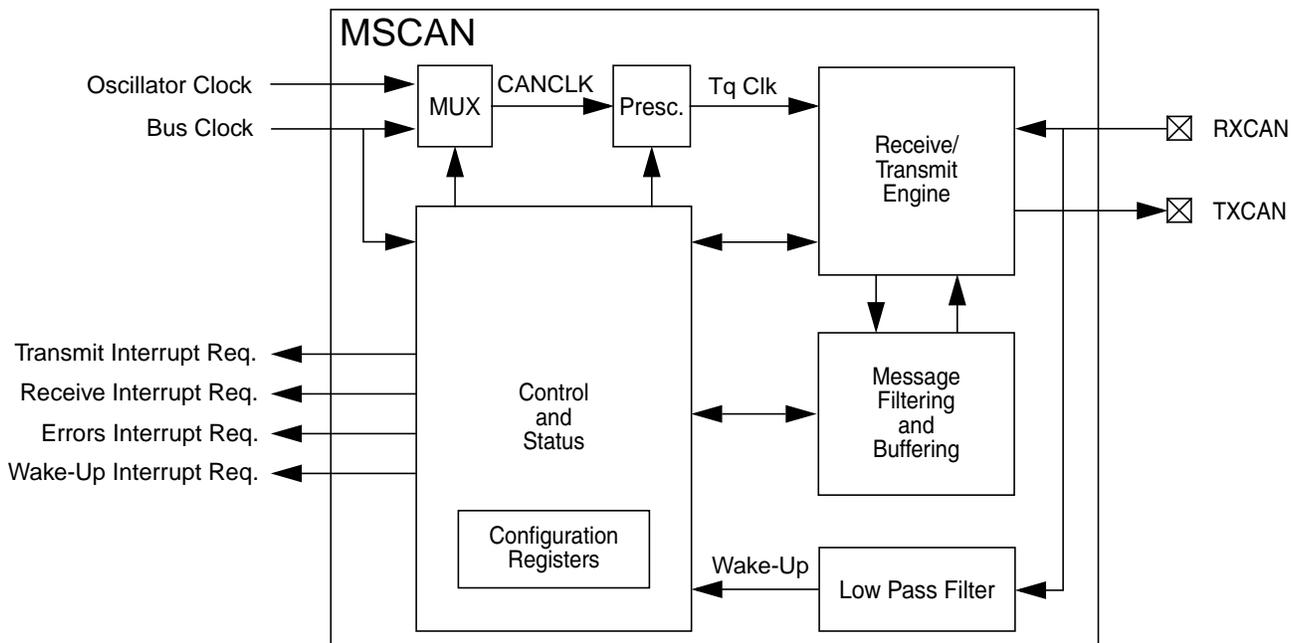


Figure 1-1. MSCAN Block Diagram

## 1.1.3 Features

The basic features of the MSCAN are as follows:

- Implementation of the CAN protocol — Version 2.0A/B
  - Standard and extended data frames
  - Zero to eight bytes data length
  - Programmable bit rate up to 1 Mbps<sup>1</sup>
  - Support for remote frames
- Five receive buffers with FIFO storage scheme
- Three transmit buffers with internal prioritization using a “local priority” concept
- Flexible maskable identifier filter supports two full-size (32-bit) extended identifier filters, or four 16-bit filters, or eight 8-bit filters
- Programmable wakeup functionality with integrated low-pass filter
- Programmable loopback mode supports self-test operation
- Programmable listen-only mode for monitoring of CAN bus
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (warning, error passive, bus-off)
- Programmable MSCAN clock source either bus clock or oscillator clock
- Internal timer for time-stamping of received and transmitted messages
- Three low-power modes: sleep, power down, and MSCAN enable

1. Depending on the actual bit timing and the clock jitter of the PLL.

- Global initialization of configuration registers

### 1.1.4 Modes of Operation

The following modes of operation are specific to the MSCAN. See [Section 1.4, “Functional Description,”](#) for details.

- Listen-Only Mode
- MSCAN Sleep Mode
- MSCAN Initialization Mode
- MSCAN Power Down Mode

## 1.2 External Signal Description

The MSCAN uses two external pins:

### 1.2.1 RXCAN — CAN Receiver Input Pin

RXCAN is the MSCAN receiver input pin.

### 1.2.2 TXCAN — CAN Transmitter Output Pin

TXCAN is the MSCAN transmitter output pin. The TXCAN output pin represents the logic level on the CAN bus:

- 0 = Dominant state
- 1 = Recessive state

### 1.2.3 CAN System

A typical CAN system with MSCAN is shown in [Figure 1-2](#). Each CAN station is connected physically to the CAN bus lines through a transceiver device. The transceiver is capable of driving the large current needed for the CAN bus and has current protection against defective CAN or defective stations.

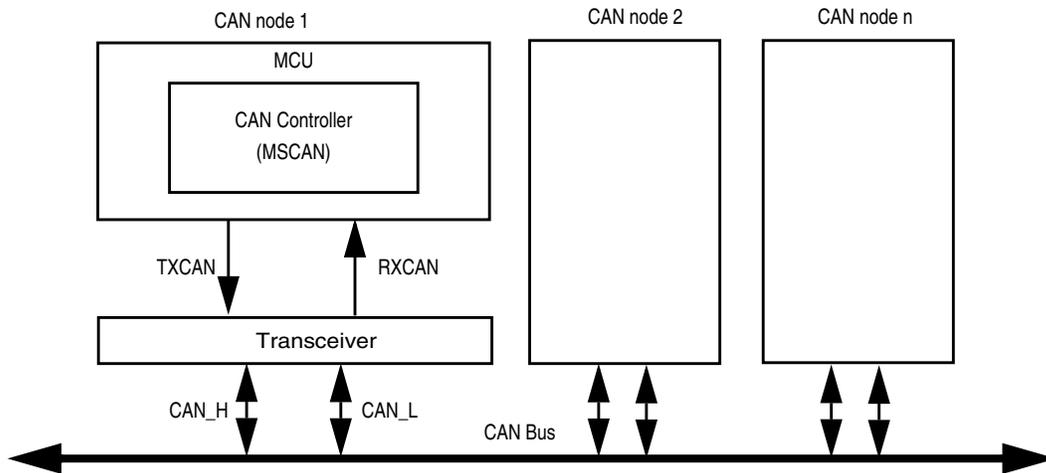


Figure 1-2. CAN System

### 1.3 Memory Map and Register Definition

This section provides a detailed description of all registers accessible in the MSCAN.

#### 1.3.1 Module Memory Map

Figure 1-3 gives an overview on all registers and their individual bits in the MSCAN memory map. The *register address* results from the addition of *base address* and *address offset*. The *base address* is determined at the MCU level and can be found in the MCU memory map description. The *address offset* is defined at the module level.

The MSCAN occupies 64 bytes in the memory space. The base address of the MSCAN module is determined at the MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset.

The detailed register descriptions follow in the order they appear in the register map.

Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0000 CANCTL0	R W RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
0x0001 CANCTL1	R W CANE	CLKSRC	LOOPB	LISTEN		WUPM	SLPAK	INITAK

= Unimplemented or Reserved
 u = Unaffected

Figure 1-3. MSCAN Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0002 CANBTR0	R W	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1 BRP0
0x0003 CANBTR1	R W	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11 TSEG10
0x0004 CANRFLG	R W	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF RXF
0x0005 CANRIER	R W	WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE RXFIE
0x0006 CANTFLG	R W	0	0	0	0	0	TXE2	TXE1 TXE0
0x0007 CANTIER	R W	0	0	0	0	0	TXEIE2	TXEIE1 TXEIE0
0x0008 CANTARQ	R W	0	0	0	0	0	ABTRQ2	ABTRQ1 ABTRQ0
0x0009 CANTAACK	R W	0	0	0	0	0	ABTAK2	ABTAK1 ABTAK0
0x000A CANTBSEL	R W	0	0	0	0	0	TX2	TX1 TX0
0x000B CANIDAC	R W	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1 IDHIT0
0x000C–0x000D Reserved	R W	0	0	0	0	0	0	0 0
0x000E CANRXERR	R W	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1 RXERR0
0x000F CANTXERR	R W	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1 TXERR0
0x0010–0x0013 CANIDAR0–3	R W	AC7	AC6	AC5	AC4	AC3	AC2	AC1 AC0
0x0014–0x0017 CANIDMRx	R W	AM7	AM6	AM5	AM4	AM3	AM2	AM1 AM0

= Unimplemented or Reserved
 u = Unaffected

Figure 1-3. MSCAN Register Summary (continued)

Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0018–0x001B CANIDAR4–7	R W AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
0x001C–0x001F CANIDMR4–7	R W AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
0x0020–0x002F CANRXFG	R W See Section 1.3.3, "Programmer's Model of Message Storage"							
0x0030–0x003F CANTXFG	R W See Section 1.3.3, "Programmer's Model of Message Storage"							

= Unimplemented or Reserved
 u = Unaffected

Figure 1-3. MSCAN Register Summary (continued)

### 1.3.2 Register Descriptions

This section describes in detail all the registers and register bits in the MSCAN module. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order. All bits of all registers in this module are completely synchronous to internal clocks during a register read.

#### 1.3.2.1 MSCAN Control Register 0 (CANCTL0)

The CANCTL0 register provides various control bits of the MSCAN module as described below.

Module Base + 0x0000

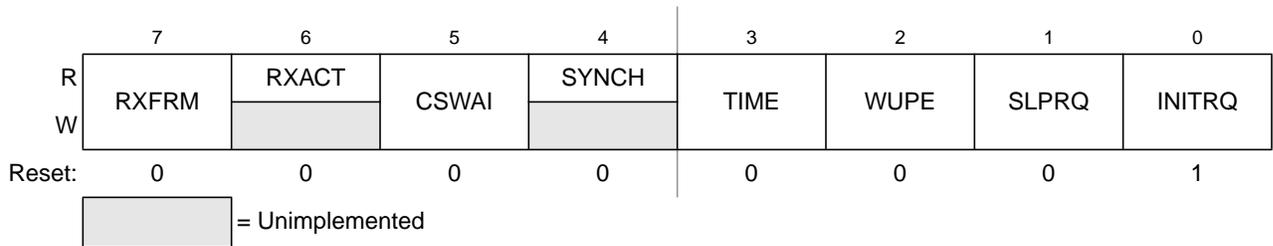


Figure 1-4. MSCAN Control Register 0 (CANCTL0)

#### NOTE

The CANCTL0 register, except WUPE, INITRQ, and SLPRQ, is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

Read: Anytime

Write: Anytime when out of initialization mode; exceptions are read-only RXACT and SYNCH, RXFRM (which is set by the module only), and INITRQ (which is also writable in initialization mode).

Table 1-1. CANCTL0 Register Field Descriptions

Field	Description
7 RXFRM <sup>1</sup>	<b>Received Frame Flag</b> — This bit is read and clear only. It is set when a receiver has received a valid message correctly, independently of the filter configuration. After it is set, it remains set until cleared by software or reset. Clearing is done by writing a 1. Writing a 0 is ignored. This bit is not valid in loopback mode. 0 No valid message was received since last clearing this flag 1 A valid message was received since last clearing of this flag
6 RXACT	<b>Receiver Active Status</b> — This read-only flag indicates the MSCAN is receiving a message. The flag is controlled by the receiver front end. This bit is not valid in loopback mode. 0 MSCAN is transmitting or idle <sup>2</sup> 1 MSCAN is receiving a message (including when arbitration is lost) <sup>2</sup>
5 CSWAI <sup>3</sup>	<b>CAN Stops in Wait Mode</b> — Enabling this bit allows for lower power consumption in wait mode by disabling all the clocks at the CPU bus interface to the MSCAN module. 0 The module is not affected during wait mode 1 The module ceases to be clocked during wait mode
4 SYNCH	<b>Synchronized Status</b> — This read-only flag indicates whether the MSCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the MSCAN. 0 MSCAN is not synchronized to the CAN bus 1 MSCAN is synchronized to the CAN bus
3 TIME	<b>Timer Enable</b> — This bit activates an internal 16-bit wide free running timer which is clocked by the bit clock rate. If the timer is enabled, a 16-bit time stamp will be assigned to each transmitted/received message within the active TX/RX buffer. Right after the EOF of a valid message on the CAN bus, the time stamp is written to the highest bytes (0x000E, 0x000F) in the appropriate buffer (see <a href="#">Section 1.3.3, "Programmer's Model of Message Storage"</a> ). The internal timer is reset (all bits set to 0) when disabled. This bit is held low in initialization mode. 0 Disable internal MSCAN timer 1 Enable internal MSCAN timer
2 WUPE <sup>4</sup>	<b>Wake-Up Enable</b> — This configuration bit allows the MSCAN to restart from sleep mode when traffic on CAN is detected (see <a href="#">Section 1.4.5.4, "MSCAN Sleep Mode"</a> ). 0 Wake-up disabled — The MSCAN ignores traffic on CAN 1 Wake-up enabled — The MSCAN is able to restart

Table 1-1. CANCTL0 Register Field Descriptions (continued)

Field	Description
1 SLPRQ <sup>5</sup>	<p><b>Sleep Mode Request</b> — This bit requests the MSCAN to enter sleep mode, which is an internal power saving mode (see Section 1.4.5.4, “MSCAN Sleep Mode”). The sleep mode request is serviced when the CAN bus is idle, i.e., the module is not receiving a message and all transmit buffers are empty. The module indicates entry to sleep mode by setting SLPK = 1 (see Section 1.3.2.2, “MSCAN Control Register 1 (CANCTL1)”). SLPRQ cannot be set while the WUPIF flag is set (see Section 1.3.2.5, “MSCAN Receiver Flag Register (CANRFLG)”). Sleep mode will be active until SLPRQ is cleared by the CPU or, depending on the setting of WUPE, the MSCAN detects activity on the CAN bus and clears SLPRQ itself.</p> <p>0 Running — The MSCAN functions normally 1 Sleep mode request — The MSCAN enters sleep mode when CAN bus idle</p>
0 INITRQ <sup>6,7</sup>	<p><b>Initialization Mode Request</b> — When this bit is set by the CPU, the MSCAN skips to initialization mode (see Section 1.4.5.5, “MSCAN Initialization Mode”). Any ongoing transmission or reception is aborted and synchronization to the CAN bus is lost. The module indicates entry to initialization mode by setting INITAK = 1 (Section 1.3.2.2, “MSCAN Control Register 1 (CANCTL1)”).</p> <p>The following registers enter their hard reset state and restore their default values: CANCTL0<sup>8</sup>, CANRFLG<sup>9</sup>, CANRIER<sup>10</sup>, CANTFLG, CANTIER, CANTARQ, CANTAACK, and CANTBSEL.</p> <p>The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0-7, and CANIDMR0-7 can only be written by the CPU when the MSCAN is in initialization mode (INITRQ = 1 and INITAK = 1). The values of the error counters are not affected by initialization mode.</p> <p>When this bit is cleared by the CPU, the MSCAN restarts and then tries to synchronize to the CAN bus. If the MSCAN is not in bus-off state, it synchronizes after 11 consecutive recessive bits on the CAN bus; if the MSCAN is in bus-off state, it continues to wait for 128 occurrences of 11 consecutive recessive bits.</p> <p>Writing to other bits in CANCTL0, CANRFLG, CANRIER, CANTFLG, or CANTIER must be done only after initialization mode is exited, which is INITRQ = 0 and INITAK = 0.</p> <p>0 Normal operation 1 MSCAN in initialization mode</p>

<sup>1</sup> The MSCAN must be in normal mode for this bit to become set.

<sup>2</sup> See the Bosch CAN 2.0A/B specification for a detailed definition of transmitter and receiver states.

<sup>3</sup> In order to protect from accidentally violating the CAN protocol, the TXCAN pin is immediately forced to a recessive state when the CPU enters wait (CSWAI = 1) or stop mode (see Section 1.4.5.2, “Operation in Wait Mode” and Section 1.4.5.3, “Operation in Stop Mode”).

<sup>4</sup> The CPU has to make sure that the WUPE register and the WUPIE wake-up interrupt enable register (see Section 1.3.2.6, “MSCAN Receiver Interrupt Enable Register (CANRIER)”) is enabled, if the recovery mechanism from stop or wait is required.

<sup>5</sup> The CPU cannot clear SLPRQ before the MSCAN has entered sleep mode (SLPRQ = 1 and SLPK = 1).

<sup>6</sup> The CPU cannot clear INITRQ before the MSCAN has entered initialization mode (INITRQ = 1 and INITAK = 1).

<sup>7</sup> In order to protect from accidentally violating the CAN protocol, the TXCAN pin is immediately forced to a recessive state when the initialization mode is requested by the CPU. Thus, the recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPK = 1) before requesting initialization mode.

<sup>8</sup> Not including WUPE, INITRQ, and SLPRQ.

<sup>9</sup> TSTAT1 and TSTAT0 are not affected by initialization mode.

