BCANPSV2.0/D
Rev. 3

BCANPSV2.0

CAN

Bosch Controller Area Network (CAN) Version 2.0

PROTOCOL
STANDARD

PROTOCOL STANDARD

MOTOROLA

MOTOROLA

Bosch Controller Area Network
Version 2.0
Protocol Standard

Conventions

Where abbreviations are used in the text, an explanation can be found in the glossary, at the back of this manual. Register and bit mnemonics are defined in the paragraphs describing them.

An overbar is used to designate an active-low signal, eg: $\overline{\text{RESET}}$.

Unless otherwise stated, shaded cells in a register diagram indicate that the bit is either unused or reserved; ‘u’ is used to indicate an undefined state (on reset).

CUSTOMER FEEDBACK QUESTIONNAIRE (CAN PROTOCOL)

Motorola wishes to continue to improve the quality of its documentation. We would welcome your feedback on the publication you have just received. Having used the document, please complete this card (or a photocopy of it, if you prefer).

1. How would you rate the quality of the document? Check one box in each category.

	Excellent			Poor		Excellent			Poor
Organization	❑	❑	❑	❑	Tables	❑	❑	❑	❑
Readability	❑	❑	❑	❑	Table of contents	❑	❑	❑	❑
Understandability	❑	❑	❑	❑	Index	❑	❑	❑	❑
Accuracy	❑	❑	❑	❑	Page size/binding	❑	❑	❑	❑
Illustrations	❑	❑	❑	❑	Overall impression	❑	❑	❑	❑

Comments: _____

2. What is your intended use for this document? If more than one option applies, please rank them (1, 2, 3).

Selection of device for new application ❑
System design ❑
Training purposes ❑
Other ❑ Please specify: _____

3. How well does this manual enable you to perform the task(s) outlined in question 2?

Completely			Not at all
❑	❑	❑	❑

Comments: _____

4. How easy is it to find the information you are looking for?

Easy			Difficult
❑	❑	❑	❑

Comments: _____

5. Is the level of technical detail in the following sections sufficient to allow you to understand how the device functions?

		Too little detail				Too much detail
SECTION 1	INTRODUCTION	❑	❑	❑	❑	❑
SECTION 2	BASIC CONCEPTS	❑	❑	❑	❑	❑
SECTION 3	MESSAGE TRANSFER	❑	❑	❑	❑	❑
SECTION 4	ERROR HANDLING	❑	❑	❑	❑	❑
SECTION 5	FAULT CONFINEMENT	❑	❑	❑	❑	❑
SECTION 6	BIT TIMING REQUIREMENTS	❑	❑	❑	❑	❑
SECTION 7	INCREASING OSCILLATOR TOLERANCE	❑	❑	❑	❑	❑
SECTION 8	THE PHYSICAL LAYER	❑	❑	❑	❑	❑
SECTION 9	INTRODUCTION	❑	❑	❑	❑	❑
SECTION 10	BASIC CONCEPTS	❑	❑	❑	❑	❑
SECTION 11	MESSAGE TRANSFER	❑	❑	❑	❑	❑
SECTION 12	ERROR HANDLING	❑	❑	❑	❑	❑
SECTION 13	FAULT CONFINEMENT	❑	❑	❑	❑	❑
SECTION A	THE MOTOROLA CAN (MCAN) MODULE	❑	❑	❑	❑	❑
SECTION B	TOUCAN	❑	❑	❑	❑	❑
SECTION C	THE MOTOROLA SCALEABLE CAN (MSCAN08) MODULE	❑	❑	❑	❑	❑
SECTION D	THE MOTOROLA SCALEABLE CAN (MSCAN12) MODULE	❑	❑	❑	❑	❑

6. Have you found any errors? If so, please comment: _____

7. From your point of view, is anything missing from the document? If so, please say what: _____

– Cut along this line to remove –

8. How could we improve this document? _____

_____

_____

9. How would you rate Motorola's documentation?

	Excellent			Poor
– In general	❑	❑	❑	❑
– Against other semiconductor suppliers	❑	❑	❑	❑

10. Which semiconductor manufacturer provides the best technical documentation? _____
11. Which company (in any field) provides the best technical documentation? _____
12. How many years have you worked with microprocessors?

Less than 1 year ❑ 1–3 years ❑ 3–5 years ❑ More than 5 years ❑

– Second fold back along this line –

By air mail
Par avion

IBRS NUMBER PHQ-B/207/G
CCRI NUMERO PHQ-B/207/G

NE PAS AFFRANCHIR
NO STAMP REQUIRED

REPONSE PAYEE
GRANDE-BRETAGNE

Motorola Ltd.,
Colvilles Road,
Kelvin Industrial Estate,
EAST KILBRIDE,
G75 8BR.
GREAT BRITAIN.

F.A.O. Technical Publications Manager
(re: BCANPSV2.0/D)

MOTOROLA
Semiconductor Products Sector

– First fold back along this line –

– Cut along this line to remove –

– Third fold back along this line –

13. Currently there is some discussion in the semiconductor industry regarding a move towards providing data sheets in electronic form. If you have any opinion on this subject, please comment. _____

_____

14. We would be grateful if you would supply the following information (at your discretion), or attach your card.

Name: _____ Phone No: _____

Position: _____ FAX No: _____

Department: _____

Company: _____

Address: _____

_____

Thank you for helping us improve our documentation,
Technical Publications Manager, Motorola Ltd., Scotland.

– Finally, tuck this edge into opposite flap –

TABLE OF CONTENTS

Paragraph Number | TITLE | Page Number

3
MESSAGE TRANSFER

4
ERROR HANDLING

5
FAULT CONFINEMENT

6
BIT TIMING REQUIREMENTS

7
INCREASING OSCILLATOR TOLERANCE

PART B

8
INTRODUCTION

9
BASIC CONCEPTS

10
MESSAGE TRANSFER

13
BIT TIMING REQUIREMENTS

A
THE MOTOROLA CAN (MCAN) MODULE

B
TOUCAN

C
THE MOTOROLA SCALEABLE CAN (MSCAN08) MODULE

D
THE MOTOROLA SCALEABLE CAN (MSCAN12) MODULE

LIST OF FIGURES

LIST OF TABLES

PREFACE

The Controller Area Network (CAN) Specification 2.0, as defined by BOSCH Gmbh, consists of two parts, A and B, as follows:

- Part A describes the CAN message format as defined in CAN Specification 1.2
- Part B describes both standard and extended message formats

This publication covers both Part A and Part B of CAN Specification 2.0.

The standard format, as originally defined, provided 11 identifier bits. The extended format, provides a larger address range defined by 29 bits. Users of CAN who do not need the extended format can continue to use the original 11 bit identifier range. CAN implementations that are designed according to Part A of this specification, or according to previous CAN specifications, may communicate with CAN implementations that are designed according to Part B of this specification so long as the Extended Format is not used.

THIS PAGE LEFT BLANK INTENTIONALLY

BOSCH
CONTROLLER AREA NETWORK (CAN)
VERSION 2.0

PART A

THIS PAGE LEFT BLANK INTENTIONALLY

1
INTRODUCTION

The Controller Area Network (CAN) is a serial communications protocol that efficiently supports distributed real-time control with a very high level of data integrity.

Though conceived and defined by BOSCH in Germany for automotive applications, CAN is not restricted to that industry. CAN fulfils the communication needs of a wide range of applications, from high-speed networks to low-cost multiplex wiring.

For example, in automotive electronics, engine control units, sensors and anti-skid systems may be connected using CAN, with bit-rates up to 1 Mbit/s. At the same time, it is cost effective to build CAN into vehicle body electronics, such as lamp clusters and electric windows, to replace the wiring harness otherwise required.

The intention of the CAN specification is to achieve compatibility between any two CAN implementations. Compatibility, however, has different aspects with respect to, for example, electrical features and the interpretation of data to be transferred.

To achieve design transparency and implementation flexibility CAN has been subdivided into three layers:

- The Object Layer
- The Transfer Layer
- The Physical Layer

The Object Layer and the Transfer Layer comprise all services and functions of the data link layer defined by the ISO/OSI model.

The scope of the Object Layer includes: determining which messages are to be transmitted; deciding which messages received by the Transfer Layer are actually to be used; and, providing an interface to the Application Layer related hardware. There is considerable freedom in defining object handling.

The Transfer Layer is principally concerned with the transfer protocol, i.e. controlling the framing, performing arbitration, error checking, error signalling and fault confinement. Within the Transfer Layer it is decided whether the bus is free for starting a new transmission, or whether reception of a message is just starting. Also, some general features of the bit-timing are regarded as part of the Transfer Layer. Modifications to the Transfer Layer cannot be made.

The Physical Layer covers the actual transfer of the bits between the different nodes, with respect to all electrical properties. Within a network the physical layer has to be the same for all nodes. However, there are many possible implementations of the Physical Layer.

The remainder of this document is principally concerned with the definition of the Transfer Layer, and the consequences of the CAN protocol for the surrounding layers.

2
BASIC CONCEPTS

2.1 Layered structure of a CAN node

The Object Layer is concerned with message filtering as well as status and message handling.

The Transfer Layer represents the kernel of the CAN protocol. It presents messages received to the Object Layer and accepts messages to be transmitted by the Object Layer. The Transfer Layer is responsible for bit timing and synchronization, message framing, arbitration, acknowledgement, error detection and signalling, and fault confinement.

The Physical Layer defines how signals are actually transmitted. The Physical Layer is not defined here, as it will vary according to the requirements of individual applications (for example, transmission medium and signal level implementations).

2.2 Messages

Information on the bus is sent in fixed format messages of different but limited length (see Section 3). When the bus is free, any connected node may start to transmit a new message.

2.2.1 Information routing

In CAN systems a node does not make use of any information about the system configuration (e.g. node addresses). This has several important consequences, which are described below.

2.2.1.1 System flexibility

Nodes may be added to the CAN network without requiring any change in the software or hardware of any node or the application layer.

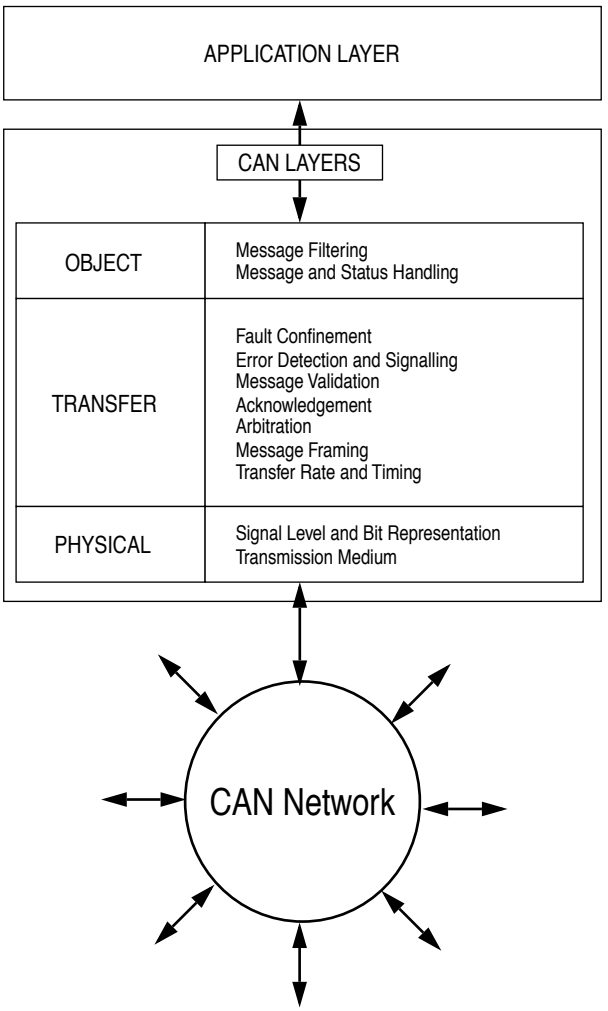


Figure 2-1 CAN layers

2.2.1.2 Message routing

The content of a message is described by an Identifier. The Identifier does not indicate the destination of the message, but describes the meaning of the data, so that all nodes in the network are able to decide by Message filtering whether the data is to be acted upon by them or not.

2.2.1.3 Multicast

As a consequence of the concept of Message filtering any number of nodes may receive and act simultaneously upon the same message.

2.2.1.4 Data consistency

Within a CAN network it is guaranteed that a message is accepted simultaneously either by all nodes or by no node. Thus data consistency is a property of the system achieved by the concepts of multicast and by error handling.

2.3 Bit-rate

The speed of CAN may be different in different systems. However, in a given system the bit-rate is uniform and fixed.

2.4 Priorities

The Identifier defines a static message priority during bus access.

2.5 Remote data request

By sending a Remote frame a node requiring data may request another node to send the corresponding Data frame. The Data frame and the corresponding Remote frame have the same Identifier.

2.6 Multi-master

When the bus is free any node may start to transmit a message. The node with the message of highest priority to be transmitted gains bus access.

2.7 Arbitration

Whenever the bus is free, any node may start to transmit a message. If two or more nodes start transmitting messages at the same time, the bus access conflict is resolved by bit-wise arbitration using the Identifier. The mechanism of arbitration guarantees that neither information nor time is lost. If a Data frame and a Remote frame with the same Identifier are initiated at the same time, the Data frame prevails over the Remote frame. During arbitration every transmitter compares the level of the bit transmitted with the level that is monitored on the bus. If these levels are equal the node may continue to send. When a recessive level is sent, but a dominant level is monitored (see Section 2.13), the node has lost arbitration and must withdraw without sending any further bits.

2.8 Data integrity

In order to achieve a very high integrity of data transfer, powerful measures of error detection, signalling and self-checking are implemented in every CAN node.

2.8.1 Error detection

To detect errors the following measures have been taken:

- Monitoring (each transmitter compares the bit levels detected on the bus with the bit levels being transmitted)
- Cyclic Redundancy Check (CRC)
- Bit-Stuffing
- Message Frame Check

2.8.2 Performance of error detection

The error detection mechanisms have the following properties:

- Monitoring:
  - All global errors are detected
  - All local errors at transmitters are detected
- CRC:
  - Up to 5 randomly transmitted errors within a sequence are detected
  - Burst errors of length less than 15 in a message are detected
  - Errors of any odd number of bits in a message are detected

The total residual error probability of undetected corrupted messages is less than 4.7×10^{-11}.

2.9 Error signalling and recovery time

Corrupted messages are flagged by any node detecting an error. Such messages are aborted and are retransmitted automatically. The recovery time from detecting an error until the start of the next message is at most 29 bit times, provided there is no further error.

2.10 Fault confinement

CAN nodes are able to distinguish between short disturbances and permanent failures. Defective nodes are switched off.

2.11 Connections

The CAN serial communication link is a bus to which a number of nodes may be connected. This number has no theoretical limit. Practically, the total number of nodes will be limited by delay times and/or electrical loads on the bus line.

2.12 Single channel

The bus consists of a single bidirectional channel that carries bits. From this data, resynchronization information can be derived. The way in which this channel is implemented is not fixed in this specification, e.g. single wire (plus ground), two differential wires, optical fibres, etc.

2.13 Bus values

The bus can have one of two complementary values: dominant or recessive. During simultaneous transmission of dominant and recessive bits, the resulting bus value will be dominant. For example, in the case of a wired-AND implementation of the bus, the dominant level would be represented by a logical ‘0’ and the recessive level by a logical ‘1’.

Physical states (e.g. electrical voltage, light) that represent the logical levels are not given in this specification.

2.14 Acknowledgement

All receivers check the consistency of the message being received and will acknowledge a consistent message and flag an inconsistent message.

2.15 Sleep mode/wake-up

To reduce the system’s power consumption, a CAN device may be set into sleep mode, in which there is no internal activity and the bus drivers are disconnected. The sleep mode is finished with a wake-up by any bus activity or by internal conditions of the system. On wake-up, the internal activity is restarted, although the transfer layer will wait for the system’s oscillator to stabilize and then wait until it has synchronized itself to the bus activity (by checking for eleven consecutive recessive bits), before the bus drivers are set to the ‘on-bus’ state again. In order to wake up other sleeping nodes on the network, a special wake-up message with the dedicated, lowest possible Identifier (rrr rrrd rrrr; r=recessive, d=dominant) may be used.

3
MESSAGE TRANSFER

3.1 Definition of transmitter/receiver

3.1.1 Transmitter

A node originating a message is called the TRANSMITTER of that message. The node continues to be TRANSMITTER until the bus is idle or the node loses ARBITRATION.

3.1.2 Receiver

A node is called the RECEIVER of a message if it is not the TRANSMITTER of that message, and the bus is not idle.

3.2 Frame types

Message transfer is manifested and controlled by four different frame types:

- A Data frame carries data from a transmitter to the receivers.
- A Remote frame is transmitted by a bus node to request the transmission of the Data frame with the same Identifier.
- An Error frame is transmitted by any node on detecting a bus error.
- An Overload frame is used to provide for an extra delay between the preceding and the succeeding Data or Remote frames.

Data frames and Remote frames are separated from preceding frames by an Interframe space.

Figure 3-11 at the end of this section summarizes all the frame formats.

3.2.1 Data frame

A Data frame is composed of seven different bit fields: Start of frame, Arbitration field, Control field, Data field, CRC field, ACK field, End of frame. The Data field can be of length zero.

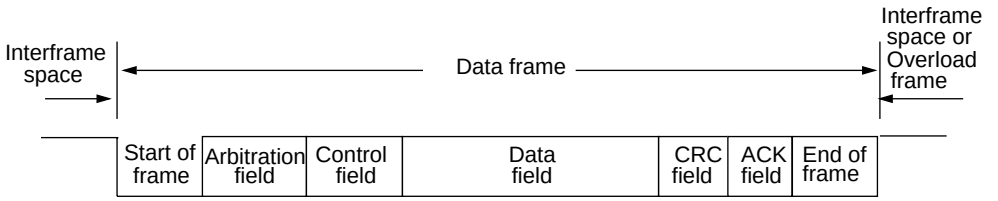


Figure 3-1 Data frame

3.2.1.1 Start of frame

Start of frame marks the beginning of Data frames and Remote frames. It consists of a single dominant bit.

A node is only allowed to start transmission when the bus is idle (see Section 3.2.5.2). All nodes have to synchronize to the leading edge caused by Start of frame(see Section 6.9.1) of the node starting transmission first.

3.2.1.2 Arbitration field

The Arbitration field consists of the Identifier and the RTR bit.

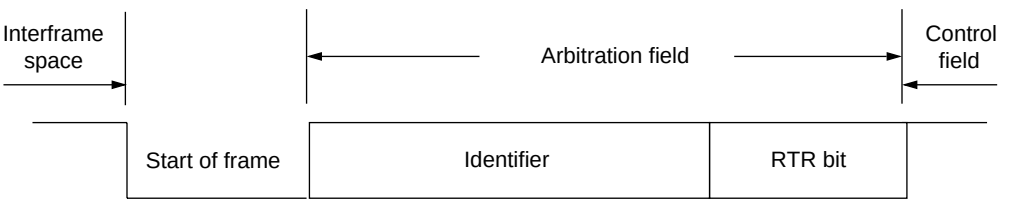


Figure 3-2 Arbitration field

Identifier

The Identifier's length is 11 bits. These bits are transmitted in the order from ID10 to ID0. The least significant bit is ID0. The 7 most significant bits must not be all recessive.

RTR bit (Remote transmission request bit)

In Data frames the RTR bit must be dominant. Within a Remote frame the RTR bit must be recessive.

3.2.1.3 Control field

The Control field consists of six bits. It includes the Data length code and two bits reserved for future expansion. The reserved bits must be sent as dominant. Receivers accept dominant and recessive bits in all combinations.

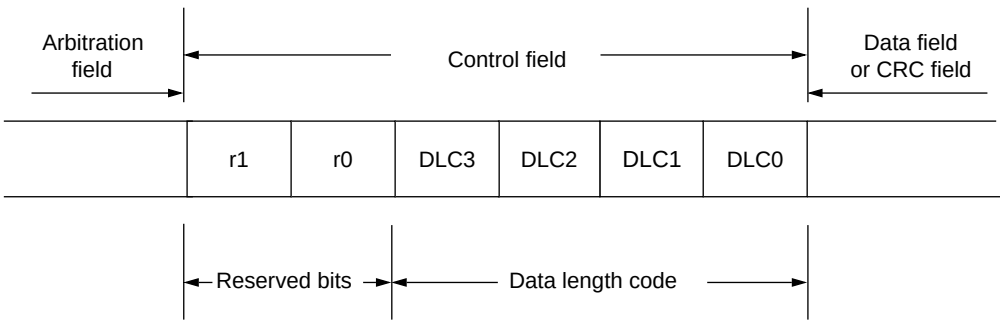


Figure 3-3 Control field

DATA Length Code

The number of bytes in the Data field is indicated by the Data length code. This Data length code is 4 bits wide and is transmitted within the Control field. The DLC bits can code data lengths from 0 to 8 bytes; other values are not permitted.

3.2.1.4 Data field

The Data field consists of the data to be transferred within a Data frame. It can contain from 0 to 8 bytes, each of which contain 8 bits which are transferred MSB first.

Table 3-1 Data length coding

DATA LENGTH CODE				DATA BYTE COUNT
DLC3	DLC2	DLC1	DLC0	
d	d	d	d	0
d	d	d	r	1
d	d	r	d	2
d	d	r	r	3
d	r	d	d	4
d	r	d	r	5
d	r	r	d	6
d	r	r	r	7
r	d	d	d	8
d = “dominant r = “recessive”				

3.2.1.5 CRC field

The CRC field contains the CRC Sequence followed by a CRC Delimiter.

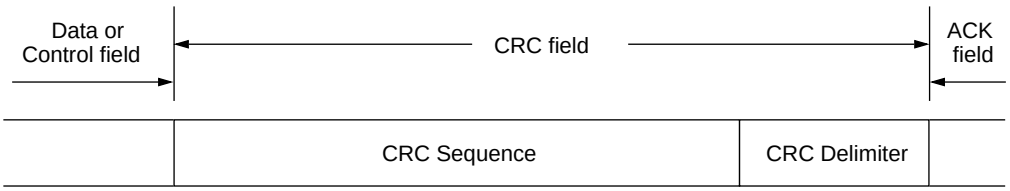


Figure 3-4 CRC field

CRC Sequence

The frame check sequence is derived from a cyclic redundancy code best suited to frames with bit counts less than 127 bits (BCH code).

In order to carry out the CRC calculation the polynomial to be divided is defined as the polynomial whose coefficients are given by the destuffed bit-stream consisting of Start of frame, Arbitration field, Control field, Data field(if present) and, for the 15 lowest coefficients, by 0. This polynomial is divided (the coefficients are calculated modulo-2) by the generator-polynomial:

$X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$

The remainder of this polynomial division is the CRC Sequence transmitted over the bus.

In order to implement this function, a 15-bit shift register CRC_RG(14:0) can be used. If NXTBIT denotes the next bit of the bit-stream, given by the destuffed bit sequence from Start of frame until the end of the Data field, the CRC Sequence is calculated as follows:

```
CRC_RG = 0                              //initialize shift register
REPEAT
       CRCNXT = NXTBIT EXOR CRC_RG(14)
       CRC_RG(14:1) = CRC_RG(13:0)      //shift left by...
       CRC_RG(0) = 0                    //...one position
       IF CRCNXT THEN
              CRC_RG(14:0) = CRC_RG(14:0) EXOR (4599 hex)
       ENDIF
UNTIL (CRC SEQUENCE starts or there is an ERROR condition)
```

After the transmission/reception of the last bit of the Data field, CRC_RG(14:0) contains the CRC Sequence.

CRC Delimiter

The CRC Sequence is followed by the CRC Delimiter which consists of a single recessive bit.

3.2.1.6 ACK field

The ACK field is two bits long and contains the ACK Slot and the ACK Delimiter. In the ACK field the transmitting node sends two recessive bits.

A RECEIVER which has received a valid message correctly reports this to the TRANSMITTER by sending a dominant bit during the ACK Slot (i.e. it sends ACK).

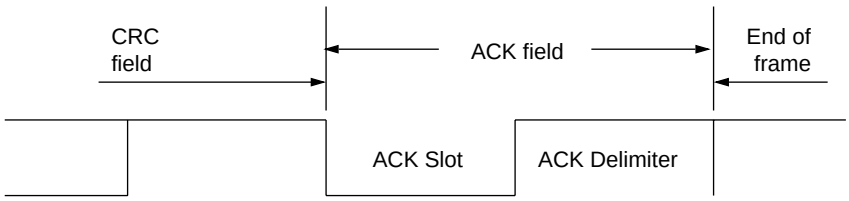


Figure 3-5 ACK field

ACK Slot

All nodes having received the matching CRC Sequence report this within the ACK Slot by overwriting the recessive bit of the TRANSMITTER by a dominant bit.

ACK Delimiter

The ACK Delimiter is the second bit of the ACK field and has to be a recessive bit. As a consequence, the ACK Slot is surrounded by two recessive bits (CRC Delimiter, ACK Delimiter).

3.2.1.7 End of frame

Each Data frame and Remote frame is delimited by a flag sequence consisting of seven recessive bits.

3.2.2 Remote frame

A node acting as a RECEIVER for certain data can stimulate the relevant source node to transmit the data by sending a Remote frame.

A Remote frame is composed of six different bit fields: Start of frame, Arbitration field, Control field, CRC field, ACK field, End of frame.

The RTR bit of a Remote frame is always recessive (cf. Data frames where the RTR bit is dominant).

There is no Data field in a Remote frame, irrespective of the value of the Data length code which is that of the corresponding Data frame and may be assigned any value within the admissible range 0... 8.

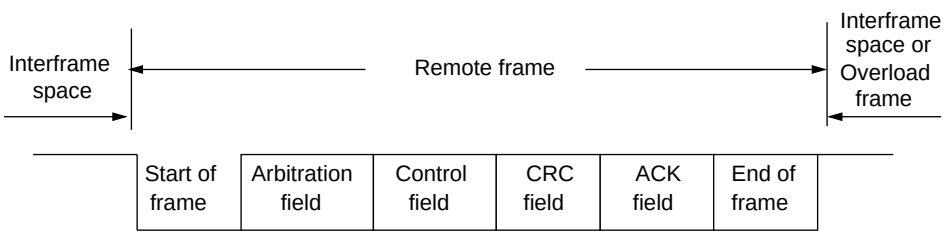


Figure 3-6 Remote frame

The polarity of the RTR bit indicates whether a transmitted frame is a Data frame (RTR bit dominant) or a Remote frame(RTR bit recessive).

3.2.3 Error frame

The Error frame consists of two distinct fields. The first field is given by the superposition of Error flags contributed from different nodes. The second field is the Error delimiter.

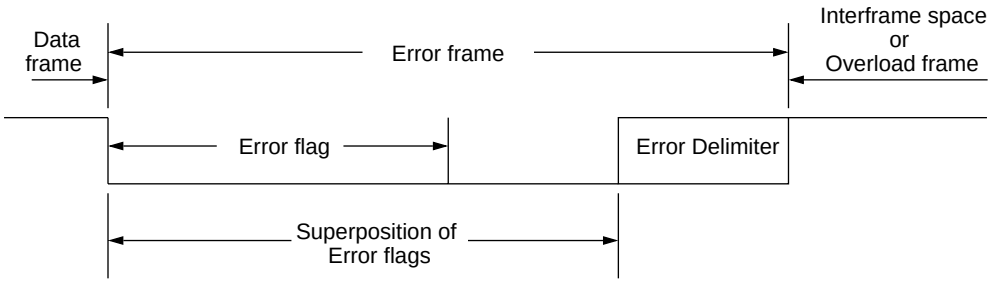


Figure 3-7 Error frame

In order to terminate an Error frame correctly, an error-passive node may need the bus to be bus-idle for at least three bit times (if there is a local error at an error-passive receiver). Therefore the bus should not be loaded to 100%.

3.2.3.1 Error flag

There are two forms of error flag: an ACTIVE Error flag and a PASSIVE Error flag.

1) ACTIVE Error flag consists of six consecutive dominant bits.
2) The PASSIVE Error flag consists of six consecutive recessive bits unless it is overwritten by dominant bits from other nodes.

An error-active node detecting an error condition signals this by the transmission of an ACTIVE Error flag. The Error flag's form violates the law of bit stuffing (see Section 3.4), applied to all fields from Start of frame to CRC Delimiter, or destroys the fixed form ACK field or End of frame field. As a consequence, all other nodes detect an error condition and each starts to transmit an Error flag. So the sequence of dominant bits which actually can be monitored on the bus results from a superposition of different Error flags transmitted by individual nodes. The total length of this sequence varies between a minimum of six and a maximum of twelve bits.

An error-passive node detecting an error condition tries to signal this by transmitting a PASSIVE Error flag. The error-passive node waits for six consecutive bits of equal polarity, beginning at the start of the PASSIVE Error flag. The PASSIVE Error flag is complete when these six equal bits have been detected.

3.2.3.2 Error Delimiter

The Error Delimiter consists of eight recessive bits.

After transmission of an Error flag each node sends recessive bits and monitors the bus until it detects a recessive bit. Afterwards it starts transmitting seven more recessive bits.

3.2.4 Overload frame

The Overload frame contains two bit fields, Overload flag and Overload Delimiter.

There are two kinds of Overload condition, both of which lead to the transmission of an Overload flag:

1) Where the internal conditions of a receiver are such that the receiver requires a delay of the next Data frameor Remote frame,
2) On detection of a dominant bit during INTERMISSION.

An Overload frame resulting from Overload condition 1 is only allowed to start at the first bit time of an expected INTERMISSION, whereas an Overload frame resulting from Overload condition 2 starts one bit after detecting the dominant bit.

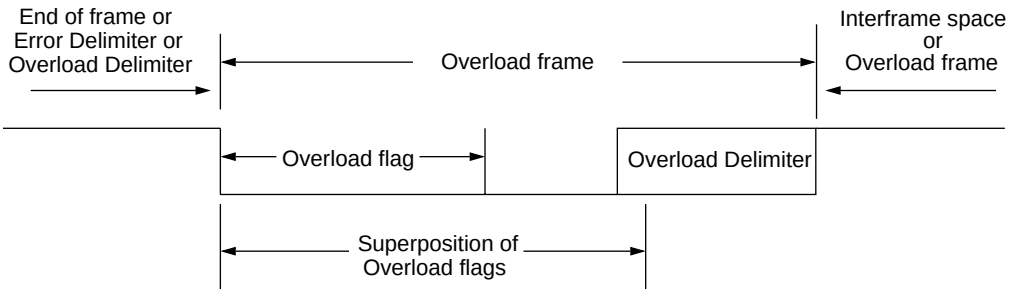


Figure 3-8 Overload frame

At most, two Overload frames may be generated to delay the next Data frame or Remote frame.

3.2.4.1 Overload flag

The Overload flag consists of six dominant bits. The overall form corresponds to that of the ACTIVE Error flag.

The Overload flag's form destroys the fixed form of the INTERMISSION field. As a consequence, all other nodes also detect an Overload condition and each starts to transmit an Overload flag. (In the event that there is a dominant bit detected during the third bit of INTERMISSION locally at some node, the other nodes will not interpret the Overload flag correctly, but interpret the first of these six dominant bits as Start of frame. The sixth dominant bit violates the rule of bit stuffing, thereby causing an error condition.)

3.2.4.2 Overload Delimiter

The Overload Delimiter consists of eight recessive bits.

The Overload Delimiter is of the same form as the Error Delimiter. After transmission of an Overload flag the node monitors the bus until it detects a transition from a dominant to a recessive bit. At this point of time every bus node has finished sending its Overload flag and all nodes start transmission of seven more recessive bits in coincidence.

3.2.5 Interframe space

Data frames and Remote frames are separated from preceding frames, whatever type they may be (Data frame, Remote frame, Error frame, Overload frame), by a field called Interframe space. In contrast, Overload frames and Error frames are not preceded by an Interframe space and multiple Overload frames are not separated by an Interframe space.

The Interframe space contains the bit fields INTERMISSION and Bus idle and, for error- passive nodes, which have been the transmitter of the previous message, Suspend transmission

(1) For nodes which are not error-passive or have been a receiver of the previous message, see Figure 3-9 and Figure 3-11(page 1 of 2).

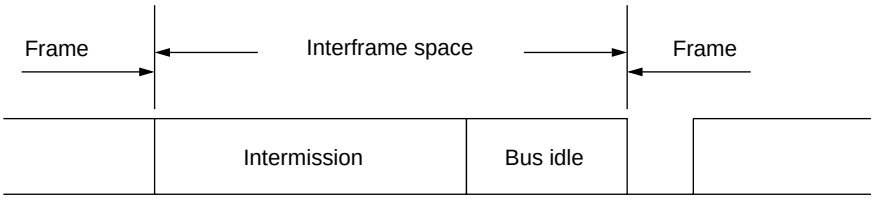


Figure 3-9 Interframe space (1)

(2) For error-passive nodes which have been the transmitter of the previous message, see Figure 3-10 and Figure 3-11 (page 2 of 2).

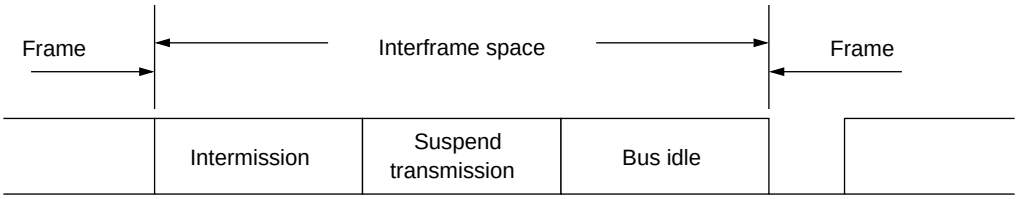


Figure 3-10 Interframe space (2)

3.2.5.1 INTERMISSION

INTERMISSION consists of three recessive bits.

During INTERMISSION no node is allowed to start transmission of a Data frame or Remote frame. The only action permitted is signalling of an Overload condition.

3.2.5.2 Bus idle

The period of Bus idle may be of arbitrary length. The bus is recognized to be free, and any node having something to transmit can access the bus. A message, pending during the transmission of another message, is started in the first bit following INTERMISSION.

The detection of a dominant bit on the bus is interpreted as Start of frame.

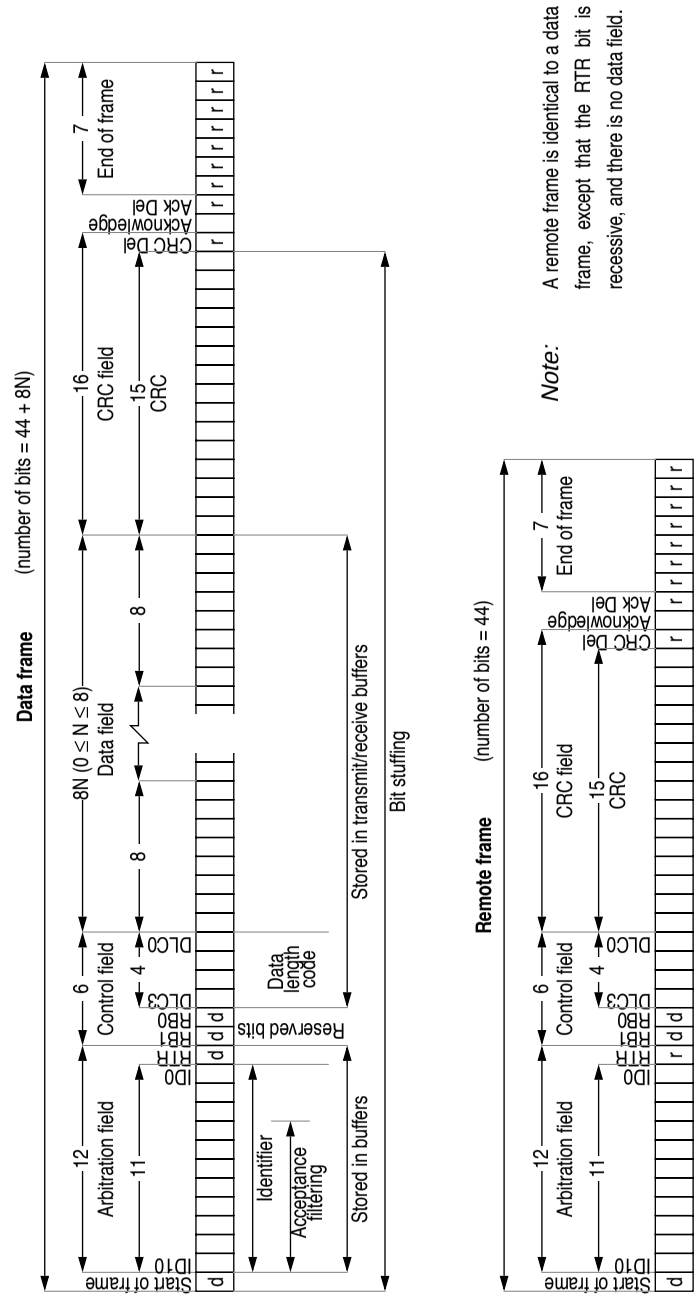


Figure 3-11 CAN frame formats (Page 1 of 2)

Figure 3-11 CAN frame formats (Page 2 of 2)

Note: An error frame can start anywhere in the middle of a frame.

Note: INT = Intermission
Suspend transmission is only for error passive nodes.

Note: An overload frame can only start at the end of a frame.
Maximum echo of overload flag is one bit.

3.2.5.3 Suspend transmission

After an error-passive node has transmitted a frame, it sends eight recessive bits following INTERMISSION, before starting to transmit a further message or recognizing the bus to be idle. If, meanwhile, a transmission (caused by another node) starts, the node will become the receiver of this message.

3.3 Message validation

The point in time at which a message is taken to be valid is different for the transmitter and the receivers of the message.

3.3.1 Transmitter

The message is valid for the transmitter if there is no error until the End of frame. If a message is corrupted, retransmission will follow automatically and according to the rules of prioritization. In order to be able to compete for bus access with other messages, retransmission has to start as soon as the bus is idle.

3.3.2 Receiver

The message is valid for the receiver if there is no error until the last but one bit of End of frame.

3.4 Bit-stream coding

The frame segments Start of frame, Arbitration field, Control field, Data field and CRC Sequence are coded by the method of bit stuffing. Whenever a transmitter detects five consecutive bits of identical value in the bit-stream to be transmitted, it automatically inserts a complementary bit in the actual transmitted bit-stream.

The remaining bit fields of the Data frame or Remote frame(CRC Delimiter, ACK field and End of frame) are of fixed form and not stuffed.

The Error frame and the Overload frame are also of fixed form and are not coded by the method of bit stuffing.

The bit-stream in a message is coded according to the Non-Return-to-Zero (NRZ) method. This means that during the total bit time the generated bit level is either dominant or recessive.

THIS PAGE LEFT BLANK INTENTIONALLY

4
ERROR HANDLING

4.1 Error detection

There are five different error types (which are not mutually exclusive). The following sections describe these errors.

4.1.1 Bit error

A node which is sending a bit on the bus also monitors the bus. The node must detect, and interpret as a Bit error, the situation where the bit value monitored is different from the bit value being sent. An exception to this is the sending of a recessive bit during the stuffed bit-stream of the Arbitration field or during the ACK Slot; in this case no Bit error occurs when a dominant bit is monitored.

A transmitter sending a PASSIVE Error flag and detecting a dominant bit does not interpret this as a Bit error.

4.1.2 Stuff error

A Stuff error must be detected and interpreted as such at the bit time of the sixth consecutive equal bit level (6 consecutive dominant or 6 consecutive recessive levels), in a message field which should be coded by the method of bit stuffing.

4.1.3 CRC error

The CRC sequence consists of the result of the CRC calculation by the transmitter.

The receivers calculate the CRC in the same way as the transmitter. A CRC error must be recognized if the calculated result is not the same as that received in the CRC sequence.

4.1.4 Form error

A FORM error must be detected when a fixed-form bit field contains one or more illegal bits.

4.1.5 Acknowledgement error

An Acknowledgement error must be detected by a transmitter whenever it does not monitor a dominant bit during ACK Slot

4.2 Error signalling

A node detecting an error condition signals this by transmitting an Error flag. An error-active node will transmit an ACTIVE Error flag; an error-passive node will transmit a PASSIVE Error flag.

Whenever a Bit error, a Stuff error, a Form error or an Acknowledgement error is detected by any node, that node will start transmission of an Error flag at the next bit time.

Whenever a CRC error is detected, transmission of an Error flag will start at the bit following the ACK Delimiter, unless an Error flag for another error condition has already been started.

5
FAULT CONFINEMENT

5.1 CAN node status

With respect to fault confinement, a node may be in one of three states: error-active, error-passive, or bus-off.

An error active node can normally take part in bus communication and sends an ACTIVE Error flag when an error has been detected.

An error-passive node must not send an ACTIVE Error flag. It takes part in bus communication, but when an error has been detected only a PASSIVE Error flag is sent. Also after a transmission, an error-passive node will wait before initiating a further transmission. (See Section 3.2.5.3)

A bus-off node is not allowed to have any influence on the bus (e.g. output drivers switched off).

5.2 Error counts

To facilitate fault confinement two counts are implemented in every bus node:

- TRANSMIT ERROR COUNT
- RECEIVE ERROR COUNT

These counts are modified according to the following 12 rules:

Note: More than one rule may apply during a given message transfer

1) When a RECEIVER detects an error, the RECEIVE ERROR COUNT will be increased by 1, except when the detected error was a Bit error during the sending of an ACTIVE Error flag or an Overload.
2) When a RECEIVER detects a dominant bit as the first bit after sending an Error flag, the RECEIVE ERROR COUNT will be increased by 8.

3) When a TRANSMITTER sends an Error flag, the TRANSMIT ERROR COUNT is increased by 8.

Exception 1

The TRANSMIT ERROR COUNT is not changed if:

– The TRANSMITTER is error-passive

and

– the TRANSMITTER detects an Acknowledgement error because of not detecting a dominant ACK

and

– the TRANSMITTER does not detect a dominant bit while sending its PASSIVE Error flag

Exception 2

The TRANSMIT ERROR COUNT is not changed if:

– The TRANSMITTER sends an Error flag because a Stuff error occurred during Arbitration (whereby the Stuff bit is located before the RTR bit)

and

– the Stuff bit should have been recessive

and

– the Stuff bit has been sent as recessive but is monitored as dominant

4) An error-active TRANSMITTER detects a Bit error while sending an ACTIVE Error flag or an Overload flag, the TRANSMIT ERROR COUNT is increased by 8.
5) An error-active RECEIVER detects a bit error while sending an ACTIVE Error flag or an Overload flag, the RECEIVE ERROR COUNT is increased by 8.
6) Any node tolerates up to 7 consecutive dominant bits after sending an ACTIVE Error flag or a PASSIVE Error flag. After detecting the eighth consecutive dominant bit s and after each sequence of additional eight consecutive dominant bits, every TRANSMITTER increases its TRANSMIT ERROR COUNT by 8 and every RECEIVER increases its RECEIVE ERROR COUNT by 8.
7) After the successful transmission of a message (getting ACK and no error until End of frame is finished), the TRANSMIT ERROR COUNT is decreased by 1, unless it was already 0.

8) After the successful reception of a message (reception without error up to the ACK Slot and the successful sending of the ACK bit), the RECEIVE ERROR COUNT is decreased by 1, if it was between 1 and 127. If the RECEIVE ERROR COUNT was 0, it stays 0, and if it was greater than 127, then it will be set to a value between 119 and 127.

9) A node is error passive when the TRANSMIT ERROR COUNT equals or exceeds 128, or when the RECEIVE ERROR COUNT equals or exceeds 128. An error condition letting a node become error-passive causes the node to send an ACTIVE Error flag.

10) A node is bus-off when the TRANSMIT ERROR COUNT is greater than or equal to 256.

11) An error-passive node becomes error-active again when both the TRANSMIT ERROR COUNT and the RECEIVE ERROR COUNT are less than or equal to 127.

12) A node which is bus-off is permitted to become error-active (no longer bus-off) with its error counters both set to 0 after 128 occurrences of 11 consecutive recessive bits have been monitored on the bus.

Note: An error count value greater than about 96 indicates a heavily disturbed bus. It may be advantageous to provide the means to test for this condition.

Note: Start-up/Wake-up
If during system start-up only one node is on line, and if this node transmits some message, it will get no acknowledgement, detect an error and repeat the message. It can become “error-passive” but not bus-off due to this reason.

5

THIS PAGE LEFT BLANK INTENTIONALLY

6
BIT TIMING REQUIREMENTS

6.1 Nominal bit rate

The Nominal bit rate is the number of bits per second transmitted in the absence of resynchronization by an ideal transmitter.

6.2 Nominal bit time

NOMINAL BIT TIME = 1 / NOMINAL BIT RATE

The Nominal bit rate can be thought of as being divided into separate non-overlapping time segments. These segments are as shown below, and form the bit time as shown in Figure 6-1.

