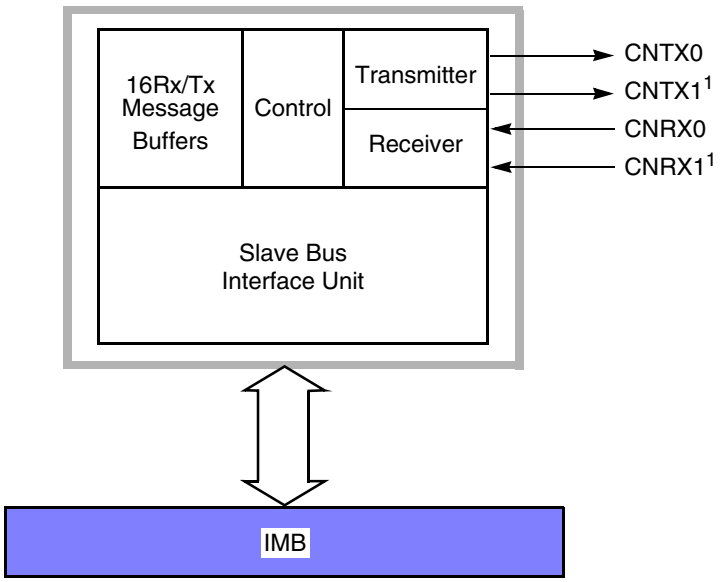
SECTION 16
CAN 2.0B CONTROLLER MODULE

The MPC555 / MPC556 contains two CAN 2.0B controller modules (TouCAN). Each TouCAN is a communication controller that implements the controller area network (CAN) protocol, an asynchronous communications protocol used in automotive and industrial control systems. It is a high speed (1 Mbit/sec), short distance, priority based protocol that can run over a variety of mediums (for example, fiber optic cable or an unshielded twisted pair of wires). The TouCAN supports both the standard and extended identifier (ID) message formats specified in the CAN protocol specification, revision 2.0, part B.

Each TouCAN module contains 16 message buffers, which are used for transmit and receive functions. It also contains message filters, which are used to qualify the received message IDs when comparing them to the receive buffer identifiers.

Figure 16-1 shows a block diagram of a TouCAN module.



1. In the MPC555 / MPC556, the CNTX1 and CNRX1 signals are not available.

Figure 16-1 TouCAN Block Diagram

16.1 Features

Each TouCAN module provides these features:

- Full implementation of CAN protocol specification, version 2.0 A/B
  — Standard data and remote frames (up to 109 bits long)

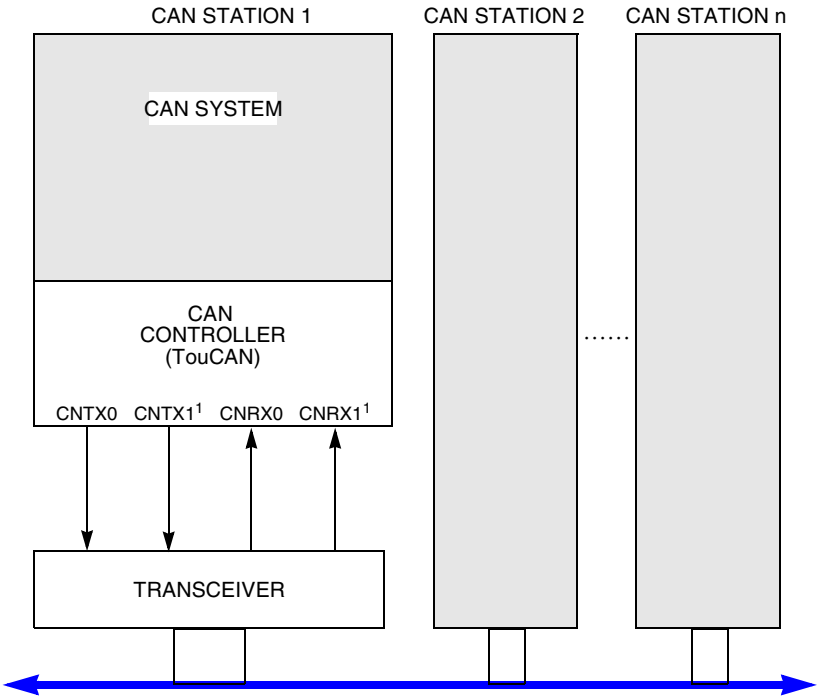
  - — Extended data and remote frames (up to 127 bits long)
  - — 0 to 8 bytes data length
  - — Programmable bit rate up to 1 Mbit/sec
- 16 RX/TX message buffers of 0-8 bytes data length
- Content-related addressing
- No read/write semaphores required
- Three programmable mask registers: global (for message buffers zero through 13), special for message buffer 14, and special for message buffer 15
- Programmable transmit-first scheme: lowest ID or lowest buffer number
- “Time stamp”, based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Programmable I/O modes
- Maskable interrupts
- Independent of the transmission medium (external transceiver is assumed)
- Open network architecture
- Multimaster concept
- High immunity to EMI
- Short latency time for high-priority messages
- Low power sleep mode with programmable wakeup on bus activity
- Outputs have open drain drivers
- Can be used to implement recommended practices SAE J1939 and SAE J2284
- Can also be used to implement popular industrial automation standards such as DeviceNet™ and Smart Distributed System
- Motorola IMB-family modular architecture

16.2 External Pins

The TouCAN module interface to the CAN bus consists of four pins: CANTX0 and CANTX1, which transmit serial data, and CANRX0 and CANRX1, which receive serial In the MPC555 / MPC556, the CNTX1 and CNRX1 signals are not available.

NOTE

In the MPC555 / MPC556, the CNTX1 and CNRX1 signals are not available.



1. In the MPC555, the CNTX1 and CNRX1 signals are not available.

Figure 16-2 Typical CAN Network

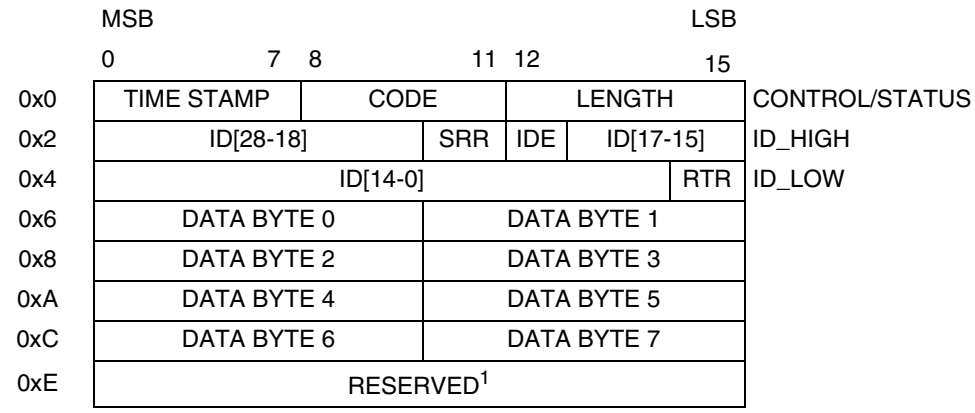
Each CAN station is connected physically to the CAN bus through a transceiver. The transceiver provides the transmit drive, waveshaping, and receive/compare functions required for communicating on the CAN bus. It can also provide protection against damage to the TouCAN caused by a defective CAN bus or a defective CAN station.

16.3 TouCAN Architecture

The TouCAN module uses a flexible design that allows each of its 16 message buffers to be designated either a transmit (Tx) buffer or a receive (Rx) buffer. In addition, to reduce the CPU overhead required for message handling, each message buffer is assigned an interrupt flag bit to indicate that the transmission or reception completed successfully.

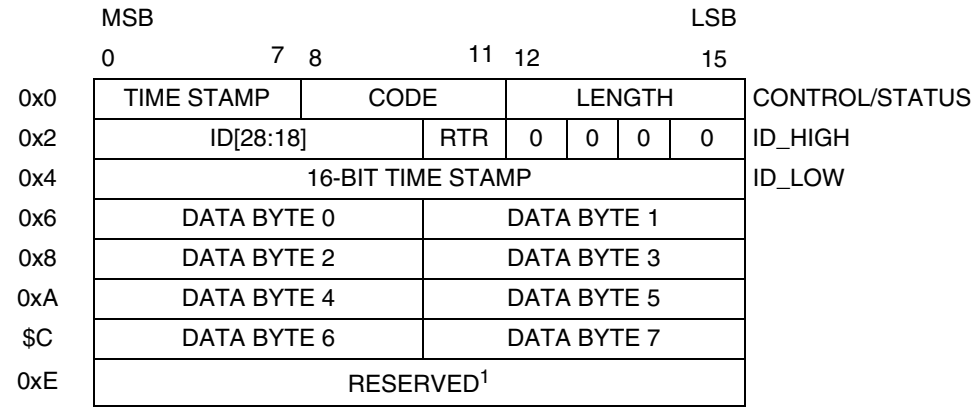
16.3.1 TX/RX Message Buffer Structure

Figure 16-3 displays the extended (29-bit) ID message buffer structure. **Figure 16-4** displays the standard (11-bit) ID message buffer structure.



NOTES:
1. The reading of a reserved location in memory may cause an RCPU exception.

