– Wakeup condition, SOF monitored on the CAN Rx signal.
– Entry into Sleep mode.

## 24.9 CAN registers

The peripheral registers have to be accessed by words (32 bits).

### 24.9.1 Register access protection

Erroneous access to certain configuration registers can cause the hardware to temporarily disturb the whole CAN network. Therefore the CAN_BTR register can be modified by software only while the CAN hardware is in initialization mode.

Although the transmission of incorrect data will not cause problems at the CAN network level, it can severely disturb the application. A transmit mailbox can be only modified by software while it is in empty state, refer to *Figure 228*.

The filter values can be modified either deactivating the associated filter banks or by setting the FINIT bit. Moreover, the modification of the filter configuration (scale, mode and FIFO assignment) in CAN_FMxR, CAN_FSxR and CAN_FFAR registers can only be done when the filter initialization mode is set (FINIT=1) in the CAN_FMR register.

### 24.9.2 CAN control and status registers

Refer to *Section 2.2 on page 45* for a list of abbreviations used in register descriptions.

#### CAN master control register (CAN_MCR)

Address offset: 0x00
Reset value: 0x0001 0002

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | DBF |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESET | Reserved | | | | | | | TTCM | ABOM | AWUM | NART | RFLM | TXFP | SLEEP | INRQ |
| rs | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DBF:** Debug freeze
0: CAN working during debug
1: CAN reception/transmission frozen during debug. Reception FIFOs can still be accessed/controlled normally.

Bit 15 **RESET:** bxCAN software master reset
0: Normal operation.
1: Force a master reset of the bxCAN -> Sleep mode activated after reset (FMP bits and CAN_MCR register are initialized to the reset values). This bit is automatically reset to 0.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **TTCM**: Time triggered communication mode

0: Time Triggered Communication mode disabled.
1: Time Triggered Communication mode enabled

*Note: For more information on Time Triggered Communication mode refer to Section 24.7.2.*

Bit 6 **ABOM:** Automatic bus-off management

This bit controls the behavior of the CAN hardware on leaving the Bus-Off state.
0: The Bus-Off state is left on software request, once 128 occurrences of 11 recessive bits have been monitored and the software has first set and cleared the INRQ bit of the CAN_MCR register.
1: The Bus-Off state is left automatically by hardware once 128 occurrences of 11 recessive bits have been monitored.
For detailed information on the Bus-Off state refer to *Section 24.7.6*.

Bit 5 **AWUM**: Automatic wakeup mode

This bit controls the behavior of the CAN hardware on message reception during Sleep mode.
0: The Sleep mode is left on software request by clearing the SLEEP bit of the CAN_MCR register.
1: The Sleep mode is left automatically by hardware on CAN message detection.
The SLEEP bit of the CAN_MCR register and the SLAK bit of the CAN_MSR register are cleared by hardware.

Bit 4 **NART**: No automatic retransmission

0: The CAN hardware will automatically retransmit the message until it has been successfully transmitted according to the CAN standard.
1: A message will be transmitted only once, independently of the transmission result (successful, error or arbitration lost).

Bit 3 **RFLM**: Receive FIFO locked mode

0: Receive FIFO not locked on overrun. Once a receive FIFO is full the next incoming message will overwrite the previous one.
1: Receive FIFO locked against overrun. Once a receive FIFO is full the next incoming message will be discarded.

Bit 2 **TXFP**: Transmit FIFO priority

This bit controls the transmission order when several mailboxes are pending at the same time.
0: Priority driven by the identifier of the message
1: Priority driven by the request order (chronologically)

Bit 1 **SLEEP**: Sleep mode request

This bit is set by software to request the CAN hardware to enter the Sleep mode. Sleep mode will be entered as soon as the current CAN activity (transmission or reception of a CAN frame) has been completed.
This bit is cleared by software to exit Sleep mode.
This bit is cleared by hardware when the AWUM bit is set and a SOF bit is detected on the CAN Rx signal.
This bit is set after reset - CAN starts in Sleep mode.

Bit 0 **INRQ**: Initialization request

The software clears this bit to switch the hardware into normal mode. Once 11 consecutive recessive bits have been monitored on the Rx signal the CAN hardware is synchronized and ready for transmission and reception. Hardware signals this event by clearing the INAK bit in the CAN_MSR register.
Software sets this bit to request the CAN hardware to enter initialization mode. Once software has set the INRQ bit, the CAN hardware waits until the current CAN activity (transmission or reception) is completed before entering the initialization mode. Hardware signals this event by setting the INAK bit in the CAN_MSR register.

### CAN master status register (CAN_MSR)

Address offset: 0x04
Reset value: 0x0000 0C02

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved. | | | | RX | SAMP | RXM | TXM | Reserved | | | SLAKI | WKUI | ERRI | SLAK | INAK |
| | | | | r | r | r | r | | | | rc_w1 | rc_w1 | rc_w1 | r | r |

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RX**: CAN Rx signal

Monitors the actual value of the **CAN_RX** Pin.

Bit 10 **SAMP**: Last sample point

The value of RX on the last sample point (current received bit value).

Bit 9 **RXM**: Receive mode

The CAN hardware is currently receiver.

Bit 8 **TXM**: Transmit mode

The CAN hardware is currently transmitter.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SLAKI**: Sleep acknowledge interrupt

When SLKIE=1, this bit is set by hardware to signal that the bxCAN has entered Sleep Mode. When set, this bit generates a status change interrupt if the SLKIE bit in the CAN_IER register is set.
This bit is cleared by software or by hardware, when SLAK is cleared.

