Controller Area Network (CAN)

This chapter describes the controller area network for the device.

Topic **Page**

23.1 Introduction

23.1.1 DCAN Features

The general features of the DCAN controller are:

- Supports CAN protocol version 2.0 part A, B (ISO 11898-1)
- Bit rates up to 1 MBit/s
- Dual clock source
- 16, 32, 64 or 128 message objects (instantiated as 64 on this device)
- Individual identifier mask for each message object
- Programmable FIFO mode for message objects
- Programmable loop-back modes for self-test operation
- Suspend mode for debug support
- Software module reset
- Automatic bus on after Bus-Off state by a programmable 32-bit timer
- Message RAM parity check mechanism
- Direct access to Message RAM during test mode
- CAN Rx / Tx pins configurable as general purpose IO pins
- Two interrupt lines (plus additional parity-error interrupt line)
- RAM initialization
- DMA support

23.1.2 Unsupported DCAN Features

The DCAN module in this device does not support GPIO pin mode. All GPIO functionality is mapped through the GPIO modules and muxed at the pins. GPIO pin control signals from the DCAN modules are not connected.

23.2 Integration

The Controller Area Network is a serial communications protocol which efficiently supports distributed realtime control with a high level of security. The DCAN module supports bitrates up to 1 Mbit/s and is compliant to the CAN 2.0B protocol specification. The core IP within DCAN is provided by Bosch.

This device includes two instantiations of the DCAN controller: DCAN0 and DCAN1. Figure 23-1 shows the DCAN module integration.

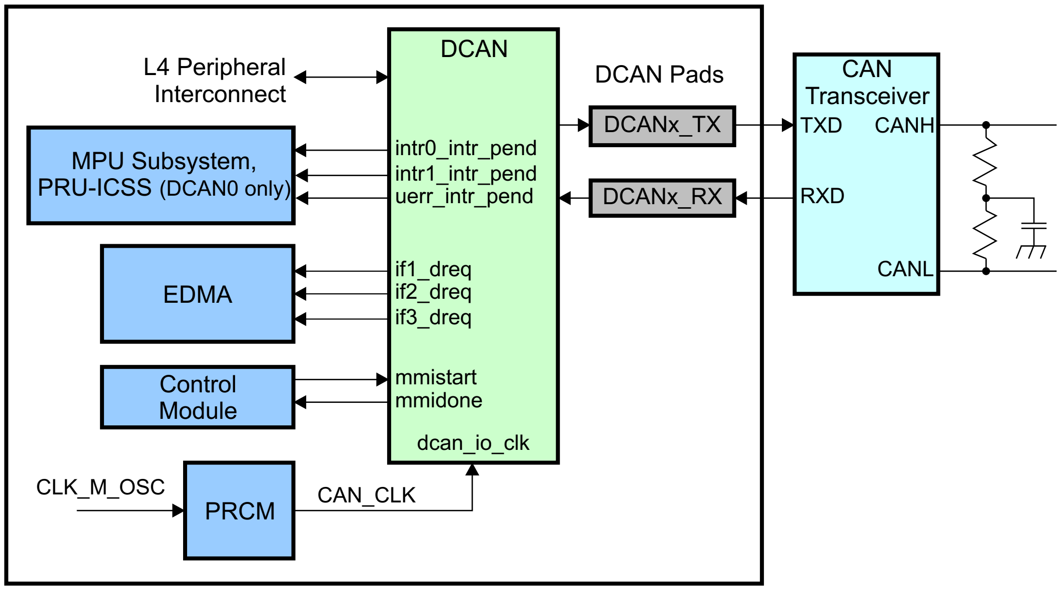


Figure 23-1. DCAN Integration

23.2.1 DCAN Connectivity Attributes

The general connectivity attributes for the DCAN module are shown in Table 23-1.

Table 23-1. DCAN Connectivity Attributes

Attributes	Type
Power Domain	Peripheral Domain
Clock Domain	PD_PER_L4LS_GCLK (OCP) PD_PER_CAN_CLK (Func)
Reset Signals	PER_DOM_RST_N
Idle/Wakeup Signals	Smart Idle
Interrupt Requests	3 Interrupts per instance Intr0 (DCANx_INT0) – Error, Status, Msg Object interrupt Intr1 (DCANx_INT1) – Msg Object interrupt Uerr (DCANx_PARITY) – Parity error interrupt All DCAN0 interrupts to MPU Subsystem and PRU-ICSS All DCAN1 interrupts to only MPU Subsystem
DMA Requests	3 DMA requests per instance to EDMA (CAN_IFxDMA)
Physical Address	L4 Peripheral slave port

23.2.2 DCAN Clock and Reset Management

The DCAN controllers have separate bus interface and functional clocks.

Table 23-2. DCAN Clock Signals

Clock Signal	Max Freq	Reference / Source	Comments
DCAN_ocp_clk Interface clock	100 MHz	CORE_CLKOUTM4 / 2	pd_per_l4ls_gclk from PRCM
DCAN_io_clk Functional clock	26 MHz	CLK_M_OSC	pd_per_can_clk from PRCM

23.2.3 DCAN Pin List

The external signals for the DCAN module are shown in the following table.

Table 23-3. DCAN Pin List

Pin	Type	Description
DCAN*x*_TX	O	DCAN transmit line
DCAN*x*_RX	I	DCAN receive line

23.3 Functional Description

The DCAN module performs CAN protocol communication according to ISO 11898-1. The bit rate can be programmed to values up to 1 MBit/s. Additional transceiver hardware is required for the connection to the physical layer (CAN bus).

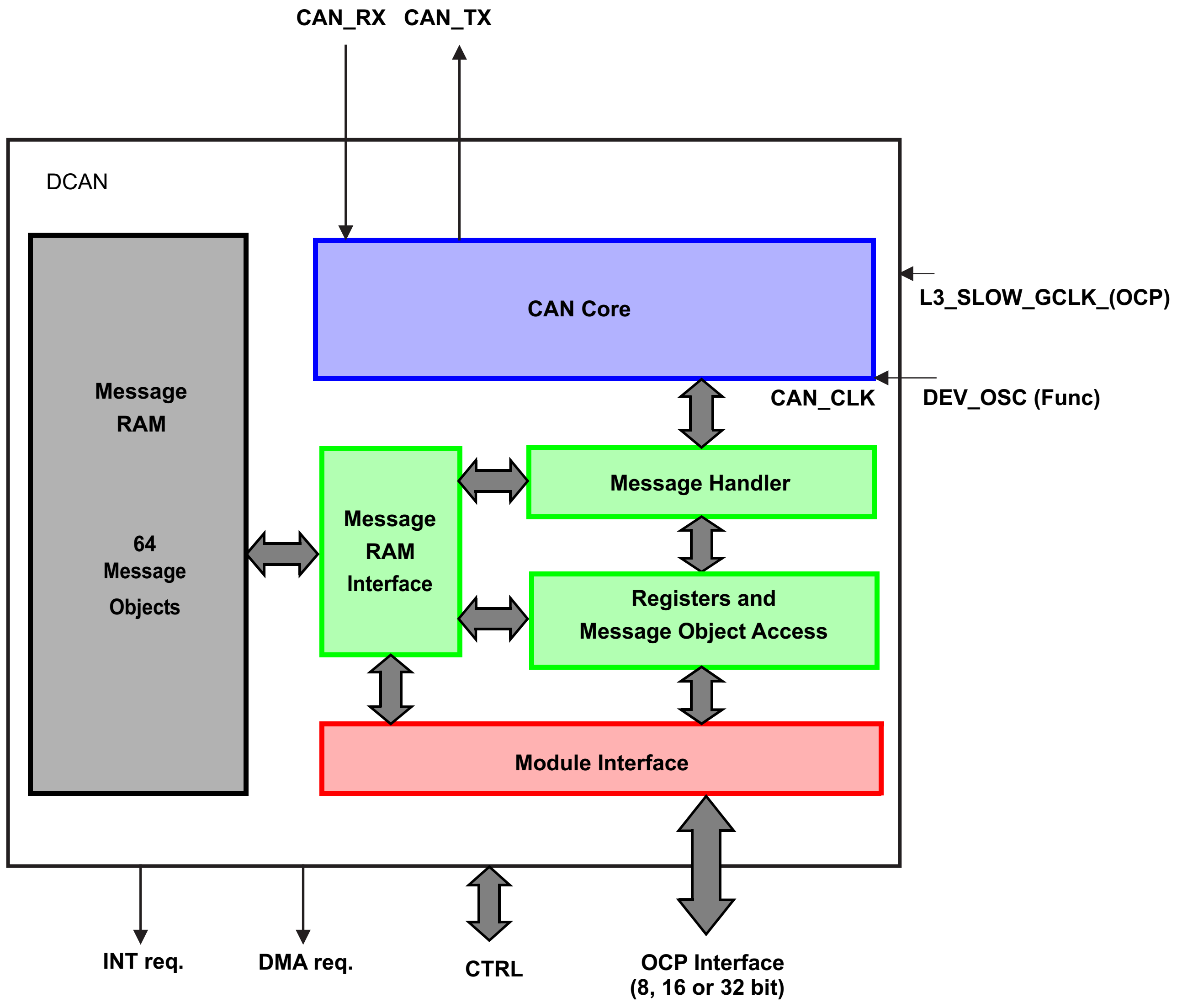
For communication on a CAN network, individual message objects can be configured. The message objects and identifier masks are stored in the message RAM.

All functions concerning the handling of messages are implemented in the message handler. Those functions are acceptance filtering, the transfer of messages between the CAN core and the message RAM, and the handling of transmission requests, as well as the generation of interrupts or DMA requests.

The register set of the DCAN module can be accessed directly by the CPU via the module interface. These registers are used to control and configure the CAN core and the message handler, and to access the message RAM.

Figure 23-2 shows the DCAN block diagram and its features are described below.

Figure 23-2. DCAN Block Diagram



23.3.1 CAN Core

The CAN core consists of the CAN protocol controller and the Rx/Tx shift register. It handles all ISO 11898-1 protocol functions.

23.3.2 Message Handler

The message handler is a state machine that controls the data transfer between the single-ported message RAM and the CAN core's Rx/Tx shift register. It also handles acceptance filtering and the interrupt/DMA request generation as programmed in the control registers.

23.3.3 Message RAM

The DCAN0 and DCAN1 enables a storage of 64 CAN messages.

23.3.4 Message RAM Interface

Three interface register sets control the CPU read and write accesses to the message RAM. There are two interface registers sets for read and write access, IF1 and IF2, and one interface register set for read access only, IF3. Additional information can be found in Section 23.3.15.12.

The interface registers have the same word-length as the message RAM.

23.3.5 Registers and Message Object Access

Data consistency is ensured by indirect accesses to the message objects. During normal operation, all CPU and DMA accesses to the message RAM are done through interface registers. In a dedicated test mode, the message RAM is memory mapped and can be directly accessed by either CPU or DMA.

23.3.6 Module Interface

The DCAN module registers are accessed by the CPU or user software through a 32-bit peripheral bus interface.

23.3.7 Dual Clock Source

Two clock domains are provided to the DCAN module: the peripheral synchronous clock domain (L3_SLOW_GCLK) and the peripheral asynchronous clock source domain (CLK_M_OSC) for CAN_CLK.

23.3.8 CAN Operation

After hardware reset, the Init bit in the CAN control register (CTL) is set and all CAN protocol functions are disabled. The CAN module must be initialized before operating it. Figure 23-3 illustrates the basic initialization flow for the CAN module.

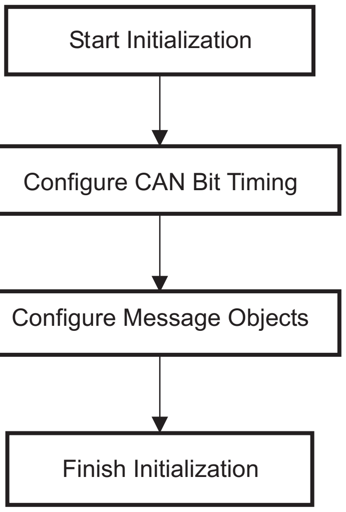
23.3.8.1 CAN Module Initialization

A general CAN module initialization would mean the following two critical steps:

- Configuration of the CAN bit timing
- Configuration of message objects

To initialize the CAN controller, the CPU has to set up the CAN bit timing and those message objects that have to be used for CAN communication. Message objects that are not needed, can be deactivated.

Figure 23-3. CAN Module General Initialization Flow



23.3.8.1.1 Configuration of CAN Bit Timing

The CAN module must be in initialization mode to configure the CAN bit timing.

For CAN bit timing software configuration flow, see Figure 23-4.

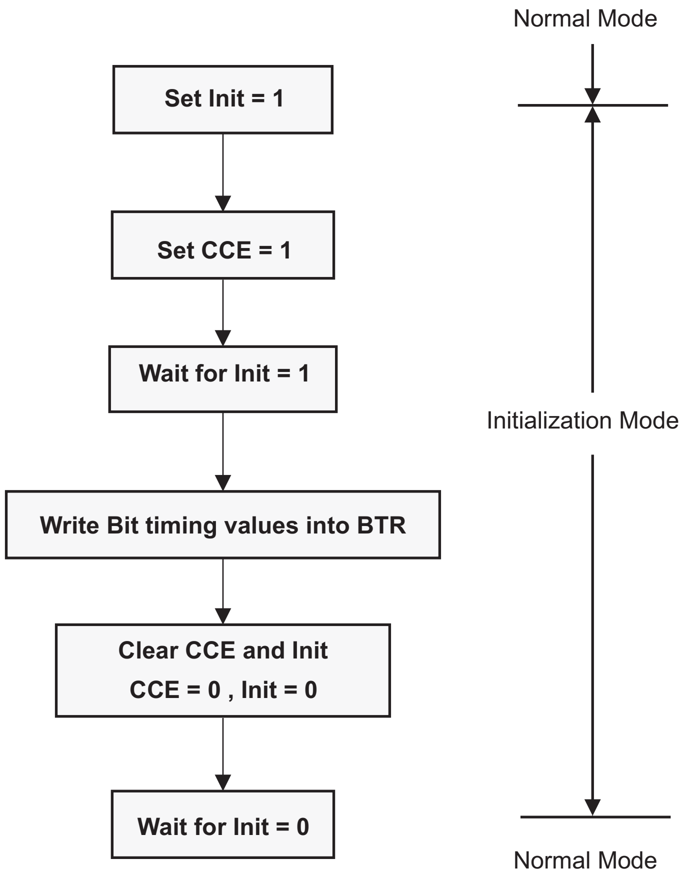
Step 1: Enter *initialization mode* by setting the Init (Initialization) bit in the CAN control register.

While the Init bit is set, the message transfer from and to the CAN bus is stopped, and the status of the CAN_TX output is recessive (high).

The CAN error counters are not updated. Setting the Init bit does not change any other configuration register.

Also, note that the CAN module is in initialization mode on hardware reset and during Bus-Off.

Figure 23-4. CAN Bit-Timing Configuration



Step 2: Set the Configure Change Enable (CCE) bit in the CAN control register.

The access to the Bit Timing register (BTR) for the configuration of the bit timing is enabled when both the Init and CCE bits in the CAN Control register are set.

Step 3: Wait for the Init bit to get set. This would make sure that the module has entered *Initialization mode*.

Step 4: Write the bit timing values into the bit timing register. See Section 23.3.16.2 for the BTR value calculation for a given bit timing.

Step 5: Clear the CCE and Init bit.

Step 6: Wait for the Init bit to clear. This would ensure that the module has come out of *initialization mode*.

Following these steps, the module comes to operation by synchronizing itself to the CAN bus, provided the BTR is configured as per the CAN bus baud rate, although the message objects have to be configured before carrying out any communication.

NOTE: The module will not come out of the *initialization mode* if any incorrect BTR values are written in step 4.

NOTE: The required message objects should be configured as transmit or receive objects before the start of data transfer as explained in Section 23.3.8.1.

23.3.8.1.2 Configuration of Message Objects

The message objects can be configured only through the interface registers; the CPU does not have direct access to the message object (message RAM) . Familiarize yourself with the interface register set (IFx) usage (see Section 23.3.17) and the message object structure (see Section 23.3.18) before configuring the message objects.

For more information regarding the procedure to configure the message objects, see Section 23.3.14. All the message objects should be configured to particular identifiers or set to not valid before the message transfer is started. It is possible to change the configuration of message objects during normal operation (that is between data transfers).

NOTE: The message objects initialization is independent of the bit-timing configuration.

23.3.8.1.3 DCAN RAM Hardware Initialization

The memory hardware initialization for the DCAN module is enabled in the device control register (DCAN_RAMINIT) which initializes the RAM with zeros and sets parity bits accordingly. Wait for the RAMINIT_DONE bit to be set to ensure successful RAM initialization. Ensure the clock to the DCAN module is enabled before starting this initialization.

For more details on RAM hardware initialization, see Chapter 9, *Control Module*.

23.3.8.2 CAN Message Transfer (Normal Operation)

Once the DCAN is initialized and the Init bit is reset to zero, the CAN core synchronizes itself to the CAN bus and is ready for message transfer as per the configured message objects.

The CPU may enable the interrupt lines (setting IE0 and IE1 to ‘1’) at the same time when it clears Init and CCE. The status interrupts EIE and SIE may be enabled simultaneously.

The CAN communication can be carried out in any of the following two modes: interrupt and polling.

The interrupt register points to those message objects with IntPnd = ‘1’. It is updated even if the interrupt lines to the CPU are disabled (IE0/IE1 are zero).

The CPU may poll all MessageObject’s NewDat and TxRqst bits in parallel from the NewData X registers and the Transmission Request X Registers (TXRQ X). Polling can be made easier if all transmit objects are grouped at the low numbers and all receive objects are grouped at the high numbers.

Received messages are stored into their appropriate message objects if they pass acceptance filtering.

The whole message (including all arbitration bits, DLC and up to eight data bytes) is stored into the message object. As a consequence (e.g., when the identifier mask is used), the arbitration bits which are masked to “don’t care” may change in the message object when a received message is stored.

The CPU may read or write each message at any time via the interface registers, as the message handler guarantees data consistency in case of concurrent accesses.

If a permanent message object (arbitration and control bits set up during configuration and leaving unchanged for multiple CAN transfers) exists for the message, it is possible to only update the data bytes.

If several transmit messages should be assigned to one message object, the whole message object has to be configured before the transmission of this message is requested.

The transmission of multiple message objects may be requested at the same time. They are subsequently transmitted, according to their internal priority.

Messages may be updated or set to not valid at any time, even if a requested transmission is still pending. However, the data bytes will be discarded if a message is updated before a pending transmission has started.

Depending on the configuration of the message object, a transmission may be automatically requested by the reception of a remote frame with a matching identifier.

23.3.8.2.1 Automatic Retransmission

According to the CAN Specification (ISO11898), the DCAN provides a mechanism to automatically retransmit frames which have lost arbitration or have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed.

By default, this automatic retransmission is enabled. It can be disabled by setting the disable automatic retransmission (DAR) bit in the CTL register. Further details to this mode are provided in Section 23.3.15.3.

23.3.8.2.2 Auto-Bus-On

By default, after the DCAN has entered Bus-Off state, the CPU can start a Bus-Off-Recovery sequence by resetting the Init bit. If this is not done, the module will stay in Bus-Off state.

The DCAN provides an automatic Auto-Bus-On feature which is enabled by bit ABO in the CTL register. If set, the DCAN module will automatically start the Bus-Off-Recovery sequence. The sequence can be delayed by a user-defined number of L3_SLOW_GCLK cycles which can be defined in the Auto-Bus-On Time register (ABOTR).

NOTE: If the DCAN goes to Bus-Off state due to a massive occurrence of CAN bus errors, it stops all bus activities and automatically sets the Init bit. Once the Init bit has been reset by the CPU or due to the Auto-Bus-On feature, the device will wait for 129 occurrences of bus Idle (equal to 129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus-Off recovery sequence, the error counters will be reset.

23.3.8.3 Test Modes

The DCAN module provides several test modes which are mainly intended for production tests or self test.

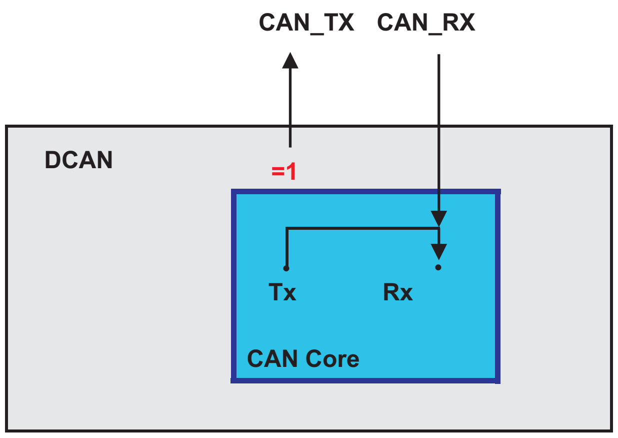
For all test modes, the Test bit in the CTL register needs to be set to one. This enables write access to the test register (TEST).

23.3.8.3.1 Silent Mode

The silent mode may be used to analyze the traffic on the CAN bus without affecting it by sending dominant bits (e.g., acknowledge bit, overload flag, active error flag). The DCAN is still able to receive valid data frames and valid remote frames, but it will not send any dominant bits. However, these are internally routed to the CAN core.

Figure 23-5 shows the connection of signals CAN_TX and CAN_RX to the CAN core in silent mode. Silent mode can be activated by setting the Silent bit in the TEST register to one. In ISO 11898-1, the silent mode is called the bus monitoring mode.

Figure 23-5. CAN Core in Silent Mode



23.3.8.3.2 Loopback Mode

The loopback mode is mainly intended for hardware self-test functions. In this mode, the CAN core uses internal feedback from Tx output to Rx input. Transmitted messages are treated as received messages, and can be stored into message objects if they pass acceptance filtering. The actual value of the CAN_RX input pin is disregarded by the CAN core. Transmitted messages can still be monitored at the CAN_TX pin.

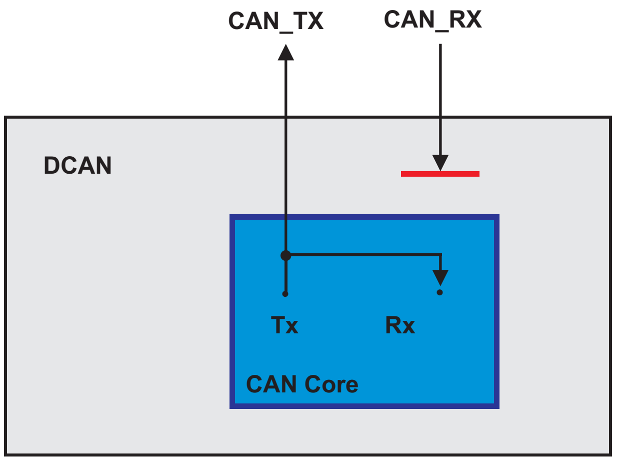
In order to be independent from external stimulation, the CAN core ignores acknowledge sampled in the acknowledge slot of a data/remote frame.

Figure 23-6 shows the connection of signals CAN_TX and CAN_RX to the CAN core in loopback mode.

Loopback mode can be activated by setting bit LBack in the TEST register to one.

NOTE: In loopback mode, the signal path from CAN core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to CAN core are disregarded. For including these into the testing, see Section 23.3.8.3.3.

Figure 23-6. CAN Core in Loopback Mode



23.3.8.3.3 External Loopback Mode

The external loopback mode is similar to the loopback mode; however, it includes the signal path from CAN core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to CAN core. When external loopback mode is selected, the input of the CAN core is connected to the input buffer of the Tx pin.

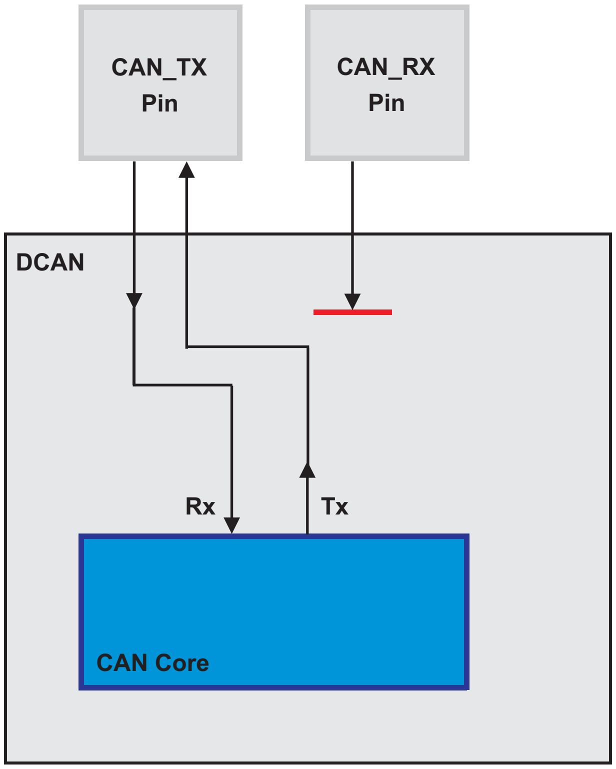
With this configuration, the Tx pin IO circuit can be tested.