<sup>10</sup> RSTAT1 and RSTAT0 are not affected by initialization mode.

### 1.3.2.2 MSCAN Control Register 1 (CANCTL1)

The CANCTL1 register provides various control bits and handshake status information of the MSCAN module as described below.

Module Base + 0x0001

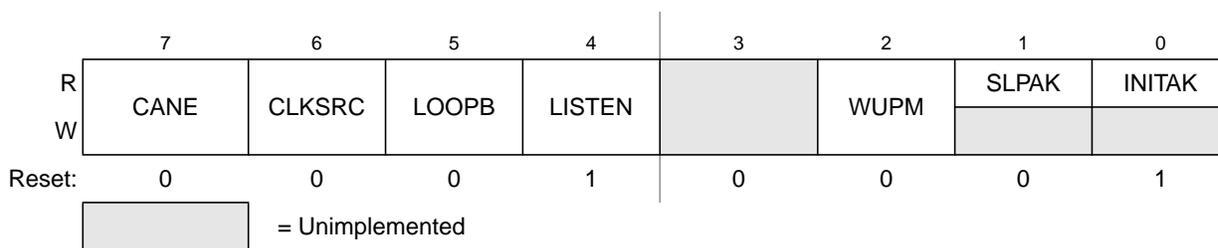


Figure 1-5. MSCAN Control Register 1 (CANCTL1)

Read: Anytime

Write: Anytime when INITRQ = 1 and INITAK = 1, except CANE which is write once in normal and anytime in special system operation modes when the MSCAN is in initialization mode (INITRQ = 1 and INITAK = 1).

Table 1-2. CANCTL1 Register Field Descriptions

Field	Description
7 CANE	<b>MSCAN Enable</b> 0 MSCAN module is disabled 1 MSCAN module is enabled
6 CLKSRC	<b>MSCAN Clock Source</b> — This bit defines the clock source for the MSCAN module (only for systems with a clock generation module; <a href="#">Section 1.4.3.2, “Clock System,”</a> and <a href="#">Section Figure 1-44., “MSCAN Clocking Scheme,”</a> ). 0 MSCAN clock source is the oscillator clock 1 MSCAN clock source is the bus clock
5 LOOPB	<b>Loopback Self Test Mode</b> — When this bit is set, the MSCAN performs an internal loopback which can be used for self test operation. The bit stream output of the transmitter is fed back to the receiver internally. The RXCAN input pin is ignored and the TXCAN output goes to the recessive state (logic 1). The MSCAN behaves as it does normally when transmitting and treats its own transmitted message as a message received from a remote node. In this state, the MSCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated. 0 Loopback self test disabled 1 Loopback self test enabled
4 LISTEN	<b>Listen Only Mode</b> — This bit configures the MSCAN as a CAN bus monitor. When LISTEN is set, all valid CAN messages with matching ID are received, but no acknowledgement or error frames are sent out (see <a href="#">Section 1.4.4.4, “Listen-Only Mode”</a> ). In addition, the error counters are frozen. Listen only mode supports applications which require “hot plugging” or throughput analysis. The MSCAN is unable to transmit any messages when listen only mode is active. 0 Normal operation 1 Listen only mode activated
2 WUPM	<b>Wake-Up Mode</b> — If WUPE in CANCTL0 is enabled, this bit defines whether the integrated low-pass filter is applied to protect the MSCAN from spurious wake-up (see <a href="#">Section 1.4.5.4, “MSCAN Sleep Mode”</a> ). 0 MSCAN wakes up on any dominant level on the CAN bus 1 MSCAN wakes up only in case of a dominant pulse on the CAN bus that has a length of $T_{wup}$

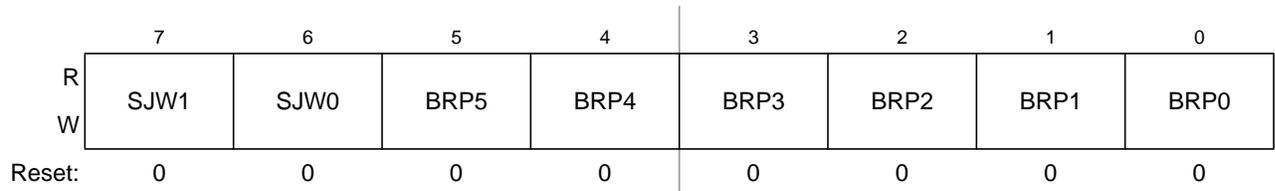
**Table 1-2. CANCTL1 Register Field Descriptions (continued)**

Field	Description
1 SLPAK	<b>Sleep Mode Acknowledge</b> — This flag indicates whether the MSCAN module has entered sleep mode (see Section 1.4.5.4, "MSCAN Sleep Mode"). It is used as a handshake flag for the SLPRQ sleep mode request. Sleep mode is active when SLPRQ = 1 and SLPAK = 1. Depending on the setting of WUPE, the MSCAN will clear the flag if it detects activity on the CAN bus while in sleep mode. 0 Running — The MSCAN operates normally 1 Sleep mode active — The MSCAN has entered sleep mode
0 INITAK	<b>Initialization Mode Acknowledge</b> — This flag indicates whether the MSCAN module is in initialization mode (see Section 1.4.5.5, "MSCAN Initialization Mode"). It is used as a handshake flag for the INITRQ initialization mode request. Initialization mode is active when INITRQ = 1 and INITAK = 1. The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0–CANIDAR7, and CANIDMR0–CANIDMR7 can be written only by the CPU when the MSCAN is in initialization mode. 0 Running — The MSCAN operates normally 1 Initialization mode active — The MSCAN has entered initialization mode

### 1.3.2.3 MSCAN Bus Timing Register 0 (CANBTR0)

The CANBTR0 register configures various CAN bus timing parameters of the MSCAN module.

Module Base + 0x0002



**Figure 1-6. MSCAN Bus Timing Register 0 (CANBTR0)**

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 1-3. CANBTR0 Register Field Descriptions**

Field	Description
7:6 SJW[1:0]	<b>Synchronization Jump Width</b> — The synchronization jump width defines the maximum number of time quanta (Tq) clock cycles a bit can be shortened or lengthened to achieve resynchronization to data transitions on the CAN bus (see Table 1-5).
5:0 BRP[5:0]	<b>Baud Rate Prescaler</b> — These bits determine the time quanta (Tq) clock which is used to build up the bit timing (see Table 1-6).

**Table 1-4. Synchronization Jump Width**

SJW1	SJW0	Synchronization Jump Width
0	0	1 Tq clock cycle
0	1	2 Tq clock cycles
1	0	3 Tq clock cycles
1	1	4 Tq clock cycles

Table 1-5. Baud Rate Prescaler

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Prescaler value (P)
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
:	:	:	:	:	:	:
1	1	1	1	1	1	64

### 1.3.2.4 MSCAN Bus Timing Register 1 (CANBTR1)

The CANBTR1 register configures various CAN bus timing parameters of the MSCAN module.

Module Base + 0x0003

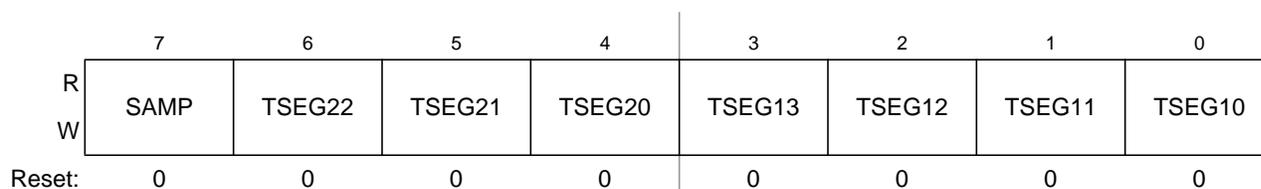


Figure 1-7. MSCAN Bus Timing Register 1 (CANBTR1)

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

Table 1-6. CANBTR1 Register Field Descriptions

Field	Description
7 SAMP	<b>Sampling</b> — This bit determines the number of CAN bus samples taken per bit time. 0 One sample per bit. 1 Three samples per bit <sup>1</sup> . If SAMP = 0, the resulting bit value is equal to the value of the single bit positioned at the sample point. If SAMP = 1, the resulting bit value is determined by using majority rule on the three total samples. For higher bit rates, it is recommended that only one sample is taken per bit time (SAMP = 0).
6:4 TSEG2[2:0]	<b>Time Segment 2</b> — Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point (see Figure 1-45). Time segment 2 (TSEG2) values are programmable as shown in Table 1-8.
3:0 TSEG1[3:0]	<b>Time Segment 1</b> — Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point (see Figure 1-45). Time segment 1 (TSEG1) values are programmable as shown in Table 1-9.

<sup>1</sup> In this case, PHASE\_SEG1 must be at least 2 time quanta (Tq).

**Table 1-7. Time Segment 2 Values**

TSEG22	TSEG21	TSEG20	Time Segment 2
0	0	0	1 Tq clock cycle <sup>1</sup>
0	0	1	2 Tq clock cycles
:	:	:	:
1	1	0	7 Tq clock cycles
1	1	1	8 Tq clock cycles

<sup>1</sup> This setting is not valid. Please refer to [Table 1-36](#) for valid settings.

**Table 1-8. Time Segment 1 Values**

TSEG13	TSEG12	TSEG11	TSEG10	Time segment 1
0	0	0	0	1 Tq clock cycle <sup>1</sup>
0	0	0	1	2 Tq clock cycles <sup>1</sup>
0	0	1	0	3 Tq clock cycles <sup>1</sup>
0	0	1	1	4 Tq clock cycles
:	:	:	:	:
1	1	1	0	15 Tq clock cycles
1	1	1	1	16 Tq clock cycles

<sup>1</sup> This setting is not valid. Please refer to [Table 1-36](#) for valid settings.

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (Tq) clock cycles per bit (as shown in [Table 1-8](#) and [Table 1-9](#)).

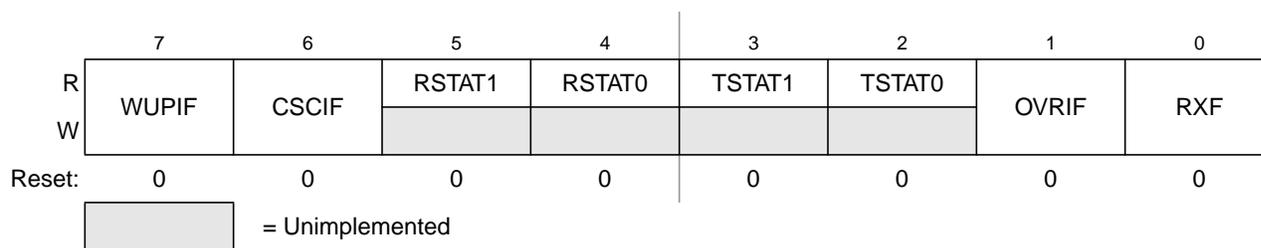
*Eqn. 1-1*

$$\text{Bit Time} = \frac{(\text{Prescaler value})}{f_{\text{CANCLK}}} \cdot (1 + \text{TimeSegment1} + \text{TimeSegment2})$$

### 1.3.2.5 MSCAN Receiver Flag Register (CANRFLG)

A flag can be cleared only by software (writing a 1 to the corresponding bit position) when the condition which caused the setting is no longer valid. Every flag has an associated interrupt enable bit in the CANRIER register.

Module Base + 0x0004


**Figure 1-8. MSCAN Receiver Flag Register (CANRFLG)**
**NOTE**

The CANRFLG register is held in the reset state<sup>1</sup> when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

Read: Anytime

Write: Anytime when out of initialization mode, except RSTAT[1:0] and TSTAT[1:0] flags which are read-only; write of 1 clears flag; write of 0 is ignored.

**Table 1-9. CANRFLG Register Field Descriptions**

Field	Description
7 WUPIF	<b>Wake-Up Interrupt Flag</b> — If the MSCAN detects CAN bus activity while in sleep mode (see Section 1.4.5.4, “MSCAN Sleep Mode,”) and WUPE = 1 in CANTCTL0 (see Section 1.3.2.1, “MSCAN Control Register 0 (CANCTL0)”), the module will set WUPIF. If not masked, a wake-up interrupt is pending while this flag is set. 0 No wake-up activity observed while in sleep mode 1 MSCAN detected activity on the CAN bus and requested wake-up
6 CSCIF	<b>CAN Status Change Interrupt Flag</b> — This flag is set when the MSCAN changes its current CAN bus status due to the actual value of the transmit error counter (TEC) and the receive error counter (REC). An additional 4-bit (RSTAT[1:0], TSTAT[1:0]) status register, which is split into separate sections for TEC/REC, informs the system on the actual CAN bus status (see Section 1.3.2.6, “MSCAN Receiver Interrupt Enable Register (CANRIER)”). If not masked, an error interrupt is pending while this flag is set. CSCIF provides a blocking interrupt. That guarantees that the receiver/transmitter status bits (RSTAT/TSTAT) are only updated when no CAN status change interrupt is pending. If the TECs/RECs change their current value after the CSCIF is asserted, which would cause an additional state change in the RSTAT/TSTAT bits, these bits keep their status until the current CSCIF interrupt is cleared again. 0 No change in CAN bus status occurred since last interrupt 1 MSCAN changed current CAN bus status
5:4 RSTAT[1:0]	<b>Receiver Status Bits</b> — The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate receiver related CAN bus status of the MSCAN. The coding for the bits RSTAT1, RSTAT0 is: 00 RxOK: 0 ≤ receive error counter ≤ 96 01 RxWRN: 96 < receive error counter ≤ 127 10 RxERR: 127 < receive error counter 11 Bus-off <sup>1</sup> : transmit error counter > 255

1. The RSTAT[1:0], TSTAT[1:0] bits are not affected by initialization mode.

**Table 1-9. CANRFLG Register Field Descriptions (continued)**

Field	Description
3:2 TSTAT[1:0]	<b>Transmitter Status Bits</b> — The values of the error counters control the actual CAN bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set, these bits indicate the appropriate transmitter related CAN bus status of the MSCAN. The coding for the bits TSTAT1, TSTAT0 is: 00 TxOK: 0 ≤ transmit error counter ≤ 96 01 TxWRN: 96 < transmit error counter ≤ 127 10 TxERR: 127 < transmit error counter ≤ 255 11 Bus-Off: transmit error counter > 255
1 OVRIF	<b>Overrun Interrupt Flag</b> — This flag is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this flag is set. 0 No data overrun condition 1 A data overrun detected
0 RXF <sup>2</sup>	<b>Receive Buffer Full Flag</b> — RXF is set by the MSCAN when a new message is shifted in the receiver FIFO. This flag indicates whether the shifted buffer is loaded with a correctly received message (matching identifier, matching cyclic redundancy code (CRC) and no other errors detected). After the CPU has read that message from the RxFG buffer in the receiver FIFO, the RXF flag must be cleared to release the buffer. A set RXF flag prohibits the shifting of the next FIFO entry into the foreground buffer (RxFG). If not masked, a receive interrupt is pending while this flag is set. 0 No new message available within the RxFG 1 The receiver FIFO is not empty. A new message is available in the RxFG

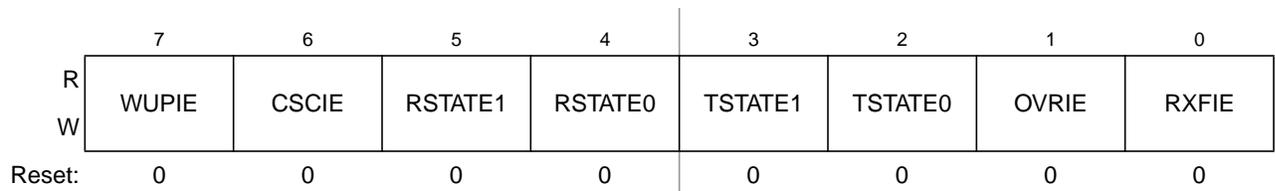
<sup>1</sup> Redundant Information for the most critical CAN bus status which is “bus-off”. This only occurs if the Tx error counter exceeds a number of 255 errors. Bus-off affects the receiver state. As soon as the transmitter leaves its bus-off state the receiver state skips to RxOK too. Refer also to TSTAT[1:0] coding in this register.

<sup>2</sup> To ensure data integrity, do not read the receive buffer registers while the RXF flag is cleared. For MCUs with dual CPUs, reading the receive buffer registers while the RXF flag is cleared may result in a CPU fault condition.

### 1.3.2.6 MSCAN Receiver Interrupt Enable Register (CANRIER)

This register contains the interrupt enable bits for the interrupt flags described in the CANRFLG register.

Module Base + 0x0005



**Figure 1-9. MSCAN Receiver Interrupt Enable Register (CANRIER)**

#### NOTE

The CANRIER register is held in the reset state when the initialization mode is active (INITRQ=1 and INITAK=1). This register is writable when not in initialization mode (INITRQ=0 and INITAK=0).