*Note: When SLKIE=0, no polling on SLAKI is possible. In this case the SLAK bit can be polled.*

Bit 3 **WKUI**: Wakeup interrupt

This bit is set by hardware to signal that a SOF bit has been detected while the CAN hardware was in Sleep mode. Setting this bit generates a status change interrupt if the WKUIE bit in the CAN_IER register is set.
This bit is cleared by software.

Bit 2 **ERRI**: Error interrupt

This bit is set by hardware when a bit of the CAN_ESR has been set on error detection and the corresponding interrupt in the CAN_IER is enabled. Setting this bit generates a status change interrupt if the ERRIE bit in the CAN_IER register is set.
This bit is cleared by software.

Bit 1 **SLAK**: Sleep acknowledge

This bit is set by hardware and indicates to the software that the CAN hardware is now in Sleep mode. This bit acknowledges the Sleep mode request from the software (set SLEEP bit in CAN_MCR register).
This bit is cleared by hardware when the CAN hardware has left Sleep mode (to be synchronized on the CAN bus). To be synchronized the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.

*Note: The process of leaving Sleep mode is triggered when the SLEEP bit in the CAN_MCR register is cleared. Refer to the AWUM bit of the CAN_MCR register description for detailed information for clearing SLEEP bit*

Bit 0 **INAK**: Initialization acknowledge

This bit is set by hardware and indicates to the software that the CAN hardware is now in initialization mode. This bit acknowledges the initialization request from the software (set INRQ bit in CAN_MCR register).
This bit is cleared by hardware when the CAN hardware has left the initialization mode (to be synchronized on the CAN bus). To be synchronized the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.

### CAN transmit status register (CAN_TSR)

Address offset: 0x08
Reset value: 0x1C00 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOW2 | LOW1 | LOW0 | TME2 | TME1 | TME0 | CODE[1:0] | | ABRQ2 | Reserved | | | TERR2 | ALST2 | TXOK2 | RQCP2 |
| r | r | r | r | r | r | r | r | rs | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ABRQ1 | Reserved Res. | | | TERR1 | ALST1 | TXOK1 | RQCP1 | ABRQ0 | Reserved | | | TERR0 | ALST0 | TXOK0 | RQCP0 |
| rs | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rs | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

Bit 31 **LOW2**: Lowest priority flag for mailbox 2

This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 2 has the lowest priority.

Bit 30 **LOW1**: Lowest priority flag for mailbox 1

This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 1 has the lowest priority.

Bit 29 **LOW0**: Lowest priority flag for mailbox 0

This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 0 has the lowest priority.

*Note: The LOW[2:0] bits are set to zero when only one mailbox is pending.*

Bit 28 **TME2**: Transmit mailbox 2 empty

This bit is set by hardware when no transmit request is pending for mailbox 2.

Bit 27 **TME1**: Transmit mailbox 1 empty

This bit is set by hardware when no transmit request is pending for mailbox 1.

Bit 26 **TME0**: Transmit mailbox 0 empty
This bit is set by hardware when no transmit request is pending for mailbox 0.

Bits 25:24 **CODE[1:0]**: Mailbox code
In case at least one transmit mailbox is free, the code value is equal to the number of the next transmit mailbox free.
In case all transmit mailboxes are pending, the code value is equal to the number of the transmit mailbox with the lowest priority.

Bit 23 **ABRQ2**: Abort request for mailbox 2
Set by software to abort the transmission request for the corresponding mailbox.
Cleared by hardware when the mailbox becomes empty.
Setting this bit has no effect when the mailbox is not pending for transmission.

Bits 22:20 Reserved, must be kept at reset value.

Bit 19 **TERR2**: Transmission error of mailbox 2
This bit is set when the previous TX failed due to an error.

Bit 18 **ALST2**: Arbitration lost for mailbox 2
This bit is set when the previous TX failed due to an arbitration lost.

Bit 17 **TXOK2**: Transmission OK of mailbox 2
The hardware updates this bit after each transmission attempt.
0: The previous transmission failed
1: The previous transmission was successful
This bit is set by hardware when the transmission request on mailbox 2 has been completed successfully. Refer to *Figure 228*.

Bit 16 **RQCP2**: Request completed mailbox2
Set by hardware when the last request (transmit or abort) has been performed.
Cleared by software writing a “1” or by hardware on transmission request (TXRQ2 set in CAN_TMID2R register).
Clearing this bit clears all the status bits (TXOK2, ALST2 and TERR2) for Mailbox 2.

Bit 15 **ABRQ1**: Abort request for mailbox 1
Set by software to abort the transmission request for the corresponding mailbox.
Cleared by hardware when the mailbox becomes empty.
Setting this bit has no effect when the mailbox is not pending for transmission.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **TERR1**: Transmission error of mailbox1
This bit is set when the previous TX failed due to an error.

Bit 10 **ALST1**: Arbitration lost for mailbox1
This bit is set when the previous TX failed due to an arbitration lost.

Bit 9 **TXOK1**: Transmission OK of mailbox1
The hardware updates this bit after each transmission attempt.
0: The previous transmission failed
1: The previous transmission was successful
This bit is set by hardware when the transmission request on mailbox 1 has been completed successfully. Refer to *Figure 228*

Bit 8 **RQCP1**: Request completed mailbox1
Set by hardware when the last request (transmit or abort) has been performed.
Cleared by software writing a “1” or by hardware on transmission request (TXRQ1 set in CAN_TI1R register).
Clearing this bit clears all the status bits (TXOK1, ALST1 and TERR1) for Mailbox 1.