External loopback mode can be activated by setting bit **EXL** in the TEST register to one.

Figure 23-7 shows the connection of signals CAN_TX and CAN_RX to the CAN core in external loopback mode.

NOTE: When loopback mode is active (LBack bit set), the ExL bit will be ignored.

Figure 23-7. CAN Core in External Loopback Mode

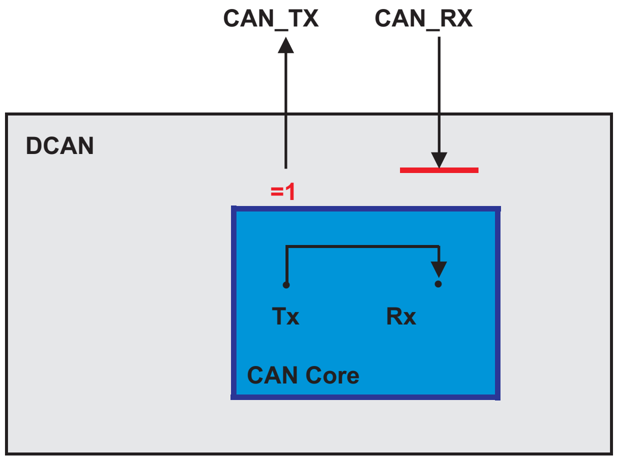


23.3.8.3.4 Loopback Mode Combined With Silent Mode

It is also possible to combine loopback mode and silent mode by setting bits LBack and Silent at the same time. This mode can be used for a “Hot Selftest,” that is, the DCAN hardware can be tested without affecting the CAN network. In this mode, the CAN_RX pin is disconnected from the CAN core and no dominant bits will be sent on the CAN_TX pin.

Figure 23-8 shows the connection of the signals CAN_TX and CAN_RX to the CAN core in case of the combination of loopback mode with silent mode.

Figure 23-8. CAN Core in Loop Back Combined With Silent Mode



23.3.8.3.5 Software Control of CAN_TX pin

Four output functions are available for the CAN transmit pin CAN_TX. In addition to its default function (serial data output), the CAN_TX pin can drive constant dominant or recessive values, or it can drive the CAN sample point signal to monitor the CAN core's bit timing.

Combined with the readable value of the CAN_RX pin, this function can be used to check the physical layer of the CAN bus.

The output mode of pin CAN_TX is selected by programming the TEST register bits Tx[1:0].

NOTE: The software control for the CAN_TX pin interferes with CAN protocol functions. For CAN message transfer or any of the test modes (loopback mode, external loopback mode or silent mode), the CAN_TX pin should operate in its default functionality.

23.3.9 Dual Clock Source

Two clock domains are provided to the DCAN module: the peripheral synchronous clock domain (L3_SLOW_GCLK) as the general module clock source, and the peripheral asynchronous clock source domain (CLK_M_OSC) provided to the CAN core (as clock source CAN_CLK) for generating the CAN bit timing.

Both clock domains can be derived from the same clock source (so that L3_SLOW_GCLK = CLK_M_OSC).

For more information on how to configure the relevant clock source registers in the system module, see Chapter 8, Power and Clock Management.

Between the two clock domains, a synchronization mechanism is implemented in the DCAN module in order to ensure correct data transfer.

NOTE: If the dual clock functionality is used, then L3_SLOW_GCLK must always be higher or equal to CAN_CLK (CLK_M_OSC) (derived from the asynchronous clock source), in order to achieve a stable functionality of the DCAN. Here also the frequency shift of the modulated L3_SLOW_GCLK has to be considered:

$f_{0,\ \mathrm{L3\_SLOW\_GCLK}}(\mathrm{OCP}) \pm \Delta f_{\mathrm{FM},\ \mathrm{L3\_SLOW\_GCLK}}(\mathrm{OCP}) \geq f_{\mathrm{CANCLK}}$

NOTE: The CAN core has to be programmed to at least 8 clock cycles per bit time. To achieve a transfer rate of 1 MBaud when using the asynchronous clock domain as the clock source for CAN_CLK (CLK_M_OSC), an oscillator frequency of 8 MHz or higher has to be used.

23.3.10 Interrupt Functionality

Interrupts can be generated on two interrupt lines: DCANINT0 and DCANINT1. These lines can be enabled by setting the IE0 and IE1 bits, respectively, in the CTL register. The interrupts are level triggered at the chip level.

The DCAN provides three groups of interrupt sources: message object interrupts, status change interrupts, and error interrupts (see Figure 23-9 and Figure 23-10).

The source of an interrupt can be determined by the interrupt identifiers Int0ID/Int1ID in the interrupt register (see INT). When no interrupt is pending, the register will hold the value zero.

Each interrupt line remains active until the dedicated field (Int0ID or Int1ID) in the interrupt register (INT) again reach zero, meaning the cause of the interrupt is reset, or until IE0 or IE1 are reset.

The value 0x8000 in the Int0ID field indicates that an interrupt is pending because the CAN core has updated (not necessarily changed) the Error and Status register (ES). This interrupt has the highest priority. The CPU can update (reset) the status bits WakeUpPnd, RxOk, TxOk and LEC by reading the ES register, but a write access of the CPU will never generate or reset an interrupt.

Values between 1 and the number of the last message object indicates that the source of the interrupt is one of the message objects, Int0ID or Int1ID will point to the pending message interrupt with the highest priority. The Message Object 1 has the highest priority; the last message object has the lowest priority.

An interrupt service routine that reads the message that is the source of the interrupt may read the message and reset the message object's IntPnd at the same time (ClrIntPnd bit in the IF1CMD or IF2CMD register). When IntPnd is cleared, the interrupt register will point to the next message object with a pending interrupt.

23.3.10.1 Message Object Interrupts

Message object interrupts are generated by events from the message objects. They are controlled by the flags IntPND, TxIE and RxIE that are described in Section 23.3.18.1.

Message object interrupts can be routed to either DCANINT0 or DCANINT1 line, controlled by the interrupt multiplexer register (INTMUX12 to INTMUX78).

23.3.10.2 Status Change Interrupts

The events WakeUpPnd, RxOk, TxOk and LEC in the ES register belong to the status change interrupts. The status change interrupt group can be enabled by bit in CTL register.

If SIE is set, a status change interrupt will be generated at each CAN frame, independent of bus errors or valid CAN communication, and also independent of the message RAM configuration.

Status change interrupts can only be routed to interrupt line DCAN0INT, which has to be enabled by setting the IE0 bit in the CTL register.

NOTE: Reading the error and status register will clear the WakeUpPnd flag. If in global power-down mode, the WakeUpPnd flag is cleared by such a read access before the DCAN module has been waken up by the system, the DCAN may re-assert the WakeUpPnd flag, and a second interrupt may occur.

23.3.10.3 Error Interrupts

The events PER, BOff and EWarn, monitored in the ES register, belong to the error interrupts. The error interrupt group can be enabled by setting the EIE bit in the CTL register.

Error interrupts can only be routed to interrupt line DCAN0INT, which has to be enabled by setting the IE0 bit in the CTL register.

Figure 23-9. CAN Interrupt Topology 1

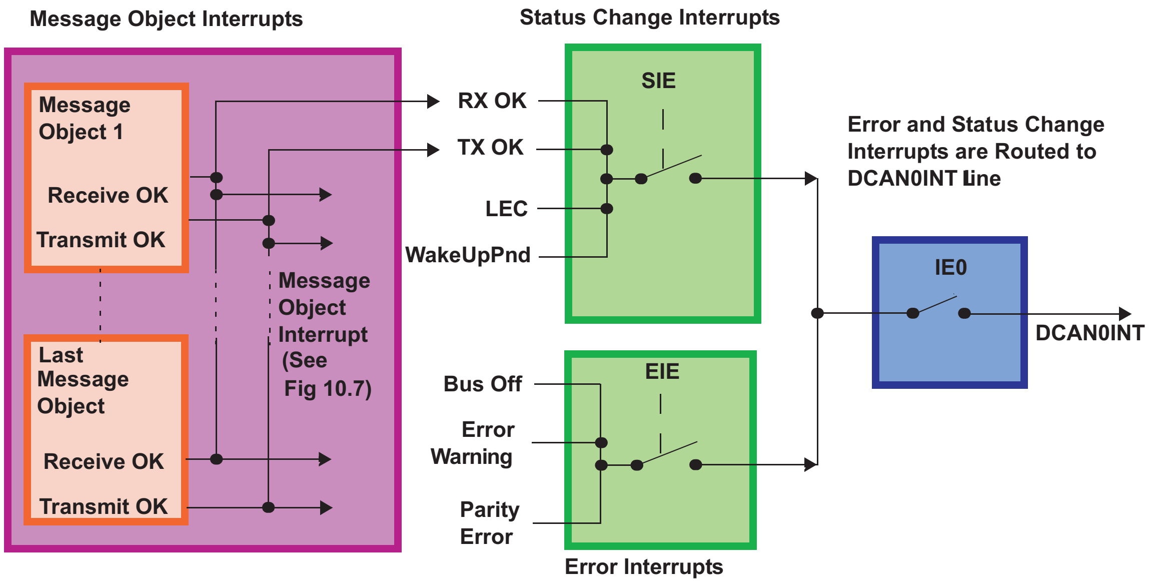
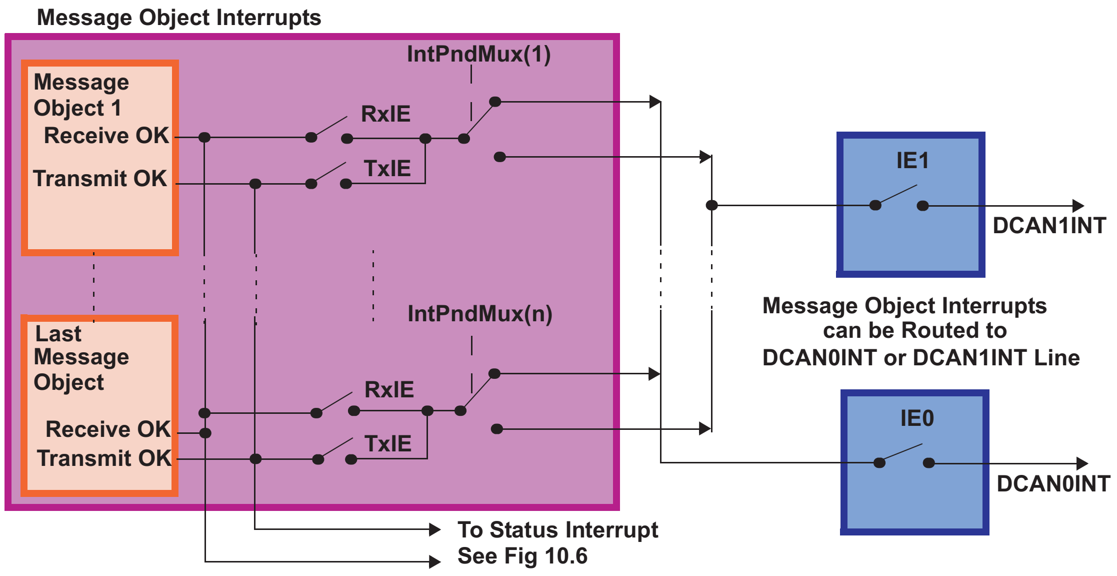


Figure 23-10. CAN Interrupt Topology 2



23.3.11 Local Power-Down Mode

The DCAN supports a local power-down mode, which can be controlled within the CTL register.

23.3.11.1 Entering Local Power-Down Mode

The local power-down mode is requested by setting the PDR bit in the CTL register.

The DCAN then finishes all transmit requests of the message objects. When all requests are done, the DCAN module waits until a bus idle state is recognized. The module will automatically set the Init bit in the CTL register to prevent any further CAN transfers, and it will also set the PDA bit in the CAN error and status register (ES). With setting the PDA bits, the DCAN module indicates that the local power-down mode has been entered.

During local power-down mode, the internal clocks of the DCAN module are turned off, but there is wakeup logic (see Section 23.3.11.2) that can be active, if enabled. Also, the actual contents of the control registers can be read back.

NOTE: In local low-power mode, the application should not clear the Init bit while PDR is set. If there are any messages in the message RAM which are configured as transmit messages and the application resets the init bit, these messages may get sent.

23.3.11.2 Wakeup From Local Power Down

There are two ways to wake up the DCAN from local power-down mode:

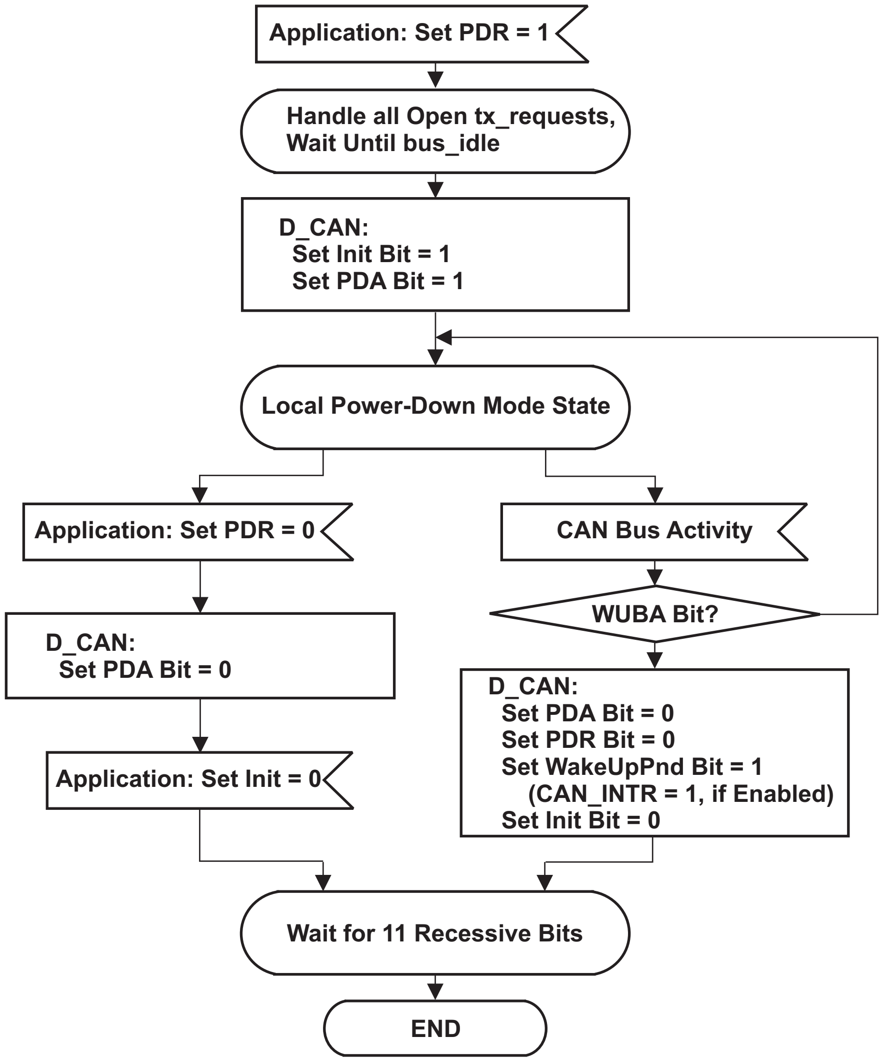
- The application could wake up the DCAN module manually by clearing the PDR bit and then clearing the Init bit in the CTL register.
- Alternatively, a CAN bus activity detection circuit can be activated by setting the wakeup on bus activity (WUBA) bit in the CTL register. If this circuit is active, on occurrence of a dominant CAN bus level, the DCAN will automatically start the wakeup sequence. It will clear the PDR bit in the CTL register and also clear the PDA bit in the error and status register. The WakeUpPnd bit in the ES register will be set. If status interrupts are enabled, also an interrupt will be generated. Finally the Init bit in the CTL register will be cleared.

After the Init bit has been cleared, the module waits until it detects 11 consecutive recessive bits on the CAN_RX pin and then goes bus-active again.

NOTE: The CAN transceiver circuit has to stay active in order to detect any CAN bus activity while the DCAN is in local power down mode. The first CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power-down and automatic wake-up mode, is lost.

Figure 23-11 shows a flow diagram about entering and leaving local power-down mode.

Figure 23-11. Local Power-Down Mode Flow Diagram


Application: Set PDR = 1
Handle all Open tx_requests, Wait Until bus_idle
D_CAN: Set Init Bit = 1 Set PDA Bit = 1
Local Power-Down Mode State
Application: Set PDR = 0
CAN Bus Activity
WUBA Bit?
D_CAN: Set PDA Bit = 0
D_CAN: Set PDA Bit = 0 Set PDR Bit = 0 Set WakeUpPnd Bit = 1 (CAN_INTR = 1, if Enabled) Set Init Bit = 0
Application: Set Init = 0
Wait for 11 Recessive Bits
END

23.3.12 Parity Check Mechanism

The DCAN provides a parity check mechanism to ensure data integrity of message RAM data. For each word (32 bits) in message RAM, one parity bit will be calculated. The formation of the different words is according to the message RAM representation in RDA mode, see Section 23.3.18.4.

Parity information is stored in the message RAM on write accesses and will be checked against the stored parity bit from message RAM on read accesses.

The Parity check functionality can be enabled or disabled by PMD bit field in the CTL register.

In case of disabled parity check, the parity bits in message RAM will be left unchanged on write access to data area and no check will be done on read access.

If parity checking is enabled, parity bits will be automatically generated and checked by the DCAN. The parity bits could be read in debug/suspend mode (see Section 23.3.18.3) or in RDA mode (see Section 23.3.18.4). However, direct write access to the parity bits is only possible in these two modes with parity check disabled.

A parity bit will be set, if the modulo-2-sum of the data bits is 1. This definition is equivalent to: The parity bit will be set, if the number of 1 bits in the data is odd.

NOTE: The parity scheme is tied to even parity at the device level.

23.3.12.1 Behavior on Parity Error

On any read access to message RAM (e.g., during start of a CAN frame transmission), the parity of the message object will be checked. If a parity error is detected, the PER bit in the ES register will be set. If error interrupts are enabled, an interrupt would also be generated. In order to avoid the transmission of invalid data over the CAN bus, the D bit of the message object will be reset.

The message object data can be read by the host CPU, independently of parity errors. Thus, the application has to ensure that the read data is valid, for example, by immediately checking the parity error code register (PERR) on parity error interrupt.

NOTE: During RAM initialization, no parity check will be done.

23.3.12.2 Parity Testing

Testing the parity mechanism can be done by enabling the bit RamDirectAccess (RDA) and manually writing the parity bits directly to the dedicated RAM locations. With this, data and parity bits could be checked when reading directly from RAM.

NOTE: If parity check was disabled, the application has to ensure correct parity bit handling in order to prevent parity errors later on when parity check is enabled.

23.3.13 Debug/Suspend Mode

The module supports the usage of an external debug unit by providing functions like pausing DCAN activities and making message RAM content accessible via OCP interface.

Before entering debug/suspend mode, the circuit will either wait until a started transmission or reception will be finished and bus idle state is recognized, or immediately interrupt a current transmission or reception. This is depending on bit IDS in the CTL register.

Afterwards, the DCAN enters debug/suspend mode, indicated by InitDbg flag in the CTL register.

During debug/suspend mode, all DCAN registers can be accessed. Reading reserved bits will return ‘0’. Writing to reserved bits will have no effect.

Also, the message RAM will be memory mapped. This allows the external debug unit to read the message RAM. For the memory organization, see Section 23.3.18.3).

NOTE: During debug/suspend mode, the message RAM cannot be accessed via the IFx register sets.

Writing to control registers in debug/suspend mode may influence the CAN state machine and further message handling.

For debug support, the auto clear functionality of the following DCAN registers is disabled:

- ES register (clear of status flags by read)
- IF1CMD and IF2CMD (clear of DMAActive flag by read/write)

23.3.14 Configuration of Message Objects

The whole message RAM should be configured before the end of the initialization, however it is also possible to change the configuration of message objects during CAN communication.

The CAN software driver library should offer subroutines that:

- Transfer a complete message structure into a message object. (Configuration)
- Transfer the data bytes of a message into a message object and set TxRqst and NewDat. (Start a new transmission)
- Get the data bytes of a message from a message object and clear NewDat (and IntPnd). (Read received data)
- Get the complete message from a message object and clear NewDat (and IntPnd). (Read a received message, including identifier, from a message object with UMask = ‘1’)

Parameters of the subroutines are the Message Number and a pointer to a complete message structure or to the data bytes of a message structure.

Two examples of assigning the IFx interface register sets to these subroutines are shown here:

In the first method, the tasks of the application program that may access the module are divided into two groups. Each group is restricted to the use of one of the interface register sets. The tasks of one group may interrupt tasks of the other group, but not of the same group.

In the second method, which may be a special case of the first method, there are only two tasks in the application program that access the module. A Read_Message task that uses IF2 or IF3 to get received messages from the message RAM and a Write_Message task that uses IF1 to write messages to be transmitted (or to be configured) into the message RAM. Both tasks may interrupt each other.

23.3.14.1 Configuration of a Transmit Object for Data Frames

Table 23-4 shows how a transmit object can be initialized.

Table 23-4. Initialization of a Transmit Object

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

The arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of the outgoing message. If an 11-bit identifier (standard frame) is used (Xtd = ‘0’), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored.

The data registers (DLC[3:0] and Data0-7) are given by the application. TxRqst and RmtEn should not be set before the data is valid.

If the TxIE bit is set, the IntPnd bit will be set after a successful transmission of the message object.

If the RmtEn bit is set, a matching received remote frame will cause the TxRqst bit to be set; the remote frame will autonomously be answered by a data frame.

The mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask=’1’) to allow groups of remote frames with similar identifiers to set the TxRqst bit. The Dir bit should not be masked. For details, see Section 23.3.15.8. Identifier masking must be disabled (UMask = ‘0’) if no remote frames are allowed to set the TxRqst bit (RmtEn = ‘0’).

23.3.14.2 Configuration of a Transmit Object for Remote Frames

It is not necessary to configure transmit objects for the transmission of remote frames. Setting TxRqst for a receive object causes the transmission of a remote frame with the same identifier as the data frame for which this receive object is configured.

23.3.14.3 Configuration of a Single Receive Object for Data Frames

Table 23-5 shows how a receive object for data frames can be initialized.

Table 23-5. Initialization of a single Receive Object for Data Frames

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

The arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (standard frame) is used (Xtd = ‘0’), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a data frame with an 11-bit Identifier is received, ID[17:0] is set to ‘0’.

The data length code (DLC[3:0]) is given by the application. When the message handler stores a data frame in the message object, it will store the received data length code and eight data bytes. If the data length code is less than 8, the remaining bytes of the message object may be overwritten by non specified values.

The mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask = ’1’) to allow groups of data frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. If some bits of the mask bits are set to “don’t care,” the corresponding bits of the arbitration register will be overwritten by the bits of the stored data frame.

If the RxIE bit is set, the IntPnd bit will be set when a received data frame is accepted and stored in the message object.

If the TxRqst bit is set, the transmission of a remote frame with the same identifier as actually stored in the arbitration bits will be triggered. The content of the arbitration bits may change if the mask bits are used (UMask = ’1’) for acceptance filtering.

23.3.14.4 Configuration of a Single Receive Object for Remote Frames

Table 23-6 shows how a receive object for remote frames can be initialized.

Table 23-6. Initialization of a Single Receive Object for Remote Frames

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	appl.	0	0	0	0

A receive object for remote frames may be used to monitor remote frames on the CAN bus. The remote frame stored in the receive object will not trigger the transmission of a data frame. Receive objects for remote frames may be expanded to a FIFO buffer (see Section 23.3.14.5).

UMask must be set to ‘1.’ The mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be set to “must-match” or to “don’t care,” to allow groups of remote frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. For details, see Section 23.3.15.8.

The arbitration bits (ID[28:0] and Xtd bit) may be given by the application. They define the identifier and type of accepted received remote frames. If some bits of the mask bits are set to “don’t care,” the corresponding bits of the arbitration bits will be overwritten by the bits of the stored remote frame. If an 11-bit Identifier (standard frame) is used (Xtd = ‘0’), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a remote frame with an 11-bit Identifier is received, ID[17:0] will be set to ‘0.’

The data length code (DLC[3:0]) may be given by the application. When the message handler stores a remote frame in the message object, it will store the received data length code. The data bytes of the message object will remain unchanged.