- SYNCHRONIZATION SEGMENT (SYNC_SEG)
- PROPAGATION TIME SEGMENT (PROP_SEG)
- PHASE BUFFER SEGMENT1 (PHASE_SEG1)
- PHASE BUFFER SEGMENT2 (PHASE_SEG2)

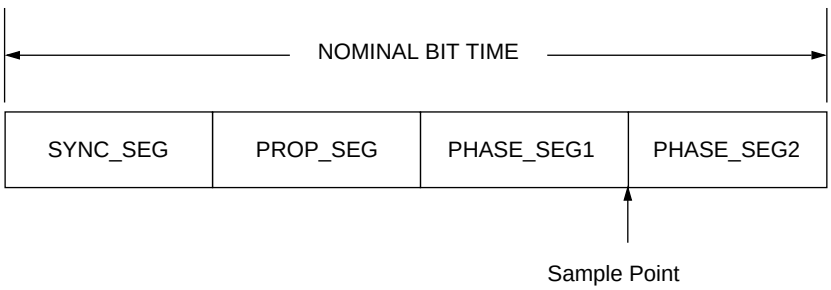


Figure 6-1 Nominal bit time

6.3 SYNC_SEG

This part of the bit time is used to synchronize the various nodes on the bus. An edge is expected to lie within this segment

6.4 PROP_SEG

This part of the bit time is used to compensate for the physical delay times within the network. It is twice the sum of the signal's propagation time on the bus line, the input comparator delay, and the output driver delay.

6.5 PHASE_SEG1, PHASE_SEG2

These Phase-Buffer-Segments are used to compensate for edge phase errors. These segments can be lengthened or shortened by resynchronization.

6.6 Sample point

The Sample point is the point in time at which the bus level is read and interpreted as the value of that respective bit. Its location is at the end of PHASE_SEG1.

6.7 Information processing time

The Information processing time is the time segment starting with the Sample point reserved for calculation of the subsequent bit level.

6.8 Time quantum

The Time quantum is a the fixed unit of time which can be derived from the oscillator period. There is a programmable prescaler, with integral values (with a range of at least 1 to 32) which allows a fixed unit of time, the Time quantum can have a length of

$$\text{TIME QUANTUM} = m \times \text{MINIMUM TIME QUANTUM}$$

where m is the value of the prescaler.

6.8.1 Length of time segments

- SYNC_SEG is 1 Time quantum long.
- PROP_SEG is programmable to be 1, 2, ...8 Time quanta long.
- PHASE_SEG1 is programmable to be 1, 2, ...8 Time quanta long.
- PHASE_SEG2 is the maximum of PHASE_SEG1 and the Information processing time.
- The Information processing time is less than or equal to 2 Time quanta long.

The total number of Time quanta in a bit time must be programmable over a range of at least 8 to 25.

6.9 Synchronization

6.9.1 Hard synchronization

After a Hard synchronization the internal bit time is restarted with SYNC_SEG. Thus Hard synchronization forces the edge which has caused the Hard synchronization to lie within the Synchronization segment of the restarted bit time.

6.9.2 Resynchronization jump width

As a result of resynchronization, PHASE_SEG1 may be lengthened or PHASE_SEG2 may be shortened. The amount by which the Phase buffer segments may be altered may not be greater than the Resynchronization jump width. The Resynchronization jump width is programmable between 1 and the smaller of 4 and PHASE_SEG1 Time quanta.

Clocking information may be derived from transitions from one bit value to the other. The property that only a fixed maximum number of successive bits have the same value provides the possibility of resynchronising a bus node to the bit-stream during a frame.

The maximum length between two transitions which can be used for resynchronization is 29 bit times.

6.9.3 Phase error of an edge

The PHASE error of an edge is given by the position of the edge relative to SYNC_SEG, measured in Time quanta. The sign of PHASE error is defined as follows:

e < 0 if the edge lies after the Sample point of the previous bit,

e = 0 if the edge lies within SYNC_SEG,

e > 0 if the edge lies before the Sample point.

6.9.4 Resynchronization

The effect of a Resynchronizationis the same as that of Hard synchronization, when the magnitude of the PHASE error of the edge which causes the Resynchronization is less than or equal to the programmed value of the Resynchronization jump width.

When the magnitude of the PHASE error is larger than the Resynchronizationjump width, and if the PHASE error is positive, then PHASE_SEG1 is lengthened by an amount equal to the Resynchronization jump width.

When the magnitude of the PHASE error is larger than the Resynchronization jump width, and if the PHASE error is negative, then PHASE_SEG2 is shortened by an amount equal to the Resynchronization jump width.

6.9.5 Synchronization rules

Hard synchronization and Resynchronization are the two forms of synchronization. They obey the following rules:

1) Only one synchronization within one bit time is allowed.
2) An edge will be used for synchronization only if the value detected at the previous Sample point (previous read bus value) differs from the bus value immediately after the edge.
3) Hard synchronization is performed whenever there is a recessive to dominant edge during Bus idle.
4) All other recessive to dominant edges (and optionally dominant to recessive edges in the case of low bit rates) fulfilling the rules 1 and 2 will be used for Resynchronization with the exception that a transmitter will not perform a Resynchronization as a result of a recessive to dominant edge with a positive PHASE error, if only recessive to dominant edges are used for Resynchronization.

7
INCREASING OSCILLATOR TOLERANCE

This section describes an upwards compatible modification to the CAN protocol, as specified in Sections 1 to 6.

7.1 Protocol modifications

In order to increase the maximum oscillator tolerance above the 0.5% currently possible, the following modifications, which are upwards compatible with the existing CAN specification, are necessary:

1) If a CAN node samples a dominant bit at the third bit of INTERMISSION, then it will interpret this bit as a Start of frame bit.
2) If a CAN node has a message waiting for transmission and it samples a dominant bit at the third bit of INTERMISSION, it will interpret this as a Start of frame bit, and, with the next bit, start transmitting its message with the first bit of its Identifier without first transmitting a Start of frame bit and without becoming a receiver.
3) If a CAN node samples a dominant bit at the eighth bit (the last bit) of an ERROR Delimiter or Overload Delimiter, it will, at the next bit, start transmitting an Overload frame (not an Error frame). The Error Counters will not be incremented.
4) Only recessive to dominant edges will be used for synchronization.

In agreement with the existing specification, the following rules are still valid:

5) All CAN controllers synchronize on the Start of frame bit with a hard synchronization.
6) No CAN controller will send a Start of frame bit until it has counted three recessive bits of INTERMISSION.

These modifications allow a maximum oscillator tolerance of 1.58% and the use of a ceramic resonator at bus speeds up to 125 kbits/second. For the full bus speed range of the CAN protocol, a quartz oscillator is still required. The compatibility of the enhanced and the existing protocol is maintained, as long as:

7) CAN controllers with the enhanced and existing protocols, used in one and the same network, are all provided with a quartz oscillator.

The chip with the highest requirement for its oscillator accuracy determines the oscillator accuracy which is required from all the other nodes. Ceramic resonators can only be used when all the nodes in the network use the enhanced protocol.

7.2 Determination of the maximum synchronization length

As a basis for the calculation of the maximum oscillator tolerance, the maximum distance between two edges used for resynchronization and the minimum synchronization length necessary to correctly extract the information coded into the bit-stream will be determined.

7.2.1 Local error, where at least two of the nodes are Error ACTIVE

The distance from the last recessive to dominant edge to the next possible Start of frame is 29 bits. According to the rules of fault confinement, a receiver will increment its RECEIVE ERROR COUNT depending on the first bit after sending an Error flag. Therefore, receivers have to be able to distinguish between sequences of 12 and of 13 dominant bits.

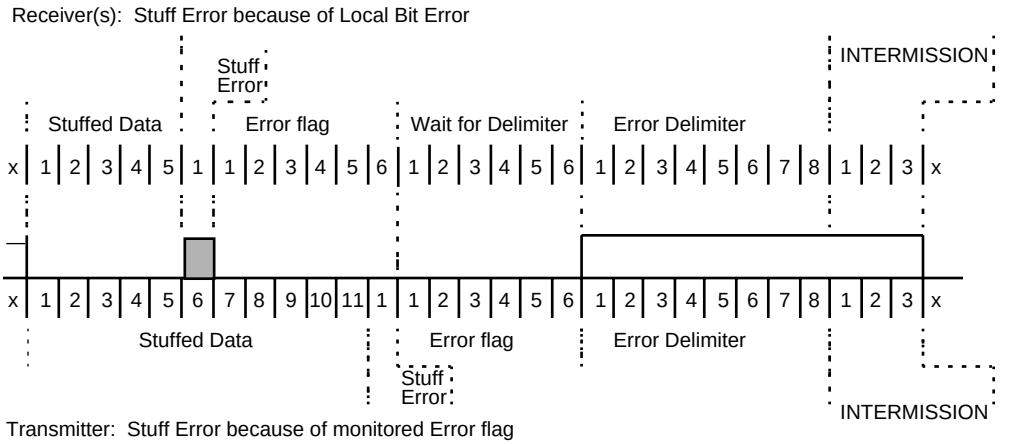


Figure 7-1 Local error

7.2.2 Two consecutive Overload frames

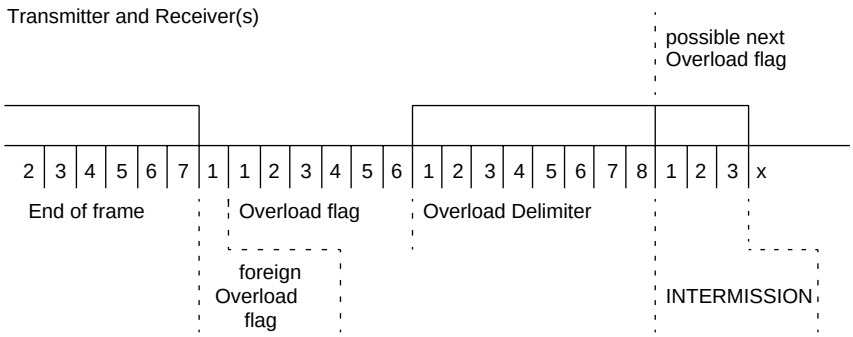


Figure 7-2 Two consecutive Overload frames

The distance from the start of the first Overload flag to the next recessive to dominant edge at the start of the second Overflow is 15 bits.

7.2.3 Acknowledge error at transmitter, where all nodes are Error PASSIVE

The distance from the last recessive to dominant edge to the next possible Start of frame is 29 bits. Since there is only one transmitter, the exact length of SUSPEND TRANSMISSION is not significant in this case, as long as it is at least 3 bits.

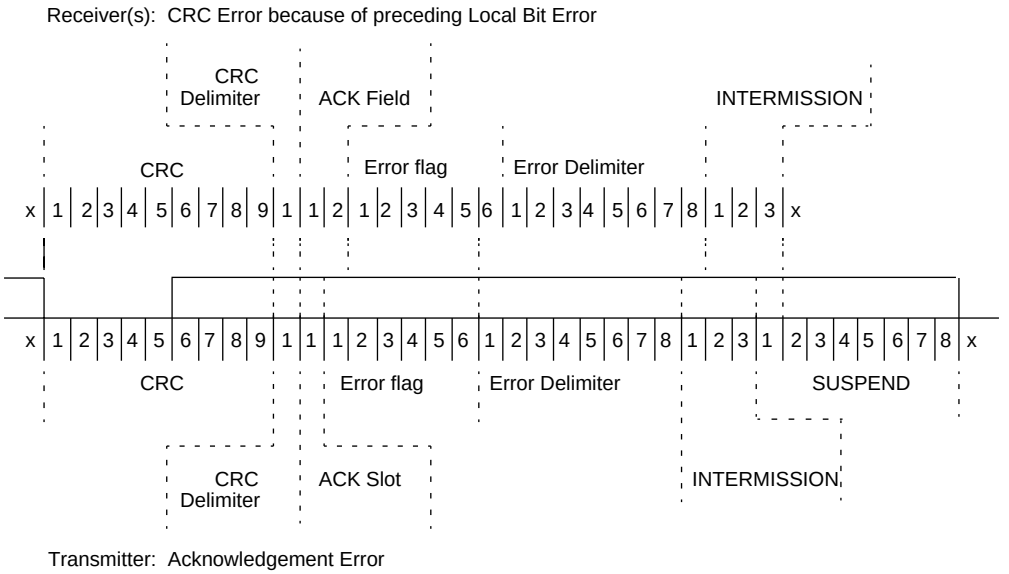


Figure 7-3 Acknowledge error at transmitter, all nodes Error PASSIVE

7.2.4 Local error at transmitter, where all nodes are Error PASSIVE

The distance from the last recessive to dominant edge to the next possible Start of frame is 28 bits for receivers and 30 bits for transmitters (during Arbitration, there can be more than one transmitter and therefore, SUSPEND TRANSMISSION has to be taken into consideration).

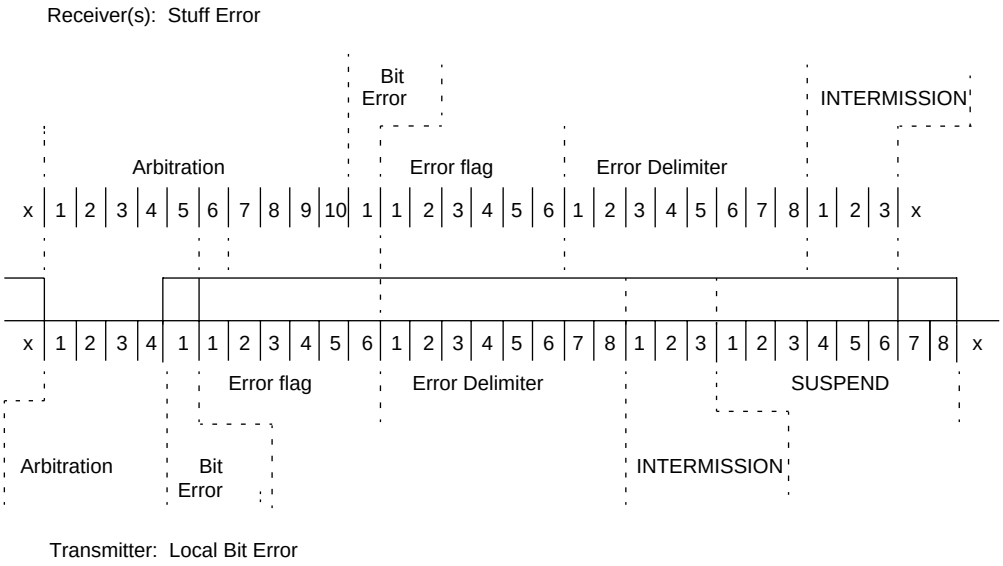


Figure 7-4 Local error at transmitter, all nodes Error PASSIVE

7.3 Bit timing

7.3.1 Construction of the bit timing for maximum oscillator tolerance

The maximum oscillator tolerance is reached when the length of the PHASE BUFFER SEGMENTs is the same as the maximum Resynchronization jump width and when only one Time quantum is used for delay compensation.

The delay to be compensated for due to the bit-wise arbitration mechanism is two times the sum of:

- the delay of the driver circuit
- the delay of the bus line
- the delay of the receiver circuit

Assumptions (dependent on the external circuitry):

delay of driver	=	200 ns
delay of bus line (40 m)	=	220 ns
delay of receiver circuit	=	80 ns

This allows a Time quantum of 1 µs and a maximum bit rate of 100 kbits/second with the maximum possible oscillator tolerance.

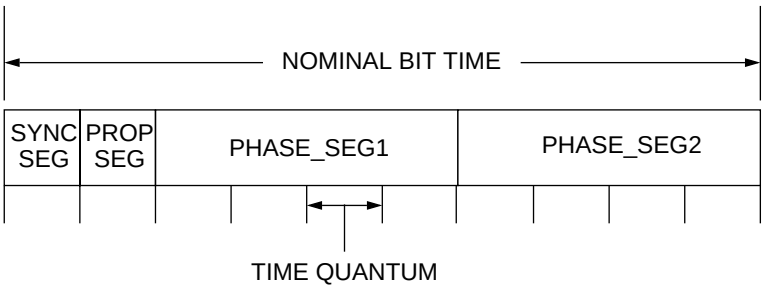


Figure 7-5 Bit timing for maximum oscillator tolerance

7.3.2 Construction of the bit timing for maximum bit rate

The largest possible part of the bit time will be used for delay compensation. PHASE_SEG1 and Resynchronization jump width will be limited to 1 Time quantum.

Assumptions (dependent on the external circuitry):

delay of driver = 50 ns
delay of bus line (40 m) = 220 ns
delay of receiver circuit = 30 ns

With a Time quantum of 100 ns, 6 Time quanta are needed for delay compensation. This allows a bit rate of 1 Mbits/second.

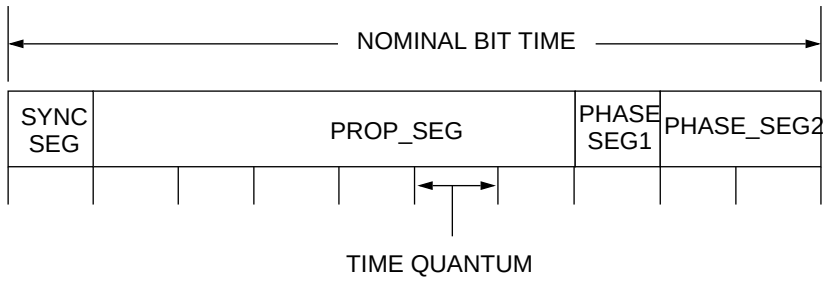


Figure 7-6 Bit timing for maximum bit rate

7.4 Calculation of the oscillator tolerance

In the following discussion, BT is the NOMINAL BIT TIME, SJW the RESYNCHRONIZATION JUMP WIDTH, PS1 and PS2 the length of the PHASE SEGMENTs, and df the modulus of the difference between the actual and nominal oscillator frequency relative to the nominal oscillator frequency.

The CAN protocol requires that:

$$SJW \leq \min(PS1, PS2) \quad [1]$$

$$BT > PS1 + PS2 \quad [2]$$

$$BT > 2\ SJW \quad [3]$$

$$PS2 \geq PS1 \quad [4]$$

In order to be able to sample correctly the first bit after sending an ACTIVE Error flag (essential for the correct localization of bus errors, see 7.2.1), the oscillator tolerance is limited to:

$$(2 \times df) \times (13 \times BT - PS2) < \min(PS1, PS2) \quad [5]$$

The worst case of 13 bits occurs after a STUFF Error on what should have been a recessive stuff bit. In this case the 13th bit after the last recessive to dominant edge needs to be correctly sampled for fault confinement.

For correct synchronization in the stuffed part of the bit-stream:

$$(2 \times df) \times 10 \times BT < SJW \quad [6]$$

For correct resynchronization until the next Start of frame (worst case, see Figure 6.9):

$$(2 \times df) \times (30 \times BT - PS2) < \min(PS1, PS2) \quad [7]$$

Similarly, if resynchronization occurs on both edges:

$$(2 \times df) \times (26 \times BT - PS2) < \min(PS1, PS2) \quad [8]$$

If [7] and [8] are fulfilled, [5] and [6] are fulfilled likewise.

In the following, the maximum oscillator tolerances of the actual and the enhanced CAN protocol are examined.

7.5 Maximum oscillator tolerances

From 7.4 it follows that with PS1, PS2 = 0.4 x BT and SJW = 0.4 x BT the oscillator tolerance is at its maximum.

7.5.1 Oscillator tolerance for existing CAN protocol

Synchronization only on edges from recessive to dominant:

[7] ⇒

$$(2 \times df) \times (30 \times BT - 0.4 \times BT) < 0.4 \times BT$$

$$df < 0.675\ \%$$

Synchronization only both edges:

[8] ⇒

$$(2 \times df) \times (26 \times BT - 0.4 \times BT) < 0.4 \times BT$$

$$df < 0.781\ \%$$

7.5.2 Oscillator tolerance for enhanced CAN protocol

The effect of the modifications described here is to reduce the number of consecutive bit times over which synchronization must be maintained. This is done by allowing the node to tolerate, after certain particularly long sequences of equal bits, phase shifts of up to a whole bit time.

Between the last recessive to dominant edge of a frame and the start of the next frame, there can be up to 30 bits. Modification 1 allows a tolerance of one logical bit here and ensures that if [9] holds (implied by [5]), that bus arbitration according to message Identifier priority will take place.

[3], [5], [6] ⇒

$$(2 \times df) \times (33 \times BT - PS2) < BT + \min(PS1, PS2) \qquad [9]$$

Therefore, if [5] is satisfied, we can tolerate sequences of up to at least 33 bits from a synchronization if an error of one logical bit can be tolerated in the synchronized sampled bit-stream.

Example 7.2.2 regards the case if the first bit of INTERMISSION is the 16th bit after the last recessive to dominant edge at the start of the first Overload frame. This implies that we have to allow a tolerance of one logical bit here for the start of the second Overload frame. Modification 3 caters for this.

All other sequences are covered by the preceding cases or by a thirteen bit synchronization length.

With the first three modifications to the CAN protocol, resychronization on recessive to dominant edges as well as dominant to recessive edges has absolutely no advantage over

resynchronization only on recessive to dominant edges. Hence modification 4 is introduced to avoid unnecessary complications to CAN implementations.

With the enhanced protocol the oscillator tolerance is given by

$$df < \min(PS1, PS2) / (2 \times (13 \times BT - PS2))$$

and

$$df < SJW / (2 \times (10 \times BT))$$

With PS1, PS2 = 0.4 x BT and SJW = 0.4 x BT an oscillator tolerance of

$$df < 1.58\ \%$$

can be allowed.

7.6 Resynchronization

In the existing CAN specification, the CAN controller was programmable to allow the use of only recessive to dominant edges for resynchronization, or of both recessive to dominant and dominant to recessive. With these protocol modifications, the oscillator tolerance is the same for both ways of resynchronization and therefore, the use of both edges brings no advantage and this feature can be removed.

7.7 Compatibility of existing and enhanced CAN protocols

Controllers using the existing CAN protocol must be equipped with a quartz oscillator. Controllers which use the enhanced protocol may be equipped with a quartz oscillator or a ceramic resonator. The following example shows that it is not possible to employ controllers using the existing CAN protocol together with controllers using the enhanced CAN protocol and driven by a ceramic resonator.

If no Error frame occurs, the longest bit sequence without the possibility of resynchronization occurs at the end of the message. In this case the synchronization length is 13 bits, resulting in a maximum phase shift between a quartz and ceramic node of (see Section 6.8), which can be tolerated by both existing and enhanced protocols.

Otherwise, if an Error frame or an Overload frame occurs, the next Start of frame bit, of a transmitter driven by a ceramic oscillator of minimum frequency, will be interpreted by a receiver driven by a quartz oscillator of maximum frequency and using the original CAN protocol as an Overload flag and will cause the transmission of another Overload frame. This will be repeated until the RECEIVE ERROR COUNT of the transmitter of the Start of frame bit (counting stuff errors after losing arbitration) reaches the error passive limit.

This review comes to the conclusion, that CAN controllers with the enhanced and existing protocols can be used in one and the same network, provided that all nodes are driven with a quartz oscillator. The chip with the highest requirement for its oscillator accuracy determines the oscillator accuracy which is required from all the other nodes. Ceramic resonators can only be used when all the nodes in the network use the enhanced protocol.

7.8 Assessment

Standard devices are subject to production variations, temperature dependency, and ageing which are specified as a tolerance. Most standard devices meet the following tolerances:

$$\text{quartz crystal } df \leq 0.1\ \%$$

$$\text{ceramic resonator } df \leq 1.2\ \%$$

Together with the results from sections 7.3 and 7.5, this means that, with the enhanced protocol, ceramic resonators can be used for bit rates up to and including 125 kbits/second. For higher bit rates a quartz oscillator is still required.

THIS PAGE LEFT BLANK INTENTIONALLY

BOSCH
CONTROLLER AREA NETWORK (CAN)
VERSION 2.0

PART B

THIS PAGE LEFT BLANK INTENTIONALLY

8
INTRODUCTION

The Controller Area Network (CAN) is a serial communications protocol that efficiently supports distributed real-time control with a very high level of data integrity.

Though conceived and defined by BOSCH in Germany for automotive applications, CAN is not restricted to that industry. CAN fulfils the communication needs of a wide range of applications, from high-speed networks to low-cost multiplex wiring.

For example, in automotive electronics, engine control units, sensors and anti-skid systems may be connected using CAN, with bit-rates up to 1 Mbit/s. At the same time, it is cost effective to build CAN into vehicle body electronics, such as lamp clusters and electric windows, to replace the wiring harness otherwise required.

The intention of the CAN specification is to achieve compatibility between any two CAN implementations. Compatibility, however, has different aspects with respect to, for example, electrical features and the interpretation of data to be transferred.

To achieve design transparency and implementation flexibility CAN has been subdivided into different layers according to the ISO/OSI Reference Model:

- The Data Link Layer
  - the Logical Link Control (LLC) sublayer
  - the Medium Access Control (MAC) sublayer
- The Physical Layer

In previous versions of the CAN specification the services and functions of the LLC and MAC sublayers of the Data Link Layer were described as layers called Object Layer and Transfer Layer.

The scope of the LLC sublayer includes: determining which messages received by the LLC sublayer are actually to be accepted; providing services for data transfer and for remote data request; and providing the means for recovery management and overload notifications. There is considerable freedom in defining object handling.

The MAC sublayer is principally concerned with the transfer protocol, i.e. controlling the framing, performing arbitration, error checking, error signalling and fault confinement. Within the MAC sublayer it is decided whether the bus is free for starting a new transmission, or whether reception of a message is just starting. Also, some general features of the bit-timing are regarded as part of the MAC sublayer. Modifications to the MAC sublayer cannot be made.

The Physical Layer covers the actual transfer of the bits between the different nodes, with respect to all electrical properties. Within a network the physical layer has to be the same for all nodes. However, there are many possible implementations of the Physical Layer.

The remainder of this document is principally concerned with the definition of the MAC sublayer and a small part of the LLC sublayer of the Data Link Layer, and the consequences of the CAN protocol for the surrounding layers.

9
BASIC CONCEPTS

9.1 Layered structure of a CAN node

The LLC sublayer is concerned with message filtering, overload notification and recovery management.

The MAC sublayer represents the kernel of the CAN protocol. It presents messages received to the LLC sublayer and accepts messages to be transmitted by the LLC sublayer. The MAC sublayer is responsible for message framing, arbitration, acknowledgement, error detection and signalling. The MAC sublayer is supervised by a self checking mechanism, called fault confinement, which distinguishes short disturbances from permanent failures.

The Physical Layer defines how signals are actually transmitted, dealing with the descriptions of bit timing, bit encoding and synchronization. The Physical Layer is not defined here, as it will vary according to the requirements of individual applications (for example, transmission medium and signal level implementations).

9.2 Messages

Information on the bus is sent in fixed format messages of different but limited length (see Section 10). When the bus is free, any connected node may start to transmit a new message.

9.2.1 Information routing

In CAN systems a node does not make use of any information about the system configuration (e.g. node addresses). This has several important consequences, which are described below.

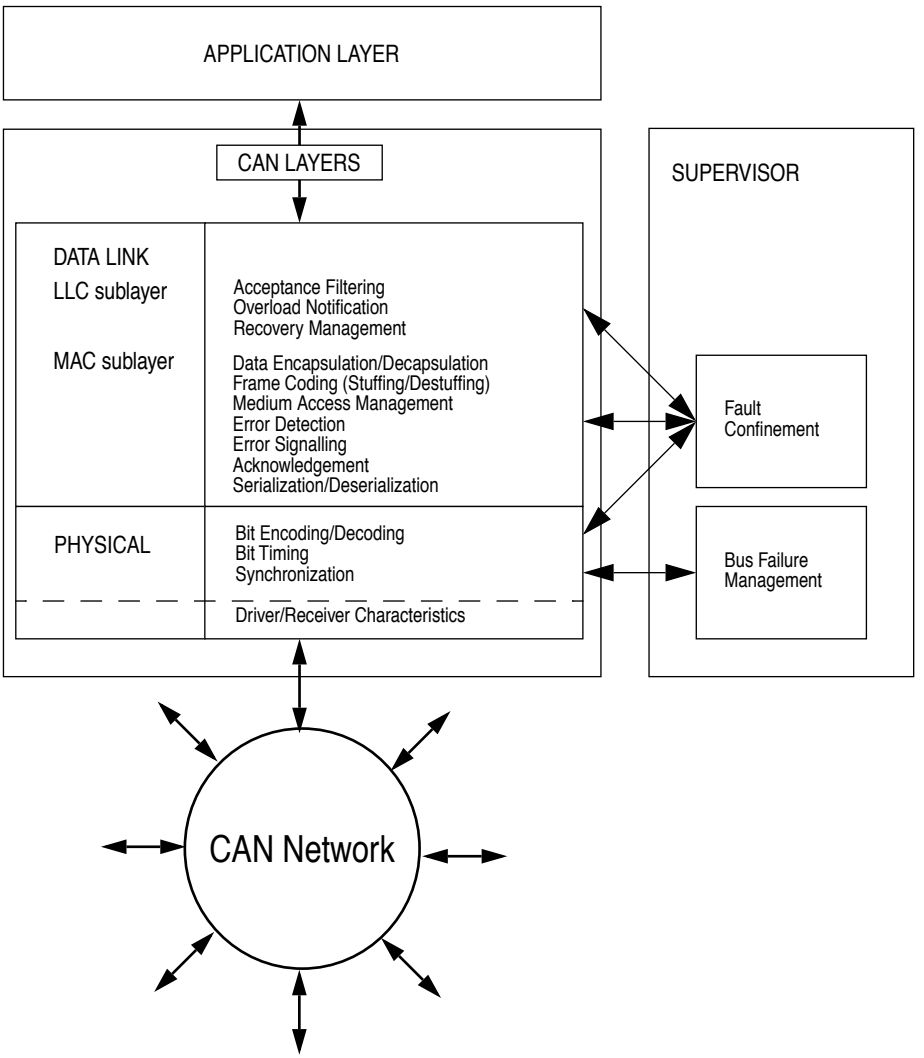


Figure 9-1 CAN layers

9.2.1.1 System flexibility

Nodes may be added to the CAN network without requiring any change in the software or hardware of any node or the application layer.

9.2.1.2 Message routing

The content of a message is described by an Identifier. The Identifier does not indicate the destination of the message, but describes the meaning of the data, so that all nodes in the network are able to decide by MESSAGE FILTERING whether the data is to be acted upon by them or not.

9.2.1.3 Multicast

As a consequence of the concept of MESSAGE FILTERING any number of nodes may receive and act simultaneously upon the same message.

9.2.1.4 Data consistency

Within a CAN network it is guaranteed that a message is accepted simultaneously either by all nodes or by no node. Thus data consistency is a property of the system achieved by the concepts of multicast and by error handling.

9.3 Bit-rate

The speed of CAN may be different in different systems. However, in a given system the bit-rate is uniform and fixed.

9.4 Priorities

The Identifier defines a static message priority during bus access.

9.5 Remote data request

By sending a Remote frame a node requiring data may request another node to send the corresponding Data frame. The Data frame and the corresponding Remote frame have the same Identifier.

9.6 Multi-master

When the bus is free any node may start to transmit a message. The node with the message of highest priority to be transmitted gains bus access.

9.7 Arbitration

Whenever the bus is free, any node may start to transmit a message. If two or more nodes start transmitting messages at the same time, the bus access conflict is resolved by bit-wise arbitration using the Identifier. The mechanism of arbitration guarantees that neither information nor time is lost. If a Data frame and a Remote frame with the same Identifier are initiated at the same time, the Data frame prevails over the Remote frame. During arbitration every transmitter compares the level of the bit transmitted with the level that is monitored on the bus. If these levels are equal the node may continue to send. When a recessive level is sent, but a dominant level is monitored (see Section 9.13), the node has lost arbitration and must withdraw without sending any further bits.

9.8 Data integrity

In order to achieve a very high integrity of data transfer, powerful measures of error detection, signalling and self-checking are implemented in every CAN node.

9.8.1 Error detection

To detect errors the following measures have been taken:

- Monitoring (each transmitter compares the bit levels detected on the bus with the bit levels being transmitted)
- Cyclic Redundancy Check (CRC)
- Bit-Stuffing
- Message Frame Check

9.8.2 Performance of error detection

The error detection mechanisms have the following properties:

- Monitoring:
  - All global errors are detected
  - All local errors at transmitters are detected
- CRC:
  - Up to 5 randomly transmitted errors within a sequence are detected
  - Burst errors of length less than 15 in a message are detected
  - Errors of any odd number of bits in a message are detected

The total residual error probability of undetected corrupted messages is less than 4.7×10^{-11}.

9.9 Error signalling and recovery time

Corrupted messages are flagged by any node detecting an error. Such messages are aborted and will be retransmitted automatically. The recovery time from detecting an error until the start of the next message is at most 31 bit times, provided there is no further error.

9.10 Fault confinement

CAN nodes are able to distinguish between short disturbances and permanent failures. Defective nodes are switched off.

9.11 Connections

The CAN serial communication link is a bus to which a number of nodes may be connected. This number has no theoretical limit. Practically, the total number of nodes will be limited by delay times and/or electrical loads on the bus line.

9.12 Single channel

The bus consists of a single bidirectional channel that carries bits. From this data, resynchronization information can be derived. The way in which this channel is implemented is not fixed in this specification, e.g. single wire (plus ground), two differential wires, optical fibres, etc.

9.13 Bus values

The bus can have one of two complementary values: dominant or recessive. During simultaneous transmission of dominant and recessive bits, the resulting bus value will be dominant. For example, in the case of a wired-AND implementation of the bus, the dominant level would be represented by a logical ‘0’ and the recessive level by a logical ‘1’.

Physical states (e.g. electrical voltage, light) that represent the logical levels are not given in this specification.

9.14 Acknowledgement

All receivers check the consistency of the message being received and will acknowledge a consistent message and flag an inconsistent message.

9.15 Sleep mode/wake-up

To reduce the system’s power consumption, a CAN device may be set into sleep mode, in which there is no internal activity and the bus drivers are disconnected. The sleep mode is finished with a wake-up by any bus activity or by internal conditions of the system. On wake-up, the internal activity is restarted, although the MAC sublayer will wait for the system’s oscillator to stabilize and then wait until it has synchronized itself to the bus activity (by checking for eleven consecutive recessive bits), before the bus drivers are set to the on-bus state again.

9.16 Oscillator Tolerance

A maximum oscillator tolerance of 1.58% is given allowing ceramic resonators to be used in applications with transmission rates of up to 125 kbit/s, as a general rule. For a more precise evaluation, refer to the following Technical Paper.

“Impact of Bit Representation on Transport Capacity and Clock Accuracy in Serial Data Streams” by S. Dais and M. Chapman.
SAE Technical Paper Series 890532, Multiplexing in Automobil SP-773, March 1989.

A quartz oscillator is required to achieve the full bus speed range of the CAN protocol.

The chip of the CAN network with the highest requirement for oscillator accuracy determines the oscillator accuracy from all the other nodes.

Note: CAN controllers following this CAN Specification and controllers following previous CAN Specifications 1.0 and 1.1, when used together in one network, must all be equipped with a quartz oscillator. Ceramic resonators can only be used in a network where all the nodes of the network follow CAN Specification 1.2 or later.

THIS PAGE LEFT BLANK INTENTIONALLY

10
MESSAGE TRANSFER

10.1 Definition of transmitter/receiver

10.1.1 Transmitter

A node originating a message is called TRANSMITTER of that message. The node continues to be TRANSMITTER until the bus is idle or the node loses Arbitration.

10.1.2 Receiver

A node is called RECEIVER of a message if it is not the TRANSMITTER of that message, and the bus is not idle.

10.2 Frame formats

There are two different frame formats which differ from each other by the length of their Identifier fields. Frames with 11 bit Identifier fields are denoted Standard Frames, while frames with 12 bit Identifier fields are denoted Extended frames.

10.3 Frame types

Message transfer is manifested and controlled by four different frame types:

- A Data frame carries data from a transmitter to the receivers.
- A Remote frame is transmitted by a bus node to request the transmission of the Data frame with the same Identifier.

- An Error frame is transmitted by any node on detecting a bus error.
- An Overload frame is used to provide for an extra delay between the preceding and the succeeding Data or Remote frames.

Data frames and Remote frames are separated from preceding frames by an Interframe space.

Figure 10-12 at the end of this section summarizes all the frame formats.

10.3.1 Data frame

A Data frame is composed of seven different bit fields: Start of frame, Arbitration field, Control field, Data field, CRC field, ACKfield, End of frame. The Data field can be of length zero.

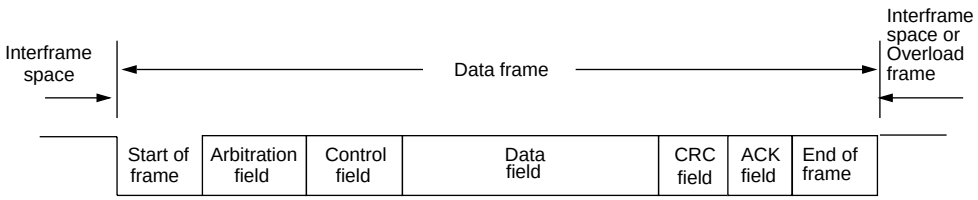


Figure 10-1 Data frame

10.3.1.1 Start of frame

Start of frame marks the beginning of Data frames and Remote frames. It consists of a single dominant bit.

A node is only allowed to start transmission when the bus is idle (see Section 10.3.5.2). All nodes have to synchronize to the leading edge caused by Start of frame (see Section 13.9.1) of the node starting transmission first.

10.3.1.2 Arbitration field

The format of the Arbitration field is different for Standard Format and Extended Format frames.

- In Standard Format the Arbitration field consists of the 11 bit Identifier and the RTR-Bit. The Identifier bits are denoted ID-28 ... ID-18.
- In Extended Format the Arbitration field consists of the 29 bit Identifier, the SRR-Bit, the IDE-Bit, and the RTR-Bit. The Identifier bits are denoted ID-28 ... ID-0.

Note: In order to distinguish between Standard Format and Extended Format the reserved bit, r1, in previous CAN specifications version 1.0-1.2 is now denoted as the IDE bit.

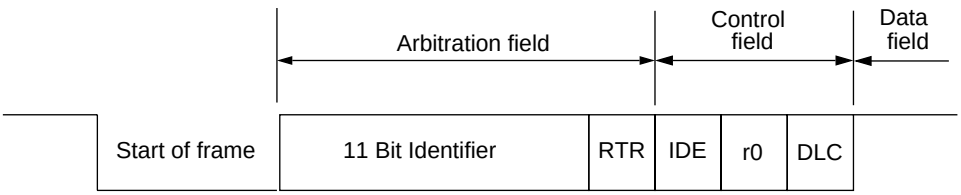


Figure 10-2 Arbitration field; Standard Format

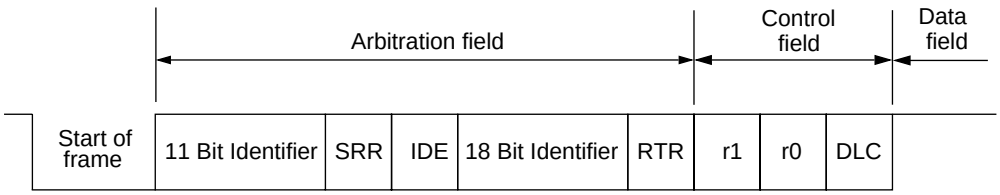


Figure 10-3 Arbitration field; Extended Format

Identifier

In Standard Format the Identifier's length is 11 bits and corresponds to the Base ID in Extended Format. These bits are transmitted in the order from ID-28 to ID-18. The least significant bit is ID-18. The 7 most significant bits (ID-28 - ID-22) must not be all recessive.

In Extended Format the Identifier's length is 29 bits. The format comprises two sections; Base ID with 11 bits and the Extended ID with 18 bits.

- Base ID: The Base ID consists of 11 bits. It is transmitted in the order from ID-28 to ID-18 and is equivalent to the format of the Standard Identifier. The Base ID defines the Extended Frame's base priority.
- Extended ID: The Extended ID consists of 18 bits. It is transmitted in the order of ID-17 to ID-0.

In a Standard Frame the Identifier is followed by the RTR bit.

RTR BIT (Standard Format and Extended Format)

Remote Transmission Request bit.

In Data frames the RTR bit has to be dominant. Within a Remote frame the RTR bit has to be recessive.

In an Extended Frame the Base ID is transmitted first, followed by the IDE bit and the SRR bit. The Extended ID is transmitted after the SRR bit.

SRR BIT (Extended Format)

Substitute Remote Request bit.

The SRR is a recessive bit. In Extended Frames the SRR bit is transmitted at the position of the RTR bit in Standard Frames and so substitutes for the RTR bit in the Standard Frame.

As a consequence, collisions between a Standard Frame and an Extended Frame, where the Base ID (see IDE BIT) of both frames is the same, are resolved in such a way that the Standard Frame prevails over the Extended Frame.

IDE BIT (Extended Format)

Identifier Extension bit.

The IDE bit belongs to:

- the Arbitration field for the Extended Format
- the Control field for the Standard Format

The IDE bit in the Standard Format is transmitted dominant, whereas in the Extended Format the IDE bit is recessive.

10.3.1.3 Control field

The Control field consists of six bits. The format of the Control field is different for Standard Format and Extended Format. Frames in Standard Format include the Data length code, the IDE bit, which is transmitted dominant, and the reserved bit r0. Frames in Extended Format include the Data length code and two reserved bits, r0 and r1. The reserved bits must be sent dominant, but the Receivers accept dominant and recessive bits in all combinations.

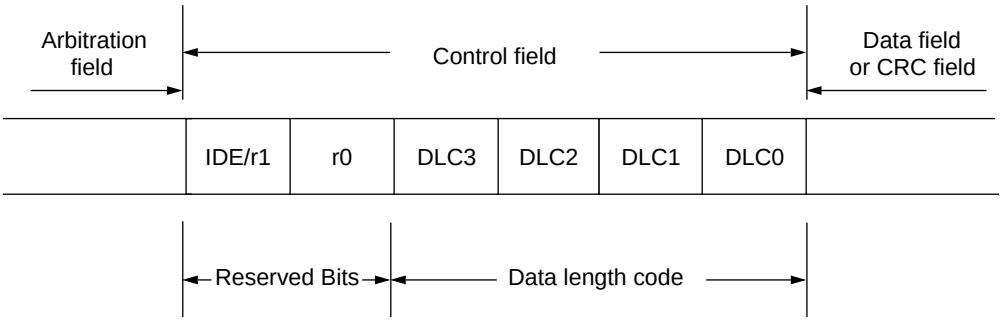


Figure 10-4 Control field; Standard Format and Extended Format

DATA LENGTH CODE (Standard Format and Extended Format)

The number of bytes in the Data field is indicated by the Data length code. This Data length code is 4 bits wide and is transmitted within the Control field. The DLC bits can code data lengths from 0 to 8 bytes; other values are not permitted.

10.3.1.4 Data field

The Data field consists of the data to be transferred within a Data frame. It can contain from 0 to 8 bytes, each of which contain 8 bits which are transferred MSB first.

Table 10-1 Data length coding

DATA LENGTH CODE				DATA BYTE COUNT
DLC3	DLC2	DLC1	DLC0	
d	d	d	d	0
d	d	d	r	1
d	d	r	d	2
d	d	r	r	3
d	r	d	d	4
d	r	d	r	5
d	r	r	d	6
d	r	r	r	7
r	d	d	d	8
d = dominant r = recessive				

10.3.1.5 CRC field (Standard Format and Extended Format)

The CRC field contains the CRC Sequence followed by a CRC Delimiter.

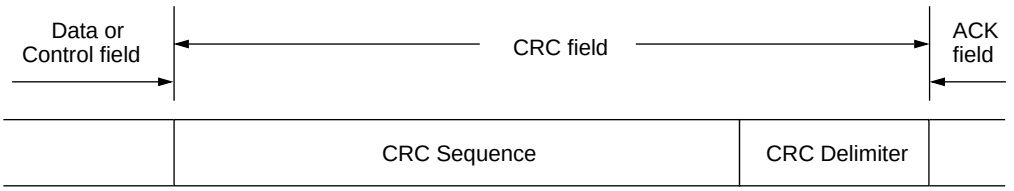


Figure 10-5 CRC field

CRC Sequence

The frame check sequence is derived from a cyclic redundancy code best suited to frames with bit counts less than 127 bits (BCH Code).

In order to carry out the CRC calculation the polynomial to be divided is defined as the polynomial whose coefficients are given by the destuffed bit-stream consisting of Start of frame, Arbitration field, Control field, Data field (if present) and, for the 15 lowest coefficients, by 0. This polynomial is divided (the coefficients are calculated modulo-2) by the generator-polynomial:

$X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$

The remainder of this polynomial division is the CRC Sequence transmitted over the bus.

In order to implement this function, a 15-bit shift register CRC_RG(14:0) can be used. If NXTBIT denotes the next bit of the bit-stream, given by the destuffed bit sequence from Start of frame until the end of the Data field, the CRC Sequence is calculated as follows:

```
CRC_RG = 0                              //initialize shift register
REPEAT
       CRCNXT = NXTBIT EXOR CRC_RG(14)
       CRC_RG(14:1) = CRC_RG(13:0)      //shift left by...
       CRC_RG(0) = 0                    //...one position
       IF CRCNXT THEN
              CRC_RG(14:0) = CRC_RG(14:0) EXOR (4599 hex)
       ENDIF
UNTIL (CRC SEQUENCE starts or there is an ERROR condition)
```

After the transmission/reception of the last bit of the Data field, CRC_RG(14:0) contains the CRC Sequence.

CRC Delimiter

The CRC Sequence is followed by the CRC Delimiter which consists of a single recessive bit.

10.3.1.6 ACK field (Standard Format and Extended Format)

The ACK field is two bits long and contains the ACK Slot and the ACK Delimiter. In the ACK field the transmitting node sends two recessive bits.