The RSTATE[1:0], TSTATE[1:0] bits are not affected by initialization mode.

Read: Anytime

Write: Anytime when not in initialization mode

Table 1-10. CANRIER Register Field Descriptions

Field	Description
7 WUPIE <sup>1</sup>	<b>Wake-Up Interrupt Enable</b> 0 No interrupt request is generated from this event. 1 A wake-up event causes a Wake-Up interrupt request.
6 CSCIE	<b>CAN Status Change Interrupt Enable</b> 0 No interrupt request is generated from this event. 1 A CAN Status Change event causes an error interrupt request.
5:4 RSTATE[1:0]	<b>Receiver Status Change Enable</b> — These RSTAT enable bits control the sensitivity level in which receiver state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level the RSTAT flags continue to indicate the actual receiver state and are only updated if no CSCIF interrupt is pending. 00 Do not generate any CSCIF interrupt caused by receiver state changes. 01 Generate CSCIF interrupt only if the receiver enters or leaves “bus-off” state. Discard other receiver state changes for generating CSCIF interrupt. 10 Generate CSCIF interrupt only if the receiver enters or leaves “RxErr” or “bus-off” <sup>2</sup> state. Discard other receiver state changes for generating CSCIF interrupt. 11 Generate CSCIF interrupt on all state changes.
3:2 TSTATE[1:0]	<b>Transmitter Status Change Enable</b> — These TSTAT enable bits control the sensitivity level in which transmitter state changes are causing CSCIF interrupts. Independent of the chosen sensitivity level, the TSTAT flags continue to indicate the actual transmitter state and are only updated if no CSCIF interrupt is pending. 00 Do not generate any CSCIF interrupt caused by transmitter state changes. 01 Generate CSCIF interrupt only if the transmitter enters or leaves “bus-off” state. Discard other transmitter state changes for generating CSCIF interrupt. 10 Generate CSCIF interrupt only if the transmitter enters or leaves “TxErr” or “bus-off” state. Discard other transmitter state changes for generating CSCIF interrupt. 11 Generate CSCIF interrupt on all state changes.
1 OVRIE	<b>Overrun Interrupt Enable</b> 0 No interrupt request is generated from this event. 1 An overrun event causes an error interrupt request.
0 RXFIE	<b>Receiver Full Interrupt Enable</b> 0 No interrupt request is generated from this event. 1 A receive buffer full (successful message reception) event causes a receiver interrupt request.

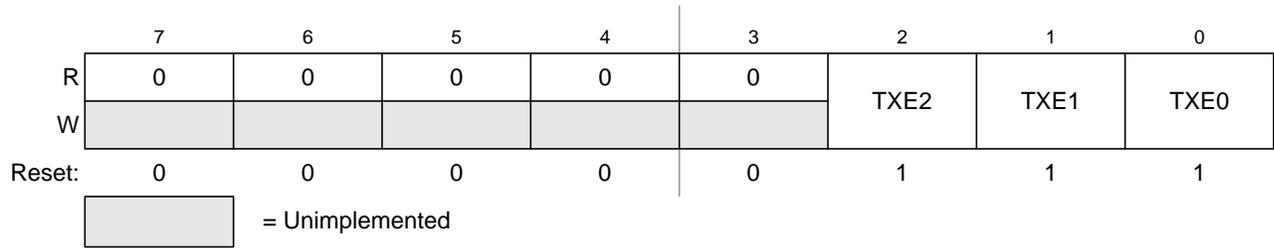
<sup>1</sup> WUPIE and WUPE (see Section 1.3.2.1, “MSCAN Control Register 0 (CANCTL0)”) must both be enabled if the recovery mechanism from stop or wait is required.

<sup>2</sup> Bus-off state is defined by the CAN standard (see Bosch CAN 2.0A/B protocol specification: for only transmitters. Because the only possible state change for the transmitter from bus-off to TxOK also forces the receiver to skip its current state to RxOK, the coding of the RXSTAT[1:0] flags define an additional bus-off state for the receiver (see Section 1.3.2.5, “MSCAN Receiver Flag Register (CANRFLG)”).

### 1.3.2.7 MSCAN Transmitter Flag Register (CANTFLG)

The transmit buffer empty flags each have an associated interrupt enable bit in the CANTIER register.

Module Base + 0x0006



**Figure 1-10. MSCAN Transmitter Flag Register (CANTFLG)**

**NOTE**

The CANTFLG register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

Read: Anytime

Write: Anytime for TXEx flags when not in initialization mode; write of 1 clears flag, write of 0 is ignored

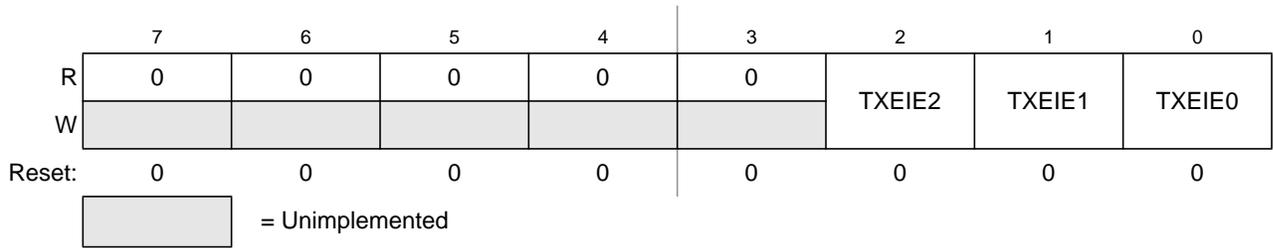
**Table 1-11. CANTFLG Register Field Descriptions**

Field	Description
2:0 TXE[2:0]	<p><b>Transmitter Buffer Empty</b> — This flag indicates that the associated transmit message buffer is empty, and thus not scheduled for transmission. The CPU must clear the flag after a message is set up in the transmit buffer and is due for transmission. The MSCAN sets the flag after the message is sent successfully. The flag is also set by the MSCAN when the transmission request is successfully aborted due to a pending abort request (see Section 1.3.2.9, “MSCAN Transmitter Message Abort Request Register (CANTARQ)”). If not masked, a transmit interrupt is pending while this flag is set.</p> <p>Clearing a TXEx flag also clears the corresponding ABTAkx (see Section 1.3.2.10, “MSCAN Transmitter Message Abort Acknowledge Register (CANTAACK)”). When a TXEx flag is set, the corresponding ABTRQx bit is cleared (see Section 1.3.2.9, “MSCAN Transmitter Message Abort Request Register (CANTARQ)”).</p> <p>When listen-mode is active (see Section 1.3.2.2, “MSCAN Control Register 1 (CANCTL1)”) the TXEx flags cannot be cleared and no transmission is started.</p> <p>Read and write accesses to the transmit buffer will be blocked, if the corresponding TXEx bit is cleared (TXEx = 0) and the buffer is scheduled for transmission.</p> <p>0 The associated message buffer is full (loaded with a message due for transmission)</p> <p>1 The associated message buffer is empty (not scheduled)</p>

**1.3.2.8 MSCAN Transmitter Interrupt Enable Register (CANTIER)**

This register contains the interrupt enable bits for the transmit buffer empty interrupt flags.

Module Base + 0x0007



**Figure 1-11. MSCAN Transmitter Interrupt Enable Register (CANTIER)**

**NOTE**

The CANTIER register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

Read: Anytime

Write: Anytime when not in initialization mode

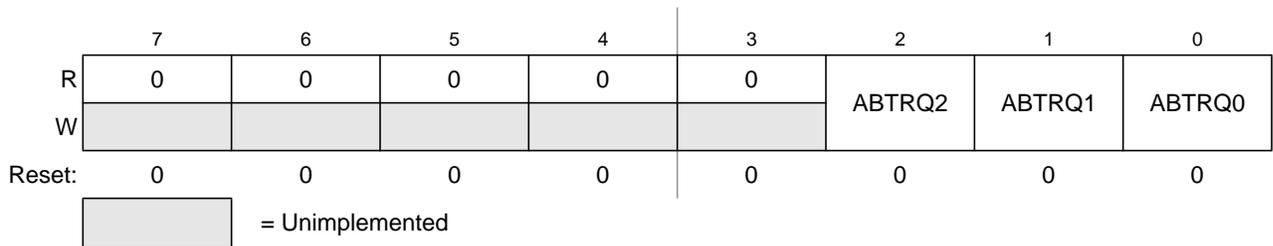
**Table 1-12. CANTIER Register Field Descriptions**

Field	Description
2:0 TXEIE[2:0]	<p><b>Transmitter Empty Interrupt Enable</b></p> <p>0 No interrupt request is generated from this event.</p> <p>1 A transmitter empty (transmit buffer available for transmission) event causes a transmitter empty interrupt request.</p>

**1.3.2.9 MSCAN Transmitter Message Abort Request Register (CANTARQ)**

The CANTARQ register allows abort request of queued messages as described below.

Module Base + 0x0008



**Figure 1-12. MSCAN Transmitter Message Abort Request Register (CANTARQ)**

**NOTE**

The CANTARQ register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

Read: Anytime

Write: Anytime when not in initialization mode

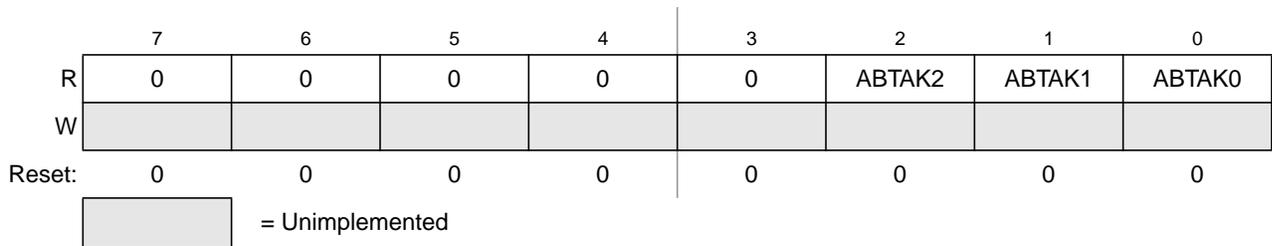
**Table 1-13. CANTARQ Register Field Descriptions**

Field	Description
2:0 ABTRQ[2:0]	<p><b>Abort Request</b> — The CPU sets the ABTRQx bit to request that a scheduled message buffer (TXEx = 0) be aborted. The MSCAN grants the request if the message has not already started transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted, the associated TXE (see Section 1.3.2.7, "MSCAN Transmitter Flag Register (CANTFLG)") and abort acknowledge flags (ABTAK, see Section 1.3.2.10, "MSCAN Transmitter Message Abort Acknowledge Register (CANTAACK)") are set and a transmit interrupt occurs if enabled. The CPU cannot reset ABTRQx. ABTRQx is reset whenever the associated TXE flag is set.</p> <p>0 No abort request 1 Abort request pending</p>

### 1.3.2.10 MSCAN Transmitter Message Abort Acknowledge Register (CANTAACK)

The CANTAACK register indicates the successful abort of a queued message, if requested by the appropriate bits in the CANTARQ register.

Module Base + 0x0009



**Figure 1-13. MSCAN Transmitter Message Abort Acknowledge Register (CANTAACK)**

#### NOTE

The CANTAACK register is held in the reset state when the initialization mode is active (INTRQ = 1 and INITAK = 1).

Read: Anytime

Write: Unimplemented for ABTAKx flags

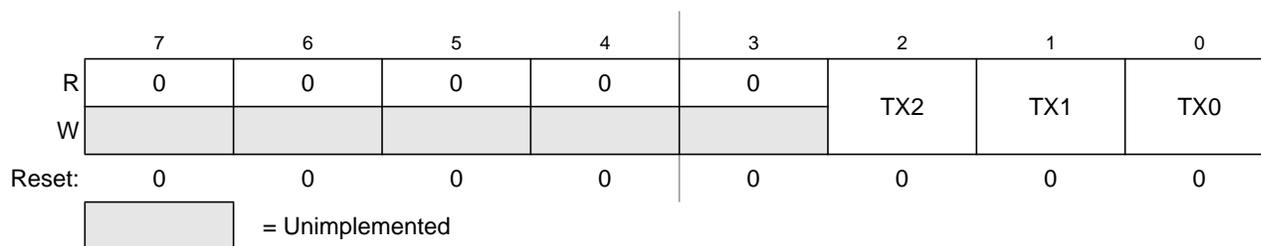
**Table 1-14. CANTAACK Register Field Descriptions**

Field	Description
2:0 ABTAK[2:0]	<p><b>Abort Acknowledge</b> — This flag acknowledges that a message was aborted due to a pending abort request from the CPU. After a particular message buffer is flagged empty, this flag can be used by the application software to identify whether the message was aborted successfully or was sent anyway. The ABTAKx flag is cleared whenever the corresponding TXE flag is cleared.</p> <p>0 The message was not aborted. 1 The message was aborted.</p>

### 1.3.2.11 MSCAN Transmit Buffer Selection Register (CANTBSEL)

The CANTBSEL register allows the selection of the actual transmit message buffer, which then will be accessible in the CANTXFG register space.

Module Base + 0x000A

**Figure 1-14. MSCAN Transmit Buffer Selection Register (CANTBSEL)****NOTE**

The CANTBSEL register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK=1). This register is writable when not in initialization mode (INITRQ = 0 and INITAK = 0).

Read: Find the lowest ordered bit set to 1, all other bits will be read as 0

Write: Anytime when not in initialization mode

**Table 1-15. CANTBSEL Register Field Descriptions**

Field	Description
2:0 TX[2:0]	<p><b>Transmit Buffer Select</b> — The lowest numbered bit places the respective transmit buffer in the CANTXFG register space (e.g., TX1 = 1 and TX0 = 1 selects transmit buffer TX0; TX1 = 1 and TX0 = 0 selects transmit buffer TX1). Read and write accesses to the selected transmit buffer will be blocked, if the corresponding TXEx bit is cleared and the buffer is scheduled for transmission (see <a href="#">Section 1.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”</a>).</p> <p>0 The associated message buffer is deselected 1 The associated message buffer is selected, if lowest numbered bit</p>

The following gives a short programming example of the usage of the CANTBSEL register:

To get the next available transmit buffer, application software must read the CANTFLG register and write this value back into the CANTBSEL register. In this example Tx buffers TX1 and TX2 are available. The value read from CANTFLG is therefore 0b0000\_0110. When writing this value back to CANTBSEL, the Tx buffer TX1 is selected in the CANTXFG because the lowest numbered bit set to 1 is at bit position 1. Reading back this value out of CANTBSEL results in 0b0000\_0010, because only the lowest numbered bit position set to 1 is presented. This mechanism eases the application software the selection of the next available Tx buffer.

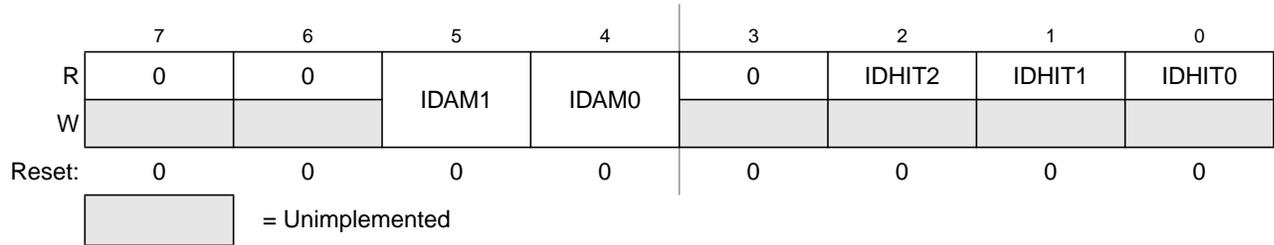
- LDD CANTFLG; value read is 0b0000\_0110
- STD CANTBSEL; value written is 0b0000\_0110
- LDD CANTBSEL; value read is 0b0000\_0010

If all transmit message buffers are deselected, no accesses are allowed to the CANTXFG registers.

**1.3.2.12 MSCAN Identifier Acceptance Control Register (CANIDAC)**

The CANIDAC register is used for identifier acceptance control as described below.

Module Base + 0x000B



**Figure 1-15. MSCAN Identifier Acceptance Control Register (CANIDAC)**

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1), except bits IDHITx, which are read-only

**Table 1-16. CANIDAC Register Field Descriptions**

Field	Description
5:4 IDAM[1:0]	<b>Identifier Acceptance Mode</b> — The CPU sets these flags to define the identifier acceptance filter organization (see Section 1.4.3, “Identifier Acceptance Filter”). Table 1-18 summarizes the different settings. In filter closed mode, no message is accepted such that the foreground buffer is never reloaded.
2:0 IDHIT[2:0]	Identifier Acceptance Hit Indicator — The MSCAN sets these flags to indicate an identifier acceptance hit (see Section 1.4.3, “Identifier Acceptance Filter”). Table 1-19 summarizes the different settings.