If the RxIE bit is set, the IntPnd bit will be set when a received remote frame is accepted and stored in the message object.

23.3.14.5 Configuration of a FIFO Buffer

With the exception of the EoB bit, the configuration of receive objects belonging to a FIFO buffer is the same as the configuration of a single receive object.

To concatenate multiple message objects to a FIFO buffer, the identifiers and masks (if used) of these message objects have to be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest number will be the first message object of the FIFO buffer. The EoB bit of all message objects of a FIFO buffer except the last one have to be programmed to zero. The EoB bits of the last message object of a FIFO Buffer is set to one, configuring it as the end of the block.

23.3.15 Message Handling

When initialization is finished, the DCAN module synchronizes itself to the traffic on the CAN bus. It does acceptance filtering on received messages and stores those frames that are accepted into the designated message objects. The application has to update the data of the messages to be transmitted and to enable and request their transmission. The transmission is requested automatically when a matching remote frame is received.

The application may read messages which are received and accepted. Messages that are not read before the next messages is accepted for the same message object will be overwritten.

Messages may be read interrupt-driven or after polling of NewDat.

23.3.15.1 Message Handler Overview

The message handler state machine controls the data transfer between the Rx/Tx shift register of the CAN core and the message RAM. It performs the following tasks:

- Data transfer from message RAM to CAN core (messages to be transmitted)
- Data transfer from CAN core to the message RAM (received messages)
- Data transfer from CAN core to the acceptance filtering unit
- Scanning of message RAM for a matching message object (acceptance filtering)
- Scanning the same message object after being changed by IF1/IF2 registers when priority is the same or higher as the message the object found by last scanning
- Handling of TxRqst flags
- Handling of interrupt flags

The message handler registers contains status flags of all message objects grouped into the following topics:

- Transmission Request Flags
- New Data Flags
- Interrupt Pending Flags
- Message Valid Registers

Instead of collecting above listed status information of each message object via IFx registers separately, these message handler registers provides a fast and easy way to get an overview, for example, about all pending transmission requests.

All message handler registers are read-only.

23.3.15.2 Receive/Transmit Priority

The receive/transmit priority for the message objects is attached to the message number, not to the CAN identifier. Message object 1 has the highest priority, while the last implemented message object has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding message object so messages with the highest priority, for example, can be placed in the message objects with the lowest numbers.

The acceptance filtering for received data frames or remote frames is also done in ascending order of message objects, so a frame that has been accepted by a message object cannot be accepted by another message object with a higher message number. The last message object may be configured to accept any data frame or remote frame that was not accepted by any other message object, for nodes that need to log the complete message traffic on the CAN bus.

23.3.15.3 Transmission of Messages in Event-Driven CAN Communication

If the shift register of the CAN core is ready for loading and if there is no data transfer between the IFx registers and message RAM, the D bits in the Message Valid register (MSGVAL12 to MSGVAL78) and the TxRqst bits in the transmission request register are evaluated. The valid message object with the highest priority pending transmission request is loaded into the shift register by the message handler and the transmission is started. The message object’s NewDat bit is reset.

After a successful transmission and if no new data was written to the message object (NewDat = ‘0’) since the start of the transmission, the TxRqst bit will be reset. If TxIE is set, IntPnd will be set after a successful transmission. If the DCAN module has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. If meanwhile the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

If automatic retransmission mode is disabled by setting the DAR bit in the CTL register, the behavior of bits TxRqst and NewDat in the Message Control register of the interface register set is as follows:

- When a transmission starts, the TxRqst bit of the respective interface register set is reset, while bit NewDat remains set.
- When the transmission has been successfully completed, the NewDat bit is reset.

When a transmission failed (lost arbitration or error) bit NewDat remains set. To restart the transmission, the application has to set TxRqst again.

Received remote frames do not require a receive object. They will automatically trigger the transmission of a data frame, if in the matching transmit object the RmtEn bit is set.

23.3.15.4 Updating a Transmit Object

The CPU may update the data bytes of a transmit object any time via the IF1 and IF2 interface registers, neither D nor TxRqst have to be reset before the update.

Even if only part of the data bytes is to be updated, all four bytes in the corresponding IF1 or IF2 Data A register (IF1DATA or IF2DATA) or IF1 or IF2 Data B register (IF1DATB or IF2DATB) have to be valid before the content of that register is transferred to the message object. Either the CPU has to write all four bytes into the IF1/IF2 data register or the message object is transferred to the IF1/IF2 data register before the CPU writes the new data bytes.

When only the data bytes are updated, first 0x87 can be written to bits [23:16] of the IF1 and IF2 Command register (IF1CMD and IF2CMD) and then the number of the message object is written to bits [7:0] of the command register, concurrently updating the data bytes and setting TxRqst with NewDat.

To prevent the reset of TxRqst at the end of a transmission that may already be in progress while the data is updated, NewDat has to be set together with TxRqst in event driven CAN communication. For details, see Section 23.3.15.3.

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has started.

23.3.15.5 Changing a Transmit Object

If the number of implemented message objects is not sufficient to be used as permanent message objects only, the transmit objects may be managed dynamically. The CPU can write the whole message (arbitration, control, and data) into the interface register. The bits [23:16] of the command register can be set to 0xB7 for the transfer of the whole message object content into the message object. Neither D nor TxRqst have to be reset before this operation.

If a previously requested transmission of this message object is not completed but already in progress, it will be continued; however, it will not be repeated if it is disturbed.

To only update the data bytes of a message to be transmitted, bits [23:16] of the command register should be set to 0x87.

NOTE: After the update of the transmit object, the interface register set will contain a copy of the actual contents of the object, including the part that had not been updated.

23.3.15.6 Acceptance Filtering of Received Messages

When the arbitration and control bits (Identifier + IDE + RTR + DLC) of an incoming message are completely shifted into the shift register of the CAN core, the message handler starts the scan of the message RAM for a matching valid message object:

- The acceptance filtering unit is loaded with the arbitration bits from the CAN core shift register.
- Then the arbitration and mask bits (including MsgVal, UMask, NewDat, and EoB) of message object 1 are loaded into the acceptance filtering unit and are compared with the arbitration bits from the shift register. This is repeated for all following message objects until a matching message object is found, or until the end of the message RAM is reached.
- If a match occurs, the scanning is stopped and the message handler proceeds depending on the type of the frame (data frame or remote frame) received.

23.3.15.7 Reception of Data Frames

The message handler stores the message from the CAN core shift register into the respective message object in the message RAM. Not only the data bytes, but all arbitration bits and the data length code are stored into the corresponding message object. This ensures that the data bytes stay associated to the identifier even if arbitration mask registers are used.

The NewDat bit is set to indicate that new data (not yet seen by the CPU) has been received. The CPU should reset the NewDat bit when it reads the message object. If at the time of the reception the NewDat bit was already set, MsgLst is set to indicate that the previous data (supposedly not seen by the CPU) is lost. If the RxIE bit is set, the IntPnd bit is set, causing the interrupt register (INT) to point to this message object.

The TxRqst bit of this message object is reset to prevent the transmission of a remote frame, while the requested data frame has just been received.

23.3.15.8 Reception of Remote Frames

When a remote frame is received, three different configurations of the matching message object have to be considered:

- Dir = ‘1’ (direction = transmit), RmtEn = ‘1’, UMask = ‘1’ or ‘0’

  The TxRqst bit of this message object is set at the reception of a matching remote frame. The rest of the message object remains unchanged.
- Dir = ‘1’ (direction = transmit), RmtEn = ‘0’, UMask = ‘0’

  The remote frame is ignored, this message object remains unchanged.
- Dir = ‘1’ (direction = transmit), RmtEn = ‘0’, UMask = ‘1’

  The remote frame is treated similar to a received data frame. At the reception of a matching Remote Message Frame, the TxRqst bit of this message object is reset. The arbitration and control bits (Identifier + IDE + RTR + DLC) from the shift register are stored in the message object in the message RAM and the NewDat bit of this message object is set. The data bytes of the message object remain unchanged.

23.3.15.9 Reading Received Messages

The CPU may read a received message any time via the IFx interface register. The data consistency is guaranteed by the message handler state machine. Typically the CPU will write first 0x7F to bits [23:16] and then the number of the message object to bits [7:0] of the command register. That combination will transfer the entire received message from the message RAM into the interface register set. Additionally, the bits NewDat and IntPnd are cleared in the message RAM (not in the interface register set). The values of these bits in the message control register always reflect the status before resetting the bits. If the message object uses masks for acceptance filtering, the arbitration bits show which of the different matching messages has been received.

The actual value of NewDat shows whether a new message has been received since last time when this message object was read. The actual value of MsgLst shows whether more than one message has been received since the last time when this message object was read. MsgLst will not be automatically reset.

23.3.15.10 Requesting New Data for a Receive Object

By means of a remote frame, the CPU may request another CAN node to provide new data for a receive object. Setting the TxRqst bit of a receive object will cause the transmission of a remote frame with the identifier of the receive object. This remote frame triggers the other CAN node to start the transmission of the matching data frame. If the matching data frame is received before the remote frame could be transmitted, the TxRqst bit is automatically reset. Setting the TxRqst bit without changing the contents of a message object requires the value 0x84 in bits [23:16] of the IFxCMD register.

23.3.15.11 Storing Received Messages in FIFO Buffers

Several message objects may be grouped to form one or more FIFO buffers. Each FIFO buffer configured to store received messages with a particular (group of) identifier(s). Arbitration and mask registers of the FIFO buffer’s message objects are identical. The end of buffer (EoB) bits of all but the last of the FIFO buffer’s message objects are ‘0’; in the last one the EoB bit is ‘1.’

Received messages with identifiers matching to a FIFO buffer are stored into a message object of this FIFO buffer, starting with the message object with the lowest message number. When a message is stored into a message object of a FIFO buffer, the NewDat bit of this message object is set. By setting NewDat while EoB is ‘0’, the message object is locked for further write accesses by the message handler until the CPU has cleared the NewDat bit.

Messages are stored into a FIFO buffer until the last message object of this FIFO buffer is reached. If none of the preceding message objects is released by writing NewDat to ‘0,’ all further messages for this FIFO buffer will be written into the last message object of the FIFO buffer (EoB = ‘1’) and therefore overwrite previous messages in this message object.

23.3.15.12 Reading From a FIFO Buffer

Several messages may be accumulated in a set of message objects which are concatenated to form a FIFO buffer before the application program is required (in order to avoid the loss of data) to empty the buffer.

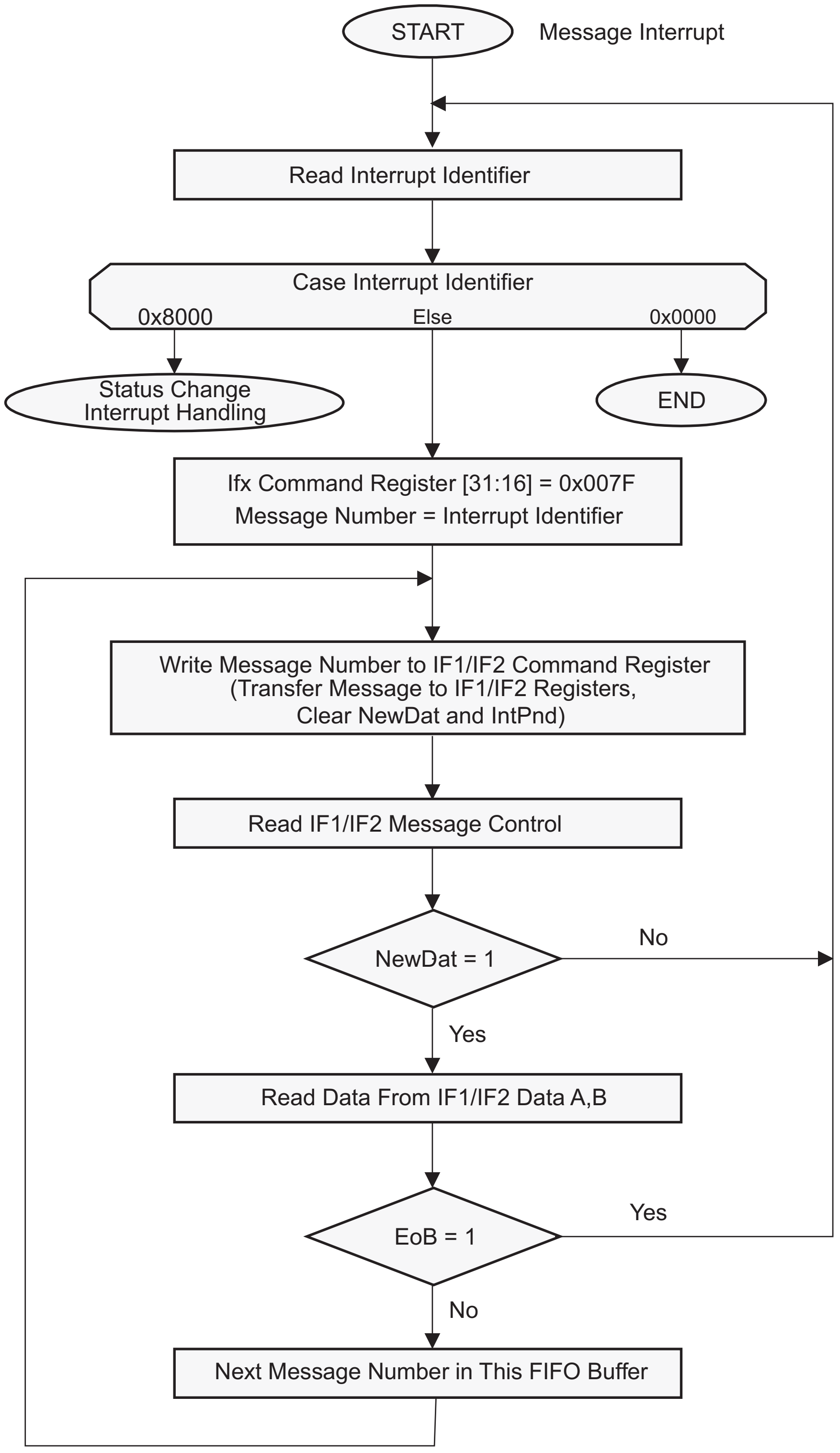
A FIFO buffer of length N will store –1 plus the last received message since last time it was cleared.

A FIFO buffer is cleared by reading and resetting the NewDat bits of all its message objects, starting at the FIFO Object with the lowest message number. This should be done in a subroutine following the example shown in Figure 23-12.

NOTE: All message objects of a FIFO buffer need to be read and cleared before the next batch of messages can be stored. Otherwise, true FIFO functionality can not be guaranteed, since the message objects of a partly read buffer will be re-filled according to the normal (descending) priority.

Reading from a FIFO buffer message object and resetting its NewDat bit is handled the same way as reading from a single message object.

Figure 23-12. CPU Handling of a FIFO Buffer (Interrupt Driven)


START
Message Interrupt
Read Interrupt Identifier
Case Interrupt Identifier
0x8000
Else
0x0000
Status Change Interrupt Handling
END
Ifx Command Register [31:16] = 0x007F
Message Number = Interrupt Identifier
Write Message Number to IF1/IF2 Command Register (Transfer Message to IF1/IF2 Registers, Clear NewDat and IntPnd)
Read IF1/IF2 Message Control
NewDat = 1
No
Yes
Read Data From IF1/IF2 Data A,B
EoB = 1
Yes
No
Next Message Number in This FIFO Buffer

23.3.16 CAN Bit Timing

The DCAN supports bit rates between less than 1 kBit/s and 1000 kBit/s.

Each member of the CAN network has its own clock generator, typically derived from a crystal oscillator. The bit timing parameters can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods (f_{osc}) may be different.

The frequencies of these oscillators are not absolutely stable. Small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range (df), the CAN nodes are able to compensate for the different bit rates by resynchronizing to the bit stream.

In many cases, the CAN bit synchronization will amend a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive.

The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

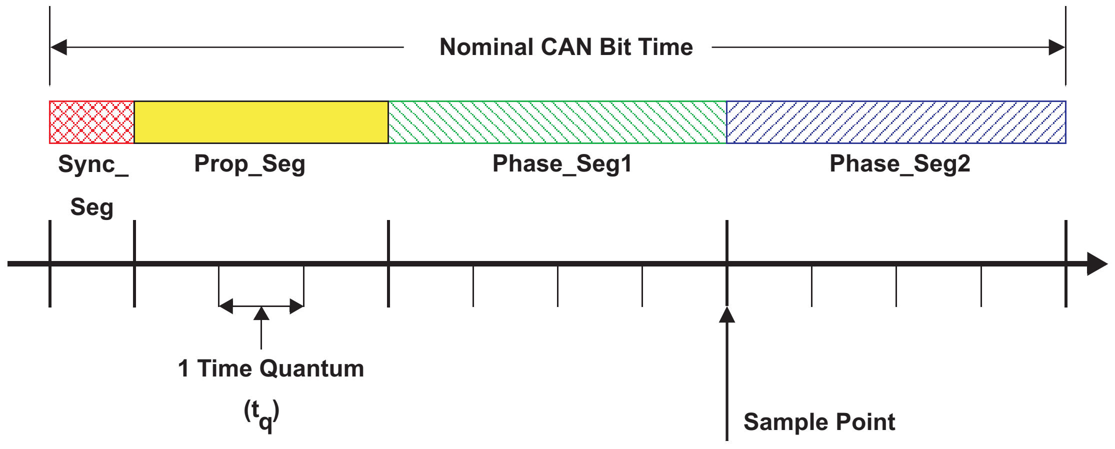
Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

23.3.16.1 Bit Time and Bit Rate

According to the CAN specification, the bit time is divided into four segments (see Figure 23-13):

- Synchronization segment (Sync_Seg)
- Propagation time segment (Prop_Seg)
- Phase buffer segment 1 (Phase_Seg1)
- Phase buffer segment 2 (Phase_Seg2)

Figure 23-13. Bit Timing



Each segment consists of a specific number of time quanta. The length of one time quantum (t_q), which is the basic time unit of the bit time, is given by the CAN_CLK and the baud rate prescalers (BRPE and BRP). With these two baud rate prescalers combined, divider values from 1 to 1024 can be programmed:

t_q = Baud Rate Prescaler / CAN_CLK

Apart from the fixed length of the synchronization segment, these numbers are programmable. Table 23-7 describes the minimum programmable ranges required by the CAN protocol.

A given bit rate may be met by different bit time configurations.

Table 23-7. Parameters of the CAN Bit Time

Parameter	Range	Remark
Sync_Seg	1 t_q (fixed)	Synchronization of bus input to CAN_CLK
Prop_Seg	[1 … 8] t_q	Compensates for the physical delay times
Phase_Seg1	[1 … 8] t_q	May be lengthened temporarily by synchronization
Phase_Seg2	[1 … 8] t_q	May be shortened temporarily by synchronization
Synchronization Jump Width (SJW)	[1 … 4] t_q	May not be longer than either Phase Buffer Segment

NOTE: For proper functionality of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

23.3.16.1.1 Synchronization Segment

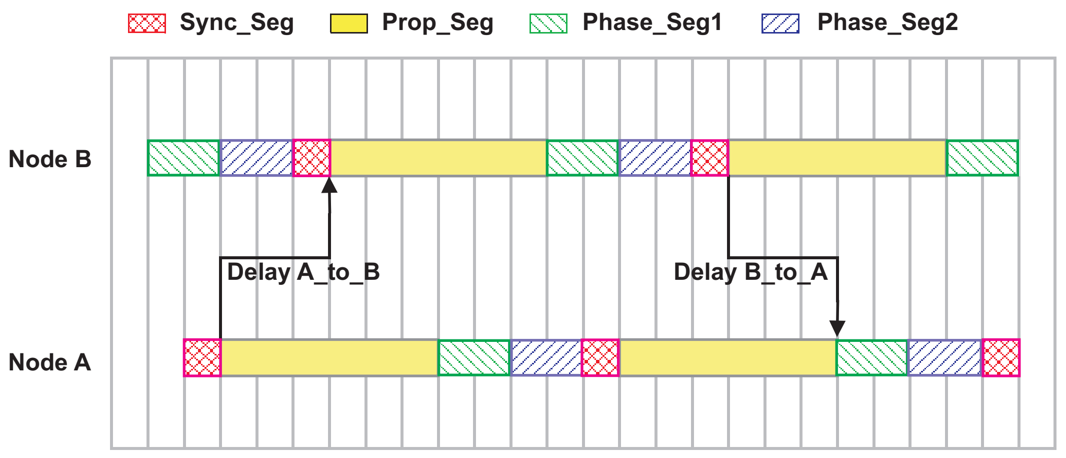
The synchronization segment (Sync_Seg) is the part of the bit time where edges of the CAN bus level are expected to occur. If an edge occurs outside of Sync_Seg, its distance to the Sync_Seg is called the phase error of this edge.

23.3.16.1.2 Propagation Time Segment

This part of the bit time is used to compensate physical delay times within the CAN network. These delay times consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

Any CAN node synchronized to the bit stream on the CAN bus can be out of phase with the transmitter of the bit stream, caused by the signal propagation time between the two nodes. The CAN protocol's nondestructive bitwise arbitration and the dominant acknowledge bit provided by receivers of CAN messages require that a CAN node transmitting a bit stream must also be able to receive dominant bits transmitted by other CAN nodes that are synchronized to that bit stream. The example in Figure 23-14 shows the phase shift and propagation times between two CAN nodes.

Figure 23-14. The Propagation Time Segment



Delay A_to_B >= node output delay(A) + bus line delay(AÆB) + node input delay(B)

Prop_Seg >= Delay A_to_B + Delay B_to_A

Prop_Seg >= 2 • [max(node output delay+bus line delay + node input delay)]

In this example, both nodes A and B are transmitters performing an arbitration for the CAN bus. The node A has sent its start of frame bit less than one bit time earlier than node B, therefore node B has synchronized itself to the received edge from recessive to dominant. Since node B has received this edge delay(A_to_B) after it has been transmitted, node B's bit timing segments are shifted with regard to node A. Node B sends an identifier with higher priority and so it will win the arbitration at a specific identifier bit when it transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B will arrive at node A after the delay (B_to_A).

Due to oscillator tolerances, the actual position of node A's sample point can be anywhere inside the nominal range of node A's Phase Buffer Segments, so the bit transmitted by node B must arrive at node A before the start of Phase_Seg1. This condition defines the length of Prop_Seg.

If the edge from recessive to dominant transmitted by node B would arrive at node A after the start of Phase_Seg1, it could happen that node A samples a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

This error only occurs when two nodes arbitrate for the CAN bus which have oscillators of opposite ends of the tolerance range and are separated by a long bus line; this is an example of a minor error in the bit timing configuration (Prop_Seg too short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3 Sample Mode. The DCAN does not. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of 1 t_q, requiring a longer Prop_Seg.

23.3.16.1.3 Phase Buffer Segments and Synchronization

The phase buffer segments (Phase_Seg1 and Phase_Seg2) and the synchronization jump width (SJW) are used to compensate for the oscillator tolerance.

The phase buffer segments surround the sample point and may be lengthened or shortened by synchronization.

The synchronization jump width (SJW) defines how far the resynchronizing mechanism may move the sample point inside the limits defined by the phase buffer segments to compensate for edge phase errors.

Synchronizations occur on edges from recessive to dominant. Their purpose is to control the distance between edges and sample points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous sample point. A synchronization may be done only if a recessive bit was sampled at the previous sample point and if the actual time quantum's bus level is dominant.

An edge is synchronous if it occurs inside of Sync_Seg; otherwise, its distance to the Sync_Seg is the edge phase error, measured in time quanta. If the edge occurs before Sync_Seg, the phase error is negative, else it is positive.

Two types of synchronization exist: hard synchronization and resynchronizing. A hard synchronization is done once at the start of a frame; inside a frame, only resynchronization is possible.

- Hard Synchronization

  After a hard synchronization, the bit time is restarted with the end of Sync_Seg, regardless of the edge phase error. Thus hard synchronization forces the edge which has caused the hard synchronization, to lie within the synchronization segment of the restarted bit time.

- Bit Resynchronizations

  Resynchronization leads to a shortening or lengthening of the Bit time such that the position of the sample point is shifted with regard to the edge.

  When the phase error of the edge which causes resynchronization is positive, Phase_Seg1 is lengthened. If the magnitude of the phase error is less than SJW, Phase_Seg1 is lengthened by the magnitude of the phase error, else it is lengthened by SJW.

  When the phase error of the edge which causes Resynchronization is negative, Phase_Seg2 is shortened. If the magnitude of the phase error is less than SJW, Phase_Seg2 is shortened by the magnitude of the phase error, else it is shortened by SJW.

If the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of hard synchronization and resynchronization are the same. If the magnitude of the phase error is larger than SJW, the resynchronization cannot compensate the phase error completely, and an error of (phase error - SJW) remains.