Bit 7 **ABRQ0**: Abort request for mailbox0

Set by software to abort the transmission request for the corresponding mailbox.
Cleared by hardware when the mailbox becomes empty.
Setting this bit has no effect when the mailbox is not pending for transmission.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **TERR0**: Transmission error of mailbox0

This bit is set when the previous TX failed due to an error.

Bit 2 **ALST0**: Arbitration lost for mailbox0

This bit is set when the previous TX failed due to an arbitration lost.

Bit 1 **TXOK0**: Transmission OK of mailbox0

The hardware updates this bit after each transmission attempt.
0: The previous transmission failed
1: The previous transmission was successful
This bit is set by hardware when the transmission request on mailbox 1 has been completed successfully. Refer to *Figure 228*

Bit 0 **RQCP0**: Request completed mailbox0

Set by hardware when the last request (transmit or abort) has been performed.
Cleared by software writing a “1” or by hardware on transmission request (TXRQ0 set in CAN_TI0R register).
Clearing this bit clears all the status bits (TXOK0, ALST0 and TERR0) for Mailbox 0.

### CAN receive FIFO 0 register (CAN_RF0R)

Address offset: 0x0C
Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | RFOM0 | FOVR0 | FULL0 | Res. | FMP0[1:0] | |
| | | | | | | | | | | rs | rc_w1 | rc_w1 | | r | r |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **RFOM0**: Release FIFO 0 output mailbox

Set by software to release the output mailbox of the FIFO. The output mailbox can only be released when at least one message is pending in the FIFO. Setting this bit when the FIFO is empty has no effect. If at least two messages are pending in the FIFO, the software has to release the output mailbox to access the next message.
Cleared by hardware when the output mailbox has been released.

Bit 4 **FOVR0**: FIFO 0 overrun

This bit is set by hardware when a new message has been received and passed the filter while the FIFO was full.
This bit is cleared by software.

Bit 3 **FULL0**: FIFO 0 full

Set by hardware when three messages are stored in the FIFO.
This bit is cleared by software.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FMP0[1:0]**: FIFO 0 message pending

These bits indicate how many messages are pending in the receive FIFO.
FMP is increased each time the hardware stores a new message in to the FIFO. FMP is decreased each time the software releases the output mailbox by setting the RFOM0 bit.

### CAN receive FIFO 1 register (CAN_RF1R)

Address offset: 0x10
Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | RFOM1 | FOVR1 | FULL1 | Res. | FMP1[1:0] | |
| | | | | | | | | | | rs | rc_w1 | rc_w1 | | r | r |

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **RFOM1**: Release FIFO 1 output mailbox

Set by software to release the output mailbox of the FIFO. The output mailbox can only be released when at least one message is pending in the FIFO. Setting this bit when the FIFO is empty has no effect. If at least two messages are pending in the FIFO, the software has to release the output mailbox to access the next message.
Cleared by hardware when the output mailbox has been released.

Bit 4 **FOVR1**: FIFO 1 overrun

This bit is set by hardware when a new message has been received and passed the filter while the FIFO was full.
This bit is cleared by software.

Bit 3 **FULL1**: FIFO 1 full

Set by hardware when three messages are stored in the FIFO.
This bit is cleared by software.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FMP1[1:0]**: FIFO 1 message pending

These bits indicate how many messages are pending in the receive FIFO1.
FMP1 is increased each time the hardware stores a new message in to the FIFO1. FMP is decreased each time the software releases the output mailbox by setting the RFOM1 bit.

### CAN interrupt enable register (CAN_IER)

Address offset: 0x14
Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | SLKIE | WKUIE |
| | | | | | | | | | | | | | | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ERRIE | Reserved | | | LEC IE | BOF IE | EPV IE | EWG IE | Res. | FOV IE1 | FF IE1 | FMP IE1 | FOV IE0 | FF IE0 | FMP IE0 | TME IE |
| rw | | | | rw | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw |

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **SLKIE**: Sleep interrupt enable
0: No interrupt when SLAKI bit is set.
1: Interrupt generated when SLAKI bit is set.

Bit 16 **WKUIE**: Wakeup interrupt enable
0: No interrupt when WKUI is set.
1: Interrupt generated when WKUI bit is set.

Bit 15 **ERRIE**: Error interrupt enable
0: No interrupt will be generated when an error condition is pending in the CAN_ESR.
1: An interrupt will be generation when an error condition is pending in the CAN_ESR.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **LECIE**: Last error code interrupt enable
0: ERRI bit will not be set when the error code in LEC[2:0] is set by hardware on error detection.
1: ERRI bit will be set when the error code in LEC[2:0] is set by hardware on error detection.

Bit 10 **BOFIE**: Bus-off interrupt enable
0: ERRI bit will not be set when BOFF is set.
1: ERRI bit will be set when BOFF is set.

Bit 9 **EPVIE**: Error passive interrupt enable
0: ERRI bit will not be set when EPVF is set.
1: ERRI bit will be set when EPVF is set.

Bit 8 **EWGIE**: Error warning interrupt enable
0: ERRI bit will not be set when EWGF is set.
1: ERRI bit will be set when EWGF is set.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FOVIE1**: FIFO overrun interrupt enable
0: No interrupt when FOVR is set.
1: Interrupt generation when FOVR is set.

Bit 5 **FFIE1**: FIFO full interrupt enable
0: No interrupt when FULL bit is set.
1: Interrupt generated when FULL bit is set.