A RECEIVER which has received a valid message correctly reports this to the TRANSMITTER by sending a dominant bit during the ACK Slot (i.e. it sends ACK).

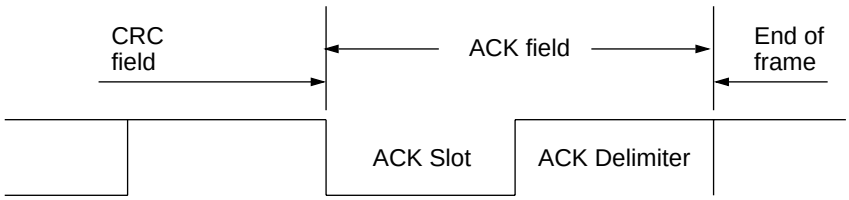


Figure 10-6 ACK field

ACK Slot

All nodes having received the matching CRC Sequence report this within the ACK Slot by overwriting the recessive bit of the TRANSMITTER by a dominant bit.

ACK Delimiter

The ACK Delimiter is the second bit of the ACK field and has to be a recessive bit. As a consequence, the ACK Slot is surrounded by two recessive bits (CRC Delimiter, ACK Delimiter).

10.3.1.7 End of frame

Each Data frame and Remote frame is delimited by a flag sequence consisting of seven recessive bits.

10.3.2 Remote frame

A node acting as a RECEIVER for certain data can stimulate the relevant source node to transmit the data by sending a Remote frame.

A Remote frame is composed of six different bit fields: Start of frame, Arbitration field, Control field, CRC field, ACK field, End of frame.

The RTR bit of a Remote frame is always recessive (cf. Data frames where the RTR bit is dominant).

There is no Data field in a Remote frame, irrespective of the value of the Data length code which is that of the corresponding Data frame and may be assigned any value within the admissible range 0... 8.

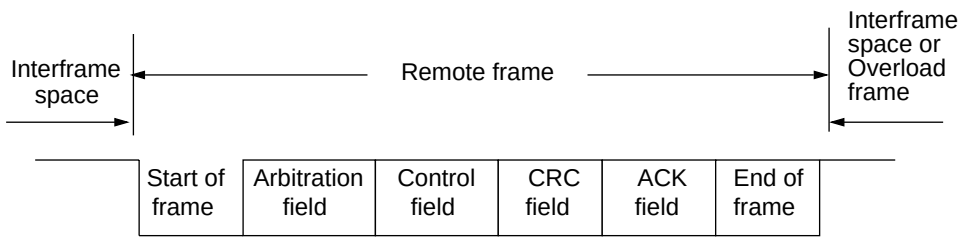


Figure 10-7 Remote frame

The polarity of the RTR bit indicates whether a transmitted frame is a Data frame (RTR bit dominant) or a Remote frame (RTR bit recessive).

10.3.3 Error frame

The Error frame consists of two distinct fields. The first field is given by the superposition of Error flags contributed from different nodes. The second field is the Error Delimiter.

In order to terminate an Error frame correctly, an error-passive node may need the bus to be bus-idle for at least three bit times (if there is a local error at an error-passive receiver). Therefore the bus should not be loaded to 100%.

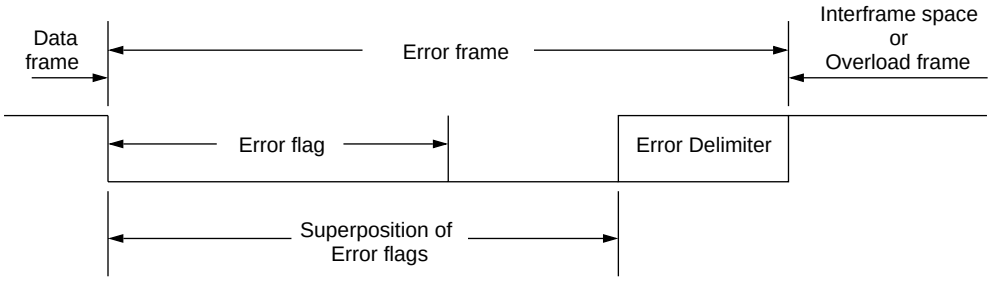


Figure 10-8 Error frame

10.3.3.1 Error flag

There are two forms of error flag: an ACTIVE Error flag and a PASSIVE Error flag.

1) ACTIVE Error flag consists of six consecutive dominant bits.
2) The PASSIVE Error flag consists of six consecutive recessive bits unless it is overwritten by dominant bits from other nodes.

An error-active node detecting an error condition signals this by the transmission of an ACTIVE Error flag. The Error flag's form violates the law of bit stuffing (see Section 10.7), applied to all fields from Start of frame to CRC Delimiter, or destroys the fixed form ACK field or End of frame field. As a consequence, all other nodes detect an error condition and each starts to transmit an Error flag. So the sequence of dominant bits which actually can be monitored on the bus results from a superposition of different Error flags transmitted by individual nodes. The total length of this sequence varies between a minimum of six and a maximum of twelve bits.

An error-passive node detecting an error condition tries to signal this by transmitting a PASSIVE Error flag. The error-passive node waits for six consecutive bits of equal polarity, beginning at the start of the PASSIVE Error flag. The PASSIVE Error flag is complete when these six equal bits have been detected.

10.3.3.2 Error delimiter

The Error Delimiter consists of eight recessive bits.

After transmission of an Error flag each node sends recessive bits and monitors the bus until it detects a recessive bit. Afterwards it starts transmitting seven more recessive bits.

10.3.4 Overload frame

The Overload frame contains two bit fields, Overload flag and Overload Delimiter.

There are three kinds of Overload condition which lead to the transmission of an Overload flag:

1) Where the internal conditions of a receiver are such that the receiver requires a delay of the next Data frame or Remote frame.
2) On detection of a dominant bit during INTERMISSION.
3) If a CAN node samples a dominant bit at the eighth bit (i.e. the last bit) of an Error Delimiter or Overload Delimiter, it will start transmitting an Overload frame (not an Error frame). The Error Counters will not be incremented.

An Overload frame resulting from Overload condition 1 is only allowed to start at the first bit time of an expected INTERMISSION, whereas Overload frames resulting from Overload conditions 2 and 3 start one bit after detecting the dominant bit.

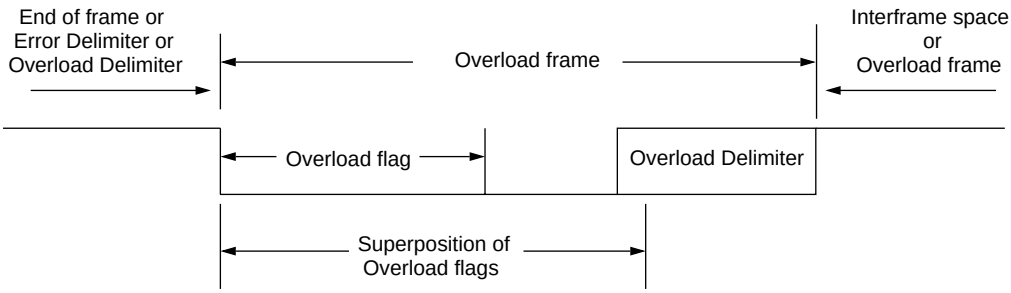


Figure 10-9 Overload frame

At most, two Overload frames may be generated to delay the next Data frame or Remote frame.

10.3.4.1 Overload flag

The Overload flag consists of six dominant bits. The overall form corresponds to that of the ACTIVE Error flag.

The Overload flag's form destroys the fixed form of the INTERMISSION field. As a consequence, all other nodes also detect an Overload condition and each starts to transmit an Overload flag. In the event that there is a dominant bit detected during the third bit of INTERMISSION, then it will interpret this bit as Start of frame.

Note: Controllers based on the CAN Specification version 1.0 and 1.1 interpret the of third bit of INTERMISSION in a different way. If the dominant bit was detected locally at some node, the other nodes will not interpret the Overload flag correctly, but will interpret the first of these six dominant bits as Start of frame; the sixth of these dominant bits violates the rule of bit stuffing causing an error condition.

10.3.4.2 Overload delimiter

The Overload Delimiter consists of eight recessive bits.

The Overload Delimiter is of the same form as the Error Delimiter. After transmission of an Overload flag the node monitors the bus until it detects a transition from a dominant to a recessive bit. At this point of time every bus node has finished sending its Overload flag and all nodes start transmission of seven more recessive bits in coincidence.

10.3.5 Interframe space

Data frames and Remote frames are separated from preceding frames, whatever type they may be (Data frame, Remote frame, Error frame, Overload frame), by a field called Interframe space. In contrast, Overload frames and Error frames are not preceded by an Interframe space and multiple Overload frames are not separated by an Interframe space.

The Interframe space contains the bit fields INTERMISSION and Bus idle and, for error- passive nodes, which have been the transmitter of the previous message, SUSPEND TRANSMISSION

(1) For nodes which are not error-passive or have been a receiver of the previous message, see Figure 10-10 and Figure 10-12 (page 1 of 2).

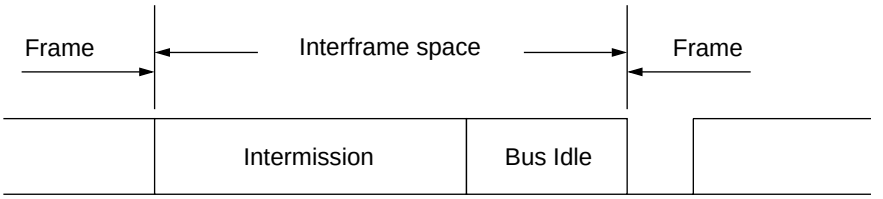


Figure 10-10 Interframe space (1)

(2) For error-passive nodes which have been the transmitter of the previous message, see Figure 10-11 and Figure 10-12 (page 2 of 2).

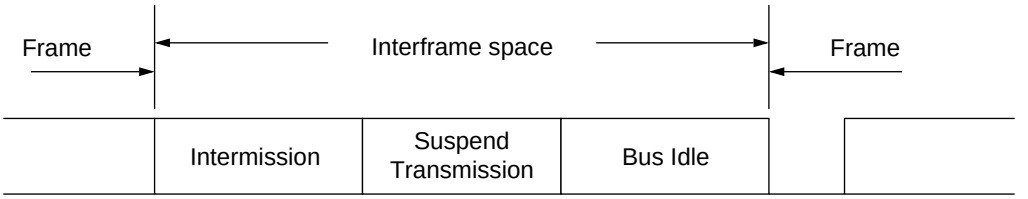


Figure 10-11 Interframe space (2)

10.3.5.1 INTERMISSION

INTERMISSION consists of three recessive bits.

During INTERMISSION no node is allowed to start transmission of a Data frame or Remote frame. The only action permitted is signalling of an Overload condition.

Note: If a CAN node has a message waiting for transmission and it samples a dominant bit at the third bit of INTERMISSION, it will interpret this as a Start of frame bit. With the next bit, it will start transmitting its message with the first bit of its Identifier without first transmitting a Start of frame bit and without becoming Receiver.

10.3.5.2 Bus idle

The period of Bus idle may be of arbitrary length. The bus is recognized to be free, and any node having something to transmit can access the bus. A message, pending during the transmission of another message, is started in the first bit following INTERMISSION.

The detection of a dominant bit on the bus is interpreted as Start of frame.

10.3.5.3 Suspend transmission

After an error-passive node has transmitted a frame, it sends eight recessive bits following INTERMISSION, before starting to transmit a further message or recognizing the bus to be idle. If, meanwhile, a transmission (caused by another node) starts, the node will become the receiver of this message.

10.4 Conformance with regard to frame formats

The Standard Format is equivalent to the Data/Remote frame Format as it is described in the CAN Specification 1.2. In contrast the Extended Format is a new feature of the CAN protocol. In order to allow the design of relatively simple controllers, the implementation of the Extended Format to its full extend is not required (e.g. send messages or accept data from messages in Extended Format), whereas the Standard Format must be supported without restriction.

New controllers are considered to be in conformance with this CAN Specification, if they have at least the following properties with respect to the Frame Formats defined in Section 10.2 and Section 10.3:

- Every new controller supports the Standard Format
- Every new controller can receive messages of the Extended Format. This requires that Extended frames are not destroyed just because of their format. It is, however, not required that the Extended Format must be supported by new controllers.

10.5 Message filtering

Message filtering is based upon the whole Identifier. Optional mask registers that allow any Identifier bit to be set 'don't care' for message filtering, may be used to select groups of Identifiers to be mapped into the attached receive buffers.

If mask registers are implemented every bit of the mask registers must be programmable, i.e. they can be enabled or disabled for message filtering. The length of the mask register can comprise the whole Identifier or only part of it.

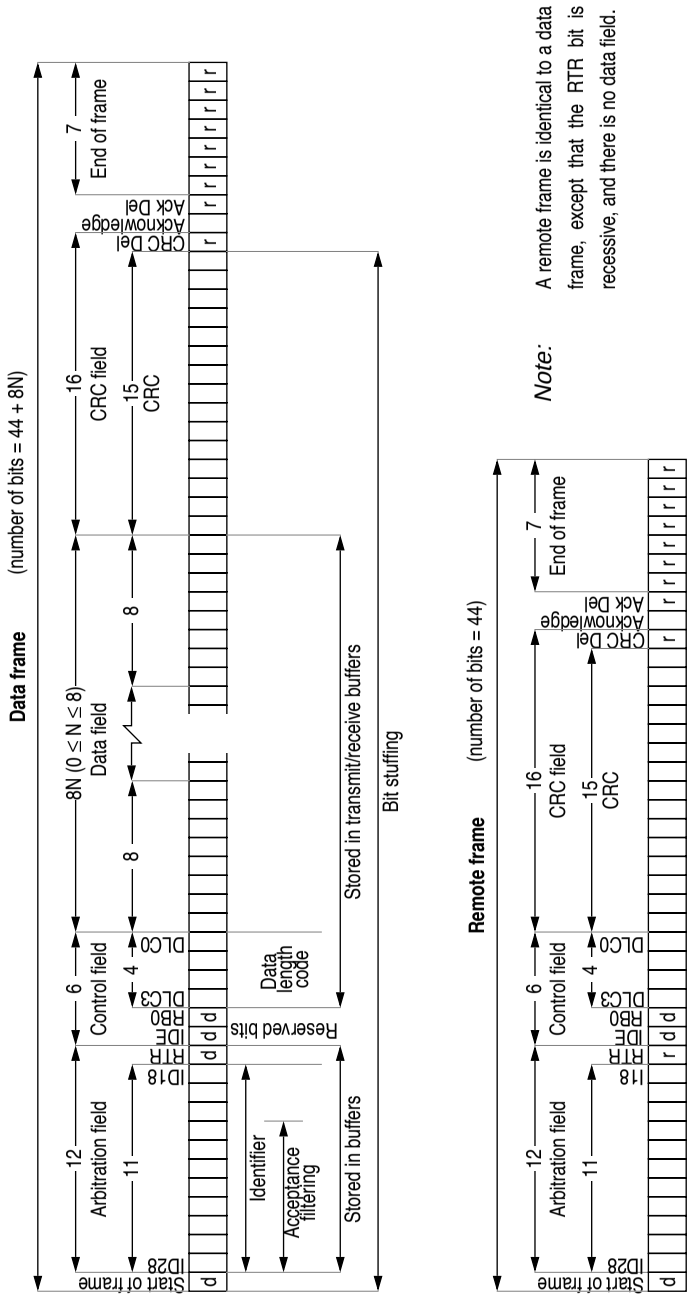


Figure 10-12 CAN frame format - Standard Format (Page 1 of 2)

Note: An error frame can start anywhere in the middle of a frame.

Note: INT = Intermission
Suspend transmission is only for error passive nodes.

Note: An overload frame can only start at the end of a frame.
Maximum echo of overload flag is one bit.

Figure 10-12 CAN frame format - Standard Format (Page 2 of 2)

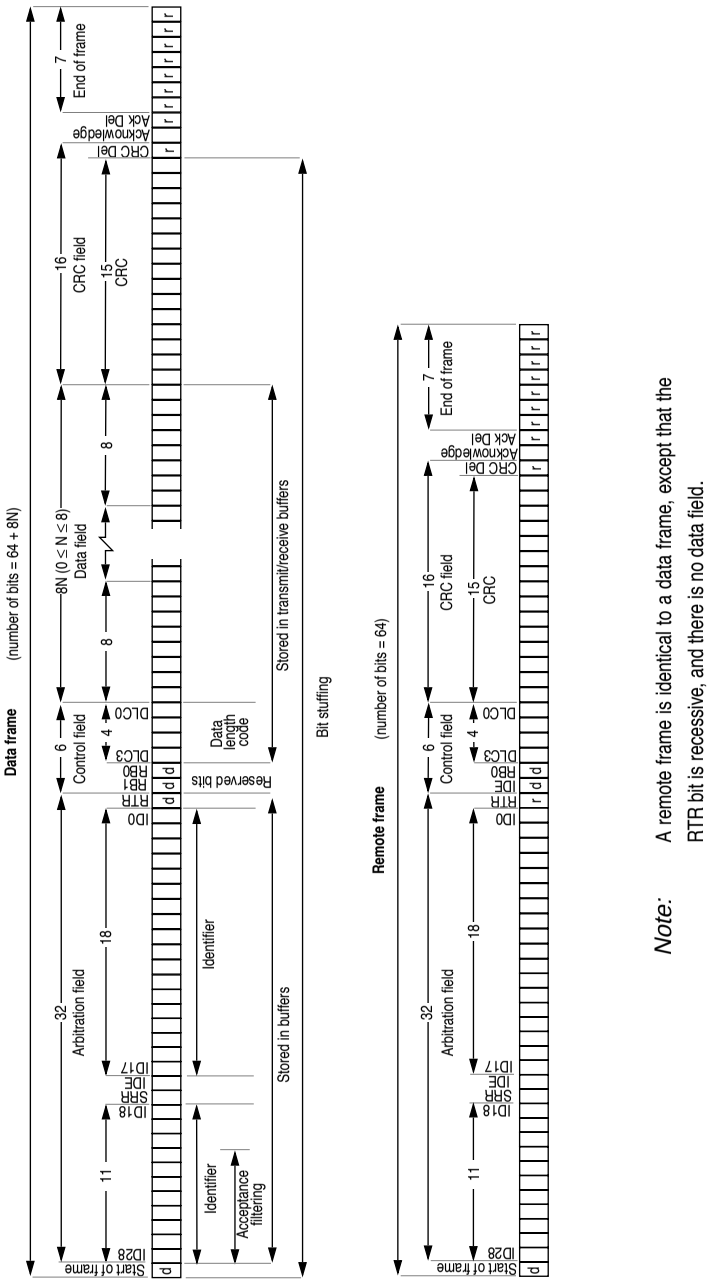


Figure 10-13 CAN frame format - Extended Format

Note: A remote frame is identical to a data frame, except that the RTR bit is recessive, and there is no data field.

10.6 Message validation

The point in time at which a message is taken to be valid is different for the transmitter and the receivers of the message.

10.6.1 Transmitter

The message is valid for the transmitter if there is no error until the End of frame. If a message is corrupted, retransmission will follow automatically and according to the rules of prioritization. In order to be able to compete for bus access with other messages, retransmission has to start as soon as the bus is idle.

10.6.2 Receiver

The message is valid for the receiver if there is no error until the last but one bit of End of frame.

10.7 Bit-stream coding

The frame segments Start of frame, Arbitration field, Control field, Data field and CRC Sequence are coded by the method of bit stuffing. Whenever a transmitter detects five consecutive bits of identical value in the bit-stream to be transmitted, it automatically inserts a complementary bit in the actual transmitted bit-stream.

The remaining bit fields of the Data frame or Remote frame(CRC Delimiter, ACK field and End of frame) are of fixed form and not stuffed.

The Error frame and the Overload frame are also of fixed form and are not coded by the method of bit stuffing.

The bit-stream in a message is coded according to the Non-Return-to-Zero (NRZ) method. This means that during the total bit time the generated bit level is either dominant or recessive.

THIS PAGE LEFT BLANK INTENTIONALLY

11
ERROR HANDLING

11.1 Error detection

There are five different error types (which are not mutually exclusive). The following sections describe these errors.

11.1.1 Bit error

A node which is sending a bit on the bus also monitors the bus. The node must detect, and interpret as a Bit Error, the situation where the bit value monitored is different from the bit value being sent. An exception to this is the sending of a recessive bit during the stuffed bit-stream of the Arbitration Field or during the ACK Slot; in this case no Bit Error occurs when a dominant bit is monitored.

A transmitter sending a PASSIVE Error Flag and detecting a dominant bit does not interpret this as a Bit Error.

11.1.2 Stuff error

A Stuff Error must be detected and interpreted as such at the bit time of the sixth consecutive equal bit level (6 consecutive dominant or 6 consecutive recessive levels), in a message field which should be coded by the method of bit stuffing.

11.1.3 CRC error

The CRC sequence consists of the result of the CRC calculation by the transmitter.

The receivers calculate the CRC in the same way as the transmitter. A CRC Error must be recognized if the calculated result is not the same as that received in the CRC sequence.

11.1.4 Form error

A Form Error must be detected when a fixed-form bit field contains one or more illegal bits.

Note: For a Receiver, a dominant bit during the last bit of End of Frame is not treated as Form Error.

11.1.5 Acknowledgement error

An Acknowledgement Error must be detected by a transmitter whenever it does not monitor a dominant bit during ACK Slot.

11.2 Error signalling

A node detecting an error condition signals this by transmitting an Error Flag. An error-active node will transmit an ACTIVE Error Flag; an error-passive node will transmit a PASSIVE Error Flag.

Whenever a Bit Error, a Stuff Error, a Form Error or an Acknowledgement Error is detected by any node, that node will start transmission of an Error Flag at the next bit time.

Whenever a CRC Error is detected, transmission of an Error Flag will start at the bit following the ACK Delimiter, unless an Error Flag for another error condition has already been started.

12
FAULT CONFINEMENT

12.1 CAN node status

With respect to fault confinement, a node may be in one of three states: error-active, error-passive, or bus-off.

An error active node can normally take part in bus communication and sends an ACTIVE Error Flag when an error has been detected.

An error-passive node must not send an ACTIVE Error Flag. It takes part in bus communication, but when an error has been detected only a PASSIVE Error Flag is sent. Also after a transmission, an error-passive node will wait before initiating a further transmission.

A bus-off node is not allowed to have any influence on the bus (e.g. output drivers switched off).

12.2 Error counts

To facilitate fault confinement two counts are implemented in every bus node:

- TRANSMIT ERROR COUNT
- RECEIVE ERROR COUNT

These counts are modified according to the following 12 rules:

Note: More than one rule may apply during a given message transfer

1) When a RECEIVER detects an error, the Receive Error Count will be increased by 1, except when the detected error was a Bit Error during the sending of an ACTIVE Error Flag or an Overload Flag.
2) When a RECEIVER detects a dominant bit as the first bit after sending an Error Flag, the Receive Error Count will be increased by 8.

3) When a TRANSMITTER sends an Error Flag, the Transmit Error Count is increased by 8.

Exception 1

The Transmit Error Countis not changed if:

– The TRANSMITTER is error-passive

and

– the TRANSMITTER detects an Acknowledgement Error because of not detecting a dominant”ACK

and

– the TRANSMITTER does not detect a dominant bit while sending its PASSIVE Error Flag

Exception 2

The Transmit Error Count is not changed if:

– The TRANSMITTER sends an Error Flag because a Stuff Error occurred during Arbitration

and

– the Stuff bit should have been recessive

and

– the Stuff Bit has been sent as recessive but is monitored as dominant

4) An error-active TRANSMITTER detects a Bit Error while sending an ACTIVE Error Flag or an Overload Flag, the Transmit Error Count is increased by 8.
5) An error-active RECEIVER detects a bit error while sending an ACTIVE Error Flag or an Overload Flag, the Receive Error Count is increased by 8.
6) Any node tolerates up to 7 consecutive dominant bits after sending an ACTIVE Error Flag or a PASSIVE Error Flag. After detecting the eighth consecutive dominant bit and after each sequence of additional eight consecutive dominant bits, every TRANSMITTER increases its Transmit Error Count by 8 and every RECEIVER increases its Receive Error Count by 8.
7) After the successful transmission of a message (getting ACK and no error until End of Frame is finished), the Transmit Error Count is decreased by 1, unless it was already 0.
8) After the successful reception of a message (reception without error up to the ACK Slot and the successful sending of the ACK bit), the Receive Error

Count is decreased by 1, if it was between 1 and 127. If the Receive Error Count was 0, it stays 0, and if it was greater than 127, then it will be set to a value between 119 and 127.

9) A node is error passive when the Transmit Error Count equals or exceeds 128, or when the Receive Error Count equals or exceeds 128. An error condition letting a node become error-passive causes the node to send an ACTIVE Error Flag.
10) A node is bus-off when the Transmit Error Count is greater than or equal to 256.
11) An error-passive node becomes error-active again when both the Transmit Error Count and the Receive Error Countare less than or equal to 127.
12) A node which is bus-off is permitted to become error-active (no longer bus-off) with its error counters both set to 0 after 128 occurrences of 11 consecutive recessive bits have been monitored on the bus.

Note: An error count value greater than about 96 indicates a heavily disturbed bus. It may be advantageous to provide the means to test for this condition.

Note: Start-up/Wake-up
If during system start-up only one node is on line, and if this node transmits some message, it will get no acknowledgement, detect an error and repeat the message. It can become error-passive but not bus-off due to this reason.

THIS PAGE LEFT BLANK INTENTIONALLY

13
BIT TIMING REQUIREMENTS

13.1 Nominal bit rate

The Nominal bit rate is the number of bits per second transmitted in the absence of resynchronization by an ideal transmitter.

13.2 Nominal bit time

$$\text{NOMINAL BIT TIME} = \frac{1}{\text{NOMINAL BIT RATE}}$$

The Nominal bit time can be thought of as being divided into separate non-overlapping time segments. These segments are as shown below, and form the bit time as shown in Figure 13-1.

- SYNCHRONIZATION SEGMENT (SYNC_SEG)
- PROPAGATION TIME SEGMENT (PROP_SEG)
- PHASE BUFFER SEGMENT1 (PHASE_SEG1)
- PHASE BUFFER SEGMENT2 (PHASE_SEG2)

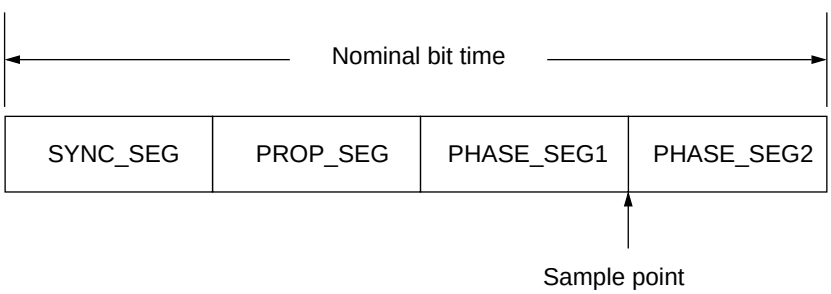


Figure 13-1 Nominal bit time

13.3 SYNC_SEG

This part of the bit time is used to synchronize the various nodes on the bus. An edge is expected to lie within this segment

13.4 PROP_SEG

This part of the bit time is used to compensate for the physical delay times within the network. It is twice the sum of the signal's propagation time on the bus line, the input comparator delay, and the output driver delay.

13.5 PHASE_SEG1, PHASE_SEG2

These Phase-Buffer-Segments are used to compensate for edge phase errors. These segments can be lengthened or shortened by resynchronization.

13.6 Sample point

The Sample point is the point in time at which the bus level is read and interpreted as the value of that respective bit. Its location is at the end of PHASE_SEG1.

13.7 Information processing time

The Information processing time is the time segment starting with the Sample point reserved for calculation of the subsequent bit level.

13.8 Time quantum

The Time quantum is a the fixed unit of time which can be derived from the oscillator period. There is a programmable prescaler, with integral values (with a range of at least 1 to 32) which allows a fixed unit of time, the Time quantum can have a length of

$$\text{TIME QUANTUM} = m \times \text{MINIMUM TIME QUANTUM}$$

where m is the value of the prescaler.

13.8.1 Length of time segments

- SYNC_SEG is 1 Time quantum long
- PROP_SEG is programmable to be 1, 2, ...8 Time quanta long
- PHASE_SEG1 is programmable to be 1, 2, ...8 Time quanta long
- PHASE_SEG2 is the maximum of PHASE_SEG1 and the Information processing time
- The Information processing time is less than or equal to 2 Time quanta long

The total number of Time quanta in a bit time must be programmable over a range of at least 8 to 25.

Note: Control units normally do not use different oscillators for the local CPU and its communication device. Therefore the oscillator frequency of a CAN device tends to be that of the local CPU and is determined by the requirements of the control unit. In order to derive the desired bit rate, programmability of the bit timing is necessary. In the case of CAN implementations that are designed for use without a local CPU the bit timing cannot be programmable. However, these devices allow the choice of an external oscillator in such a way that the device is adjusted to the appropriate bit rate so that the programmability is dispensable for such components. The position of the sample point, however, should be selected in common for all nodes. Therefore the bit timing of CAN devices without a local CPU must be compatible with the following definition of the bit time.

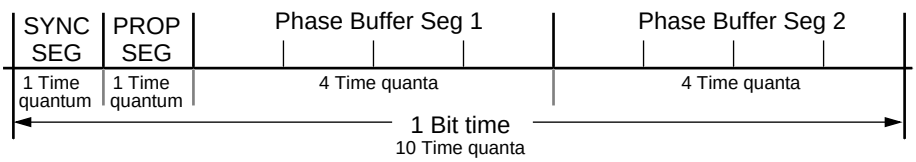


Figure 13-2 Bit timing of CAN devices without local CPU

13.9 Synchronization

13.9.1 Hard synchronization

After a Hard synchronization the internal bit time is restarted with SYNC_SEG. Thus Hard synchronization forces the edge which has caused the Hard synchronization to lie within the Synchronization segment of the restarted bit time.

13.9.2 Resynchronization jump width

As a result of resynchronization, PHASE_SEG1 may be lengthened or PHASE_SEG2 may be shortened. The amount by which the Phase buffer segments may be altered may not be greater than the Resynchronization jump width. The Resynchronization jump width is programmable between 1 and the smaller of 4 and PHASE_SEG1 Time quanta.

Clocking information may be derived from transitions from one bit value to the other. The property that only a fixed maximum number of successive bits have the same value provides the possibility of resynchronising a bus node to the bit-stream during a frame.

The maximum length between two transitions which can be used for resynchronization is 29 bit times.

13.9.3 Phase error of an edge

The Phase error of an edge is given by the position of the edge relative to SYNC_SEG, measured in Time quanta. The sign of Phase error is defined as follows:

- e < 0 if the edge lies after the Sample point of the previous bit,
- e = 0 if the edge lies within SYNC_SEG,
- e > 0 if the edge lies before the Sample point.

13.9.4 Resynchronization

The effect of a Resynchronization is the same as that of Hard synchronization, when the magnitude of the Phase error of the edge which causes the Resynchronization is less than or equal to the programmed value of the Resynchronization jump width.

When the magnitude of the Phase error is larger than the Resynchronization jump width, and if the Phase error is positive, then PHASE_SEG1 is lengthened by an amount equal to the Resynchronization jump width.

When the magnitude of the Phase error is larger than the Resynchronization jump width, and if the Phase error is negative, then PHASE_SEG2 is shortened by an amount equal to the Resynchronization jump width.

13.9.5 Synchronization rules

Hard synchronization and Resynchronization are the two forms of Synchronization. They obey the following rules:

1) Only one Synchronization within one bit time is allowed.
2) An edge will be used for Synchronization only if the value detected at the previous Sample point (previous read bus value) differs from the bus value immediately after the edge.
3) Hard synchronization is performed whenever there is a recessive to dominant edge during Bus idle.
4) All other recessive to dominant edges (and optionally dominant to recessive edges in the case of low bit rates) fulfilling the rules 1 and 2 will be used for Resynchronization with the exception that a transmitter will not perform a Resynchronization as a result of a recessive to dominant edge with a positive Phase error, if only recessive to dominant edges are used for Resynchronization.

THIS PAGE LEFT BLANK INTENTIONALLY

A
THE MOTOROLA CAN (MCAN) MODULE

A.1 Functional overview

The MCAN includes all hardware modules necessary to implement the CAN Transfer Layer, which represents the kernel of the CAN bus protocol as defined by BOSCH GmbH, the originators of the CAN specification.

Up to the message level, the MCAN is totally compatible with CAN Specification 2.0 Part A. Functional differences are related to the object layer only. Whereas a full CAN controller provides dedicated hardware for handling a set of messages, the MCAN is restricted to receiving and/or transmitting messages on a message by message basis.

The MCAN will never initiate an Overload Frame. If the MCAN starts to receive a valid message (one that passes the Acceptance Filter) and there is no receive buffer available for it then the Overrun Flag in the CPU Status register will be set. The MCAN will respond to Overload Frames generated by other CAN nodes, as required by the CAN protocol.

A diagram of the major blocks of the MCAN is shown in Figure A-1.

A.1.1 IML – interface management logic

The IML interprets the commands from the CPU, controls the allocation of the message buffers TBF, RBF0 and RBF1, and supplies interrupts and status information to the CPU via the Controller Interface Logic (CIL).

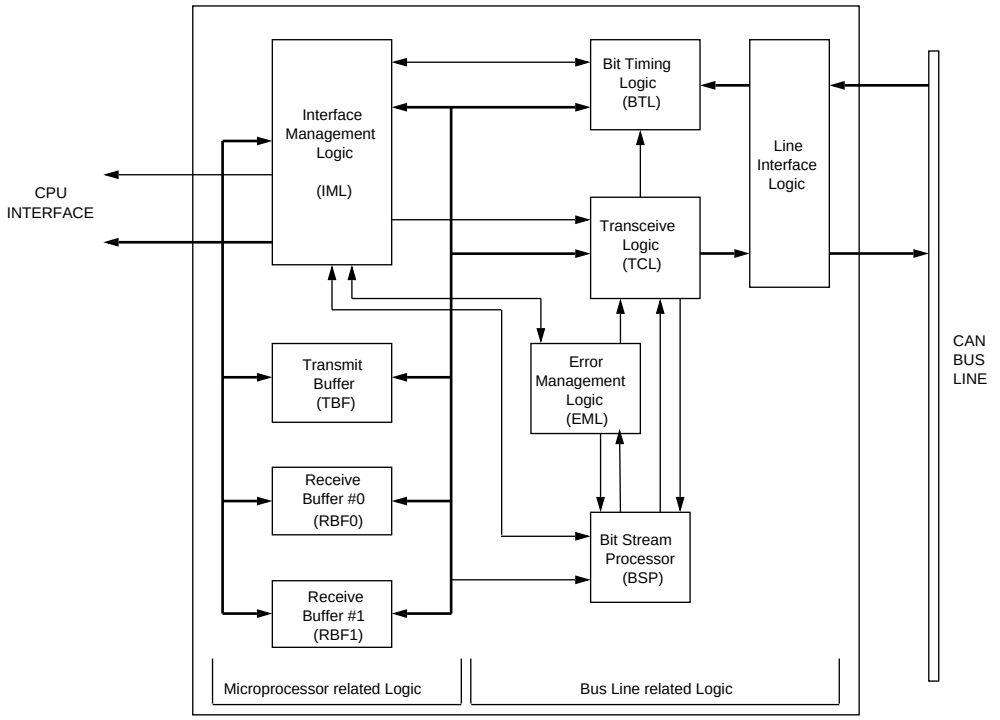


Figure A-1 MCAN module block diagram

A

A.1.2 TBF – transmit buffer

The transmit buffer is an interface between the CPU and the Bit Stream Processor (BSP) and is able to store a complete message. The buffer is written by the CPU and read by the BSP. The CPU may access this buffer whenever TRANSMIT BUFFER ACCESS is set to released. On requesting a transmission, by setting TRANSMISSION REQUEST in the CAN COMMAND REGISTER = present, TRANSMIT BUFFER ACCESS is set to locked, giving the BSP exclusive access to this buffer. The transmit buffer is released after the message transfer has been completed or aborted.

The TBF is 10 bytes long and holds the Identifier (1 byte), the Control Field (1 byte) and the Data Field (maximum length 8 bytes). The buffer is implemented as a single-ported RAM, with mutual exclusive access by the CPU and the BSP.

A.1.3 RBF – receive buffer

The receive buffer is an interface between the BSP and the CPU and stores a message received from the bus line. Once filled by the BSP and allocated to the CPU by the IML, the receive buffer cannot be used to store subsequent received messages until the CPU has acknowledged the reading of the buffer's contents. Thus, unless the CPU releases an RBF within a protocol defined time frame, future messages to be received may be lost.

To reduce the requirements on the CPU, two receive buffers (RBF0 and RBF1) are implemented. While one receive buffer is allocated to the CPU, the BSP may write to the other buffer. RBF0 and RBF1 are each 10 bytes long and hold the Identifier (1 byte), the Control Field(1 byte) and the Data Field(maximum length 8 bytes). The buffers are implemented as single-ported RAMs with mutual exclusive access from the CPU and the BSP. The BSP only writes into a receive buffer when the message being received or transmitted has an Identifier which passes the Acceptance Filter. Note that a message being transmitted will be written to the receive buffer if its Identifier passes the Acceptance Filter, as it cannot be known until after the first byte has been stored whether or not the message will lose arbitration to another transmitter.

A.1.4 BSP – bit stream processor

This is a sequencer controlling the data stream between the transmit and receive buffers (parallel data) and the bus line (serial data). The BSP also controls the Transceive Logic (TCL) and the Error Management Logic (EML) such that the processes of reception, arbitration, transmission and error signalling are performed according to the protocol and the bus rules. The BSP also provides signals to the IML indicating when a receive buffer contains a valid message and also when the transmit buffer is no longer required after a successful transmission. Note that the automatic retransmission of messages which have been corrupted by noise or other external error conditions on the bus line is effectively handled by the BSP.

A.1.5 BTL – bit timing logic

This block monitors the bus line using the INPUT COMPARATOR and handles the bus line related bit timing.

The BTL synchronizes on a recessive to dominant bus line transition at the Start of Frame (hard synchronization), and resynchronizes on further transitions during a reception of a frame (soft synchronization). A CPU programmable control bit (SPEED MODE) determines which edges are used for resynchronization.

The BTL also provides programmable time segments to compensate for the propagation delay times and phase shifts and to define the sampling time and the number of samples (1 or 3) within the bus time slot.

A.1.6 TCL – transceive logic

The TCL is a generic term for a group of logic elements consisting of a programmable output driver, bit stuff logic, CRC logic and the transmit shift register. The coordination of these components is controlled by the BSP.

A.1.7 EML – error management logic

The EML is responsible for the error confinement of the MCAN Module. It receives notification of errors from the BSP and then informs the BSP, TCL and IML about error statistics.

Note: The BSP, TCL, BTL and EML together are described collectively as the bus line related logic. Similarly, the IML, CIL, TBF, RBF0 and RBF1 are described as microprocessor related logic.

A

A.2 MCAN interface

To be able to function as a serial communication interface, the MCAN Module itself has to be supplemented by an additional module, the Controller Interface Unit (CIL).

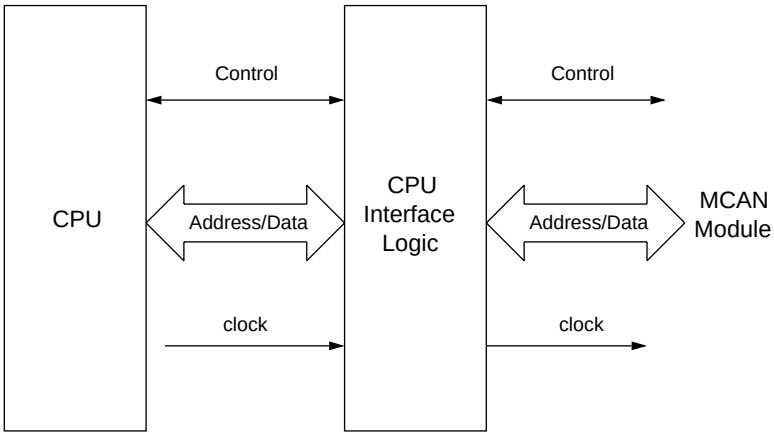


Figure A-2 Block diagram of the MCAN interface

A.2.1 CIL – controller interface unit

The CIL links the MCAN Module to the CPU. It connects the CPU buses to the 8-bit MCAN buses. It also generates internal MCAN control signals from internal CPU signals.

The CIL receives the various signals for wake-up/sleep inhibit from the rest of the circuit and the go to sleep signal from the IML

A

A.2.2 Address allocation

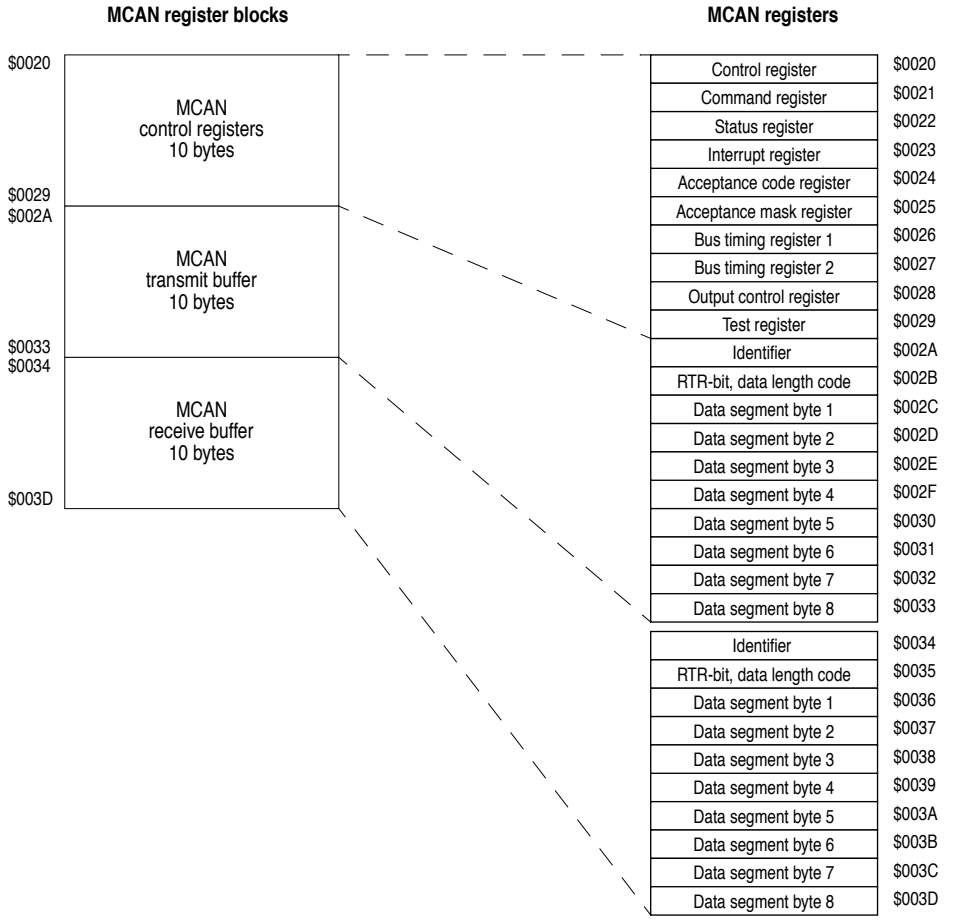


Figure A-3 MCAN module memory map

A

A.2.3 Control registers

The interchange of commands, status and control signals between the CPU and the MCAN Module takes place via the control registers. The layout of these registers is shown in Figure A-3.

Note: The acceptance code register, acceptance mask register, bus timing register 0, bus timing register 1 and the output control register are only accessible when the RESET REQUEST bit in the MCAN control register is set to present. It is not foreseen that these registers will be referenced again after the initial reset sequence.

Table A-1 Control registers

Register	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Control (CCNTRL)	$0020	MODE	SPD		OIE	EIE	TIE	RIE	RR
Command (CCOM)	$0021	RX0	RX1	COMP-SEL	SLEEP	COS	RRB	AT	TR
Status (CSTAT)	$0022	BS	ES	TS	RS	TCS	TBA	DO	RBS
Interrupt (CINT)	$0023				WIF	OIF	EIF	TIF	RIF
Acceptance code (CACC)	$0024	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Acceptance mask (CACM)	$0025	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Bus timing 0 (CBTO)	$0026	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
Bus timing 1 (CBT1)	$0027	SAMP	TSEG-22	TSEG-21	TSEG-20	TSEG13	TSEG-12	TSEG-11	TSEG-10
Output control (COCNTRL)	$0028	OCT-P1	OCT-N1	OCPOL1	OCT-P0	OCT-N0	OCPOL0	OCM1	OCM0

A.2.4 MCAN control register (CCNTRL)

This register may be read or written to by the MCU; only the RR bit is affected by the MCAN.

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	Reset condition	State on reset
MCAN control (CCNTRL)	$0020	MODE	SPD		OIE	EIE	TIE	RIE	RR	External reset	0u - u uuu1
										RR bit set	0u - u uuu1

MODE — Undefined mode

This bit must never be set by the CPU as this would result in the transmit and receive buffers being mapped out of memory. The bit is cleared on reset, and should be left in this state for normal operation.