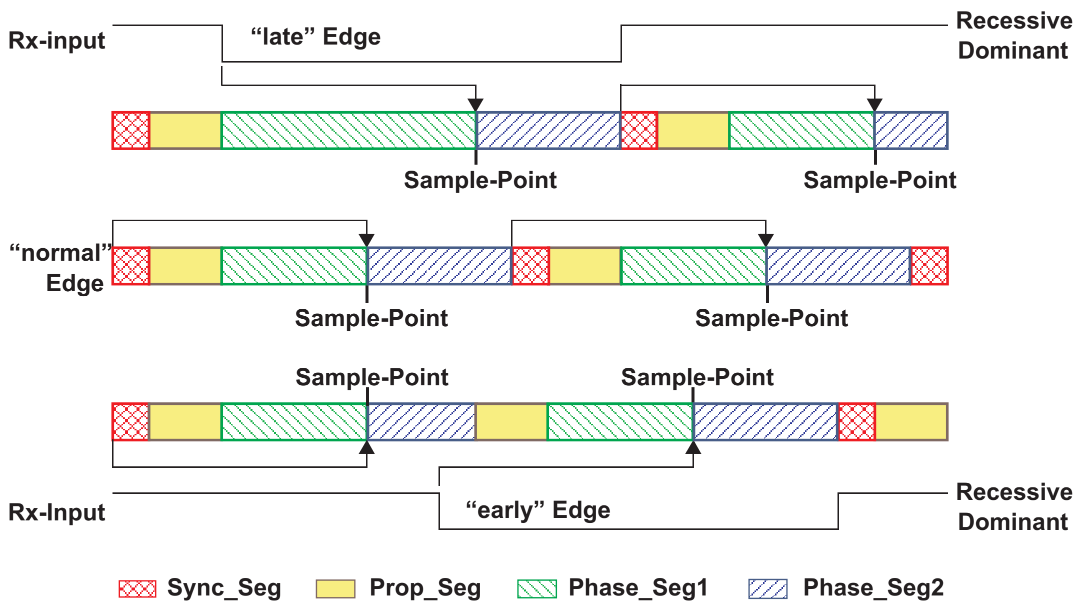
Only one synchronization may be done between two sample points. The synchronizations maintain a minimum distance between edges and sample points, giving the bus level time to stabilize and filtering out spikes that are shorter than (Prop_Seg + Phase_Seg1).

Apart from noise spikes, most synchronizations are caused by arbitration. All nodes synchronize “hard” on the edge transmitted by the “leading” transceiver that started transmitting first, but due to propagation delay times, they cannot become ideally synchronized. The leading transmitter does not necessarily win the arbitration; therefore, the receivers have to synchronize themselves to different transmitters that subsequently take the lead and that are differently synchronized to the previously leading transmitter. The same happens at the acknowledge field, where the transmitter and some of the receivers will have to synchronize to that receiver that takes the lead in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration will be caused by oscillator tolerance, when the differences in the oscillator’s clock periods of transmitter and receivers sum up during the time between synchronizations (at most ten bits). These summarized differences may not be longer than the SJW, limiting the oscillator’s tolerance range.

Figure 23-15 shows how the phase buffer segments are used to compensate for phase errors. There are three drawings of each two consecutive bit timings. The upper drawing shows the synchronization on a “late” edge, the lower drawing shows the synchronization on an “early” edge, and the middle drawing is the reference without synchronization.

Figure 23-15. Synchronization on Late and Early Edges



In the first example, an edge from recessive to dominant occurs at the end of Prop_Seg. The edge is "late" since it occurs after the Sync_Seg. Reacting to the late edge, Phase_Seg1 is lengthened so that the distance from the edge to the sample point is the same as it would have been from the Sync_Seg to the sample point if no edge had occurred. The phase error of this late edge is less than SJW, so it is fully compensated and the edge from dominant to recessive at the end of the bit, which is one nominal bit time long, occurs in the Sync_Seg.

In the second example, an edge from recessive to dominant occurs during Phase_Seg2. The edge is "early" since it occurs before a Sync_Seg. Reacting to the early edge, Phase_Seg2 is shortened and Sync_Seg is omitted, so that the distance from the edge to the sample point is the same as it would have been from a Sync_Seg to the sample point if no edge had occurred. As in the previous example, the magnitude of this early edge’s phase error is less than SJW, so it is fully compensated.

The phase buffer segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

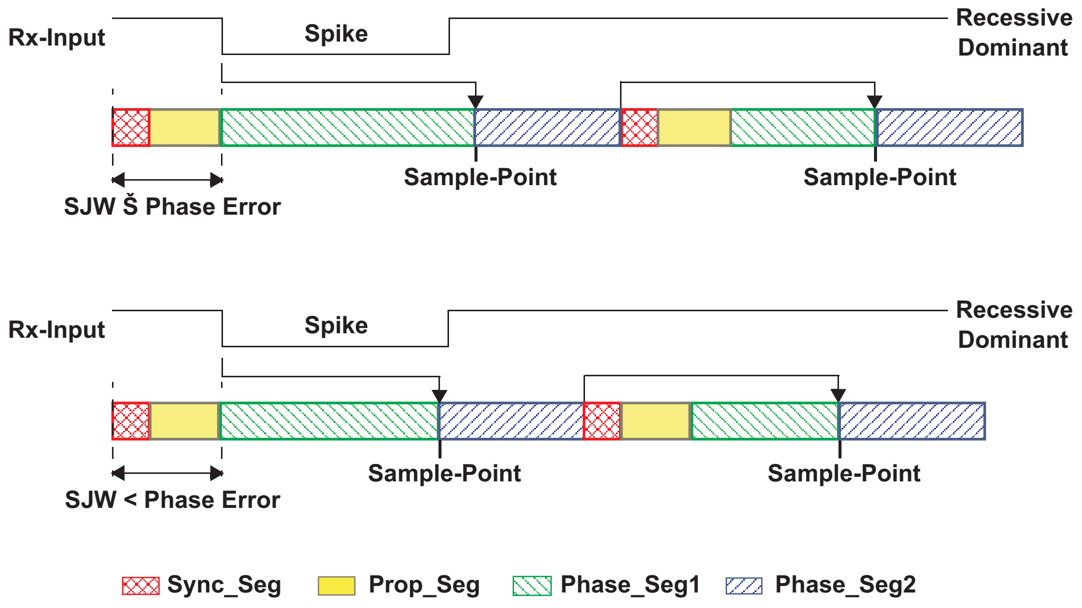
In these examples, the bit timing is seen from the point of view of the CAN implementation's state machine, where the bit time starts and ends at the sample points. The state machine omits Sync_Seg when synchronizing on an early edge because it cannot subsequently redefine that time quantum of Phase_Seg2 where the edge occurs to be the Sync_Seg.

Figure 23-16 shows how short dominant noise spikes are filtered by synchronizations. In both examples, the spike starts at the end of Prop_Seg and has the length of (Prop_Seg + Phase_Seg1).

In the first example, the synchronization jump width is greater than or equal to the phase error of the spike's edge from recessive to dominant. Therefore the sample point is shifted after the end of the spike; a recessive bus level is sampled.

In the second example, SJW is shorter than the phase error, so the sample point cannot be shifted far enough; the dominant spike is sampled as actual bus level.

Figure 23-16. Filtering of Short Dominant Spikes



23.3.16.1.4 Oscillator Tolerance Range

With the introduction of CAN protocol version 1.2, the option to synchronize on edges from dominant to recessive became obsolete. Only edges from recessive to dominant are considered for synchronization. The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range df for an oscillator's frequency f_{osc} around the nominal frequency f_{nom} with:

$$(1 - df) \bullet f_{nom} \le f_{osc} \le (1 + df) \bullet f_{nom}$$

depends on the proportions of Phase_Seg1, Phase_Seg2, SJW, and the bit time. The maximum tolerance df is the defined by two conditions (both shall be met):

$$\text{I:} \quad df \le \frac{\min(TSeg1, TSeg2)}{2x(13x(bit\_time - TSeg2))}$$

$$\text{II:} \quad df \le \frac{SJW}{20xbit\_time}$$

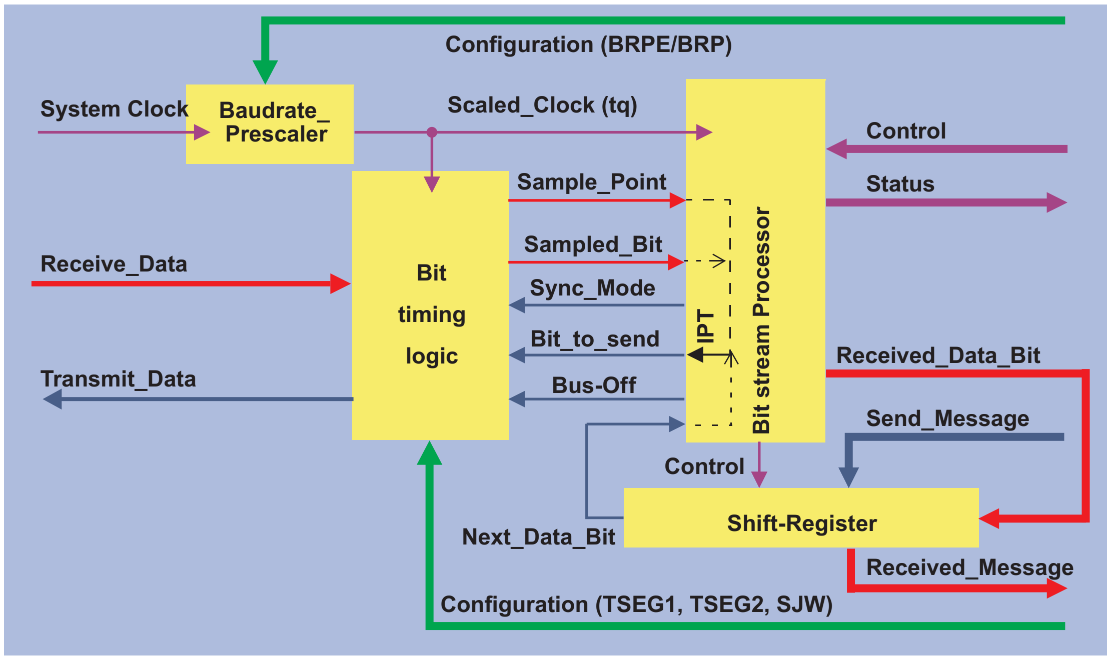
It has to be considered that SJW may not be larger than the smaller of the phase buffer segments and that the propagation time segment limits that part of the bit time that may be used for the phase buffer segments.

The combination Prop_Seg = 1 and Phase_Seg1 = Phase_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a propagation time segment of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 kBit/s (bit time = 8 μs) with a bus length of 40 m.

23.3.16.2 DCAN Bit Timing Registers

In the DCAN, the bit timing configuration is programmed in two register bytes, additionally a third byte for a baud rate prescaler extension of four bits (BREP) is provided. The sum of Prop_Seg and Phase_Seg1 (as TSEG1) is combined with Phase_Seg2 (as TSEG2) in one byte, SJW and BRP (plus BRPE in third byte) are combined in the other byte (see Figure 23-17).

Figure 23-17. Structure of the CAN Core's CAN Protocol Controller



In this bit timing register, the components TSEG1, TSEG2, SJW and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1…n], values in the range of [0…–1] are programmed. That way, e.g., SJW (functional range of [1…4]) is represented by only two bits.

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

The data in the bit timing register (BTR) is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by BRPE/BRP) defines the length of the time quantum (the basic time unit of the bit time); the bit timing logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the sample point, and occasional synchronizations are controlled by the bit timing state machine, which is evaluated once each time quantum. The rest of the CAN protocol controller, the bit stream processor (BSP) state machine, is evaluated once each bit time, at the sample point.

The shift register serializes the messages to be sent and parallelizes received messages. Its loading and shifting is controlled by the BSP. The BSP translates messages into frames and vice versa. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the sample point and processes the sampled bus input bit. The time after the sample point that is needed to calculate the next bit to be sent (e.g., data bit, CRC bit, stuff bit, error flag, or idle) is called the information processing time (IPT), which is 0 tq for the DCAN.

Generally, the IPT is CAN controller-specific, but may not be longer than 2 tq. The IPC length is the lower limit of the programmed length of Phase_Seg2. In case of a synchronization, Phase_Seg2 may be shortened to a value less than IPT, which does not affect bus timing.

23.3.16.2.1 Calculation of the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting bit time (1 / Bit rate) must be an integer multiple of the CAN clock period.

NOTE: 8 MHz is the minimum CAN clock frequency required to operate the DCAN at a bit rate of 1 MBit/s.

The bit time may consist of 8 to 25 time quanta. The length of the time quantum t_q is defined by the baud rate prescaler with t_q = (Baud Rate Prescaler) / CAN_CLK. Several combinations may lead to the desired bit time, allowing iterations of the following steps.

First part of the bit time to be defined is the Prop_Seg. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandible CAN bus systems. The resulting time for Prop_Seg is converted into time quanta (rounded up to the nearest integer multiple of t_q).

The Sync_Seg is 1 t_q long (fixed), leaving (bit time – Prop_Seg – 1) t_q for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, Phase_Seg2 = Phase_Seg1, else Phase_Seg2 = Phase_Seg1 + 1.

The minimum nominal length of Phase_Seg2 has to be regarded as well. Phase_Seg2 may not be shorter than any CAN controller's Information Processing Time in the network, which is device dependent and can be in the range of [0…2] t_q.

The length of the synchronization jump width is set to its maximum value, which is the minimum of four (4) and Phase_Seg1.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formulas given in Section 23.3.16.2.3.

If more than one configurations are possible to reach a certain Bit rate, it is recommended to choose the configuration which allows the highest oscillator tolerance range.

CAN nodes with different clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol compliant configuration of the CAN bit timing.

The resulting configuration is written into the bit timing register:

Tseg2 = Phase_Seg2-1

Tseg1 = Phase_Seg1+ Prop_Seg-1

SJW = SynchronizationJumpWidth-1

BRP = Prescaler-1

23.3.16.2.2 Example for Bit Timing at High Baud Rate

In this example, the frequency of CAN_CLK is 10 MHz, BRP is 0, the bit rate is 1 MBit/s.

t_q	100 ns	=	t_{CAN_CLK}
delay of bus driver	60 ns	=	
delay of receiver circuit	40 ns	=	
delay of bus line (40 m)	220 ns	=	
t_{Prop}	700 ns	=	INT (2*delays + 1) = 7 • t_q
t_{SJW}	100 ns	=	1 • t_q
t_{TSeg1}	800 ns	=	$t_{Prop} + t_{SJW}$
t_{TSeg2}	100 ns	=	Information Processing Time + 1 • t_q
$t_{Sync-Seg}$	100 ns	=	1 • t_q
bit time	1000 ns	=	$t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$
tolerance for CAN_CLK	0.43 %	=	$\frac{(\min(TSeg1, TSeg2))}{2x(13x(bit\_time - TSeg2))}$
		=	$\frac{0.1\,\mu s}{2x(13x(1\,\mu s - 0.1\,\mu s))}$

In this example, the concatenated bit time parameters are $(1-1)_3\&(8-1)_4\&(1-1)_2\&(1-1)_6$, so the bit timing register is programmed to = 00000700.

23.3.16.2.3 Example for Bit Timing at Low Baud Rate

In this example, the frequency of CAN_CLK is 2 MHz, BRP is 1, the bit rate is 100 KBit/s.

t_q	1 µs	=	t_{CAN_CLK}
Delay of bus driver	200 ns	=	
Delay of receiver circuit	80 ns	=	
Delay of bus line (40 m)	220 ns	=	
t_{Prop}	1 µs	=	1 • t_q
t_{SJW}	4 µs	=	4 • t_q
t_{TSeg1}	5 µs	=	$t_{Prop} + t_{SJW}$
t_{TSeg2}	3 µs	=	Information Processing Time + 3 • t_q
$t_{Sync-Seg}$	1 µs	=	1 • t_q
Bit time	9 µs	=	$t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$
Tolerance for CAN_CLK		=	$\frac{(\min(TSeg1, TSeg2))}{2x(13x(bit\_time - TSeg2))}$
		=	$\frac{0.1\,\mu s}{2x(13x(1\,\mu s - 0.1\,\mu s))}$

In this example, the concatenated bit time parameters are $(3-1)_3\&(5-1)_4\&(4-1)_2\&(2-1)_6$, so the bit timing register is programmed to = 0x000024C1.

23.3.17 Message Interface Register Sets

The interface register sets control the CPU read and write accesses to the message RAM. There are two interface registers sets for read/write access, IF1 and IF2 and one interface register set for read access only, IF3.

Due to the structure of the message RAM, it is not possible to change single bits or bytes of a message object. Instead, always a complete message object in the message RAM is accessed. Therefore the data transfer from the IF1/IF2 registers to the message RAM requires the message handler to perform a read-modify-write cycle: First those parts of the message object that are not to be changed are read from the message RAM into the interface register set, and after the update the whole content of the interface register set is written into the message object.

After the partial write of a message object, those parts of the interface register set that are not selected in the command register, will be set to the actual contents of the selected message object.

After the partial read of a message object, those parts of the interface register set that are not selected in the command register, will be left unchanged.

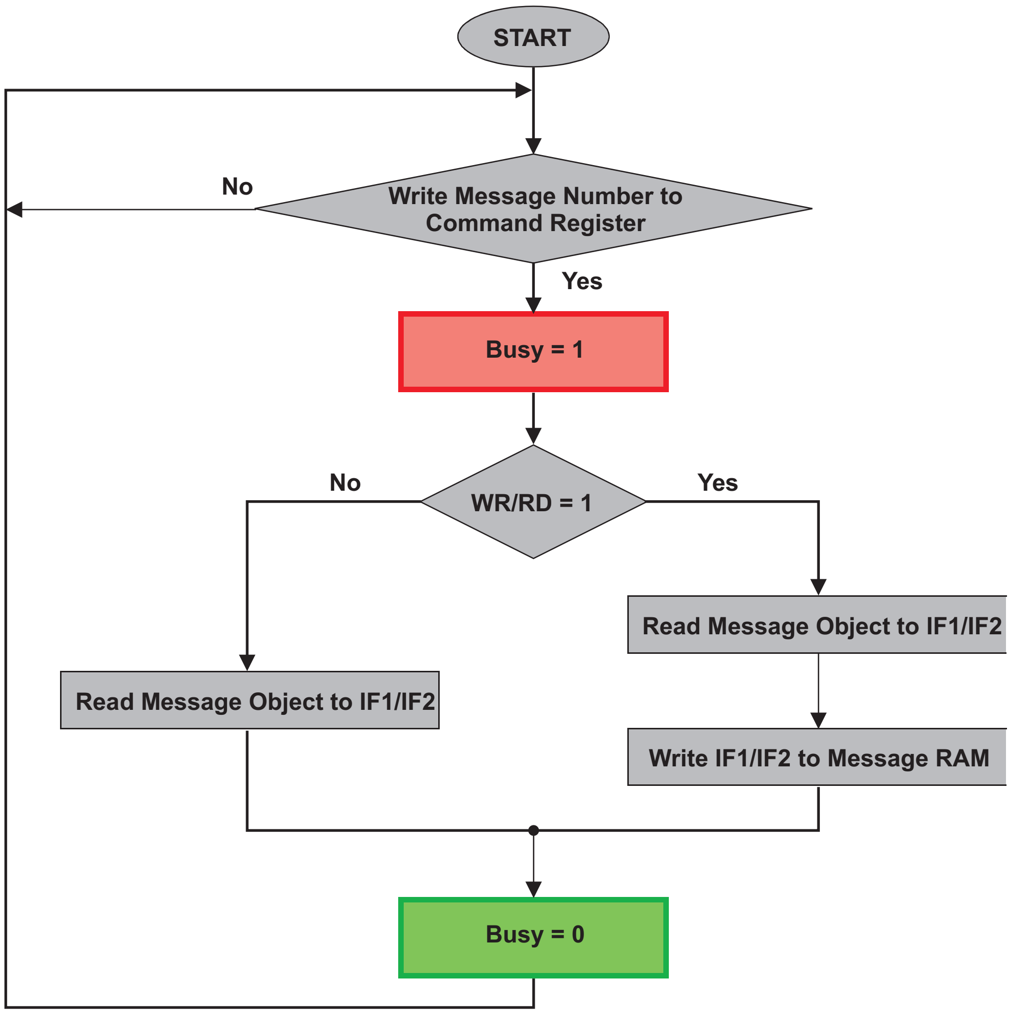
By buffering the data to be transferred, the interface register sets avoid conflicts between concurrent CPU accesses to the message RAM and CAN message reception and transmission. A complete message object (see Section 23.3.18.1) or parts of the message object may be transferred between the message RAM and the IF1/IF2 register set (see) in one single transfer. This transfer, performed in parallel on all selected parts of the message object, guarantees the data consistency of the CAN message.

23.3.17.1 Message Interface Register Sets 1 and 2

The IF1 and IF2 register sets control the data transfer to and from the message object. The IF1CMD and IF2CMD registers address the desired message object in the message RAM and specify whether a complete message object or only parts should be transferred. The data transfer is initiated by writing the message number to the bits [7:0] of the IF1CMD and IF2CMD register.

When the CPU initiates a data transfer between the IF1/IF2 registers and message RAM, the message handler sets the busy bit in the respective command register to ‘1’. After the transfer has completed, the busy bit is set back to ‘0’ (see Figure 23-18).

Figure 23-18. Data Transfer Between IF1/IF2 Registers and Message RAM


START
No
Write Message Number to Command Register
Yes
Busy = 1
No
WR/RD = 1
Yes
Read Message Object to IF1/IF2
Read Message Object to IF1/IF2
Write IF1/IF2 to Message RAM
Busy = 0

23.3.17.2 IF3 Register Set

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from message RAM by CPU. The intention of this feature of IF3 is to provide an interface for the DMA to read packets efficiently. The automatic update functionality can be programmed for each message object using the update enable registers IF3UPD12 to IF3UPD78.

All valid message objects in message RAM which are configured for automatic update, will be checked for active NewDat flags. If such a message object is found, it will be transferred to the IF3 register (if no previous DMA transfers are ongoing), controlled by IF3 Observation register (IF3OBS). If more than one NewDat flag is active, the message object with the lowest number has the highest priority for automatic IF3 update.

The NewDat bit in the message object will be reset by a transfer to IF3.

If DCAN internal IF3 update is complete, a DMA request is generated. The DMA request stays active until first read access to one of the IF3 registers. The DMA functionality has to be enabled by setting bit DE3 in the CTL register. Please refer to the device datasheet to find out if this DMA source is available.

NOTE: The IF3 register set can not be used for transferring data into message objects.

23.3.18 Message RAM

The DCAN message RAM contains message objects and parity bits for the message objects. There are up to 64 message objects in the message RAM.

During normal operation, accesses to the message RAM are performed via the interface register sets, and the CPU cannot directly access the message RAM.

The interface register sets IF1 and IF2 provide indirect read/write access from the CPU to the message RAM. The IF1 and IF2 register sets can buffer control and user data to be transferred to and from the message objects.

The third interface register set IF3 can be configured to automatically receive control and user data from the message RAM when a message object has been updated after reception of a CAN message. The CPU does not need to initiate the transfer from message RAM to IF3 register set.

The message handler avoids potential conflicts between concurrent accesses to message RAM and CAN frame reception/transmission.

There are two modes where the message RAM can be directly accessed by the CPU:

- Debug/Suspend mode (see Section 23.3.18.3)
- RAM Direct Access (RDA) mode (see Section 23.3.18.4)

23.3.18.1 Structure of Message Objects

Table 23-8 shows the structure of a message object.

The grayed fields are those parts of the message object which are represented in dedicated registers. For example, the transmit request flags of all message objects are represented in centralized transmit request registers.