Bit 4 **FMPIE1**: FIFO message pending interrupt enable
0: No interrupt generated when state of FMP[1:0] bits are not 00b.
1: Interrupt generated when state of FMP[1:0] bits are not 00b.

Bit 3 **FOVIE0**: FIFO overrun interrupt enable
0: No interrupt when FOVR bit is set.
1: Interrupt generated when FOVR bit is set.

Bit 2 **FFIE0**: FIFO full interrupt enable
0: No interrupt when FULL bit is set.
1: Interrupt generated when FULL bit is set.

Bit 1 **FMPIE0**: FIFO message pending interrupt enable
0: No interrupt generated when state of FMP[1:0] bits are not 00b.
1: Interrupt generated when state of FMP[1:0] bits are not 00b.

Bit 0 **TMEIE**: Transmit mailbox empty interrupt enable
0: No interrupt when RQCPx bit is set.
1: Interrupt generated when RQCPx bit is set.
*Note: Refer to Section 24.8: bxCAN interrupts.*

### CAN error status register (CAN_ESR)

Address offset: 0x18
Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REC[7:0] | | | | | | | | | TEC[7:0] | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | LEC[2:0] | | | Res. | BOFF | EPVF | EWGF |
| | | | | | | | | | rw | rw | rw | | r | r | r |

Bits 31:24 **REC[7:0]**: Receive error counter
The implementing part of the fault confinement mechanism of the CAN protocol. In case of an error during reception, this counter is incremented by 1 or by 8 depending on the error condition as defined by the CAN standard. After every successful reception the counter is decremented by 1 or reset to 120 if its value was higher than 128. When the counter value exceeds 127, the CAN controller enters the error passive state.

Bits 23:16 **TEC[7:0]**: Least significant byte of the 9-bit transmit error counter
The implementing part of the fault confinement mechanism of the CAN protocol.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **LEC[2:0]**: Last error code
This field is set by hardware and holds a code which indicates the error condition of the last error detected on the CAN bus. If a message has been transferred (reception or transmission) without error, this field will be cleared to ‘0’.
The LEC[2:0] bits can be set to value 0b111 by software. They are updated by hardware to indicate the current communication status.
000: No Error
001: Stuff Error
010: Form Error
011: Acknowledgment Error
100: Bit recessive Error
101: Bit dominant Error
110: CRC Error
111: Set by software

Bit 3 Reserved, must be kept at reset value.

Bit 2 **BOFF**: Bus-off flag

This bit is set by hardware when it enters the bus-off state. The bus-off state is entered on TEC overflow, greater than 255, refer to *Section 24.7.6*.

Bit 1 **EPVF**: Error passive flag

This bit is set by hardware when the Error Passive limit has been reached (Receive Error Counter or Transmit Error Counter>127).

Bit 0 **EWGF**: Error warning flag

This bit is set by hardware when the warning limit has been reached (Receive Error Counter or Transmit Error Counter≥96).

### CAN bit timing register (CAN_BTR)

Address offset: 0x1C
Reset value: 0x0123 0000

This register can only be accessed by the software when the CAN hardware is in initialization mode.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SILM | LBKM | Reserved | | | | SJW[1:0] | | Res. | TS2[2:0] | | | TS1[3:0] | | | |
| rw | rw | | | | | rw | rw | | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | BRP[9:0] | | | | | | | | | |
| | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **SILM**: Silent mode (debug)

0: Normal operation
1: Silent Mode

Bit 30 **LBKM**: Loop back mode (debug)

0: Loop Back Mode disabled
1: Loop Back Mode enabled

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **SJW[1:0]**: Resynchronization jump width

These bits define the maximum number of time quanta the CAN hardware is allowed to lengthen or shorten a bit to perform the resynchronization.
$t_{RJW} = t_q \times (SJW[1:0] + 1)$

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **TS2[2:0]**: Time segment 2

These bits define the number of time quanta in Time Segment 2.
$t_{BS2} = t_q \times (TS2[2:0] + 1)$

Bits 19:16 **TS1[3:0]**: Time segment 1
These bits define the number of time quanta in Time Segment 1
$t_{BS1} = t_q \times (TS1[3:0] + 1)$
For more information on bit timing refer to *Section 24.7.7*.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **BRP[9:0]**: Baud rate prescaler
These bits define the length of a time quanta.
$t_q = (BRP[9:0]+1) \times t_{PCLK}$

### 24.9.3 CAN mailbox registers

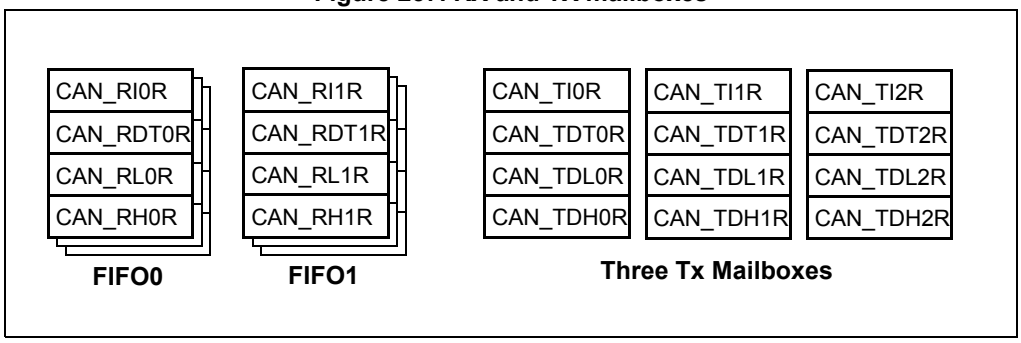
This chapter describes the registers of the transmit and receive mailboxes. Refer to *Section 24.7.5: Message storage* for detailed register mapping.