A

SPD — Speed mode

1 (set) – Slow – Bus line transitions from both recessive to dominant and from dominant to recessive will be used for resynchronization.

0 (clear) – Fast – Only transitions from recessive to dominant will be used for resynchronization.

OIE — Overrun interrupt enable

1 (set) – Enabled – The CPU will get an interrupt request whenever the Overrun Status bit gets set.

0 (clear) – Disabled – The CPU will get no overrun interrupt request.

EIE — Error interrupt enable

1 (set) – Enabled – The CPU will get an interrupt request whenever the error status or bus status bits in the CSTAT register change.

0 (clear) – Disabled – The CPU will get no error interrupt request.

TIE — Transmit interrupt enable

1 (set) – Enabled – The CPU will get an interrupt request whenever a message has been successfully transmitted, or when the transmit buffer is accessible again following an ABORT command.

0 (clear) – Disabled – The CPU will get no transmit interrupt request.

RIE — Receive interrupt enable

1 (set) – Enabled – The CPU will get an interrupt request whenever a message has been received free of errors.

0 (clear) – Disabled – The CPU will get no receive interrupt request.

RR — Reset request

When the MCAN detects that RR has been set it aborts the current transmission or reception of a message and enters the reset state. A reset request may be generated by either an external reset or by the CPU or by the MCAN. The RR bit can be cleared only by the CPU. After the RR bit has been cleared, the MCAN will start normal operation in one of two ways. If RR was generated by an external reset or by the CPU, then the MCAN starts normal operation after the first occurrence of 11 recessive bits. If, however, the RR was generated by the MCAN due to the BS bit being set (see Section A.2.6) the MCAN waits for 128 occurrences of 11 recessive bits before starting normal operation.

A reset request should not be generated by the CPU during a message transmission. Ensure that a message is not being transmitted as follows:

if TCS in CSTAT is clear – set AT in CCOM (use STA or STX), read CSTAT.

if TS in CSTAT is set – wait until TS is clear.

Note that a CPU-generated reset request does not change the values in the transmit and receive error counters.

1 (set) – Present – MCAN will be reset.

0 (clear) – Absent – MCAN will operate normally.

Note: The following registers may only be accessed when reset request = present: CACC, CACM, CBT0, CBT1, and COCNTRL.

A.2.5 MCAN command register (CCOM)

This is a write only register; a read of this location will always return the value $FF.

This register may be written only when the RR bit in CCNTRL is clear.

Do not use read-modify-write instructions on this register (e.g. BSET, BCLR).

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	Reset condition	State on reset
MCAN command (CCOM)	$0021	RX0	RX1	COMPSEL	SLEEP	COS	RRB	AT	TR	External reset	00u0 0000
										RR bit set	00u0 0000

RX0 — Receive pin 0 (passive) (Refer to Figure A-6)

1 (set) – VDD/2 will be connected to the input comparator. The RX0 pin is disconnected.

0 (clear) – The RX0 pin will be connected to the input comparator. VDD/2 is disconnected.

RX1 — Receive pin 1 (passive) (Refer to Figure A-6)

1 (set) – VDD/2 will be connected to the input comparator. The RX1 pin is disconnected.

0 (clear) – The RX1 pin will be connected to the input comparator. VDD/2 is disconnected.

Note: If both RX0 and RX1 are set, or both are clear, then neither of the RX pins will be disconnected.

A

COMPSEL — Comparator selector

1 (set) – RX0 and RX1 will be compared with VDD/2 during sleep mode (see Figure A-6).

0 (clear) – RX0 will be compared with RX1 during sleep mode.

SLEEP — Go to sleep

1 (set) – Sleep – The MCAN will go into sleep mode, as long as there are no interrupts pending and there is no activity on the bus. Otherwise the MCAN will issue a wake-up interrupt.

0 (clear) – Wake-up – The MCAN will function normally. If SLEEP is cleared by the CPU then the MCAN will waken up, but will not issue a wake-up interrupt.

Note: If SLEEP is set during the reception or transmission of a message, the MCAN will generate an immediate wake-up interrupt. (This allows for a more orthogonal software implementation on the CPU.) This will have no effect on the transfer layer, i.e. no message will be lost or corrupted.

The CAF flag in the EEPROM control register (or Port Configuration register for HC(7)05X4), indicates whether or not sleep mode was entered successfully.

A node that was sleeping and has been awakened by bus activity will not be able to receive any messages until its oscillator has started and it has found a valid end of frame sequence (11 recessive bits). The designer must take this into consideration when planning to use the sleep command.

COS — Clear overrun status

1 (set) – This clears the read-only data overrun status bit in the CSTAT register (see Section A.2.6). It may be written at the same time as RRB.

0 (clear) – No action.

RRB — Release receive buffer

When set this releases the receive buffer currently attached to the CPU, allowing the buffer to be reused by the MCAN. This may result in another message being received, which could cause another receive interrupt request (if RIE is set). This bit is cleared automatically when a message is received, i.e. when the RS bit (see Section A.2.6) becomes set.

1 (set) – Released – receive buffer is available to the MCAN.

0 (clear) – No action.

AT — Abort transmission

When this bit is set a pending transmission will be cancelled if it is not already in progress, allowing the transmit buffer to be loaded with a new (higher priority) message when the buffer is released. If the CPU tries to write to the buffer when it is locked, the information will be lost without being signalled. The status register can be checked to see if transmission was aborted or is still in progress.

1 (set) – Present – Abort transmission of any pending messages.

0 (clear) – No action.

TR — Transmission request

1 (set) – Present – Depending on the transmission buffer's content, a data frame or a remote frame will be transmitted.

0 (clear) – No action. This will not cancel a previously requested transmission; the abort transmission command must be used to do this.

A.2.6 MCAN status register (CSTAT)

This is a read only register; only the MCAN can change its contents.

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	Reset condition	State on reset
MCAN status (CSTAT)	$0022	BS	ES	TS	RS	TCS	TBA	DO	RBS	External reset	0000 1100
										RR bit set	uu00 1100

BS — Bus status

This bit is set (off-bus) by the MCAN when the transmit error counter reaches 256. The MCAN will then set RR and will remain off-bus until the CPU clears RR again. At this point the MCAN will wait for 128 successive occurrences of a sequence of 11 recessive bits before clearing BS and resetting the read and write error counters. While off-bus the MCAN does not take part in bus activities.

1 (set) – Off-bus – The MCAN is not participating in bus activities.

0 (clear) – On-bus – The MCAN is operating normally.

ES — Error status

1 (set) – Error – Either the read or the write error counter has reached the CPU warning limit of 96.

0 (clear) – Neither of the error counters has reached 96.

TS — Transmit status

1 (set) – Transmit – The MCAN has started to transmit a message.

0 (clear) – Idle – If the receive status bit is also clear then the MCAN is idle; otherwise it is in receive mode.

RS — Receive status

1 (set) – Receive – The MCAN entered receive mode from idle, or by losing arbitration during transmission.

0 (clear) – Idle – If the transmit status bit is also clear then the MCAN is idle; otherwise it is in transmit mode.

TCS — Transmission complete status

This bit is cleared by the MCAN when TR becomes set. When TCS is set it indicates that the last requested transmission was successfully completed. If, after TCS is cleared, but before transmission begins, an abort transmission command is issued then the transmit buffer will be released and TCS will remain clear. TCS will then only be set after a further transmission is both requested and successfully completed.

1 (set) – Complete – Last requested transmission successfully completed.

0 (clear) – Incomplete – Last requested transmission not complete.

TBA — Transmit buffer access

When clear, the transmit buffer is locked and cannot be accessed by the CPU. This indicates that either a message is being transmitted, or is awaiting transmission. If the CPU writes to the transmit buffer while it is locked, then the bytes will be lost without this being signalled.

1 (set) – Released – The transmit buffer may be written to by the CPU.

0 (clear) – Locked – The CPU cannot access the transmit buffer.

DO — Data overrun

This bit is set when both receive buffers are full and there is a further message to be stored. In this case the new message is dropped, but the internal logic maintains the correct protocol. The MCAN does not receive the message, but no warning is sent to the transmitting node. The MCAN clears DO when the CPU sets the COS bit in the CCOM register.

Note that data overrun can also be caused by a transmission, since the MCAN will temporarily store an outgoing frame in a receive buffer in case arbitration is lost during transmission.

1 (set) – Overrun – Both receive buffers were full and there was another message to be stored.

0 (clear) – Normal operation.

RBS — Receive buffer status

This bit is set by the MCAN when a new message is available. When clear this indicates that no message has become available since the last RRB command. The bit is cleared when RRB is set. However, if the second receive buffer already contains a message, then control of that buffer is given to the CPU and RBS is immediately set again. The first receive buffer is then available for the next incoming message from the MCAN.

1 (set) – Full – A new message is available for the CPU to read.

0 (clear) – Empty – No new message is available.

A.2.7 MCAN interrupt register (CINT)

All bits of this register are read only; all are cleared by a read of the register.

This register must be read in the interrupt handling routine in order to enable further interrupts.

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	Reset condition	State on reset
MCAN interrupt (CINT)	$0023				WIF	OIF	EIF	TIF	RIF	External reset	- - - 0 0000
										RR bit set	- - - u 0u00

WIF — Wake-up interrupt flag

If the MCAN detects bus activity whilst it is asleep, it clears the SLEEP bit in the CCOM register; the WIF bit will then be set. WIF is cleared by reading the MCAN interrupt register (CINT), or by an external reset.

1 (set) – MCAN has detected activity on the bus and requested wake-up.

0 (clear) – No wake-up interrupt has occurred.

OIF — Overrun interrupt flag

When OIE is set then this bit will be set when a data overrun condition is detected. Like all the bits in this register, OIF is cleared by reading the register, or when reset request is set.

1 (set) – A data overrun has been detected.

0 (clear) – No data overrun has occurred.

EIF — Error interrupt flag

When EIE is set then this bit will be set by a change in the error or bus status bits in the MCAN status register. Like all the bits in this register, EIF is cleared by reading the register, or by an external reset.

1 (set) – There has been a change in the error or bus status bits in CSTAT.

0 (clear) – No error interrupt has occurred.

TIF — Transmit interrupt flag

The TIF bit is set at the end of a transmission whenever both the TBA and TIE bits are set. Like all the bits in this register, TIF is cleared by reading the register, or when reset request is set.

1 (set) – Transmission complete, the transmit buffer is accessible.

0 (clear) – No transmit interrupt has occurred.

RIF — Receive interrupt flag

The RIF bit is set by the MCAN when a new message is available in the receive buffer, and the RIE bit in CCNTRL is set. At the same time RBS is set. Like all the bits in this register, RIF is cleared by reading the register, or when reset request is set.

1 (set) – A new message is available in the receive buffer.

0 (clear) – No receive interrupt has occurred.

A.2.8 MCAN acceptance code register (CACC)

On reception each message is written into the current receive buffer. The MCU is only signalled to read the message however, if it passes the criteria in the acceptance code and acceptance mask registers (accepted); otherwise, the message will be overwritten by the next message (dropped).

Note: This register can only be accessed when the reset request bit in the CCNTRL register is set.

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
MCAN acceptance code (CACC)	$0024	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Undefined

AC7 – AC0 — Acceptance code bits

AC7 – AC0 comprise a user defined sequence of bits with which the 8 most significant bits of the data identifier (ID10 – ID3) are compared. The result of this comparison is then masked with the acceptance mask register. Once a message has passed the acceptance criterion the respective identifier, data length code and data are sequentially stored in a receive buffer, providing there is one free. If there is no free buffer, the data overrun condition will be signalled.

On acceptance the receive buffer status bit is set to full and the receive interrupt bit is set (provided RIE = enabled).

A.2.9 MCAN acceptance mask register (CACM)

The acceptance mask register specifies which of the corresponding bits in the acceptance code register are relevant for acceptance filtering.

Note: This register can only be accessed when the reset request bit in the CCNTRL register is set.

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
MCAN acceptance mask (CACM)	$0025	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	Undefined

AM0 – AM7 — Acceptance mask bits

When a particular bit in this register is clear this indicates that the corresponding bit in the acceptance code register must be the same as its identifier bit, before a match will be detected. The message will be accepted if all such bits match. When a bit is set, it indicates that the state of the corresponding bit in the acceptance code register will not affect whether or not the message is accepted.

1 (set) – Ignore corresponding acceptance code register bit.

0 (clear) – Match corresponding acceptance code register and identifier bits.

A.2.10 MCAN bus timing register 0 (CBT0)

Note: This register can only be accessed when the reset request bit in the CCNTRL register is set.

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
MCAN bus timing 0 (CBT0)	$0026	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Undefined

SJW1, SJW0 — Synchronization jump width bits

The synchronization jump width defines the maximum number of system clock (t_{SCL}) cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see Table A-2).

A

Table A-2 Synchronization jump width

SJW1	SJW0	Synchronization jump width
0	0	1 t_{SCL} cycle
0	1	2 t_{SCL} cycles
1	0	3 t_{SCL} cycles
1	1	4 t_{SCL} cycles

BRP5 – BRP0 — Baud rate prescaler bits

These bits determine the MCAN system clock cycle time (t_{SCL}), which is used to build up the individual bit timing, according to Table A-3 and the formula in Figure A-4.

Table A-3 Baud rate prescaler

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

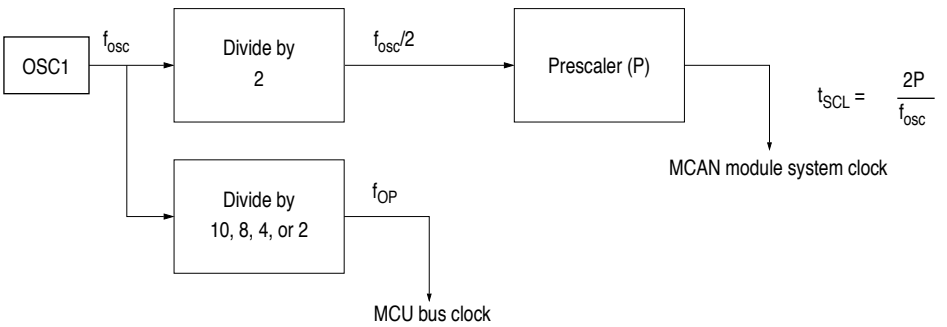


Figure A-4 Oscillator block diagram

A.2.11 MCAN bus timing register 1 (CBT1)

This register can only be accessed when the reset request bit in the CCNTRL register is set.

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
MCAN bus timing 1 (CBT1)	$0027	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10	Undefined

SAMP — Sampling

This bit determines the number of samples of the serial bus to be taken per bit time. When set three samples per bit are taken. This sample rate gives better rejection of noise on the bus, but introduces a one bit delay to the bus sampling. For higher bit rates SAMP should be cleared, which means that only one sample will be taken per bit.

1 (set) – Three samples per bit.

0 (clear) – One sample per bit.

TSEG22 – TSEG10 — Time segment bits

Time segments within the bit time fix the number of clock cycles per bit time, and the location of the sample point.

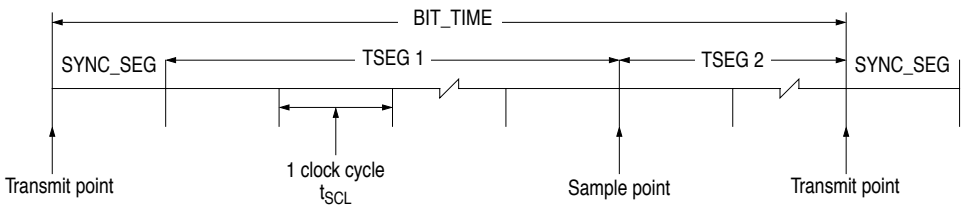


Figure A-5 Segments within the bit time

SYNC_SEG System expects transitions to occur on the bus during this period.

Transmit point A node in transmit mode will transfer a new value to the MCAN bus at this point.

Sample point A node in receive mode will sample the bus at this point. If the three samples per bit option is selected then this point marks the position of the third sample.

Time segment 1 (TSEG1) and time segment 2 (TSEG2) are programmable as shown in Table A-4.

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of bus clock cycles (t_{SCL}) per bit (as shown above).

A

Table A-4 Time segment values

TSEG13	TSEG12	TSEG11	TSEG10	Time segment 1
0	0	0	1	2 t_{SCL} cycles
0	0	1	0	3 t_{SCL} cycles
0	0	1	1	4 t_{SCL} cycles
.	.	.	.	.
.	.	.	.	.
1	1	1	1	16 t_{SCL} cycles

TSEG22	TSEG21	TSEG20	Time segment 2
0	0	1	2 t_{SCL} cycles
.	.	.	.
.	.	.	.
1	1	1	8 t_{SCL} cycles

Calculation of the bit time

$$BIT\_TIME = SYNC\_SEG + TSEG1 + TSEG2$$

Note: TSEG2 must be at least 2 t_{SCL}, i.e. the configuration bits must not be 000. (If three samples per bit mode is selected then TSEG2 must be at least 3 t_{SCL}.)

TSEG1 must be at least as long as TSEG2.

The synchronization jump width (SJW) may not exceed TSEG2, and must be at least t_{SCL} shorter than TSEG1 to allow for physical propagation delays.

i.e. in terms of t_{SCL}:

$SYNC\_SEG = 1$

$TSEG1 \geq SJW + 1$

$TSEG1 \geq TSEG2$

$TSEG2 \geq SJW$

and $TSEG2 \geq 2$ (SAMP = 0)

or $TSEG2 \geq 3$ (SAMP = 1)

These boundary conditions result in minimum bit times of 5 t_{SCL}, for one sample, and 7 t_{SCL}, for three samples per bit.

A

A.2.12 MCAN output control register (COCNTRL)

This register allows the setup of different output driver configurations under software control. The user may select active pull-up, pull-down, float or push-pull output.

Note: This register can only be accessed when the reset request bit in the CCNTRL register is set.

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
MCAN output control (COCNTRL)	$0028	OCTP1	OCTN1	OCPOL1	OCTP0	OCTN0	OCPOL0	OCM1	OCM0	Undefined

OCM1 and OCM0 — Output control mode bits

The values of these two bits determine the output mode, as shown in Table A-5.

Table A-5 Output control modes

OCM1	OCM0	Function
0	0	Biphase mode
0	1	Not used
1	0	Normal mode 1 Bit stream transmitted on both TX0 and TX1
1	1	Normal mode 2 TX0 - bit sequence TX1 - bus clock (t_{xclk})

Note: The transmit clock (t_{xclk}) is used to indicate the end of the bit time and will be high during the SYNC_SEG.

For all the following modes of operation, a dominant bit is internally coded as a zero, a recessive as a one. The other output control bits are used to determine the actual voltage levels transmitted to the MCAN bus for dominant and recessive bits.

Biphase mode

If the CAN modules are isolated from the bus lines by a transformer then the bit stream has to be coded so that there is no resulting dc component. There is a flip-flop within the MCAN that keeps the last dominant configuration; its direct output goes to TX0 and its complement to TX1. The flip-flop is toggled for each dominant bit; dominant bits are thus sent alternately on TX0 and TX1;

i.e. the first dominant bit is sent on TX0, the second on TX1, the third on TX0 and so on. During recessive bits, all output drivers are deactivated (i.e. high impedance).

Normal mode 1

In contrast to biphase mode the bit representation is time invariant and not toggled.

Normal mode 2

For the TX0 pin this is the same as normal mode 1, however the data stream to TX1 is replaced by the transmit clock. The rising edge of the transmit clock marks the beginning of a bit time. The clock pulse will be t_{SCL} long.

Other output control bits

The other six bits in this register control the output driver configurations, to determine the format of the output signal for a given data value (see Figure A-6).

OCTP0/1 – These two bits control whether the P-type output control transistors are enabled.

OCTN0/1 – These two bits control whether the N-type output control transistors are enabled.

OCPOL0/1 – These two bits determine the driver output polarity for each of the MCAN bus lines (TX0, TX1).

TP0/1 and TN0/1 – These are the resulting states of the output transistors.

TD – This is the internal value of the data bit to be transferred across the MCAN bus. (A zero corresponds to a dominant bit, a one to a recessive.)

The actions of these bits in the output control register are as shown in Table A-6.

A

Table A-6 MCAN driver output levels

Mode	TD	OCPOLi	OCTPi	OCTNi	TPi	TNi	TXi output level
Float	0	0	0	0	Off	Off	Float
	1	0	0	0	Off	Off	Float
	0	1	0	0	Off	Off	Float
	1	1	0	0	Off	Off	Float
Pull-down	0	0	0	1	Off	On	Low
	1	0	0	1	Off	Off	Float
	0	1	0	1	Off	Off	Float
	1	1	0	1	Off	On	Low
Pull-up	0	0	1	0	Off	Off	Float
	1	0	1	0	On	Off	High
	0	1	1	0	On	Off	High
	1	1	1	0	Off	Off	Float
Push-pull	0	0	1	1	Off	On	Low
	1	0	1	1	On	Off	High
	0	1	1	1	On	Off	High
	1	1	1	1	Off	On	Low

A.2.13 Transmit buffer identifier register (TBI)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Transmit buffer identifier (TBI)	$002A	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	Undefined

ID10 – ID3 — Identifier bits

The identifier consists of 11 bits (ID10 – ID0). ID10 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. The three least significant bits are contained in the TRTDL register. The seven most significant bits must not all be recessive.

A.2.14 Remote transmission request and data length code register (TRTDL)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
RTR and data length code (TRTDL)	$002B	ID2	ID1	ID0	RTR	DLC3	DLC2	DLC1	DLC0	Undefined

ID2 – ID0 — Identifier bits

These bits contain the least significant bits of the transmit buffer identifier.

RTR — Remote transmission request

1 (set) – A remote frame will be transmitted.

0 (clear) – A data frame will be transmitted.

DLC3 – DLC0 — Data length code bits.

The data length code contains the number of bytes (data byte count) of the respective message. At transmission of a remote frame, the data length code is ignored, forcing the number of bytes to be 0. The data byte count ranges from 0 to 8 for a data frame. Table A-7 shows the effect of setting the DLC bits.

Table A-7 Data length codes

Data length code				Data byte count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

A

A.2.15 Transmit data segment registers (TDS) 1 – 8

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Transmit data segment (TDS)	$002C – $0033	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined

DB7 – DB0 — data bits

These data bits in the eight data segment registers make up the bytes of data to be transmitted. The number of bytes to be transmitted is determined by the data length code.

A.2.16 Receive buffer identifier register (RBI)

The layout of this register is identical to the TBI register (see Section A.2.13).

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Receive buffer identifier (RBI)	$0034	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	Undefined

(Note that there are actually two receive buffer register sets, but switching between them is handled internally by the MCAN.)

A.2.17 Remote transmission request and data length code register (RRTDL)

The layout of this register is identical to the TRTDL register (see Section A.2.14).

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
RTR and data length code (RRTDL)	$0035	ID2	ID1	ID0	RTR	DLC3	DLC2	DLC1	DLC0	Undefined

A.2.18 Receive data segment registers (RDS) 1 – 8

The layout of these registers is identical to the TDSx registers (see Section A.2.15).

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Receive data segment (RDS)	$0036 – $003D	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Undefined

(Note that there are actually two receive buffer register sets, but switching between them is handled internally by the MCAN.)

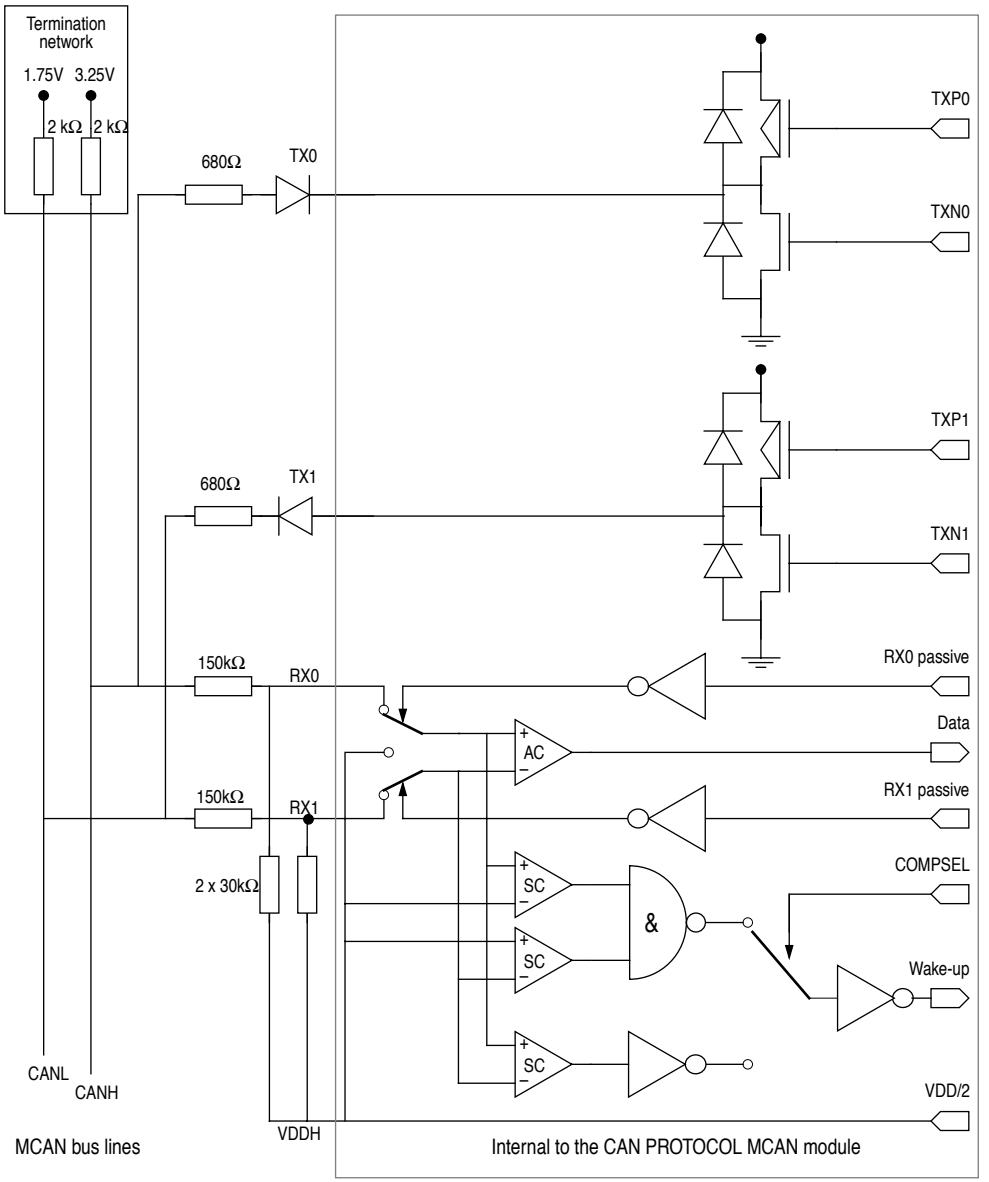


Figure A-6 A typical physical interface between the MCAN and the MCAN bus lines

A.2.19 Organization of buffers

Further details on these registers will be found in the appropriate device data sheet.

Table 1-8 MCAN data buffers

Register	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Transmit Buffer Identity (TBI)	$002A	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
RTRbit, Data length code (TRTDL)	$002B	ID2	ID1	ID0	RTR	DLC3	DLC2	DLC1	DCL0
Transmit Data segment 1 (TDS1)	$002C	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Transmit Data segment 2 (TDS2)	$002D	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Transmit Data segment 3 (TDS3)	$002E	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Transmit Data segment 4 (TDS4)	$002F	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Transmit Data segment 5 (TDS5)	$00030	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Transmit Data segment 6 (TDS6)	$00031	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Transmit Data segment 7 (TDS7)	$00032	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Transmit Data segment 8 (TDS8)	$00033	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Receive Buffer Identity (RBI)	$00034	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
RTRbit, Data length code (RRTDL)	$00035	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
Receive Data segment 1 (RDS1)	$00036	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Receive Data segment 2 (RDS2)	$00037	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Receive Data segment 3 (RDS3)	$00038	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Receive Data segment 4 (RDS4)	$00039	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Receive Data segment 5 (RDS5)	$0003A	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Receive Data segment 6 (RDS6)	$0003B	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Receive Data segment 7 (RDS7)	$0003C	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Receive Data segment 8 (RDS8)	$0003D	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

THIS PAGE LEFT BLANK INTENTIONALLY

B
TOUCAN

B.1 Introduction

The TOUCAN Module is designed in a modular structure for use in Motorola's Modular Microcontroller Family (MMF) or for the RISC family. The TOUCAN module is a communication controller implementing the CAN protocol. A general working knowledge of the IMB3 signals and bus control is assumed in the writing of this document.

The TOUCAN supports 2 methods of Interrupt architechture (MODULAR and RISC), which can be chosen by a mask programming option (IRQ_PLUG).

The TOUCAN supports 2 methods of berr architechture (MODULAR and RISC), which can be chosen by a mask programming option (BERR_PLUG).

The CAN protocol was primarily, but not exclusively, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field, i.e. real-time processing, cost-effectiveness, required bandwidth and reliable operation in the EMI environment of a vehicle.

B.2 TOUCAN module features

- Motorola IMB3-Family Modular Architecture
- Full implementation of the CAN Protocol Specification, Version 2.0
  - Standard data and remote frames (up to 109 bits long)
  - Extended data and remote frames (up to 127 bits long)
  - 0-8 bytes data length
  - Programmable bit rate up to 1Mbit/sec
- 16 message buffers (MBs) of 0-8 bytes data length; each of these can be configured as Rx or Tx and can support standard and extended messages
- Content-related addressing

- No read/write semaphores
- Three programmable mask registers: Global (for MBs 0-13), Special for MB14 and Special for MB15
- Programmable ‘transmit-first’ scheme based on 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 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 wake-up on bus activity

A block diagram describing the various sub-modules of the TOUCAN module is shown in Figure B-1. Each sub-module is described in detail in subsequent sections.

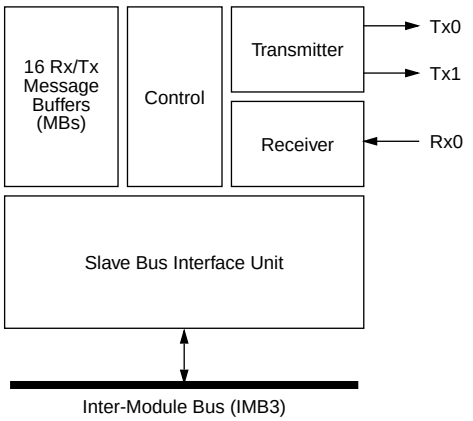


Figure B-1 TOUCAN block diagram and pinout

B.3 External Pins

The TOUCAN module interface to the CAN bus is composed of 3pins: Tx0 and Tx1, which are the serial transmitted data, and Rx0, which are the serial received data. Dominant state is defined as Tx0=0 and Tx1=1. The opposite state is defined as recessive state. The same applies respectively to the Rx0 pin.

The minimum set of pins that may be bonded out in a chip is the Tx0 and Rx0 pins only. Such a configuration is based on the use of an external transceiver to interface to the CAN bus.

B.4 The CAN system

A typical CAN system is shown in Figure B-2.

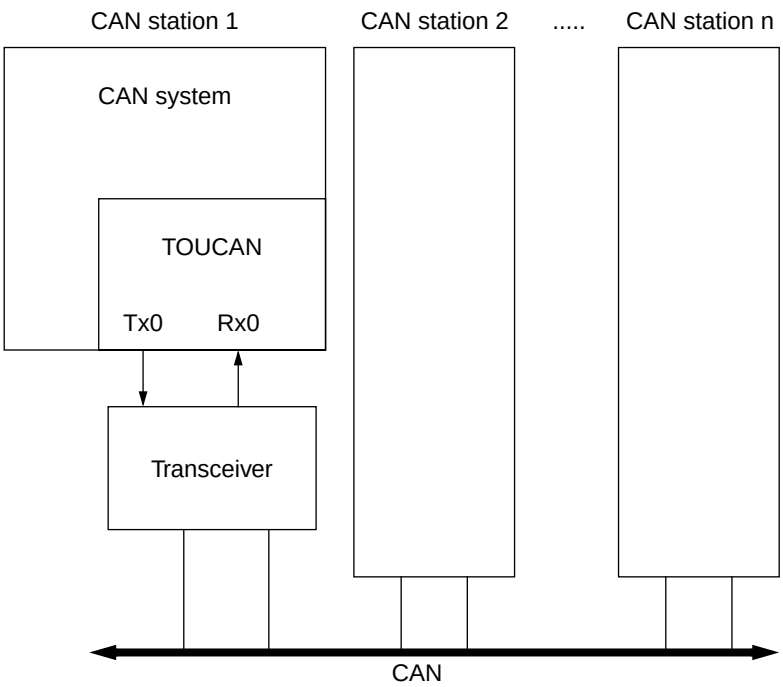


Figure B-2 Typical CAN system

Each CAN station is physically connected to the CAN bus through a transceiver. The transceiver is capable of driving the large current needed for the CAN bus and has current protection against a defective CAN bus or defective stations.

B.5 Message buffer structure

Figure B-3 describes the message buffer structure.

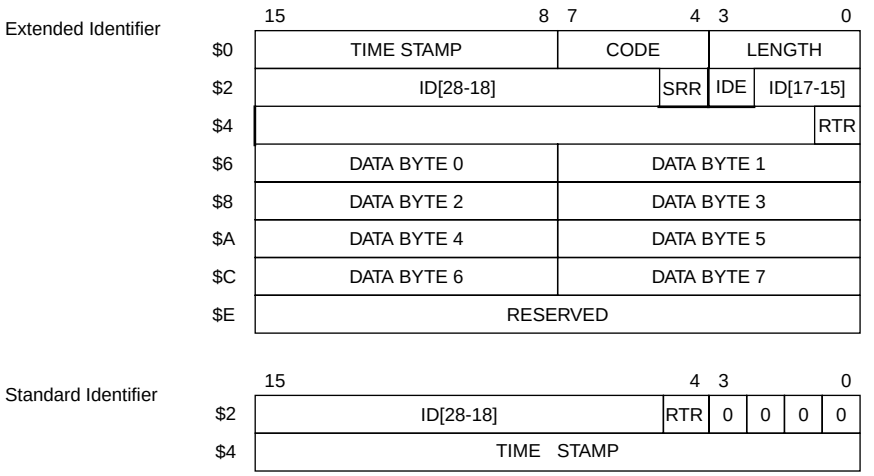


Figure B-3 Message buffer structure

B.6 Common fields to extended and standard format frames

B.6.1 CODE

The CODE bits are described for Rx buffers and Tx buffers in Table B-1 and Table B-2 respectively.

Table B-1 Message buffer code for Rx buffers

Rx code *before* Rx new frame	Description	Rx code *after* Rx new frame	Comment
0000	Not active: MB is not active.	—	
0100	Empty: MB is active and empty.	0010	
0010	Full: MB is full.	0110	If a CPU read occurs before the new frame, new Rx code is 0010.
0110	Overrun: Second frame was received into a full buffer before the CPU read the first one.	0110	
0XY1(1)	Busy: MB is now being filled with a new receive frame. This condition will be cleared within 20 cycles.	0010	An empty buffer was filled (XY was 10).
		0110	A full/overrun buffer was filled (Y was 1).

(1) Note that for Tx MBs (see Table B-2) upon read, the BUSY bit should be ignored.

Table B-2 Message buffer code for Tx buffers

RTR	Initial Tx code	Code after successful transmission	Description
X	1000	—	MB not ready for transmission.
0	1100	1000	Data frame to be transmitted once, unconditionally.
1	1100	0100	Remote frame to be transmitted once, and MB becomes a receive MB for data frames.
0	1010(1)	1010	Data frame to be transmitted only as a response to remote frame.
0	1110	1010	Data frame to be transmitted once, unconditionally. It then becomes an MB of the previous type.

(1) Note that when a matching remote request frame is detected, the code for such an MB becomes '1110'.

B.6.2 LENGTH (receive mode)

This is the length (in bytes) of the Rx data stored in offset $6-$D of this buffer. This field is written by the TOUCAN module, copied from the Data Length Code (DLC) field of the received frame.

B.6.3 LENGTH (transmit mode)

This is the length (in bytes) of the data to be transmitted, located in offset $6-$D of this buffer. This field is written by the CPU and is also the DLC field value. Note that if RTR=1, the frame is a remote frame and will be transmitted without data field, regardless of the length field.

B.6.4 DATA BYTE 0..7

Up to eight data bytes can be stored for a frame. For Rx frames, the data is stored as it is received from the line; for Tx frames the CPU prepares the data field to be transmitted within the frame.

B.6.5 RESERVED

This word entry (16 bits) should not be accessed by the CPU.

B.7 Fields for extended format frames

B.7.1 TIME STAMP

This is a copy of the high byte of the free-running timer, which was captured at the beginning of the identifier field of this buffer's frame on the CAN bus.

B.7.2 ID[28-18, 17-15]

These are the 14 MSBs of the extended identifier, located in the ID_HIGH word of the message buffer.

B.7.3 SRR — Substitute remote request

Fixed recessive bit, used only in extended format. Should be set to '1' by the user for Tx buffers, and will be stored as received on the CAN bus, for Rx buffers.

B.7.4 IDE — ID Extended

This field should be set to '1' if extended format frame should be used. If this bit is set to '0', refer to Section B.8.

B.7.5 ID[14-0]

Bits 14-0 of the Extended Identifier, located in the ID_LOW word of the message buffer.

B.7.6 RTR — Remote transmission request

This bit is the least significant bit of the ID_LOW word of the message buffer.

1 (set) – This is a remote frame.

0 (clear) – This is a data frame.

B.8 Fields for standard format frames

B.8.1 TIME STAMP

The ID_LOW word, which is not required for standard format, is used in standard format buffer to store the value of the free-running timer captured at the beginning of the Identifier field of this buffer's frame on the CAN bus.

B.8.2 ID[28-18]

Bits 28-18 of the Identifier, located in the ID_HIGH word of the message buffer. Note that the four least significant bits in the Standard Identifier (bits 3-0 in ID_HIGH word) must be set to 0000 to ensure proper operation of TOUCAN.

B.8.3 RTR — Remote transmission request

This bit is located in the ID_HIGH word of the message buffer. Its operation is as follows.

1 (set) – This is a remote frame

0 (clear) – This is a data frame

B.8.4 RTR/SRR bit treatment

If TOUCAN transmits this bit as '1' and receives it as '0', then it interprets it as arbitration loss; if this bit is transmitted as '0', then received as a '1', TOUCAN treats it as a bit error; if the value received matches the value transmitted, it is considered to be a successful bit transmission.

B.9 Functional overview

The TOUCAN module is flexible in that each one of its 16 message buffers can be assigned either as a Tx buffer or an Rx buffer. Each message buffer is also assigned an interrupt flag bit, to indicate successful completion of transmission or receipt. Note that for both processes, the first CPU action in preparing a message buffer should be to deactivate it by setting its code field to the proper value (refer to Table B-1). This requirement is mandatory to ensure proper operation.

B.10 Transmit process

The CPU prepares/changes a message buffer for transmission by executing the following steps:

i) Writing the control/status word to hold inactive Tx message buffer (code = 1000)

ii) Writing the ID_HIGH and ID_LOW words

iii) Writing the data bytes

iv) Writing the control/status word (active code, length)

Note: The first and last steps are mandatory.

Starting with step iv), this message buffer will participate in the internal arbitration process, which takes place every time the CAN bus is sensed as being free by the receiver or at the inter-frame space, and there is at least one message buffer ready for transmission. This internal arbitration process is intended to select the message buffer from which the next frame is transmitted.

When this process is over, and there is a winner message buffer for transmission, the frame is transferred to the serial message buffer (SMB) for transmission (Move Out).

While transmitting, TOUCAN transmits up to eight data bytes, even if the DLC is bigger in value.

At the end of the 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 IFLAGH/IFLAGL register.

B.11 Receive process

The CPU prepares/changes a new message buffer for frame receipt by executing the following steps:

i) Writing the control/status word to hold inactive Rx message buffer (code = 0000)

ii) Writing the ID-HIGH and ID_LOW words

iii) Writing the control/status word to mark a receive message buffer is active and empty

Note: The first and last steps are mandatory.

Starting with step iii), this message buffer is an active Rx buffer and will participate in the internal matching process, which takes place every time the receiver receives an error-free frame. In this process, all active Rx buffers compare their ID value to the newly received one. If a match occurs, the frame is transferred (move in) to the first (i.e. lowest entry) matching message buffer, i.e. 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 ID, data field (8 bytes maximum) and the LENGTH field are stored, the code field is updated and a status flag is set in the IFLAGH/IFLAGL register.

The CPU should read an Rx frame from its message buffer as follows:

– Control/status word (mandatory-activates internal lock for this buffer)

– ID (optional-only required if a mask was used)

– Data field word(s)

– Free-running timer (releases internal lock)

The read of the free-running timer is not mandatory. If not executed, the message buffer remains locked unless the CPU starts the read process for another message buffer. Note that only a single message buffer is locked at any one time. The only mandatory CPU read operation is of the control/status word, to ensure data coherency; if, however, the BUSY bit is set in the message buffer code, then the CPU should defer until this bit is negated. Refer to Table B-1.

The CPU should synchronize to frame receipt by the status flag for the specific message buffer (see Section B.25.3.), and not by the Control/Status word code field for that message buffer; this is because polling after the Control/Status word may lock the message buffer (see above), and the code may change before the full frame is received into the message buffer.

Note: The received Identifier field is always stored in the matching message buffer, thus the contents of the Identifier Field in an message buffer may change if the match was due to a mask.

B.11.1 Self-received frames

TOUCAN receives self transmitted frames if an Rx matching message buffer exists.

B.12 Message buffer handling

In order to maintain data coherency and proper TOUCAN operation, the CPU must obey the rules listed in Section B.10 and Section B.11. Deactivation of a message buffer is a host action that causes that message buffer to be excluded from TOUCAN transmit or receive processes; any CPU write access to a Control/Status word of message buffer structure deactivates that message buffer, thus excluding it from Rx/Tx processes. Also, any form of CPU access to a message buffer structure (other than those listed in Section B.10 and Section B.11) may cause TOUCAN to behave in an unpredictable way.

The Match/Arbitration processes are conducted only once by TOUCAN. Once a winner/match is determined, no re-evaluation is conducted whatsoever, i.e. an Rx frame may be lost. If two or more message buffers have an ID which matches that of a received frame, then receipt by TOUCAN is not guaranteed if the matching message buffer has been deactivated after the second one has been scanned.

B.12.1 Tx message buffer deactivation

There is a point in time before which deactivation of a Tx message buffer causes it not to be transmitted (End of Move), and after which the message buffer is transmitted, but no interrupt is issued and the code is not updated. If a message buffer containing the lowest ID is deactivated after TOUCAN has scanned it while in the arbitration process, TOUCAN may transmit a message buffer with an ID which may not be the lowest at the time.

B.12.2 Rx message buffer deactivation

If the deactivation occurs during move in, then it is stopped and no interrupt is issued, but the message buffer contains mixed data from two different frames. In order to prevent the host from writing data into an Rx message buffer data word(s) while it is being moved in, its Control/Status word is changed to reflect FULL or OVRN, but no interrupt will be permitted-this is expressly forbidden.

B.13 Lock/release/BUSY mechanism and SMB usage

This mechanism is implemented in order to assure data coherency in both the receive and transmit processes. The mechanism includes lock status for a message buffer, and two SMBs to buffer frame transfers within TOUCAN.

The following points should be noted:

- A CPU read of a control/status word of a message buffer triggers a lock for that message buffer, e.g. a new Rx frame which matches this message buffer, cannot be written into it
- In order to release a locked message buffer, the CPU should either lock another message buffer (by reading its control/status word), or globally release any locked message buffer (by reading the free-running timer)
- If an Rx frame with a matching ID is received while a message buffer is locked, then it cannot be stored within that message buffer, and it remains in the SMB. No indication of this situation is given
- If two or more Rx frames with matching ID are received while an message buffer is locked, then the last received frame is kept within the SMB, while all preceding ones are lost. No indication of this situation is given
- If a locked message buffer is released, and a matching frame exists within the SMB, this frame is then transferred to the matching message buffer
- If the CPU reads a Rx message buffer while it is receiving (from SMB), then the BUSY code bit is set in the control/status word, and to ensure data coherency the CPU should wait until this bit is negated before further reading from that message buffer. Note that such a message buffer is not locked
- If the CPU deactivates a locked message buffer, then its lock status is negated, but no data is transferred into that message buffer

B.14 Remote frames

A remote frame is a special kind of frame: The user initializes a remote frame as a Transmit message buffer with it's RTR bit set to '1'. After the remote frame is transmitted successfully, it's message buffer becomes a receive message buffer, with the same ID as before. When the remote frame is received by TOUCAN, its ID is compared to the IDs of the transmit message buffers with a code of 1010. If there is a matching ID, then this message buffer's frame will be transmitted.

Note: If the matching identifier message buffer holds the RTR bit set, then TOUCAN will transmit a remote frame as a response.

A received remote Request frame is not stored in a Rx buffer, but is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits of the incoming received frame should match.

In the case that a remote Request frame is received which matches a message buffer, this message buffer immediately enters the internal arbitration process, but is treated as a normal Tx message buffer, with no higher priority. The frame's data length is independent of the DLC field in the remote frame that initiated its transmission.

For further information, refer to Table B-2.