Table 23-8. Structure of a Message Object

Message Object													
UMask	**Msk[28:0]**	**MXtd**	**MDir**	**EoB**	**unused**	**NewDat**	**MsgLst**	**RxIE**	**TxIE**	**IntPnd**	**RmtEn**	**TxRqst**	
MsgVal	ID[28:0]	Xtd	Dir	DLC[3:0]	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	

Table 23-9. Field Descriptions

Name	Value	Description
MsgVal		Message valid
	0	The message object is ignored by the message handler.
	1	The message object is to be used by the message handler.
		Note: This bit may be kept at level ‘1’ even when the identifier bits ID[28:0], the control bits Xtd, Dir, or the data length code are changed. It should be reset if the Messages Object is no longer required.
UMask		Use acceptance mask
	0	Mask bits (Msk[28:0], MXtd and MDir) are ignored and not used for acceptance filtering.
	1	Mask bits are used for acceptance filtering.
		Note: If the UMask bit is set to one, the message object’s mask bits have to be programmed during initialization of the message object before MsgVal is set to one.
ID[28:0]		Message identifier
	ID[28:0]	29-bit (“extended”) identifier bits
	ID[28:18]	11-bit (“standard”) identifier bits
Msk[28:0]		Identifier mask
	0	The corresponding bit in the message identifier is not used for acceptance filtering (don’t care).
	1	The corresponding bit in the message identifier is used for acceptance filtering.
Xtd		Mask extended identifier
	0	The extended identifier bit (IDE) has no effect on the acceptance filtering.
	1	The extended identifier bit (IDE) is used for acceptance filtering.
		Note: When 11-bit (“standard”) Identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.
Dir		Message direction
	0	Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, the message is stored in this message object.
	1	Direction = transmit: On TxRqst, a data frame is transmitted. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = one).
MDir		Mask message direction
	0	The message direction bit (Dir) has no effect on the acceptance filtering.
	1	The message direction bit (Dir) is used for acceptance filtering.
EOB		End of block
	0	The message object is part of a FIFO Buffer block and is not the last message object of this FIFO Buffer block.
	1	The message object is a single message object or the last message object in a FIFO Buffer Block.
		Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.

Table 23-9. Field Descriptions (continued)

Name	Value	Description
NewDat		New data
	0	No new data has been written into the data bytes of this message object by the message handler since the last time when this flag was cleared by the CPU.
	1	The message handler or the CPU has written new data into the data bytes of this message object.
MsgLst		Message lost (only valid for message objects with direction = receive)
	0	No message was lost since the last time when this bit was reset by the CPU.
	1	The message handler stored a new message into this message object when NewDat was still set, so the previous message has been overwritten.
RxIE		Receive interrupt enable
	0	IntPnd will not be triggered after the successful reception of a frame.
	1	IntPnd will be triggered after the successful reception of a frame.
TxIE		Transmit interrupt enable
	0	IntPnd will not be triggered after the successful transmission of a frame.
	1	IntPnd will be triggered after the successful transmission of a frame.
IntPnd		Interrupt pending
	0	This message object is not the source of an interrupt.
	1	This message object is the source of an interrupt. The interrupt Identifier in the interrupt register will point to this message object if there is no other interrupt source with higher priority.
RmtEn		Remote enable
	0	At the reception of a remote frame, TxRqst is not changed.
	1	At the reception of a remote frame, TxRqst is set.
TxRqst		Transmit request
	0	This message object is not waiting for a transmission.
	1	The transmission of this message object is requested and is not yet done.
DLC[3:0]		Data length code
	0	Data frame has 0 to 8 data bytes.
	1	Data frame has 8 data bytes.
		Note: The data length code of a message object must be defined to the same value as in the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.
Data 0		1st data byte of a CAN data frame
Data 1		2nd data byte of a CAN data frame
Data 2		3rd data byte of a CAN data frame
Data 3		4th data byte of a CAN data frame
Data 4		5th data byte of a CAN data frame
Data 5		6th data byte of a CAN data frame
Data 6		7th data byte of a CAN data frame
Data 7		8th data byte of a CAN data frame
		Note: Byte Data 0 is the first data byte shifted into the shift register of the CAN core during a reception, byte Data 7 is the last. When the message handler stores a data frame, it will write all the eight data bytes into a message object. If the data length code is less than 8, the remaining bytes of the message object may be overwritten by undefined values.

23.3.18.2 Addressing Message Objects in RAM

The starting location of a particular message object in RAM is:
Message RAM base address + (message object number) * 0x20

This means that message object 1 starts at offset 0x0020; message object 2 starts at offset 0x0040, etc.

‘0’ is not a valid message object number.

The base address for DCAN0 RAM is 0x481C_D000 and DCAN1 RAM is 0x481D_1000.

Message object number 1 has the highest priority.

Table 23-10. Message RAM addressing in Debug/Suspend and RDA Mode

Message Object Number	Offset From Base Address	Word Number	Debug/Suspend Mode, see Section 23.3.18.3	RDA mode, see Section 23.3.18.4
1	0x0020	1	Parity	Data Bytes 4-7
	0x0024	2	MXtd,MDir,Mask	Data Bytes 0-3
	0x0028	3	Xtd,Dir,ID	ID[27:0],DLC
	0x002C	4	Ctrl	Mask,Xtd,Dir,ID[28]
	0x0030	5	Data Bytes 3-0	Parity,Ctrl,MXtd,MDir
	0x0034	6	Data Bytes 7-4	…
…	…	…	…	…
31	0x03E0	1	Parity	Data Bytes 4-7
	0x03E4	2	MXtd,MDir,Mask	Data Bytes 0-3
	0x03E8	3	Xtd,Dir,ID	ID[27:0],DLC
	0x03EC	4	Ctrl	Mask,Xtd,Dir,ID[28]
	0x03F0	5	Data Bytes 3-0	Parity,Ctrl,MXtd,MDir
	0x03F4	6	Data Bytes 7-4	…
…	…	…	…	…
63	0x07E0	1	Parity	Data Bytes 4-7
	0x07E4	2	MXtd,MDir,Mask	Data Bytes 0-3
	0x07E8	3	Xtd,Dir,ID	ID[27:0],DLC
	0x07EC	4	Ctrl	Mask,Xtd,Dir,ID[28]
	0x07F0	5	Data Bytes 3-0	Parity,Ctrl,MXtd,MDir
	0x07F4	6	Data Bytes 7-4	…
last implemented (32(DCAN3) or 64)	0x0000	1	Parity	Data Bytes 4-7
	0x0004	2	MXtd,MDir,Mask	Data Bytes 0-3
	0x0008	3	Xtd,Dir,ID	ID[27:0],DLC
	0x000C	4	Ctrl	Mask,Xtd,Dir,ID[28]
	0x0010	5	Data Bytes 3-0	Parity,Ctrl,MXtd,MDir
	0x0014	6	Data Bytes 7-4	…

23.3.18.3 Message RAM Representation in Debug/Suspend Mode

In debug/suspend mode, the message RAM will be memory mapped. This allows the external debug unit to access the message RAM.

NOTE: During debug/suspend mode, the message RAM cannot be accessed via the IFx register sets.

Table 23-11. Message RAM Representation in Debug/Suspend Mode

Bit #	31/ 15	30/ 14	29/ 13	29/ 12	27/ 11	26/ 10	25/ 9	24/ 8	23/ 7	22/ 6	21/ 5	20/ 4	19/ 3	18/ 2	17/ 1	16/ 0
MsgAddr + 0x00	Reserved															
	Reserved											Parity[4:0]				
MsgAddr + 0x04	MXtd	MDir	Rsvd	Msk[28:16]												
	Msk[15:0]															
MsgAddr + 0x08	Rsvd	Xtd	Dir	ID[28:16]												
	ID[15:0]															
MsgAddr + 0x0C	Reserved															
	Rsvd	MsgLst	Rsvd	UMask	TxIE	RxTE	RmtEn	Rsvd	EOB	Reserved			DLC[3:0]			
MsgAddr + 0x10	Data 3								Data 2							
	Data 1								Data 0							
MsgAddr + 0x14	Data 7								Data 6							
	Data 5								Data 4							

23.3.18.4 Message RAM Representation in Direct Access Mode

When the RDA bit in the TEST register is set while the DCAN module is in test mode (test bit in the CTL register is set), the CPU has direct access to the message RAM. Due to the 32-bit bus structure, the RAM is split into word lines to support this feature. The CPU has access to one word line at a time only.

In RAM direct access mode, the RAM is represented by a continuous memory space within the address frame of the DCAN module, starting at the message RAM base address.

Note: During direct access mode, the message RAM cannot be accessed via the IFx register sets.

Any read or write to the RAM addresses for RamDirectAccess during normal operation mode (TestMode bit or RDA bit not set) will be ignored.

Table 23-12. Message RAM Representation in RAM Direct Access Mode

Bit #	31/ 15	30/ 14	29/ 13	29/ 12	27/ 11	26/ 10	25/ 9	24/ 8	23/ 7	22/ 6	21/ 5	20/ 4	19/ 3	18/ 2	17/ 1	16/ 0
MsgAddr + 0x00	Data 4								Data 5							
	Data 6								Data 7							
MsgAddr + 0x04	Data 0								Data 1							
	Data 2								Data 3							
MsgAddr + 0x08	ID[27:12]															
	ID[11:0]												DLC[3:0]			
	Msk[28:13]															

Table 23-12. Message RAM Representation in RAM Direct Access Mode (continued)

Bit #	31/ 15	30/ 14	29/ 13	29/ 12	27/ 11	26/ 10	25/ 9	24/ 8	23/ 7	22/ 6	21/ 5	20/ 4	19/ 3	18/ 2	17/ 1	16/ 0
MsgAddr + 0x0C	Msk[12:0]													Xtd	Dir	ID[28]
MsgAddr + 0x10	Reserved															Parity [4]
	MsgLst				Unused				Msg Lst	UMa sk	TxIE	RxT E	Rmt En	EOB	MX td	MDir

NOTE: Writes to unused bits have no effect.

23.3.19 GIO Support

The CAN_RX and CAN_TX pins of the DCAN module can be used as general purpose IO pins, if CAN functionality is not needed. This function is controlled by the CAN TX IO control register (TIOC) and the CAN RX IO control register (RIOC).

23.4 Registers

23.4.1 DCAN Registers

Table 23-13 lists the memory-mapped registers for the DCAN. All register offset addresses not listed in Table 23-13 should be considered as reserved locations and the register contents should not be modified.

Table 23-13. DCAN Registers

Offset	Acronym	Register Name	Section
0h	CTL	CAN Control Register	Section 23.4.1.1
4h	ES	Error and Status Register	Section 23.4.1.2
8h	ERRC	Error Counter Register	Section 23.4.1.3
Ch	BTR	Bit Timing Register	Section 23.4.1.4
10h	INT	Interrupt Register	Section 23.4.1.5
14h	TEST	Test Register	Section 23.4.1.6
1Ch	PERR	Parity Error Code Register	Section 23.4.1.7
80h	ABOTR	Auto-Bus-On Time Register	Section 23.4.1.8
84h	TXRQ_X	Transmission Request X Register	Section 23.4.1.9
88h	TXRQ12	Transmission Request Register 12	Section 23.4.1.10
8Ch	TXRQ34	Transmission Request Register 34	Section 23.4.1.11
90h	TXRQ56	Transmission Request Register 56	Section 23.4.1.12
94h	TXRQ78	Transmission Request Register 78	Section 23.4.1.13
98h	NWDAT_X	New Data X Register	Section 23.4.1.14
9Ch	NWDAT12	New Data Register 12	Section 23.4.1.15
A0h	NWDAT34	New Data Register 34	Section 23.4.1.16
A4h	NWDAT56	New Data Register 56	Section 23.4.1.17
A8h	NWDAT78	New Data Register 78	Section 23.4.1.18
ACh	INTPND_X	Interrupt Pending X Register	Section 23.4.1.19
B0h	INTPND12	Interrupt Pending Register 12	Section 23.4.1.20
B4h	INTPND34	Interrupt Pending Register 34	Section 23.4.1.21
B8h	INTPND56	Interrupt Pending Register 56	Section 23.4.1.22
BCh	INTPND78	Interrupt Pending Register 78	Section 23.4.1.23
C0h	MSGVAL_X	Message Valid X Register	Section 23.4.1.24
C4h	MSGVAL12	Message Valid Register 12	Section 23.4.1.25
C8h	MSGVAL34	Message Valid Register 34	Section 23.4.1.26
CCh	MSGVAL56	Message Valid Register 56	Section 23.4.1.27
D0h	MSGVAL78	Message Valid Register 78	Section 23.4.1.28
D8h	INTMUX12	Interrupt Multiplexer Register 12	Section 23.4.1.29
DCh	INTMUX34	Interrupt Multiplexer Register 34	Section 23.4.1.30
E0h	INTMUX56	Interrupt Multiplexer Register 56	Section 23.4.1.31
E4h	INTMUX78	Interrupt Multiplexer Register 78	Section 23.4.1.32
100h	IF1CMD	IF1 Command Registers	Section 23.4.1.33
104h	IF1MSK	IF1 Mask Register	Section 23.4.1.34
108h	IF1ARB	IF1 Arbitration Register	Section 23.4.1.35
10Ch	IF1MCTL	IF1 Message Control Register	Section 23.4.1.36
110h	IF1DATA	IF1 Data A Register	Section 23.4.1.37
114h	IF1DATB	IF1 Data B Register	Section 23.4.1.38
120h	IF2CMD	IF2 Command Registers	Section 23.4.1.39
124h	IF2MSK	IF2 Mask Register	Section 23.4.1.40
128h	IF2ARB	IF2 Arbitration Register	Section 23.4.1.41
12Ch	IF2MCTL	IF2 Message Control Register	Section 23.4.1.42

Table 23-13. DCAN Registers (continued)

Offset	Acronym	Register Name	Section
130h	IF2DATA	IF2 Data A Register	Section 23.4.1.43
134h	IF2DATB	IF2 Data B Register	Section 23.4.1.44
140h	IF3OBS	IF3 Observation Register	Section 23.4.1.45
144h	IF3MSK	IF3 Mask Register	Section 23.4.1.46
148h	IF3ARB	IF3 Arbitration Register	Section 23.4.1.47
14Ch	IF3MCTL	IF3 Message Control Register	Section 23.4.1.48
150h	IF3DATA	IF3 Data A Register	Section 23.4.1.49
154h	IF3DATB	IF3 Data B Register	Section 23.4.1.50
160h	IF3UPD12	IF3 Update Enable Register 12	Section 23.4.1.51
164h	IF3UPD34	IF3 Update Enable Register 34	Section 23.4.1.52
168h	IF3UPD56	IF3 Update Enable Register 56	Section 23.4.1.53
16Ch	IF3UPD78	IF3 Update Enable Register 78	Section 23.4.1.54
1E0h	TIOC	CAN TX IO Control Register	Section 23.4.1.55
1E4h	RIOC	CAN RX IO Control Register	Section 23.4.1.56

23.4.1.1 CTL Register (offset = 0h) [reset = 1401h]

CTL is shown in Figure 23-19 and described in Table 23-14.

The Bus-Off recovery sequence (refer to CAN specification) cannot be shortened by setting or resetting Init bit. If the module goes Bus-Off, it will automatically set the Init bit and stop all bus activities. When the Init bit is cleared by the application again, the module will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the bus-off recovery sequence, the error counters will be reset. After the Init bit is reset, each time when a sequence of 11 recessive bits is monitored, a Bit0 error code is written to the error and status register, enabling the CPU to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the proceeding of the bus-off recovery sequence.

Figure 23-19. CTL Register

31	30	29	28	27	26	25	24
RESERVED						WUBA	PDR
R-0h						R/W-0h	R/W-0h
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED			DE3	DE2	DE1	IE1	InitDbg
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SWR	RESERVED	PMD				ABO	IDS
R/WP-0h	R-0h	R/W-5h				R/W-0h	R/W-0h
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Test	CCE	DAR	RESERVED	EIE	SIE	IE0	Init
R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-14. CTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-26	RESERVED	R	0h	
25	WUBA	R/W	0h	Automatic wake up on bus activity when in local power-down mode. Note: The CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power down and automatic wake-up mode, will be lost. 0h (R/W) = No detection of a dominant CAN bus level while in local power-down mode. 1h (R/W) = Detection of a dominant CAN bus level while in local power-down mode is enabled. On occurrence of a dominant CAN bus level, the wake up sequence is started.
24	PDR	R/W	0h	Request for local low power-down mode 0h (R/W) = No application request for local low power-down mode. If the application has cleared this bit while DCAN in local power-down mode, also the Init bit has to be cleared. 1h (R/W) = Local power-down mode has been requested by application. The DCAN will acknowledge the local power-down mode by setting bit PDA in the error and status register. The local clocks will be turned off by DCAN internal logic.
23-21	RESERVED	R	0h	
20	DE3	R/W	0h	Enable DMA request line for IF3. Note: A pending DMA request for IF3 remains active until first access to one of the IF3 registers. 0h (R/W) = Disabled 1h (R/W) = Enabled

Table 23-14. CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
19	DE2	R/W	0h	Enable DMA request line for IF2. Note: A pending DMA request for IF2 remains active until first access to one of the IF2 registers. 0h (R/W) = Disabled 1h (R/W) = Enabled
18	DE1	R/W	0h	Enable DMA request line for IF1. Note: A pending DMA request for IF1 remains active until first access to one of the IF1 registers. 0h (R/W) = Disabled 1h (R/W) = Enabled
17	IE1	R/W	0h	Interrupt line 1 enable 0h (R/W) = Disabled - Module interrupt DCAN1INT is always low. 1h (R/W) = Enabled - interrupts will assert line DCAN1INT to one; line remains active until pending interrupts are processed.
16	InitDbg	R	0h	Internal init state while debug access 0h (R/W) = Not in debug mode, or debug mode requested but not entered. 1h (R/W) = Debug mode requested and internally entered; the DCAN is ready for debug accesses.
15	SWR	R/WP	0h	SW reset enable. Note: To execute software reset, the following procedure is necessary: (a) Set Init bit to shut down CAN communication and (b) Set SWR bit additionally to Init bit. 0h (R/W) = Normal Operation 1h (R/W) = Module is forced to reset state. This bit will automatically get cleared after execution of SW reset after one OCP clock cycle.
14	RESERVED	R	0h	
13-10	PMD	R/W	5h	Parity on/off. 5 = Parity function disabled. Others = Parity function enabled.
9	ABO	R/W	0h	Auto-Bus-On enable 0h (R/W) = The Auto-Bus-On feature is disabled 1h (R/W) = The Auto-Bus-On feature is enabled
8	IDS	R/W	0h	Interruption debug support enable 0h (R/W) = When Debug/Suspend mode is requested, DCAN will wait for a started transmission or reception to be completed before entering Debug/Suspend mode 1h (R/W) = When Debug/Suspend mode is requested, DCAN will interrupt any transmission or reception, and enter Debug/Suspend mode immediately.
7	Test	R/W	0h	Test mode enable 0h (R/W) = Normal Operation 1h (R/W) = Test Mode
6	CCE	R/W	0h	Configuration change enable 0h (R/W) = The CPU has no write access to the configuration registers. 1h (R/W) = The CPU has write access to the configuration registers (when Init bit is set).
5	DAR	R/W	0h	Disable automatic retransmission 0h (R/W) = Automatic retransmission of not successful messages enabled. 1h (R/W) = Automatic retransmission disabled.
4	RESERVED	R	0h	

Table 23-14. CTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	EIE	R/W	0h	Error interrupt enable 0h (R/W) = Disabled - PER, BOff and EWarn bits can not generate an interrupt. 1h (R/W) = Enabled - PER, BOff and EWarn bits can generate an interrupt at DCAN0INT line and affect the interrupt register.
2	SIE	R/W	0h	Status change interrupt enable 0h (R/W) = Disabled - WakeUpPnd, RxOk, TxOk and LEC bits can not generate an interrupt. 1h (R/W) = Enabled - WakeUpPnd, RxOk, TxOk and LEC can generate an interrupt at DCAN0INT line and affect the interrupt register.
1	IE0	R/W	0h	Interrupt line 0 enable 0h (R/W) = Disabled - Module interrupt DCAN0INT is always low. 1h (R/W) = Enabled - interrupts will assert line DCAN0INT to one; line remains active until pending interrupts are processed.
0	Init	R/W	1h	Initialization 0h (R/W) = Normal operation 1h (R/W) = Initialization mode is entered

23.4.1.2 ES Register (offset = 4h) [reset = 6Fh]

ES is shown in Figure 23-20 and described in Table 23-15.

Interrupts are generated by bits PER, BOff and EWarn (if EIE bit in CAN control register is set) and by bits WakeUpPnd, RxOk, TxOk, and LEC (if SIE bit in CAN control register is set). A change of bit EPass will not generate an interrupt. Reading the error and status register clears the WakeUpPnd, PER, RxOk and TxOk bits and set the LEC to value '7.' Additionally, the status interrupt value (0x8000) in the interrupt register will be replaced by the next lower priority interrupt value. The EOI for all other interrupts (DCANINT0 and DCANINT1) are automatically handled by hardware. For debug support, the auto clear functionality of error and status register (clear of status flags by read) is disabled when in debug/suspend mode.

Figure 23-20. ES Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED					PDA	WakeUp_Pnd	PER_
R-0h					R-0h	0h	0h
7	6	5	4	3	2	1	0
BOff	EWarn	EPass	RxOk	TxOk	LEC		
R-0h	R-1h	R-1h	0h	1h	7h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-15. ES Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	
10	PDA	R	0h	Local power-down mode acknowledge 0h (R/W) = DCAN is not in local power-down mode. 1h (R/W) = Application request for setting DCAN to local power-down mode was successful. DCAN is in local power-down mode.
9	WakeUp_Pnd		0h	Wake up pending. This bit can be used by the CPU to identify the DCAN as the source to wake up the system. This bit will be reset if error and status register is read. 0h (R/W) = No Wake Up is requested by DCAN. 1h (R/W) = DCAN has initiated a wake up of the system due to dominant CAN bus while module power down.
8	PER_		0h	Parity error detected. This bit will be reset if error and status register is read. 0h (R/W) = No parity error has been detected since last read access. 1h (R/W) = The parity check mechanism has detected a parity error in the Message RAM.
7	BOff	R	0h	Bus-Off state 0h (R/W) = The CAN module is not bus-off state. 1h (R/W) = The CAN module is in bus-off state.
6	EWarn	R	1h	Warning state 0h (R/W) = Both error counters are below the error warning limit of 96. 1h (R/W) = At least one of the error counters has reached the error warning limit of 96.

Table 23-15. ES Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
5	EPass	R	1h	Error passive state 0h (R/W) = On CAN Bus error, the DCAN could send active error frames. 1h (R/W) = The CAN core is in the error passive state as defined in the CAN Specification.
4	RxOk		0h	Received a message successfully. This bit will be reset if error and status register is read. 0h (R/W) = No message has been successfully received since the last time when this bit was read by the CPU. This bit is never reset by DCAN internal events. 1h (R/W) = A message has been successfully received since the last time when this bit was reset by a read access of the CPU (independent of the result of acceptance filtering).
3	TxOk		1h	Transmitted a message successfully. This bit will be reset if error and status register is read. 0h (R/W) = No message has been successfully transmitted since the last time when this bit was read by the CPU. This bit is never reset by DCAN internal events. 1h (R/W) = A message has been successfully transmitted (error free and acknowledged by at least one other node) since the last time when this bit was reset by a read access of the CPU.
2-0	LEC		7h	Last error code. The LEC field indicates the type of the last error on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. 0h (R/W) = No error 1h (R/W) = Stuff error. More than five equal bits in a row have been detected in a part of a received message where this is not allowed. 2h (R/W) = Form error. A fixed format part of a received frame has the wrong format. 3h (R/W) = Ack error. The message this CAN core transmitted was not acknowledged by another node. 4h (R/W) = Bit1 error. During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant. 5h (R/W) = Bit0 error. During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (logical value '0'), but the monitored bus level was recessive. During Bus-Off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus-Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 6h (R/W) = CRC error. In a received message, the CRC check sum was incorrect. (CRC received for an incoming message does not match the calculated CRC for the received data). 7h (R/W) = No CAN bus event was detected since the last time the CPU read the error and status register. Any read access to the error and status register re-initializes the LEC to value '7.'

23.4.1.3 ERRC Register (offset = 8h) [reset = 0h]

ERRC is shown in Figure 23-21 and described in Table 23-16.

Figure 23-21. ERRC Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC_6:0_							TEC_7:0_							
R-0h	R-0h							R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-16. ERRC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	
15	RP	R	0h	Receive error passive 0h (R/W) = The receive error counter is below the error passive level. 1h (R/W) = The receive error counter has reached the error passive level as defined in the CAN specification.
14-8	REC_6:0_	R	0h	Receive error counter. Actual state of the receive error counter (values from 0 to 255).
7-0	TEC_7:0_	R	0h	Transmit error counter. Actual state of the transmit error counter (values from 0 to 255).

23.4.1.4 BTR Register (offset = Ch) [reset = 2301h]

BTR is shown in Figure 23-22 and described in Table 23-17.

This register is only writable if CCE and Init bits in the CAN control register are set. The CAN bit time may be programmed in the range of 8 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to1024 CAN_CLK periods. With a CAN_CLK of 8 MHz and BRPE = 0x00, the reset value of 0x00002301 configures the DCAN for a bit rate of 500kBit/s.