Figure 16-3 Extended ID Message Buffer Structure



NOTES:
1. The reading of a reserved location in memory may cause an RCPU exception.

Figure 16-4 Standard ID Message Buffer Structure

16.3.1.1 Common Fields for Extended and Standard Format Frames

Table 16-1 describes the message buffer fields that are common to both extended and standard identifier format frames.

Table 16-1 Common Extended/Standard Format Frames

Field	Description
Time Stamp	Contains a copy of the high byte of the free running timer, which is captured at the beginning of the identifier field of the frame on the CAN bus.
Code	Refer to **Table 16-2** and **Table 16-3**.
Rx Length	Length (in bytes) of the Rx data stored in offset 0x6 through 0xD of the buffer.This field is written by the TouCAN module, copied from the DLC (data length code) field of the received frame.
Tx Length	Length (in bytes) of the data to be transmitted, located in offset 0x6 through 0xD of the buffer. This field is written by the CPU and is used as the DLC field value. If RTR (remote transmission request) = 1, the frame is a remote frame and will be transmitted without data field, regardless of the value in Tx length.
Data	This field can store up to eight data bytes for a frame. For Rx frames, the data is stored as it is received from the bus. For Tx frames, the CPU provides the data to be transmitted within the frame.
Reserved	The CPU controls access to this word entry field (16 bits).

Table 16-2 Message Buffer Codes for Receive Buffers

Rx Code Before Rx New Frame	Description	Rx Code After Rx New Frame	Comment
0b0000	NOT ACTIVE — message buffer is not active.	—	—
0b0100	EMPTY — message buffer is active and empty.	0b0010	—
0b0010	FULL — message buffer is full.	0b0110	If a CPU read occurs before the new frame, new receive code is 0010.
0b0110	OVERRUN — second frame was received into a full buffer before the CPU read the first one.		
0b0XY1[1]	BUSY — message buffer is now being filled with a new receive frame. This condition will be cleared within 20 cycles.	0b0010	An empty buffer was filled (XY was 10).
		0b0110	A full/overrun buffer was filled (Y was 1).

NOTES:
1. For Tx message buffers, upon read, the BUSY bit should be ignored.

Table 16-3 Message Buffer Codes for Transmit Buffers

RTR	Initial Tx Code	Description	Code After Successful Transmission
X	0b1000	Message buffer not ready for transmit.	—
0	0b1100	Data frame to be transmitted once, unconditionally.	0b1000
1	0b1100	Remote frame to be transmitted once, and message buffer becomes an RX message buffer for data frames.	0b0100
0	0b1010[1]	Data frame to be transmitted only as a response to a remote frame, always.	0b1010
0	0b1110	Data frame to be transmitted only once, unconditionally, and then only as a response to remote frame, always.	0b1010

NOTES:
1. When a matching remote request frame is detected, the code for such a message buffer is changed to be 1110.

16.3.1.2 Fields for Extended Format Frames

Table 16-4 describes the message buffer fields used only for extended identifier format frames.

Table 16-4 Extended Format Frames

Field	Description
ID[28:18]/[17:15]	Contains the 14 most significant bits of the extended identifier, located in the ID HIGH word of the message buffer.
Substitute Remote Request (SRR)	Contains a fixed recessive bit, used only in extended format. Should be set to one by the user for Tx buffers. It will be stored as received on the CAN bus for Rx buffers.
ID Extended (IDE)	If extended format frame is used, this field should be set to one. If zero, standard format frame should be used.
ID[14:0]	Bits [14:0] of the extended identifier, located in the ID LOW word of the message buffer.
Remote Transmission Request (RTR)	This bit is located in the least significant bit of the ID LOW word of the message buffer; 0 = Data Frame, 1 = Remote Frame.

16.3.1.3 Fields for Standard Format Frames

Table 16-5 describes the message buffer fields used only for standard identifier format frames.

Table 16-5 Standard Format Frames

Field	Description
16-bit Time Stamp	The ID LOW word, which is not needed for standard format, is used in a standard format buffer to store the 16-bit value of the free-running timer which is captured at the beginning of the identifier field of the frame on the CAN bus.
ID[28:18]	Contains bits [28:18] of the identifier, located in the ID HIGH word of the message buffer. The four least significant bits in this register (corresponding to the IDE bit and ID[17:15] for an extended identifier message) must all be written as logic zeros to ensure proper operation of the TouCAN.
RTR	This bit is located in the ID HIGH word of the message buffer; 0 = data frame, 1 = remote frame.
RTR/SRR Bit Treatment	If the TouCAN transmits this bit as a one and receives it as a zero, an “arbitration loss” is indicated. If the TouCAN transmits this bit as a zero and is receives it as a one, a bit error is indicated. If the TouCAN transmits a value and receives a matching response, a successful bit transmission is indicated.

16.3.1.4 Serial Message Buffers

To allow double buffering of messages, the TouCAN has two shadow buffers called serial message buffers. The TouCAN uses these two buffers for buffering both received messages and messages to be transmitted. Only one serial message buffer is active at a time, and its function depends upon the operation of the TouCAN at that time. At no time does the user have access to or visibility of these two buffers.

16.3.1.5 Message Buffer Activation/Deactivation Mechanism

Each message buffer must be activated once the user configures it for the desired operation. A buffer is activated by writing the appropriate code to the control/status word for that buffer. Once the buffer is activated, it will begin participating in the normal transmit and receive processes.

A buffer is deactivated by writing the appropriate deactivation code to the control/status word for that buffer. A buffer is typically deactivated when the user desires to reconfigure the buffer (for example to change the buffer's function from Rx to Tx or Tx to Rx). The buffer should also be deactivated before changing a receive buffer's message identifier or before loading a new message to be transmitted into a transmit buffer.

For more details on activation and deactivation of message buffers and the effects on message buffer operation, refer to **16.4 TouCAN Operation**.

16.3.1.6 Message Buffer Lock/Release/Busy Mechanism

In addition to the activation/deactivation mechanism, the TouCAN also uses a lock/release/busy mechanism to ensure data coherency during the receive process. The mechanism includes a lock status for each message buffer and uses the two serial message buffers to facilitate frame transfers within the TouCAN.

Reading the control/status word of a receive message buffer triggers the lock for that buffer. While locked, a received message cannot be transferred into that buffer from one of the serial message buffers.

If a message transfer between the message buffer and a serial message buffer is in progress when the control/status word is read, the BUSY status is indicated in the code field, and the lock is not activated.

The user can release the lock on a message buffer in one of two ways. Reading the control/status word of another message buffer locks that buffer, releasing the previously locked buffer. A global release can also be performed on any locked message buffer by reading the free-running timer.

Once a lock is released, any message transfers between a serial message buffer and a message buffer that were delayed due to that buffer being locked will take place. For more details on the message buffer locking mechanism, and the effects on message buffer operation, refer to **16.4 TouCAN Operation**.

16.3.2 Receive Mask Registers

The receive mask registers are used as acceptance masks for received frame IDs. The following masks are defined:

- A global mask, used for receive buffers 0-13
- Two separate masks for buffers 14 and 15

The value of the mask registers should not be changed during normal operation. If the mask register data is changed after the masked identifier of a received message is

matched to a locked message buffer, that message will be transferred into that message buffer once it is unlocked, regardless of whether that message's masked identifier still matches the receive buffer identifier. Table 16-6 shows mask bit values.

Table 16-6 Receive Mask Register Bit Values

Mask Bit	Values
0	The corresponding incoming ID bit is "don't care"
1	The corresponding ID bit is checked against the incoming ID bit to see if a match exists

Table 16-7 shows mask examples for normal and extended messages. Refer to 16.7 Programmer's Model for more information on Rx mask registers.

Table 16-7 Mask Examples for Normal/Extended Messages

Message Buffer (MB) /Mask	Base ID ID[28:18]	IDE	Extended ID ID[17:0]	Match
MB2	1 1 1 1 1 1 1 1 0 0 0	0	—	—
MB3	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
MB4	0 0 0 0 0 0 0 1 1 1 1	0	—	—
MB5	0 0 0 0 0 0 0 1 1 1 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
MB14	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
RX Global Mask	1 1 1 1 1 1 1 1 1 1 0		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1	—
RX Message In	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	3[1]
	1 1 1 1 1 1 1 1 0 0 1	0	—	2[2]
	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0	—[3]
	0 1 1 1 1 1 1 1 0 0 0	0	—	—[4]
	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—[5]
RX 14 Mask	0 1 1 1 1 1 1 1 1 1 1		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0	—
RX Message In	1 0 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—[6]
	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	14[7]

NOTES:
1. Match for extended format (MB3).
2. Match for standard format (MB2).
3. No match for MB3 because of ID0.
4. No match for MB2 because of ID28.
5. No match for MB3 because of ID28, match for MB14.
6. No match for MB14 because of ID27.
7. Match for MB14.

16.3.3 Bit Timing

The TouCAN module uses three 8-bit registers to set up the bit timing parameters required by the CAN protocol. Control registers one and two (CANCTRL1, CANCTRL2) contain the PROPSEG, PSEG1, PSEG2, and the RJW fields which allow the user to configure the bit timing parameters. The prescaler divide register (PRESDIV) allows the user to select the ratio used to derive the S-clock from the IMB clock. The time

quanta clock operates at the S-clock frequency. **Table 16-8** provides examples of IMB clock, CAN bit rate, and S-clock bit timing parameters. Refer to **16.7 Programmer's Model** for more information on the bit timing registers.