Transmit and receive mailboxes have the same registers except:

- The FMI field in the CAN_RDTxR register.
- A receive mailbox is always write protected.
- A transmit mailbox is write-enabled only while empty, corresponding TME bit in the CAN_TSR register set.

There are three TX Mailboxes and two RX Mailboxes , as shown in *Figure 237*. Each RX Mailbox allows access to a 3-level depth FIFO, the access being offered only to the oldest received message in the FIFO. Each mailbox consist of four registers.

**Figure 237. RX and TX mailboxes**



### CAN TX mailbox identifier register (CAN_TIxR) (x=0..2)

Address offsets: 0x180, 0x190, 0x1A0
Reset value: 0xXXXX XXXX (except bit 0, TXRQ = 0)

All TX registers are write protected when the mailbox is pending transmission (TMEx reset).

This register also implements the TX request control (bit 0) - reset value 0.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STID[10:0]/EXID[28:18] | | | | | | | | | | | EXID[17:13] | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EXID[12:0] | | | | | | | | | | | | | IDE | RTR | TXRQ |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:21 **STID[10:0]/EXID[28:18]**: Standard identifier or extended identifier
The standard identifier or the MSBs of the extended identifier (depending on the IDE bit value).

Bits 20:3 **EXID[17:0]**: Extended identifier
The LSBs of the extended identifier.

Bit 2 **IDE**: Identifier extension
This bit defines the identifier type of message in the mailbox.
0: Standard identifier.
1: Extended identifier.

Bit 1 **RTR**: Remote transmission request
0: Data frame
1: Remote frame

Bit 0 **TXRQ**: Transmit mailbox request
Set by software to request the transmission for the corresponding mailbox.
Cleared by hardware when the mailbox becomes empty.

### CAN mailbox data length control and time stamp register (CAN_TDTxR) (x=0..2)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x184, 0x194, 0x1A4
Reset value: 0xXXXX XXXX

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIME[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | TGT | Reserved | | | | DLC[3:0] | | | |
| | | | | | | | rw | | | | | rw | rw | rw | rw |

Bits 31:16 **TIME[15:0]**: Message time stamp
This field contains the 16-bit timer value captured at the SOF transmission.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TGT**: Transmit global time
This bit is active only when the hardware is in the Time Trigger Communication mode, TTCM bit of the CAN_MCR register is set.
0: Time stamp TIME[15:0] is not sent.
1: Time stamp TIME[15:0] value is sent in the last two data bytes of the 8-byte message: TIME[7:0] in data byte 7 and TIME[15:8] in data byte 6, replacing the data written in CAN_TDHxR[31:16] register (DATA6[7:0] and DATA7[7:0]). DLC must be programmed as 8 in order these two bytes to be sent over the CAN bus.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **DLC[3:0]**: Data length code
This field defines the number of data bytes a data frame contains or a remote frame request. A message can contain from 0 to 8 data bytes, depending on the value in the DLC field.

### CAN mailbox data low register (CAN_TDLxR) (x=0..2)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x188, 0x198, 0x1A8
Reset value: 0xXXXX XXXX

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA3[7:0] | | | | | | | | DATA2[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA1[7:0] | | | | | | | | DATA0[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:24 **DATA3[7:0]**: Data byte 3
Data byte 3 of the message.

Bits 23:16 **DATA2[7:0]**: Data byte 2
Data byte 2 of the message.

Bits 15:8 **DATA1[7:0]**: Data byte 1
Data byte 1 of the message.

Bits 7:0 **DATA0[7:0]**: Data byte 0
Data byte 0 of the message.
A message can contain from 0 to 8 data bytes and starts with byte 0.

### CAN mailbox data high register (CAN_TDHxR) (x=0..2)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x18C, 0x19C, 0x1AC
Reset value: 0xXXXX XXXX

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA7[7:0] | | | | | | | | DATA6[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA5[7:0] | | | | | | | | DATA4[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:24 **DATA7[7:0]**: Data byte 7
Data byte 7 of the message.
*Note: **If** TGT of this message and TTCM are active, DATA7 and DATA6 will be replaced by the TIME stamp value.*

Bits 23:16 **DATA6[7:0]**: Data byte 6
Data byte 6 of the message.

Bits 15:8 **DATA5[7:0]**: Data byte 5
Data byte 5 of the message.

Bits 7:0 **DATA4[7:0]**: Data byte 4
Data byte 4 of the message.

### CAN receive FIFO mailbox identifier register (CAN_RIxR) (x=0..1)

Address offsets: 0x1B0, 0x1C0
Reset value: 0xXXXX XXXX

All RX registers are write protected.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STID[10:0]/EXID[28:18] | | | | | | | | | | | EXID[17:13] | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EXID[12:0] | | | | | | | | | | | | | IDE | RTR | Res. |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | |

Bits 31:21 **STID[10:0]/EXID[28:18]**: Standard identifier or extended identifier
The standard identifier or the MSBs of the extended identifier (depending on the IDE bit value).

Bits 20:3 **EXID[17:0]**: Extended identifier
The LSBs of the extended identifier.

Bit 2 **IDE**: Identifier extension
This bit defines the identifier type of message in the mailbox.
0: Standard identifier.
1: Extended identifier.

Bit 1 **RTR**: Remote transmission request
0: Data frame
1: Remote frame

Bit 0 Reserved, must be kept at reset value.