**Table 1-17. Identifier Acceptance Mode Settings**

IDAM1	IDAM0	Identifier Acceptance Mode
0	0	Two 32-bit acceptance filters
0	1	Four 16-bit acceptance filters
1	0	Eight 8-bit acceptance filters
1	1	Filter closed

**Table 1-18. Identifier Acceptance Hit Indication**

IDHIT2	IDHIT1	IDHIT0	Identifier Acceptance Hit
0	0	0	Filter 0 hit
0	0	1	Filter 1 hit
0	1	0	Filter 2 hit
0	1	1	Filter 3 hit
1	0	0	Filter 4 hit
1	0	1	Filter 5 hit
1	1	0	Filter 6 hit
1	1	1	Filter 7 hit

The IDHITx indicators are always related to the message in the foreground buffer (RxFG). When a message gets shifted into the foreground buffer of the receiver FIFO the indicators are updated as well.

### 1.3.2.13 MSCAN Reserved Registers

These registers are reserved for factory testing of the MSCAN module and is not available in normal system operation modes.

Module Base + 0x000C, 0x000D

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 1-16. MSCAN Reserved Registers**

Read: Always read 0x0000 in normal system operation modes

Write: Unimplemented in normal system operation modes

#### NOTE

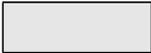
Writing to this register when in special modes can alter the MSCAN functionality.

### 1.3.2.14 MSCAN Receive Error Counter (CANRXERR)

This register reflects the status of the MSCAN receive error counter.

Module Base + 0x000E

	7	6	5	4	3	2	1	0
R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
W								
Reset:	0	0	0	0	0	0	0	0

 = Unimplemented

**Figure 1-17. MSCAN Receive Error Counter (CANRXERR)**

Read: Only when in sleep mode (SLPRQ = 1 and SLPK = 1) or initialization mode (INITRQ = 1 and INITAK = 1)

Write: Unimplemented

#### NOTE

Reading this register when in any other mode other than sleep or initialization mode may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition.

Writing to this register when in special modes can alter the MSCAN functionality.

### 1.3.2.15 MSCAN Transmit Error Counter (CANTXERR)

This register reflects the status of the MSCAN transmit error counter.

Module Base + 0x000F



**Figure 1-18. MSCAN Transmit Error Counter (CANTXERR)**

Read: Only when in sleep mode (SLPRQ = 1 and SLPK = 1) or initialization mode (INITRQ = 1 and INITAK = 1)

Write: Unimplemented

#### NOTE

Reading this register when in any other mode other than sleep or initialization mode, may return an incorrect value. For MCUs with dual CPUs, this may result in a CPU fault condition.

Writing to this register when in special modes can alter the MSCAN functionality.

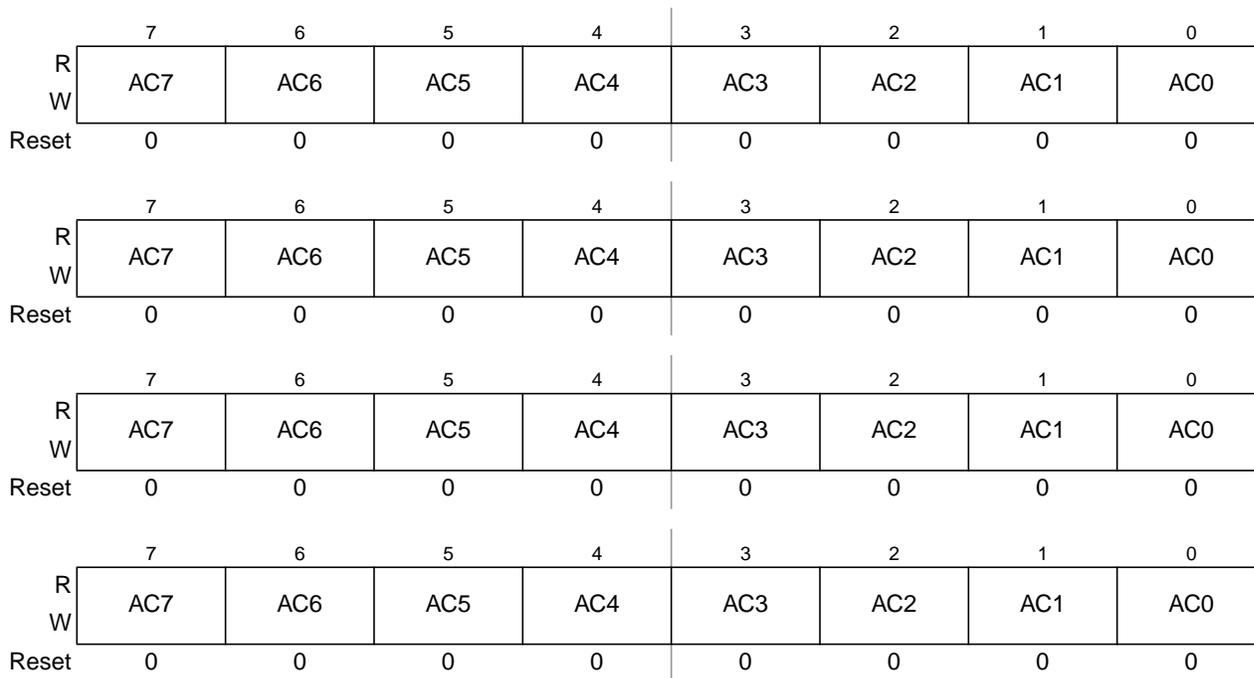
### 1.3.2.16 MSCAN Identifier Acceptance Registers (CANIDAR0-7)

On reception, each message is written into the background receive buffer. The CPU is only signalled to read the message if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message is overwritten by the next message (dropped).

The acceptance registers of the MSCAN are applied on the IDR0–IDR3 registers (see [Section 1.3.3.1, “Identifier Registers \(IDR0–IDR3\)”](#)) of incoming messages in a bit by bit manner (see [Section 1.4.3, “Identifier Acceptance Filter”](#)).

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers, only the first two (CANIDAR0/1, CANIDMR0/1) are applied.

Module Base + 0x0010 (CANIDAR0)  
 0x0011 (CANIDAR1)  
 0x0012 (CANIDAR2)  
 0x0013 (CANIDAR3)



**Figure 1-19. MSCAN Identifier Acceptance Registers (First Bank) — CANIDAR0–CANIDAR3**

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 1-19. CANIDAR0–CANIDAR3 Register Field Descriptions**

Field	Description
7:0 AC[7:0]	<b>Acceptance Code Bits</b> — AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

Module Base + 0x0018 (CANIDAR4)  
 0x0019 (CANIDAR5)  
 0x001A (CANIDAR6)  
 0x001B (CANIDAR7)

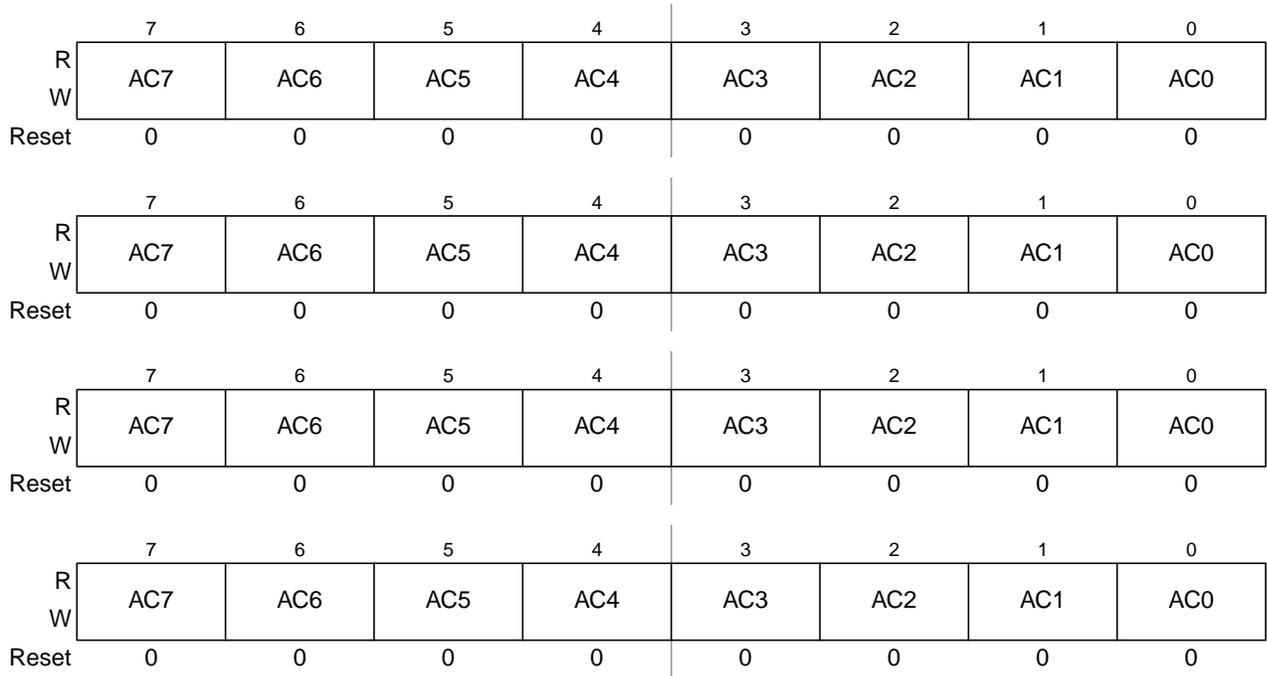


Figure 1-20. MSCAN Identifier Acceptance Registers (Second Bank) — CANIDAR4–CANIDAR7

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

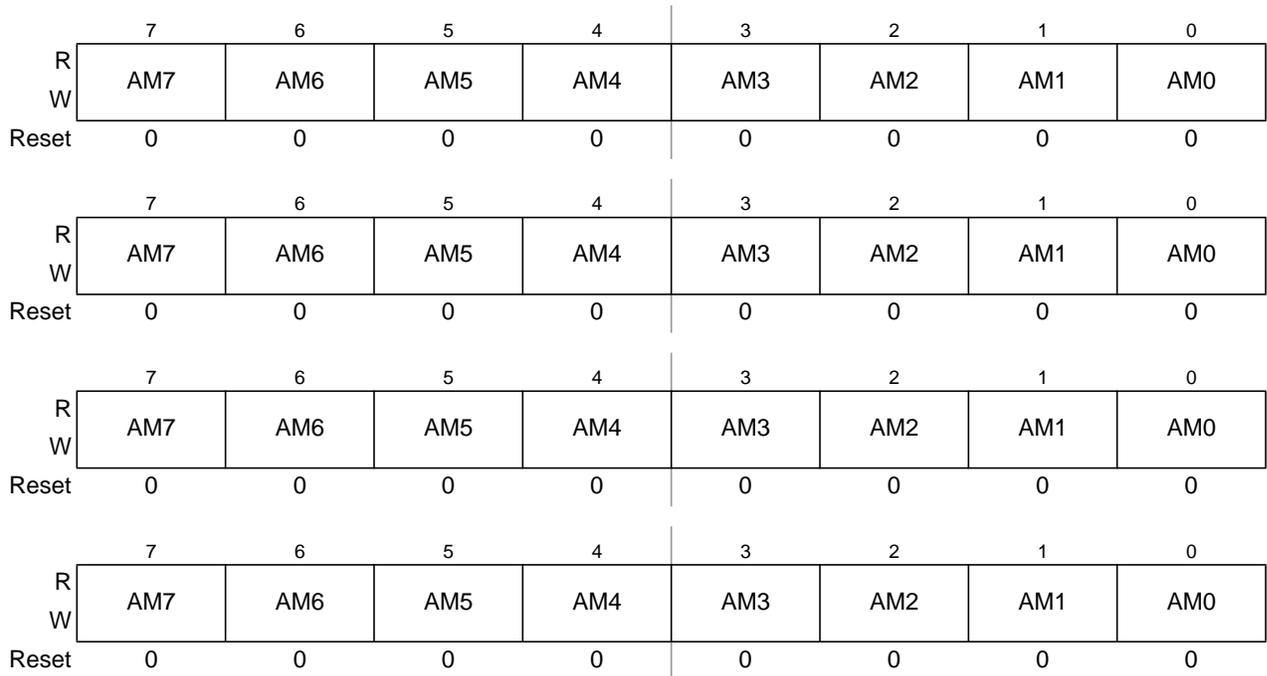
Table 1-20. CANIDAR4–CANIDAR7 Register Field Descriptions

Field	Description
7:0 AC[7:0]	<b>Acceptance Code Bits</b> — AC[7:0] comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

### 1.3.2.17 MSCAN Identifier Mask Registers (CANIDMR0–CANIDMR7)

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. To receive standard identifiers in 32 bit filter mode, it is required to program the last three bits (AM[2:0]) in the mask registers CANIDMR1 and CANIDMR5 to “don’t care.” To receive standard identifiers in 16 bit filter mode, it is required to program the last three bits (AM[2:0]) in the mask registers CANIDMR1, CANIDMR3, CANIDMR5, and CANIDMR7 to “don’t care.”

Module Base + 0x0014 (CANIDMR0)  
 0x0015 (CANIDMR1)  
 0x0016 (CANIDMR2)  
 0x0017 (CANIDMR3)



**Figure 1-21. MSCAN Identifier Mask Registers (First Bank) — CANIDMR0–CANIDMR3**

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 1-21. CANIDMR0–CANIDMR3 Register Field Descriptions**

Field	Description
7:0 AM[7:0]	<p><b>Acceptance Mask Bits</b> — If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted.</p> <p>0 Match corresponding acceptance code register and identifier bits                      1 Ignore corresponding acceptance code register bit</p>

Module Base + 0x001C (CANIDMR4)  
 0x001D (CANIDMR5)  
 0x001E (CANIDMR6)  
 0x001F (CANIDMR7)

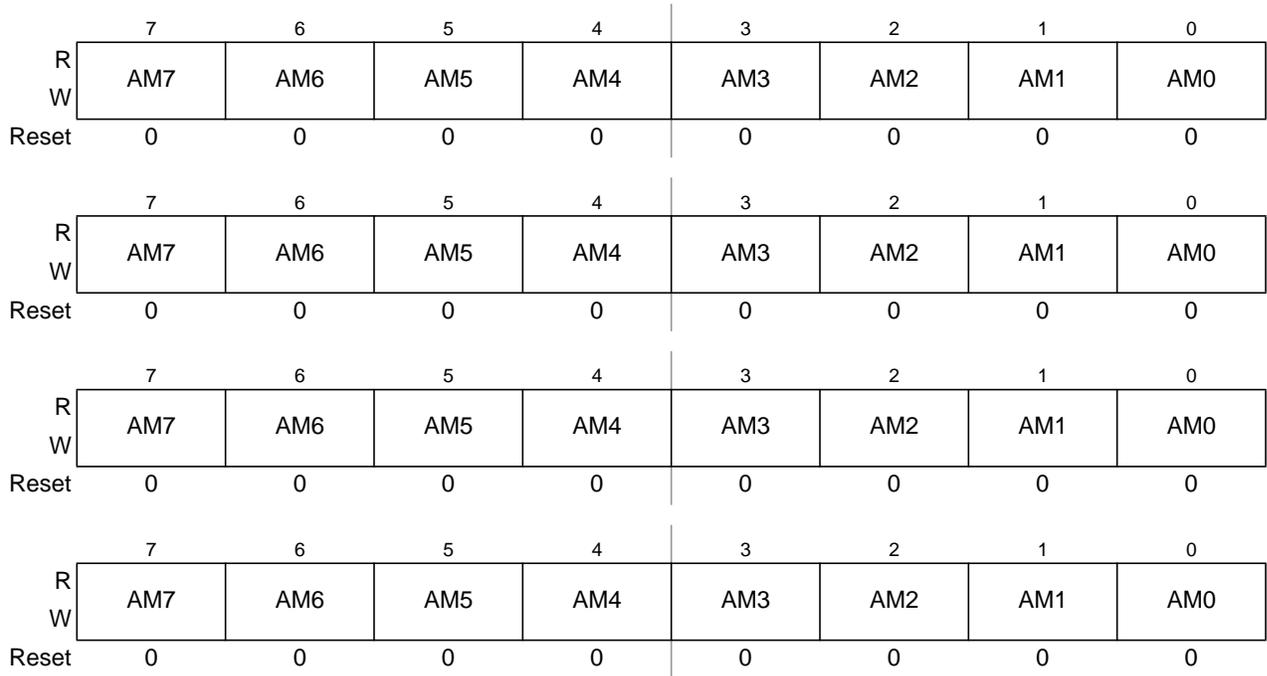


Figure 1-22. MSCAN Identifier Mask Registers (Second Bank) — CANIDMR4–CANIDMR7

Read: Anytime

Write: Anytime in initialization mode (INTRQ = 1 and INITAK = 1)

Table 1-22. CANIDMR4–CANIDMR7 Register Field Descriptions

Field	Description
7:0 AM[7:0]	<p><b>Acceptance Mask Bits</b> — If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted.</p> <p>0 Match corresponding acceptance code register and identifier bits                      1 Ignore corresponding acceptance code register bit</p>

### 1.3.3 Programmer's Model of Message Storage

The following section details the organization of the receive and transmit message buffers and the associated control registers.