Table 16-8 Example IMB Clock, CAN Bit Rate and S-Clock Frequencies

IMB Clock Frequency (MHz)	CAN Bit Rate (MHz)	Possible S-Clock Frequency (MHz)	Possible Number of Time Quanta/Bit	PRESDIV Value + 1
25	1	25	25	1
20	1	10, 20	10, 20	2, 1
16	1	8, 16	8, 16	2, 1
25	0.125	1, 1.25, 2.5	8,10, 20	25, 20,10
20	0.125	1, 2, 2.5	8, 16, 20	20, 10, 8
16	0.125	1, 2	8,16	16, 8

16.3.3.1 Configuring the TouCAN Bit Timing

The following considerations must be observed when programming bit timing functions.

- If the programmed PRESDIV value results in a single IMB clock per one time quantum, then the PSEG2 field in CANCTRL2 register must not be programmed to zero.
- If the programmed PRESDIV value results in a single IMB clock per one time quantum, then the information processing time (IPT) equals three time quanta; otherwise it equals two time quanta. If PSEG2 equals two, then the TouCAN transmits one time quantum late relative to the scheduled sync segment.
- If the prescaler and bit timing control fields are programmed to values that result in fewer than 10 IMB clock periods per CAN bit time and the CAN bus loading is 100%, then any time the rising edge of a start-of-frame (SOF) symbol transmitted by another node occurs during the third bit of the intermission between messages, the TouCAN may not be able to prepare a message buffer for transmission in time to begin its own transmission and arbitrate against the message which transmitted the early SOF.
- The TouCAN bit time must be programmed to be greater than or equal to nine IMB clocks, or correct operation is not guaranteed.

16.3.4 Error Counters

The TouCAN has two error counters, the transmit (Tx) error counter and the receive (Rx) error counter. Refer to **16.7 Programmer's Model** for more information on error counters.The rules for increasing and decreasing these counters are described in the CAN protocol, and are fully implemented in the TouCAN. Each counter has the following features:

- 8-bit up/down-counter
- Increment by eight (Rx error counter also increments by one)
- Decrement by one
- Avoid decrement when equal to zero

- Rx error counter reset to a value between 119 and 127 inclusive, when the TouCAN transitions from error passive to error active
- Following reset, both counters reset to zero
- Detect values for error passive, bus off and error active transitions
- Cascade usage of Tx error counter with an additional internal counter to detect the 128 occurrences of 11 consecutive recessive bits necessary to transition from bus off into error active.

Both counters are read-only (except in test/freeze/halt modes).

The TouCAN responds to any bus state as described in the CAN protocol, transmitting an error active or error passive flag, delaying its transmission start time (error passive) and avoiding any influence on the bus when in the bus off state. The following are the basic rules for TouCAN bus state transitions:

- If the value of the Tx error counter or Rx error counter increments to a value greater than or equal to 128, the fault confinement state (FCS[1:0]) field in the error status register is updated to reflect an error passive state.
- If the TouCAN is in an error passive state, and either the Tx error counter or Rx error counter decrements to a value less than or equal to 127 while the other error counter already satisfies this condition, the FCS[1:0] field in the error status register is updated to reflect an error active state.
- If the value of the Tx error counter increases to a value greater than 255, the FCS[1:0] field in the error status register is updated to reflect a bus off state, and an interrupt may be issued. The value of the Tx error counter is reset to zero.
- If the TouCAN is in the bus off state, the Tx error counter and an additional internal counter are cascaded to count 128 occurrences of 11 consecutive recessive bits on the bus. To do this, the Tx error counter is first reset to zero, and then the internal counter begins counting consecutive recessive bits. Each time the internal counter counts 11 consecutive recessive bits, the Tx error counter is incremented by one and the internal counter is reset to zero. When the Tx error counter reaches the value of 128, the FCS[1:0] field in the error status register is updated to be error active, and both error counters are reset to zero. Any time a dominant bit is detected following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero but does not affect the Tx error counter value.
- If only one node is operating in a system, the Tx error counter is incremented with each message it attempts to transmit, due to the resulting acknowledgment errors. However, acknowledgment errors never cause the TouCAN to change from the error passive state to the bus off state.
- If the Rx error counter increments to a value greater than 127, it stops incrementing, even if more errors are detected while being a receiver. After the next successful message reception, the counter is reset to a value between 119 and 127, to enable a return to the error active state.

16.3.5 Time Stamp

The value of the free-running 16-bit timer is sampled at the beginning of the identifier field on the CAN bus. For a message being received, the time stamp is stored in the time stamp entry of the receive message buffer at the time the message is written into

that buffer. For a message being transmitted, the time stamp entry is written into the transmit message buffer once the transmission has completed successfully.

The free-running timer can optionally be reset upon the reception of a frame into message buffer 0. This feature allows network time synchronization to be performed.

16.4 TouCAN Operation

The basic operation of the TouCAN can be divided into three areas:

- Reset and initialization of the module
- Transmit message handling
- Receive message handling

Example sequences for performing each of these processes is given in the following paragraphs.

16.4.1 TouCAN Reset

The TouCAN can be reset in two ways:

- Hard reset, using one of the IMB3 reset lines
- Soft reset, using the SOFTRST bit in the module configuration register

Following the negation of reset, the TouCAN is not synchronized with the CAN bus, and the HALT, FRZ, and FRZACK bits in the module configuration register are set. In this state, the TouCAN does not initiate frame transmissions or receive any frames from the CAN bus. The contents of the message buffers are not changed following reset.

Any configuration change or initialization requires that the TouCAN be frozen by either the assertion of the HALT bit in the module configuration register or by reset.

16.4.2 TouCAN Initialization

Initialization of the TouCAN includes the initial configuration of the message buffers and configuration of the CAN communication parameters following a reset, as well as any reconfiguration which may be required during operation. The following is a general initialization sequence for the TouCAN:

1. Initialize all operation modes
   a. Initialize the transmit and receive pin modes in control register 0 (CANCTRL0)
   b. Initialize the bit timing parameters PROPSEG, PSEGS1, PSEG2, and RJW in control registers 1 and 2 (CANCTRL[1:2])
   c. Select the S-clock rate by programming the PRESDIV register
   d. Select the internal arbitration mode (LBUF bit in CANCTRL1)
2. Initialize message buffers

a. The control/status word of all message buffers must be written either as an active or inactive message buffer.

b. All other entries in each message buffer should be initialized as required

3. Initialize mask registers for acceptance mask as needed
4. Initialize TouCAN interrupt handler

a. Initialize the interrupt configuration register (CANICR) with a specific request level

b. Set the required mask bits in the IMASK register (for all message buffer interrupts), in CANCTRL0 (for bus off and error interrupts), and in CANMCR for the WAKE interrupt

5. Negate the HALT bit in the module configuration register

a. At this point, the TouCAN attempts to synchronize with the CAN bus

NOTE

In both the transmit and receive processes, the first action in preparing a message buffer must be to deactivate the buffer by setting its code field to the proper value. This step is mandatory to ensure data coherency.

16.4.3 Transmit Process

The transmit process includes preparation of a message buffer for transmission, as well as the internal steps performed by the TouCAN to decide which message to transmit. For the user, this involves loading the message and ID to be transmitted into a message buffer and then activating that buffer as an active transmit buffer. Once this is done, the TouCAN performs all additional steps necessary to transmit the message onto the CAN bus.

The user should prepare or change a message buffer for transmission by executing the following steps.

1. Write the control/status word to hold the transmit buffer inactive (code = 0b1000)
2. Write the ID_HIGH and ID_LOW words
3. Write the data bytes
4. Write the control/status word (active Tx code, Tx length)

NOTE

Steps one and four are mandatory to ensure data coherency.

Once an active transmit code is written to a transmit message buffer, that buffer begins participating in an internal arbitration process as soon as the receiver senses that the CAN bus is free, or at the inter-frame space. If there are multiple messages awaiting transmission, this internal arbitration process selects the message buffer from which the next frame is transmitted.

When this process is over and a message buffer is selected for transmission, the frame from that message buffer is transferred to the serial message buffer for transmission.

The TouCAN transmits no more than eight data bytes, even if the transmit length contains a value greater than eight.

At the end of a successful transmission, the value of the free-running timer (which was captured at the beginning of the identifier field on the CAN bus), is written into the time stamp field in the message buffer. The code field in the control/status word of the message buffer is updated and a status flag is set in the IFLAG register.

16.4.3.1 Transmit Message Buffer Deactivation

Any write access to the control/status word of a transmit message buffer during the process of selecting a message buffer for transmission immediately deactivates that message buffer, removing it from the transmission process.

If the user deactivates a transmit message buffer while a message is being transferred from it to a serial message buffer, the message is not transmitted.

If the user deactivates the transmit message buffer after the message is transferred to the serial message buffer, the message is transmitted, but no interrupt is requested, and the transmit code is not updated.

If a message buffer containing the lowest ID is deactivated while that message is undergoing the internal arbitration process to determine which message should be sent, then that message may not be transmitted.

16.4.3.2 Reception of Transmitted Frames

The TouCAN receives a frame it has transmitted if an empty message buffer with a matching identifier exists.