### CAN receive FIFO mailbox data length control and time stamp register (CAN_RDTxR) (x=0..1)

Address offsets: 0x1B4, 0x1C4
Reset value: 0xXXXX XXXX

All RX registers are write protected.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIME[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FMI[7:0] | | | | | | | | Reserved | | | | DLC[3:0] | | | |
| r | r | r | r | r | r | r | r | | | | | r | r | r | r |

Bits 31:16 **TIME[15:0]**: Message time stamp
This field contains the 16-bit timer value captured at the SOF detection.

Bits 15:8 **FMI[7:0]**: Filter match index
This register contains the index of the filter the message stored in the mailbox passed through. For more details on identifier filtering refer to *Section 24.7.4*

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **DLC[3:0]**: Data length code
This field defines the number of data bytes a data frame contains (0 to 8). It is 0 in the case of a remote frame request.

### CAN receive FIFO mailbox data low register (CAN_RDLxR) (x=0..1)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x1B8, 0x1C8
Reset value: 0xXXXX XXXX

All RX registers are write protected.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA3[7:0] | | | | | | | | DATA2[7:0] | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA1[7:0] | | | | | | | | DATA0[7:0] | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:24 **DATA3[7:0]**: Data Byte 3
Data byte 3 of the message.

Bits 23:16 **DATA2[7:0]**: Data Byte 2
Data byte 2 of the message.

Bits 15:8 **DATA1[7:0]**: Data Byte 1
Data byte 1 of the message.

Bits 7:0 **DATA0[7:0]**: Data Byte 0
Data byte 0 of the message.
A message can contain from 0 to 8 data bytes and starts with byte 0.

### CAN receive FIFO mailbox data high register (CAN_RDHxR) (x=0..1)

Address offsets: 0x1BC, 0x1CC
Reset value: 0xXXXX XXXX

All RX registers are write protected.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA7[7:0] | | | | | | | | DATA6[7:0] | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA5[7:0] | | | | | | | | DATA4[7:0] | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bits 31:24 **DATA7[7:0]**: Data Byte 7
Data byte 3 of the message.

Bits 23:16 **DATA6[7:0]**: Data Byte 6
Data byte 2 of the message.

Bits 15:8 **DATA5[7:0]**: Data Byte 5
Data byte 1 of the message.

Bits 7:0 **DATA4[7:0]**: Data Byte 4
Data byte 0 of the message.

## 24.9.4 CAN filter registers

### CAN filter master register (CAN_FMR)

Address offset: 0x200
Reset value: 0x2A1C 0E01

All bits of this register are set and cleared by software.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | CAN2SB[5:0] | | | | | | Reserved | | | | | | | FINIT |
| | | rw | rw | rw | rw | rw | rw | | | | | | | | rw |

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **CAN2SB[5:0]**: CAN2 start bank
These bits are set and cleared by software. They define the start bank for the CAN2 interface (Slave) in the range 0 to 27.
*Note: When CAN2SB[5:0] = 28d, all the filters to CAN1 can be used.*
*When CAN2SB[5:0] is set to 0, no filters are assigned to CAN1.*

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **FINIT**: Filter init mode
Initialization mode for filter banks
0: Active filters mode.
1: Initialization mode for the filters.

### CAN filter mode register (CAN_FM1R)

Address offset: 0x204
Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN_FMR register.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | FBM27 | FBM26 | FBM25 | FBM24 | FBM23 | FBM22 | FBM21 | FBM20 | FBM19 | FBM18 | FBM17 | FBM16 |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FBM15 | FBM14 | FBM13 | FBM12 | FBM11 | FBM10 | FBM9 | FBM8 | FBM7 | FBM6 | FBM5 | FBM4 | FBM3 | FBM2 | FBM1 | FBM0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

*Note: Refer to Figure 230.*

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **FBM*x***: Filter mode
Mode of the registers of Filter x.
0: Two 32-bit registers of filter bank x are in Identifier Mask mode.
1: Two 32-bit registers of filter bank x are in Identifier List mode.
*Note: Bits 27:14 are available in connectivity line devices only and are reserved otherwise.*

### CAN filter scale register (CAN_FS1R)

Address offset: 0x20C
Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN_FMR register.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | FSC27 | FSC26 | FSC25 | FSC24 | FSC23 | FSC22 | FSC21 | FSC20 | FSC19 | FSC18 | FSC17 | FSC16 |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FSC15 | FSC14 | FSC13 | FSC12 | FSC11 | FSC10 | FSC9 | FSC8 | FSC7 | FSC6 | FSC5 | FSC4 | FSC3 | FSC2 | FSC1 | FSC0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **FSC*x***: Filter scale configuration
These bits define the scale configuration of Filters 13-0.
0: Dual 16-bit scale configuration
1: Single 32-bit scale configuration
*Note: Bits 27:14 are available in connectivity line devices only and are reserved otherwise.*

*Note: Refer to Figure 230.*

### CAN filter FIFO assignment register (CAN_FFA1R)

Address offset: 0x214
Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN_FMR register.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | FFA27 | FFA26 | FFA25 | FFA24 | FFA23 | FFA22 | FFA21 | FFA20 | FFA19 | FFA18 | FFA17 | FFA16 |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FFA15 | FFA14 | FFA13 | FFA12 | FFA11 | FFA10 | FFA9 | FFA8 | FFA7 | FFA6 | FFA5 | FFA4 | FFA3 | FFA2 | FFA1 | FFA0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **FFA*x***: Filter FIFO assignment for filter x