To simplify the programmer interface, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13 byte data structure.

An additional transmit buffer priority register (TBPR) is defined for the transmit buffers. Within the last two bytes of this memory map, the MSCAN stores a special 16-bit time stamp, which is sampled from an internal timer after successful transmission or reception of a message. This feature is only available for transmit and receiver buffers, if the TIME bit is set (see [Section 1.3.2.1, “MSCAN Control Register 0 \(CANCTL0\)”](#)).

The time stamp register is written by the MSCAN. The CPU can only read these registers.

**Table 1-23. Message Buffer Organization**

Offset Address	Register	Access
0x00X0	Identifier Register 0	
0x00X1	Identifier Register 1	
0x00X2	Identifier Register 2	
0x00X3	Identifier Register 3	
0x00X4	Data Segment Register 0	
0x00X5	Data Segment Register 1	
0x00X6	Data Segment Register 2	
0x00X7	Data Segment Register 3	
0x00X8	Data Segment Register 4	
0x00X9	Data Segment Register 5	
0x00XA	Data Segment Register 6	
0x00XB	Data Segment Register 7	
0x00XC	Data Length Register	
0x00XD	Transmit Buffer Priority Register <sup>1</sup>	
0x00XE	Time Stamp Register (High Byte) <sup>2</sup>	
0x00XF	Time Stamp Register (Low Byte) <sup>3</sup>	

<sup>1</sup> Not applicable for receive buffers

<sup>2</sup> Read-only for CPU

<sup>3</sup> Read-only for CPU

[Figure 1-25](#) shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in [Figure 1-26](#).

All bits of the receive and transmit buffers are ‘x’ out of reset because of RAM-based implementation<sup>1</sup>. All reserved or unused bits of the receive and transmit buffers always read ‘x’.

1. Exception: The transmit priority registers are 0 out of reset.

Register Name		Bit 7	6	5	4	3	2	1	Bit0
0x00X0 IDR0	R W	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
0x00X1 IDR1	R W	ID20	ID19	ID18	SRR (=1)	IDE (=1)	ID17	ID16	ID15
0x00X2 IDR2	R W	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
0x00X3 IDR3	R W	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
0x00X4 DSR0	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X5 DSR1	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X6 DSR2	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X7 DSR3	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X8 DSR4	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00X9 DSR5	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00XA DSR6	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00XB DSR7	R W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0x00XC DLR	R W					DLC3	DLC2	DLC1	DLC0

= Unused, always read 'x'

**Figure 1-23. Receive/Transmit Message Buffer — Extended Identifier Mapping**

Read: For transmit buffers, anytime when TXEx flag is set (see [Section 1.3.2.7, “MSCAN Transmitter Flag Register \(CANTFLG\)”](#)) and the corresponding transmit buffer is selected in CANTBSEL (see [Section 1.3.2.11, “MSCAN Transmit Buffer Selection Register \(CANTBSEL\)”](#)). For receive buffers, only when RXF flag is set (see [Section 1.3.2.5, “MSCAN Receiver Flag Register \(CANRFLG\)”](#)).

Write: For transmit buffers, anytime when TXEx flag is set (see Section 1.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 1.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”). Unimplemented for receive buffers.

Reset: Undefined (0x00XX) because of RAM-based implementation

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
IDR0 0x00X0	R W	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
IDR1 0x00X1	R W	ID2	ID1	ID0	RTR	IDE (=0)			
IDR2 0x00X2	R W								
IDR3 0x00X3	R W								

= Unused, always read 'x'

Figure 1-24. Receive/Transmit Message Buffer — Standard Identifier Mapping

### 1.3.3.1 Identifier Registers (IDR0–IDR3)

The identifier registers for an extended format identifier consist of a total of 32 bits; ID[28:0], SRR, IDE, and RTR bits. The identifier registers for a standard format identifier consist of a total of 13 bits; ID[10:0], RTR, and IDE bits.

#### 1.3.3.1.1 IDR0–IDR3 for Extended Identifier Mapping

Module Base + 0x00X1

	7	6	5	4	3	2	1	0
R W	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
Reset:	x	x	x	x	x	x	x	x

Figure 1-25. Identifier Register 0 (IDR0) — Extended Identifier Mapping

Table 1-24. IDR0 Register Field Descriptions — Extended

Field	Description
7:0 ID[28:21]	<b>Extended Format Identifier</b> — The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

Module Base + 0x00X1

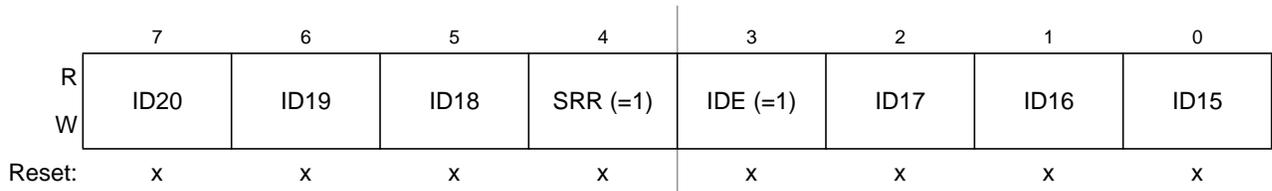


Figure 1-26. Identifier Register 1 (IDR1) — Extended Identifier Mapping

Table 1-25. IDR1 Register Field Descriptions — Extended

Field	Description
7:5 ID[20:18]	<b>Extended Format Identifier</b> — The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.
4 SRR	<b>Substitute Remote Request</b> — This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and is stored as received on the CAN bus for receive buffers.
3 IDE	<b>ID Extended</b> — This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send. 0 Standard format (11 bit) 1 Extended format (29 bit)
2:0 ID[17:15]	<b>Extended Format Identifier</b> — The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

Module Base + 0x00X2

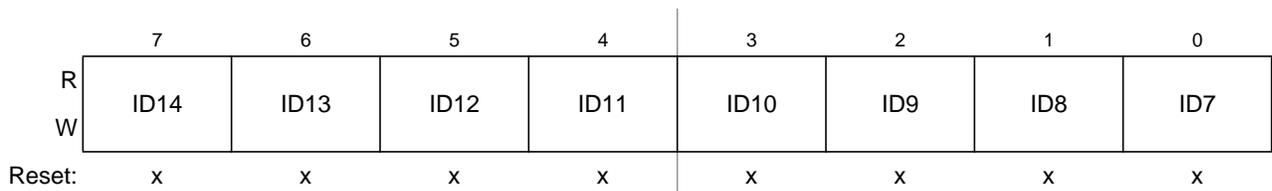


Figure 1-27. Identifier Register 2 (IDR2) — Extended Identifier Mapping

Table 1-26. IDR2 Register Field Descriptions — Extended

Field	Description
7:0 ID[14:7]	<b>Extended Format Identifier</b> — The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

Module Base + 0x00X3

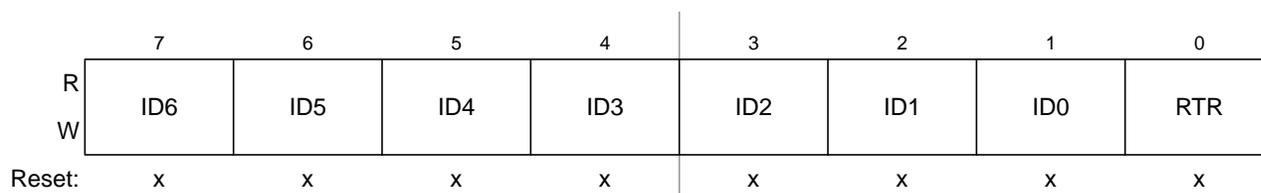


Figure 1-28. Identifier Register 3 (IDR3) — Extended Identifier Mapping

Table 1-27. IDR3 Register Field Descriptions — Extended

Field	Description
7:1 ID[6:0]	<b>Extended Format Identifier</b> — The identifiers consist of 29 bits (ID[28:0]) for the extended format. ID28 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.
0 RTR	<b>Remote Transmission Request</b> — This flag reflects the status of the remote transmission request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent. 0 Data frame 1 Remote frame

### 1.3.3.1.2 IDR0–IDR3 for Standard Identifier Mapping

Module Base + 0x00X0

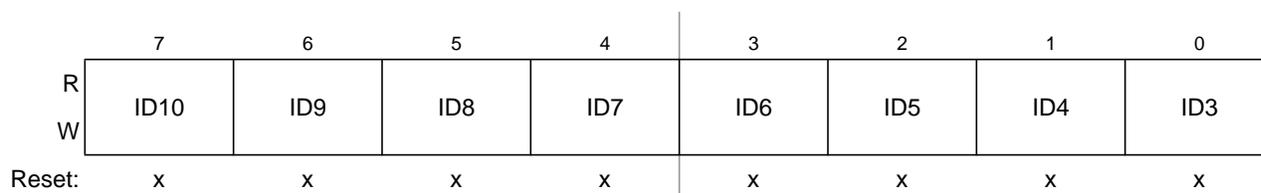


Figure 1-29. Identifier Register 0 — Standard Mapping

Table 1-28. IDR0 Register Field Descriptions — Standard

Field	Description
7:0 ID[10:3]	<b>Standard Format Identifier</b> — The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. See also ID bits in <a href="#">Table 1-31</a> .

Module Base + 0x00X1

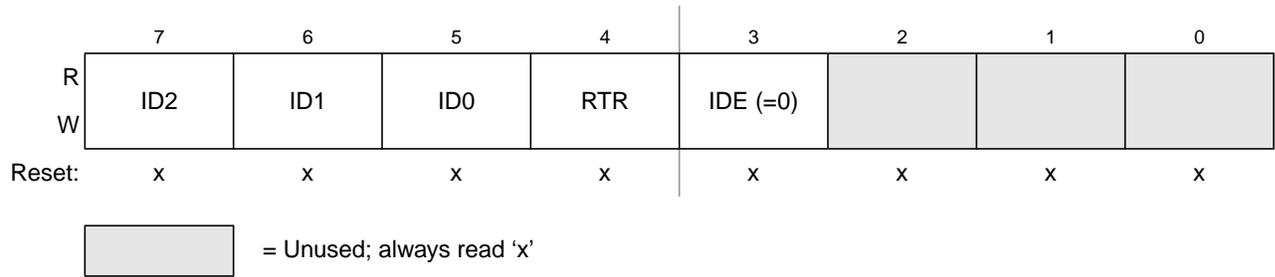


Figure 1-30. Identifier Register 1 — Standard Mapping

Table 1-29. IDR1 Register Field Descriptions

Field	Description
7:5 ID[2:0]	<b>Standard Format Identifier</b> — The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. See also ID bits in Table 1-30.
4 RTR	<b>Remote Transmission Request</b> — This flag reflects the status of the Remote Transmission Request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent. 0 Data frame 1 Remote frame
3 IDE	<b>ID Extended</b> — This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send. 0 Standard format (11 bit) 1 Extended format (29 bit)

Module Base + 0x00X2

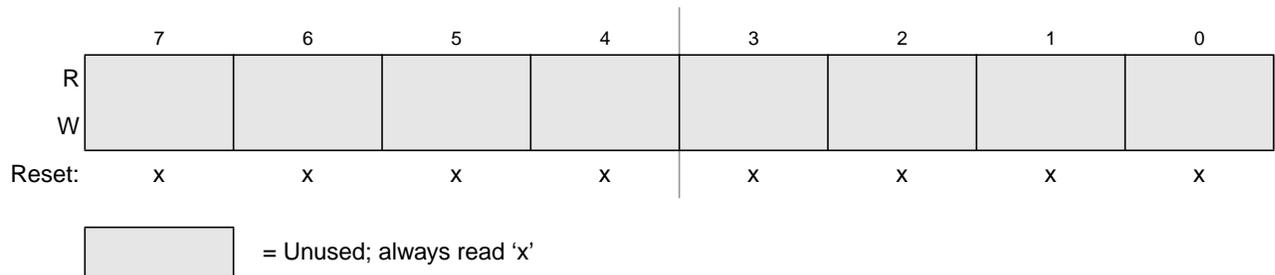


Figure 1-31. Identifier Register 2 — Standard Mapping

Module Base + 0x00X3

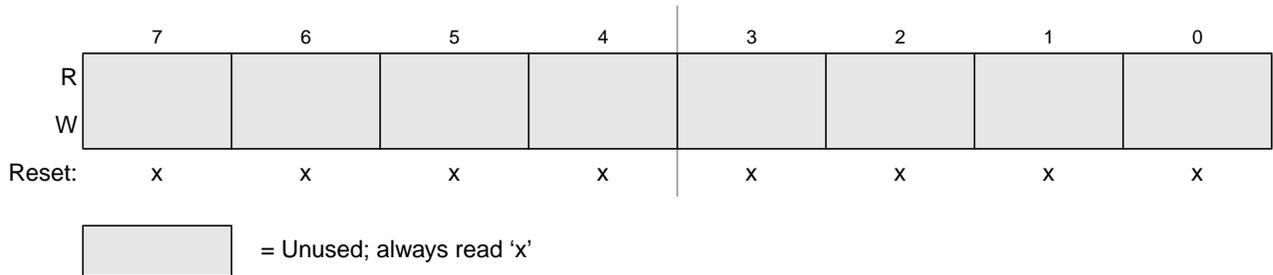


Figure 1-32. Identifier Register 3 — Standard Mapping

### 1.3.3.2 Data Segment Registers (DSR0-7)

The eight data segment registers, each with bits DB[7:0], contain the data to be transmitted or received. The number of bytes to be transmitted or received is determined by the data length code in the corresponding DLR register.

- Module Base + 0x0004 (DSR0)
- 0x0005 (DSR1)
- 0x0006 (DSR2)
- 0x0007 (DSR3)
- 0x0008 (DSR4)
- 0x0009 (DSR5)
- 0x000A (DSR6)
- 0x000B (DSR7)

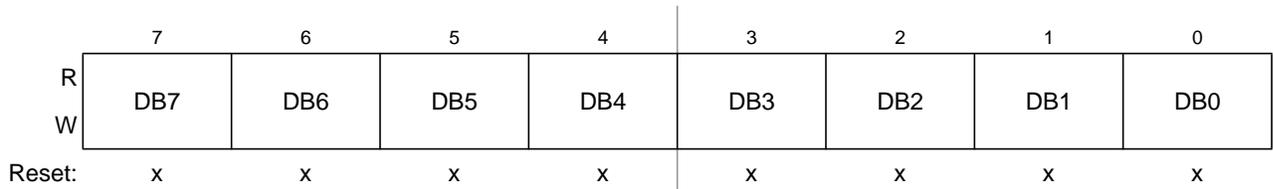


Figure 1-33. Data Segment Registers (DSR0–DSR7) — Extended Identifier Mapping

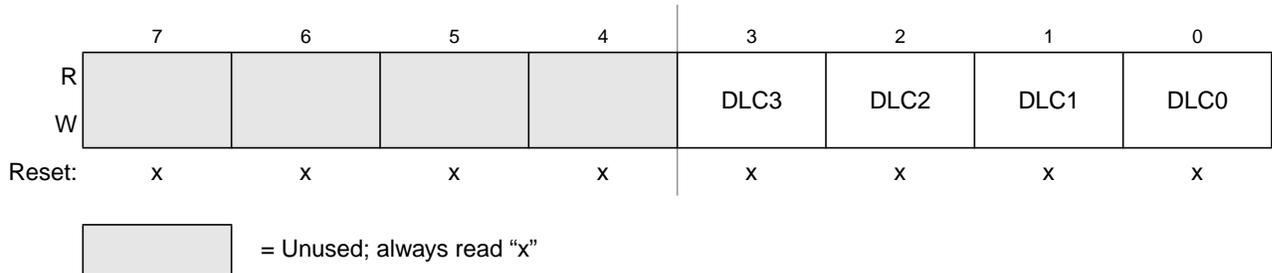
Table 1-30. DSR0–DSR7 Register Field Descriptions

Field	Description
7:0 DB[7:0]	Data bits 7:0

### 1.3.3.3 Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

Module Base + 0x00XB



**Figure 1-34. Data Length Register (DLR) — Extended Identifier Mapping**

**Table 1-31. DLR Register Field Descriptions**

Field	Description
3:0 DLC[3:0]	<b>Data Length Code Bits</b> — The data length code contains the number of bytes (data byte count) of the respective message. During the transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted data bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. <a href="#">Table 1-34</a> shows the effect of setting the DLC bits.