B.15 Overload frames

TOUCAN does not initiate a transmission of an overload frame. It does however transmit overload frames due to detection of the following conditions on the CAN bus:

- Detection of a dominant bit in the first/second bit of INTERMISSION
- Detection of a dominant bit at the 7th (last) bit of End_of_frame field (Rx frames)
- Detection of a dominant bit at the 8th (last) bit of error frame's delimiter or overload frame's delimiter

B.16 Time stamp

The value of the free running 16-bit timer, is sampled at the beginning of the identifier field on the CAN bus, and is stored at the end-of-frame time in the TIME STAMP entry. Knowledge of network behaviour with respect to time is therefore gained. This feature can help in network development and diagnostics.

Note: The free running timer can be reset upon a specific frame receipt, enabling network time synchronisation. Refer to the TSYNC bit in CTRL1 register.

B.17 Bit-timing configuration

TOUCAN supports a variety of means to set up the bit-timing parameters that are required by the CAN protocol. There are three 8-bit registers that enable the user to determine the value of various fields of the bit timing parameters: PROPSEG, PSEG1, PSEG2 and the RJW are programmed through fields in CTRL1 and CTRL2 registers. Also, TOUCAN contains a prescaler that enables the ratio between the system clock (IMB3 ICLOCK) and the time quanta clock (SCLOCK) to be determined. Refer to Table B-3.

Table B-3 Examples of system clock/CAN bit-rate/SCLOCK

System clock frequency (MHz)	CAN bit-rate (MHz)	Possible Sclock frequency (MHz)	Possible number of time-quanta/ bit	Prescaler programmed value + 1	Comments
24	1	8, 12, 24	8, 12, 24	3, 2, 1	Min. 8 time-quanta Max. 25 time-quanta
20	1	10, 20	10, 20	2, 1	
16	1	8, 16	8, 16	2, 1	
24	0.125	1, 1.5, 2, 3	8, 12, 16, 24	24, 16, 12, 8	
20	0.125	1, 2, 2.5	8, 16, 20	20, 10, 8	
16	0.125	1, 2	8, 16	16, 8	

Note: Bit Time = 1 + (PROPSEG + 1) + (PSEG1 + 1) + (PSEG2 + 1) x time quanta

B.18 Bit-timing operation notes

- In cases where the programmed value indicates single system clock per time quantum, then the PSEG2 field in CTRL2 should *not* be programmed to be less than 1
- In cases where the programmed value indicates single system clock per time quantum, the Information Processing Time IPT = 3, otherwise IPT = 2. Note that if PSEG2 = 2, then TOUCAN transmits 1 time quantum late relative to the scheduled sync segment
- In cases where the programmed values in the prescaler and the bit-timing control fields indicate that the number of system clocks per one bit time is less than 10 clocks, then for 100% loaded CAN bus and if the start-of-frame always comes in the 3rd bit of transmission, the TOUCAN may not complete preparing a message buffer for transmission on time, hence the TOUCAN may not go out for transmission
- At least nine system clocks per bit must be programmed in TOUCAN, otherwise correct operation is not guaranteed

B.19 TOUCAN initialisation sequence

TOUCAN may be reset in two ways: a hard reset using one of the IMB3 reset lines, or by asserting SFTRST in MCR. Following the negation of the reset, TOUCAN is out of synchronization with the CAN bus, the HALT and FRZ1 bits in MCR are set, the main control is disabled and the FRZAK and NTRDY bits in MCR are set. The TOUCAN Tx pins are in recessive state and no frames are transmitted or received. The message buffer's contents are not changed following reset.

For any configuration change/initialization, TOUCAN should either be frozen (by asserting the HALT bit in MCR), or reset (refer to Section B.20).

The following is a generic initialisation sequence applicable to TOUCAN:

i) Initialise all operation modes

- Tx and Rx pin modes (CTRL0 register)
- Bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW (CTRL1 & CTRL2 registers)
- Determine the bit rate by programming the PRESDIV register
- Determine internal arbitration mode (LBUF bit in CTRL1 register)

ii) Initialize message buffers

- The control/status word of all message buffers *must* be written either as an active or inactive message buffer
- Other entries in each message buffer should be initialized as required

iii) Initialize MASK registers for acceptance mask as required

iv) Initialize TOUCAN's interrupt handler

- Initialize ICR register's field (request level and vector value)
- Initialise interrupt arbitration identifier to a non-zero value (if interrupts from TOUCAN are desired) in MCR reegister
- Set required MASK bits in IMASKH/IMASKL register (for all message buffers interrupts), in CTRL0 register (for BOFF and error interrupts) and in MCR register for wake interrupt

v) Negate the HALT bit in the MCR register

- Starting with this event, TOUCAN attempts to synchronise with the CAN bus

B.20 Special operating modes

B.20.1 DEBUG mode

This is a special debug mode which is entered by asserting the HALT bit in MCR, or by asserting the IMB3 FREEZE line. Entering the DEBUG mode however, also depends on the state of the FRZ1 bit in MCR; for further information refer to Section B.22.4. Once in DEBUG mode, TOUCAN will wait until it is in either: intermission, passive error, BUSOFF, or IDLE state. After one of these conditions is satisfied, TOUCAN will wait for all activity (other than that in the CAN bus interface) to finish and then the following steps take place:

- TOUCAN will stop transmitting/receiving frames
- The prescaler is stopped, halting all related activities
- The CPU can read and *write* into the Error counters register
- TOUCAN ignores its Rx input pin and drives its Tx pins as recessive
- TOUCAN loses synchronization with the CAN bus; the NTRDY and FRZAK bits in MCR are set

After asserting the DEBUG mode configuration bits, the user *must* wait for the FRZAK bit in MCR to be set before requesting any further actions from TOUCAN. Failure to adhere to this condition may cause TOUCAN to operate in an unpredictable way.

Exiting the DEBUG mode is done in one of the following ways:

- Both IMB3 FREEZE and HALT bits are negated
- The CPU negates the FRZ1/FRZ0 mode bits

After exiting from DEBUG mode, TOUCAN will try to resynchronise with the CAN bus by waiting for 11 consecutive recessive bits

B.20.2 STOP mode

The STOP mode in TOUCAN is intended for power saving. Before STOP mode is selected, TOUCAN checks to see if either the CAN bus is in IDLE mode, or the third bit of intermission is a recessive bit. If one of these conditions is met, TOUCAN waits for all internal activity (other than that in the CAN bus interface) to finish and then the following steps take place:

- TOUCAN shuts down its clocks, stopping most of the internal circuits. Thus maximum power saving is achieved
- The BIU logic continues operation, enabling CPU to access MCR
- TOUCAN ignores its Rx input pin and drives its Tx pins as recessive
- TOUCAN loses synchronization with the CAN bus; the STPAK and NTRDY bits in MCR are set

Exiting the STOP mode is done in one of the following ways:

- Resetting TOUCAN (either by IMB3 hard reset, or by asserting the SFTRST bit in MCR
- Negation of the STOP bit in MCR
- Self-wake mechanism; if the SWAKE bit in MCR was set at the time TOUCAN entered STOP mode, then upon detection of recessive-dominant transition on the CAN bus, TOUCAN resets the STOP bit in MCR and resumes its clocks

When in STOP mode (or in LPSTOP), a recessive-dominant transition on the CAN bus causes the WKINT bit in STATL to be set. This event can cause a CPU interrupt if the WKMSK bit in MCR is set.

B.20.2.1 STOP mode operation notes

- When in STOP/SELF_WAKE mode, TOUCAN tries to receive the frame that woke it up, i.e. it assumes that the dominant bit detected is a start-of-frame bit. It does *not* arbitrate for the CAN bus
- Before asserting the STOP mode, the CPU should disable all interrupts in TOUCAN, otherwise it may be interrupted while in STOP mode upon a non wake-up condition. If desired, the WKMSK bit should be set to enable the WAKE_INT
- If STOP is asserted while TOUCAN is BUSOFF (refer to Section B.25.1), then TOUCAN enters STOP mode and stops counting the synchronization sequence. This count is continued when STOP is negated
- The correct procedure to enter STOP with self-wake is as follows
  - Assert SELF_WAKE *at the same time* as STOP
  - Wait for the STPAK bit to be set
- The correct procedure to exit STOP with self-wake is as follows
  - Negate SELF_WAKE *at the same time* as STOP
  - Wait for the STPAK bit to be negated
- SELF_WAKE should be set only when the STOP bit in MCR is negated and TOUCAN is ready (i.e. when the NTRDY bit in MCR is negated)
- if a recessive-dominant edge appears on the CAN bus immediately after STOP and SELF_WAKE are set, then the STOP_ACK bit in MCR may never be set, and the STOP bit in MCR is reset.
- If it is undesirable to have old frames sent when TOUCAN is awakened, all Tx sources (including remote response) should be disabled before STOP mode is entered
- If DEBUG mode is active at the time of the STOP bit being asserted, then TOUCAN assumes that the DEBUG mode should be exited; hence it tries to synchronize to the CAN bus (11 consecutive recessive bits), and only then does it search for the correct conditions to STOP.

- Trying to STOP TOUCAN immediately after reset is allowed only after basic initialization has been performed
- If STOP with self-wake is activated, and TOUCAN operates with single system clock per time-quanta, then there are extreme cases in which TOUCAN's wake-up upon recessive-dominant edge may not conform to the Bosch CAN protocol in that the TOUCAN synchronization is shifted by one time quanta from that required. This shift lasts until the next recessive-dominant edge, when TOUCAN is resynchronised to conform to the Protocol. The same is also true for Auto Power Save mode (refer to Section B.20.3) upon wake-up by a recessive-dominant edge.

B.20.3 Auto Power Save mode

This TOUCAN mode is intended to enable normal operation with optimized power saving. Upon setting the AUTOPOWERSAVE bit in MCR, TOUCAN looks for a set of conditions in which there is no need for clocks to be running. If all these conditions are met, then TOUCAN stops its clocks. If, while its clocks are stopped, any of the conditions outlined below becomes untrue, then TOUCAN resumes its clocks. It then continues to monitor the conditions and stops/resumes its clocks accordingly.

The conditions for automatic clock shut off are:

- No Rx/Tx frame in progress
- No moving of Rx/Tx frames between SMB and message buffer, and no Tx frame is pending for transmission in any message buffer
- No host access to TOUCAN
- TOUCAN is neither in DEBUG mode (MCR bit 8), in STOP mode (MCR bit 15) or in BUSOFF

B.20.4 Support BERR for RISC architechture (BERR_PLUG)

The TOUCAN supports the BERR behaviour according th o the IMB3 specification.

B.20.4.1 Modular family (BERR_PLUG = 0)

The TOUCAN never asserts the BERR signal in the IMB3.

B.20.4.2 RISC family (BERR_PLUG = 1)

The TOUCAN will terminate the cycle with BERR in the following cases:

- access to unimplemented registers
- user access to supervisor registers
- write to read only registers
- access to TCR register during normal mode

B.21 Interrupts

The TOUCAN supports two methods of Interrupt architecture, which can be chosen by a mask programming option.

B.21.1 Modular family archtecture (IRQ_PLUG = 0)

The TOUCAN module is capable of generating one interrupt level on the IMB3. This level is programmed into the Priority Level bits in the Interrupt Configuration Register. This value determines which interrupt signal (IIRQB7-1) is driven onto the bus during an interrupt request.

When an interrupt is requested, the CPU initiates an IACK cycle. The module decodes the IACK cycle and compares the CPU recognized level to the level that the module is currently requesting. If a match occurs, then arbitration begins. During arbitration, the arbitration number is driven in bit serial form, alternating between the IIARB0 and IIARB1 lines. The most significant bit of the arbitration number is driven first. Since the bus is a wire-AND type, a ‘low’ level wins any contentions. Thus the arbitration number is verified on a bit-by-bit basis. If contention is detected, (driving high and detecting low), then the module has lost the arbitration and immediately stops driving its arbitration number.

If the module wins the arbitration, it generates a uniquely encoded interrupt vector which indicates which event is requesting service. This encoding scheme is as follows. The higher 3 bits of the interrupt vector (called the Interrupt Vector Base Address) come from a 3-bit field of that name in the Interrupt Configuration Register. The lower 5 bits are an encoded value (called the Interrupt Source Number) and indicate which of the 19 interrupt sources is requesting service. Figure B-4 shows a block diagram of the interrupt hardware.

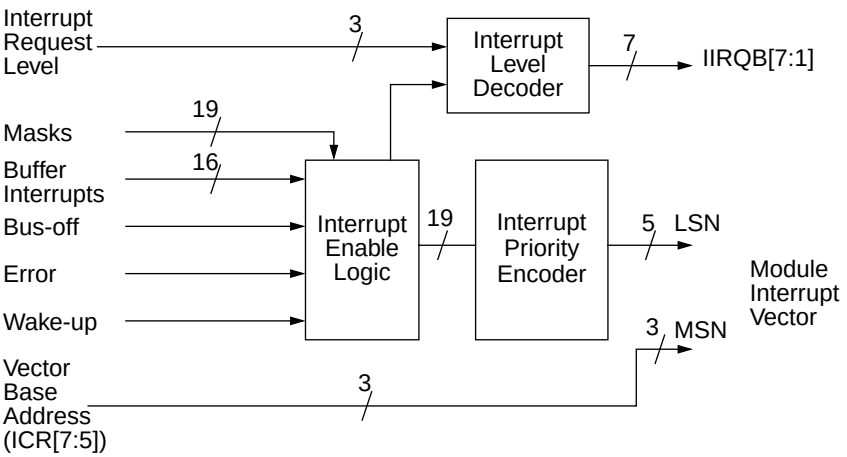


Figure B-4 TOUCAN interrupt vector generation

Each one of the 16 message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or receipt, and thus its interrupt routine can be fixed at compilation time. Each of the buffers is assigned a bit in the IFLAGH/IFLAGL register. The bit is set when the corresponding buffer completes a successful transmission/receipt, and cleared when the CPU reads the interrupt flag register (IFLAGH/IFLAGL) 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). This is known as the Standard Mechanism for IMB3 interrupts.

The other three interrupt sources (Bus-off, Error and Wake-up) act in the same way, and are located in the Error and Status register. The Bus-off and Error interrupt mask bits are located in the Control 0 register, and the Wake-up interrupt mask bit is located in the MCR.

Table B-4 Interrupt priorities and vector addresses

Name	Function	Vector address
IBUF0	Buffer0 interrupt	$X0
IBUF1	Buffer1 interrupt	$X1
IBUF2	Buffer2 interrupt	$X2
IBUF3	Buffer3 interrupt	$X3
IBUF4	Buffer4 interrupt	$X4
IBUF5	Buffer5 interrupt	$X5
IBUF6	Buffer6 interrupt	$X6
IBUF7	Buffer7 interrupt	$X7
IBUF8	Buffer8 interrupt	$X8
IBUF9	Buffer9 interrupt	$X9
IBUF10	Buffer10 interrupt	$XA
IBUF11	Buffer11 interrupt	$XB
IBUF12	Buffer12 interrupt	$XC
IBUF13	Buffer13 interrupt	$XD
IBUF14	Buffer14 interrupt	$XE
IBUF15	Buffer15 interrupt	$XF
IBOFF	Bus-off interrupt	$Y0
IERROR	Error interrupt	$Y1
IWAKE	Wake-up interrupt	$Y2 (Lowest priority)

Note: X = bbb0
Y = bbb1
bbb = ICR[7:5]

B.21.2 RISC family architecture (IRQ_PLUG = 1)

The interrupt structure of the IMB3 supports a total of 32 interrupt levels that are time multiplexed on the IRQB[0:7] lines as seen in Table B-5. In this structure, all interrupt sources place their asserted level on a time multiplexed bus during four different time slots, with eight level communicated per slot. (However, each group of levels actually occur, one system clock cycle after the associated IMB3 IILBS signal is asserted). The ILBS[0:1] signals indicate which group of eight are being driven on the interrupt request lines.

Table B-5 Interrupt levels

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

The TOUCAN modules is capable of generating 1 of 32 possible interrupt levels on the IMB3. The level that the TOUCAN will drive can be programmed into the Interrupt Request Level bits located in the Interrupt Configuration Register (IRL[2:0] bit field - bits [8:0] in the ICR register). The 2 bits ILBS[1:0] in the ICR register (ILBS bit field - bits [7:6] in the ICR register) determine on which slot the TOUCAN should drive its interrupt signal (one of IRQB[0-7]). Figure B-5 shows the timing of slot multiplexing on the IMB3.

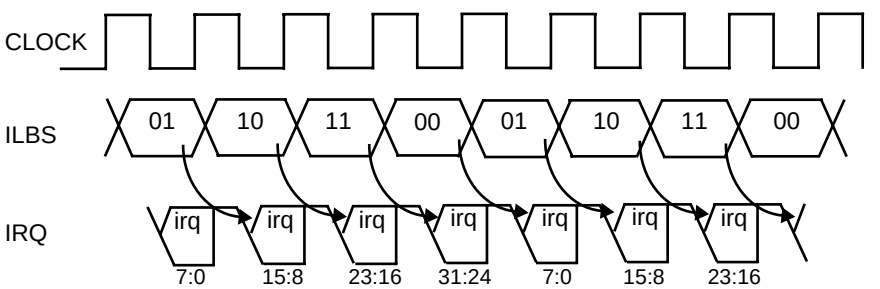


Figure B-5 Int request multiplex timing

B

B.22 Programmer's model

This section describes the registers and data structures in the TOUCAN module. The base address of the module is hardware programmable as defined by the IMB3 Specification. The address space occupied by TOUCAN is continuous: 128 bytes starting at the base address, and 256 extra bytes starting at the base address + 128. The upper 256 are fully used for the message buffer structures. Only part of the lower 128 bytes is occupied by various registers.

The register decode map is fixed and begins at the first address of the Module Base Address. Table B-5 shows the registers associated with the TOUCAN module and their relative offset from the base address.

Table B-6 TOUCAN memory map

		Offset	S/U[1]	Register Function	R/W Access	Reset Value
				System Registers		
MCR		$00	S	Module Configuration Register	R/W	$5980
TCR		$02	S	Test Configuration Register	Test	$0000
ICR		$04	S	Interrupt Configuration Register	R/W	$000F
				Control Registers		
CTRL0	CTRL1	$06	U	Control Register	R/W	$0000
PRESDIV	CTRL2	$08	U	Control & Prescaler Divider	R/W	$0000
TIMER		$0A	U	Free Running Timer	R/W	$0000
				Rx Mask Registers		
RXGMASK_HIGH		$10	U	Rx Global Mask (high)	R/W	$FFEF
RXGMASK_LOW		$12	U	Rx Global Mask (low)	R/W	$FFFE
RX14MASK_HIGH		$14	U	Rx Buffer 14 Mask (high)	R/W	$FFEF
RX14MASK_LOW		$16	U	Rx Buffer 14 Mask (low)	R/W	$FFFE
RX15MASK_HIGH		$18	U	Rx Buffer 15 Mask (high)	R/W	$FFEF
RX15MASK_LOW		$1A	U	Rx Buffer 15 Mask (low)	R/W	$FFFE
				Global Info Registers		
Error & Status		$20	U	Error & Status	R/W	$0000
IMASK		$22	U	Interrupt Masks	R/W	$0000
IFLAGH/IFLAGL		$24	U	Interrupt Flags	R/W	$0000
Rx Err Cntr	Tx Err Cntr	$26	U	Rx & Tx Error counters	R/W	$0000
Control/status		$80	U	Message Buffer 0	–	–
ID_HIGH		$82				
ID_LOW		$84				
8 bytes data field		$86 : $8C				
reserved		$8E				
		$90	U	Message Buffer 1	–	–
		$A0	U	Message Buffer 2.		
—			—		–	–
—			—			
—			—			
		$170	U	Message Buffer 15		

(1) Supervisor/Unrestricted

B

B.22.1 Programming validity

Note that TOUCAN has no hard wired protection against invalid byte/field programming within its registers; specifically, no protection is given in case the programming does not meet CAN protocol requirement (for example, invalid bit segment values).

Also, programming TOUCAN control registers should be done at the initialization phase, prior to TOUCAN becoming synchronized with the CAN bus, or after assertion of the HALT/FREEZE mode while TOUCAN is in DEBUG state and NTRDY bit in MCR is set.

B.22.2 Reserved bits

In some cases the TOUCAN registers contain bit locations marked as 'reserved'. These bits should always be written as logic '0'.

B.22.3 System registers

These three registers (MCR, TCR, and ICR) define global configuration of the TOUCAN module, such as interrupt level and base vector address, in addition to various operation modes (e.g. sleep) and testing modes (e.g. internal logic visibility and controlability).

B.22.4 MCR — Module configuration register

	Address	bit 15	14	13	12	11	10	9	bit 8	State on reset
Module configuration register (MCR)	$00	STOP	FRZ1	FRZ0	HALT	NTRDY	WKMSK	SFTRST	FRZAK	0101 1001

	bit 7	6	5	4	3	2	1	0	
$01	SUPV	SWAKE	PWRSV	STPAK	IAI4	IAI3	IAI2	IAI1	1000 0000

STOP — Low power sleep mode

This bit may be set by the CPU, or negated by either the CPU or by TOUCAN if the SELF-WAKE bit in MCR is set.

1 (set) – Shut down TOUCAN clocks.

0 (clear) – Enable TOUCAN clocks.

FRZ1 — FREEZE enable bit 1

The FRZ1 bit specifies the TOUCAN response either to the FREEZE signal on the IMB3 being asserted, or to the HALT bit in the MCR being asserted. This bit is initialised to '1' during reset.

1 (set) – Refer to IMB3 FREEZE/HALT

0 (clear) – Ignore FREEZE/HALT

When FRZ1 = 0, TOUCAN ignores both the FREEZE signal on the IMB3 and the HALT bit in the MCR.

When FRZ1 = 1, it enables TOUCAN to enter the DEBUG HALT/FREEZE mode. In order to enter this mode, the FRZ1 bit should be set to '1', and either the IMB3 FREEZE line should be asserted, or the HALT bit in MCR should be set. Negation of this bit field causes TOUCAN to exit from the FREEZE/HALT mode. For further details refer to Section B.20.1.

FRZ0 — FREEZE enable bit 0

FRZ0 is not used in TOUCAN.

HALT — Halt TOUCAN Sclock

1 (set) – Enter DEBUG mode if FRZ1 = 1.

0 (clear) – No TOUCAN internal request to enter DEBUG mode.

Assertion of this bit has the same effect as the assertion of the FREEZE signal on the IMB3, as described in the FRZ1/FRZ0 sections above. However, it does not require that the FREEZE signal be asserted in order to enter DEBUG mode.

The bit is initialized to '1' (DEBUG mode). It is cleared by the CPU after the message buffers and control registers have been initialized. When the HALT bit is asserted, the CPU also has write-access to the error counters.

For a detailed description of the DEBUG mode, refer to Section B.20.1.

NTRDY — TOUCAN not ready

1 (set) – TOUCAN has entered either STOP or DEBUG mode.

0 (clear) – TOUCAN has exited either STOP or DEBUG mode.

B

This bit indicates that TOUCAN is in either STOP or DEBUG mode. This bit is read only. Whenever one of these two modes is asserted, this bit becomes set when TOUCAN enters that mode; it then becomes negated only when TOUCAN exits the mode, either by synchronisation to the BUS (11 recessive bits) or by the self-wake mechanism. For further details refer to descriptions of the HALT, FRZ1/FRZ0, FRZAK, STOP and STPAK bits in this section.

WKMSK — Wake-up interrupt mask

This bit enables the wake-up interrupt generation.

1 (set) – Wake-up interrupt enabled.

0 (clear) – Wake-up interrupt disabled.

SFTRST — Soft reset

1 (set) – Request for soft reset initiated.

0 (clear) – Normal operation.

After this bit is asserted, TOUCAN resets its internal machines (sequencer, error counters, error flags, timer) and the host-interface registers (MCR, ICR, TCR, IMASK, IFLAGH/IFLAGL).

The configuration bits that control the interface with the CAN bus (CTRL0, CTRL1, CTRL2 and PRESDIV) remain unchanged, as do the message buffers and the Rx message masks. This enables the CPU to use the SFTRST as a debug feature during run-time of the system. SFTRST also affects the MCR register, thus the STOP bit in MCR is reset, causing TOUCAN to resume its clocks after coming out of STOP low-power mode.

Note that the next CPU access, after setting the SFTRST bit, should not be to TOUCAN, thus allowing the TOUCAN internal circuitry to be fully reset. This bit is self-negated.

FRZAK — TOUCAN disabled and unsynchronised with CAN bus

1 (set) – TOUCAN is in DEBUG mode.

0 (clear) – TOUCAN is not in DEBUG mode, and the prescaler is running.

FRZAK is a read-only bit which is set to ‘1’ when TOUCAN enters DEBUG mode, and ‘0’ when DEBUG mode is negated and the prescaler is running.

When TOUCAN enters DEBUG mode, it sets the FRZAK bit. The CPU can poll this bit to find out if TOUCAN entered DEBUG mode. If DEBUG mode is negated then FRZAK is also negated once the TOUCAN prescaler is running. Refer also to the NTRDY bit in this section.

SUPV — Supervisor mode

Some registers on TOUCAN are always Supervisor data space, while others are programmable as either Supervisor or Unrestricted data space by ignoring the FC2 line. This bit is concerned only with the latter. The SUPV bit is initialised to logic '1' during reset.

1 (set) – Assigned registers are restricted (FC2 line is decoded) and all TOUCAN registers are placed in supervisor-only space. For any access with a user data space function code, address acknowledge (AACK) is not returned, and the bus cycle is transferred externally.

0 (clear) – Assigned registers are unrestricted (FC2 line is ignored). AACK is returned for accesses with either supervisor or user data space function codes, and the cycle remains internal. If a supervisor-only register is accessed with a user data space function code, the module responds as though an access had been made to an unimplemented register location.

SWAKE — Self wake up

1 (set) – Self wake-up enabled.

0 (clear) – Self wake-up disabled.

This bit enables the self wake-up of TOUCAN after STOP, without CPU intervention. If this bit is set when entering STOP, TOUCAN will look for a dominant bit on the bus during STOP. If a recessive-dominant transition is detected, then TOUCAN will negate the STOP bit immediately and resume the clocks.

This bit should be treated as a command, i.e. it is not always updated as written; the user should verify if the value that has been written was captured in the register by reading MCR.

Note: This bit should not be set if the LPSTOP command is executed. Fur further information refer to Section B.20.2.

PWRSV — Auto power save

1 (set) – Auto power save mode active; clocks stop and resume when required.

0 (clear) – Auto power save mode not active; clocks run normally.

This bit enables TOUCAN to automatically shut off its clocks when it has no process to execute, and then to resume them when it has a task to execute. There is no CPU intervention.

Note: The BIU clocks are not stopped, thus enabling host access. Also, the auto power save action does not depend on the values of SWAKE or WKMSK.

STPAK — TOUCAN stopped

1 (set) – TOUCAN is in STOP mode.

0 (clear) – Normal TOUCAN operation exists.

This bit is read-only.

When TOUCAN enters STOP mode and shuts its clocks, STPAK becomes set. The CPU can poll this bit to find out if TOUCAN entered STOP mode. If the STOP bit is negated then this bit is negated once the TOUCAN clocks are running.

IAI[4:1] — Interrupt arbitration identifier

This four bit encoded field contains the interrupt arbitration number of this particular TOUCAN module with respect to all other subsystems and peripherals that may generate interrupts. The interrupt arbitration ID is used to arbitrate the IMB3 (relevant only when IRQ_PLUG = 0) when two or more modules have an interrupt on the same priority level pending simultaneously. This field is initialized to the non-arbitrating state, $0, during reset. If no arbitration takes place during the IACK cycle, the spurious interrupt vector is generated after a time-out by the SIM module, alerting the system to the fact that an interrupt arbitration ID has not been initialized.

B.22.5 TCR — Test configuration register

This register is for factory test purposes only.

B.22.6 ICR — Interrupt configuration register (Modular family IRQ_PLUG = 0)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Interrupt configuration (ICR) (high)	$04	0	0	0	0	0	IRQ3	IRQ2	IRQ1	0000 0000
Interrupt configuration (ICR) (low)	$05	IVB3	IVB2	IVB1	0	1	1	1	1	0000 1111

IRQ[3:1] — Interrupt request level

The interrupt request level field contains the priority level of the TOUCAN interrupts for the CPU, where 111 indicates a nonmaskable interrupt, while 000 indicates that interrupts have been disabled. If an interrupt flag is asserted and the corresponding mask bit is set to ‘1’, no interrupt is generated unless the interrupt request level is a non-zero value. The interrupt request level field, therefore, acts as master enable for the interrupts. The interrupt request level field is initialized to zero during reset; this prevents the module from generating an interrupt until this register has been initialized.

IVB[3:1] — Interrupt vector base address

The interrupt vector base address is specified by bits IVB[3:1]. This field specifies the most significant nibble of all the vector numbers generated by the different TOUCAN interrupt sources. This field is initialized to zero during reset.

ICR[4:0]

ICR[4:0] always read as '01111'. If TOUCAN issues an interrupt request after RESET and before initializing the low byte of ICR, it will drive $0F as the interrupt vector, in response to a CPU interrupt acknowledge cycle, regardless of the specific interrupt event. This is the 'uninitialised interrupt' vector, as defined in the IMB3 specification and in the CPU16 reference manual.

B.22.7 ICR — Interrupt configuration register (RISC family IRQ_PLUG = 1)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Interrupt configuration (ICR) (high)	$04	0	0	0	0	0	IRL3	IRL2	IRL1	0000 0000
Interrupt configuration (ICR) (low)	$05	ILBS2	ILBS1	0	0	1	1	1	1	0000 1111

IRL[2:0] — Interrupt request level

The interrupt request level field contains the priority level of the TOUCAN interrupts for the CPU.

ILBS[1:0]

These bits indicate on which slot the TOUCAN should drive its interrupt..

Table B-7 Interrupt levels

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

B

B.23 Control registers

These registers provide control related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, and a global free-running timer.

B.23.1 CTRL0 — Control register 0

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Control 0 (CTRL0)	$06	BOFF	ERR	reserved	reserved	RXMD1	RXMD0	TXMD1	TXMD0	0000 0000

BOFF — Bus-off mask

This bit provides a mask for the bus-off interrupt.

1 (set) – Interrupt enabled.

0 (clear) – Interrupt disabled.

ERR — Error mask

This bit provides a mask for the error interrupt.

1 (set) – Interrupt enabled.

0 (clear) – Interrupt disabled.

RXMD[1,0] — Rx modes

Configuration control of the Rx0 pin.

RXMD1 is reserved; RXMD0 represents the polarity interpretation of Rx0. Operation of this bit is as follows.

1 (set) – A dominant level is interpreted as a logical ‘1’; a recessive level is interpreted as a logical 0.

0 (clear) – A dominant level is interpreted as a logical ‘0’; a recessive level is interpreted as a logical 1.

TXMD[1,0] — Tx modes

Configuration control of Tx0 and Tx1 pins. The operation of these bits is as shown in Table B-8.

Table B-8 Configuration control of Tx0, Tx1 pins

TXMD1	TXMD0	
0	0	Full CMOS; positive polarity (i.e. Tx0=0, Tx1=1 is a dominant level)
0	1	Full CMOS; negative polarity (i.e. Tx0=1, Tx1=0 is a dominant level)
1	x	Open drain; positive polarity

Note: "Full CMOS" drive means both dominant and recessive levels are driven by the chip. "Open drain" drive means that only a dominant level is driven by the chip. During a recessive level the Tx0, Tx1 pins are disabled (tri-state), and the electrical level is achieved by external pull-up/pull-down devices.
The assertion of both Tx modes bits causes the polarity inversion to be cancelled, i.e. open drain mode forces the polarity to be positive.

B.23.2 CTRL1 — Control register 1

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Control 1 (CTRL1)	$07	SAMP	LOOP	TSYNC	LBUF	reserved	PROP2	PROP1	PROP0	0000 0000

SAMP — Sampling mode

1 (set) – Three samples are used to determine the value of the received bit; the regular one (sample point) and two preceding samples, using a majority rule.

0 (clear) – One sample is used to determine the value of a received bit.

TSYNC — Timer synchronise mode

1 (set) – Timer synchronise enabled.

0 (clear) – Timer synchronise disabled.

This bit enables a mechanism that resets (clears) the free-running timer each time a message is received in MB0. This feature provides means to synchronize multiple TOUCAN stations with a special "SYNC" message (i.e., global network time). A buffer0 interrupt is also available.

Note: There is a possibility of a 4-5 tick count skew between the different TOUCAN stations that would operate this mode.

LBUF — Lowest buffer transmitted first

This bit defines the transmit-first scheme.

1 (set) – Lowest buffer is transmitted first.

0 (clear) – Lowest ID is transmitted first.

Bit 3 — Reserved

Caution: This bit must not be written as ‘1’.

PROPSEG[2:0] — Propagation segment

This field defines the length of the propagation segment in the bit time. The valid programmed values are 0-7.

Propagation segment time = (PROPSEG + 1) * Time-Quanta

(time-quanta = 1 Sclock. Refer to Section B.23.3.)

B.23.3 PRESDIV — Prescaler divide register

	Address	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	State on reset
Prescale divide (PRESDIV)	$08	(bit 15)	(14)	(13)	(12)	(11)	(10)	(9)	(bit 8)	0000 0000

PRESDIV[15:8]

This field determines the ratio between the system clock frequency and the Sclock (Serial clock). (1 Sclock = 1 time quantum).

The Sclock is equal to the system clock divided by (value of this register + 1).

The reset value of this register is $0000, which means that the Sclock is the same as the system clock frequency.

The maximum value of this 8-bit register is $FF, which gives the minimum Sclock frequency of (system clock/256). For a 16MHz system clock, this gives a 64kHz Sclock, which can operate a CAN bit rate of 8K bit/s. For further information refer to Section B.17.

B.23.4 CTRL2 — Control register 2

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Control 2 (CTRL2)	$09	RJW1	RJW0	PSEG12	PSEG11	PSEG10	PSEG22	PSEG21	PSEG20	0000 0000

RJW[1:0] — Resynchronise jump width

This field defines the maximum number of time quanta a bit time may be changed by one resynchronisation. The valid programmed values are 00-11.

Resynchronise jump width = RJW value + 1

PSEG1[2:0] — Phase segment 1

This field defines the length of phase buffer segment 1 in the bit time.

The valid programmed values are 000-111.

Phase buffer segment 1 = (PSEG1 + 1) x Time quanta

PSEG2[2:0] — Phase segment 2

This field defines the length of phase buffer segment 2 in the bit time.

The valid programmed values are 000-111.

Phase buffer segment 2 = (PSEG2 + 1) x Time quanta

B.23.5 TIMER — Free running timer

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Free running timer (TIMER)	$0A	(bit 15)	(14)	(13)	(12)	(11)	(10)	(9)	(bit 8)	0000 0000
		(bit 7)	(6)	(5)	(4)	(3)	(2)	(1)	(bit 0)	0000 0000

This is a 16-bit free running counter which can be read and written by the CPU. The timer starts from $0000 after reset, counts linearly to $FFFF, and wraps around.

This 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 counts the nominal bit rate.

The timer value is captured at the beginning of the ID field of any frame on the CAN bus; this value is then written into the TIME STAMP entry in a message buffer after a successful receipt/transmission of a message.

B.24 Rx mask registers

These registers are used as acceptance masks for received frame ID's. Three masks are defined: a global mask, used for all Rx buffers 0-13, and two separate masks for buffers 14 and 15. The following applies for all the mask bits within these registers.

1 (set) – The corresponding ID bit is checked against the incoming ID bit, to see if a match exists.

0 (clear) – The corresponding incoming ID bit is 'don't care'.

Note: These masks are used for both standard and extended ID formats. The value of mask registers should *not* be changed while in normal operation, as locked frames which matched a message buffer through a mask may be transferred into the message buffer (upon release) but may no longer match. Refer to Table B-9.

Table B-9 Mask examples for normal/extended messages

	Base ID ID28...ID18	IDE	Extended ID ID17...ID0	Match	
MB2 - ID	11111111000	0			
MB3 - ID	11111111000	1	010101010101010101		
MB4 - ID	00000011111	0			
MB5 - ID	00000011101	1	010101010101010101		
MB14 - ID	11111111000	1	010101010101010101		
Rx global mask	11111111110		111111100000000001		
Rx_msg in	11111111001	1	010101010101010101	3	(1)
Rx_msg in	11111111001	0		2	(2)
Rx_msg in	11111111001	1	010101010101010101		(3)
Rx_msg in	01111111000	0			(4)
Rx_msg in	01111111000	1	010101010101010101		(5)
Rx_14_mask	01111111111		111111100000000000		
Rx_msg in	10111111000	1	010101010101010101		(6)
Rx_msg in	01111111000	1	010101010101010101	14	(7)

(1) Match for extended format (MB3).

(2) Match for standard format (MB2).

(3) Un-match for MB3 because of ID0.

(4) Un-match for MB2 because of ID28.

(5) Un-match for MB3 because of ID28; match for MB14.

(6) Un-match for MB14 because of ID27.

(7) Match for MB14.

B.24.1 RXMASK — Rx global mask register

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Rx global mask (RXMASK)	$10	MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	1111 1111
	$11	MID20	MID19	MID18	0	1	MID17	MID16	MID15	1110 1111
	$12	MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	1111 1111
	$13	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0	1111 1110

The Rx global mask registers are composed of 4 bytes. The mask bits are applied to all Rx identifiers excluding Rx buffers 14-15 which have their specific Rx mask registers.

Base ID: ID28-ID18

These bits are the same mask bits for standard or extended format.

Extended ID: ID17-ID0

These bits are used to mask comparison only in extended format.

RTR/SRR (bits 20 and 0)

The RTR bit of a received frame is never compared to the corresponding bit in the message buffer ID field. Also, remote request frames are never received into message buffers. These bits are always ‘0’, regardless of any CPU write to these bits.

IDE

The IDE bit of a received frame is always compared. This bit is always ‘1’, regardless of any CPU write to this bit.

B.24.2 RX14MASK — Rx buffer 14 mask

The Rx buffer 14 mask register has the same structure as the Rx global mask register and is used to mask buffer 14. The register is located at address $14-$17.

B.24.3 RX15MASK — Rx buffer 15 mask

The Rx buffer 15 mask register has the same structure as the Rx global mask register and is used to mask buffer 15. The register is located at address $18-$1B.

B.25 Global information registers

To negate an interrupt flag, the flag must first be read as ‘1’ and then written as ‘0’. For future reference, this method of negating the interrupt flag is referred to as the **Motorola Standard Mechanism**.

B.25.1 STATH, STATL — Error and status report registers

	Address	bit 15	14	13	12	11	10	9	bit 8	State on reset
Error and status report high (STATH)	$0070	B1ERR	B0ERR	ACKER	CRCER	FMERR	STERR	TXWRN	RXWRN	0000 0000

		bit 7	6	5	4	3	2	1	0	
Error and status report low (STATL)	$0071	IDLE	TX/RX	BUS_STATE		0	BOFINT	ERRINT	WKINT	0000 0000

These registers include error condition information, general status information and three interrupt sources. The reported error conditions are those which have occurred since the last time the register was read. These bits are cleared after a read.

All bits in STATH and STATL are read-only, except for the interrupt sources (BOFINT, ERRINT, WKINT). For further information refer to Section B.21 (Interrupts).

Table B-10 Bit error status

B1ERR/B0ERR	Bit error status
0 0	No transmit error
0 1	At least one bit sent as dominant is received as recessive
1 0	At least one bit sent as recessive is received as dominant
1 1	Not used

B1ERR — Bit 1 error bit See Table B-10 for details

Note: This bit is *not* set by a transmitter in the case of an arbitration field or ACK slot, or in the case of a node sending a passive error flag that detects dominant bits.

B0ERR — Bit 0 error bit See Table B-10 for details

ACKER — ACK error

1 (set) – An ACK error has occurred since the last read of this register.

0 (clear) – No ACK error has occurred since the last read of this register.

CRCER — CRC error

1 (set) – A CRC error has occurred since the last read of this register.

0 (clear) – No CRC error has occurred since the last read of this register.

FMERR — FORM error

1 (set) – A FORM error has occurred since the last read of this register.

0 (clear) – No FORM error has occurred since the last read of this register.

STERR — STUFF error

1 (set) – A STUFF error has occurred since the last read of this register.

0 (clear) – No STUFF error has occurred since the last read of this register.

TXWRN — Tx warn

1 (set) – Tx_Error_Counter ≥ 96.

0 (clear) – Tx_Error_Counter < 96.

This status flag does not cause an interrupt.

RXWRN — Rx warn

1 (set) – Rx_Error_Counter ≥ 96.

0 (clear) – Rx_Error_Counter < 96.

This status flag does not cause an interrupt.

IDLE

1 (set) – The CAN bus is now idle.

0 (clear) – The CAN bus is not idle.

TX/RX — Transmit/receive

1 (set) – TOUCAN is transmitting a message if IDLE = ‘0’.

0 (clear) – TOUCAN is receiving a message if IDLE = ‘0’.

This bit has no meaning in the case where IDLE = ‘1’.

BUS_STATE — Fault confinement state

This two-bit field describes the state of TOUCAN:

Table B-11 Fault confinement sate of TOUCAN

Bit 5	Bit 4	State
0	0	Error active
0	1	Error passive
1	x	BUSOFF

If SFTRST in MCR is asserted while TOUCAN is in BUSOFF state, then STATH and STATL are reset (including the BUS_STATE bits), but when exiting the DEBUG mode state, the BUS_STATE bits return to reflect the BUSOFF state.

BOFINT — Bus off interrupt

1 (set) – TOUCAN’s bus state has changed to BUSOFF.

0 (clear) – No such occurrence.

Each time the TOUCAN state changes to BUSOFF, this bit is set, and if the corresponding mask bit (BOFF) is set, an interrupt is generated. This interrupt is *not* generated after reset. Use of the Motorola Standard Mechanism is required to reset this flag to ‘0’ and negate its corresponding interrupt. Writing a ‘1’ to this bit has no effect.

ERRINT — Error interrupt

1 (set) – TOUCAN has detected a CAN bus error *or* one of the error conditions has been set.

0 (clear) – No such occurrence.

Each time one of the error bits (bits 15:10) is set (even if already set), this bit is set, and if the ERR bit in CTRL0 is set, an interrupt is generated to the host. Using the Motorola Standard Mechanism is required to reset this flag to ‘0’ and negate its corresponding interrupt. Writing a ‘1’ to this bit has no effect.

WKINT — Wake interrupt

1 (set) – Recessive to dominant transition event has occurred on the CAN bus when in STOP mode.

0 (clear) – No such occurrence.

If a recessive to dominant transition is detected on the CAN bus while TOUCAN is in low-power SLEEP mode (STOP=1 in MCR), and if WKMSK bit in MCR is set, then this bit becomes set and an interrupt is generated to the CPU. Refer to Section B.22.4 (Module Configuration Register (MCR)) for further details. Using the Motorola Standard Mechanism is required to reset this flag to '0' and negate its corresponding interrupt. Writing a '1' to this bit has no effect.

B.25.2 IMASKH, IMASKL— Interrupt mask registers

	Address	bit 15	14	13	12	11	10	9	bit 8	State on reset
Interrupt mask high (IMASKH)	$0022	BUF15M	BUF14M	BUF13M	BUF12M	BUF11M	BUF10M	BUF9M	BUF8M	0000 0000

		bit 7	6	5	4	3	2	1	0	
Interrupt mask low (IMASKL)	$0023	BUF7M	BUF6M	BUF5M	BUF4M	BUF3M	BUF2,	BUF1M	BUF0M	0000 0000

Caution: Setting or clearing a bit in the IMASKH/IMASKL register can assert or negate an interrupt request, respectively.

This register contains one interrupt mask bit per buffer. It enables the CPU to determine which buffer will generate an interrupt after a successful transmission/receipt (i.e. when the corresponding IFLAGH/IFLAGL bit is set).

IMASKH and IMASKL contain two 8-bit fields: bits 15-8 (IMASKH) and bits 7-0 (IMASKL). The register can be accessed by the master as a 16-bit register, or each byte can be accessed individually using an 8-bit (1 byte) access cycle.

BUFM[15:0]

1 (set) – The corresponding buffer interrupt is enabled.

0 (clear) – The corresponding buffer interrupt is disabled.

B.25.3 IFLAGH, IFLAGL — Interrupt flag registers

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Interrupt flag high (IFLAGH)	$0024	BUF15I	BUF14I	BUF13I	BUF12I	BUF11I	BUF10I	BUF9I	BUF8I	0000 0000
Interrupt flag low (IFLAGL)	$0025	BUF7I	BUF6I	BUF5I	BUF4I	BUF3I	BUF2I	BUF1I	BUF0I	0000 0000

The interrupt flag registers contain one interrupt flag bit per buffer. Each successful transmission/receipt sets the corresponding IFLAG bit and, if the corresponding IMASK bit is set, will generate an interrupt.

The register contains two 8-bit fields: bits 15-8 (IFLAG_H) and bits 7-0 (IFLAG_L). The register can be accessed by the master as a 16-bit register, or each byte can be accessed individually using an 8-bit (1 byte) access cycle.

IBUFl[15:0]

1 (set) – The corresponding buffer has successfully completed transmission or receipt.

0 (clear) – The corresponding buffer has not completed transmission or receipt.

Should a new interrupt occur between the time that the CPU reads the flag as '1' and writes the flag as '0' to clear it, the flag will *not* be cleared in order to indicate the new interrupt request. The register is initialized to zero during reset. This register is writeable to '0's only, as defined in the standard mechanism.