16.4.4 Receive Process

During the receive process, the following events occur:

- The user configures the message buffers for reception
- The TouCAN transfers received messages from the serial message buffers to the receive message buffers with matching IDs
- The user retrieves these messages

The user should prepare or change a message buffer for frame reception by executing the following steps.

1. Write the control/status word to hold the receive buffer inactive (code = 0b0000)
2. Write the ID_HIGH and ID_LOW words
3. Write the control/status word to mark the receive message buffer as active and empty

NOTE

Steps one and three are mandatory for data coherency.

Once these steps are performed, the message buffer functions as an active receive buffer and participates in the internal matching process, which takes place every time the TouCAN receives an error-free frame. In this process, all active receive buffers compare their ID value to the newly received one. If a match is detected, the following actions occur:

1. The frame is transferred to the first (lowest entry) matching receive message buffer
2. The value of the free-running timer (captured at the beginning of the identifier field on the CAN bus) is written into the time stamp field in the message buffer
3. The ID field, data field, and RX length field are stored
4. The code field is updated
5. The status flag is set in the IFLAG register

The user should read a received frame from its message buffer in the following order:

1. Control/status word (mandatory, as it activates the internal lock for this buffer)
2. ID (optional, since it is needed only if a mask was used)
3. Data field word(s)
4. Free-running timer (optional, as it releases the internal lock)

If the free running timer is not read, that message buffer remains locked until the read process starts for another message buffer. Only a single message buffer is locked at a time. When a received message is read, the only mandatory read operation is that of the control/status word. This ensures data coherency.

If the BUSY bit is set in the message buffer code, the CPU should defer accessing that buffer until this bit is negated. Refer to **Table 16-2**.

NOTE

The user should check the status of a message buffer by reading the status flag in the IFLAG register and not by reading the control/status word code field for that message buffer. This prevents the buffer from being locked inadvertently.

Because the received identifier field is always stored in the matching receive message buffer, the contents of the identifier field in a receive message buffer may change if one or more of the ID bits are masked.

16.4.4.1 Receive Message Buffer Deactivation

Any write access to the control/status word of a receive message buffer during the process of selecting a message buffer for reception immediately deactivates that message buffer, removing it from the reception process.

If a receive message buffer is deactivated while a message is being transferred into it, the transfer is halted and no interrupt is requested. If this occurs, that receive message buffer may contain mixed data from two different frames.

Data must never be written into a receive message buffer. If this occurs while a message is being transferred from a serial message buffer, the control/status word will reflect a full or overrun condition, but no interrupt is requested.

16.4.4.2 Locking and Releasing Message Buffers

The lock/release/busy mechanism is designed to guarantee data coherency during the receive process. The following examples demonstrate how the the lock/release/busy mechanism affects TouCAN operation.

1. Reading a control/status word of a message buffer triggers a lock for that message buffer. A new received message frame which matches the message buffer cannot be written into this message buffer while it is locked.
2. To release a locked message buffer, the CPU either locks another message buffer by reading its control/status word or globally releases any locked message buffer by reading the free-running timer.
3. If a receive frame with a matching ID is received during the time the message buffer is locked, the receive frame is not immediately transferred into that message buffer, but remains in the serial message buffer. There is no indication when this occurs.
4. When a locked message buffer is released, if a frame with a matching identifier exists within the serial message buffer, then this frame is transferred to the matching message buffer.
5. If two or more receive frames with matching IDs are received while a message buffer with a matching ID is locked, the last received frame with that ID is kept within the serial message buffer, while all preceding ones are lost. There is no indication when this occurs.
6. If the user reads the control/status word of a receive message buffer while a frame is being transferred from a serial message buffer, the BUSY code is indicated. The user should wait until this code is cleared before continuing to read from the message buffer to ensure data coherency. In this situation, the read of the control/status word does not lock the message buffer.

Polling the control/status word of a receive message buffer can lock it, preventing a message from being transferred into that buffer. If the control/status word of a receive message buffer is read, it should be followed by a read of the control/status word of another buffer, or by a read of the free-running timer, to ensure that the locked buffer is unlocked.

16.4.5 Remote Frames

The remote frame is a message frame that is transmitted to request a data frame. The TouCAN can be configured to transmit a data frame automatically in response to a remote frame, or to transmit a remote frame and then wait for the responding data frame to be received.

To transmit a remote frame, the user initializes a message buffer as a transmit message buffer with the RTR bit set to one. Once this remote frame is transmitted successfully, the transmit message buffer automatically becomes a receive message buffer, with the same ID as the remote frame that was transmitted.

When the TouCAN receives a remote frame, it compares the remote frame ID to the IDs of all transmit message buffers programmed with a code of 1010. If there is an exact matching ID, the data frame in that message buffer is transmitted. If the RTR bit in the matching transmit message buffer is set, the TouCAN transmits a remote frame as a response.

A received remote frame is not stored in a receive message buffer. It is only used to trigger the automatic transmission of a frame in response. The mask registers are not used in remote frame ID matching. All ID bits (except RTR) of the incoming received frame must match for the remote frame to trigger a response transmission.

16.4.6 Overload Frames

The TouCAN does not initiate overload frame transmissions unless it detects the following conditions on the CAN bus:

- A dominant bit in the first or second bit of intermission
- A dominant bit in the seventh (last) bit of the end-of-frame (EOF) field in receive frames
- A dominant bit in the eighth (last) bit of the error frame delimiter or overload frame delimiter

16.5 Special Operating Modes

The TouCAN module has three special operating modes:

- Debug mode
- Low-power stop mode
- Auto power save mode

16.5.1 Debug Mode

Debug mode is entered when the FRZ1 bit in CANMCR is set and one of the following events occurs:

- The HALT bit in the CANMCR is set; or
- The IMB3 FREEZE line is asserted

Once entry into debug mode is requested, the TouCAN waits until an intermission or idle condition exists on the CAN bus, or until the TouCAN enters the error passive or bus off state. Once one of these conditions exists, the TouCAN waits for the completion of all internal activity. Once this happens, the following events occur:

- The TouCAN stops transmitting or receiving frames
- The prescaler is disabled, thus halting all CAN bus communication
- The TouCAN ignores its Rx pins and drives its Tx pins as recessive
- The TouCAN loses synchronization with the CAN bus and the NOTRDY and FRZACK bits in CANMCR are set
- The CPU is allowed to read and write the error counter registers

After engaging one of the mechanisms to place the TouCAN in debug mode, the user must wait for the FRZACK bit to be set before accessing any other registers in the TouCAN; otherwise unpredictable operation may occur.

To exit debug mode, the IMB FREEZE line must be negated or the HALT bit in CANMCR must be cleared.

Once debug mode is exited, the TouCAN resynchronizes with the CAN bus by waiting for 11 consecutive recessive bits before beginning to participate in CAN bus communication.

16.5.2 Low-Power Stop Mode

Before entering low-power stop mode, the TouCAN waits for the CAN bus to be in an idle state, or for the third bit of intermission to be recessive. The TouCAN then waits for the completion of all internal activity (except in the CAN bus interface) to be complete. Then the following events occur:

- The TouCAN shuts down its clocks, stopping most internal circuits, thus achieving maximum power savings
- The bus interface unit continues to operate, allowing the CPU to access the module configuration register
- The TouCAN ignores its Rx pins and drives its Tx pins as recessive
- The TouCAN loses synchronization with the CAN bus, and the STOPACK and NOTRDY bits in the module configuration register are set

To exit low-power stop mode:

- Reset the TouCAN either by asserting one of the IMB3 reset lines or by asserting the SOFTRST bit CANMCR.
- Clear the STOP bit in CANMCR.
- The TouCAN module can optionally exit low-power stop mode via the self wake mechanism. If the SELFWAKE bit in CANMCR was set at the time the TouCAN entered stop mode, then upon detection of a recessive to dominant transition on the CAN bus, the TouCAN clears the STOP bit in CANMCR and its clocks begin running.

When the TouCAN is in low-power stop mode, a recessive to dominant transition on the CAN bus causes the WAKEINT bit in the error and status register (ESTAT) to be set. This event generates an interrupt if the WAKEMSK bit in CANMCR is set.

Consider the following notes regarding low-power stop mode:

- When the self wake mechanism is activated, the TouCAN tries to receive the frame that woke it up. (It assumes that the dominant bit detected is a start-of-frame bit.) It will not arbitrate for the CAN bus at this time.
- If the STOP bit is set while the TouCAN is in the bus off state, then the TouCAN enters low-power stop mode and stops counting recessive bit times. The count continues when STOP is cleared.
- To place the TouCAN in low-power stop mode with the self wake mechanism engaged, write to CANMCR with both STOP and SELFWAKE set, and then wait

for the TouCAN to set the STOPACK bit.