**Table 1-32. Data Length Codes**

Data Length Code				Data Byte Count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

### 1.3.3.4 Transmit Buffer Priority Register (TBPR)

This register defines the local priority of the associated message buffer. The local priority is used for the internal prioritization process of the MSCAN and is defined to be highest for the smallest binary number. The MSCAN implements the following internal prioritization mechanisms:

- All transmission buffers with a cleared TXEx flag participate in the prioritization immediately before the SOF (start of frame) is sent.
- The transmission buffer with the lowest local priority field wins the prioritization.

In cases of more than one buffer having the same lowest priority, the message buffer with the lower index number wins.

Module Base + 0xXXXD

	7	6	5	4	3	2	1	0
R	PRI07	PRI06	PRI05	PRI04	PRI03	PRI02	PRI01	PRI00
W								
Reset:	0	0	0	0	0	0	0	0

**Figure 1-35. Transmit Buffer Priority Register (TBPR)**

**Read:** Anytime when TXEx flag is set (see Section 1.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 1.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”).

**Write:** Anytime when TXEx flag is set (see Section 1.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 1.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”).

### 1.3.3.5 Time Stamp Register (TSRH–TSRL)

If the TIME bit is enabled, the MSCAN will write a time stamp to the respective registers in the active transmit or receive buffer right after the EOF of a valid message on the CAN bus (see Section 1.3.2.1, “MSCAN Control Register 0 (CANCTL0)”). In case of a transmission, the CPU can only read the time stamp after the respective transmit buffer has been flagged empty.

The timer value, which is used for stamping, is taken from a free running internal CAN bit clock. A timer overrun is not indicated by the MSCAN. The timer is reset (all bits set to 0) during initialization mode. The CPU can only read the time stamp registers.

Module Base + 0xXXXE

	7	6	5	4	3	2	1	0
R	TSR15	TSR14	TSR13	TSR12	TSR11	TSR10	TSR9	TSR8
W								
Reset:	x	x	x	x	x	x	x	x

**Figure 1-36. Time Stamp Register — High Byte (TSRH)**

Module Base + 0xXXXF

	7	6	5	4	3	2	1	0
R	TSR7	TSR6	TSR5	TSR4	TSR3	TSR2	TSR1	TSR0
W								
Reset:	x	x	x	x	x	x	x	x

**Figure 1-37. Time Stamp Register — Low Byte (TSRL)**

Read: Anytime when TXEx flag is set (see Section 1.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 1.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”).

Write: Unimplemented

## 1.4 Functional Description

### 1.4.1 General

This section provides a complete functional description of the MSCAN. It describes each of the features and modes listed in the introduction.

## 1.4.2 Message Storage

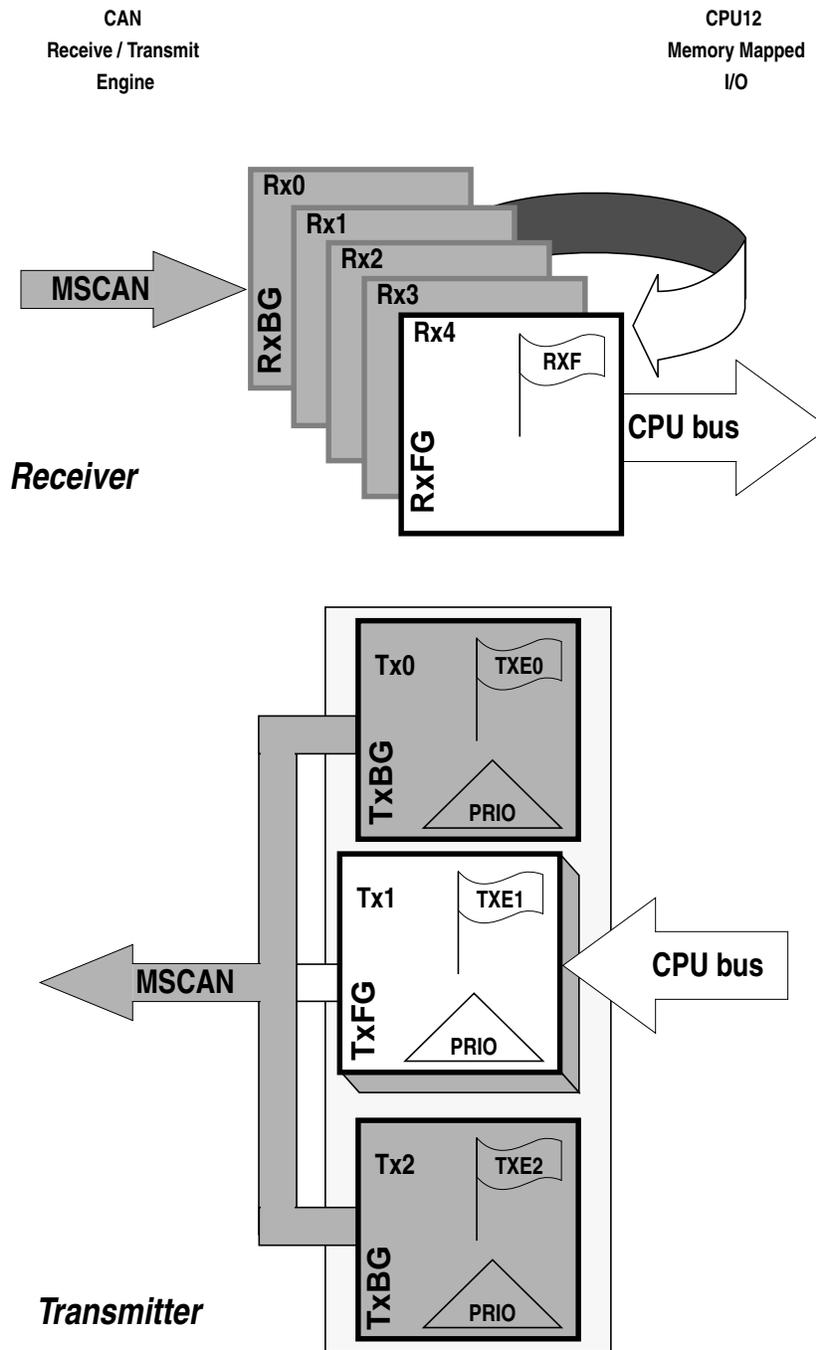


Figure 1-38. User Model for Message Buffer Organization

MSCAN facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

### 1.4.2.1 Message Transmit Background

Modern application layer software is built upon two fundamental assumptions:

- Any CAN node is able to send out a stream of scheduled messages without releasing the CAN bus between the two messages. Such nodes arbitrate for the CAN bus immediately after sending the previous message and only release the CAN bus in case of lost arbitration.
- The internal message queue within any CAN node is organized such that the highest priority message is sent out first, if more than one message is ready to be sent.

The behavior described in the bullets above cannot be achieved with a single transmit buffer. That buffer must be reloaded immediately after the previous message is sent. This loading process lasts a finite amount of time and must be completed within the inter-frame sequence (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme de-couples the reloading of the transmit buffer from the actual message sending and, therefore, reduces the reactivity requirements of the CPU. Problems can arise if the sending of a message is finished while the CPU re-loads the second buffer. No buffer would then be ready for transmission, and the CAN bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN implements with the “local priority” concept described in [Section 1.4.2.2, “Transmit Structures.”](#)

### 1.4.2.2 Transmit Structures

The MSCAN triple transmit buffer scheme optimizes real-time performance by allowing multiple messages to be set up in advance. The three buffers are arranged as shown in [Figure 1-40](#).

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see [Section 1.3.3, “Programmer’s Model of Message Storage”](#)). An additional [Section 1.3.3.4, “Transmit Buffer Priority Register \(TBPR\)”](#) contains an 8-bit local priority field (PRIO) (see [Section 1.3.3.4, “Transmit Buffer Priority Register \(TBPR\)”](#)). The remaining two bytes are used for time stamping of a message, if required (see [Section 1.3.3.5, “Time Stamp Register \(TSRH–TSRL\)”](#)).

To transmit a message, the CPU must identify an available transmit buffer, which is indicated by a set transmitter buffer empty (TXEx) flag (see [Section 1.3.2.7, “MSCAN Transmitter Flag Register \(CANTFLG\)”](#)). If a transmit buffer is available, the CPU must set a pointer to this buffer by writing to the CANTBSEL register (see [Section 1.3.2.11, “MSCAN Transmit Buffer Selection Register \(CANTBSEL\)”](#)). This makes the respective buffer accessible within the CANTXFG address space (see [Section 1.3.3, “Programmer’s Model of Message Storage”](#)). The algorithmic feature associated with the CANTBSEL register simplifies the transmit buffer selection. In addition, this scheme makes the handler software simpler because only one address area is applicable for the transmit process, and the required address space is minimized.

The CPU then stores the identifier, the control bits, and the data content into one of the transmit buffers. Finally, the buffer is flagged as ready for transmission by clearing the associated TXE flag.

The MSCAN then schedules the message for transmission and signals the successful transmission of the buffer by setting the associated TXE flag. A transmit interrupt (see [Section 1.4.7.2, “Transmit Interrupt”](#)) is generated<sup>1</sup> when TXEx is set and can be used to drive the application software to re-load the buffer.

If more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN uses the local priority setting of the three buffers to determine the prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software programs this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being transmitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority. The internal scheduling process takes place whenever the MSCAN arbitrates for the CAN bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message in one of the three transmit buffers. Because messages that are already in transmission cannot be aborted, the user must request the abort by setting the corresponding abort request bit (ABTRQ) (see [Section 1.3.2.9, “MSCAN Transmitter Message Abort Request Register \(CANTARQ\)”](#).) The MSCAN then grants the request, if possible, by:

1. Setting the corresponding abort acknowledge flag (ABTAK) in the CANTAACK register.
2. Setting the associated TXE flag to release the buffer.
3. Generating a transmit interrupt. The transmit interrupt handler software can determine from the setting of the ABTAK flag whether the message was aborted (ABTAK = 1) or sent (ABTAK = 0).

### 1.4.2.3 Receive Structures

The received messages are stored in a five stage input FIFO. The five message buffers are alternately mapped into a single memory area (see [Figure 1-40](#)). The background receive buffer (RxBG) is exclusively associated with the MSCAN, but the foreground receive buffer (RxFG) is addressable by the CPU (see [Figure 1-40](#)). This scheme simplifies the handler software because only one address area is applicable for the receive process.

All receive buffers have a size of 15 bytes to store the CAN control bits, the identifier (standard or extended), the data contents, and a time stamp, if enabled (see [Section 1.3.3, “Programmer’s Model of Message Storage”](#)).

The receiver full flag (RXF) (see [Section 1.3.2.5, “MSCAN Receiver Flag Register \(CANRFLG\)”](#)) signals the status of the foreground receive buffer. When the buffer contains a correctly received message with a matching identifier, this flag is set.

On reception, each message is checked to see whether it passes the filter (see [Section 1.4.3, “Identifier Acceptance Filter”](#)) and simultaneously is written into the active RxBG. After successful reception of a valid message, the MSCAN shifts the content of RxBG into the receiver FIFO<sup>2</sup>, sets the RXF flag, and generates a receive interrupt (see [Section 1.4.7.3, “Receive Interrupt”](#)) to the CPU<sup>3</sup>. The user’s receive handler must read the received message from the RxFG and then reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message, which can follow immediately after the IFS

1. The transmit interrupt occurs only if not masked. A polling scheme can be applied on TXEx also.

2. Only if the RXF flag is not set.

3. The receive interrupt occurs only if not masked. A polling scheme can be applied on RXF also.

field of the CAN frame, is received into the next available RxBG. If the MSCAN receives an invalid message in its RxBG (wrong identifier, transmission errors, etc.) the actual contents of the buffer will be over-written by the next message. The buffer will then not be shifted into the FIFO.

When the MSCAN module is transmitting, the MSCAN receives its own transmitted messages into the background receive buffer, RxBG, but does not shift it into the receiver FIFO, generate a receive interrupt, or acknowledge its own messages on the CAN bus. The exception to this rule is in loopback mode (see [Section 1.3.2.2, “MSCAN Control Register 1 \(CANCTL1\)”](#)) where the MSCAN treats its own messages exactly like all other incoming messages. The MSCAN receives its own transmitted messages in the event that it loses arbitration. If arbitration is lost, the MSCAN must be prepared to become a receiver.

An overrun condition occurs when all receive message buffers in the FIFO are filled with correctly received messages with accepted identifiers and another message is correctly received from the CAN bus with an accepted identifier. The latter message is discarded and an error interrupt with overrun indication is generated if enabled (see [Section 1.4.7.5, “Error Interrupt”](#)). The MSCAN remains able to transmit messages while the receiver FIFO being filled, but all incoming messages are discarded. As soon as a receive buffer in the FIFO is available again, new valid messages will be accepted.

### 1.4.3 Identifier Acceptance Filter

The MSCAN identifier acceptance registers (see [Section 1.3.2.12, “MSCAN Identifier Acceptance Control Register \(CANIDAC\)”](#)) define the acceptable patterns of the standard or extended identifier (ID[10:0] or ID[28:0]). Any of these bits can be marked ‘don’t care’ in the MSCAN identifier mask registers (see [Section 1.3.2.18, “MSCAN Identifier Mask Registers \(CANIDMR0–CANIDMR7\)”](#)).

A filter hit is indicated to the application software by a set receive buffer full flag (RXF = 1) and three bits in the CANIDAC register (see [Section 1.3.2.12, “MSCAN Identifier Acceptance Control Register \(CANIDAC\)”](#)). These identifier hit flags (IDHIT[2:0]) clearly identify the filter section that caused the acceptance. They simplify the application software’s task to identify the cause of the receiver interrupt. If more than one hit occurs (two or more filters match), the lower hit has priority.

A very flexible programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes (see Bosch CAN 2.0A/B protocol specification):

- Two identifier acceptance filters, each to be applied to:
  - The full 29 bits of the extended identifier and to the following bits of the CAN 2.0B frame:
    - Remote transmission request (RTR)
    - Identifier extension (IDE)
    - Substitute remote request (SRR)
  - The 11 bits of the standard identifier plus the RTR and IDE bits of the CAN 2.0A/B messages<sup>1</sup>. This mode implements two filters for a full length CAN 2.0B compliant extended identifier. [Figure 1-41](#) shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces a filter 0 hit. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces a filter 1 hit.

<sup>1</sup>.Although this mode can be used for standard identifiers, it is recommended to use the four or eight identifier acceptance filters for standard identifiers

- Four identifier acceptance filters, each to be applied to
  - a) the 14 most significant bits of the extended identifier plus the SRR and IDE bits of CAN 2.0B messages or
  - b) the 11 bits of the standard identifier, the RTR and IDE bits of CAN 2.0A/B messages.
 Figure 1-42 shows how the first 32-bit filter bank (CANIDAR0–CANIDA3, CANIDMR0–3CANIDMR) produces filter 0 and 1 hits. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces filter 2 and 3 hits.
- Eight identifier acceptance filters, each to be applied to the first 8 bits of the identifier. This mode implements eight independent filters for the first 8 bits of a CAN 2.0A/B compliant standard identifier or a CAN 2.0B compliant extended identifier. Figure 1-43 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces filter 0 to 3 hits. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces filter 4 to 7 hits.
- Closed filter. No CAN message is copied into the foreground buffer RxFG, and the RXF flag is never set.