B.25.4 Error counters

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Rx Error Counter	$26	(bit 15)	(14)	(13)	(12)	(11)	(10)	(9)	(bit 8)	0000 0000
Tx Error Counter	$27	(bit 7)	(6)	(5)	(4)	(3)	(2)	(1)	(bit 0)	0000 0000

There are two error counters in TOUCAN; transmit error counter and receive error counter. The rules for increasing and decreasing these counters are described in the CAN Protocol Specification, Version 2.0 and are fully implemented in TOUCAN. Each counter comprises the following.

- 8-bit up/down counter
- Increment by 8 (Rx_Err_Counter also by 1)
- Decrement by 1
- Avoid decrement when equal to zero
- Rx_Err_Counter preset to a value $119 \leq x \leq 127$
- Value after reset = zero
- Detect values for error_passive, error_active and BUSOFF transitions and for alerting the host
- Cascade usage of Tx_Err_Counter with an internal other counter to detect the 128 occurrences of 11 consecutive recessive bits to determine move from Bus-Off into error_active.

Both counters are read only (except for Test/Freeze/halt modes).

TOUCAN responds to any bus state as described in the protocol, e.g. transmit error active or error passive flag, delay its transmission start time (Error Passive) and avoid any influence on the bus when in Bus Off state. The following are the basic rules for TOUCAN bus state transitions:

- If the value of Tx_Err_Counter or Rx_Err_Counter becomes ≥ 128, the BUS_STATE field in the Error Status Register is updated to reflect this (Error Passive state is set).
- If the TOUCAN state is Error Passive, and either the Tx_Err_Counter counter or the Rx_Err_Counter becomes ≤ 127 while the other already satisfies this condition, the BUS_STATE field in the Error Status Register is updated to reflect this (Error Active state is set).
- If the value of the Tx_Err_Counter increases to be greater than or equal to 256, the BUS_STATE field in the Error Status Register is updated to reflect it (set BUSOFF state) and an interrupt may be issued. The value of Tx_Error_Counter is then reset to zero.
- If the TOUCAN state is BUSOFF, then Tx_Error_Counter, together with an internal counter are cascaded to count the 128 occurrences of 11 consecutive recessive bits on the bus. Hence, Tx_Error_Counter is reset to zero, and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the Tx_Err_Counter. When Tx_Err_Counter reaches the value of 128, BUS_STATE field in the Error Status Register is updated to be Error Active, and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero, but does NOT affect the Tx_Err_Counter value.
- If during system start-up, only one node is operating, then its Tx_Err_Counter increases each message it's trying to transmit, as a result of ACK_ERROR. A transition to bus state Error Passive should be executed as described, while this device never enters the BUSOFF state.
- If the Rx_Err_Counter increases to a value greater than 127, it is prevented from being increased any further, even if more errors are detected while being a receiver. At the next successful message receipt, the counter is set to a value between 119 and 127, to enable Error Active state to be resumed.

THIS PAGE LEFT BLANK INTENTIONALLY

C
THE MOTOROLA SCALEABLE CAN (MSCAN08) MODULE

The MSCAN08 is the specific implementation of the Motorola Scalable CAN (MSCAN) concept targeted for the Motorola M68HC08 Microcontroller Family.

The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification dated September 1991.

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth.

MSCAN08 utilises an advanced buffer arrangement resulting in a predictable real-time behaviour and simplifies the application software.

C.1 Features

The basic features of the MSCAN08 are as follows:

- Modular Architecture
- Implementation of the CAN protocol - Version 2.0A/B
  - Standard and extended data frames.
  - 0 - 8 bytes data length.
  - Programmable bit rate up to 1 Mbps†.
- Support for Remote Frames.
- Double buffered receive storage scheme.
- Triple buffered transmit storage scheme with internal priorisation using a *local priority* concept.

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

- Flexible maskable identifier filter supports alternatively one full size extended identifier filter or two 16 bit filters or four 8 bit filters.
- Programmable wake-up functionality with integrated low-pass filter.
- Programmable Loop-Back mode supports self-test operation.
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (Warning, Error Passive, Bus-Off).
- Programmable MSCAN08 clock source either CPU bus clock or crystal oscillator output.
- Programmable link to on-chip Timer Interface Module (TIM) for time-stamping and network synchronisation.
- Low power sleep mode.

C.2 External Pins

The MSCAN08 uses 2 external pins, 1 input (RxCAN) and 1 output (TxCAN). The TxCAN output pin represents the logic level on the CAN: ‘0’ is for a dominant state, and ‘1’ is for a recessive state.

A typical CAN system with MSCAN08 is shown in Figure C-1.

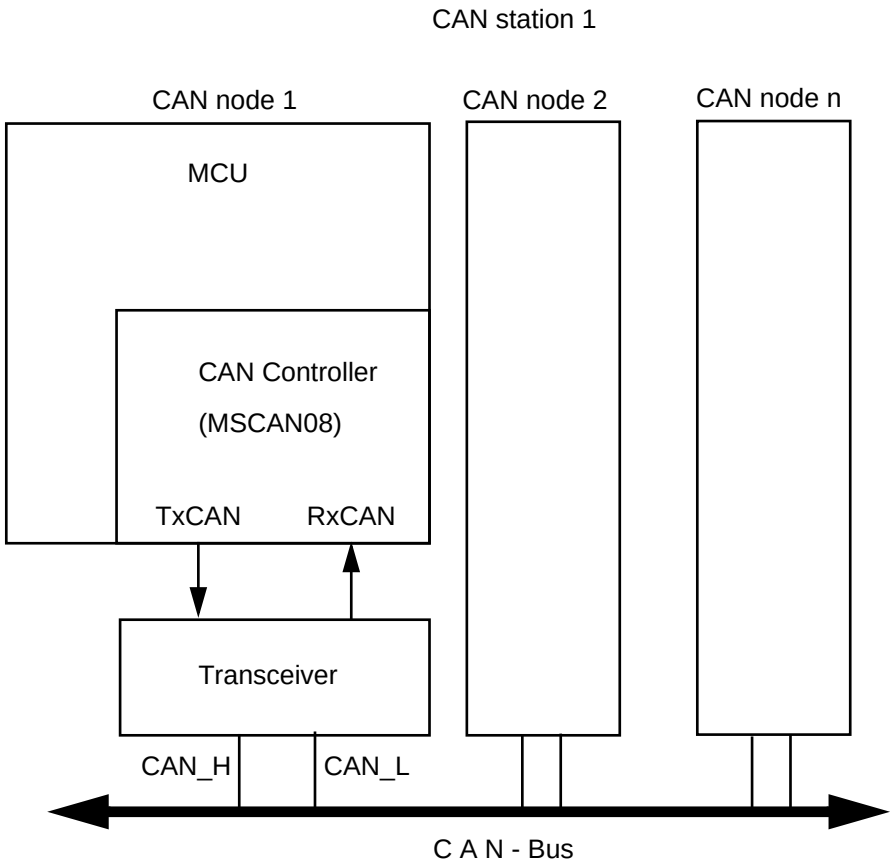


Figure C-1 The CAN System

Each CAN station is connected physically to the CAN bus lines through a transceiver chip. The transceiver is capable of driving the large current needed for the CAN and has current protection, against defected CAN or defected stations.

C

C.3 Message Storage

MSCAN08 facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

C.3.1 Background

Modern application layer software is built under two fundamental assumptions:

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

The above behaviour can not be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message has been sent. This loading process lasts a definite amount of time and has to be completed within the Inter-Frame Sequence (IFS) in order to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme would de-couple the re-loading of the transmit buffers from the actual message sending and as such reduces the reactiveness requirements on the CPU. Problems may arise if the sending of a message would be finished just while the CPU re-loads the second buffer, no buffer would then be ready for transmission and the bus would be released.

At least three transmit buffers are required to meet the first of above requirements under all circumstances. The MSCAN08 has three transmit buffers.

The second requirement calls for some sort of internal priorisation which the MSCAN08 implements with the “local priority” concept described below.

C.3.2 Receive Structures

The received messages are stored in a two stage input FIFO. The two message buffers are mapped using a ‘ping pong’ arrangement into a single memory area (see Figure C-2). While the background receive buffer (RxBG) is exclusively associated to the MSCAN08, the foreground receive buffer (RxFG) is addressable by the CPU08. This scheme simplifies the handler software as only one address area is applicable for the receive process.

Both buffers have a size of 13 byte to store the CAN control bits, the identifier (standard or extended) and the data content (for details see Section C.11).

The Receiver Full flag (RXF) in the MSCAN08 Receiver Flag Register (CRFLG) (see Section C.12.6) signals the status of the foreground receive buffer. When the buffer contains a correctly received message with matching identifier this flag is set.

After the MSCAN08 successfully received a message into the background buffer it copies the content of RxBG into RxFG†, sets the RXF flag, and emits a receive interrupt to the CPU‡. A new message - which may follow immediately after the IFS field of the CAN frame - will be received into RxBG.

The user’s receive handler has to read the received message from RxFG and to reset the RXF flag in order to acknowledge the interrupt and to release the foreground buffer.

An overrun conditions occurs when both the foreground and the background receive message buffers are filled with correctly received messages, and a further message is being received from the bus. The latter message will be discarded and an error interrupt with overrun indication will occur if enabled. The over-writing of the background buffer is independent of the identifier filter function. While in the overrun situation, the MSCAN08 will stay synchronized to the CAN bus and is able to transmit messages but will discard all incoming messages.

Note: MSCAN08 will receive its own messages into the background receive buffer RxBG but will NOT overwrite RxFG and will NOT emit a receive interrupt nor will it acknowledge (ACK) its own messages on the CAN bus. The exception to this rule is that when in loop-back mode MSCAN08 will treat its own messages exactly like all other incoming messages.

† Only if the RXF flag is not set.

‡ The receive interrupt will occur only if not masked. A polling scheme can be applied on RXF also.

C

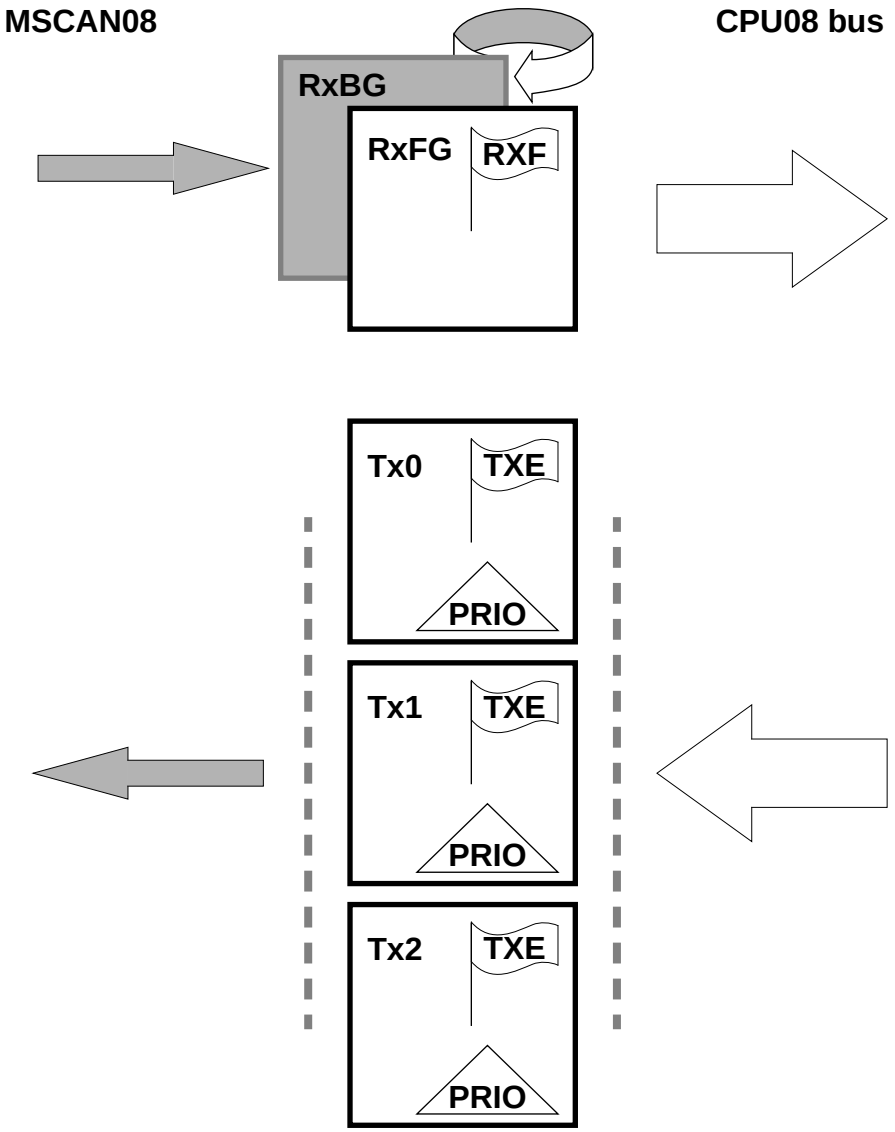


Figure C-2 User Model for Message Buffer Organization

C

C.3.3 Transmit Structures

The MSCAN08 has a triple transmit buffer scheme in order to allow multiple messages to be set up in advance and to achieve an optimized real-time performance. The three buffers are arranged as shown in Figure C-2.

All three buffers have a 13 byte data structure similar to the outline of the receive buffers (see Section C.11). An additional Transmit Buffer Priority Register (TBPR) contains an 8-bit so called Local Priority field (PRIO) (see Section C.11.5).

In order to transmit a message, the CPU08 has to identify an available transmit buffer which is indicated by a set Transmit Buffer Empty (TXE) Flag in the MSCAN08 Transmitter Flag Register (CTFLG) (see Section C.12.8).

The CPU08 then stores the Identifier, the control bits and the data content into one of the transmit buffers. Finally, the buffer has to be flagged as being ready for transmission by clearing the TXE flag.

The MSCAN08 will then schedule the message for transmission and will signal the successful transmission of the buffer by setting the TXE flag. A transmit interrupt will be emitted† when TXE is set and can be used to drive the application software to re-load the buffer.

In case more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN08 uses the local priority setting of the three buffers for priorisation. For this purpose every transmit buffer has an 8-bit local priority field (PRIO). The application software sets this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being emitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority.

The internal scheduling process takes places whenever the MSCAN08 arbitrates for the bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software it may become necessary to abort a lower priority message being set up in one of the three transmit buffers. As messages that are already under transmission can not be aborted, the user has to request the abort by setting the corresponding Abort Request Flag (ABTRQ) in the Transmission Control Register (CTCR). The MSCAN08 will then grant the request if possible by setting the corresponding Abort Request Acknowledge (ABTAK) and the TXE flag in order to release the buffer and by emitting a transmit interrupt. The transmit interrupt handler software can tell from the setting of the ABTAK flag whether the message was actually aborted (ABTAK=1) or has been sent in the meantime (ABTAK=0).

† The transmit interrupt will occur only if not masked. A polling scheme can be applied on TXE also.

C.4 Identifier acceptance Filter

A very flexible programmable generic identifier acceptance filter has been introduced in order to reduce the CPU interrupt loading. The filter is programmable to operate in three different modes:

- Single identifier acceptance filter to be applied to the full 29 bits of the identifier and to the following bits of the CAN frame: RTR, IDE, SRR. This mode implements a single filter for a full length CAN 2.0B compliant extended identifier.
- Double identifier acceptance filter to be applied to
  - the 11 bits of the identifier and the RTR bit of CAN 2.0A mesages or
  - the 14 most significant bits of the identifier of CAN 2.0B messages.
- Quadruple identifier acceptance filter to be applied to the first 8 bits of the identifier. This mode implements four independent filters for the first 8 bit of a CAN 2.0A compliant standard identifier.

The Identifier Acceptance Registers (CIAR) defines the acceptable pattern of the standard or extended identifier (ID10 - ID0 or ID28 - ID0). Any of these bits can be marked *don't care* in the Identifier Mask Register (CIMR).

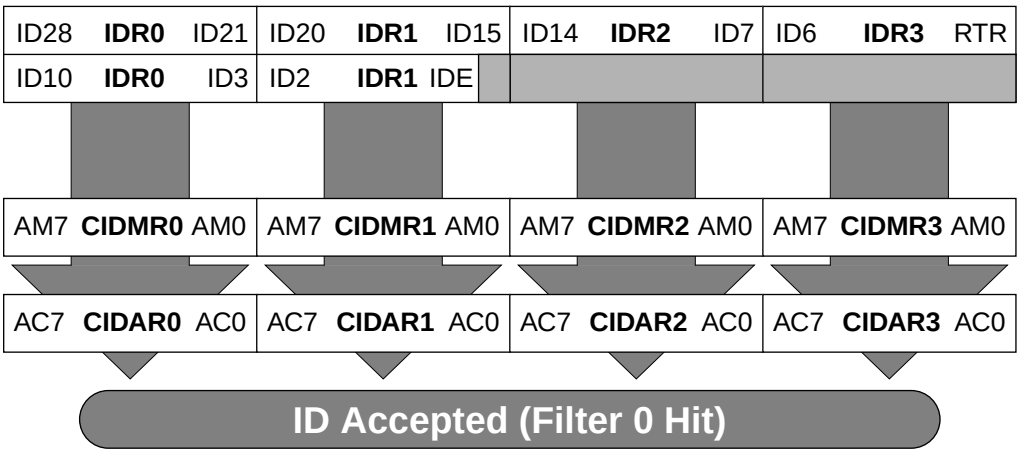


Figure C-3 Single 32 bit Maskable Identifier Acceptance Filter

The background buffer RxBG will be copied into the foreground buffer RxFG and the RXF flag will be set only in case of an accepted identifier (an identifier acceptance filter hit). A hit will also cause a receiver interrupt if enabled.

C

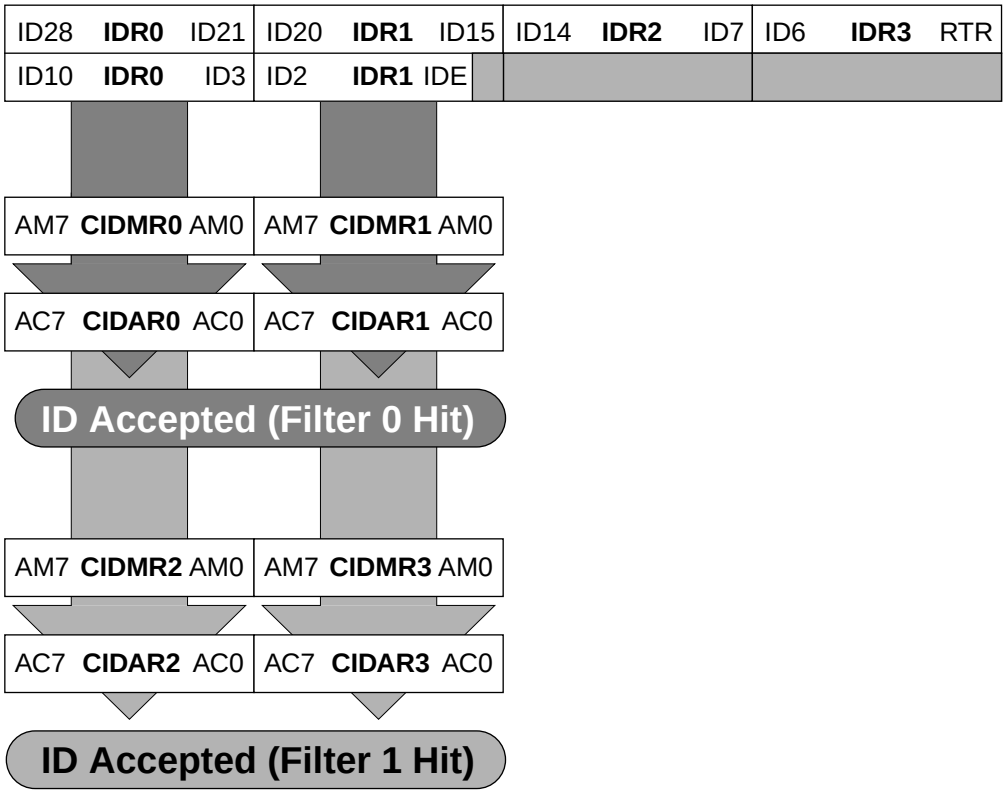


Figure C-4 Dual 16 bit Maskable Acceptance Filters

A filter hit is indicated to the application software by a set RXF (Receive Buffer Full Flag, see Section C.12.6) and two bits in the Identifier Acceptance Control Register (see Section C.12.10). These Identifier Hit Flags (IDHIT1-0) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. In case that more than one hit occurs (two or more filters match) the lower hit has priority.

C

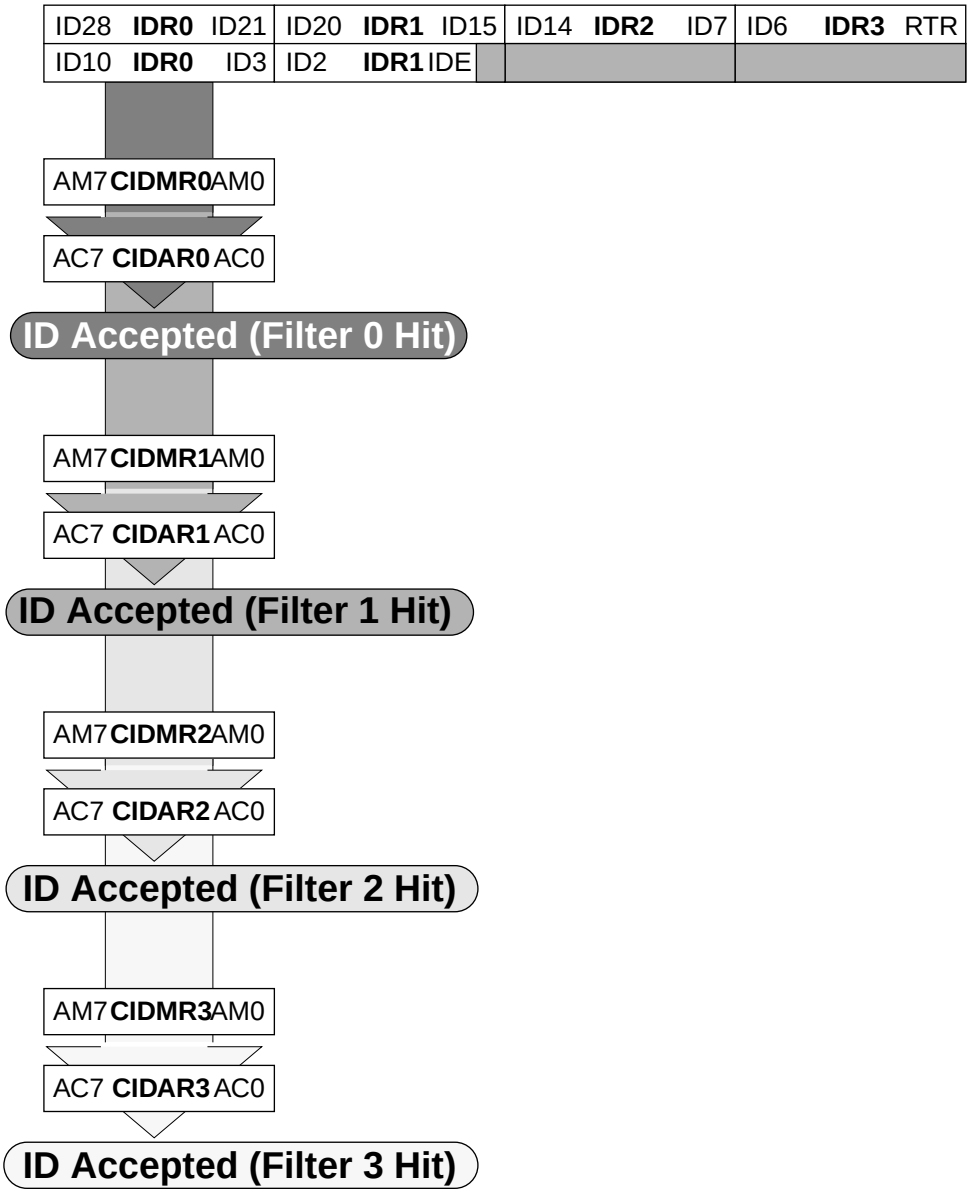


Figure C-5 Quadruple 8 bit Maskable Acceptance Filters

C

C.5 Interrupts

The MSCAN08 supports four interrupt vectors mapped onto eleven different interrupt sources, any of which can be individually masked (for details see Section C.12.6 to Section C.12.9):

- *Transmit Interrupt*: At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXE flags of the empty message buffers are set.
- *Receive Interrupt*: A message has been successfully received and loaded into the foreground receive buffer. This interrupt will be emitted immediately after receiving the EOF symbol. The RXF flag is set.
- *Wake-Up Interrupt*: An activity on the CAN bus occurred during MSCAN08 internal sleep mode.
- *Error Interrupt*: An overrun, error or warning condition occurred. The Receiver Flag Register (CRFLG) will indicate one of the following conditions:
  - *Overrun:* An overrun condition as described in Section C.3.2 has occurred.
  - *Receiver Warning*: The Receive Error Counter has reached the CPU Warning limit of 96.
  - *Transmitter Warning*: The Transmit Error Counter has reached the CPU Warning limit of 96.
  - *Receiver Error Passive*: The Receive Error Counter has exceeded the Error Passive limit of 127 and MSCAN08 has gone to Error Passive state.
  - *Transmitter Error Passive*: The Transmit Error Counter has exceeded the Error Passive limit of 127 and MSCAN08 has gone to Error Passive state.
  - *Bus Off*: The Transmit Error Counter has exceeded 255 and MSCAN08 has gone to Bus Off state.

C.5.1 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the MSCAN08 Receiver Flag Register (CRFLG) or the MSCAN08 Transmitter Control Register (CTCR). Interrupts are pending as long as one of the corresponding flags is set. The flags in above registers must be reset within the interrupt handler in order to handshake the interrupt. The flags are reset through writing a ‘1’ to the corresponding bit position. A flag can not be cleared if the respective condition still prevails.

Caution: Bit manipulation instructions (BSET) shall not be used to clear interrupt flags.

C

C.5.2 Interrupt Vectors

The MSCAN08 supports four interrupt vectors as shown in Table C-1. The vector addresses are dependent on the chip integration and to be defined. The relative interrupt priority is also integration dependent and to be defined.

Table C-1 MSCAN08 Interrupt Vectors

Function	Source	Local Mask	Global Mask
Wake-Up	WUPIF	WUPIE	I Bit
Error Interrupts	RWRNIF	RWRNIE	
	TWRNIF	TWRNIE	
	RERRIF	RERRIE	
	TERRIF	TERRIE	
	BOFFIF	BOFFIE	
	OVRIF	OVRIE	
Receive	RXF	RXFIE	
Transmit	TXE0	TXEIE0	
	TXE1	TXEIE1	
	TXE2	TXEIE2	

C.6 Protocol Violation Protection

The MSCAN08 will protect the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters can not be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN08 can not be modified while the MSCAN08 is on-line. The SFTRES bit in the MSCAN08 Module Control Register (see Section C.12.2) serves as a lock to protect the following registers:
  - MSCAN08 Module Control Register 1 (CMCR1)
  - MSCAN08 Bus Timing Register 0 and 1 (CBTR0, CBTR1)
  - MSCAN08 Identifier Acceptance Control Register (CIDAC)
  - MSCAN08 Identifier Acceptance Registers (CIDAR0-3)
  - MSCAN08 Identifier Mask Registers (CIDMR0-3)
- The TxCAN pin is forced to Recessive if the CPU goes into STOP mode.

C

C.7 Low Power Modes

The MSCAN08 has three modes with reduced power consumption compared to Normal Mode. In Sleep and Soft Reset Mode, power consumption is reduced by stopping all clocks except those to access the registers. In Power Down Mode, all clocks are stopped and no power is consumed.

The WAIT and STOP instruction put the MCU in low power consumption stand-by mode.Table C-2 summarizes the combinations of MSCAN08 and CPU modes. A particular combination of modes is entered for the given settings of the bits SLPAK and SFTRES. For all modes, an MSCAN08 wake-up interrupt can occur only if SLPAK=WUPIE=1. While the CPU is in Wait Mode, the MSCAN08 is operated as in Normal Mode.

Table C-2 MSCAN08 vs. CPU operating modes

MSCAN Mode	CPU Mode	
	STOP	WAIT or RUN
Power Down	SLPAK = X(1) SFTRES = X	
Sleep		SLPAK = 1 SFTRES = 0
Soft Reset		SLPAK = 0 SFTRES = 1
Normal		SLPAK = 0 SFTRES = 0

(1) 'X' means don't care.

C.7.1 MSCAN08 Internal Sleep Mode

The CPU can request the MSCAN08 to enter this low-power mode by asserting the SLPRQ bit in the Module Configuration Register (see Figure C-6). The time when the MSCAN08 will then enter Sleep Mode depends on its current activity:

- if it is transmitting, it will continue to transmit until there is no more message to be transmitted, and then go into Sleep Mode
- if it is receiving, it will wait for the end of this message and then go into Sleep Mode
- if it is neither transmitting nor receiving, it will immediately go into Sleep Mode

The application software must avoid to set up a transmission (by clearing one or more TXE flag(s)) and immediately request Sleep Mode (by setting SLPRQ). It will then depend on the exact sequence of operations whether the MSCAN08 will start transmitting or go into Sleep Mode directly.

During Sleep Mode, the SLPAK flag is set. The application software should use this flag as a handshake indication for the request to go into Sleep Mode. When in Sleep Mode, the MSCAN08 stops its own clocks and the TxCAN pin will stay in recessive state.

The MSCAN08 will leave Sleep Mode (wake-up) when

- bus activity occurs or
- the MCU clears the SLPRQ bit.

Note: The MCU can not clear the SLPRQ bit before the MSCAN08 is in Sleep Mode (SLPAK=1).

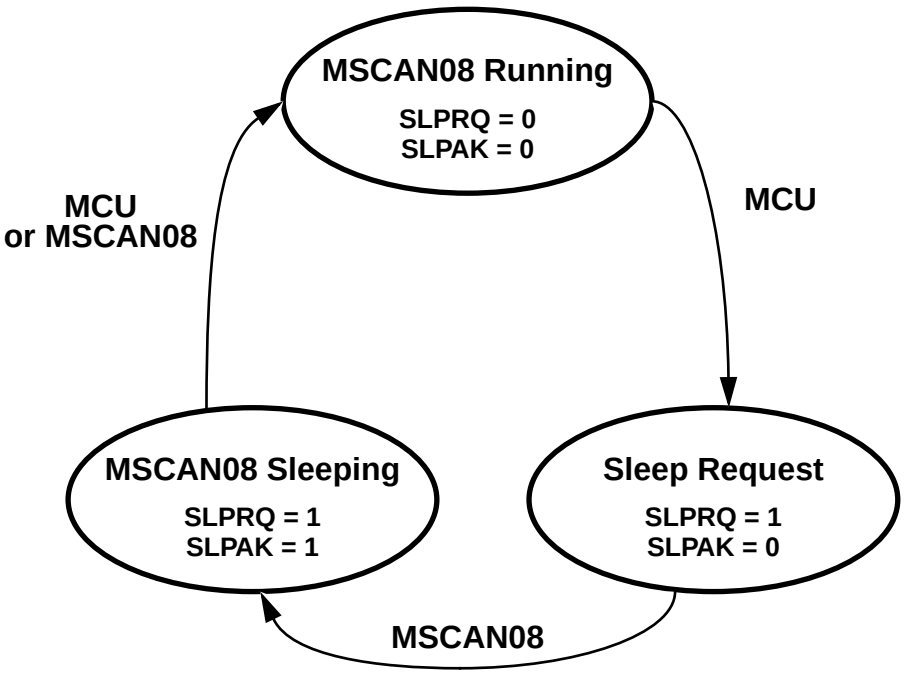


Figure C-6 Sleep Request / Acknowledge Cycle

C

C.7.2 MSCAN08 Soft Reset Mode

In Soft Reset Mode, the MSCAN08 is stopped. Registers can still be accessed. This mode is used to initialize the module configuration, bit timing, and the CAN message filter. See Section C.12.2 for a complete description of the Soft Reset Mode.

C.7.3 MSCAN08 Power Down Mode

The MSCAN08 is in Power Down Mode when the CPU is in Stop Mode.

When entering the Power Down Mode, the MSCAN08 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations. The user is responsible to take care that the MSCAN08 is not active when Power Down Mode is entered. The recommended procedure is to bring the MSCAN08 into Sleep Mode before the STOP instruction is executed.

To protect the CAN bus system from fatal consequences of violations to above rule, the MSCAN08 will drive the TxCAN pin into recessive state.

C.7.4 CPU Wait Mode

The MSCAN08 module remains active during CPU wait mode. The MSCAN08 will stay synchronized to the CAN bus and will generate enabled transmit, receive and error interrupts to the CPU. Any such interrupt will bring the MCU out of wait mode.

C.7.5 Programmable Wake-Up Function

The MSCAN08 can be programmed to apply a low-pass filter function to the RxCAN input line while in internal Sleep Mode (see control bit WUPM in the Module Control Register, Section C.12.2). This feature can be used to protect the MSCAN08 from wake-up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic inference within noisy environments.

C.8 Timer Link

The MSCAN08 will generate a timer signal whenever a valid frame has been received. Because the CAN specification defines a frame to be valid if no errors occurred before the EOF field has been transmitted successfully, the timer signal will be generated right after the EOF. A pulse of one bit time is generated. As the MSCAN08 receiver engine receives also the frames being sent by itself, a timer signal will also be generated after a successful transmission.

The previously described timer signal can be routed into the on-chip Timer Interface Module (TIM). Under the control of the Timer Link Enable (TLNKEN) bit in the CMCR0 will this signal be connected to the Timer n Channel m input†.

After Timer n has been programmed to capture rising edge events it can be used to generate 16-bit time stamps which can be stored under software control with the received message.

C.9 Clock System

Figure C-7 shows the structure of the MSCAN08 clock generation circuitry and its interaction with the Clock Generation Module (CGM). With this flexible clocking scheme the MSCAN08 is able to handle CAN bus rates ranging from 10 kbps up to 1 Mbps.

C

† The timer channel being used for the timer link is integration dependent.

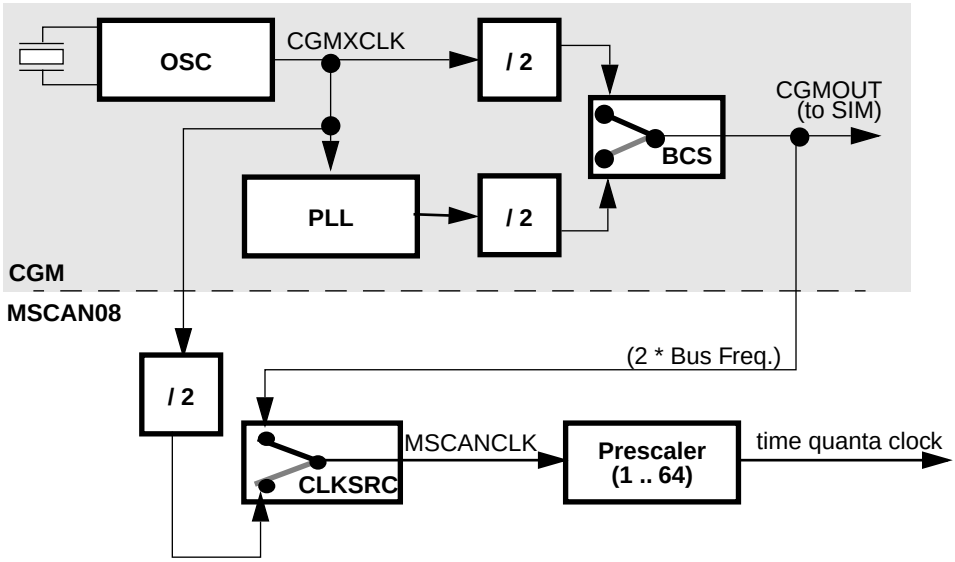


Figure C-7 Clocking Scheme

The Clock Source Flag (CLKSRC) in the MSCAN08 Module Control Register (CMCR1) (see Section C.12.3) defines whether the MSCAN08 is connected to the output of the crystal oscillator or to the Clock Generator Module output.

A programmable prescaler is used to generate from the MSCAN08 clock the time quanta (Tq) clock. A time quantum is the atomic unit of time handled by the MSCAN08. A bit time is subdivided into three segments†:

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time segment 2: This segment represents the PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

The Synchronization Jump Width can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

Above parameters can be set by programming the Bus Timing Registers (CBTR0-1, see Section C.12.4 and Section C.12.5).

† For further explanation of the under-lying concepts please refer to ISO/DIS 11519-1, Section 10.3.

C

It is the user's responsibility to make sure that his bit time settings are in compliance with the CAN standard. Figure C-8 and Table C-3 give an overview on the CAN conforming segment settings and the related parameter values.

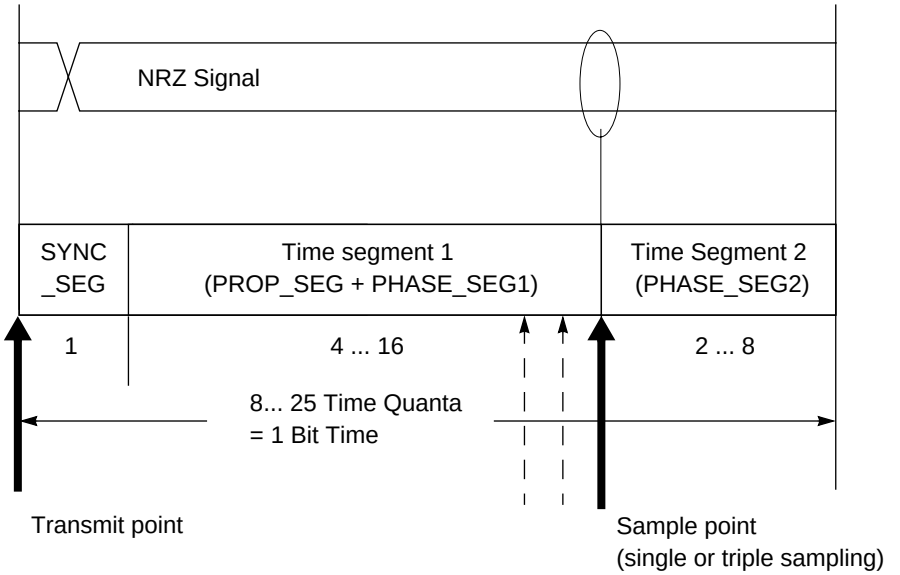


Figure C-8 Segments within the Bit Time

Table C-3 CAN Standard Compliant Bit Time Segment Settings

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

C

C.10 Memory Map

The MSCAN08 occupies 128 Bytes in the CPU08 memory space. The absolute mapping is implementation dependent with the base address being a multiple of 128. The background receive buffer can only be read in test mode.

Table C-4 MSCAN08 Memory Map

$xx00 $xx08	CONTROL REGISTERS 9 BYTES
$xx09 $xx0D	RESERVED 5 BYTES
$xx0E $xx0F	ERROR COUNTERS 2 BYTES
$xx10 $xx17	IDENTIFIER FILTER 8 BYTES
$xx18 $xx3F	RESERVED 40 BYTES
$xx40 $xx4F	RECEIVE BUFFER
$xx50 $xx5F	TRANSMIT BUFFER 0
$xx60 $xx6F	TRANSMIT BUFFER 1
$xx70 $xx7F	TRANSMIT BUFFER 2

C.11 Programmer's Model of message storage

The following section details the organisation of the receive and transmit message buffers and the associated control registers. For reasons of programmer interface simplification the receive and transmit message buffers have the same outline. Each message buffer allocates 16 byte in the memory map containing a 13 byte data structure. An additional Transmit Buffer Priority Register (TBPR) is defined for the transmit buffers.

Table C-5 Message Buffer Organisation

Address	Register Name
xxb0	Identifier Register 0
xxb1	Identifier Register 1
xxb2	Identifier Register 2
xxb3	Identifier Register 3
xxb4	Data Segment Register 0
xxb5	Data Segment Register 1
xxb6	Data Segment Register 2
xxb7	Data Segment Register 3
xxb8	Data Segment Register 4
xxb9	Data Segment Register 5
xxbA	Data Segment Register 6
xxbB	Data Segment Register 7
xxbC	Data Length Register
xxbD	Transmit Buffer Priority Register(1)
xxbE	unused
xxbF	unused

(1) Not Applicable for Receive Buffers

C.11.1 Message Buffer Outline

Figure C-9 shows the common 13 byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in Figure C-10. All bits of the 13 byte data structure are undefined out of reset.

C.11.2 Identifier Registers (IDRn)

The identifiers consist of either 11 bits (ID10 – ID0) for the standard, or 29 bits (ID28 - ID0) for the extended format. ID10/28 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

Register	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Identifier register 0 (IDR0)	$xxb0	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
Identifier register 1 (IDR1)	$xxb1	ID20	ID19	ID18	SRR(1)	IDE(1)	ID17	ID16	ID15
Identifier register 2 (IDR2)	$xxb2	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
Identifier register 3 (IDR3)	$xxb3	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
Data segment register 0 (DSR0)	$xxb4	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 1 (DSR1)	$xxb5	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 2 (DSR2)	$xxb6	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 3 (DSR3)	$xxb7	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 4 (DSR4)	$xxb8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 5 (DSR5)	$xxb9	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 6 (DSR6)	$xxbA	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 7 (DSR7)	$xxbB	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data length register (DLR)	$xxbC					DLC3	DLC2	DLC1	DLC0

Figure C-9 Receive/transmit message buffer extended identifier registers

SRR - Substitute Remote Request

This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and will be stored as received on the CAN bus for receive buffers

IDE — ID Extended

This flag indicates whether the extended or standard identifier format is applied in this buffer. In case of a receive buffer the flag is set as being received and indicates to the CPU how to process the buffer identifier registers. In case of a transmit buffer the flag indicates to the MSCAN08 what type of identifier to send.

1 (set) – Extended format (29 bit)

0 (clear) – Standard format (11 bit)

C

RTR — Remote transmission request

This flag reflects the status of the Remote Transmission Request bit in the CAN frame. In case of a receive buffer it indicates the status of the received frame and allows to support the transmission of an answering frame in software. In case of a transmit buffer this flag defines the setting of the RTR bit to be sent.

1 (set) – Remote frame

0 (clear) – Data frame

Register	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Identifier register 0 (IDR0)	$xxb0	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
Identifier register 1 (IDR1)	$xxb1	ID2	ID1	ID0	RTR	IDE(0)			
Identifier register 2 (IDR2)	$xxb2								
Identifier register 3 (IDR3)	$xxb3								

Figure C-10 Standard identifier mapping registers

C.11.3 Data Length Register (DLR)

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

DLC3 – DLC0 — Data length code bits

The data length code contains the number of bytes (data byte count) of the respective message. At transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. Table C-6 shows the effect of setting the DLC bits.

Table C-6 Data length codes

Data length code				Data byte count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

C

C.11.4 Data Segment Registers (DSRn)

The eight data segment registers contain the data to be transmitted or being received. The number of bytes to be transmitted or being received is determined by the data length code in the corresponding DLR.

C.11.5 Transmit Buffer Priority Registers (TBPR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Transmit buffer priority reg. (TBPR)	$xxbD	PRIO7	PRIO6	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0	Undefined

PRIO7 - PRIO0— Local Priority

This field defines the local priority of the associated message buffer. The local priority is used for the internal priorisation process of the MSCAN08 and is defined to be highest for the smallest binary number. The MSCAN08 implements the following internal priorisation mechanism:

- All transmission buffers with a cleared TXE flag participate in the priorisation right before the SOF (Start of Frame) is sent.
- The transmission buffer with the lowest local priority field wins the priorisation.
- In case of more than one buffer having the same lowest priority the message buffer with the lower index number wins.

Caution: To ensure data integrity, no registers of the transmit buffers shall be written while the associated TXE flag is cleared.

Caution: To ensure data integrity, no registers of the receive buffer shall be read while the RXF flag is cleared.

C.12 Programmer's Model of Control Registers

C.12.1 Overview

The programmer's model has been laid out for maximum simplicity and efficiency. Table C-7 gives an overview on the control register block of the MSCAN08:

Table C-7 MSCAN08 Control register structure

Register	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
CMCR0	$xx00	0	0	0	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
CMCR1*	$xx01	0	0	0	0	0	LOOPB	WUPM	CLKSRC
CBTR0*	$0002	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
CBTR1*	$xx03	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
CRFLG	$xx04	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
CRIER	$xx05	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
CTFLG	$xx06	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
CTCR	$xx07	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
CIDAC*	$xx08	0	0	IDAM1	IDAM0	0	0	IDHIT1	IDHIT0
Reserved	$xx09-$xx0D								
CRXERR	$xx0E	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
CTXERR	$xx0F	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
CIDAR0*	$xx10	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
CIDAR1*	$xx11	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
CIDAR2*	$xx12	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
CIDAR3*	$xx13	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
CIDMRO*	$xx14	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
CIDMR1*	$xx15	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
CIDMR2*	$xx16	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
CIDMR3*	$xx17	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0

*. Writeable only when SFTRES is set.

C.12.2 MSCAN08 Module Control Register (CMCR0).