- To take the TouCAN out of low-power stop mode when the self wake mechanism is enabled, write to CANMCR with both STOP and SELFWAKE clear, and then wait for the TouCAN to clear the STOPACK bit.
- The SELFWAKE bit should not be set after the TouCAN has already entered low-power stop mode.
- If both STOP and SELFWAKE are set and a recessive to dominant edge immediately occurs on the CAN bus, the TouCAN may never set the STOPACK bit, and the STOP bit will be cleared.
- To prevent old frames from being sent when the TouCAN awakes from low-power stop mode via the self wake mechanism, disable all transmit sources, including transmit buffers configured for remote request responses, before placing the TouCAN in low-power stop mode.
- If the TouCAN is in debug mode when the STOP bit is set, the TouCAN assumes that debug mode should be exited. As a result, it tries to synchronize with the CAN bus, and only then does it await the conditions required for entry into low-power stop mode.
- Unlike other modules, the TouCAN does not come out of reset in low-power stop mode. The basic TouCAN initialization procedure should be executed before placing the module in low-power stop mode. (Refer to **16.4.2 TouCAN Initialization**.)
- If the TouCAN is in low-power stop mode with the self wake mechanism engaged and is operating with a single IMB clock per time quantum, there can be extreme cases in which TouCAN wake-up on recessive to dominant edge may not conform to the CAN protocol. TouCAN synchronization is shifted one time quantum from the wake-up event. This shift lasts until the next recessive-to-dominant edge, which resynchronizes the TouCAN to be in conformance with the CAN protocol. The same holds true when the TouCAN is in auto power save mode and awakens on a recessive to dominant edge.

16.5.3 Auto Power Save Mode

Auto power save mode enables normal operation with optimized power savings. Once the auto power save (APS) bit in CANMCR is set, the TouCAN looks for a set of conditions in which there is no need for the clocks to be running. If these conditions are met, the TouCAN stops its clocks, thus saving power. The following conditions activate auto power save mode.

- No Rx/Tx frame in progress
- No transfer of Rx/Tx frames to and from a serial message buffer, and no Tx frame awaiting transmission in any message buffer
- No CPU access to the TouCAN module
- The TouCAN is not in debug mode, low-power stop mode, or the bus off state

While its clocks are stopped, if the TouCAN senses that any one of the aforementioned conditions is no longer true, it restarts its clocks. The TouCAN then continues to monitor these conditions and stops or restarts its clocks accordingly.

16.6 Interrupts

The TouCAN can generate one interrupt level on the IMB. This level is programmed into the priority level bits in the interrupt configuration register (CANICR). This value determines which interrupt signal is driven onto the bus when an interrupt is requested.

Each one of the 16 message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between transmit and receive interrupts for a particular buffer. Each of the buffers is assigned a bit in the IFLAG register. An IFLAG bit is set when the corresponding buffer completes a successful transmission/reception. An IFLAG bit is cleared when the CPU reads IFLAG while the associated bit is set, and then writes it back as zero (and no new event of the same type occurs between the read and the write actions).

The other three interrupt sources (bus off, error and wake up) act in the same way, and have flag bits located in the error and status register (ESTAT). The bus off and error interrupt mask bits (BOFFMSK and ERRMSK) are located in CANCTRL0, and the wake up interrupt mask bit (WAKEMSK) is located in the module configuration register. Refer to **16.7 Programmer's Model** for more information on these registers.

The TouCAN module is capable of generating one of the 32 possible interrupt levels on the IMB3. The 32 interrupt levels are time multiplexed on the IMB3 $\overline{\text{IRQ}}$[0:7] lines. All interrupt sources place their asserted level on a time multiplexed bus during four different time slots, with eight levels communicated per slot. The ILBS[0:1] signals indicate which group of eight are being driven on the interrupt request lines.

Table 16-9 Interrupt Levels

ILBS[0:1]	Levels
00	0:7
01	8:15
10	16:23
11	24:31

The level that the TouCAN will drive onto $\overline{\text{IRQ}}$[7:0] is programmed in the three interrupt request level (IRL) bits located in the interrupt configuration register. The two ILBS bits in the ICR register determine on which slot the TouCAN should drive its interrupt signal. Under the control of ILBS, each interrupt request level is driven during the time multiplexed bus during one of four different time slots, with eight levels communicated per time slot. No hardware priority is assigned to interrupts. Furthermore, if more than one source on a module requests an interrupt at the same level, the system software must assign a priority to each source requesting at that level. **Figure 16-5** displays the interrupt levels on IRQ with ILBS.

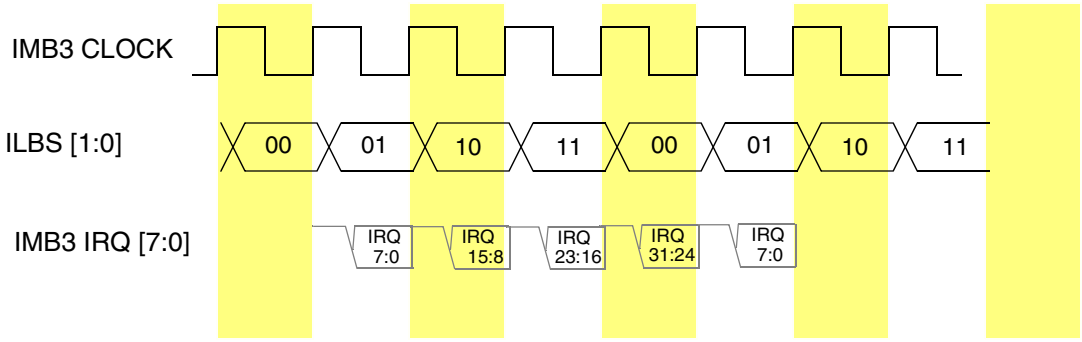


Figure 16-5 Interrupt levels on IRQ with ILBS

16.7 Programmer's Model

Table 16-10 shows the TouCAN address map. The lowercase "x" appended to each register name represents "A" or "B" for the TouCAN_A or TouCAN_B module, respectively. Refer to **1.3 MPC555 / MPC556 Address Map** to locate each TouCAN module in the MPC555 / MPC556 address map.

The column labeled "Access" indicates the privilege level at which the CPU must be operating to access the register. A designation of "S" indicates that supervisor mode is required. A designation of "S/U" indicates that the register can be programmed for either supervisor mode access or unrestricted access.

The address space for each TouCAN module is split, with 128 bytes starting at the base address, and an extra 256 bytes starting at the base address +128. The upper 256 are fully used for the message buffer structures. Of the lower 128 bytes, some are not used. Registers with bits marked as "reserved" should always be written as logic 0.

Typically, the TouCAN control registers are programmed during system initialization, before the TouCAN becomes synchronized with the CAN bus. The configuration registers can be changed after synchronization by halting the TouCAN module. This is done by setting the HALT bit in the TouCAN module configuration register (CANMCR). The TouCAN responds by asserting the CANMCR NOTRDY bit. Additionally, the control registers can be modified while the MCU is in background debug mode.

NOTE

The TouCAN has no hard-wired protection against invalid bit/field programming within its registers. Specifically, no protection is provided if the programming does not meet CAN protocol requirements.

Table 16-10 TouCAN Register Map

Access	Offset	MSB 0	LSB 15
S	0x30 7080, 0x30 7480	TouCAN Module Configuration Register (TCNMCR_x) See **Table 16-11** for bit descriptions.	
S	0x30 7082, 0x30 7482	TouCAN Test Register (CANTCR_x)	
S	0x30 7084, 0x30 7484	TouCAN Interrupt Register (CANICR_x)	
S/U	0x30 7086, 0x30 7486	Control Register 0 (CANCTRL0_x) See **Table 16-13** and **Table 16-16** for bit descriptions.	Control Register 1 (CANCTRL1_x)
S/U	0x30 7088, 0x30 7488	Control and Prescaler Divider Register (PRESDIV_x) See **Table 16-17** and **Table 16-18** for bit descriptions.	Control Register 2 (CTRL2_x)
S/U	0x30 708A, 0x30 748A	Free-Running Timer Register (TIMER_x) See **Table 16-19** for bit descriptions.	
—	0x30 708C, 0x30 748C — 0x30 708E, 0x30 748E	Reserved	
S/U	0x30 7090, 0x30 7490	Receive Global Mask – High (RXGMSKHI_x) See **Table 16-20** for bit descriptions.	
S/U	0x30 7092, 0x30 7492	Receive Global Mask – Low (RXGMSKLO_x) See **Table 16-20** for bit descriptions.	
S/U	0x30 7094, 0x30 7494	Receive Buffer 14 Mask – High (RX14MSKHI_x) See **16.7.10 Receive Buffer 14 Mask Registers** for bit descriptions.	
S/U	0x30 7096, 0x30 7496	Receive Buffer 14 Mask – Low (RX14MSKLO_x) See **16.7.10 Receive Buffer 14 Mask Registers** for bit descriptions.	
S/U	0x30 7098, 0x30 7498	Receive Buffer 15 FMask – High (RX15MSKHI_x) See **16.7.11 Receive Buffer 15 Mask Registers** for bit descriptions.	
S/U	0x30 709A, 0x30 749A	Receive Buffer 15 Mask – Low (RX15MSKLO_x) See **16.7.11 Receive Buffer 15 Mask Registers** for bit descriptions.	
—	0x30 709C, 0x30 749C — 0x30 709E, 0x30 749E	Reserved	
S/U	0x30 70A0, 0x30 74A0	Error and Status Register (ESTAT_x) See **Table 16-21** for bit descriptions.	
S/U	0x30 70A2, 0x30 74A2	Interrupt Masks (IMASK_x) See **Table 16-24** for bit descriptions.	
S/U	0x30 70A4, 0x30 74A4	Interrupt Flags (IFLAG_x) See **Table 16-25** for bit descriptions.	
S/U	0x30 70A6, 0x30 74A6	Receive Error Counter (RXECTR_x) See **Table 16-26** for bit descriptions.	Transmit Error Counter (TXECTR_x)