Figure 23-22. BTR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							

23	22	21	20	19	18	17	16
RESERVED				BRPE			
R-0h				R-0h			

15	14	13	12	11	10	9	8
RESERVED	TSeg2			TSeg1			
R-0h	2h			3h			

7	6	5	4	3	2	1	0
SJW		BRP					
0h		1h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-17. BTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	
19-16	BRPE	R	0h	Baud rate prescaler extension. Valid programmed values are 0 to 15. By programming BRPE the baud rate prescaler can be extended to values up to 1024.
15	RESERVED	R	0h	
14-12	TSeg2		2h	Time segment after the sample point. Valid programmed values are 0 to 7. The actual TSeg2 value which is interpreted for the bit timing will be the programmed TSeg2 value + 1.
11-8	TSeg1		3h	Time segment before the sample point. Valid programmed values are 1 to 15. The actual TSeg1 value interpreted for the bit timing will be the programmed TSeg1 value + 1.
7-6	SJW		0h	Synchronization Jump Width. Valid programmed values are 0 to 3. The actual SJW value interpreted for the synchronization will be the programmed SJW value + 1.
5-0	BRP		1h	Baud rate prescaler. Value by which the CAN_CLK frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid programmed values are 0 to 63. The actual BRP value interpreted for the bit timing will be the programmed BRP value + 1.

23.4.1.5 INT Register (offset = 10h) [reset = 0h]

INT is shown in Figure 23-23 and described in Table 23-18.

Figure 23-23. INT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								Int1ID_23:16_							
R-0h								R-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Int0ID_15_0															
R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 23-18. INT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	
23-16	Int1ID_23:16_	R	0h	Interrupt 1 Identifier (indicates the message object with the highest pending interrupt). If several interrupts are pending, the CAN interrupt register will point to the pending interrupt with the highest priority. The DCAN1INT interrupt line remains active until Int1ID reaches value 0 (the cause of the interrupt is reset) or until IE1 is cleared. A message interrupt is cleared by clearing the message object's IntPnd bit. Among the message interrupts, the message object's interrupt priority decreases with increasing message number. 0x 00: No interrupt is pending. 0x 01-0x 80: Number of message object which caused the interrupt. 0xFF: Unused.
15-0	Int0ID_15_0	R	0h	Interrupt Identifier (the number here indicates the source of the interrupt). If several interrupts are pending, the CAN interrupt register will point to the pending interrupt with the highest priority. The DCAN0INT interrupt line remains active until Int0ID reaches value 0 (the cause of the interrupt is reset) or until IE0 is cleared. The Status interrupt has the highest priority. Among the message interrupts, the message object's interrupt priority decreases with increasing message number. 0x 0000: No interrupt is pending. 0x 0001-0x 0080: Number of message object which caused the interrupt. 0x 0081-0x7FFF: Unused (values 0081 to 7FFF). 0x 8000: Error and status register value is not 0x07. 0xFFFF: Unused.

23.4.1.6 TEST Register (offset = 14h) [reset = 0h]

TEST is shown in Figure 23-24 and described in Table 23-19.

For all test modes, the test bit in CAN control register needs to be set to one. If test bit is set, the RDA, EXL, Tx1, Tx0, LBack and Silent bits are writable. Bit Rx monitors the state of pin CAN_RX and therefore is only readable. All test register functions are disabled when test bit is cleared. The test register is only writable if test bit in CAN control register is set. Setting Tx[1:0] other than '00' will disturb message transfer. When the internal loop-back mode is active (bit LBack is set), bit EXL will be ignored.

Figure 23-24. TEST Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						RDA	EXL
R-0h						0h	0h
7	6	5	4	3	2	1	0
Rx	Tx_1:0_		LBack	Silent	RESERVED		
R-0h	0h		0h	0h	R-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-19. TEST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	
9	RDA		0h	RAM direct access enable 0h (R/W) = Normal operation 1h (R/W) = Direct access to the RAM is enabled while in test mode
8	EXL		0h	External loopback mode 0h (R/W) = Disabled 1h (R/W) = Enabled
7	Rx	R	0h	Receive pin. Monitors the actual value of the CAN_RX pin 0h (R/W) = The CAN bus is dominant 1h (R/W) = The CAN bus is recessive
6-5	Tx_1:0_		0h	Control of CAN_TX pin. 0h (R/W) = Normal operation, CAN_TX is controlled by the CAN core. 1h (R/W) = Sample point can be monitored at CAN_TX pin. 10h (R/W) = CAN_TX pin drives a dominant value. 11h (R/W) = CAN_TX pin drives a recessive value.
4	LBack		0h	Loopback mode 0h (R/W) = Disabled 1h (R/W) = Enabled
3	Silent		0h	Silent mode 0h (R/W) = Disabled 1h (R/W) = Enabled
2-0	RESERVED	R	0h	

23.4.1.7 PERR Register (offset = 1Ch) [reset = 0h]

PERR is shown in Figure 23-25 and described in Table 23-20.

If a parity error is detected, the PER flag will be set in the error and status register. This bit is not reset by the parity check mechanism; it must be reset by reading the error and status register. In addition to the PER flag, the parity error code register will indicate the memory area where the parity error has been detected (message number and word number). If more than one word with a parity error was detected, the highest word number with a parity error will be displayed. After a parity error has been detected, the register will hold the last error code until power is removed.

Figure 23-25. PERR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED					Word_Number		
R-0h					R-0h		
7	6	5	4	3	2	1	0
Message_Number							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-20. PERR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	
10-8	Word_Number	R	0h	Word number where parity error has been detected. 0x 01-0x 05: RDA word number (1 to 5) of the message object (according to the message RAM representation in RDA mode).
7-0	Message_Number	R	0h	Message number. 0x 01-0x 80: Message object number where parity error has been detected.

23.4.1.8 ABOTR Register (offset = 80h) [reset = 0h]

ABOTR is shown in Figure 23-26 and described in Table 23-21.

On write access to the CAN control register while Auto-Bus-On timer is running, the Auto-Bus-On procedure will be aborted. During Debug/Suspend mode, running Auto-Bus-On timer will be paused.

Figure 23-26. ABOTR Register

31–0
ABO_Time
R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 23-21. ABOTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ABO_Time	R/W	0h	Number of OCP clock cycles before a Bus-Off recovery sequence is started by clearing the Init bit. This function has to be enabled by setting bit ABO in CAN control register. The Auto-Bus-On timer is realized by a 32 bit counter that starts to count down to zero when the module goes Bus-Off. The counter will be reloaded with the preload value of the ABO time register after this phase.

23.4.1.9 TXRQ_X Register (offset = 84h) [reset = 0h]

TXRQ_X is shown in Figure 23-27 and described in Table 23-22.

Example 1. Bit 0 of the transmission request X register represents byte 0 of the transmission request 1 register. If one or more bits in this byte are set, bit 0 of the transmission request X register will be set.

Figure 23-27. TXRQ_X Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED							
R-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TxRqstReg8		TxRqstReg7		TxRqstReg6		TxRqstReg5	
R-0h		R-0h		R-0h		R-0h	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TxRqstReg4		TxRqstReg3		TxRqstReg2		TxRqstReg1	
R-0h		R-0h		R-0h		R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-22. TXRQ_X Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	
15-14	TxRqstReg8	R	0h	TxRqstReg8
13-12	TxRqstReg7	R	0h	TxRqstReg7
11-10	TxRqstReg6	R	0h	TxRqstReg6
9-8	TxRqstReg5	R	0h	TxRqstReg5
7-6	TxRqstReg4	R	0h	TxRqstReg4
5-4	TxRqstReg3	R	0h	TxRqstReg3
3-2	TxRqstReg2	R	0h	TxRqstReg2
1-0	TxRqstReg1	R	0h	TxRqstReg1

23.4.1.10 TXRQ12 Register (offset = 88h) [reset = 0h]

TXRQ12 is shown in Figure 23-28 and described in Table 23-23.

The TXRQ12 to TXRQ78 registers hold the TxRqst bits of the implemented message objects. By reading out these bits, the CPU can check for pending transmission requests. The TxRqst bit in a specific message object can be set/reset by the CPU via the IF1/IF2 message interface registers, or by the message handler after reception of a remote frame or after a successful transmission.

Figure 23-28. TXRQ12 Register

31-16	15-0
TxRqs_32:17_	TxRqs_16:1_
R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-23. TXRQ12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	TxRqs_32:17_	R	0h	Transmission request bits (for all message objects) 0h (R/W) = No transmission has been requested for this message object. 1h (R/W) = The transmission of this message object is requested and is not yet done.
15-0	TxRqs_16:1_	R	0h	Transmission request bits (for all message objects) 0h (R/W) = No transmission has been requested for this message object. 1h (R/W) = The transmission of this message object is requested and is not yet done.

23.4.1.11 TXRQ34 Register (offset = 8Ch) [reset = 0h]

TXRQ34 is shown in Figure 23-29 and described in Table 23-24.

The TXRQ12 to TXRQ78 registers hold the TxRqst bits of the implemented message objects. By reading out these bits, the CPU can check for pending transmission requests. The TxRqst bit in a specific message object can be set/reset by the CPU via the IF1/IF2 message interface registers, or by the message handler after reception of a remote frame or after a successful transmission.

Figure 23-29. TXRQ34 Register

31-16	15-0
TxRqs_64:49_	TxRqs_48:33_
R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-24. TXRQ34 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	TxRqs_64:49_	R	0h	Transmission request bits (for all message objects) 0h (R/W) = No transmission has been requested for this message object. 1h (R/W) = The transmission of this message object is requested and is not yet done.
15-0	TxRqs_48:33_	R	0h	Transmission request bits (for all message objects) 0h (R/W) = No transmission has been requested for this message object. 1h (R/W) = The transmission of this message object is requested and is not yet done.

23.4.1.12 TXRQ56 Register (offset = 90h) [reset = 0h]

TXRQ56 is shown in Figure 23-30 and described in Table 23-25.

The TXRQ12 to TXRQ78 registers hold the TxRqst bits of the implemented message objects. By reading out these bits, the CPU can check for pending transmission requests. The TxRqst bit in a specific message object can be set/reset by the CPU via the IF1/IF2 message interface registers, or by the message handler after reception of a remote frame or after a successful transmission.

Figure 23-30. TXRQ56 Register

31-16	15-0
TxRqs_96:81_	TxRqs_80:65_
R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-25. TXRQ56 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	TxRqs_96:81_	R	0h	Transmission request bits (for all message objects) 0h (R/W) = No transmission has been requested for this message object. 1h (R/W) = The transmission of this message object is requested and is not yet done.
15-0	TxRqs_80:65_	R	0h	Transmission request bits (for all message objects) 0h (R/W) = No transmission has been requested for this message object. 1h (R/W) = The transmission of this message object is requested and is not yet done.

23.4.1.13 TXRQ78 Register (offset = 94h) [reset = 0h]

TXRQ78 is shown in Figure 23-31 and described in Table 23-26.

The TXRQ12 to TXRQ78 registers hold the TxRqst bits of the implemented message objects. By reading out these bits, the CPU can check for pending transmission requests. The TxRqst bit in a specific message object can be set/reset by the CPU via the IF1/IF2 message interface registers, or by the message handler after reception of a remote frame or after a successful transmission.

Figure 23-31. TXRQ78 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
TxRqs_128:113_																TxRqs_112:97_															
R-0h																R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-26. TXRQ78 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	TxRqs_128:113_	R	0h	Transmission request bits (for all message objects) 0h (R/W) = No transmission has been requested for this message object. 1h (R/W) = The transmission of this message object is requested and is not yet done.
15-0	TxRqs_112:97_	R	0h	Transmission request bits (for all message objects) 0h (R/W) = No transmission has been requested for this message object. 1h (R/W) = The transmission of this message object is requested and is not yet done.

23.4.1.14 NWDAT_X Register (offset = 98h) [reset = 0h]

NWDAT_X is shown in Figure 23-32 and described in Table 23-27.

With the new data X register, the CPU can detect if one or more bits in the different new data registers are set. Each register bit represents a group of eight message objects. If at least on of the NewDat bits of these message objects are set, the corresponding bit in the new data X register will be set. Example 1. Bit 0 of the new data X register represents byte 0 of the new data 1 register. If one or more bits in this byte are set, bit 0 of the new data X register will be set.

Figure 23-32. NWDAT_X Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED							
R-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NewDatReg8		NewDatReg7		NewDatReg6		NewDatReg5	
R-0h		R-0h		R-0h		R-0h	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NewDatReg4		NewDatReg3		NewDatReg2		NewDatReg1	
R-0h		R-0h		R-0h		R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-27. NWDAT_X Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	
15-14	NewDatReg8	R	0h	NewDatReg8
13-12	NewDatReg7	R	0h	NewDatReg7
11-10	NewDatReg6	R	0h	NewDatReg6
9-8	NewDatReg5	R	0h	NewDatReg5
7-6	NewDatReg4	R	0h	NewDatReg4
5-4	NewDatReg3	R	0h	NewDatReg3
3-2	NewDatReg2	R	0h	NewDatReg2
1-0	NewDatReg1	R	0h	NewDatReg1

23.4.1.15 NWDAT12 Register (offset = 9Ch) [reset = 0h]

NWDAT12 is shown in Figure 23-33 and described in Table 23-28.

These registers hold the NewDat bits of the implemented message objects. By reading out these bits, the CPU can check for new data in the message objects. The NewDat bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after reception of a data frame or after a successful transmission.

Figure 23-33. NWDAT12 Register

31-16	15-0
NewDat_32:17_	NewDat_16:1_
R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-28. NWDAT12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	NewDat_32:17_	R	0h	New Data Bits (for all message objects) 0h (R/W) = No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1h (R/W) = The message handler or the CPU has written new data into the data portion of this message object.
15-0	NewDat_16:1_	R	0h	New Data Bits (for all message objects) 0h (R/W) = No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1h (R/W) = The message handler or the CPU has written new data into the data portion of this message object.

23.4.1.16 NWDAT34 Register (offset = A0h) [reset = 0h]

NWDAT34 is shown in Figure 23-34 and described in Table 23-29.

These registers hold the NewDat bits of the implemented message objects. By reading out these bits, the CPU can check for new data in the message objects. The NewDat bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after reception of a data frame or after a successful transmission.

Figure 23-34. NWDAT34 Register

31-16	15-0
NewDat_64:49_	NewDat_48:33_
R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-29. NWDAT34 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	NewDat_64:49_	R	0h	New Data Bits (for all message objects) 0h (R/W) = No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1h (R/W) = The message handler or the CPU has written new data into the data portion of this message object.
15-0	NewDat_48:33_	R	0h	New Data Bits (for all message objects) 0h (R/W) = No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1h (R/W) = The message handler or the CPU has written new data into the data portion of this message object.

23.4.1.17 NWDAT56 Register (offset = A4h) [reset = 0h]

NWDAT56 is shown in Figure 23-35 and described in Table 23-30.

These registers hold the NewDat bits of the implemented message objects. By reading out these bits, the CPU can check for new data in the message objects. The NewDat bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after reception of a data frame or after a successful transmission.

Figure 23-35. NWDAT56 Register

31-16	15-0
NewDat_96:81_	NewDat_80:65_
R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 23-30. NWDAT56 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	NewDat_96:81_	R	0h	New Data Bits (for all message objects) 0h (R/W) = No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1h (R/W) = The message handler or the CPU has written new data into the data portion of this message object.
15-0	NewDat_80:65_	R	0h	New Data Bits (for all message objects) 0h (R/W) = No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1h (R/W) = The message handler or the CPU has written new data into the data portion of this message object.

23.4.1.18 NWDAT78 Register (offset = A8h) [reset = 0h]

NWDAT78 is shown in Figure 23-36 and described in Table 23-31.

These registers hold the NewDat bits of the implemented message objects. By reading out these bits, the CPU can check for new data in the message objects. The NewDat bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after reception of a data frame or after a successful transmission.

Figure 23-36. NWDAT78 Register

31-16	15-0
NewDat_128:113_	NewDat_112:97_
R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-31. NWDAT78 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	NewDat_128:113_	R	0h	New Data Bits (for all message objects) 0h (R/W) = No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1h (R/W) = The message handler or the CPU has written new data into the data portion of this message object.
15-0	NewDat_112:97_	R	0h	New Data Bits (for all message objects) 0h (R/W) = No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1h (R/W) = The message handler or the CPU has written new data into the data portion of this message object.

23.4.1.19 INTPND_X Register (offset = ACh) [reset = 0h]

INTPND_X is shown in Figure 23-37 and described in Table 23-32.

With the interrupt pending X register, the CPU can detect if one or more bits in the different interrupt pending registers are set. Each bit of this register represents a group of eight message objects. If at least one of the IntPnd bits of these message objects are set, the corresponding bit in the interrupt pending X register will be set. Example 2. Bit 0 of the interrupt pending X register represents byte 0 of the interrupt pending 1 register. If one or more bits in this byte are set, bit 0 of the interrupt pending X register will be set.

Figure 23-37. INTPND_X Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
IntPndReg8		IntPndReg7		IntPndReg6		IntPndReg5	
R-0h		R-0h		R-0h		R-0h	
7	6	5	4	3	2	1	0
IntPndReg4		IntPndReg3		IntPndReg2		IntPndReg1	
R-0h		R-0h		R-0h		R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 23-32. INTPND_X Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	
15-14	IntPndReg8	R	0h	IntPndReg8
13-12	IntPndReg7	R	0h	IntPndReg7
11-10	IntPndReg6	R	0h	IntPndReg6
9-8	IntPndReg5	R	0h	IntPndReg5
7-6	IntPndReg4	R	0h	IntPndReg4
5-4	IntPndReg3	R	0h	IntPndReg3
3-2	IntPndReg2	R	0h	IntPndReg2
1-0	IntPndReg1	R	0h	IntPndReg1

23.4.1.20 INTPND12 Register (offset = B0h) [reset = 0h]

INTPND12 is shown in Figure 23-38 and described in Table 23-33.

These registers hold the IntPnd bits of the implemented message objects. By reading out these bits, the CPU can check for pending interrupts in the message objects. The IntPnd bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after a reception or a successful transmission.

Figure 23-38. INTPND12 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
IntPnd_32:17_																IntPnd_16:1_															
R-0h																R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-33. INTPND12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	IntPnd_32:17_	R	0h	Interrupt Pending Bits (for all message objects) 0h (R/W) = This message object is not the source of an interrupt. 1h (R/W) = This message object is the source of an interrupt.
15-0	IntPnd_16:1_	R	0h	Interrupt Pending Bits (for all message objects) 0h (R/W) = This message object is not the source of an interrupt. 1h (R/W) = This message object is the source of an interrupt.

23.4.1.21 INTPND34 Register (offset = B4h) [reset = 0h]

INTPND34 is shown in Figure 23-39 and described in Table 23-34.

These registers hold the IntPnd bits of the implemented message objects. By reading out these bits, the CPU can check for pending interrupts in the message objects. The IntPnd bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after a reception or a successful transmission.

Figure 23-39. INTPND34 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
IntPnd_64:49_																IntPnd_48:33_															
R-0h																R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-34. INTPND34 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	IntPnd_64:49_	R	0h	Interrupt Pending Bits (for all message objects) 0h (R/W) = This message object is not the source of an interrupt. 1h (R/W) = This message object is the source of an interrupt.
15-0	IntPnd_48:33_	R	0h	Interrupt Pending Bits (for all message objects) 0h (R/W) = This message object is not the source of an interrupt. 1h (R/W) = This message object is the source of an interrupt.

23.4.1.22 INTPND56 Register (offset = B8h) [reset = 0h]

INTPND56 is shown in Figure 23-40 and described in Table 23-35.

These registers hold the IntPnd bits of the implemented message objects. By reading out these bits, the CPU can check for pending interrupts in the message objects. The IntPnd bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after a reception or a successful transmission.

Figure 23-40. INTPND56 Register

31-16	15-0
IntPnd_96:81_	IntPnd_80:65_
R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 23-35. INTPND56 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	IntPnd_96:81_	R	0h	Interrupt Pending Bits (for all message objects) 0h (R/W) = This message object is not the source of an interrupt. 1h (R/W) = This message object is the source of an interrupt.
15-0	IntPnd_80:65_	R	0h	Interrupt Pending Bits (for all message objects) 0h (R/W) = This message object is not the source of an interrupt. 1h (R/W) = This message object is the source of an interrupt.

23.4.1.23 INTPND78 Register (offset = BCh) [reset = 0h]

INTPND78 is shown in Figure 23-41 and described in Table 23-36.

These registers hold the IntPnd bits of the implemented message objects. By reading out these bits, the CPU can check for pending interrupts in the message objects. The IntPnd bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after a reception or a successful transmission.

Figure 23-41. INTPND78 Register

31-16	15-0
IntPnd_128:113_	IntPnd_112:97_
R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-36. INTPND78 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	IntPnd_128:113_	R	0h	Interrupt Pending Bits (for all message objects) 0h (R/W) = This message object is not the source of an interrupt. 1h (R/W) = This message object is the source of an interrupt.
15-0	IntPnd_112:97_	R	0h	Interrupt Pending Bits (for all message objects) 0h (R/W) = This message object is not the source of an interrupt. 1h (R/W) = This message object is the source of an interrupt.

23.4.1.24 MSGVAL_X Register (offset = C0h) [reset = 0h]

MSGVAL_X is shown in Figure 23-42 and described in Table 23-37.

With the message valid X register, the CPU can detect if one or more bits in the different message valid registers are set. Each bit of this register represents a group of eight message objects. If at least one of the MsgVal bits of these message objects are set, the corresponding bit in the message valid X register will be set. Example 3. Bit 0 of the message valid X register represents byte 0 of the message valid 1 register. If one or more bits in this byte are set, bit 0 of the message valid X register will be set.

Figure 23-42. MSGVAL_X Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							

23	22	21	20	19	18	17	16
RESERVED							
R-0h							

15	14	13	12	11	10	9	8
MsgValReg8		MsgValReg7		MsgValReg6		MsgValReg5	
R-0h		R-0h		R-0h		R-0h	

7	6	5	4	3	2	1	0
MsgValReg4		MsgValReg3		MsgValReg2		MsgValReg1	
R-0h		R-0h		R-0h		R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-37. MSGVAL_X Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	
15-14	MsgValReg8	R	0h	MsgValReg8
13-12	MsgValReg7	R	0h	MsgValReg7
11-10	MsgValReg6	R	0h	MsgValReg6
9-8	MsgValReg5	R	0h	MsgValReg5
7-6	MsgValReg4	R	0h	MsgValReg4
5-4	MsgValReg3	R	0h	MsgValReg3
3-2	MsgValReg2	R	0h	MsgValReg2
1-0	MsgValReg1	R	0h	MsgValReg1

23.4.1.25 MSGVAL12 Register (offset = C4h) [reset = 0h]

MSGVAL12 is shown in Figure 23-43 and described in Table 23-38.

These registers hold the MsgVal bits of the implemented message objects. By reading out these bits, the CPU can check which message objects are valid. The MsgVal bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after a reception or a successful transmission.

Figure 23-43. MSGVAL12 Register

31-16	15-0
MsgVal_32:17_	MsgVal_16:1_
R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-38. MSGVAL12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	MsgVal_32:17_	R	0h	Message valid bits (for all message objects) 0h (R/W) = This message object is ignored by the message handler. 1h (R/W) = This message object is configured and will be considered by the message handler.
15-0	MsgVal_16:1_	R	0h	Message valid bits (for all message objects) 0h (R/W) = This message object is ignored by the message handler. 1h (R/W) = This message object is configured and will be considered by the message handler.

23.4.1.26 MSGVAL34 Register (offset = C8h) [reset = 0h]

MSGVAL34 is shown in Figure 23-44 and described in Table 23-39.

These registers hold the MsgVal bits of the implemented message objects. By reading out these bits, the CPU can check which message objects are valid. The MsgVal bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after a reception or a successful transmission.

Figure 23-44. MSGVAL34 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
MsgVal_64:49_																MsgVal_48:33_															
R-0h																R-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-39. MSGVAL34 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	MsgVal_64:49_	R	0h	Message valid bits (for all message objects) 0h (R/W) = This message object is ignored by the message handler. 1h (R/W) = This message object is configured and will be considered by the message handler.
15-0	MsgVal_48:33_	R	0h	Message valid bits (for all message objects) 0h (R/W) = This message object is ignored by the message handler. 1h (R/W) = This message object is configured and will be considered by the message handler.

23.4.1.27 MSGVAL56 Register (offset = CCh) [reset = 0h]

MSGVAL56 is shown in Figure 23-45 and described in Table 23-40.

These registers hold the MsgVal bits of the implemented message objects. By reading out these bits, the CPU can check which message objects are valid. The MsgVal bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after a reception or a successful transmission.

Figure 23-45. MSGVAL56 Register

31-16	15-0
MsgVal_96:81_	MsgVal_80:65_
R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-40. MSGVAL56 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	MsgVal_96:81_	R	0h	Message valid bits (for all message objects) 0h (R/W) = This message object is ignored by the message handler. 1h (R/W) = This message object is configured and will be considered by the message handler.
15-0	MsgVal_80:65_	R	0h	Message valid bits (for all message objects) 0h (R/W) = This message object is ignored by the message handler. 1h (R/W) = This message object is configured and will be considered by the message handler.