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Module control reg. 0 (CMCR0)	$xx00	0	0	0	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES	0000 0001

SYNCH — Synchronized Status

This bit indicates whether the MSCAN08 is synchronized to the CAN bus and as such can participate in the communication process.

1 (set) – MSCAN08 is synchronized to the CAN bus

0 (clear) – MSCAN08 is not synchronized to the CAN bus

TLNKEN - Timer Enable

This flag is used to establish a link between the MSCAN08 and the on-chip timer (see Section C.8).

1 (set) – The MSCAN08 timer signal output is connected to the timer.

0 (clear) – No connection.

SLPAK — Sleep Mode Acknowledge

This flag indicates whether the MSCAN08 is in module internal sleep mode. It shall be used as a handshake for the sleep mode request (see Section C.7.1).

1 (set) – Sleep – The MSCAN08 is in internal sleep mode.

0 (clear) – Wake-up – The MSCAN08 will function normally.

SLPRQ — Sleep request, Go to internal sleep mode

This flag allows to request the MSCAN08 to go into an internal power-saving mode (see Section C.7.1).

1 (set) – Sleep – The MSCAN08 will go into internal sleep mode.

0 (clear) – Wake-up – The MSCAN08 will function normally.

SFTRES— Soft Reset

When this bit is set by the CPU, the MSCAN08 immediately enters the soft reset state. Any ongoing transmission or reception is aborted and synchronisation to the bus is lost.

The following registers will go into and stay in the same state as out of hard reset: CMCR0, CRFLG, CRIER, CTFLG, CTCR.

The registers CMCR1, CBTR0, CBTR1, CIDAC, CIDAR0-3, CIDMR0-3 can only be written by the CPU when the MSCAN08 is in soft reset state. The values of the error counters are not affected by soft reset.

When this bit is cleared by the CPU, the MSCAN08 will try to synchronize to the CAN bus: If the MSCAN08 is not in bus-off state it will be synchronized after 11 recessive bits on the bus; if the MSCAN08 is in bus-off state it continues to wait for 128 occurrences of 11 recessive bits.

Clearing SFTRES and writing to other bits in CMCR0 must be in separate instructions.

1 (set) – MSCAN08 in soft reset state.

0 (clear) – Normal operation

C.12.3 MSCAN08 Module Control Register (CMCR1)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Module control reg. 1 (CMCR1)	$xx01	0	0	0	0	0	LOOPB	WUPM	CLKSRC	0000 0000

LOOPB - Loop Back Self Test Mode

When this bit is set the MSCAN08 performs an internal loop back which can be used for self test operation: the bit stream output of the transmitter is fed back to the receiver. The RxCAN input pin is ignored and the TxCAN output goes to the recessive state (1). Note that in this state the MSCAN08 ignores the ACK bit to insure proper reception of its own message and will treat messages being received while in transmission as received messages from remote nodes.

1 (set) – Activate loop back self test mode

0 (clear) – Normal operation

WUPM - Wake-Up Mode

This flag defines whether the integrated low-pass filter is applied to protect the MSCAN08 from spurious wake-ups (see Section C.7.5).

1 (set) – MSCAN08 will wake up the CPU only in case of dominant pulse on the bus which has a length of at least approximately T_{wup}.

0 (clear) – MSCAN08 will wake up the CPU after any recessive to dominant edge on the CAN bus.

CLKSRC - Clock Source

This flag defines which clock source the MSCAN08 module is driven from (see Section C.9).

1 (set) – THE MSCAN08 clock source is CGMOUT (see Figure C-7).

0 (clear) – The MSCAN08 clock source is CGMXCLK/2 (see Figure C-7).

Note: The CMCR1 register can only be written if the SFTRES bit in the MSCAN08 Module Control Register is set

C

C.12.4 MSCAN08 Bus Timing Register 0 (CBTR0)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Bus timing reg. 0 (CBTR0)	Sxx02	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	0000 0000

SJW1, SJW0 — Synchronization Jump Width

The synchronization jump width defines the maximum number of time quanta (Tq) clock cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see Table C-8).

Table C-8 Synchronization jump width

SJW1	SJW0	Synchronization jump width
0	0	1 Tq clock cycle
0	1	2 Tq clock cycles
1	0	3 Tq clock cycles
1	1	4 Tq clock cycles

BRP5 – BRP0 — Baud Rate Prescaler

These bits determine the time quanta (Tq) clock, which is used to build up the individual bit timing, according to Table C-9.

Table C-9 Baud rate prescaler

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

Note: The CBTR0 register can only be written if the SFTRES bit in the MSCAN08 Module Control Register is set.

C

C.12.5 MSCAN08 Bus Timing Register 1 (CBTR1)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Bus timing reg. 1 (CBTR1)	$xx03	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10	0000 0000

SAMP — Sampling

This bit determines the number of samples of the serial bus to be taken per bit time. If set three samples per bit are taken, the regular one (sample point) and two preceding samples, using a majority rule. For higher bit rates SAMP should be cleared, which means that only one sample will be taken per bit.

1 (set) – Three samples per bit.

0 (clear) – One sample per bit.

TSEG22 – TSEG10 — Time Segment

Time segments within the bit time fix the number of clock cycles per bit time, and the location of the sample point.

Table C-10 Time segment syntax

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

Time segment 1 (TSEG1) and time segment 2 (TSEG2) are programmable as shown in Table C-11

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (Tq) clock cycles per bit (as shown in Table C-11).

Note: The CBTR1 register can only be written if the SFTRES bit in the MSCAN08 Module Control Register is set

C

Table C-11 Time segment values

TSEG 13	TSEG 12	TSEG 11	TSEG 10	Time segment 1
0	0	0	0	1 Tq clock cycle
0	0	0	1	2 Tq clock cycles
0	0	1	0	3 Tq clock cycles
0	0	1	1	4 Tq clock cycles
.	.	.	.	.
.	.	.	.	.
1	1	1	1	16 Tq clock cycles

TSEG 22	TSEG 21	TSEG 20	Time segment 2
0	0	0	1 Tq clock cycle
0	0	1	2 Tq clock cycles
.	.	.	.
.	.	.	.
1	1	1	8 Tq clock cycles

C.12.6 MSCAN08 Receiver Flag Register (CRFLG)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Receiver flag register (CRFLG)	$xx04	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF	0000 0000

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. A flag can only be cleared when the condition which caused the setting is no more valid. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CRIER register. A hard or soft reset will clear the register.

WUPIF — Wake-up Interrupt Flag

If the MSCAN08 detects bus activity whilst it is asleep, it clears the SLPAK bit in the CMCR0 register; the WUPIF bit will then be set. If not masked, a Wake-Up interrupt is pending while this flag is set.

1 (set) – MSCAN08 has detected activity on the bus and requested wake-up.

0 (clear) – No wake-up activity has been observed while in sleep mode.

RWRNIF — Receiver Warning Interrupt Flag

This bit will be set when the MSCAN08 went into warning status due to the Receive Error counter being in the range of 96 to 127. If not masked, an Error interrupt is pending while this flag is set.

1 (set) – MSCAN08 went into receiver warning status.

0 (clear) – No receiver warning status has been reached.

TWRNIF — Transmitter Warning Interrupt Flag

This bit will be set when the MSCAN08 went into warning status due to the Transmit Error counter being in the range of 96 to 127. If not masked, an Error interrupt is pending while this flag is set.

1 (set) – MSCAN08 went into transmitter warning status.

0 (clear) – No transmitter warning status has been reached.

RERRIF — Receiver Error Passive Interrupt Flag

This bit will be set when the MSCAN08 went into error passive status due to the Receive Error counter exceeded 127. If not masked, an Error interrupt is pending while this flag is set.

1 (set) – MSCAN08 went into receiver error passive status.

0 (clear) – No receiver error passive status has been reached.

TERRIF — Transmitter Error Passive Interrupt Flag

This bit will be set when the MSCAN08 went into error passive status due to the Transmit Error counter exceeded 127. If not masked, an Error interrupt is pending while this flag is set.

1 (set) – MSCAN08 went into transmitter error passive status.

0 (clear) – No transmitter error passive status has been reached.

BOFFIF — Bus-Off Interrupt Flag

This bit will be set when the MSCAN08 went into bus-off status, due to the Transmit Error counter exceeded 255. If not masked, an Error interrupt is pending while this flag is set.

1 (set) – MSCAN08 went into bus-off status.

0 (clear) – No bus-off status has been reached.

OVRIF — Overrun Interrupt Flag

This bit will be set when a data overrun condition occurred. If not masked, an Error interrupt is pending while this flag is set.

1 (set) – A data overrun has been detected.

0 (clear) – No data overrun has occurred.

C

RXF — Receive Buffer Full

The RXF flag is set by the MSCAN08 when a new message is available in the foreground receive buffer. This flag indicates whether the buffer is loaded with a correctly received message. After the CPU has read that message from the receive buffer the RXF flag must be handshaked to release the buffer. A set RXF flag prohibits the exchange of the background receive buffer into the foreground buffer. In that case the MSCAN08 will signal an overload condition. If not masked, a Receive interrupt is pending while this flag is set.

1 (set) – The receive buffer is full. A new message is available.

0 (clear) – The receive buffer is released (not full).

Note: The CRFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.

C.12.7 MSCAN08 Receiver Interrupt Enable Register (CRIER)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Receiver interrupt enable reg. (CRIER)	$xx05	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RFXIE	0000 0000

WUPIE — Wake-up Interrupt Enable

1 (set) – A wake-up event will result in a wake-up interrupt.

0 (clear) – No interrupt will be generated from this event.

RWRNIE — Receiver Warning Interrupt Enable

1 (set) – A receiver warning status event will result in an error interrupt.

0 (clear) – No interrupt will be generated from this event.

TWRNIE — Transmitter Warning Interrupt Enable

1 (set) – A transmitter warning status event will result in an error interrupt.

0 (clear) – No interrupt will be generated from this event.

RERRIE — Receiver Error Passive Interrupt Enable

1 (set) – A receiver error passive status event will result in an error interrupt.

0 (clear) – No interrupt will be generated from this event.

TERRIE — Transmitter Error Passive Interrupt Enable

1 (set) – A transmitter error passive status event will result in an error interrupt.

0 (clear) – No interrupt will be generated from this event.

BOFFIE — Bus-Off Interrupt Enable

1 (set) – A bus-off event will result in an error interrupt.

0 (clear) – No interrupt will be generated from this event.

OVRIE — Overrun Interrupt Enable

1 (set) – An overrun event will result in an error interrupt.

0 (clear) – No interrupt will be generated from this event.

RXFIE — Receiver Full Interrupt Enable

1 (set) – A receive buffer full (successful message reception) event will result in a receive interrupt.

0 (clear) – No interrupt will be generated from this event.

Note: The CRIER register is held in the reset state when the SFTRES bit in CMCR0 is set.

C

C.12.8 MSCAN08 Transmitter Flag Register (CTFLG)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Transmitter flag reg. (CTFLG)	$xx06	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0	0000 0111

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CTCR register. A hard or soft reset will clear the register.

ABTAK2 - ABTAK0 — Abort Acknowledge

This flag acknowledges that a message has been aborted due to a pending abort request from the CPU. After a particular message buffer has been flagged empty, this flag can be used by the application software to identify whether the message has been aborted successfully or has been sent in the meantime. The flag is reset implicitly whenever the associated TXE flag is set to 0.

1 (set) – The message has been aborted.

0 (clear) – The massage has not been aborted, thus has been sent out.

TXE2 - TXE0 —Transmitter Buffer Empty

This flag indicates that the associated transmit message buffer is empty, thus not scheduled for transmission. The CPU must handshake (clear) the flag after a message has been set up in the transmit buffer and is due for transmission. The MSCAN08 will set the flag after the message has been sent successfully. The flag will also be set by the MSCAN08 when the transmission request was successfully aborted due to a pending abort request (Section C.12.9). If not masked, a Transmit interrupt is pending while this flag is set.

A reset of this flag will also reset the Abort Acknowledge (ABTAK, see above) and the Abort Request (ABTRQ, see Section C.12.9) flags of the particular buffer.

1 (set) – The associated message buffer is empty (not scheduled).

0 (clear) – The associated message buffer is full (loaded with a message due for transmission).

Note: The CTFLG register is held in the reset state when te SFTRES bit in CMCR0 is set.

C.12.9 MSCAN08 Transmitter Control Register (CTCR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Transmitter control reg. (CTCR)	$xx07	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0	0000 0000

ABTRQ2 - ABTRQ0 — Abort Request

The CPU sets this bit to request that an already scheduled message buffer (TXE = 0) shall be aborted. The MSCAN08 will grant the request when the message is not already under transmission. When a message is aborted the associated TXE and the Abort Acknowledge flag (ABTAK, see Section C.12.8) will be set and an TXE interrupt will occur if enabled. The CPU can not reset ABTRQx. ABTRQx is reset implicitly whenever the associated TXE flag is set.

1 (set) – Abort request pending.

0 (clear) – No abort request.

TXEIE2 - TXEIE0 — Transmitter Empty Interrupt Enable

1 (set) – A transmitter empty (transmit buffer available for transmission) event will result in a transmitter empty interrupt.

0 (clear) – No interrupt will be generated from this event.

Note: The CTCR register is held in the reset state when the SFTRES bit in CMCR0 is set.

C.12.10 MSCAN08 Identifier Acceptance Control Register (CIDAC)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Identifier acceptance control reg. (CIDAC)	$xx08	0	0	IDAM1	IDAM0	0	0	IDHIT1	IDHIT0	0000 0000

IDAM1- IDAM0— Identifier Acceptance Mode

The CPU sets these flags to define the identifier acceptance filter organisation (see Section C.4). Table C-12 summarizes the different settings. In Filter Closed mode no messages will be accepted such that the foreground buffer will never be reloaded.

Table C-12 Identifier Acceptance Mode Settings

IDAM1	IDAM0	Identifier Acceptance Mode
0	0	Single 32 bit Acceptance Filter
0	1	Two 16 bit Acceptance Filter
1	0	Four 8 bit Acceptance Filters
1	1	Filter Closed

IDHIT1- IDHIT0— Identifier Acceptance Hit Indicator

The MSCAN08 sets these flags to indicate an identifier acceptance hit (see Section C.4). Table C-12 summarizes the different settings.

Table C-13 Identifier Acceptance Hit Indication

IDHIT1	IDHIT0	Identifier Acceptance Hit
0	0	Filter 0 Hit
0	1	Filter 1 Hit
1	0	Filter 2 Hit
1	1	Filter 3 Hit

The IDHIT indicators are always related to the message in the foreground buffer. When a message gets copied from the background to the foreground buffer the indicators are updated as well.

Note: The CIDAC register can only be written if the SFTRES bit in the MSCAN08 Module Control Register is set.

C

C.12.11 MSCAN08 Receive Error Counter (CRXERR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Receive error counter (CRXERR)	$xx0E	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0	0000 0000

This register reflects the status of the MSCAN08 receive error counter. The register is read only.

C.12.12 MSCAN08 Transmit Error Counter (CTXERR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Transmit error counter (CTXERR)	$xx0F	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0	0000 0000

This register reflects the status of the MSCAN08 transmit error counter. The register is read only.

Note: Both error counters may only be read when in Sleep or Soft Reset Mode..

C

C.12.13 MSCAN08 Identifier Acceptance Registers (CIDAR0-3)

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

The acceptance registers of the MSCAN08 are applied on the IDR0 to IDR3 registers of incoming messages in a bit by bit manner.

For extended identifiers all four acceptance and mask registers are applied. For standard identifiers only the first two (IDAR0, IDAR1) are applied. In the latter case it is required to program the mask register CIDMR1 in the three last bits (AC2 - AC0) to *don't care*.

Register	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
CIDAR0	$xx10	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	undefined
CIDAR1	$xx11	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	undefined
CIDAR2	$xx12	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	undefined
CIDAR3	$xx13	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0	undefined

AC7 – AC0 — Acceptance Code Bits

AC7 – AC0 comprise a user defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

Note: The CIDAR0-3 registers can only be written if the SFTRES bit in the MSCAN08 Module Control Register is set

C

C.12.14 MSCAN08 Identifier Mask Registers (CIDMR0-3)

Register	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
CIDMR0	$xx14	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	undefined
CIDMR1	$xx15	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	undefined
CIDMR2	$xx16	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	undefined
CIDMR3	$xx17	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	undefined

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering.

AM7 – AM0 — Acceptance Mask Bits

If a particular bit in this register is cleared this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit, before a match will be detected. The message will be accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register will not affect whether or not the message is accepted.

1 (set) – Ignore corresponding acceptance code register bit.

0 (clear) – Match corresponding acceptance code register and identifier bits.

Note: The CIDMR0-3 registers can only be written if the SFTRES bit in the MSCAN08 Module Control Register is set.

C

D
THE MOTOROLA SCALEABLE CAN (MSCAN12) MODULE

The MSCAN12 is the specific implementation of the Motorola Scalable CAN (MSCAN) concept targeted for the Motorola M68HC12 Microcontroller Family.

The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification dated September 1991.

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth.

MSCAN12 utilises an advanced buffer arrangement resulting in a predictable real-time behaviour and simplifies the application software.

D.1 Features

The basic features of the MSCAN12 are as follows:

- Modular Architecture
- Implementation of the CAN protocol - Version 2.0A/B
  - Standard and extended data frames.
  - 0 - 8 bytes data length.
  - Programmable bit rate up to 1 Mbps†.
- Support for Remote Frames.
- Double buffered receive storage scheme.
- Triple buffered transmit storage scheme with internal priorisation using a “local priority” concept.

D

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

- Flexible maskable identifier filter supports alternatively two full size extended identifier filters or four 16-bit filters or eight 8-bit filters.
- Programmable wake-up functionality with integrated low-pass filter.
- Programmable Loop-Back mode supports self-test operation.
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (Warning, Error Passive, Bus-Off).
- Programmable MSCAN12 clock source either CPU bus clock or crystal oscillator output.
- Programmable link to on-chip Timer Interface Module (TIM) for time-stamping and network synchronisation.
- Low power Sleep Mode.

D.2 External Pins

The MSCAN12 uses 2 external pins, 1 input (RxCAN) and 1 output (TxCAN). The TxCAN output pin represents the logic level on the CAN: '0' is for a dominant state, and '1' is for a recessive state.

RxCAN is on bit 0 of Port CAN, TxCAN is on bit 1. The remaining six pins of Port CAN are controlled by registers in the MSCAN12 address space (see Section D.12.15 through Section D.12.17). (See the chapter on I/O ports of the specific MCU for the actual number of pins used for Port CAN.)

A typical CAN system with MSCAN12 is shown in Figure D-1 below.

Each CAN station is connected physically to the CAN bus lines through a transceiver chip. The transceiver is capable of driving the large current needed for the CAN and has current protection, against defected CAN or defected stations.

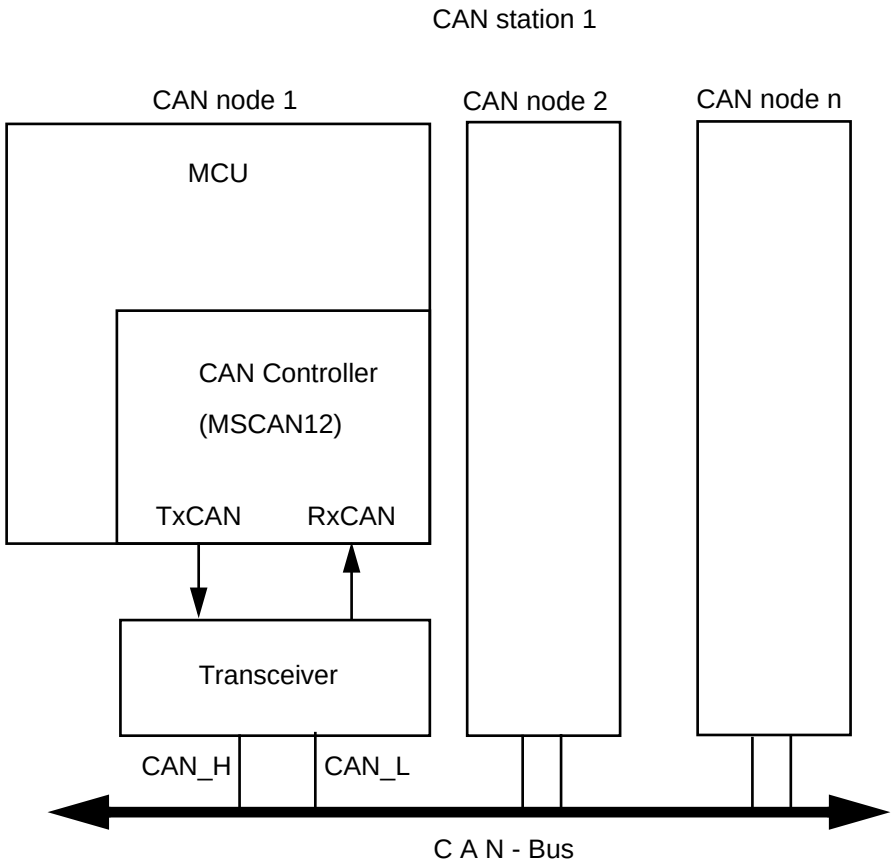


Figure D-1 The CAN System

D

D.3 Message Storage

MSCAN12 facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

D.3.1 Background

Modern application layer software is built upon two fundamental assumptions:

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

Above behaviour can not be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message has been sent. This loading process lasts a definite amount of time and has to be completed within the Inter-Frame Sequence (IFS) in order to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme would de-couple the re-loading of the transmit buffers from the actual message sending and as such reduces the reactiveness requirements on the CPU. Problems may arise if the sending of a message would be finished just while the CPU re-loads the second buffer, no buffer would then be ready for transmission and the bus would be released.

At least three transmit buffers are required to meet the first of above requirements under all circumstances. The MSCAN12 has three transmit buffers.

The second requirement calls for some sort of internal priorisation which the MSCAN12 implements with the “local priority” concept described below.

D.3.2 Receive Structures

The received messages are stored in a two stage input FIFO. The two message buffers are alternately mapped into a single memory area (see Figure D-2). While the background receive buffer (RxBG) is exclusively associated to the MSCAN12, the foreground receive buffer (RxFG) is addressable by the CPU12. This scheme simplifies the handler software as only one address area is applicable for the receive process.

Both buffers have a size of 13 bytes to store the CAN control bits, the identifier (standard or extended) and the data contents (for details see Section D.11).

The Receiver Full flag (RXF) in the MSCAN12 Receiver Flag Register (CRFLG) (see Section D.12.6) signals the status of the foreground receive buffer. When the buffer contains a correctly received message with matching identifier this flag is set.

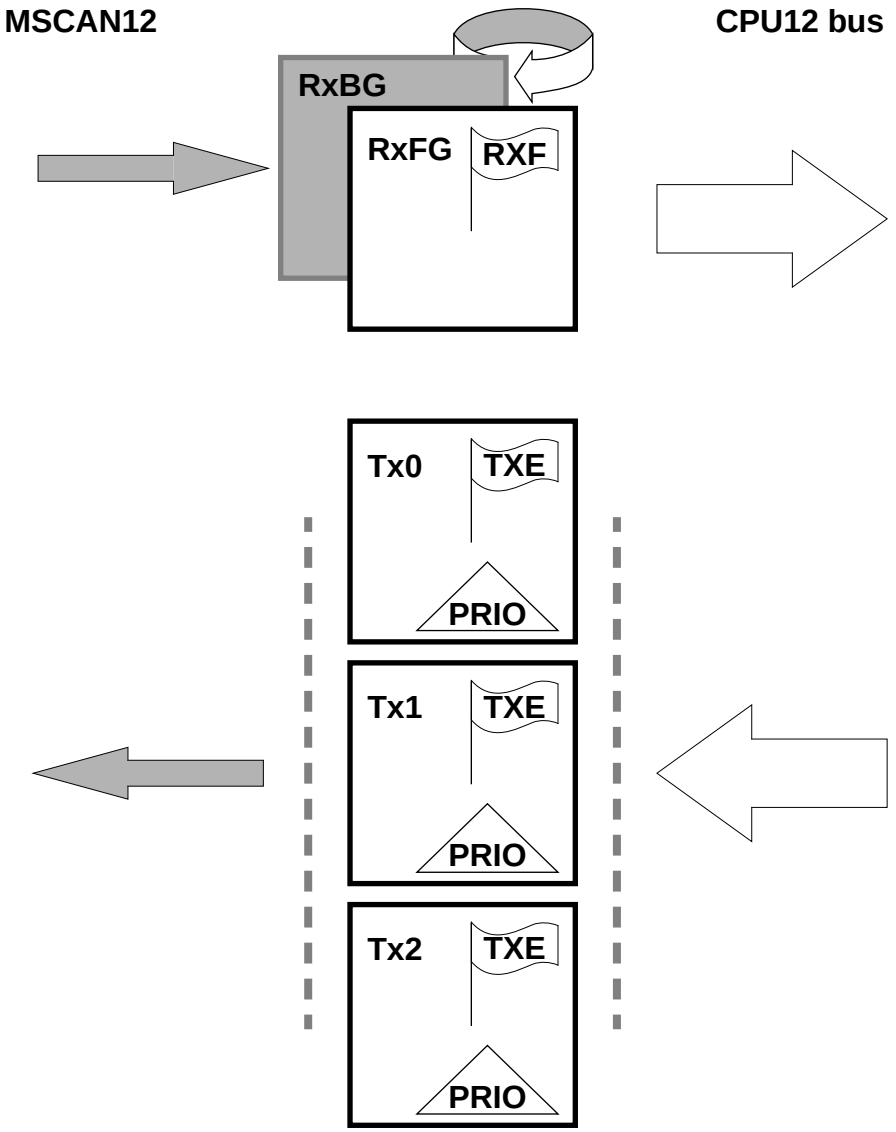
After the MSCAN12 has successfully received a message into the background buffer and if the message passes the filter (for details see Section D.4) it copies the content of RxBG into RxFG†, sets the RXF flag, and emits a receive interrupt to the CPU‡. A new message, which may follow immediately after the IFS field of the CAN frame, will be received into RxBG. The over-writing of the background buffer is independent of the identifier filter function The user's receive handler has to read the received message from RxFG and then reset the RXF flag in order to acknowledge the interrupt and to release the foreground buffer

An overrun condition occurs when both, the foreground and the background receive message buffers are filled with correctly received messages with accepted identifiers and a further message is being received from the bus. The latter message will be discarded and an error interrupt with overrun indication will occur if enabled. While in the overrun situation, the MSCAN12 will stay synchronized to the CAN bus and is able to transmit messages but will discard all incoming messages.

Note: MSCAN12 will receive its own messages into the background receive buffer RxBG but will NOT overwrite RxFG and will NOT emit a receive interrupt nor will it acknowledge (ACK) its own messages on the CAN bus. The exception to this rule is that when in loop-back mode MSCAN12 will treat its own messages exactly like all other incoming messages.

† Only if the RxF flag is not set.

‡ The receive interrupt will occur only if not masked. A polling scheme can be applied on RxF also.



D

Figure D-2 User model for message buffer organisation

D.3.3 Transmit Structures

The MSCAN12 has a triple transmit buffer scheme in order to allow multiple messages to be set up in advance and to achieve an optimized real-time performance. The three buffers are arranged as shown in Figure D-2.

All three buffers have a 13 byte data structure similar to the outline of the receive buffers (see Section D.11). An additional Transmit Buffer Priority Register (TBPR) contains an 8-bit so called “Local Priority” field (PRIO) (see Section D.11.5).

In order to transmit a message, the CPU12 has to identify an available transmit buffer which is indicated by a set Transmit Buffer Empty (TXE) Flags in the MSCAN12 Transmitter Flag Register (CTFLG) (see Section D.12.8).

The CPU12 then stores the identifier, the control bits and the data content into one of the transmit buffers. Finally, the buffer has to be flagged as being ready for transmission by clearing the TXE flag.

The MSCAN12 will then schedule the message for transmission and will signal the successful transmission of the buffer by setting the TXE flag. A transmit interrupt will be emitted† when TXE is set and this can be used to drive the application software to re-load the buffer.

In case more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN12 uses the “local priority” setting of the three buffers for priorisation. For this purpose every transmit buffer has an 8-bit local priority field (PRIO). The application software sets this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being emitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority.

The internal scheduling process takes place whenever the MSCAN12 arbitrates for the bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software it may become necessary to abort a lower priority message being set up in one of the three transmit buffers. As messages that are already under transmission can not be aborted, the user has to request the abort by setting the corresponding Abort Request Flag (ABTRQ) in the Transmission Control Register (CTCR). The MSCAN12 then grants the request, if possible, by setting the corresponding Abort Request Acknowledge (ABTAK) and the TXE flag in order to release the buffer and by emitting a transmit interrupt. The transmit interrupt handler software can tell from the setting of the ABTAK flag whether the message was aborted (ABTAK=1), or sent in the meantime (ABTAK=0).

D

† The transmit interrupt will occur only if not masked. A polling scheme can be applied on TXE also.

D.4 Identifier Acceptance Filter

A very flexible programmable generic identifier acceptance filter has been introduced in order to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes:

- Two identifier acceptance filters, each to be applied to the full 29 bits of the identifier and to the following bits of the CAN frame: RTR, IDE, SRR. This mode implements two filters for a full length CAN 2.0B compliant extended identifier. Figure D-3 shows how the first 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces a filter 0 hit. Similarly, the second filter bank (CIDAR4-7, CIDMR4-7) produces a filter 1 hit.
- Four identifier acceptance filters, each to be applied to a) the 11 bits of the identifier and the RTR bit of CAN 2.0A mesages or b) the 14 most significant bits of the identifier of CAN 2.0B messages. Figure D-4 shows how the first 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces filter 0 and 1 hits. Similarly, the second filter bank (CIDAR4-7, CIDMR4-7) produces filter 2 and 3 hits.
- Eight identifier acceptance filters, each to be applied to the first 8 bits of the identifier. This mode implements eight independent filters for the first 8 bit of a CAN 2.0A compliant standard identifier, or of a CAN 2.0B compliant extended identifier. Figure D-5 shows how the first 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces filter 0 to 3 hits. Similarly, the second filter bank (CIDAR4-7, CIDMR4-7) produces filter 4 to 7 hits.
- Closed filter. No CAN message will be copied into the foreground buffer RxFG, and the RXF flag will never be set.

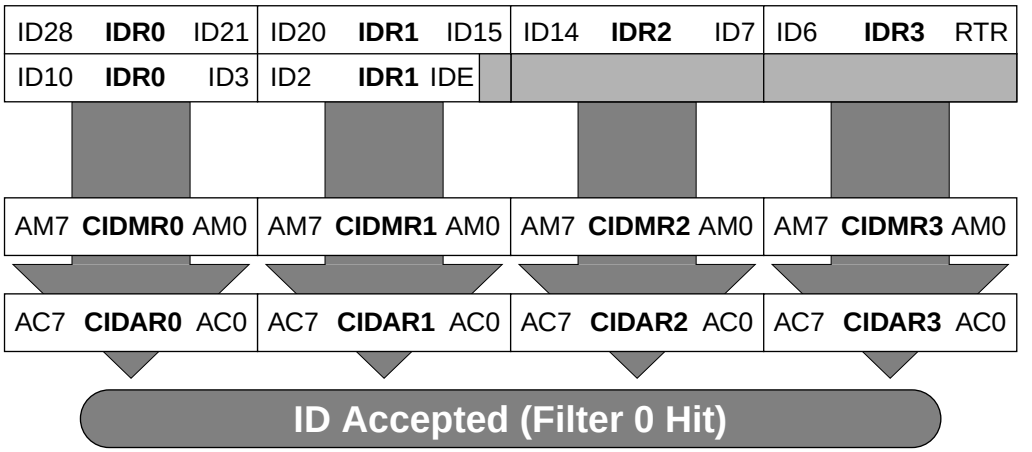


Figure D-3 32-bit Maskable Identifier Acceptance Filter

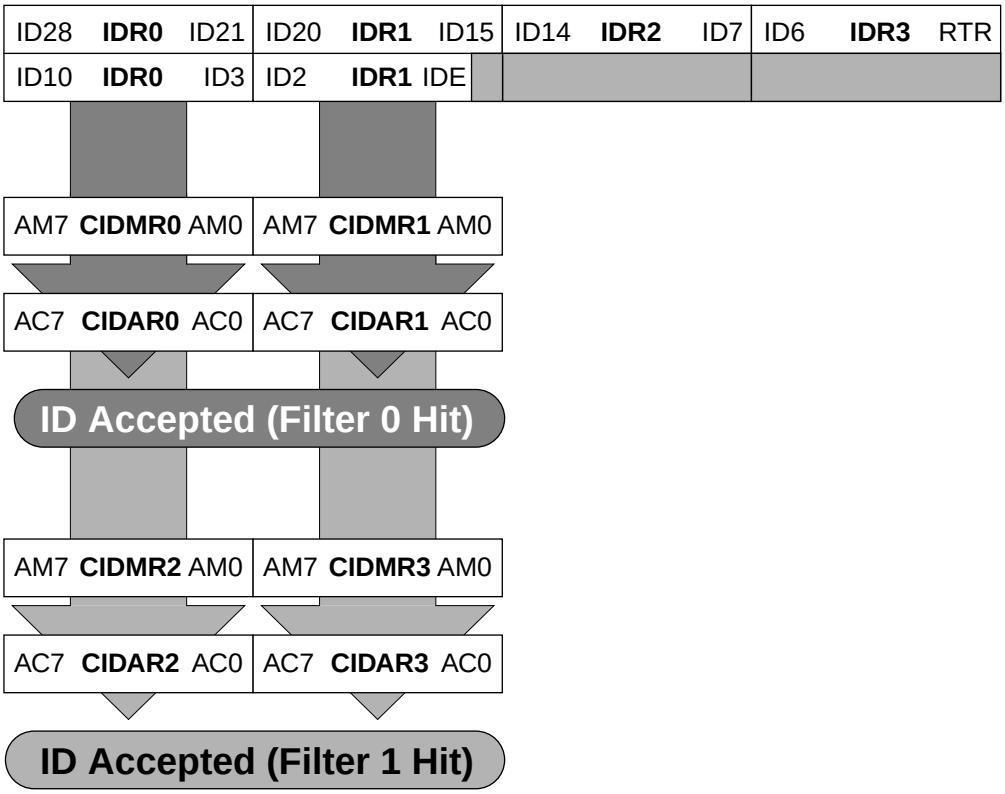


Figure D-4 16-bit Maskable Acceptance Filters

The Identifier Acceptance Registers (CIDAR0-7) define the acceptable patterns of the standard or extended identifier (ID10 - ID0 or ID28 - ID0). Any of these bits can be marked ‘don’t care’ in the Identifier Mask Registers (CIDMR0-7).

A filter hit is indicated to the application software by a set RXF (Receive Buffer Full Flag, see Section D.12.6) and three bits in the Identifier Acceptance Control Register (see Section D.12.10). These Identifier Hit Flags (IDHIT2-0) clearly identify the filter section that caused the acceptance. They simplify the application software’s task to identify the cause of the receiver interrupt. In case that more than one hit occurs (two or more filters match) the lower hit has priority.

A hit will also cause a receiver interrupt if enabled

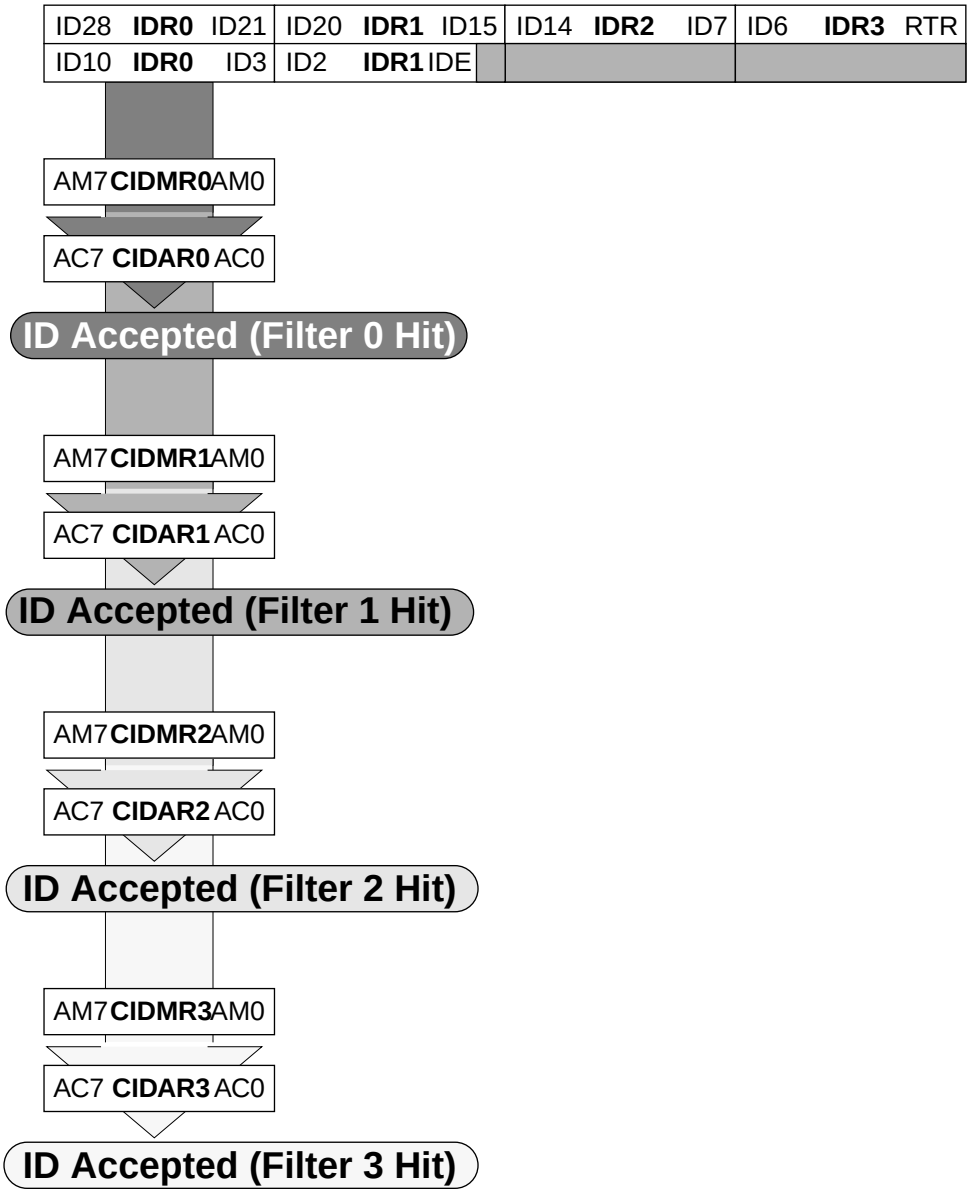


Figure D-5 8-bit Maskable Acceptance Filters

D

D.5 Interrupts

The MSCAN12 supports four interrupt vectors mapped onto eleven different interrupt sources, any of which can be individually masked (for details see Section D.12.6 to Section D.12.9):

- *Transmit Interrupt*: At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXE flags of the empty message buffers are set.
- *Receive Interrupt*: A message has been successfully received and loaded into the foreground receive buffer. This interrupt will be emitted immediately after receiving the EOF symbol. The RXF flag is set.
- *Wake-Up Interrupt*: An activity on the CAN bus occurred during MSCAN12 internal Sleep Mode.
- *Error Interrupt*: An overrun, error or warning condition occurred. The Receiver Flag Register (CRFLG) will indicate one of the following conditions:
  - *Overrun:* An overrun condition as described in Section D.3.2 has occurred.
  - *Receiver Warning*: The Receive Error Counter has reached the CPU Warning limit of 96.
  - *Transmitter Warning*: The Transmit Error Counter has reached the CPU Warning limit of 96.
  - *Receiver Error Passive*: The Receive Error Counter has exceeded the Error Passive limit of 127 and MSCAN12 has gone to Error Passive state.
  - *Transmitter Error Passive*: The Transmit Error Counter has exceeded the Error Passive limit of 127 and MSCAN12 has gone to Error Passive state.
  - *Bus Off*: The Transmit Error Counter has exceeded 255 and MSCAN12 has gone to Bus Off state.

D.5.1 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the MSCAN12 Receiver Flag Register (CRFLG) or the MSCAN12 Transmitter Control Register (CTCR). Interrupts are pending as long as one of the corresponding flags is set. The flags in above registers must be reset within the interrupt handler in order to handshake the interrupt. The flags are reset through writing a “1” to the corresponding bit position. A flag can not be cleared if the respective condition still prevails.

Caution: Bit manipulation instructions (BSET) shall not be used to clear interrupt flags.

D.5.2 Interrupt Vectors

The MSCAN12 supports four interrupt vectors as shown in Table D-1. The vector addresses and the relative interrupt priority are dependent on the chip integration to be defined.

Table D-1 MSCAN12 Interrupt Vectors

Function	Source	Local Mask	Global Mask
Wake-Up	WUPIF	WUPIE	I Bit
Error Interrupts	RWRNIF	RWRNIE	
	TWRNIF	TWRNIE	
	RERRIF	RERRIE	
	TERRIF	TERRIE	
	BOFFIF	BOFFIE	
	OVRIF	OVRIE	
Receive	RXF	RXFIE	
Transmit	TXE0	TXEIE0	
	TXE1	TXEIE1	
	TXE2	TXEIE2	

D.6 Protocol Violation Protection

The MSCAN12 will protect the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters can not be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN12 can not be modified while the MSCAN12 is on-line. The SFTRES bit in CMCR0 (see Section D.12.2) serves as a lock to protect the following registers:
  - MSCAN12 Module Control Register 1 (CMCR1)
  - MSCAN12 Bus Timing Register 0 and 1 (CBTR0, CBTR1)
  - MSCAN12 Identifier Acceptance Control Register (CIDAC)
  - MSCAN12 Identifier Acceptance Registers (CIDAR0-7)
  - MSCAN12 Identifier Mask Registers (CIDMR0-7)
- The TxCAN pin is forced to Recessive if the CPU goes into STOP mode.

D

D.7 Low Power Modes

The MSCAN12 has three modes with reduced power consumption compared to Normal Mode. In Sleep and Soft Reset Mode, power consumption is reduced by stopping all clocks except those to access the registers. In Power Down Mode, all clocks are stopped and no power is consumed.

The WAI and STOP instruction put the MCU in low power consumption stand-by modes. Table D-2 summarizes the combinations of MSCAN12 and CPU modes. A particular combination of modes is entered for the given settings of the bits CSWAI, SLPAK, and SFTRES. For all modes, an MSCAN wake-up interrupt can occur only if SLPAK=WUPIE=1. While the CPU is in Wait Mode, the MSCAN12 can be operated in Normal Mode and emit interrupts (registers can be accessed via background debug mode).

Table D-2 MSCAN12 vs. CPU operating modes

MSCAN Mode	CPU Mode		
	STOP	**WAIT**	**RUN**
Power Down	CSWAI = X(1) SLPAK = X SFTRES = X	CSWAI = 1 SLPAK = X SFTRES = X	
Sleep		CSWAI = 0 SLPAK = 1 SFTRES = 0	CSWAI = X SLPAK = 1 SFTRES = 0
Soft Reset		CSWAI = 0 SLPAK = 0 SFTRES = 1	CSWAI = X SLPAK = 0 SFTRES = 1
Normal		CSWAI = 0 SLPAK = 0 SFTRES = 0	CSWAI = X SLPAK = 0 SFTRES = 0

(1) 'X' means don't care.

D.7.1 MSCAN12 Sleep Mode

The CPU can request the MSCAN12 to enter this low-power mode by asserting the SLPRQ bit in the Module Configuration Register (see Figure D-6). The time when the MSCAN12 will then enter Sleep Mode depends on its current activity:

- if it is transmitting, it will continue to transmit until there is no more message to be transmitted, and then go into Sleep Mode.
- if it is receiving, it will wait for the end of this message and then go into Sleep Mode.
- if it is neither transmitting nor receiving, it will immediately go into Sleep Mode.

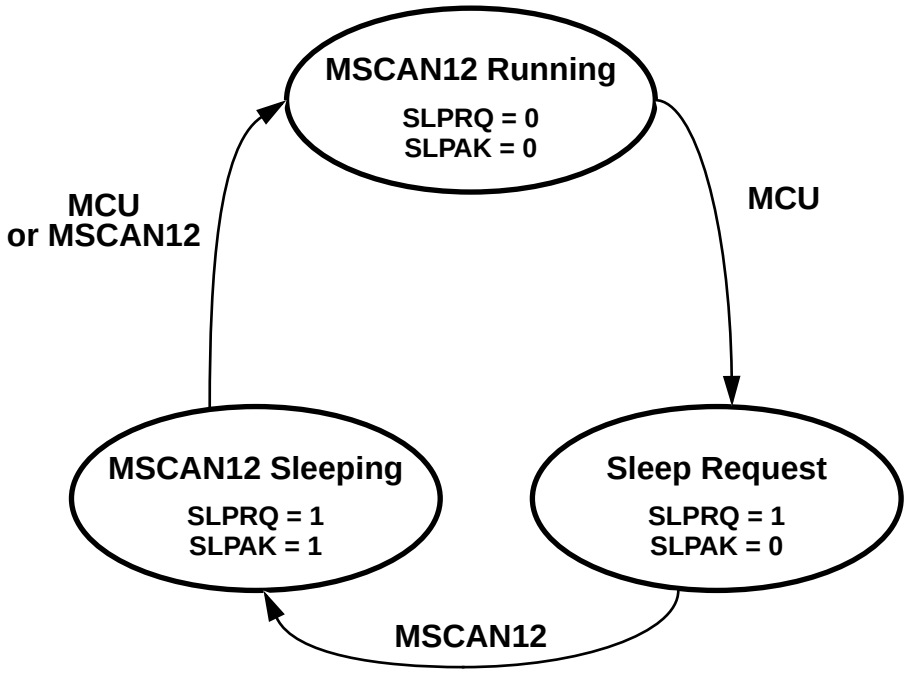


Figure D-6 Sleep Request / Acknowledge Cycle

The application software must avoid to set up a transmission (by clearing one or more TXE flag(s)) and immediately request Sleep Mode (by setting SLPRQ). It will then depend on the exact sequence of operations whether the MSCAN12 will start transmitting or go into Sleep Mode directly.