Table 16-10 TouCAN Register Map (Continued)

Access	Offset	MSB 0	LSB 15
S/U	0x30 7100 — 0x30 710F(A) 0x30 7500 — 0x30 750F(B)	MBUFF0[1] TouCAN_A Message Buffer 0. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7110 — 0x30 711F(A) 0x30 7510 — 0x30 751F(B)	MBUFF1[1] TouCAN_A Message Buffer 1. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7120 — 0x30 712F(A) 0x30 7520 — 0x30 752F(B)	MBUFF2[1] TouCAN_A Message Buffer 2. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7130 — 0x30 713F(A) 0x30 7530 — 0x30 753F(B)	MBUFF3[1] TouCAN_A Message Buffer 3. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7140 — 0x30 714F(A) 0x30 7540 — 0x30 754F(B)	MBUFF4[1] TouCAN_A Message Buffer 4. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7150 — 0x30 715F(A) 0x30 7550 — 0x30 755F(B)	MBUFF5[1] TouCAN_A Message Buffer 5. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7160 — 0x30 716F(A) 0x30 7560 — 0x30 756F(B)	MBUFF6[1] TouCAN_A Message Buffer 6. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x307170 — 0x30717F(A) 0x30 7570 — 0x30 757F(B)	MBUFF7[1] TouCAN_A Message Buffer 7. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7180 — 0x30 718F(A) 0x30 7580 — 0x30 758F(B)	MBUFF8[1] TouCAN_A Message Buffer 8. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 7190 — 0x30 719F(A) 0x30 7590 — 0x30 759F(B)	MBUFF9[1] TouCAN_A Message Buffer 9. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 71A0 — 0x30 71AF(A) 0x30 75A0 — 0x30 75AF(B)	MBUFF10[1] TouCAN_A Message Buffer 10. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 71B0 — 0x30 71BF(A) 0x30 75B0 — 0x30 75BF(B)	MBUFF11[1] TouCAN_A Message Buffer 11. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 71C0 — 0x30 71CF(A) 0x30 75C0 — 0x30 75CF(B)	MBUFF12[1] TouCAN_A Message Buffer 12. See **Table 16-3** and **Table 16-4** for message buffer definitions.	
S/U	0x30 71D0 — 0x30 71DF(A) 0x30 75D0 — 0x30 75DF(B)	MBUFF13[1] TouCAN_A Message Buffer 13. See **Table 16-3** and **Table 16-4** for message buffer definitions.	

Table 16-10 TouCAN Register Map (Continued)

Access	Offset	MSB 0	LSB 15
S/U	0x30 71E0 — 0x30 71EF(A) 0x30 75E0 — 0x30 75EF(B)	MBUFF14[1] TouCAN_A Message Buffer 14. See Table 16-3 and Table 16-4 for message buffer definitions.	
S/U	0x30 71F0 — 0x30 71FF(A) 0x30 75F0 — 0x30 75FF(B)	MBUFF15[1] TouCAN_A Message Buffer 15. See Table 16-3 and Table 16-4 for message buffer definitions.	

NOTES:
1. The last word of each of the the MBUFF arrays (address 0x....E) is reserved and may cause a RCPU exception if read.

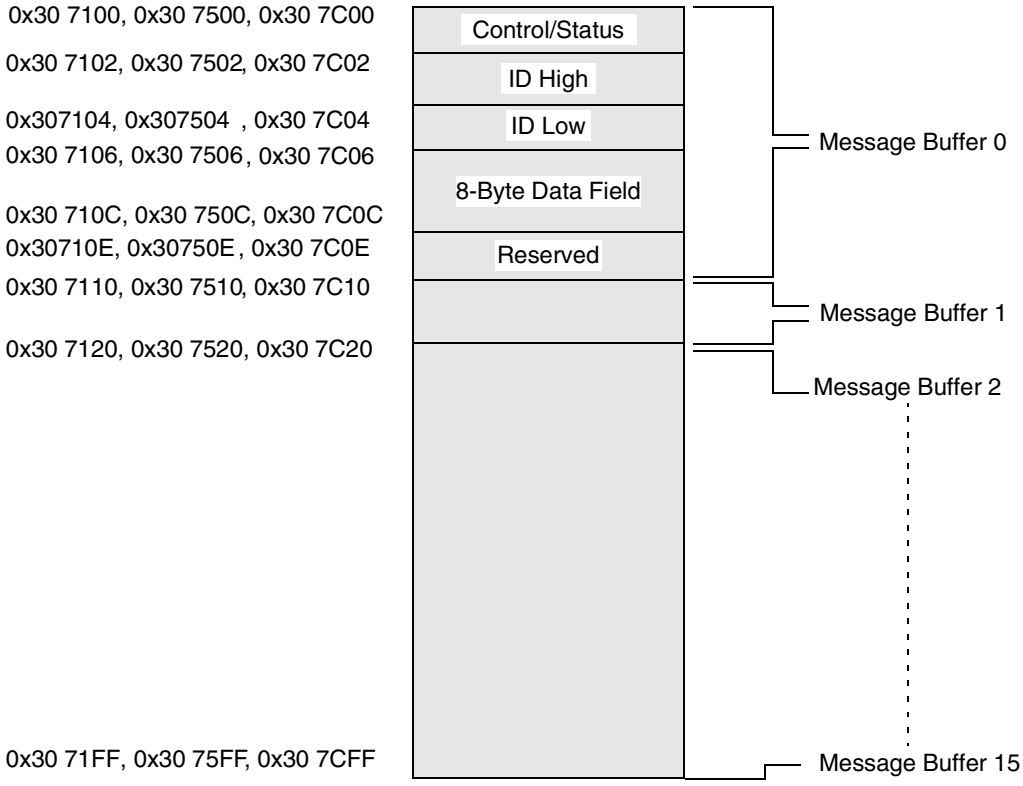


Figure 16-6 TouCAN Message Buffer Memory Map

16.7.1 TouCAN Module Configuration Register

TCNMCR — TouCAN Module Configuration Register **0x30 7080**
0x30 7480

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
STOP	FRZ	NOT USED	HALT	NOT RDY	WAKE MSK	SOFT RST	FRZ ACK	SUPV	SELF WAKE	APS	STOP ACK	RESERVED			
RESET:															
0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0

Table 16-11 TCNMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Low-power stop mode enable. The STOP bit may only be set by the CPU. It may be cleared either by the CPU or by the TouCAN, if the SELFWAKE bit is set. 0 = Enable TouCAN clocks 1 = Disable TouCAN clocks
1	FRZ	FREEZE assertion response. When FRZ = 1, the TouCAN can enter debug mode when the IMB3 FREEZE line is asserted or the HALT bit is set. Clearing this bit field causes the TouCAN to exit debug mode. Refer to **16.5.1 Debug Mode** for more information. 0 = TouCAN ignores the IMB3 FREEZE signal and the HALT bit in the module configuration register. 1 = TouCAN module enabled to enter debug mode.
2	—	Reserved
3	HALT	Halt TouCAN S-Clock. Setting the HALT bit has the same effect as assertion of the IMB3 FREEZE signal on the TouCAN without requiring that FREEZE be asserted. This bit is set to one after reset. It should be cleared after initializing the message buffers and control registers. TouCAN message buffer receive and transmit functions are inactive until this bit is cleared. When HALT is set, write access to certain registers and bits that are normally read-only is allowed. 0 = The TouCAN operates normally 1 = TouCAN enters debug mode if FRZ = 1
4	NOTRDY	TouCAN not ready. This bit indicates that the TouCAN is either in low-power stop mode or debug mode. This bit is read-only and is set only when the TouCAN enters low-power stop mode or debug mode. It is cleared once the TouCAN exits either mode, either by synchronization to the CAN bus or by the self wake mechanism. 0 = TouCAN has exited low-power stop mode or debug mode. 1 = TouCAN is in low-power stop mode or debug mode.
5	WAKEMSK	Wakeup interrupt mask. The WAKEMSK bit enables wake-up interrupt requests. 0 = Wake up interrupt is disabled 1 = Wake up interrupt is enabled

Table 16-11 TCNMCR Bit Descriptions (Continued)