23.4.1.28 MSGVAL78 Register (offset = D0h) [reset = 0h]

MSGVAL78 is shown in Figure 23-46 and described in Table 23-41.

These registers hold the MsgVal bits of the implemented message objects. By reading out these bits, the CPU can check which message objects are valid. The MsgVal bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after a reception or a successful transmission.

Figure 23-46. MSGVAL78 Register

31-16	15-0
MsgVal_128:113_	MsgVal_112:97_
R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-41. MSGVAL78 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	MsgVal_128:113_	R	0h	Message valid bits (for all message objects) 0h (R/W) = This message object is ignored by the message handler. 1h (R/W) = This message object is configured and will be considered by the message handler.
15-0	MsgVal_112:97_	R	0h	Message valid bits (for all message objects) 0h (R/W) = This message object is ignored by the message handler. 1h (R/W) = This message object is configured and will be considered by the message handler.

23.4.1.29 INTMUX12 Register (offset = D8h) [reset = 0h]

INTMUX12 is shown in Figure 23-47 and described in Table 23-42.

The IntMux flag determine for each message object, which of the two interrupt lines (DCAN0INT or DCAN1INT) will be asserted when the IntPnd of this message object is set. Both interrupt lines can be globally enabled or disabled by setting or clearing IE0 and IE1 bits in CAN control register. The IntPnd bit of a specific message object can be set or reset by the CPU via the IF1/IF2 interface register sets, or by message handler after reception or successful transmission of a frame. This will also affect the Int0ID resp Int1ID flags in the interrupt register.

Figure 23-47. INTMUX12 Register

31-0
IntMux
R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-42. INTMUX12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	IntMux	R	0h	IntMux[31:0] multiplexes IntPnd value to either DCAN0INT or DCAN1INT interrupt lines. The mapping from the bits to the message objects is as follows. Bit 0 -> Last implemented message object. Bit 1 -> Message object number 1. Bit 2 -> Message object number 2. 0h (R/W) = DCAN0INT line is active if corresponding IntPnd flag is one. 1h (R/W) = DCAN1INT line is active if corresponding IntPnd flag is one.

23.4.1.30 INTMUX34 Register (offset = DCh) [reset = 0h]

INTMUX34 is shown in Figure 23-48 and described in Table 23-43.

The IntMux flag determine for each message object, which of the two interrupt lines (DCAN0INT or DCAN1INT) will be asserted when the IntPnd of this message object is set. Both interrupt lines can be globally enabled or disabled by setting or clearing IE0 and IE1 bits in CAN control register. The IntPnd bit of a specific message object can be set or reset by the CPU via the IF1/IF2 interface register sets, or by message handler after reception or successful transmission of a frame. This will also affect the Int0ID resp Int1ID flags in the interrupt register.

Figure 23-48. INTMUX34 Register

31-0
IntMux
R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-43. INTMUX34 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	IntMux	R	0h	IntMux[63:32] multiplexes IntPnd value to either DCAN0INT or DCAN1INT interrupt lines. The mapping from the bits to the message objects is as follows. Bit 0 -> Last implemented message object. Bit 1 -> Message object number 1 Bit 2 -> Message object number 2. 0h (R/W) = DCAN0INT line is active if corresponding IntPnd flag is one. 1h (R/W) = DCAN1INT line is active if corresponding IntPnd flag is one.

23.4.1.31 INTMUX56 Register (offset = E0h) [reset = 0h]

INTMUX56 is shown in Figure 23-49 and described in Table 23-44.

The IntMux flag determine for each message object, which of the two interrupt lines (DCAN0INT or DCAN1INT) will be asserted when the IntPnd of this message object is set. Both interrupt lines can be globally enabled or disabled by setting or clearing IE0 and IE1 bits in CAN control register. The IntPnd bit of a specific message object can be set or reset by the CPU via the IF1/IF2 interface register sets, or by message handler after reception or successful transmission of a frame. This will also affect the Int0ID resp Int1ID flags in the interrupt register.

Figure 23-49. INTMUX56 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
IntMux																															
R-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-44. INTMUX56 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	IntMux	R	0h	IntMux[95:64] multiplexes IntPnd value to either DCAN0INT or DCAN1INT interrupt lines. The mapping from the bits to the message objects is as follows. Bit 0 -> Last implemented message object. Bit 1 -> Message object number 1. Bit 2 -> Message object number 2. 0h (R/W) = DCAN0INT line is active if corresponding IntPnd flag is one. 1h (R/W) = DCAN1INT line is active if corresponding IntPnd flag is one.

23.4.1.32 INTMUX78 Register (offset = E4h) [reset = 0h]

INTMUX78 is shown in Figure 23-50 and described in Table 23-45.

The IntMux flag determine for each message object, which of the two interrupt lines (DCAN0INT or DCAN1INT) will be asserted when the IntPnd of this message object is set. Both interrupt lines can be globally enabled or disabled by setting or clearing IE0 and IE1 bits in CAN control register. The IntPnd bit of a specific message object can be set or reset by the CPU via the IF1/IF2 interface register sets, or by message handler after reception or successful transmission of a frame. This will also affect the Int0ID resp Int1ID flags in the interrupt register.

Figure 23-50. INTMUX78 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
IntMux																															
R-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-45. INTMUX78 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	IntMux	R	0h	IntMux[127:96] multiplexes IntPnd value to either DCAN0INT or DCAN1INT interrupt lines. The mapping from the bits to the message objects is as follows. Bit 0 -> Last implemented message object. Bit 1 -> Message object number 1. Bit 2 -> Message object number 2. 0h (R/W) = DCAN0INT line is active if corresponding IntPnd flag is one. 1h (R/W) = DCAN1INT line is active if corresponding IntPnd flag is one.

23.4.1.33 IF1CMD Register (offset = 100h) [reset = 0h]

IF1CMD is shown in Figure 23-51 and described in Table 23-46.

The IF1 Command Register (IF1CMD) configures and initiates the transfer between the IF1 register sets and the message RAM. It is configurable which portions of the message object should be transferred. A transfer is started when the CPU writes the message number to bits [7:0] of the IF1 command register. With this write operation, the Busy bit is automatically set to '1' to indicate that a transfer is in progress. After 4 to 14 OCP clock cycles, the transfer between the interface register and the message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer concurs with a CAN message transmission, acceptance filtering, or message storage. If the CPU writes to both IF1 command registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed. While Busy bit is one, IF1 register sets are write protected. For debug support, the auto clear functionality of the IF1 command registers (clear of DMAactive flag by r/w) is disabled during Debug/Suspend mode. If an invalid Message Number is written to bits [7:0] of the IF1 command register, the message handler may access an implemented (valid) message object instead.

Figure 23-51. IF1CMD Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
WR_RD	Mask	Arb	Control	ClrIntPnd	TxRqst_NewDat	Data_A	Data_B
0h	0h	0h	0h	0h	0h	0h	0h
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Busy	DMAactive	RESERVED					
0h	0h	R-0h					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Message_Number							
0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-46. IF1CMD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	
23	WR_RD		0h	Write/Read 0h (R/W) = Direction = Read: Transfer direction is from the message object addressed by Message Number (Bits [7:0]) to the IF1 register set. 1h (R/W) = Direction = Write: Transfer direction is from the IF1 register set to the message object addressed by Message Number (Bits [7:0]).
22	Mask		0h	Access mask bits 0h (R/W) = Mask bits will not be changed 1h (R/W) = Direction = Read: The mask bits (identifier mask + MDir + MXtd) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the IF1 register set. Direction = Write: The mask bits (identifier mask + MDir + MXtd) will be transferred from the IF1 register set to the message object addressed by Message Number (Bits [7:0]).

Table 23-46. IF1CMD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21	Arb		0h	Access arbitration bits 0h (R/W) = Arbitration bits will not be changed 1h (R/W) = Direction = Read: The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1 register set. Direction = Write: The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the IF1 register set to the message object addressed by Message Number (Bits [7:0]).
20	Control		0h	Access control bits. If the TxRqst/NewDat bit in this register(Bit [18]) is set, the TxRqst/ NewDat bit in the IF1 message control register will be ignored. 0h (R/W) = Control bits will not be changed 1h (R/W) = Direction = Read: The message control bits will be transferred from the message object addressed by message number (Bits [7:0]) to the IF1 register set. Direction = Write: The message control bits will be transferred from the IF1 register set to the message object addressed by message number (Bits [7:0]).
19	ClrIntPnd		0h	Clear interrupt pending bit 0h (R/W) = IntPnd bit will not be changed 1h (R/W) = Direction = Read: Clears IntPnd bit in the message object. Direction = Write: This bit is ignored. Copying of IntPnd flag from IF1 Registers to message RAM can only be controlled by the control flag (Bit [20]).
18	TxRqst_NewDat		0h	Access transmission request bit. Note: If a CAN transmission is requested by setting TxRqst/NewDat in this register, the TxRqst/NewDat bits in the message object will be set to one independent of the values in IF1 message control Register. Note: A read access to a message object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IF1 message control register always reflect the status before resetting them. 0h (R/W) = Direction = Read: NewDat bit will not be changed. Direction = Write: TxRqst/NewDat bit will be handled according to the control bit. 1h (R/W) = Direction = Read: Clears NewDat bit in the message object. Direction = Write: Sets TxRqst/NewDat in message object.
17	Data_A		0h	Access Data Bytes 0 to 3. 0h (R/W) = Data Bytes 0-3 will not be changed. 1h (R/W) = Direction = Read: The data bytes 0-3 will be transferred from the message object addressed by the Message Number (Bits [7:0]) to the corresponding IF1 register set. Direction = Write: The data bytes 0-3 will be transferred from the IF1 register set to the message object addressed by the Message Number (Bits [7:0]). Note: The duration of the message transfer is independent of the number of bytes to be transferred.
16	Data_B		0h	Access Data Bytes 4 to 7. 0h (R/W) = Data Bytes 4-7 will not be changed. 1h (R/W) = Direction = Read: The data bytes 4-7 will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1 register set. Direction = Write: The data bytes 4-7 will be transferred from the IF1 register set to the message object addressed by message number (Bits [7:0]). Note: The duration of the message transfer is independent of the number of bytes to be transferred.

Table 23-46. IF1CMD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15	Busy		0h	Busy flag. This bit is set to one after the message number has been written to bits 7 to 0. IF1 register set will be write protected. The bit is cleared after read/write action has been finished. 0h (R/W) = No transfer between IF1 register set and message RAM is in progress. 1h (R/W) = Transfer between IF1 register set and message RAM is in progress.
14	DMAactive		0h	Activation of DMA feature for subsequent internal IF1 update. Note: Due to the auto reset feature of the DMAactive bit, this bit has to be set for each subsequent DMA cycle separately. 0h (R/W) = DMA request line is independent of IF1 activities. 1h (R/W) = DMA is requested after completed transfer between IF1 register set and message RAM. The DMA request remains active until the first read or write to one of the IF1 registers; an exception is a write to Message Number (Bits [7:0]) when DMAactive is one.
13-8	RESERVED	R	0h	
7-0	Message_Number		0h	Number of message object in message RAM which is used for data transfer. 0h (R/W) = Invalid message number. 1h (R/W) = Valid message numbers (value 01 to 80). 80h (R/W) = Valid message number. 81h (R/W) = Invalid message numbers (value 81 to FF). FFh (R/W) = Invalid message numbers.

23.4.1.34 IF1MSK Register (offset = 104h) [reset = E0000000h]

IF1MSK is shown in Figure 23-52 and described in Table 23-47.

The bits of the IF1 mask registers mirror the mask bits of a message object. While Busy bit of IF1 command register is one, IF1 register set is write protected.

Figure 23-52. IF1MSK Register

31	30	29	28	27	26	25	24
MXtd	MDir	RESERVED	Msk_28:0_				
1h	1h	R-1h	0h				
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Msk_28:0_							
0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Msk_28:0_							
0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Msk_28:0_							
0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-47. IF1MSK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MXtd		1h	Mask Extended Identifier. When 11 bit (standard) identifiers are used for a message object, the identifiers of received data frames are written into bits ID28 to ID18. For acceptance filtering, only these bits together with mask bits Msk28 to Msk18 are considered. 0h (R/W) = The extended identifier bit (IDE) has no effect on the acceptance filtering. 1h (R/W) = The extended identifier bit (IDE) is used for acceptance filtering.
30	MDir		1h	Mask Message Direction 0h (R/W) = The message direction bit (Dir) has no effect on the acceptance filtering. 1h (R/W) = The message direction bit (Dir) is used for acceptance filtering.
29	RESERVED	R	1h	
28-0	Msk_28:0_		0h	Identifier Mask 0h (R/W) = The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care). 1h (R/W) = The corresponding bit in the identifier of the message object is used for acceptance filtering.

23.4.1.35 IF1ARB Register (offset = 108h) [reset = 0h]

IF1ARB is shown in Figure 23-53 and described in Table 23-48.

The bits of the IF1 arbitration registers mirror the arbitration bits of a message object. While Busy bit of IF1 command register is one, IF1 register set is write protected.

Figure 23-53. IF1ARB Register

31	30	29	28	27	26	25	24
MsgVal	Xtd	Dir	ID28_to_ID0				
0h	0h	0h	0h				
23	22	21	20	19	18	17	16
ID28_to_ID0							
0h							
15	14	13	12	11	10	9	8
ID28_to_ID0							
0h							
7	6	5	4	3	2	1	0
ID28_to_ID0							
0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-48. IF1ARB Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MsgVal		0h	Message valid. The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN control register. This bit must also be reset before the identifier ID28 to ID0, the control bits Xtd, Dir or DLC3 to DLC0 are modified, or if the messages object is no longer required. 0h (R/W) = The message object is ignored by the message handler. 1h (R/W) = The message object is to be used by the message handler.
30	Xtd		0h	Extended identifier 0h (R/W) = The 11-bit (standard) Identifier is used for this message object. 1h (R/W) = The 29-bit (extended) Identifier is used for this message object.
29	Dir		0h	Message direction 0h (R/W) = Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, this message is stored in this message object. 1h (R/W) = Direction = transmit: On TxRqst, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).
28-0	ID28_to_ID0		0h	Message identifier. ID28 to ID0 is equal to 29 bit identifier (extended frame) ID28 to ID18 is equal to 11 bit identifier (standard frame)

23.4.1.36 IF1MCTL Register (offset = 10Ch) [reset = 0h]

IF1MCTL is shown in Figure 23-54 and described in Table 23-49.

The bits of the IF1 message control registers mirror the message control bits of a message object. While Busy bit of IF1 command register is one, IF1 register set is write protected.

Figure 23-54. IF1MCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
0h	0h	0h	0h	0h	0h	0h	0h
7	6	5	4	3	2	1	0
EoB	RESERVED			DLC			
0h	R-0h			0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-49. IF1MCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	
15	NewDat		0h	New data 0h (R/W) = No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1h (R/W) = The message handler or the CPU has written new data into the data portion of this message object.
14	MsgLst		0h	Message lost (only valid for message objects with direction = receive) 0h (R/W) = No message lost since the last time when this bit was reset by the CPU. 1h (R/W) = The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.
13	IntPnd		0h	Interrupt pending 0h (R/W) = This message object is not the source of an interrupt. 1h (R/W) = This message object is the source of an interrupt. The Interrupt Identifier in the interrupt register will point to this message object if there is no other interrupt source with higher priority.
12	UMask		0h	Use acceptance mask. If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one. 0h (R/W) = Mask ignored 1h (R/W) = Use mask (Msk[28:0], MXtd, and MDir) for acceptance filtering
11	TxIE		0h	Transmit interrupt enable 0h (R/W) = IntPnd will not be triggered after the successful transmission of a frame. 1h (R/W) = IntPnd will be triggered after the successful transmission of a frame.

Table 23-49. IF1MCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	RxIE		0h	Receive interrupt enable 0h (R/W) = IntPnd will not be triggered after the successful reception of a frame. 1h (R/W) = IntPnd will be triggered after the successful reception of a frame.
9	RmtEn		0h	Remote enable 0h (R/W) = At the reception of a remote frame, TxRqst is not changed. 1h (R/W) = At the reception of a remote frame, TxRqst is set.
8	TxRqst		0h	Transmit request 0h (R/W) = This message object is not waiting for a transmission. 1h (R/W) = The transmission of this message object is requested and is not yet done.
7	EoB		0h	Data frame has 0 to 8 data bits. Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one. 0h (R/W) = Data frame has 8 data bytes. 1h (R/W) = Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.
6-4	RESERVED	R	0h	
3-0	DLC		0h	Data length code. Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message. 0x 0-0x 8: Data frame has 0-8 data bytes. 0x 9-0x 15: Data frame has 8 data bytes.

23.4.1.37 IF1DATA Register (offset = 110h) [reset = 0h]

IF1DATA is shown in Figure 23-55 and described in Table 23-50.

The data bytes of CAN messages are stored in the IF1 registers in the following order: (1) In a CAN data frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. (2) In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Figure 23-55. IF1DATA 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
Data_3								Data_2								Data_1								Data_0							
0h								0h								0h								0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 23-50. IF1DATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	Data_3		0h	Data 3.
23-16	Data_2		0h	Data 2.
15-8	Data_1		0h	Data 1.
7-0	Data_0		0h	Data 0.

23.4.1.38 IF1DATB Register (offset = 114h) [reset = 0h]

IF1DATB is shown in Figure 23-56 and described in Table 23-51.

The data bytes of CAN messages are stored in the IF1 registers in the following order: (1) In a CAN data frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. (2) In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Figure 23-56. IF1DATB Register

31-24	23-16	15-8	7-0
Data_7	Data_6	Data_5	Data_4
0h	0h	0h	0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 23-51. IF1DATB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	Data_7		0h	Data 7.
23-16	Data_6		0h	Data 6.
15-8	Data_5		0h	Data 5.
7-0	Data_4		0h	Data 4.

23.4.1.39 IF2CMD Register (offset = 120h) [reset = 0h]

IF2CMD is shown in Figure 23-57 and described in Table 23-52.

The IF2 Command Register (IF1CMD) configures and initiates the transfer between the IF2 register sets and the message RAM. It is configurable which portions of the message object should be transferred. A transfer is started when the CPU writes the message number to bits [7:0] of the IF2 command register. With this write operation, the Busy bit is automatically set to '1' to indicate that a transfer is in progress. After 4 to 14 OCP clock cycles, the transfer between the interface register and the message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer concurs with a CAN message transmission, acceptance filtering, or message storage. If the CPU writes to both IF2 command registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed. While Busy bit is one, IF2 register sets are write protected. For debug support, the auto clear functionality of the IF2 command registers (clear of DMAactive flag by r/w) is disabled during Debug/Suspend mode. If an invalid Message Number is written to bits [7:0] of the IF2 command register, the message handler may access an implemented (valid) message object instead.

Figure 23-57. IF2CMD Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
WR_RD	Mask	Arb	Control	ClrIntPnd	TxRqst_NewDat	Data_A	Data_B
0h	0h	0h	0h	0h	0h	0h	0h
15	14	13	12	11	10	9	8
Busy	DMAactive	RESERVED					
0h	0h	R-0h					
7	6	5	4	3	2	1	0
Message_Number							
0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-52. IF2CMD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	
23	WR_RD		0h	Write/Read 0h (R/W) = Direction = Read: Transfer direction is from the message object addressed by Message Number (Bits [7:0]) to the IF2 register set. 1h (R/W) = Direction = Write: Transfer direction is from the IF2 register set to the message object addressed by Message Number (Bits [7:0]).
22	Mask		0h	Access mask bits 0h (R/W) = Mask bits will not be changed 1h (R/W) = Direction = Read: The mask bits (identifier mask + MDir + MXtd) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the IF2 register set. Direction = Write: The mask bits (identifier mask + MDir + MXtd) will be transferred from the IF2 register set to the message object addressed by Message Number (Bits [7:0]).

Table 23-52. IF2CMD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21	Arb		0h	Access arbitration bits 0h (R/W) = Arbitration bits will not be changed 1h (R/W) = Direction = Read: The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF2 register set. Direction = Write: The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the IF2 register set to the message object addressed by Message Number (Bits [7:0]).
20	Control		0h	Access control bits. If the TxRqst/NewDat bit in this register(Bit [18]) is set, the TxRqst/NewDat bit in the IF2 message control register will be ignored. 0h (R/W) = Control bits will not be changed 1h (R/W) = Direction = Read: The message control bits will be transferred from the message object addressed by message number (Bits [7:0]) to the IF2 register set. Direction = Write: The message control bits will be transferred from the IF2 register set to the message object addressed by message number (Bits [7:0]).
19	ClrIntPnd		0h	Clear interrupt pending bit 0h (R/W) = IntPnd bit will not be changed 1h (R/W) = Direction = Read: Clears IntPnd bit in the message object. Direction = Write: This bit is ignored. Copying of IntPnd flag from IF2 Registers to message RAM can only be controlled by the control flag (Bit [20]).
18	TxRqst_NewDat		0h	Access transmission request bit. Note: If a CAN transmission is requested by setting TxRqst/NewDat in this register, the TxRqst/NewDat bits in the message object will be set to one independent of the values in IF2 message control Register. Note: A read access to a message object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IF2 message control register always reflect the status before resetting them. 0h (R/W) = Direction = Read: NewDat bit will not be changed. Direction = Write: TxRqst/NewDat bit will be handled according to the control bit. 1h (R/W) = Direction = Read: Clears NewDat bit in the message object. Direction = Write: Sets TxRqst/NewDat in message object.
17	Data_A		0h	Access Data Bytes 0 to 3. 0h (R/W) = Data Bytes 0-3 will not be changed. 1h (R/W) = Direction = Read: The data bytes 0-3 will be transferred from the message object addressed by the Message Number (Bits [7:0]) to the corresponding IF2 register set. Direction = Write: The data bytes 0-3 will be transferred from the IF2 register set to the message object addressed by the Message Number (Bits [7:0]). Note: The duration of the message transfer is independent of the number of bytes to be transferred.
16	Data_B		0h	Access Data Bytes 4 to 7. 0h (R/W) = Data Bytes 4-7 will not be changed. 1h (R/W) = Direction = Read: The data bytes 4-7 will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF2 register set. Direction = Write: The data bytes 4-7 will be transferred from the IF2 register set to the message object addressed by message number (Bits [7:0]). Note: The duration of the message transfer is independent of the number of bytes to be transferred.

Table 23-52. IF2CMD Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15	Busy		0h	Busy flag. This bit is set to one after the message number has been written to bits 7 to 0. IF2 register set will be write protected. The bit is cleared after read/write action has been finished. 0h (R/W) = No transfer between IF2 register set and message RAM is in progress. 1h (R/W) = Transfer between IF2 register set and message RAM is in progress.
14	DMAactive		0h	Activation of DMA feature for subsequent internal IF2 update. Note: Due to the auto reset feature of the DMAactive bit, this bit has to be set for each subsequent DMA cycle separately. 0h (R/W) = DMA request line is independent of IF2 activities. 1h (R/W) = DMA is requested after completed transfer between IF2 register set and message RAM. The DMA request remains active until the first read or write to one of the IF2 registers; an exception is a write to Message Number (Bits [7:0]) when DMAactive is one.
13-8	RESERVED	R	0h	
7-0	Message_Number		0h	Number of message object in message RAM which is used for data transfer. 0h (R/W) = Invalid message number. 1h (R/W) = Valid message numbers (values 01 to 80). 80h (R/W) = Valid message number. 81h (R/W) = Invalid message numbers (values 81 to FF). FFh (R/W) = Invalid message number.

23.4.1.40 IF2MSK Register (offset = 124h) [reset = E0000000h]

IF2MSK is shown in Figure 23-58 and described in Table 23-53.

The bits of the IF2 mask registers mirror the mask bits of a message object. While Busy bit of IF2 command register is one, IF2 register set is write protected.

Figure 23-58. IF2MSK Register

31	30	29	28	27	26	25	24
MXtd	MDir	RESERVED	Msk_28:0_				
1h	1h	R-1h	0h				
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Msk_28:0_							
0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Msk_28:0_							
0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Msk_28:0_							
0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-53. IF2MSK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MXtd		1h	Mask Extended Identifier. When 11 bit (standard) identifiers are used for a message object, the identifiers of received data frames are written into bits ID28 to ID18. For acceptance filtering, only these bits together with mask bits Msk28 to Msk18 are considered. 0h (R/W) = The extended identifier bit (IDE) has no effect on the acceptance filtering. 1h (R/W) = The extended identifier bit (IDE) is used for acceptance filtering.
30	MDir		1h	Mask Message Direction 0h (R/W) = The message direction bit (Dir) has no effect on the acceptance filtering. 1h (R/W) = The message direction bit (Dir) is used for acceptance filtering.
29	RESERVED	R	1h	
28-0	Msk_28:0_		0h	Identifier Mask 0h (R/W) = The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care). 1h (R/W) = The corresponding bit in the identifier of the message object is used for acceptance filtering.