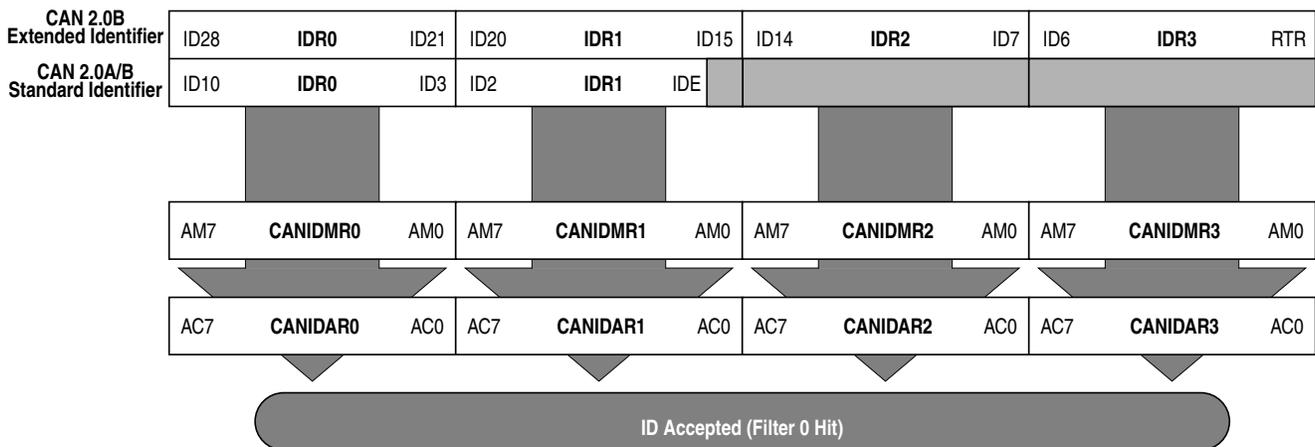


Figure 1-39. 32-bit Maskable Identifier Acceptance Filter

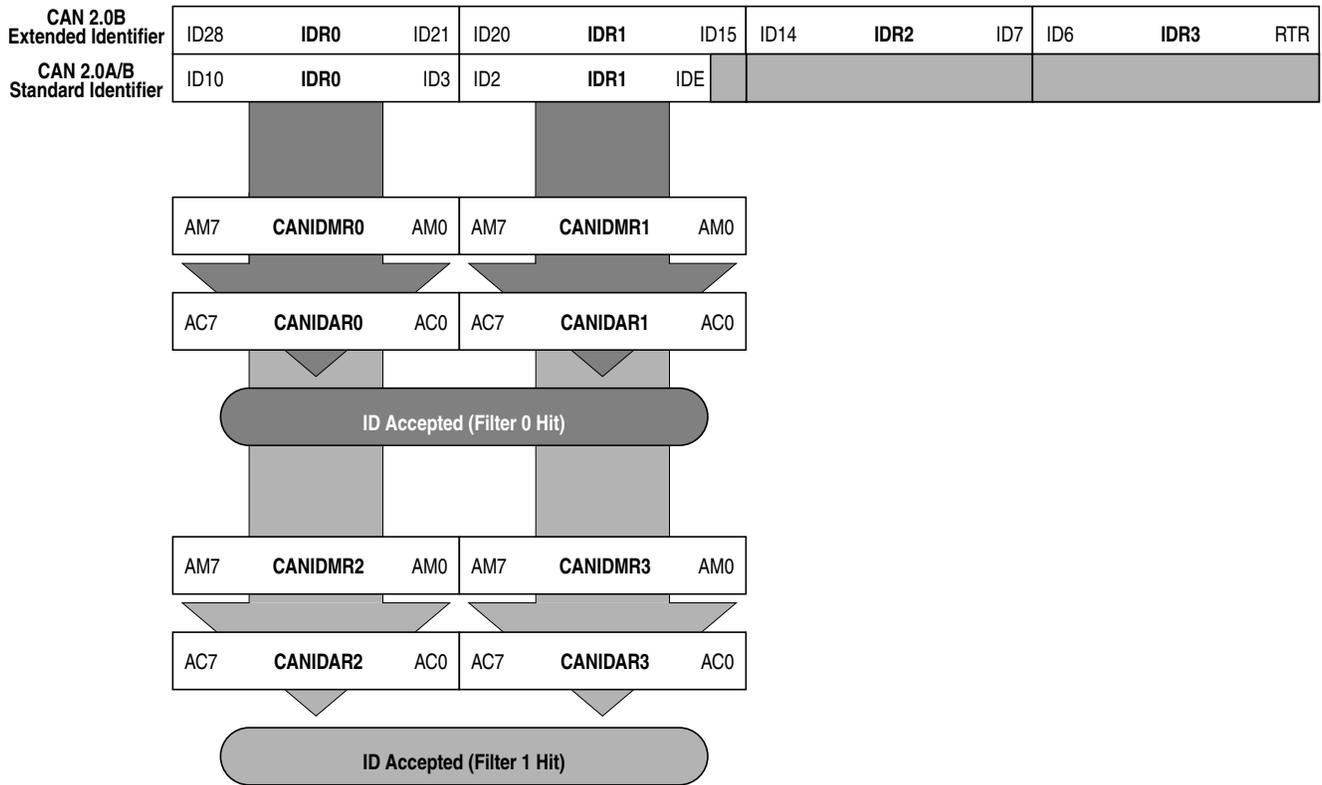
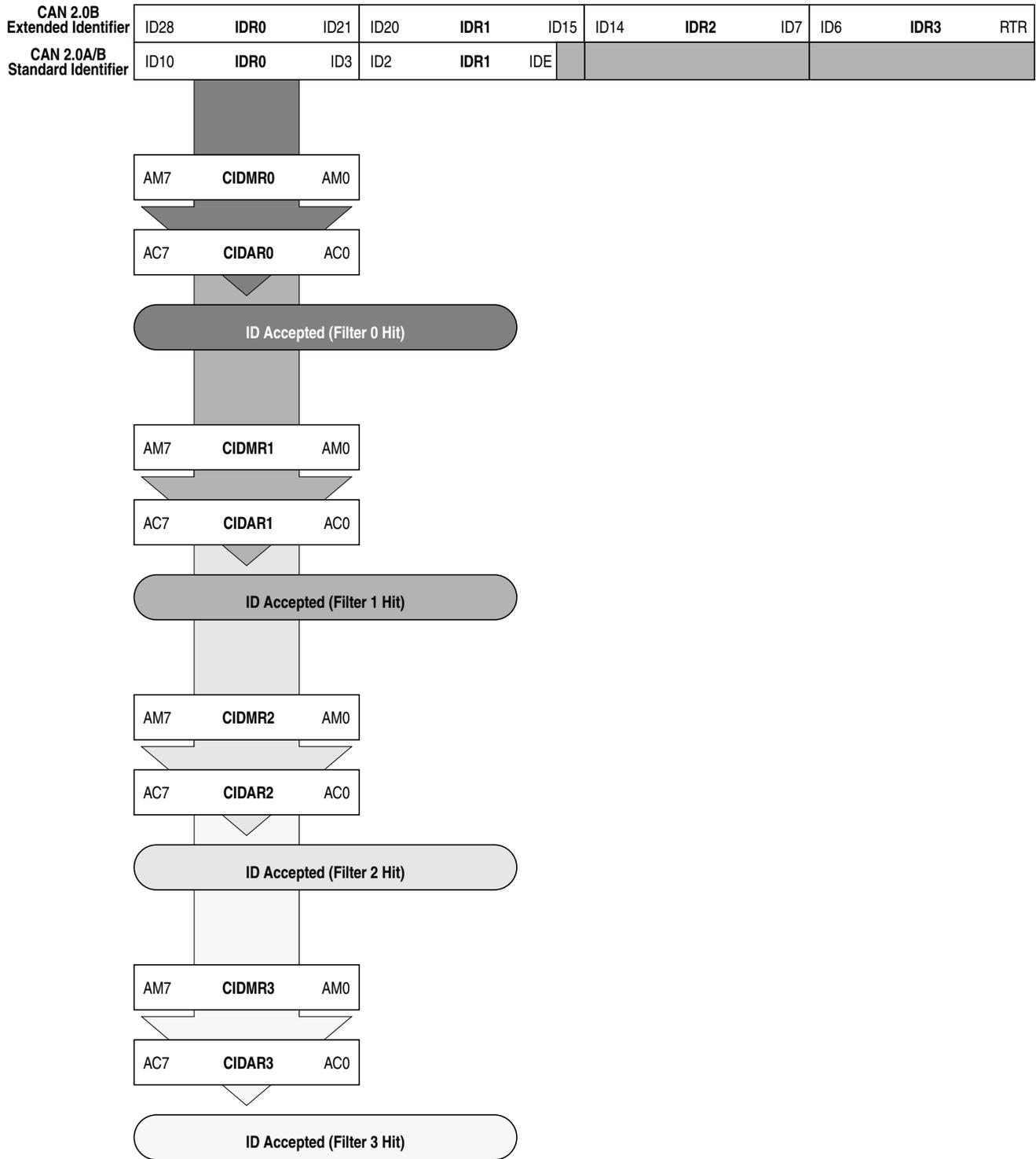


Figure 1-40. 16-bit Maskable Identifier Acceptance Filters



**Figure 1-41. 8-bit Maskable Identifier Acceptance Filters**

### 1.4.3.1 Protocol Violation Protection

The MSCAN protects the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN cannot be modified while the MSCAN is on-line. The MSCAN has to be in Initialization Mode. The corresponding INTRQ/INITAK handshake bits in the CANCTL0/CANCTL1 registers (see Section 1.3.2.1, “MSCAN Control Register 0 (CANCTL0)”) serve as a lock to protect the following registers:
  - MSCAN control 1 register (CANCTL1)
  - MSCAN bus timing registers 0 and 1 (CANBTR0, CANBTR1)
  - MSCAN identifier acceptance control register (CANIDAC)
  - MSCAN identifier acceptance registers (CANIDAR0–CANIDAR7)
  - MSCAN identifier mask registers (CANIDMR0–CANIDMR7)
- The TXCAN pin is immediately forced to a recessive state when the MSCAN goes into the power down mode or initialization mode (see Section 1.4.5.6, “MSCAN Power Down Mode,” and Section 1.4.5.5, “MSCAN Initialization Mode”).
- The MSCAN enable bit (CANE) is writable only once in normal system operation modes, which provides further protection against inadvertently disabling the MSCAN.

### 1.4.3.2 Clock System

Figure 1-44 shows the structure of the MSCAN clock generation circuitry.

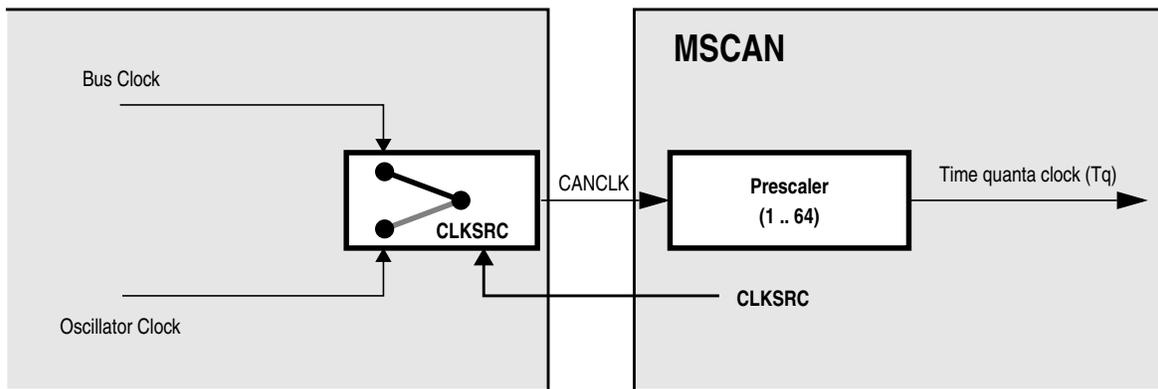


Figure 1-42. MSCAN Clocking Scheme

The clock source bit (CLKSRC) in the CANCTL1 register (1.3.2.2/1-12) defines whether the internal CANCLK is connected to the output of a crystal oscillator (oscillator clock) or to the bus clock.

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met. Additionally, for high CAN bus rates (1 Mbps), a 45% to 55% duty cycle of the clock is required.

If the bus clock is generated from a PLL, it is recommended to select the oscillator clock rather than the bus clock due to jitter considerations, especially at the faster CAN bus rates.

For microcontrollers without a clock and reset generator (CRG), CANCLK is driven from the crystal oscillator (oscillator clock).

A programmable prescaler generates the time quanta ( $T_q$ ) clock from CANCLK. A time quantum is the atomic unit of time handled by the MSCAN.

*Eqn. 1-2*

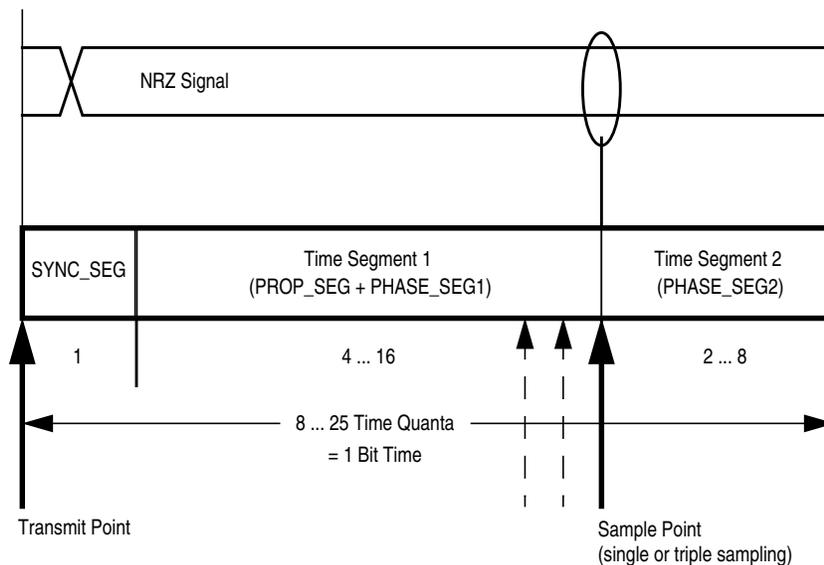
$$T_q = \frac{f_{\text{CANCLK}}}{(\text{Prescaler value})}$$

A bit time is subdivided into three segments as described in the Bosch CAN specification. (see Figure 1-45):

- SYNC\_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time Segment 1: This segment includes the PROP\_SEG and the PHASE\_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time Segment 2: This segment represents the PHASE\_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

*Eqn. 1-3*

$$\text{Bit Rate} = \frac{f_{T_q}}{(\text{number of Time Quanta})}$$



**Figure 1-43. Segments within the Bit Time**

**Table 1-33. Time Segment Syntax**

Syntax	Description
SYNC_SEG	System expects transitions to occur on the CAN bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node in receive mode samples the CAN bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The synchronization jump width (see the Bosch CAN specification for details) can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

The SYNC\_SEG, TSEG1, TSEG2, and SJW parameters are set by programming the MSCAN bus timing registers (CANBTR0, CANBTR1) (see [Section 1.3.2.3, “MSCAN Bus Timing Register 0 \(CANBTR0\)”](#) and [Section 1.3.2.4, “MSCAN Bus Timing Register 1 \(CANBTR1\)”](#)).

[Table 1-36](#) gives an overview of the CAN compliant segment settings and the related parameter values.

#### NOTE

It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard.

**Table 1-34. CAN Standard Compliant Bit Time Segment Settings**

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchronization Jump Width	SJW
5 .. 10	4 .. 9	2	1	1 .. 2	0 .. 1
4 .. 11	3 .. 10	3	2	1 .. 3	0 .. 2
5 .. 12	4 .. 11	4	3	1 .. 4	0 .. 3
6 .. 13	5 .. 12	5	4	1 .. 4	0 .. 3
7 .. 14	6 .. 13	6	5	1 .. 4	0 .. 3
8 .. 15	7 .. 14	7	6	1 .. 4	0 .. 3
9 .. 16	8 .. 15	8	7	1 .. 4	0 .. 3

## 1.4.4 Modes of Operation

### 1.4.4.1 Normal Modes

The MSCAN module behaves as described within this specification in all normal system operation modes.

### 1.4.4.2 Special Modes

The MSCAN module behaves as described within this specification in all special system operation modes.

### 1.4.4.3 Emulation Modes

In all emulation modes, the MSCAN module behaves just like normal system operation modes as described within this specification.

### 1.4.4.4 Listen-Only Mode

In an optional CAN bus monitoring mode (listen-only), the CAN node is able to receive valid data frames and valid remote frames, but it sends only “recessive” bits on the CAN bus. In addition, it cannot start a transmission. If the MAC sub-layer is required to send a “dominant” bit (ACK bit, overload flag, or active error flag), the bit is rerouted internally so that the MAC sub-layer monitors this “dominant” bit, although the CAN bus may remain in recessive state externally.

### 1.4.4.5 Security Modes

The MSCAN module has no security features.

## 1.4.5 Low-Power Options

If the MSCAN is disabled (CANE = 0), the MSCAN clocks are stopped for power saving.

If the MSCAN is enabled (CANE = 1), the MSCAN has two additional modes with reduced power consumption, compared to normal mode: sleep and power down mode. In sleep mode, power consumption is reduced by stopping all clocks except those to access the registers from the CPU side. In power down mode, all clocks are stopped and no power is consumed.

[Table 1-37](#) summarizes the combinations of MSCAN and CPU modes. A particular combination of modes is entered by the given settings on the CSWAI and SLPRQ/SLPAK bits.

For all modes, an MSCAN wake-up interrupt can occur only if the MSCAN is in sleep mode (SLPRQ = 1 and SLPAK = 1), wake-up functionality is enabled (WUPE = 1), and the wake-up interrupt is enabled (WUPIE = 1).