Bit(s)	Name	Description
6	SOFTRST	Soft reset. When this bit is asserted, the TouCAN resets its internal state machines (sequencer, error counters, error flags, and timer) and the host interface registers (CANMCR, CANICR, CANTCR, IMASK, and IFLAG). The configuration registers that control the interface with the CAN bus are not changed (CANCTRL[0:2] and PRESDIV). Message buffers and receive message masks are also not changed. This allows SOFTRST to be used as a debug feature while the system is running. Setting SOFTRST also clears the STOP bit in CANMCR. After setting SOFTRST, allow one complete bus cycle to elapse for the internal TouCAN circuitry to completely reset before executing another access to CANMCR. The TouCAN clears this bit once the internal reset cycle is completed. 0 = Soft reset cycle completed 1 = Soft reset cycle initiated
7	FRZACK	TouCAN disable. When the TouCAN enters debug mode, it sets the FRZACK bit. This bit should be polled to determine if the TouCAN has entered debug mode. When debug mode is exited, this bit is negated once the TouCAN prescaler is enabled. This is a read-only bit. 0 = The TouCAN has exited debug mode and the prescaler is enabled 1 = The TouCAN has entered debug mode, and the prescaler is disabled
8	SUPV	Supervisor/user data space. The SUPV bit places the TouCAN registers in either supervisor or user data space. 0 = Registers with access controlled by the SUPV bit are accessible in either user or supervisor privilege mode 1 = Registers with access controlled by the SUPV bit are restricted to supervisor mode
9	SELFWAKE	Self wake enable. This bit allows the TouCAN to wake up when bus activity is detected after the STOP bit is set. If this bit is set when the TouCAN enters low-power stop mode, the TouCAN will monitor the bus for a recessive to dominant transition. If a recessive to dominant transition is detected, the TouCAN immediately clears the STOP bit and restarts its clocks. If a write to CANMCR with SELFWAKE set occurs at the same time a recessive-to-dominant edge appears on the CAN bus, the bit will not be set, and the module clocks will not stop. The user should verify that this bit has been set by reading CANMCR. Refer to **16.5.2 Low-Power Stop Mode** for more information on entry into and exit from low-power stop mode. 0 = Self wake disabled 1 = Self wake enabled
10	APS	Auto power save. The APS bit allows the TouCAN to automatically shut off its clocks to save power when it has no process to execute, and to automatically restart these clocks when it has a task to execute without any CPU intervention. 0 = Auto power save mode disabled; clocks run normally 1 = Auto power save mode enabled; clocks stop and restart as needed
11	STOPACK	Stop acknowledge. When the TouCAN is placed in low-power stop mode and shuts down its clocks, it sets the STOPACK bit. This bit should be polled to determine if the TouCAN has entered low-power stop mode. When the TouCAN exits low-power stop mode, the STOPACK bit is cleared once the TouCAN's clocks are running. 0 = The TouCAN is not in low-power stop mode and its clocks are running 1 = The TouCAN has entered low-power stop mode and its clocks are stopped
12:15	—	Reserved

16.7.2 TouCAN Test Configuration Register

CANTCR — TouCAN Test Configuration Register **0x30 7082, 0x30 7482**

This register is used for factory test only.

16.7.3 TouCAN Interrupt Configuration Register

CANICR — TouCAN Interrupt Configuration Register **0x30 7084**
0x30 7484

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED					IRL			ILBS		RESERVED					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Table 16-12 CANICR Bit Descriptions

Bit(s)	Name	Description
0:4	—	Reserved
5:7	IRL	Interrupt request level. When the TouCAN generates an interrupt request, this field determines which of the interrupt request signals is asserted.
8:9	ILBS	Interrupt level byte select. This field selects one of four time-multiplexed slots during which the interrupt request is asserted. The ILBS and IRL fields together select one of 32 effective interrupt levels. 00 = Levels 0 to7 01 = Levels 8 to 15 10 = Levels 16 to 23 11 = Levels 24 to 31
10:15	—	Reserved

16.7.4 Control Register 0

CANCTRL0 — Control Register 0 **0x30 7086**
0x30 7486

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
BOFF MSK	ERR MSK	RESERVED		RXMOD		TXMODE		CANCTRL1							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-13 CANCTRL0 Bit Descriptions

Bit(s)	Name	Description
0	BOFFMSK	Bus off interrupt mask. The BOFF MASK bit provides a mask for the bus off interrupt. 0 = Bus off interrupt disabled 1 = Bus off interrupt enabled
1	ERRMSK	Error interrupt mask. The ERRMSK bit provides a mask for the error interrupt. 0 = Error interrupt disabled 1 = Error interrupt enabled
2:3	—	
4:5	RXMODE	Receive pin configuration control. These bits control the configuration of the CANRX0 and CANRX1 pins. Refer to the **Table 16-14**.
6:7	TXMODE	Transmit pin configuration control. This bit field controls the configuration of the CANTX0 and CANTX1 pins. Refer to **Table 16-15**.
8:15	CANCTRL1	See **Table 16-16**.

Table 16-14 RX MODE[1:0] Configuration

Pin	RX1	RX0	Receive Pin Configuration
CNRX1[1]	0	X	A logic zero on the CNRX1 pin is interpreted as a dominant bit; a logic one on the CNRX1 pin is interpreted as a recessive bit
	1	X	A logic one on the CNRX1 pin is interpreted as a dominant bit; a logic zero on the CNRX1 pin is interpreted as a recessive bit
CNRX0	X	0	A logic zero on the CNRX0 pin is interpreted as a dominant bit; a logic one on the CNRX0 pin is interpreted as a recessive bit
	X	1	A logic one on the CNRX0 pin is interpreted as a dominant bit; a logic zeroon the CNRX0 pin is interpreted as a recessive bit

NOTES:
1. The CNRX1 signal is not available on the MPC555 / MPC556.

Table 16-15 Transmit Pin Configuration

TXMODE[1:0]	Transmit Pin Configuration
00	Full CMOS[1]; positive polarity (CNTX0 = 0, CNTX1[2] = 1 is a dominant level)[2]
01	Full CMOS; negative polarity (CNTX0 = 1, CNTX1[2] = 0 is a dominant level)
1X	Open drain[3]; positive polarity

NOTES:
1. Full CMOS drive indicates that both dominant and recessive levels are driven by the chip.
2. The CNTX1 signal is not available on the MPC555 / MPC556.
3. Open drain drive indicates that only a dominant level is driven by the chip. During a recessive level, the CNTX0 and CNTX1 pins are disabled (three stated), and the electrical level is achieved by external pull-up/pull-down devices. The assertion of both Tx mode bits causes the polarity inversion to be cancelled (open drain mode forces the polarity to be positive).

16.7.5 Control Register 1

CANCTRL1 — Control Register 1 **0x30 7086**
0x30 7486

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CANCTRL0								SAMP	Re-served	TSYNC	LBUF	0D	PROPSE		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-16 CANCTRL1 Bit Descriptions

Bit(s)	Name	Description
0:7	CANCTRL0	See Table 16-13
8	SAMP	Sampling mode. The SAMP bit determines whether the TouCAN module will sample each received bit one time or three times to determine its value. 0 = One sample, taken at the end of phase buffer segment one, is used to determine the value of the received bit. 1 = Three samples are used to determine the value of the received bit. The samples are taken at the normal sample point and at the two preceding periods of the S-clock.
9	—	Reserved
10	TSYNC	Timer synchronize mode. The TSYNC bit enables the mechanism that resets the free-running timer each time a message is received in message buffer zero. This feature provides the means to synchronize multiple TouCAN stations with a special “SYNC” message (global network time). 0 = Timer synchronization disabled. 1 = Timer synchronization enabled. Note: there can be a bit clock skew of four to five counts between different TouCAN modules that are using this feature on the same network.
11	LBUF	Lowest buffer transmitted first. The LBUF bit defines the transmit-first scheme. 0 = Message buffer with lowest ID is transmitted first. 1 = Lowest numbered buffer is transmitted first.
12	—	Reserved
13:15	PROPSEG	Propagation segment time. PROPSEG defines the length of the propagation segment in the bit time. The valid programmed values are zero to seven. The propagation segment time is calculated as follows: Propagation Segment Time = (PROPSEG + 1) Time Quanta where 1 Time Quantum = 1 Serial Clock (S-Clock) Period

16.7.6 Prescaler Divide Register

PRESDIV — Prescaler Divide Register **0x30 7088**
0x30 7488

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PRESDIV								CANCTRL2							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-17 PRESDIV Bit Descriptions

Bit(s)	Name	Description
0:7	PRESDIV	Prescaler divide factor. PRESDIV determines the ratio between the IMB clock frequency and the serial clock (S-clock). The S-clock is determined by the following calculation: $\text{S-clock} = \frac{f_{SYS}}{\text{PRESDIV} + 1}$ The reset value of PRESDIV is 0x00, which forces the S-clock to default to the same frequency as the IMB clock. The valid programmed values are 0 through 255.
8:15	CANCTRL2	See **Table 16-18**.

16.7.7 Control Register 2

CANCTRL2 — Control Register 2 **0x30 7088**
0x30 7488

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PRESDIV								RJW		PSEG			PSEG2		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-18 CANCTRL2 Bit Descriptions

Bit(s)	Name	Description
0:7	PRESDIV	See **Table 16-17**.
8:9	RJW	Resynchronization jump width. The RJW field defines the maximum number of time quanta a bit time may be changed during resynchronization. The valid programmed values are zero through three. The resynchronization jump width is calculated as follows: Resynchronizaton Jump Width = (RJW + 1) Time Quanta
10:12	PSEG1	PSEG1[2:0] — Phase buffer segment 1. The PSEG1 field defines the length of phase buffer segment one in the bit time. The valid programmed values are zero through seven. The length of phase buffer segment 1 is calculated as follows: Phase Buffer Segment 1 = (PSEG1 + 1) Time Quanta
13:15	PSEG2	PSEG2 — Phase Buffer Segment 2. The PSEG2 field defines the length of phase buffer segment two in the bit time. The valid programmed values are zero through seven. The length of phase buffer segment two is calculated as follows: Phase Buffer Segment 2 = (PSEG2 + 1) Time Quanta

16.7.8 Free Running Timer

TIMER — Free Running Timer Register **0x30 708A**
0x30 748A

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
TIMER															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-19 TIMER Bit Descriptions

Bit(s)	Name	Description
0:15	TIMER	The free running timer counter can be read and written by the CPU. The timer starts from zero after reset, counts linearly to 0xFFFF, and wraps around. The timer is clocked by the TouCAN bit-clock. During a message, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it increments at the nominal bit rate. The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. The captured value is written into the “time stamp” entry in a message buffer after a successful reception or transmission of a message.