23.4.1.41 IF2ARB Register (offset = 128h) [reset = 0h]

IF2ARB is shown in Figure 23-59 and described in Table 23-54.

The bits of the IF2 arbitration registers mirror the arbitration bits of a message object. While Busy bit of IF2 command register is one, IF2 register set is write protected.

Figure 23-59. IF2ARB Register

31	30	29	28	27	26	25	24
MsgVal	Xtd	Dir	ID28_to_ID0				
0h	0h	0h	0h				
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ID28_to_ID0							
0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ID28_to_ID0							
0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID28_to_ID0							
0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-54. IF2ARB Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MsgVal		0h	Message valid. The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN control register. This bit must also be reset before the identifier ID28 to ID0, the control bits Xtd, Dir or DLC3 to DLC0 are modified, or if the messages object is no longer required. 0h (R/W) = The message object is ignored by the message handler. 1h (R/W) = The message object is to be used by the message handler.
30	Xtd		0h	Extended identifier 0h (R/W) = The 11-bit (standard) Identifier is used for this message object. 1h (R/W) = The 29-bit (extended) Identifier is used for this message object.
29	Dir		0h	Message direction 0h (R/W) = Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, this message is stored in this message object. 1h (R/W) = Direction = transmit: On TxRqst, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).
28-0	ID28_to_ID0		0h	Message identifier. ID28 to ID0 is equal to 29-bit identifier (extended frame) ID28 to ID18 is equal to 11-bit identifier (standard frame)

23.4.1.42 IF2MCTL Register (offset = 12Ch) [reset = 0h]

IF2MCTL is shown in Figure 23-60 and described in Table 23-55.

The bits of the IF2 message control registers mirror the message control bits of a message object. While Busy bit of IF2 command register is one, IF2 register set is write protected.

Figure 23-60. IF2MCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED							
R-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
0h	0h	0h	0h	0h	0h	0h	0h
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EoB	RESERVED			DLC			
0h	R-0h			0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-55. IF2MCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	
15	NewDat		0h	New data 0h (R/W) = No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1h (R/W) = The message handler or the CPU has written new data into the data portion of this message object.
14	MsgLst		0h	Message lost (only valid for message objects with direction = receive) 0h (R/W) = No message lost since the last time when this bit was reset by the CPU. 1h (R/W) = The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.
13	IntPnd		0h	Interrupt pending 0h (R/W) = This message object is not the source of an interrupt. 1h (R/W) = This message object is the source of an interrupt. The Interrupt Identifier in the interrupt register will point to this message object if there is no other interrupt source with higher priority.
12	UMask		0h	Use acceptance mask. If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one. 0h (R/W) = Mask ignored 1h (R/W) = Use mask (Msk[28:0], MXtd, and MDir) for acceptance filtering
11	TxIE		0h	Transmit interrupt enable 0h (R/W) = IntPnd will not be triggered after the successful transmission of a frame. 1h (R/W) = IntPnd will be triggered after the successful transmission of a frame.

Table 23-55. IF2MCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
10	RxIE		0h	Receive interrupt enable 0h (R/W) = IntPnd will not be triggered after the successful reception of a frame. 1h (R/W) = IntPnd will be triggered after the successful reception of a frame.
9	RmtEn		0h	Remote enable 0h (R/W) = At the reception of a remote frame, TxRqst is not changed. 1h (R/W) = At the reception of a remote frame, TxRqst is set.
8	TxRqst		0h	Transmit request 0h (R/W) = This message object is not waiting for a transmission. 1h (R/W) = The transmission of this message object is requested and is not yet done.
7	EoB		0h	Data frame has 0 to 8 data bits. Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one. 0h (R/W) = Data frame has 8 data bytes. 1h (R/W) = Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.
6-4	RESERVED	R	0h	
3-0	DLC		0h	Data length code. Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message. 0x 0-0x 8: Data frame has 0-8 data bytes. 0x 9-0x 15: Data frame has 8 data bytes.

23.4.1.43 IF2DATA Register (offset = 130h) [reset = 0h]

IF2DATA is shown in Figure 23-61 and described in Table 23-56.

The data bytes of CAN messages are stored in the IF2 registers in the following order: (1) In a CAN data frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. (2) In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Figure 23-61. IF2DATA 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
Data_3								Data_2								Data_1								Data_0							
0h								0h								0h								0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-56. IF2DATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	Data_3		0h	Data 3.
23-16	Data_2		0h	Data 2.
15-8	Data_1		0h	Data 1.
7-0	Data_0		0h	Data 0.

23.4.1.44 IF2DATB Register (offset = 134h) [reset = 0h]

IF2DATB is shown in Figure 23-62 and described in Table 23-57.

The data bytes of CAN messages are stored in the IF2 registers in the following order: (1) In a CAN data frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. (2) In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Figure 23-62. IF2DATB Register

31-24	23-16	15-8	7-0
Data_7	Data_6	Data_5	Data_4
0h	0h	0h	0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-57. IF2DATB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	Data_7		0h	Data 7.
23-16	Data_6		0h	Data 6.
15-8	Data_5		0h	Data 5.
7-0	Data_4		0h	Data 4.

23.4.1.45 IF3OBS Register (offset = 140h) [reset = 0h]

IF3OBS is shown in Figure 23-63 and described in Table 23-58.

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from message RAM by CPU. The observation flags (Bits [4:0]) in the IF3 observation register are used to determine, which data sections of the IF3 interface register set have to be read in order to complete a DMA read cycle. After all marked data sections are read, the DCAN is enabled to update the IF3 interface register set with new data. Any access order of single bytes or half-words is supported. When using byte or half-word accesses, a data section is marked as completed, if all bytes are read. Note: If IF3 Update Enable is used and no Observation flag is set, the corresponding message objects will be copied to IF3 without activating the DMA request line and without waiting for DMA read accesses. A write access to this register aborts a pending DMA cycle by resetting the DMA line and enables updating of IF3 interface register set with new data. To avoid data inconsistency, the DMA controller should be disabled before reconfiguring IF3 observation register. The status of the current read-cycle can be observed via status flags (Bits [12:8]). If an interrupt line is available for IF3, an interrupt will be generated by IF3Upd flag. See the device-specific data sheet for the availability of this interrupt source. With this interrupt, the observation status bits and the IF3Upd bit could be used by the application to realize the notification about new IF3 content in polling or interrupt mode.

Figure 23-63. IF3OBS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED							
R-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
IF3_Upd	RESERVED		IF3_SDB	IF3_SDA	IF3_SC	IF3_SA	IF3_SM
R-0h	R-0h		R-0h	R-0h	R-0h	R-0h	R-0h
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED			DataB	DataA	Ctrl	Arb	Mask
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-58. IF3OBS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	
15	IF3_Upd	R	0h	IF3 Update Data 0h (R/W) = No new data has been loaded since last IF3 read. 1h (R/W) = New data has been loaded since last IF3 read.
14-13	RESERVED	R	0h	
12	IF3_SDB	R	0h	IF3 Status of Data B read access 0h (R/W) = All Data B bytes are already read out, or are not marked to be read. 1h (R/W) = Data B section has still data to be read out.
11	IF3_SDA	R	0h	IF3 Status of Data A read access 0h (R/W) = All Data A bytes are already read out, or are not marked to be read. 1h (R/W) = Data A section has still data to be read out.
10	IF3_SC	R	0h	IF3 Status of control bits read access 0h (R/W) = All control section bytes are already read out, or are not marked to be read. 1h (R/W) = Control section has still data to be read out.

Table 23-58. IF3OBS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	IF3_SA	R	0h	IF3 Status of Arbitration data read access 0h (R/W) = All Arbitration data bytes are already read out, or are not marked to be read. 1h (R/W) = Arbitration section has still data to be read out.
8	IF3_SM	R	0h	IF3 Status of Mask data read access 0h (R/W) = All mask data bytes are already read out, or are not marked to be read. 1h (R/W) = Mask section has still data to be read out.
7-5	RESERVED	R	0h	
4	DataB	R/W	0h	Data B read observation 0h (R/W) = Data B section has not to be read. 1h (R/W) = Data B section has to be read to enable next IF3 update.
3	DataA	R/W	0h	Data A read observation 0h (R/W) = Data A section has not to be read. 1h (R/W) = Data A section has to be read to enable next IF3 update.
2	Ctrl	R/W	0h	Ctrl read observation 0h (R/W) = Ctrl section has not to be read. 1h (R/W) = Ctrl section has to be read to enable next IF3 update.
1	Arb	R/W	0h	Arbitration data read observation 0h (R/W) = Arbitration data has not to be read. 1h (R/W) = Arbitration data has to be read to enable next IF3 update.
0	Mask	R/W	0h	Mask data read observation 0h (R/W) = Mask data has not to be read. 1h (R/W) = Mask data has to be read to enable next IF3 update.

23.4.1.46 IF3MSK Register (offset = 144h) [reset = E0000000h]

IF3MSK is shown in Figure 23-64 and described in Table 23-59.

Figure 23-64. IF3MSK Register

31	30	29	28	27	26	25	24
MXtd	MDir	RESERVED	Msk_28:0_				
R-1h	R-1h	R-1h	0h				
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Msk_28:0_							
0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Msk_28:0_							
0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Msk_28:0_							
0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 23-59. IF3MSK Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MXtd	R	1h	Mask Extended Identifier. When 11 bit (standard) identifiers are used for a message object, the identifiers of received data frames are written into bits ID28 to ID18. For acceptance filtering, only these bits together with mask bits Msk28 to Msk18 are considered. 0h (R/W) = The extended identifier bit (IDE) has no effect on the acceptance filtering. 1h (R/W) = The extended identifier bit (IDE) is used for acceptance filtering.
30	MDir	R	1h	Mask Message Direction 0h (R/W) = The message direction bit (Dir) has no effect on the acceptance filtering. 1h (R/W) = The message direction bit (Dir) is used for acceptance filtering.
29	RESERVED	R	1h	
28-0	Msk_28:0_		0h	Identifier Mask 0h (R/W) = The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care). 1h (R/W) = The corresponding bit in the identifier of the message object is used for acceptance filtering.

23.4.1.47 IF3ARB Register (offset = 148h) [reset = 0h]

IF3ARB is shown in Figure 23-65 and described in Table 23-60.

Figure 23-65. IF3ARB Register

31	30	29	28	27	26	25	24
MsgVal	Xtd	Dir	ID28_to_ID0				
R-0h	R-0h	R-0h	R-0h				
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ID28_to_ID0							
R-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ID28_to_ID0							
R-0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID28_to_ID0							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-60. IF3ARB Register Field Descriptions

Bit	Field	Type	Reset	Description
31	MsgVal	R	0h	Message Valid. The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN control register. This bit must also be reset before the identifier ID28 to ID0, the control bits Xtd, Dir or DLC3 to DLC0 are modified, or if the messages object is no longer required. 0h (R/W) = The message object is ignored by the message handler. 1h (R/W) = The message object is to be used by the message handler.
30	Xtd	R	0h	Extended Identifier 0h (R/W) = The 11-bit (standard) Identifier is used for this message object. 1h (R/W) = The 29-bit (extended) Identifier is used for this message object.
29	Dir	R	0h	Message Direction 0h (R/W) = Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, this message is stored in this message object. 1h (R/W) = Direction = transmit: On TxRqst, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).
28-0	ID28_to_ID0	R	0h	Message Identifier. ID28 to ID0 is equal to 29 bit Identifier (extended frame). ID28 to ID18 is equal to 11 bit Identifier (standard frame).

23.4.1.48 IF3MCTL Register (offset = 14Ch) [reset = 0h]

IF3MCTL is shown in Figure 23-66 and described in Table 23-61.

Figure 23-66. IF3MCTL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
EoB	RESERVED			DLC			
R-0h	R-0h			R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-61. IF3MCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	
15	NewDat	R	0h	New Data 0h (R/W) = No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. 1h (R/W) = The message handler or the CPU has written new data into the data portion of this message object.
14	MsgLst	R	0h	Message Lost (only valid for message objects with direction = receive) 0h (R/W) = No message lost since the last time when this bit was reset by the CPU. 1h (R/W) = The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.
13	IntPnd	R	0h	Interrupt Pending 0h (R/W) = This message object is not the source of an interrupt. 1h (R/W) = This message object is the source of an interrupt. The Interrupt Identifier in the interrupt register will point to this message object if there is no other interrupt source with higher priority.
12	UMask	R	0h	Use Acceptance Mask 0h (R/W) = Mask ignored 1h (R/W) = Use mask (Msk[28:0], MXtd, and MDir) for acceptance filtering. If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one.
11	TxIE	R	0h	Transmit Interrupt enable 0h (R/W) = IntPnd will not be triggered after the successful transmission of a frame. 1h (R/W) = IntPnd will be triggered after the successful transmission of a frame.
10	RxIE	R	0h	Receive Interrupt enable 0h (R/W) = IntPnd will not be triggered after the successful reception of a frame. 1h (R/W) = IntPnd will be triggered after the successful reception of a frame.

Table 23-61. IF3MCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9	RmtEn	R	0h	Remote enable 0h (R/W) = At the reception of a remote frame, TxRqst is not changed. 1h (R/W) = At the reception of a remote frame, TxRqst is set.
8	TxRqst	R	0h	Transmit Request 0h (R/W) = This message object is not waiting for a transmission. 1h (R/W) = The transmission of this message object is requested and is not yet done.
7	EoB	R	0h	Data frame has 0 to 8 data bits. Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one. 0h (R/W) = Data frame has 8 data bytes. 1h (R/W) = Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.
6-4	RESERVED	R	0h	
3-0	DLC	R	0h	Data Length Code. Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message. 0x 0-0x 8: Data frame has 0-8 data bytes. 0x 9-0x 15: Data frame has 8 data bytes.

23.4.1.49 IF3DATA Register (offset = 150h) [reset = 0h]

IF3DATA is shown in Figure 23-67 and described in Table 23-62.

The data bytes of CAN messages are stored in the IF3 registers in the following order. In a CAN data frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Figure 23-67. IF3DATA 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
Data_3								Data_2								Data_1								Data_0							
R-0h								R-0h								R-0h								R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-62. IF3DATA Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	Data_3	R	0h	Data 3.
23-16	Data_2	R	0h	Data 2.
15-8	Data_1	R	0h	Data 1.
7-0	Data_0	R	0h	Data 0.

23.4.1.50 IF3DATB Register (offset = 154h) [reset = 0h]

IF3DATB is shown in Figure 23-68 and described in Table 23-63.

The data bytes of CAN messages are stored in the IF3 registers in the following order. In a CAN data frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Figure 23-68. IF3DATB Register

31-24	23-16	15-8	7-0
Data_7	Data_6	Data_5	Data_4
R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 23-63. IF3DATB Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	Data_7	R	0h	Data 7.
23-16	Data_6	R	0h	Data 6.
15-8	Data_5	R	0h	Data 5.
7-0	Data_4	R	0h	Data 4.

23.4.1.51 IF3UPD12 Register (offset = 160h) [reset = 0h]

IF3UPD12 is shown in Figure 23-69 and described in Table 23-64.

The automatic update functionality of the IF3 register set can be configured for each message object. A message object is enabled for automatic IF3 update, if the dedicated IF3UpdEn flag is set. This means that an active NewDat flag of this message object (e.g due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set. IF3 Update enable should not be set for transmit objects.

Figure 23-69. IF3UPD12 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
IF3UpdEn_32:17_																IF3UpdEn_16:1_															
R/W-0h																R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-64. IF3UPD12 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	IF3UpdEn_32:17_	R/W	0h	IF3 Update Enabled (for all message objects) 0h (R/W) = Automatic IF3 update is disabled for this message object. 1h (R/W) = Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active.
15-0	IF3UpdEn_16:1_	R/W	0h	IF3 Update Enabled (for all message objects) 0h (R/W) = Automatic IF3 update is disabled for this message object. 1h (R/W) = Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active.

23.4.1.52 IF3UPD34 Register (offset = 164h) [reset = 0h]

IF3UPD34 is shown in Figure 23-70 and described in Table 23-65.

The automatic update functionality of the IF3 register set can be configured for each message object. A message object is enabled for automatic IF3 update, if the dedicated IF3UpdEn flag is set. This means that an active NewDat flag of this message object (e.g due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set. IF3 Update enable should not be set for transmit objects.

Figure 23-70. IF3UPD34 Register

31-16	15-0
IF3UpdEn_64:49_	IF3UpdEn_48:33_
R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-65. IF3UPD34 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	IF3UpdEn_64:49_	R/W	0h	IF3 Update Enabled (for all message objects) 0h (R/W) = Automatic IF3 update is disabled for this message object. 1h (R/W) = Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active.
15-0	IF3UpdEn_48:33_	R/W	0h	IF3 Update Enabled (for all message objects) 0h (R/W) = Automatic IF3 update is disabled for this message object. 1h (R/W) = Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active.

23.4.1.53 IF3UPD56 Register (offset = 168h) [reset = 0h]

IF3UPD56 is shown in Figure 23-71 and described in Table 23-66.

The automatic update functionality of the IF3 register set can be configured for each message object. A message object is enabled for automatic IF3 update, if the dedicated IF3UpdEn flag is set. This means that an active NewDat flag of this message object (e.g due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set. IF3 Update enable should not be set for transmit objects.

Figure 23-71. IF3UPD56 Register

31-16	15-0
IF3UpdEn_96:81_	IF3UpdEn_80:65_
R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-66. IF3UPD56 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	IF3UpdEn_96:81_	R/W	0h	IF3 Update Enabled (for all message objects) 0h (R/W) = Automatic IF3 update is disabled for this message object. 1h (R/W) = Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active.
15-0	IF3UpdEn_80:65_	R/W	0h	IF3 Update Enabled (for all message objects) 0h (R/W) = Automatic IF3 update is disabled for this message object. 1h (R/W) = Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active.

23.4.1.54 IF3UPD78 Register (offset = 16Ch) [reset = 0h]

IF3UPD78 is shown in Figure 23-72 and described in Table 23-67.

The automatic update functionality of the IF3 register set can be configured for each message object. A message object is enabled for automatic IF3 update, if the dedicated IF3UpdEn flag is set. This means that an active NewDat flag of this message object (e.g due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set. IF3 Update enable should not be set for transmit objects.

Figure 23-72. IF3UPD78 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IF3UpdEn_128:113_															
R/W-0h															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IF3UpdEn_112:97_															
R/W-0h															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-67. IF3UPD78 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	IF3UpdEn_128:113_	R/W	0h	IF3 Update Enabled (for all message objects) 0h (R/W) = Automatic IF3 update is disabled for this message object. 1h (R/W) = Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active.
15-0	IF3UpdEn_112:97_	R/W	0h	IF3 Update Enabled (for all message objects) 0h (R/W) = Automatic IF3 update is disabled for this message object. 1h (R/W) = Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active.

23.4.1.55 TIOC Register (offset = 1E0h) [reset = 0h]

TIOC is shown in Figure 23-73 and described in Table 23-68.

The CAN_TX pin of the DCAN module can be used as general purpose IO pin if CAN function is not needed. The values of the IO control registers are only writable if Init bit of the CAN control register is set. The OD, Func, Dir and Out bits of the CAN TX IO control register are forced to certain values when Init bit of CAN control register is reset (see bit descriptions).

Figure 23-73. TIOC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED					PU	PD	OD
R-0h					R/W-0h	R/W-0h	0h
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESERVED							
R-0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED				Func	Dir	Out	In
R-0h				0h	0h	0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-68. TIOC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	
18	PU	R/W	0h	CAN_TX pull up/pull down select. This bit is only active when CAN_TX is configured to be an input. 0h (R/W) = CAN_TX pull down is selected, when pull logic is active (PD = 0). 1h (R/W) = CAN_TX pull up is selected, when pull logic is active (PD = 0).
17	PD	R/W	0h	CAN_TX pull disable. This bit is only active when CAN_TX is configured to be an input. 0h (R/W) = CAN_TX pull is active 1h (R/W) = CAN_TX pull is disabled
16	OD		0h	CAN_TX open drain enable. This bit is only active when CAN_TX is configured to be in GIO mode (TIOC.Func=0). Forced to '0' if Init bit of CAN control register is reset. 0h (R/W) = The CAN_TX pin is configured in push/pull mode. 1h (R/W) = The CAN_TX pin is configured in open drain mode.
15-4	RESERVED	R	0h	
3	Func		0h	CAN_TX function. This bit changes the function of the CAN_TX pin. Forced to '1' if Init bit of CAN control register is reset. 0h (R/W) = CAN_TX pin is in GIO mode. 1h (R/W) = CAN_TX pin is in functional mode (as an output to transmit CAN data).
2	Dir		0h	CAN_TX data direction. This bit controls the direction of the CAN_TX pin when it is configured to be in GIO mode only (TIOC.Func=0). Forced to '1' if Init bit of CAN control register is reset. 0h (R/W) = The CAN_TX pin is an input. 1h (R/W) = The CAN_TX pin is an output

Table 23-68. TIOC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	Out		0h	CAN_TX data out write. This bit is only active when CAN_TX pin is configured to be in GIO mode (TIOC.Func = 0) and configured to be an output pin (TIOC.Dir = 1). The value of this bit indicates the value to be output to the CAN_TX pin. Forced to Tx output of the CAN core, if Init bit of CAN control register is reset. 0h (R/W) = The CAN_TX pin is driven to logic low 1h (R/W) = The CAN_TX pin is driven to logic high
0	In	R	0h	CAN_TX data in. Note: When CAN_TX pin is connected to a CAN transceiver, an external pullup resistor has to be used to ensure that the CAN bus will not be disturbed (e.g. while reset of the DCAN module). 0h (R/W) = The CAN_TX pin is at logic low 1h (R/W) = The CAN_TX pin is at logic high

23.4.1.56 RIOC Register (offset = 1E4h) [reset = 0h]

RIOC is shown in Figure 23-74 and described in Table 23-69.

The CAN_RX pin of the DCAN module can be used as general purpose IO pin if CAN function is not needed. The values of the IO control registers are writable only if Init bit of CAN control register is set. The OD, Func and Dir bits of the CAN RX IO control register are forced to certain values when the Init bit of CAN control register is reset (see bit descriptions).

Figure 23-74. RIOC Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED					PU	PD	OD
R-0h					R/W-0h	R/W-0h	0h
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESERVED							
R-0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED				Func	Dir	Out	In
R-0h				0h	0h	0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 23-69. RIOC Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	
18	PU	R/W	0h	CAN_RX pull up/pull down select. This bit is only active when CAN_RX is configured to be an input. 0h (R/W) = CAN_RX pull down is selected, when pull logic is active (PD = 0). 1h (R/W) = CAN_T=RX pull up is selected, when pull logic is active(PD = 0).
17	PD	R/W	0h	CAN_RX pull disable. This bit is only active when CAN_TX is configured to be an input. 0h (R/W) = CAN_RX pull is active 1h (R/W) = CAN_RX pull is disabled
16	OD		0h	CAN_RX open drain enable. This bit is only active when CAN_RX is configured to be in GIO mode (TIOC.Func=0). Forced to '0' if Init bit of CAN control register is reset. 0h (R/W) = The CAN_RX pin is configured in push/pull mode. 1h (R/W) = The CAN_RX pin is configured in open drain mode.
15-4	RESERVED	R	0h	
3	Func		0h	CAN_RX function. This bit changes the function of the CAN_RX pin. Forced to '1' if Init bit of CAN control register is reset. 0h (R/W) = CAN_RX pin is in GIO mode. 1h (R/W) = CAN_RX pin is in functional mode (as an output to transmit CAN data).
2	Dir		0h	CAN_RX data direction. This bit controls the direction of the CAN_RX pin when it is configured to be in GIO mode only (TIOC.Func=0). Forced to '1' if Init bit of CAN control register is reset. 0h (R/W) = The CAN_RX pin is an input. 1h (R/W) = The CAN_RX pin is an output

Table 23-69. RIOC Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	Out		0h	CAN_RX data out write. This bit is only active when CAN_RX pin is configured to be in GIO mode (TIOC.Func = 0) and configured to be an output pin (TIOC.Dir = 1). The value of this bit indicates the value to be output to the CAN_RX pin. Forced to Tx output of the CAN core, if Init bit of CAN control register is reset. 0h (R/W) = The CAN_RX pin is driven to logic low 1h (R/W) = The CAN_RX pin is driven to logic high
0	In	R	0h	CAN_RX data in. Note: When CAN_RX pin is connected to a CAN transceiver, an external pullup resistor has to be used to ensure that the CAN bus will not be disturbed (for example, while reset of the DCAN module). 0h (R/W) = The CAN_RX pin is at logic low 1h (R/W) = The CAN_RX pin is at logic high