Table 1-35. CPU vs. MSCAN Operating Modes

CPU Mode	MSCAN Mode			
	Normal	Reduced Power Consumption		
		Sleep	Power Down	Disabled (CANE=0)
<b>RUN</b>	CSWAI = X <sup>1</sup> SLPRQ = 0 SLPAK = 0	CSWAI = X SLPRQ = 1 SLPAK = 1		CSWAI = X SLPRQ = X SLPAK = X
<b>WAIT</b>	CSWAI = 0 SLPRQ = 0 SLPAK = 0	CSWAI = 0 SLPRQ = 1 SLPAK = 1	CSWAI = 1 SLPRQ = X SLPAK = X	CSWAI = X SLPRQ = X SLPAK = X
<b>STOP</b>			CSWAI = X SLPRQ = X SLPAK = X	CSWAI = X SLPRQ = X SLPAK = X

<sup>1</sup> 'X' means don't care.

#### 1.4.5.1 Operation in Run Mode

As shown in [Table 1-37](#), only MSCAN sleep mode is available as low power option when the CPU is in run mode.

#### 1.4.5.2 Operation in Wait Mode

The WAI instruction puts the MCU in a low power consumption stand-by mode. If the CSWAI bit is set, additional power can be saved in power down mode because the CPU clocks are stopped. After leaving this power down mode, the MSCAN restarts its internal controllers and enters normal mode again.

While the CPU is in wait mode, the MSCAN can be operated in normal mode and generate interrupts (registers can be accessed via background debug mode). The MSCAN can also operate in any of the low-power modes depending on the values of the SLPRQ/SLPAK and CSWAI bits as seen in [Table 1-37](#).

#### 1.4.5.3 Operation in Stop Mode

The STOP instruction puts the MCU in a low power consumption stand-by mode. In stop mode, the MSCAN is set in power down mode regardless of the value of the SLPRQ/SLPAK and CSWAI bits [Table 1-37](#).

#### 1.4.5.4 MSCAN Sleep Mode

The CPU can request the MSCAN to enter this low power mode by asserting the SLPRQ bit in the CANCTL0 register. The time when the MSCAN enters sleep mode depends on a fixed synchronization delay and its current activity:

- If there are one or more message buffers scheduled for transmission (TXEx = 0), the MSCAN will continue to transmit until all transmit message buffers are empty (TXEx = 1, transmitted successfully or aborted) and then goes into sleep mode.
- If the MSCAN is receiving, it continues to receive and goes into sleep mode as soon as the CAN bus next becomes idle.
- If the MSCAN is neither transmitting nor receiving, it immediately goes into sleep mode.

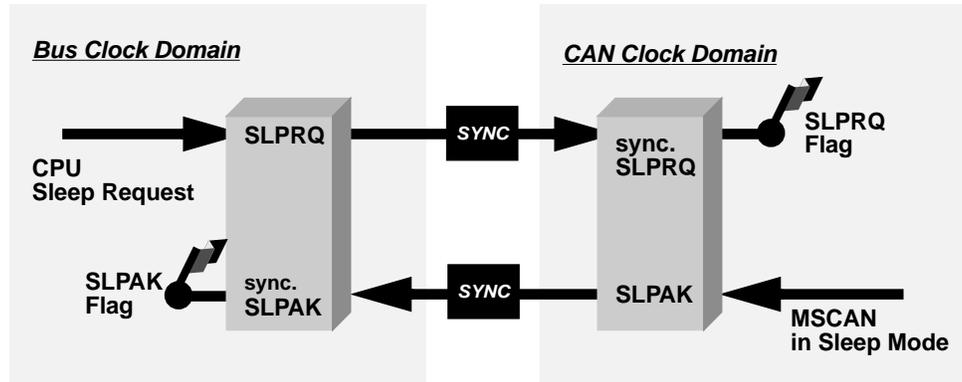


Figure 1-44. Sleep Request / Acknowledge Cycle

#### NOTE

The application software must avoid setting up a transmission (by clearing one or more TXEx flag(s)) and immediately request sleep mode (by setting SLPRQ). Whether the MSCAN starts transmitting or goes into sleep mode directly depends on the exact sequence of operations.

If sleep mode is active, the SLPRQ and SLPK bits are set (Figure 1-46). The application software must use SLPK as a handshake indication for the request (SLPRQ) to go into sleep mode.

When in sleep mode (SLPRQ = 1 and SLPK = 1), the MSCAN stops its internal clocks. However, clocks that allow register accesses from the CPU side continue to run.

If the MSCAN is in bus-off state, it stops counting the 128 occurrences of 11 consecutive recessive bits due to the stopped clocks. The TXCAN pin remains in a recessive state. If RXF = 1, the message can be read and RXF can be cleared. Shifting a new message into the foreground buffer of the receiver FIFO (RxFG) does not take place while in sleep mode.

It is possible to access the transmit buffers and to clear the associated TXE flags. No message abort takes place while in sleep mode.

If the WUPE bit in CANCLT0 is not asserted, the MSCAN will mask any activity it detects on CAN. The RXCAN pin is therefore held internally in a recessive state. This locks the MSCAN in sleep mode (Figure 1-47). WUPE must be set before entering sleep mode to take effect.

The MSCAN is able to leave sleep mode (wake up) only when:

- CAN bus activity occurs and WUPE = 1

or

- the CPU clears the SLPRQ bit

**NOTE**

The CPU cannot clear the SLPRQ bit before sleep mode (SLPRQ = 1 and SLPK = 1) is active.

After wake-up, the MSCAN waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, if the MSCAN is woken-up by a CAN frame, this frame is not received.

The receive message buffers (RxFG and RxBG) contain messages if they were received before sleep mode was entered. All pending actions will be executed upon wake-up; copying of RxBG into RxFG, message aborts and message transmissions. If the MSCAN remains in bus-off state after sleep mode was exited, it continues counting the 128 occurrences of 11 consecutive recessive bits.

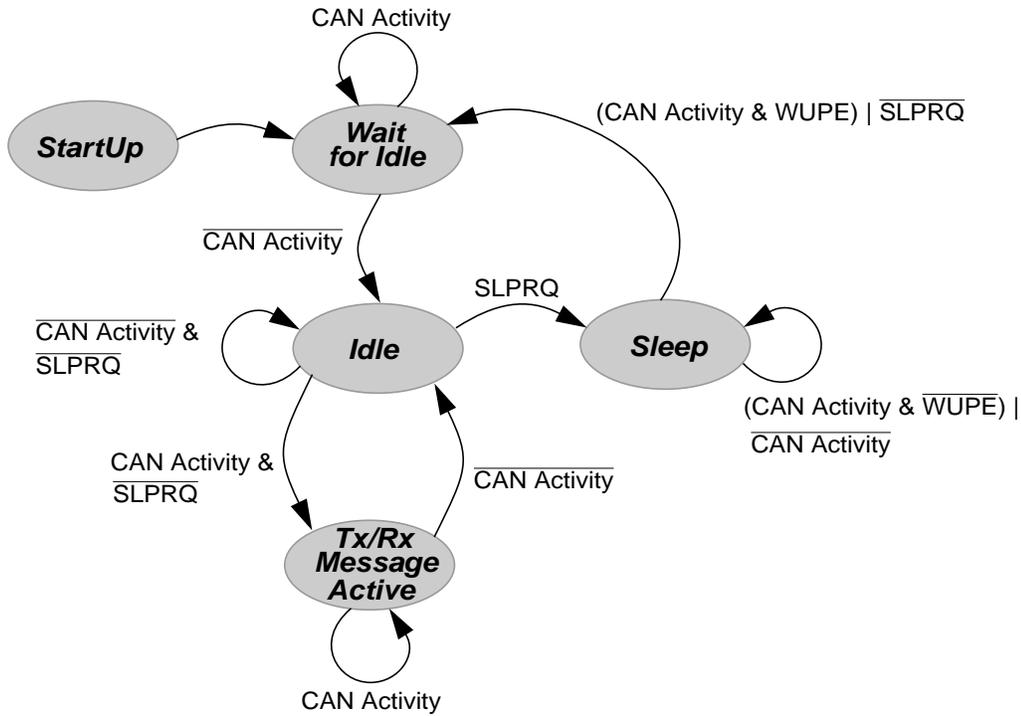


Figure 1-45. Simplified State Transitions for Entering/Leaving Sleep Mode

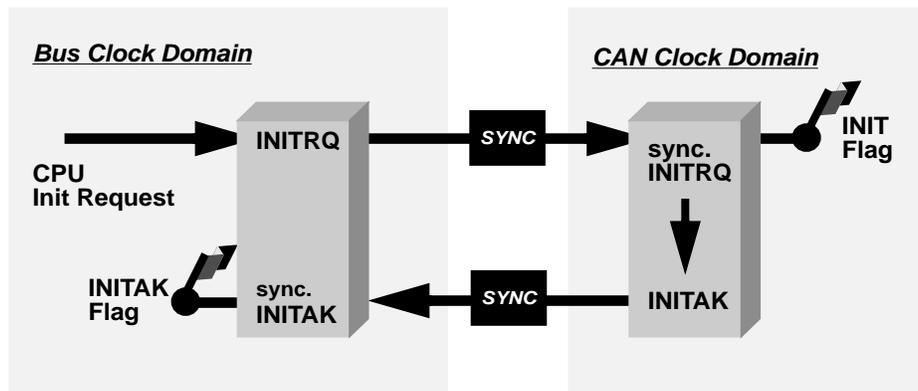
**1.4.5.5 MSCAN Initialization Mode**

In initialization mode, any on-going transmission or reception is immediately aborted and synchronization to the CAN bus is lost, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations, the MSCAN immediately drives the TXCAN pin into a recessive state.

**NOTE**

The user is responsible for ensuring that the MSCAN is not active when initialization mode is entered. The recommended procedure is to bring the MSCAN into sleep mode (SLPRQ = 1 and SLPK = 1) before setting the INITRQ bit in the CANCTL0 register. Otherwise, the abort of an on-going message can cause an error condition and can impact other CAN bus devices.

In initialization mode, the MSCAN is stopped. However, interface registers remain accessible. This mode is used to reset the CANCTL0, CANRFLG, CANRIER, CANTFLG, CANTIER, CANTARQ, CANTAACK, and CANTBSEL registers to their default values. In addition, the MSCAN enables the configuration of the CANBTR0, CANBTR1 bit timing registers; CANIDAC; and the CANIDAR, CANIDMR message filters. See Section 1.3.2.1, “MSCAN Control Register 0 (CANCTL0),” for a detailed description of the initialization mode.



**Figure 1-46. Initialization Request/Acknowledge Cycle**

Due to independent clock domains within the MSCAN, INITRQ must be synchronized to all domains by using a special handshake mechanism. This handshake causes additional synchronization delay (see Section Figure 1-48., “Initialization Request/Acknowledge Cycle”).

If there is no message transfer ongoing on the CAN bus, the minimum delay will be two additional bus clocks and three additional CAN clocks. When all parts of the MSCAN are in initialization mode, the INITAK flag is set. The application software must use INITAK as a handshake indication for the request (INITRQ) to go into initialization mode.

**NOTE**

The CPU cannot clear INITRQ before initialization mode (INITRQ = 1 and INITAK = 1) is active.

### 1.4.5.6 MSCAN Power Down Mode

The MSCAN is in power down mode (Table 1-37) when

- CPU is in stop mode

or

- CPU is in wait mode and the CSWAI bit is set

When entering the power down mode, the MSCAN immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations to the above rule, the MSCAN immediately drives the TXCAN pin into a recessive state.

#### NOTE

The user is responsible for ensuring that the MSCAN is not active when power down mode is entered. The recommended procedure is to bring the MSCAN into Sleep mode before the STOP or WAI instruction (if CSWAI is set) is executed. Otherwise, the abort of an ongoing message can cause an error condition and impact other CAN bus devices.

In power down mode, all clocks are stopped and no registers can be accessed. If the MSCAN was not in sleep mode before power down mode became active, the module performs an internal recovery cycle after powering up. This causes some fixed delay before the module enters normal mode again.

### 1.4.5.7 Programmable Wake-Up Function

The MSCAN can be programmed to wake up the MSCAN as soon as CAN bus activity is detected (see control bit WUPE in [Section 1.3.2.1, “MSCAN Control Register 0 \(CANCTL0\)”](#)). The sensitivity to existing CAN bus action can be modified by applying a low-pass filter function to the RXCAN input line while in sleep mode (see control bit WUPM in [Section 1.3.2.2, “MSCAN Control Register 1 \(CANCTL1\)”](#)).

This feature can be used to protect the MSCAN from wake-up due to short glitches on the CAN bus lines. Such glitches can result from—for example—electromagnetic interference within noisy environments.

### 1.4.6 Reset Initialization

The reset state of each individual bit is listed in [Section 1.3.2, “Register Descriptions,”](#) which details all the registers and their bit-fields.

### 1.4.7 Interrupts

This section describes all interrupts originated by the MSCAN. It documents the enable bits and generated flags. Each interrupt is listed and described separately.

#### 1.4.7.1 Description of Interrupt Operation

The MSCAN supports four interrupt vectors (see [Table 1-38](#)), any of which can be individually masked (for details see sections from [Section 1.3.2.6, “MSCAN Receiver Interrupt Enable Register \(CANRIER\),”](#) to [Section 1.3.2.8, “MSCAN Transmitter Interrupt Enable Register \(CANTIER\)”](#)).

**NOTE**

The dedicated interrupt vector addresses are defined in the [Resets and Interrupts](#) chapter.

**Table 1-36. Interrupt Vectors**

Interrupt Source	CCR Mask	Local Enable
Wake-Up Interrupt (WUPIF)	1 bit	CANRIER (WUPIE)
Error Interrupts Interrupt (CSCIF, OVRIF)	1 bit	CANRIER (CSCIE, OVRIE)
Receive Interrupt (RXF)	1 bit	CANRIER (RXFIE)
Transmit Interrupts (TXE[2:0])	1 bit	CANTIER (TXEIE[2:0])

**1.4.7.2 Transmit Interrupt**

At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXEx flag of the empty message buffer is set.

**1.4.7.3 Receive Interrupt**

A message is successfully received and shifted into the foreground buffer (RxFG) of the receiver FIFO. This interrupt is generated immediately after receiving the EOF symbol. The RXF flag is set. If there are multiple messages in the receiver FIFO, the RXF flag is set as soon as the next message is shifted to the foreground buffer.

**1.4.7.4 Wake-Up Interrupt**

A wake-up interrupt is generated if activity on the CAN bus occurs during MSCAN internal sleep mode. WUPE (see [Section 1.3.2.1, “MSCAN Control Register 0 \(CANCTL0\)”](#)) must be enabled.

**1.4.7.5 Error Interrupt**

An error interrupt is generated if an overrun of the receiver FIFO, error, warning, or bus-off condition occurs. [Section 1.3.2.5, “MSCAN Receiver Flag Register \(CANRFLG\)”](#) indicates one of the following conditions:

- **Overrun** — An overrun condition of the receiver FIFO as described in [Section 1.4.2.3, “Receive Structures,”](#) occurred.
- **CAN Status Change** — The actual value of the transmit and receive error counters control the CAN bus state of the MSCAN. As soon as the error counters skip into a critical range (Tx/Rx-warning, Tx/Rx-error, bus-off) the MSCAN flags an error condition. The status change, which caused the error condition, is indicated by the TSTAT and RSTAT flags (see [Section 1.3.2.5, “MSCAN Receiver Flag Register \(CANRFLG\)”](#) and [Section 1.3.2.6, “MSCAN Receiver Interrupt Enable Register \(CANRIER\)”](#)).

**1.4.7.6 Interrupt Acknowledge**

Interrupts are directly associated with one or more status flags in either the [Section 1.3.2.5, “MSCAN Receiver Flag Register \(CANRFLG\)”](#) or the [Section 1.3.2.7, “MSCAN Transmitter Flag Register](#)

(CANTFLG).” Interrupts are pending as long as one of the corresponding flags is set. The flags in CANRFLG and CANTFLG must be reset within the interrupt handler to handshake the interrupt. The flags are reset by writing a 1 to the corresponding bit position. A flag cannot be cleared if the respective condition prevails.

#### NOTE

It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

#### 1.4.7.7 Recovery from Stop or Wait

The MSCAN can recover from stop or wait via the wake-up interrupt. This interrupt can only occur if the MSCAN was in sleep mode (SLPRQ = 1 and SLPK = 1) before entering power down mode, the wake-up option is enabled (WUPE = 1), and the wake-up interrupt is enabled (WUPIE = 1).

## 1.5 Initialization/Application Information

### 1.5.1 MSCAN initialization

The procedure to initially start up the MSCAN module out of reset is as follows:

1. Assert CANE
2. Write to the configuration registers in initialization mode
3. Clear INITRQ to leave initialization mode and enter normal mode

If the configuration of registers which are writable in initialization mode needs to be changed only when the MSCAN module is in normal mode:

1. Bring the module into sleep mode by setting SLPRQ and awaiting SLPK to assert after the CAN bus becomes idle.
2. Enter initialization mode: assert INITRQ and await INITAK
3. Write to the configuration registers in initialization mode
4. Clear INITRQ to leave initialization mode and continue in normal mode