16.7.9 Receive Global Mask Registers

RXGMSKHI — Receive Global Mask Register High **0x30 7090, 0x30 7490**
RXGMSKLO — Receive Global Mask Register Low **0x30 7092, 0x30 7492**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
RESET:															
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
RESET:															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Table 16-20 RXGMSKHI, RXGMSKLO Bit Descriptions

Bit(s)	Name	Description
0:31	MIDx	The receive global mask registers use four bytes. The mask bits are applied to all receive-identifiers, excluding receive-buffers 14 and 15, which have their own specific mask registers. Base ID mask bits MID[28:18] are used to mask standard or extended format frames. Extended ID bits MID[17:0] are used to mask only extended format frames. The RTR/SRR bit of a received frame is never compared to the corresponding bit in the message buffer ID field. However, remote request frames (RTR = 1) once received, are never stored into the message buffers. RTR mask bit locations in the mask registers (bits 20 and zero) are always zero, regardless of any write to these bits. The IDE bit of a received frame is always compared to determine if the message contains a standard or extended identifier. Its location in the mask registers (bit 19) is always one, regardless of any write to this bit.

16.7.10 Receive Buffer 14 Mask Registers

RX14MSKHI — Receive Buffer 14 Mask Register High **0x30 7094, 0x30 7494**
RX14MSKLO — Receive Buffer 14 Mask Register Low **0x30 7096, 0x30 7496**

The receive buffer 14 mask registers have the same structure as the receive global mask registers and are used to mask buffer 14.

16.7.11 Receive Buffer 15 Mask Registers

RX15MSKHI — Receive Buffer 15 Mask Register High **0x30 7098, 0x30 7498**
RX15MSKLO — Receive Buffer 15 Mask Register Low **0x30 709A, 0x30 749A**

The receive buffer 15 mask registers have the same structure as the receive global mask registers and are used to mask buffer 15.

16.7.12 Error and Status Register

ESTAT — Error and Status Register **0x30 70A0**
0x30 74A0

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
BITERR		ACK ERR	CRC ERR	FORM ERR	STUFF ERR	TX WARN	RX WARN	IDLE	TX/RX	FCS		0	BOFF INT	ERR INT	WAKE INT
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register reflects various error conditions, general status, and has the enable bits for three of the TouCAN interrupt sources. The reported error conditions are those which have occurred since the last time the register was read. A read clears these bits to zero.

Table 16-21 ESTAT Bit Descriptions

Bit(s)	Name	Description
0:1	BITERR	Transmit bit error. The BITERR[1:0] field is used to indicate when a transmit bit error occurs. Refer to **Table 16-22**. **NOTE**: The transmit bit error field is not modified during the arbitration field or the ACK slot bit time of a message, or by a transmitter that detects dominant bits while sending a passive error frame.
2	ACKERR	Acknowledge error. The ACKERR bit indicates whether an acknowledgment has been correctly received for a transmitted message. 0 = No ACK error was detected since the last read of this register 1 = An ACK error was detected since the last read of this register
3	CRCERR	Cyclic redundancy check error. The CRCERR bit indicates whether or not the CRC of the last transmitted or received message was valid. 0 = No CRC error was detected since the last read of this register 1 = A CRC error was detected since the last read of this register
4	FORMERR	Message format error. The FORMERR bit indicates whether or not the message format of the last transmitted or received message was correct. 0 = No format error was detected since the last read of this register 1 = A format error was detected since the last read of this register
5	STUFERR	Bit stuff error. The STUFFERR bit indicates whether or not the bit stuffing that occurred in the last transmitted or received message was correct. 0 = No bit stuffing error was detected since the last read of this register 1 = A bit stuffing error was detected since the last read of this register
6	TXWARN	Transmit error status flag. The TXWARN status flag reflects the status of the TouCAN transmit error counter. 0 = Transmit error counter < 96 1 = Transmit error counter ≥ 96
7	RXWARN	Receiver error status flag. The RXWARN status flag reflects the status of the TouCAN receive error counter. 0 = Receive error counter < 96 1 = Receive error counter ≥ 96
8	IDLE	Idle status. The IDLE bit indicates when there is activity on the CAN bus. 0 = The CAN bus is not idle 1 = The CAN bus is idle
9	TX/$\overline{\text{RX}}$	Transmit/receive status. The TX/RX bit indicates when the TouCAN module is transmitting or receiving a message. TX/RX has no meaning when IDLE = 1. 0 = The TouCAN is receiving a message if IDLE = 0 1 = The TouCAN is transmitting a message if IDLE = 0
10:11	FCS	Fault confinement state. The FCS[1:0] field describes the state of the TouCAN. Refer to **Table 16-23**. If the SOFTRST bit in CANMCR is asserted while the TouCAN is in the bus off state, the error and status register is reset, including FCS[1:0]. However, as soon as the TouCAN exits reset, FCS[1:0] bits will again reflect the bus off state. Refer to**16.3.4 Error Counters** for more information on entry into and exit from the various fault confinement states.
12	—	Reserved
13	BOFFINT	Bus off interrupt. The BOFFINT bit is used to request an interrupt when the TouCAN enters the bus off state. 0 = No bus off interrupt requested 1 = When the TouCAN state changes to bus off, this bit is set, and if the BOFFMSK bit in CANCTRL0 is set, an interrupt request is generated. This interrupt is not requested after reset.

Table 16-21 ESTAT Bit Descriptions (Continued)

Bit(s)	Name	Description
14	ERRINT	Error Interrupt. The ERRINT bit is used to request an interrupt when the TouCAN detects a transmit or receive error. 0 = No error interrupt request 1 = If an event which causes one of the error bits in the error and status register to be set occurs, the error interrupt bit is set. If the ERRMSK bit in CANCTRL0 is set, an interrupt request is generated. To clear this bit, first read it as a one, then write as a zero. Writing a one has no effect.
15	WAKEINT	Wake interrupt. The WAKEINT bit indicates that bus activity has been detected while the TouCAN module is in low-power stop mode. 0 = No wake interrupt requested 1 = When the TouCAN is in low-power stop mode and a recessive to dominant transition is detected on the CAN bus, this bit is set. If the WAKEMSK bit is set in CANMCR, an interrupt request is generated.

Table 16-22 Transmit Bit Error Status

BITERR[1:0]	Bit Error Status
00	No transmit bit error
01	At least one bit sent as dominant was received as recessive
10	At least one bit sent as recessive was received as dominant
11	Not used

Table 16-23 Fault Confinement State Encoding

FCS[1:0]	Bus State
00	Error active
01	Error passive
1X	Bus off

16.7.13 Interrupt Mask Register

IMASK — Interrupt Mask Register **0x30 70A2, 0x30 74A2**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
IMASKH								IMASKL							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-24 IMASK Bit Descriptions

Bit(s)	Name	Description
0:7, 8:15	IMASKH, IMASKL	IMASK contains two 8-bit fields, IMASKH and IMASKL. IMASK can be accessed with a 16-bit read or write, and IMASKH and IMASKL can be accessed with byte reads or writes. IMASK contains one interrupt mask bit per buffer. It allows the CPU to designate which buffers will generate interrupts after successful transmission/reception. Setting a bit in IMASK enables interrupt requests for the corresponding message buffer. **NOTE**: Bit 15 (LSB) corresponds to message buffer 0. Bit 0 (MSB) corresponds to mesage buffer 15.

16.7.14 Interrupt Flag Register

IFLAG — Interrupt Flag Register

0x30 70A4
0x30 74A4

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
IFLAGH								IFLAGL							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-25 IFLAG Bit Descriptions

Bit(s)	Name	Description
0:7, 8:15	IFLAGH, IFLAGL	IFLAG contains two 8-bit fields, IFLAGH and IFLAGL. IFLAG can be accessed with a 16-bit read or write, and IFLAGH and IFLAGL can be accessed with byte reads or writes. IFLAG contains one interrupt flag bit per buffer. Each successful transmission/reception sets the corresponding IFLAG bit and, if the corresponding IMASK bit is set, an interrupt request will be generated. To clear an interrupt flag, first read the flag as a one, and then write it as a zero. Should a new flag setting event occur between the time that the CPU reads the flag as a one and writes the flag as a zero, the flag is not cleared. This register can be written to zeros only. **NOTE**: Bit 15 (LSB) corresponds to message buffer 0. Bit 0 (MSB) corresponds to mesage buffer 15.

16.7.15 Error Counters

RXECTR — Receive Error Counter
TXECTR — Transmit Error Counter

0x30 70A6, 0x30 74A6

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RXECTR								TXECTR							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-26 RXECTR, TXECTR Bit Descriptions

Bit(s)	Name	Description
0:7, 8:15	RXECTR, TXECTR	Both counters are read only, except when the TouCAN is in test or debug mode.