The message passing through this filter will be stored in the specified FIFO.
0: Filter assigned to FIFO 0
1: Filter assigned to FIFO 1

*Note: Bits 27:14 are available in connectivity line devices only and are reserved otherwise.*

### CAN filter activation register (CAN_FA1R)

Address offset: 0x21C
Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | FACT27 | FACT26 | FACT25 | FACT24 | FACT23 | FACT22 | FACT21 | FACT20 | FACT19 | FACT18 | FACT17 | FACT16 |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FACT15 | FACT14 | FACT13 | FACT12 | FACT11 | FACT10 | FACT9 | FACT8 | FACT7 | FACT6 | FACT5 | FACT4 | FACT3 | FACT2 | FACT1 | FACT0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **FACT*x***: Filter active

The software sets this bit to activate Filter x. To modify the Filter x registers (CAN_FxR[0:7]), the FACTx bit must be cleared or the FINIT bit of the CAN_FMR register must be set.
0: Filter x is not active
1: Filter x is active

*Note: Bits 27:14 are available in connectivity line devices only and are reserved otherwise.*

### Filter bank i register x (CAN_FiRx) (i=0..27, x=1, 2)

Address offsets: 0x240..0x31C
Reset value: 0xXXXX XXXX

*n connectivity line devices there are 28 filter banks, i=0 .. 27, in other devices there are 14 filter banks i = 0 ..13.* Each filter bank i is composed of two 32-bit registers, CAN_FiR[2:1].

This register can only be modified when the FACTx bit of the CAN_FAxR register is cleared or when the FINIT bit of the CAN_FMR register is set.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FB31 | FB30 | FB29 | FB28 | FB27 | FB26 | FB25 | FB24 | FB23 | FB22 | FB21 | FB20 | FB19 | FB18 | FB17 | FB16 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FB15 | FB14 | FB13 | FB12 | FB11 | FB10 | FB9 | FB8 | FB7 | FB6 | FB5 | FB4 | FB3 | FB2 | FB1 | FB0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

In all configurations:

Bits 31:0 **FB[31:0]**: Filter bits

**Identifier**
Each bit of the register specifies the level of the corresponding bit of the expected identifier.
0: Dominant bit is expected
1: Recessive bit is expected

**Mask**
Each bit of the register specifies whether the bit of the associated identifier register must match with the corresponding bit of the expected identifier or not.
0: Don't care, the bit is not used for the comparison
1: Must match, the bit of the incoming identifier must have the same level has specified in the corresponding identifier register of the filter.

*Note: Depending on the scale and mode configuration of the filter the function of each register can differ. For the filter mapping, functions description and mask registers association, refer to Section 24.7.4.*

A Mask/Identifier register in **mask mode** has the same bit mapping as in **identifier list** mode.

*For the register mapping/addresses of the filter banks refer to Table 181.*

### 24.9.5 bxCAN register map

Refer to *Section 2.3: Memory map* for the register boundary addresses. In connectivity line devices, the registers from offset 0x200 to 31C are present only in CAN1.

**Table 181. bxCAN register map and reset values**

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x000 | CAN_MCR | Reserved | | | | | | | | | | | | | | | DBF | RESET | Reserved | | | | | | | TTCM | ABOM | AWUM | NART | RFLM | TXFP | SLEEP | INRQ |
| | Reset value | | | | | | | | | | | | | | | | 1 | 0 | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0x004 | CAN_MSR | Reserved | | | | | | | | | | | | | | | | | | | | RX | SAMP | RXM | TXM | Res. | | | SLAKI | WKUI | ERRI | SLAK | INAK |
| | Reset value | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 0 | 0 | | | | 0 | 0 | 0 | 1 | 0 |
| 0x008 | CAN_TSR | LOW[2:0] | | | TME[2:0] | | | CODE[1:0] | | ABRQ2 | Res. | | | TERR2 | ALST2 | TXOK2 | RQCP2 | ABRQ1 | Res. | | | TERR1 | ALST1 | TXOK1 | RQCP1 | ABRQ0 | Res.. | | | TERR0 | ALST0 | TXOK0 | RQCP0 |
| | Reset value | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | | | | 0 | 0 | 0 | 0 | 0 | | | | 0 | 0 | 0 | 0 | 0 | | | | 0 | 0 | 0 | 0 |
| 0x00C | CAN_RF0R | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | RFOM0 | FOVR0 | FULL0 | Reserved | FMP0[1:0] | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | | 0 | 0 |
| 0x010 | CAN_RF1R | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | RFOM1 | FOVR1 | FULL1 | Reserved | FMP1[1:0] | |
| | Reset value | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | | 0 | 0 |
| 0x014 | CAN_IER | Reserved | | | | | | | | | | | | | | SLKIE | WKUIE | ERRIE | Res. | | | LECIE | BOFIE | EPVIE | EWGIE | Reserved | FOVIE1 | FFIE1 | FMPIE1 | FOVIE0 | FFIE0 | FMPIE0 | TMEIE |
| | Reset value | | | | | | | | | | | | | | | 0 | 0 | 0 | | | | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x018 | CAN_ESR | REC[7:0] | | | | | | | | TEC[7:0] | | | | | | | | | Reserved | | | | | | | | LEC[2:0] | | | Reserved | BOFF | EPVF | EWGF |
| | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | 0 | 0 | 0 | | 0 | 0 | 0 |
| 0x01C | CAN_BTR | SILM | LBKM | Reserved | | | | SJW[1:0] | | Reserved | TS2[2:0] | | | TS1[3:0] | | | | | Reserved | | | | | | | | BRP[9:0] | | | | | | | |
| | Reset value | 0 | 0 | | | | | 0 | 0 | | 0 | 1 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x020-0x17F | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x180 | CAN_TI0R | STID[10:0]/EXID[28:18] | | | | | | | | | | | EXID[17:0] | | | | | | | | | | | | | | | | | | IDE | RTR | TXRQ |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 0 |
| 0x184 | CAN_TDT0R | TIME[15:0] | | | | | | | | | | | | | | | | | Reserved | | | | | | | TGT | Reserved | | | | DLC[3:0] | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | | | | | | | | | x | | | | | x | x | x | x |
| 0x188 | CAN_TDL0R | DATA3[7:0] | | | | | | | | DATA2[7:0] | | | | | | | | | DATA1[7:0] | | | | | | | | DATA0[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