D

During Sleep Mode, the SLPAK flag is set. The application software should use this flag as a handshake indication for the request to go into Sleep Mode. When in Sleep Mode, the MSCAN12 stops its own clocks and the TxCAN pin will stay in recessive state.

The MSCAN12 will leave Sleep Mode (wake-up) when

- bus activity occurs or
- the MCU clears the SLPRQ bit.

Note: The MCU cannot clear the SLPRQ bit before the MSCAN12 is in Sleep Mode (SLPAK = 1).

D.7.2 MSCAN12 Soft Reset Mode

In Soft Reset Mode, the MSCAN12 is stopped. Registers can still be accessed. This mode is used to initialize the module configuration, bit timing, and the CAN message filter. See Section D.12.2 for a complete description of the Soft Reset Mode.

D.7.3 MSCAN12 Power Down Mode

The MSCAN12 is in Power Down Mode when

- the CPU is in Stop Mode or
- the CPU is in Wait Mode and the CSWAI bit is set (see Section D.12.2).

When entering the Power Down Mode, the MSCAN12 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations. The user is responsible to take care that the MSCAN12 is not active when Power Down Mode is entered. The recommended procedure is to bring the MSCAN12 into Sleep Mode before the STOP instruction - or the WAI instruction, if CSWAI is set - is executed.

To protect the CAN bus system from fatal consequences of violations to the above rule, the MSCAN12 will drive the TxCAN pin into recessive state.

In Power Down Mode, no registers can be accessed.

D.7.4 Programmable Wake-Up Function

The MSCAN12 can be programmed to apply a low-pass filter function to the RxCAN input line while in Sleep Mode (see control bit WUPM in the Module Control Register, Section D.12.3). This feature can be used to protect the MSCAN12 from wake-up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic inference within noisy environments.

D.8 Timer Link

The MSCAN12 will generate a timer signal whenever a valid frame has been received. Because the CAN specification defines a frame to be valid if no errors occurred before the EOF field has been transmitted successfully, the timer signal will be generated right after the EOF. A pulse of one bit time is generated. As the MSCAN12 receiver engine also receives the frames being sent by itself, a timer signal will also be generated after a successful transmission.

The previously described timer signal can be routed into the on-chip Timer Interface Module (TIM / ECT). This signal is connected to the Timer n Channel m input† under the control of the Timer Link Enable (TLNKEN) bit in CMCR0.

After Timer n has been programmed to capture rising edge events it can be used under software control to generate 16-bit time stamps which can be stored with the received message.

D.9 Clock System

Figure D-7 shows the structure of the MSCAN12 clock generation circuitry. With this flexible clocking scheme the MSCAN12 is able to handle CAN bus rates ranging from 10 kbps up to 1 Mbps.

The Clock Source bit (CLKSRC) in the MSCAN12 Module Control Register (CMCR1) (see Section D.12.4) defines whether the MSCAN12 is connected to the output of the crystal oscillator (EXTALi) or to a clock twice as fast as the system clock (ECLK).

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met. Additionally, for high CAN bus rates (1 Mbps), a 50% duty cycle of the clock is required.

For microcontrollers without the CGM module, CGMCANCLK is driven from the crystal oscillator (EXTALi).

D

† The timer channel being used for the timer link is integration dependent.

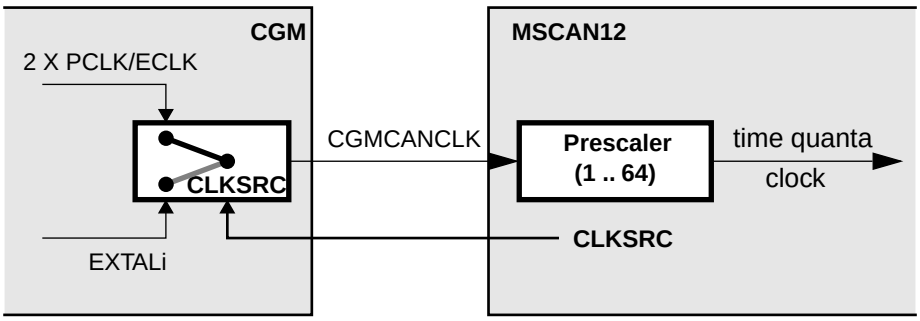


Figure D-7 Clocking Scheme

A programmable prescaler is used to generate out of MSCANCLK the time quanta (Tq) clock. A time quantum is the atomic unit of time handled by the MSCAN12. A bit time is subdivided into three segments†:

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time segment 2: This segment represents the PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

The Synchronisation Jump Width can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

Above parameters can be set by programming the Bus Timing Registers (CBTR0-1, see Section D.12.4 and Section D.12.5).

It is the user's responsibility to make sure that his bit time settings are in compliance with the CAN standard. Figure D-3 gives an overview on the CAN conforming segment settings and the related parameter values.

D

† For further explanation of the under-lying concepts please refer to ISO/DIS 11519-1, Section 10.3.

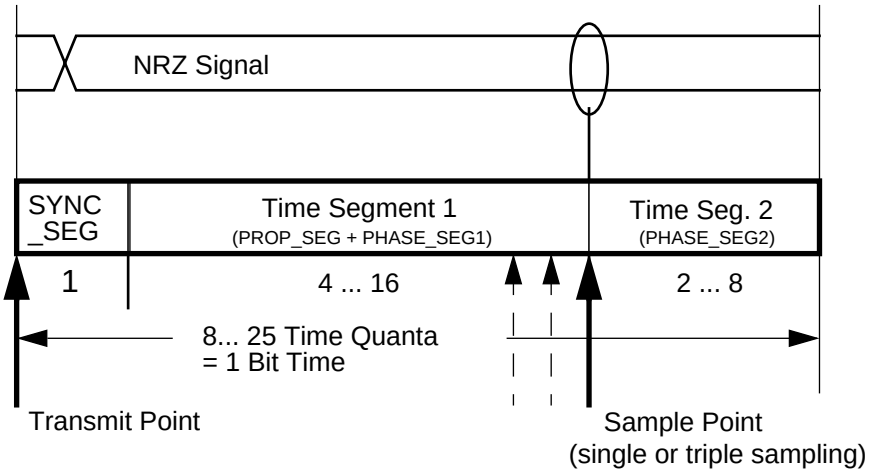


Figure D-8 Segments within the Bit Time

Table D-3 CAN Standard Compliant Bit Time Segment Settings

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

D

D.10 Memory Map

The MSCAN12 occupies 128 Byte in the CPU12 memory space. The absolute mapping is implementation dependent with the base address being a multiple of 128. The background receive buffer can only be read in test mode.

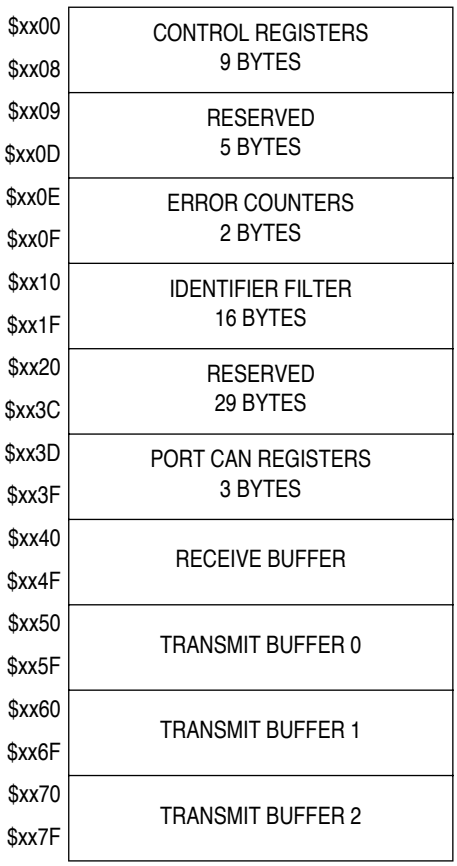


Figure D-9 MSCAN12 Memory Map

Note: Due to design requirements, the absolute addresses and bit locations may change with later releases of the specification.

D

D.11 Programmer's Model of Message Storage

The following section details the organisation of the receive and transmit message buffers and the associated control registers. For reasons of programmer interface simplification, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 byte in the memory map containing a 13 byte data structure. An additional Transmit Buffer Priority Register (TBPR) is defined for the transmit buffers.

Table D-4 Message Buffer Organisation

Address	Register Name
xxb0	Identifier Register 0
xxb1	Identifier Register 1
xxb2	Identifier Register 2
xxb3	Identifier Register 3
xxb4	Data Segment Register 0
xxb5	Data Segment Register 1
xxb6	Data Segment Register 2
xxb7	Data Segment Register 3
xxb8	Data Segment Register 4
xxb9	Data Segment Register 5
xxbA	Data Segment Register 6
xxbB	Data Segment Register 7
xxbC	Data Length Register
xxbD	Transmit Buffer Priority Register(1)
xxbE	unused
xxbF	unused

(1) Not Applicable for Receive Buffers

D.11.1 Message Buffer Outline

Figure D-10 shows the common 13 byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in Figure D-11. All bits of the 13 byte data structure are undefined out of reset.

D

Register	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Identifier register 0 (IDR0)	$xxb0	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
Identifier register 1 (IDR1)	$xxb1	ID20	ID19	ID18	SRR (1)	IDE (1)	ID17	ID16	ID15
Identifier register 2 (IDR2)	$xxb2	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
Identifier register 3 (IDR3)	$xxb3	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
Data segment register 0 (DSR0)	$xxb4	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 1 (DSR1)	$xxb5	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 2 (DSR2)	$xxb6	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 3 (DSR3)	$xxb7	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 4 (DSR4)	$xxb8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 5 (DSR5)	$xxb9	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 6 (DSR6)	$xxbA	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data segment register 7 (DSR7)	$xxbB	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Data length register (DLR)	$xxbC					DLC3	DLC2	DLC1	DLC0

Figure D-10 Receive/transmit message buffer extended identifier

Register	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Identifier register 0 (IDR0)	$xxb0	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
Identifier register 1 (IDR1)	$xxb1	ID2	ID1	ID0	RTR	IDE (0)			
Identifier register 2 (IDR2)	$xxb2								
Identifier register 3 (IDR3)	$xxb3								

Figure D-11 Standard identifier mapping

D.11.2 Identifier Registers (IDRn)

The identifiers consist of either 11 bits (ID10 – ID0) for the standard, or 29 bits (ID28 - ID0) for the extended format. ID10/28 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

SRR - Substitute Remote Request

This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and will be stored as received on the CAN bus for receive buffers.

IDE — ID Extended

This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer the flag is set as being received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer the flag indicates to the MSCAN12 what type of identifier to send.

1 (set) – Extended format (29 bit)

0 (clear) – Standard format (11 bit)

RTR — Remote transmission request

This flag reflects the status of the Remote Transmission Request bit in the CAN frame. In the case of a receive buffer it indicates the status of the received frame and allows to support the transmission of an answering frame in software. In the case of a transmit buffer this flag defines the setting of the RTR bit to be sent.

1 (set) – Remote frame

0 (clear) – Data frame

D

D.11.3 Data Length Register (DLR)

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

DLC3 – DLC0 — Data length code bits

The data length code contains the number of bytes (data byte count) of the respective message. At the transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. Table D-5 shows the effect of setting the DLC bits.

Table D-5 Data length codes

Data length code				Data byte count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

D.11.4 Data Segment Registers (DSRn)

The eight data segment registers contain the data to be transmitted or being received. The number of bytes to be transmitted or being received is determined by the data length code in the corresponding DLR.

D.11.5 Transmit Buffer Priority Registers (TBPR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Transmit buffer priority registers (TBPR)	$xxbD	PRIO7	PRIO6	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0	undefined

PRIO7 - PRIO0— Local Priority

This field defines the local priority of the associated message buffer. The local priority is used for the internal priorisation process of the MSCAN12 and is defined to be highest for the smallest binary number. The MSCAN12 implements the following internal priorisation mechanism:

- All transmission buffers with a cleared TXE flag participate in the priorisation immediately before the SOF (Start of Frame) is sent.
- The transmission buffer with the lowest local priority field wins the priorisation.
- In cases of more than one buffer having the same lowest priority the message buffer with the lower index number wins.

Caution: To ensure data integrity, no registers of the transmit buffers should be written to while the associated TXE flag is cleared.

Caution: To ensure data integrity, no registers of the receive buffer shall be read while the RXF flag is cleared.

D.12 Programmer’s Model of Control Registers

D.12.1 Overview

The programmer’s model has been laid out for maximum simplicity and efficiency. The following figure gives an overview of the control register block of the MSCAN12:

Table D-6 MSCAN12 Control Register Structure

Register	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
CMCR0	$xx00	0	0	CSWAI	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
CMCR1	$xx01	0	0	0	0	0	LOOPB	WUPM	CLKSRC
CBTR0	$xx02	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
CBTR1	$xx03	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
CRFLG	$xx04	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
CRIER	$xx05	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
CTFLG	$xx06	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
CTCR	$xx07	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
CIDAC	$xx08	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
reserved	$xx09-$xx0D								
CRXERR	$xx0E	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
CTXERR	$xx0F	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
CIDAR0	$xx10	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
CIDAR1	$xx11	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
CIDAR2	$xx12	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
CIDAR3	$xx13	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
CIDMR0	$xx14	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
CIDMR1	$xx15	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
CIDMR2	$xx16	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
CIDMR3	$xx17	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
CIDAR4	$xx18	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
CIDAR5	$xx19	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
CIDAR6	$xx1A	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
CIDAR7	$xx1B	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
CIDMR4	$xx1C	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
CIDMR5	$xx1D	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
CIDMR6	$xx1E	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
CIDMR7	$xx1F	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
reserved	$xx20-$xx3C								
PCTLCAN	$xx3D	0	0	0	0	0	0	PUECAN	RDRCAN
PORTCAN	$xx3E	PCAN7	PCAN6	PCAN5	PCAN4	PCAN3	PCAN2	TxCan	RxCan
DDRCAN	$xx3F	DDRCAN7	DDRCAN6	DDRCAN5	DDRCAN4	DDRCAN3	DDRCAN2	0	0

D

D.12.2 MSCAN12 Module Control Register 0 (CMCR0)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Module control register 0 (CMCR0)	$xx00	0	0	CSWAI	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES	0010 0001

CSWAI — CAN Stops in Wait Mode

1 (set) – The module ceases to be clocked during WAIT mode.

0 (clear) – The module is not affected during WAIT mode.

SYNCH — Synchronized Status

This bit indicates whether the MSCAN12 is synchronized to the CAN bus and as such can participate in the communication process.

1 (set) – MSCAN12 is synchronized to the CAN bus.

0 (clear) – MSCAN12 is not synchronized to the CAN bus.

TLNKEN - Timer Enable

This flag is used to establish a link between the MSCAN12 and the on-chip timer (see Section D.8).

1 (set) – The MSCAN12 timer signal output is connected to the timer input.

0 (clear) – The port is connected to the timer input.

SLPAK — Sleep Mode Acknowledge

This flag indicates whether the MSCAN12 is in module internal Sleep Mode. It shall be used as a handshake for the Sleep Mode request (see Section D.7.1).

1 (set) – Sleep – The MSCAN12 is in Sleep Mode.

0 (clear) – Wake-up – The MSCAN12 is not in Sleep Mode.

SLPRQ — Sleep request, go to Sleep Mode

This flag allows to request the MSCAN12 to go into an internal power-saving mode (see Section D.7.1).

1 (set) – Sleep request – The MSCAN12 will go into Sleep Mode.

0 (clear) – Wake-up – The MSCAN12 will function normally.

SFTRES— Soft Reset

When this bit is set by the CPU, the MSCAN12 immediately enters the soft reset state. Any ongoing transmission or reception is aborted and synchronisation to the bus is lost.

The following registers will go into and stay in the same state as out of hard reset: CMCR0, CRFLG, CRIER, CTFLG, CTCR.

The registers CMCR1, CBTR0, CBTR1, CIDAC, CIDAR0-7, CIDMR0-7 can only be written by the CPU when the MSCAN12 is in soft reset state. The values of the error counters are not affected by soft reset.

When this bit is cleared by the CPU, MSCAN12 will try to synchronize to the CAN bus: If the MSCAN12 is not in bus-off state it will be synchronized after 11 recessive bits on the bus; if the MSCAN12 is in bus-off state it continues to wait for 128 occurrences of 11 recessive bits.

Clearing SFTRES and writing to other bits in CMCR0 must be in separate instructions.

1 (set) – MSCAN12 in soft reset state.

0 (clear) – Normal operation.

D.12.3 MSCAN12 Module Control Register 1 (CMCR1)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Module control register 1 (CMCR1)	$xx01	0	0	0	0	0	LOOPB	WUPM	CLKSRC	0000 0000

LOOPB - Loop Back Self Test Mode

When this bit is set the MSCAN12 performs an internal loop back which can be used for self test operation: the bit stream output of the transmitter is fed back to the receiver. The RxCAN input pin is ignored and the TxCAN output goes to the recessive state (1). Note that in this state the MSCAN12 ignores the ACK bit to insure proper reception of its own message and will treat messages being received while in transmission as received messages from remote nodes.

1 (set) – Activate loop back self test mode.

0 (clear) – Normal operation.

WUPM - Wake-Up Mode

This flag defines whether the integrated low-pass filter is applied to protect the MSCAN12 from spurious wake-ups (see Section D.7.4).

1 (set) – MSCAN12 will wake up the CPU only in case of dominant pulse on the bus which has a length of at least approximately T_{wup}.

0 (clear) – MSCAN12 will wake up the CPU after any recessive to dominant edge on the CAN bus.

D

CLKSRC - MSCAN12 Clock Source

This flag defines which clock source the MSCAN12 module is driven from (only for system with CGM module; see Section D.9, Figure D-7).

1 (set) – The MSCAN12 clock source is twice the frequency of ECLK.

0 (clear) – The MSCAN12 clock source is EXTALi.

Note: The CMCR1 register can only be written if the SFTRES bit in CMCR0 is set.

D.12.4 MSCAN12 Bus Timing Register 0 (CBTR0)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Bus timing register 0 (CBTR0)	$xx02	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	0000 0000

SJW1, SJW0 — Synchronization Jump Width

The synchronization jump width defines the maximum number of time quanta (Tq) clock cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see Table D-7).

Table D-7 Synchronization jump width

SJW1	SJW0	Synchronization jump width
0	0	1 Tq clock cycle
0	1	2 Tq clock cycles
1	0	3 Tq clock cycles
1	1	4 Tq clock cycles

BRP5 – BRP0 — Baud Rate Prescaler

These bits determine the time quanta (Tq) clock, which is used to build up the individual bit timing, according to Table D-8.

D

Table D-8 Baud rate prescaler

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

Note: The CBTR0 register can only be written if the SFTRES bit in CMCR0 is set.

D.12.5 MSCAN12 Bus Timing Register 1 (CBTR1)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Bus timing register 1 (CBTR1)	$xx03	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10	0000 0000

SAMP — Sampling

This bit determines the number of samples of the serial bus to be taken per bit time. If set three samples per bit are taken, the regular one (sample point) and two preceding samples, using a majority rule. For higher bit rates SAMP should be cleared, which means that only one sample will be taken per bit.

1 (set) – Three samples per bit.

0 (clear) – One sample per bit.

TSEG22 – TSEG10 — Time Segment

Time segments within the bit time fix the number of clock cycles per bit time, and the location of the sample point.

D

Table D-9 Time segment syntax

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

Time segment 1 (TSEG1) and time segment 2 (TSEG2) are programmable as shown in Table D-10.

Table D-10 Time segment values

TSEG 13	TSEG 12	TSEG 11	TSEG 10	Time segment 1
0	0	0	0	1 Tq clock cycle
0	0	0	1	2 Tq clock cycles
0	0	1	0	3 Tq clock cycles
0	0	1	1	4 Tq clock cycles
.	.	.	.	.
.	.	.	.	.
1	1	1	1	16 Tq clock cycles

TSEG 22	TSEG 21	TSEG 20	Time segment 2
0	0	0	1 Tq clock cycle
0	0	1	2 Tq clock cycles
.	.	.	.
.	.	.	.
1	1	1	8 Tq clock cycles

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (Tq) clock cycles per bit (as shown above).

Note: The CBTR1 register can only be written if the SFTRES bit in CMCR0 is set.

D

D.12.6 MSCAN12 Receiver Flag Register (CRFLG)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Receiver flag register (CRFLG)	$\$xx04$	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF	0000 0000

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. A flag can only be cleared when the condition which caused the setting is no more valid. Writing a '0' has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CRIER register. A hard or soft reset will clear the register.

WUPIF — Wake-up Interrupt Flag

If the MSCAN12 detects bus activity whilst it is in Sleep Mode, it clears the SLPAK bit in the CMCR0 register; the WUPIF bit will then be set. If not masked, a Wake-Up interrupt is pending while this flag is set.

1 (set) – MSCAN12 has detected activity on the bus and requested wake-up.

0 (clear) – No wake-up activity has been observed while in Sleep Mode.

RWRNIF — Receiver Warning Interrupt Flag

This bit will be set when the MSCAN12 goes into warning status due to the Receive Error counter (REC) being in the range of 96 to 127 and neither one of the Error interrupt flags or the Bus-Off interrupt flag is set†. If not masked, an Error interrupt is pending while this flag is set.

1 (set) – MSCAN12 has gone into receiver warning status.

0 (clear) – No receiver warning status has been reached.

TWRNIF — Transmitter Warning Interrupt Flag

This bit will be set when the MSCAN12 goes into warning status due to the Transmit Error counter (TEC) being in the range of 96 to 127 and neither one of the Error interrupt flags or the Bus-Off interrupt flag is set‡. If not masked, an Error interrupt is pending while this flag is set.

1 (set) – MSCAN12 has gone into transmitter warning status.

0 (clear) – No transmitter warning status has been reached.

D

† RWRNIF = (96 ≤ REC < 128) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$

‡ TWRNIF = (96 ≤ TEC < 128) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$

RERRIF — Receiver Error Passive Interrupt Flag

This bit will be set when the MSCAN12 goes into error passive status due to the Receive Error counter exceeding 127 and the Bus-Off interrupt flag is not set†. If not masked, an Error interrupt is pending while this flag is set.

1 (set) – MSCAN12 has gone into receiver error passive status.

0 (clear) – No receiver error passive status has been reached.

TERRIF — Transmitter Error Passive Interrupt Flag

This bit will be set when the MSCAN12 goes into error passive status due to the Transmit Error counter exceeding 127 and the Bus-Off interrupt flag is not set‡. If not masked, an Error interrupt is pending while this flag is set.

1 (set) – MSCAN12 has gone into transmitter error passive status.

0 (clear) – No transmitter error passive status has been reached.

BOFFIF — Bus-Off Interrupt Flag

This bit will be set when the MSCAN12 goes into bus-off status, due to the Transmit Error counter exceeding 255. It cannot be cleared before the MSCAN12 has monitored 128 times 11 consecutive 'recessive' bits on the bus. If not masked, an Error interrupt is pending while this flag is set.

1 (set) – MSCAN12 has gone into bus-off status.

0 (clear) – No bus-off status has been reached.

OVRIF — Overrun Interrupt Flag

This bit will be set when a data overrun condition occurs. If not masked, an Error interrupt is pending while this flag is set.

1 (set) – A data overrun has been detected.

0 (clear) – No data overrun has occurred.

RXF — Receive Buffer Full

The RXF flag is set by the MSCAN12 when a new message is available in the foreground receive buffer. This flag indicates whether the buffer is loaded with a correctly received message. After the CPU has read that message from the receive buffer, the RXF flag must be handshaken in order to

D

† RERRIF = (128 ≤ REC ≤ 255) & $\overline{\text{BOFFIF}}$

‡ TERRIF = (128 ≤ TEC ≤ 255) & $\overline{\text{BOFFIF}}$: TERRIF is set at the end of the Bus-Off period (due to counting 128 * 11 consecutive 'recessive' bits on the TEC). Thus TERRIF should be cleared together with BOFFIF.

release the buffer. A set RXF flag prohibits the exchange of the background receive buffer into the foreground buffer. If not masked, a Receive interrupt is pending while this flag is set.

1 (set) – The receive buffer is full. A new message is available.

0 (clear) – The receive buffer is released (not full).

Note: The CRFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.

D.12.7 MSCAN12 Receiver Interrupt Enable Register (CRIER)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Receiver interrupt enable register (CRIER)	$xx05	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXRIE	0000 0000

WUPIE — Wake-up Interrupt Enable

1 (set) – A wake-up event will result in a wake-up interrupt.

0 (clear) – No interrupt will be generated from this event.

RWRNIE — Receiver Warning Interrupt Enable

1 (set) – A receiver warning status event will result in an error interrupt.

0 (clear) – No interrupt will be generated from this event.

TWRNIE — Transmitter Warning Interrupt Enable

1 (set) – A transmitter warning status event will result in an error interrupt.

0 (clear) – No interrupt will be generated from this event.

RERRIE — Receiver Error Passive Interrupt Enable

1 (set) – A receiver error passive status event will result in an error interrupt.

0 (clear) – No interrupt will be generated from this event.

TERRIE — Transmitter Error Passive Interrupt Enable

1 (set) – A transmitter error passive status event will result in an error interrupt.

0 (clear) – No interrupt will be generated from this event.

BOFFIE — Bus-Off Interrupt Enable

1 (set) – A bus-off event will result in an error interrupt.

0 (clear) – No interrupt will be generated from this event.

OVRIE — Overrun Interrupt Enable

1 (set) – An overrun event will result in an error interrupt.

0 (clear) – No interrupt will be generated from this event.

RXFIE — Receiver Full Interrupt Enable

1 (set) – A receive buffer full (successful message reception) event will result in a receive interrupt.

0 (clear) – No interrupt will be generated from this event.

Note: The CRIER register is held in the reset state when the SFTRES bit in CMCR0 is set.

D.12.8 MSCAN12 Transmitter Flag Register (CTFLG)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Transmitter flag register (CTFLG)	$xx06	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0	0000 0111

All of the bits in this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CTCR register. A hard or soft reset will clear the register.

ABTAK2 - ABTAK0 — Abort Acknowledge

This flag acknowledges that a message has been aborted due to a pending abort request from the CPU. After a particular message buffer has been flagged empty, this flag can be used by the application software to identify whether the message has been aborted successfully or has been sent in the meantime. The flag is reset implicitly whenever the associated TXE flag is set to 0.

1 (set) – The message has been aborted.

0 (clear) – The message has not been aborted, thus has been sent out.

TXE2 - TXE0 —Transmitter Buffer Empty

This flag indicates that the associated transmit message buffer is empty, thus not scheduled for transmission. The CPU must handshake (clear) the flag after a message has been set up in the transmit buffer and is due for transmission. The MSCAN12 will set the flag after the message has been sent successfully. The flag will also be set by the MSCAN12 when the transmission request was successfully aborted due to a pending abort request (Section D.12.9). If not masked, a Transmit interrupt is pending while this flag is set.

D

A reset of this flag will also reset the Abort Acknowledge (ABTAK, see above) and the Abort Request (ABTRQ, see Section D.12.9) flags of the particular buffer.

1 (set) – The associated message buffer is empty (not scheduled).

0 (clear) – The associated message buffer is full (loaded with a message due for transmission).

Note: The CTFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.

D.12.9 MSCAN12 Transmitter Control Register (CTCR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Transmitter control register (CTCR)	$xx07	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0	0000 0000

ABTRQ2 - ABTRQ0 — Abort Request

The CPU sets this bit to request that an already scheduled message buffer (TXE = 0) shall be aborted. The MSCAN12 will grant the request when the message is not already under transmission. When a message is aborted the associated TXE and the Abort Acknowledge flag (ABTAK, see Section D.12.8) will be set and an TXE interrupt will occur if enabled. The CPU can not reset ABTRQx. ABTRQx is reset implicitly whenever the associated TXE flag is set.

1 (set) – Abort request pending.

0 (clear) – No abort request.

The software must not clear one or more of the TXE flags in CTFGL and simultaneously set the respective ABTRQ bit(s).

TXEIE2 - TXEIE0 — Transmitter Empty Interrupt Enable

1 (set) – A transmitter empty (transmit buffer available for transmission) event will result in a transmitter empty interrupt.

0 (clear) – No interrupt will be generated from this event.

Note: The CTCR register is held in the reset state when the SFTRES bit in CMCR0 is set.

D

D.12.10 MSCAN12 Identifier Acceptance Control Register (CIDAC)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Identifier acceptance control reg. (CIDAC)	$xx08	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0	0000 0000

IDAM1- IDAM0— Identifier Acceptance Mode

The CPU sets these flags to define the identifier acceptance filter organisation (see Section D.4). Table D-10 summarizes the different settings. In Filter Closed mode no messages will be accepted such that the foreground buffer will never be reloaded.

Table D-11 Identifier Acceptance Mode Settings

IDAM1	IDAM0	Identifier Acceptance Mode
0	0	Two 32 bit Acceptance Filters
0	1	Four 16 bit Acceptance Filters
1	0	Eight 8 bit Acceptance Filters
1	1	Filter Closed

IDHIT2- IDHIT0— Identifier Acceptance Hit Indicator

The MSCAN12 sets these flags to indicate an identifier acceptance hit (see Section D.4). Table D-10 summarizes the different settings.

Table D-12 Identifier Acceptance Hit Indication

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

The IDHIT indicators are always related to the message in the foreground buffer. When a message gets copied from the background to the foreground buffer the indicators are updated as well.

Note: The CIDAC register can only be written if the SFTRES bit in CMCR0 is set.

D.12.11 MSCAN12 Receive Error Counter (CRXERR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Receive error counter (CRXERR)	$xx0E	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0	0000 0000

This register reflects the status of the MSCAN12 receive error counter. The register is read only.

D.12.12 MSCAN12 Transmit Error Counter (CTXERR)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Transmit error counter (CTXERR)	$xx0F	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0	0000 0000

This register reflects the status of the MSCAN12 transmit error counter. The register is read only.

Note: Both error counters must only be read when in Sleep or Soft Reset Mode.

D.12.13 MSCAN12 Identifier Acceptance Registers (CIDAR0-7)

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

The acceptance registers of the MSCAN12 are applied on the IDR0 to IDR3 registers of incoming messages in a bit by bit manner.

For extended identifiers all four acceptance and mask registers are applied. For standard identifiers only the first two (IDAR0, IDAR1) are applied. In the latter case it is required to program the three last bits (AC2 - AC0) in the mask register CIDMR1 to “don’t care”.

Register	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
CIDAR0	$xx10	AC&	AC6	AC5	AC4	AC3	AC2	AC1	AC0	undefined
CIDAR1	$xx11	AC&	AC6	AC5	AC4	AC3	AC2	AC1	AC0	undefined
CIDAR2	$xx12	AC&	AC6	AC5	AC4	AC3	AC2	AC1	AC0	undefined
CIDAR3	$xx12	AC&	AC6	AC5	AC4	AC3	AC2	AC1	AC0	undefined

Figure D-12 Identifier acceptance registers (1ST bank)

Register	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
CIDAR4	$xx18	AC&	AC6	AC5	AC4	AC3	AC2	AC1	AC0	undefined
CIDAR5	$xx19	AC&	AC6	AC5	AC4	AC3	AC2	AC1	AC0	undefined
CIDAR6	$xx1A	AC&	AC6	AC5	AC4	AC3	AC2	AC1	AC0	undefined
CIDAR7	$xx1B	AC&	AC6	AC5	AC4	AC3	AC2	AC1	AC0	undefined

Figure D-13 Identifier acceptance registers (2ND bank)

AC7 – AC0 — Acceptance Code Bits

AC7 – AC0 comprise a user defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

Note: The CIDAR0-7 registers can only be written if the SFTRES bit in CMCR0 is set.

D.12.14 MSCAN12 Identifier Mask Registers (CIDMR0-7)

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering.

Register	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
CIDMR0	$xx14	AM&	AM6	AM5	AM4	AM3	AM2	AM1	AM0	undefined
CIDMR1	$xx15	AM&	AM6	AM5	AM4	AM3	AM2	AM1	AM0	undefined
CIDMR2	$xx16	AM&	AM6	AM5	AM4	AM3	AM2	AM1	AM0	undefined
CIDMR3	$xx17	AM&	AM6	AM5	AM4	AM3	AM2	AM1	AM0	undefined

Figure D-14 Identifier mask registers (1ST bank)

Register	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
CIDMR4	$xx1C	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	undefined
CIDMR5	$xx1D	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	undefined
CIDMR6	$xx1E	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	undefined
CIDMR7	$xx1F	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0	undefined

Figure D-15 Identifier mask registers (2^{ND}bank)

AM7 – AM0 — Acceptance Mask Bits

If a particular bit in this register is cleared this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit, before a match will be detected. The message will be accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register will not affect whether or not the message is accepted.

Bit description:

1 (set) – Ignore corresponding acceptance code register bit.

0 (clear) – Match corresponding acceptance code register and identifier bits.

Note: The CIDMR0-7 registers can only be written if the SFTRES bit in CMCR0 is set.

D.12.15 MSCAN12 Port CAN Control Register (PCTLCAN)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Port CAN control reg. (PCTLCAN)	$xx3D	0	0	0	0	0	0	PUECAN	RDRCAN	0000 0000

The following bits control pins 7 through 2 of Port CAN. Pins 1 and 0 are reserved for the RxCan (input only) and TxCan (output only) pins.

PUECAN — Pull Enable Port CAN

1 (set) – Pull mode enabled for Port CAN.

0 (clear) – Pull mode disabled for Port CAN.

The pull mode (pull-up or pull-down) for Port CAN is defined in the chip specification.

RDRCAN — Reduced Drive Port CAN

1 (set) – Reduced drive enabled for Port CAN.

0 (clear) – Reduced drive disabled for Port CAN.

D.12.16 MSCAN12 Port CAN Data Register (PORTCAN)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Port CAN data reg. (PORTCAN)	$xx3E	PCAN7	PCAN6	PCAN5	PCAN4	PCAN3	PCAN2	TxCAN	RxCAN	undefined

PCAN7 – PCAN2 — Port CAN Data Bits

Writing to PCANx stores the bit value in an internal bit memory. This value is driven to the respective pin only if DDRCANx = 1.

Reading PCANx returns

- the value of the internal bit memory driven to the pin, if DDRCANx = 1
- the value of the respective pin, if DDRCANx = 0

Reading bits 1 and 0 returns the value of the TxCan and RxCan pins, respectively.

D.12.17 MSCAN12 Port CAN Data Direction Register (DDRCAN)

	Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on reset
Port CAN data direction register (DDRCAN)	$xx3F	DDRCAN7	DDRCAN6	DDRCAN5	DDRCAN4	DDRCAN3	DDRCAN2	0	0	0000 0000

DDRCAN7 – DDRCAN2 — Data Direction Port CAN Bits

1 (set) – Respective I/O pin is configured for output.

0 (clear) – Respective I/O pin is configured for input.

D

GLOSSARY

This section contains abbreviations and specialist words used in this data sheet and throughout the industry. Further information on many of the terms may be gleaned from a variety of standard electronics text books.

$xxxx	The digits following the '$' are in hexadecimal format.
%xxxx	The digits following the '%' are in binary format.
A/D, **ADC**	Analog-to-digital (converter).
Bootstrap mode	In this mode the device automatically loads its internal memory from an external source on reset and then allows this program to be executed.
Byte	Eight bits.
CAN	Controller area network.
CCR	Condition codes register; an integral part of the CPU.
CERQUAD	A ceramic package type, principally used for EPROM and high temperature devices.
Clear	'0' — the logic zero state; the opposite of 'set'.
CMOS	Complementary metal oxide semiconductor. A semiconductor technology chosen for its low power consumption and good noise immunity.
COP	Computer operating properly. *aka* 'watchdog'. This circuit is used to detect device runaway and provide a means for restoring correct operation.
CPU	Central processing unit.
D/A, DAC	Digital-to-analog (converter).
EEPROM	Electrically erasable programmable read only memory. *aka* 'EEROM'.
EPROM	Erasable programmable read only memory. This type of memory requires exposure to ultra-violet wavelengths in order to erase previous data. *aka* 'PROM'.
ESD	Electrostatic discharge.
Expanded mode	In this mode the internal address and data bus lines are connected to external pins. This enables the device to be used in much more complex systems, where there is a need for external memory for example.

EVS	Evaluation system. One of the range of platforms provided by Motorola for evaluation and emulation of their devices.
HCMOS	High-density complementary metal oxide semiconductor. A semiconductor technology chosen for its low power consumption and good noise immunity.
I/O	Input/output; used to describe a bidirectional pin or function.
Input capture	(IC) This is a function provided by the timing system, whereby an external event is 'captured' by storing the value of a counter at the instant the event is detected.
Interrupt	This refers to an asynchronous external event and the handling of it by the MCU. The external event is detected by the MCU and causes a predetermined action to occur.
$\overline{\text{IRQ}}$	Interrupt request. The overline indicates that this is an active-low signal format.
K byte	A kilo-byte (of memory); 1024 bytes.
LCD	Liquid crystal display.
LSB	Least significant byte.
M68HC05	Motorola's family of 8-bit MCUs.
MCU	Microcontroller unit.
MI BUS	Motorola interconnect bus. A single wire, medium speed serial communications protocol.
MSB	Most significant byte.
Nibble	Half a byte; four bits.
NRZ	Non-return to zero.
Opcode	The opcode is a byte which identifies the particular instruction and operating mode to the CPU. See also: prebyte, operand.
Operand	The operand is a byte containing information the CPU needs to execute a particular instruction. There may be from 0 to 3 operands associated with an opcode. See also: opcode, prebyte.
Output compare	(OC) This is a function provided by the timing system, whereby an external event is generated when an internal counter value matches a predefined value.
PLCC	Plastic leaded chip carrier package.
PLL	Phase-locked loop circuit. This provides a method of frequency multiplication, to enable the use of a low frequency crystal in a high frequency circuit.
Prebyte	This byte is sometimes required to qualify an opcode, in order to fully specify a particular instruction. See also: opcode, operand.

Pull-down, pull-up	These terms refer to resistors, sometimes internal to the device, which are permanently connected to either ground or V_{DD}.
PWM	Pulse width modulation. This term is used to describe a technique where the width of the high and low periods of a waveform is varied, usually to enable a representation of an analog value.
QFP	Quad flat pack package.
RAM	Random access memory. Fast read and write, but contents are lost when the power is removed.
RFI	Radio frequency interference.
RTI	Real-time interrupt.
ROM	Read-only memory. This type of memory is programmed during device manufacture and cannot subsequently be altered.
RS-232C	A standard serial communications protocol.
SAR	Successive approximation register.
SCI	Serial communications interface.
Set	'1' — the logic one state; the opposite of 'clear'.
Silicon glen	An area in the central belt of Scotland, so called because of the concentration of semiconductor manufacturers and users found there.
Single chip mode	In this mode the device functions as a self contained unit, requiring only I/O devices to complete a system.
SPI	Serial peripheral interface.
Test mode	This mode is intended for factory testing.
TTL	Transistor-transistor logic.
UART	Universal asynchronous receiver transmitter.
VCO	Voltage controlled oscillator.
Watchdog	*see* 'COP'.
Wired-OR	A means of connecting outputs together such that the resulting composite output state is the logical OR of the state of the individual outputs.
Word	Two bytes; 16 bits.
$\overline{\text{XIRQ}}$	Non-maskable interrupt request. The overline indicates that this has an active-low signal format.

THIS PAGE LEFT BLANK INTENTIONALLY

INDEX

In this index numeric entries are placed first; page references in *italics* indicate that the reference is to a figure.

D

E

F

G

H

I

L

M

N

O

P

R

S

T

W

CUSTOMER FEEDBACK QUESTIONNAIRE (CAN PROTOCOL)

Motorola wishes to continue to improve the quality of its documentation. We would welcome your feedback on the publication you have just received. Having used the document, please complete this card (or a photocopy of it, if you prefer).

1. How would you rate the quality of the document? Check one box in each category.

	Excellent			Poor		Excellent			Poor
Organization	❑	❑	❑	❑	Tables	❑	❑	❑	❑
Readability	❑	❑	❑	❑	Table of contents	❑	❑	❑	❑
Understandability	❑	❑	❑	❑	Index	❑	❑	❑	❑
Accuracy	❑	❑	❑	❑	Page size/binding	❑	❑	❑	❑
Illustrations	❑	❑	❑	❑	Overall impression	❑	❑	❑	❑

Comments: _____

2. What is your intended use for this document? If more than one option applies, please rank them (1, 2, 3).

Selection of device for new application ❑
System design ❑
Training purposes ❑
Other ❑ Please specify: _____

3. How well does this manual enable you to perform the task(s) outlined in question 2?

Completely			Not at all
❑	❑	❑	❑

Comments: _____

4. How easy is it to find the information you are looking for?

Easy			Difficult
❑	❑	❑	❑

Comments: _____

5. Is the level of technical detail in the following sections sufficient to allow you to understand how the device functions?

		Too little detail				Too much detail
SECTION 1	INTRODUCTION	❑	❑	❑	❑	❑
SECTION 2	BASIC CONCEPTS	❑	❑	❑	❑	❑
SECTION 3	MESSAGE TRANSFER	❑	❑	❑	❑	❑
SECTION 4	ERROR HANDLING	❑	❑	❑	❑	❑
SECTION 5	FAULT CONFINEMENT	❑	❑	❑	❑	❑
SECTION 6	BIT TIMING REQUIREMENTS	❑	❑	❑	❑	❑
SECTION 7	INCREASING OSCILLATOR TOLERANCE	❑	❑	❑	❑	❑
SECTION 8	THE PHYSICAL LAYER	❑	❑	❑	❑	❑
SECTION 9	INTRODUCTION	❑	❑	❑	❑	❑
SECTION 10	BASIC CONCEPTS	❑	❑	❑	❑	❑
SECTION 11	MESSAGE TRANSFER	❑	❑	❑	❑	❑
SECTION 12	ERROR HANDLING	❑	❑	❑	❑	❑
SECTION 13	FAULT CONFINEMENT	❑	❑	❑	❑	❑
SECTION A	THE MOTOROLA CAN (MCAN) MODULE	❑	❑	❑	❑	❑
SECTION B	TOUCAN	❑	❑	❑	❑	❑
SECTION C	THE MOTOROLA SCALEABLE CAN (MSCAN08) MODULE	❑	❑	❑	❑	❑
SECTION D	THE MOTOROLA SCALEABLE CAN (MSCAN12) MODULE	❑	❑	❑	❑	❑

6. Have you found any errors? If so, please comment: _____

7. From your point of view, is anything missing from the document? If so, please say what: _____

– Cut along this line to remove –

8. How could we improve this document? _____

_____

_____

9. How would you rate Motorola's documentation?

	Excellent			Poor
– In general	❑	❑	❑	❑
– Against other semiconductor suppliers	❑	❑	❑	❑

10. Which semiconductor manufacturer provides the best technical documentation? _____
11. Which company (in any field) provides the best technical documentation? _____
12. How many years have you worked with microprocessors?

Less than 1 year ❑ 1–3 years ❑ 3–5 years ❑ More than 5 years ❑

– Second fold back along this line –

By air mail
Par avion

IBRS NUMBER PHQ-B/207/G
CCRI NUMERO PHQ-B/207/G

NE PAS AFFRANCHIR
NO STAMP REQUIRED

REPONSE PAYEE
GRANDE-BRETAGNE

Motorola Ltd.,
Colvilles Road,
Kelvin Industrial Estate,
EAST KILBRIDE,
G75 8BR.
GREAT BRITAIN.

F.A.O. Technical Publications Manager
(re: BCANPSV2.0/D)

MOTOROLA
Semiconductor Products Sector

– First fold back along this line –

– Cut along this line to remove –

– Third fold back along this line –

13. Currently there is some discussion in the semiconductor industry regarding a move towards providing data sheets in electronic form. If you have any opinion on this subject, please comment. _____

_____

14. We would be grateful if you would supply the following information (at your discretion), or attach your card.

Name: _____ Phone No: _____

Position: _____ FAX No: _____

Department: _____

Company: _____

Address: _____

_____

Thank you for helping us improve our documentation,
Technical Publications Manager, Motorola Ltd., Scotland.

– Finally, tuck this edge into opposite flap –

How to reach us:

Mfax: RMFAX0@email.sps.mot.com – TOUCHTONE 1-602-244-6609

US & CANADA ONLY: http://sps.motorola.com/mfax

INTERNET: SPS home-http://motorola.com/sps

USA/EUROPE: Motorola Literature Distribution; P.O. Box 5405; Denver, Colorado 80217. 1-303-675-2140

JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, Toshikatsu Otsuki, 6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-3521-8315

HONG KONG: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

MOTOROLA