**Table 181. bxCAN register map and reset values (continued)**

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x18C | CAN_TDH0R | DATA7[7:0] | | | | | | | | DATA6[7:0] | | | | | | | | DATA5[7:0] | | | | | | | | DATA4[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x190 | CAN_TI1R | STID[10:0]/EXID[28:18] | | | | | | | | EXID[17:0] | | | | | | | | | | | | | | | | | | | | IDE | RTR | TXRQ |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 0 |
| 0x194 | CAN_TDT1R | TIME[15:0] | | | | | | | | | | | | | | | | Reserved | | | | | | | TGT | Reserved | | | | DLC[3:0] | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | | | | | | | | x | | | | | x | x | x | x |
| 0x198 | CAN_TDL1R | DATA3[7:0] | | | | | | | | DATA2[7:0] | | | | | | | | DATA1[7:0] | | | | | | | | DATA0[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x19C | CAN_TDH1R | DATA7[7:0] | | | | | | | | DATA6[7:0] | | | | | | | | DATA5[7:0] | | | | | | | | DATA4[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x1A0 | CAN_TI2R | STID[10:0]/EXID[28:18] | | | | | | | | EXID[17:0] | | | | | | | | | | | | | | | | | | | | IDE | RTR | TXRQ |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 0 |
| 0x1A4 | CAN_TDT2R | TIME[15:0] | | | | | | | | | | | | | | | | Reserved | | | | | | | TGT | Reserved | | | | DLC[3:0] | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | | | | | | | | x | | | | | x | x | x | x |
| 0x1A8 | CAN_TDL2R | DATA3[7:0] | | | | | | | | DATA2[7:0] | | | | | | | | DATA1[7:0] | | | | | | | | DATA0[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x1AC | CAN_TDH2R | DATA7[7:0] | | | | | | | | DATA6[7:0] | | | | | | | | DATA5[7:0] | | | | | | | | DATA4[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x1B0 | CAN_RI0R | STID[10:0]/EXID[28:18] | | | | | | | | EXID[17:0] | | | | | | | | | | | | | | | | | | | | IDE | RTR | Reserved |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | |
| 0x1B4 | CAN_RDT0R | TIME[15:0] | | | | | | | | | | | | | | | | FMI[7:0] | | | | | | | | Reserved | | | | DLC[3:0] | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | | | | | x | x | x | x |
| 0x1B8 | CAN_RDL0R | DATA3[7:0] | | | | | | | | DATA2[7:0] | | | | | | | | DATA1[7:0] | | | | | | | | DATA0[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x1BC | CAN_RDH0R | DATA7[7:0] | | | | | | | | DATA6[7:0] | | | | | | | | DATA5[7:0] | | | | | | | | DATA4[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x1C0 | CAN_RI1R | STID[10:0]/EXID[28:18] | | | | | | | | EXID[17:0] | | | | | | | | | | | | | | | | | | | | IDE | RTR | Reserved |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | |

**Table 181. bxCAN register map and reset values (continued)**

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x1C4 | CAN_RDT1R | TIME[15:0] | | | | | | | | | | | | | | | | FMI[7:0] | | | | | | | | Reserved | | | | DLC[3:0] | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | | | | | x | x | x | x |
| 0x1C8 | CAN_RDL1R | DATA3[7:0] | | | | | | | | DATA2[7:0] | | | | | | | | DATA1[7:0] | | | | | | | | DATA0[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x1CC | CAN_RDH1R | DATA7[7:0] | | | | | | | | DATA6[7:0] | | | | | | | | DATA5[7:0] | | | | | | | | DATA4[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x1D0-0x1FF | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x200 | CAN_FMR | Reserved | | | | | | | | | | | | | | | | | | CAN2SB[5:0] | | | | | | Reserved | | | | | | | FINIT |
| | Reset value | | | | | | | | | | | | | | | | | | | 0 | 0 | 1 | 1 | 1 | 0 | | | | | | | | 1 |
| 0x204 | CAN_FM1R | Reserved | | | | | | | | FBM[27:0] | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x208 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x20C | CAN_FS1R | Reserved | | | | | | | | FSC[27:0] | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x210 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x214 | CAN_FFA1R | Reserved | | | | | | | | FFA[27:0] | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x218 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x21C | CAN_FA1R | Reserved | | | | | | | | FACT[27:0] | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x220 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x224-0x23F | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x240 | CAN_F0R1 | FB[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x244 | CAN_F0R2 | FB[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x248 | CAN_F1R1 | FB[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

**Table 181. bxCAN register map and reset values (continued)**

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x24C | CAN_F1R2 | FB[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| . . . . | . . . . | . | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x318 | CAN_F27R1 | FB[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x31C | CAN_F27R2 | FB[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |