Recital

The acceptance and introduction of serial communication to more and more applications has led to requirements that the assignment of message identifiers to communication functions be standardized for certain applications. These applications can be realized with CAN more comfortably, if the dress range that originally has been defined by 11 identifier bits is enlarged.
Therefore a second message format ('extended format') is introduced that provides a larger address range defined by 29 bits. This will relieve the system designer from compromizes with respect to defining well-structured naming schemes. Users of CAN who do not need the identifer range offered by the extended format, can rely on the conventional 11 bit identifier range ('standard format') further on. In this case they can make use of the CAN implementations that are already available on the market, or of new controllers that implement both formats.
In order to distinguish standard and extended format the first reserved bit of the CAN message format, as it is defined in CAN Specification 1.2, is used. This is done in such a way that the message format in CAN Specification 1.2 is equivalent to the standard format and therefore is still valid. Furthermore, the extended format has been defined so that messages in standard format and extended format can coexist within the same network.

This CAN Specification 2.0 consists of two parts, with

- Part A describing the CAN message format as it is defined in CAN Specification 1.2;
- Part B describing both standard and extended message formats.

In order to be compatible with this CAN Specification 2.0 it is required that a CAN implementation be compatible with either Part A or Part B.

Note

CAN implementations that are designed according to part A of this or according to previous CAN Specifications, and CAN implementations that are designed according to part B of this specification can communicate with each other as long as it is not made use of the extended format.

PART A

1 INTRODUCTION

The Controller Area Network (CAN) is a serial communications protocol which efficiently supports distributed realtime control with a very high level of security.
Its domain of application ranges from high speed networks to low cost multiplex wiring.
In automotive electronics, engine control units, sensors, anti-skid-systems, etc. are connected using CAN with bitrates up to 1 Mbit/s. At the same time it is cost effective to build into ve-hicle body electronics, e.g. lamp clusters, electric windows etc. to replace the wiring harness otherwise required.
The intention of this specification is to achieve compatibility between any two CAN implementations. Compatibility, however, has different aspects regarding e.g. electrical features and
the interpretation of data to be transferred. To achieve design transparency and implementation flexibility CAN has been subdivided into different layers.

- the (CAN-) object layer
- the (CAN-) 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

- finding which messages are to be transmitted,
- deciding which messages received by the transfer layer are actually to be used,
- providing an interface to the application layer related hardware.

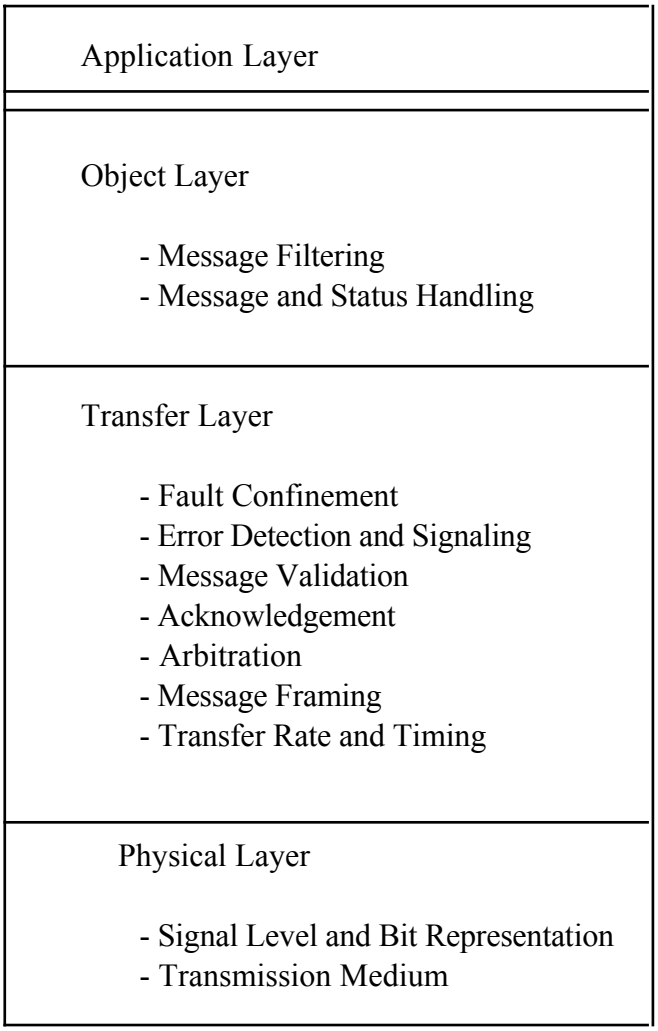
There is much freedom in defining object handling. The scope of the transfer layer mainly is the transfer protocol, i.e. controlling the framing, performing arbitration, error checking, error signalling and fault confinement. Within the transfer layer it is decided whether the bus is free for starting a new transmission or whether a reception is just starting. Also some general features of the bit timing are regarded as part of the transfer layer. It is in the nature of the trans-fer layer that there is no freedom for modifications.
The scope of the physical layer is the actual transfer of the bits between the different nodes with respect to all electrical properties. Within one network the physical layer, of course has to be the same for all nodes. There may be, however, much freedom in selecting a physical layer.
The scope of this specification is to define the transfer layer and the consequences of the CAN protocol on the surrounding layers.

2 BASIC CONCEPTS

CAN has the following properties

- prioritization of messages
- guarantee of latency times
- configuration flexibility
- multicast reception with time synchronization
- system wide data consistency
- multimaster
- error detection and error signalling
- automatic retransmission of corrupted messages as soon as the bus is idle again.
- distinction between temporary errors and permanent failures of nodes and autonomous switching off of defect nodes.

Layered Structure of a CAN Node



- The Physical Layer defines how signals are actually transmitted. Within this specification the physical layer is not defined so as to allow transmission medium and signal level implementations to be optimized for their application.

- The Transfer Layer represents the kernel of the CAN protocol. It presents messages recei-ved to the object layer and accepts messages to be transmitted from the object layer. The transfer layer is responsible for bit timing and synchronization, message framing, arbitra-tion, acknowledgement, error detection and signalling, and fault confinement.

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

The scope of this specification is to define the transfer layer and the consequences of the CAN protocol on the surrounding layers.

Messages

Information on the bus is sent in fixed format messages of different but limited length (see section 3: Message Transfer). When the bus is free any connected unit may start to transmit a new message.

Information Routing
In CAN systems a CAN node does not make use of any information about the system configuration (e.g. station addresses). This has several important consequences.

System Flexibility: Nodes can be added to the CAN network without requiring any change in the software or hardware of any node and application layer.

Message Routing: The content of a message is named by an IDENTIFIER. The IDENTI- FIER 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.

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

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

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

Priorities
The IDENTIFIER defines a static message priority during bus access.

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 are named by the same IDENTIFIER.

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

Arbitration

Whenever the bus is free, any unit may start to transmit a message. If 2 or more units start transmitting messages at the same time, the bus access conflict is resolved by bitwise 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 unit may continue to send. When a 'recessive' level is sent and a 'dominant' level is monitored (see Bus Values), the unit has lost arbitration and must withdraw without sending one more bit.

Safety

In order to achieve the utmost safety of data transfer, powerful measures for error detection, signalling and self-checking are implemented in every CAN node.

Error Detection

For detecting errors the following measures have been taken:

- Monitoring (transmitters compare the bit levels to be transmitted with the bit levels detected on the bus)
- Cyclic Redundancy Check
- Bit Stuffing
- Message Frame Check

Performance of Error Detection

The error detection mechanisms have the following properties:

- all global errors are detected.
- all local errors at transmitters are detected.
- up to 5 randomly distributed errors in a message are detected.
- burst errors of length less than 15 in a message are detected.
- errors of any odd number in a message are detected.

Total residual error probability for undetected corrupted messages: less than

$$\text{message error rate} \cdot 4.7 \cdot 10^{-11}.$$

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 29 bit times, if there is no further error.

Fault Confinement
CAN nodes are able to distinguish short disturbances frompermanent failures. Defective nodes are switched off.

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

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.

Bus values
The bus can have one of two complementary logical values: 'dominant' or 'recessive'. During simultaneous transmission of 'dominant' and 'recessive' bits, the resulting bus value will be 'dominant'. For example, in 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.

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

Sleep Mode / Wake-up
To reduce the system's power consumption, a CAN-device may be set into sleep mode with-out any internal activity and with disconnected bus drivers. 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 be waiting for the system's oscillator to stabilize and it will 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 "on-bus" again.
In order to wake up other nodes of the system, which are in sleep-mode, 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 Frame Types

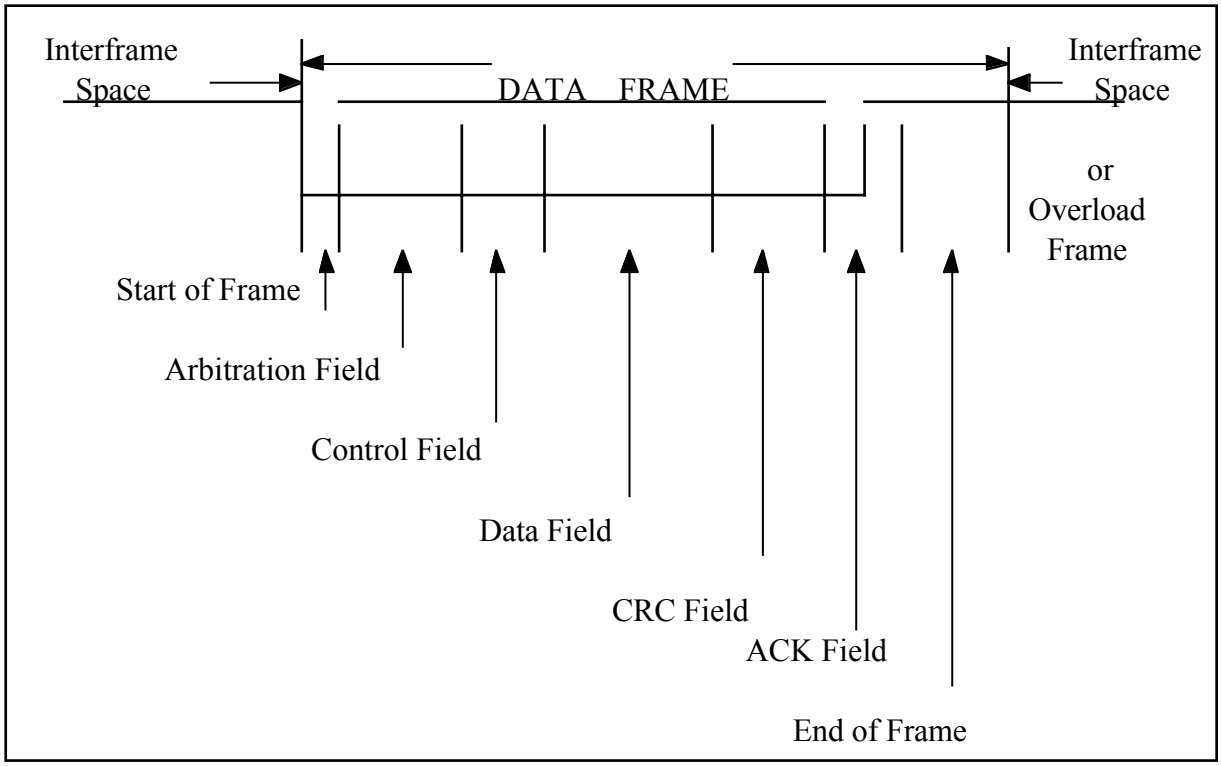
Message transfer is manifested and controlled by four different frame types:

A DATA FRAME carries data from a transmitter to the receivers.
A REMOTE FRAME is transmitted by a bus unit to request the transmission of the DATA FRAME with the same IDENTIFIER.
An ERROR FRAME is transmitted by any unit 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.

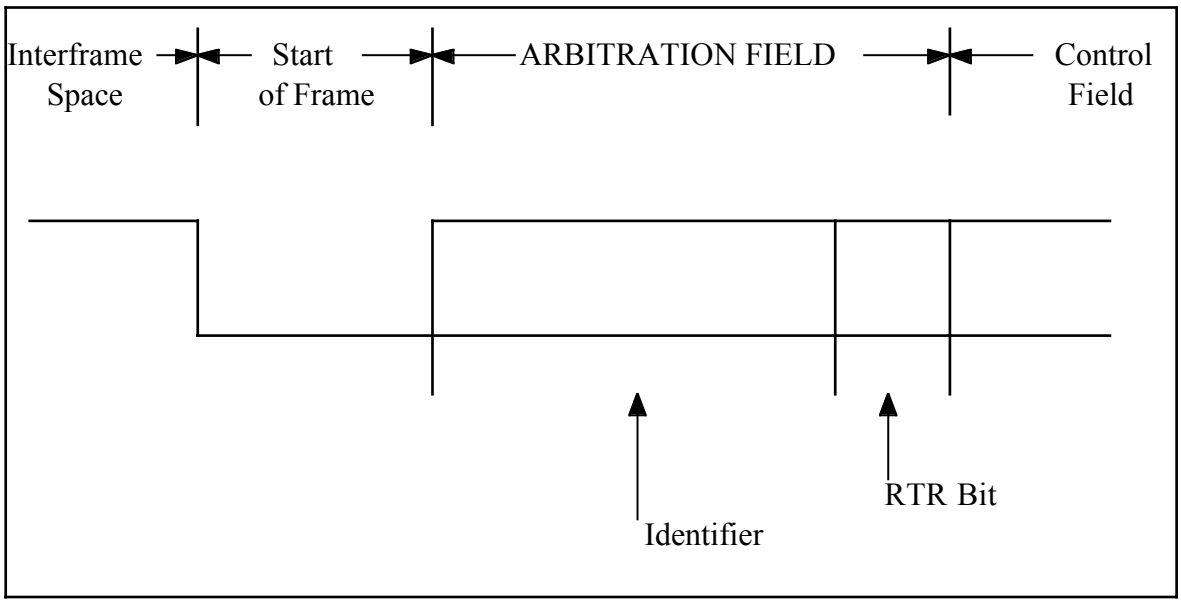
3.1.1 DATA FRAME

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.



START OF FRAME
marks the beginning of DATA FRAMES and REMOTE FRAMES. It consists of a single 'dominant' bit.
A station is only allowed to start transmission when the bus is idle (see BUS IDLE). All stations have to synchronize to the leading edge caused by START OF FRAME (see 'HARD SYNCHRONIZATION') of the station starting transmission first.

ARBITRATION FIELD
The ARBITRATION FIELD consists of the IDENTIFIER and the RTR-BIT.

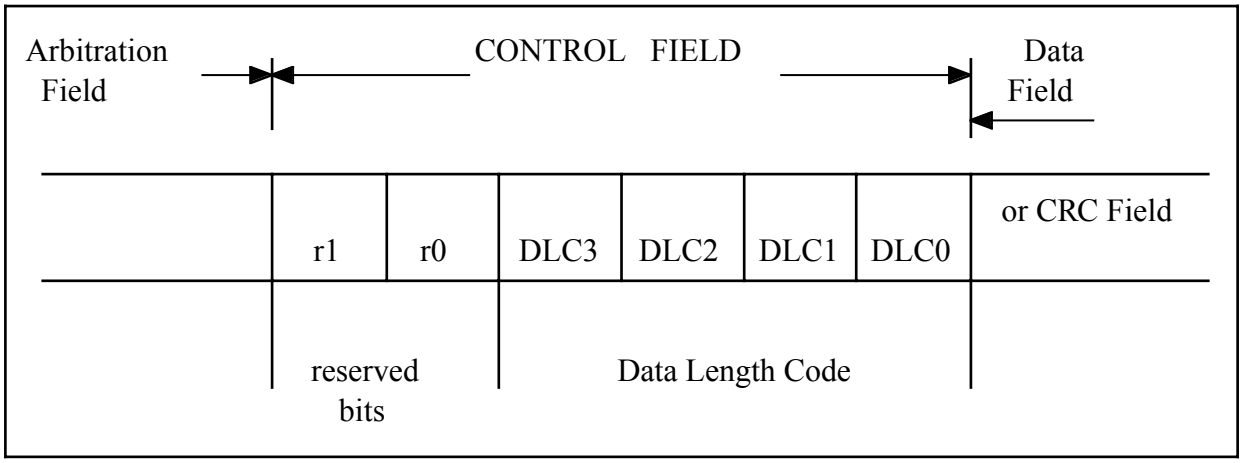


IDENTIFIER
The IDENTIFIER's length is 11 bits. These bits are transmitted in the order from ID-10 to ID-0. The least significant bit is ID-0. The 7 most significant bits (ID-10 - ID-4) must not be all 'recessive'.

RTR BIT
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'.

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 have to be sent 'dominant'. Receivers accept 'dominant' and 'recessive' bits in all combinations.

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.



Coding of the number of data bytes by the DATA LENGTH CODE

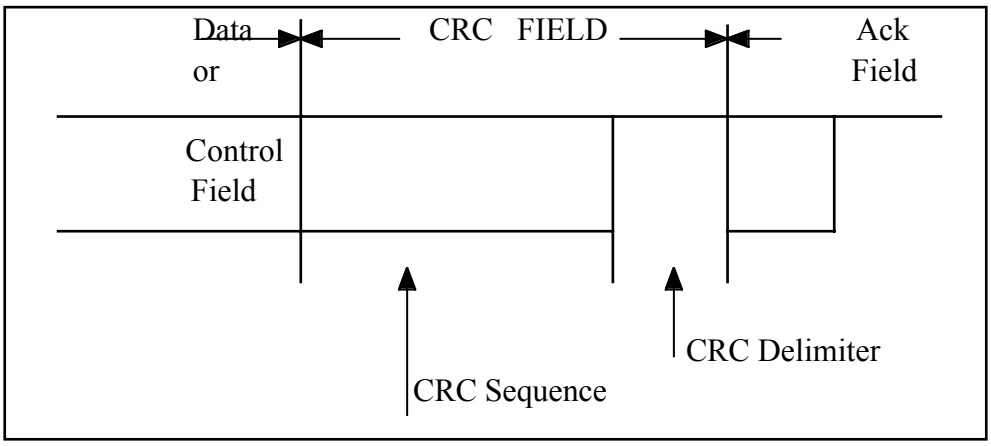
abbreviations: d 'dominant'
r 'recessive'

Number of Data Bytes	Data Length Code			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d	d	d

DATA FRAME: admissible numbers of data bytes: {0,1,....,7,8}
Other values may not be used.

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

CRC FIELD
contains the CRC SEQUENCE followed by a CRC DELIMITER.



CRC SEQUENCE
The frame check sequence is derived from a cyclic redundancy code best suited for 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 poly-nomial, the coefficients of which 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:

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

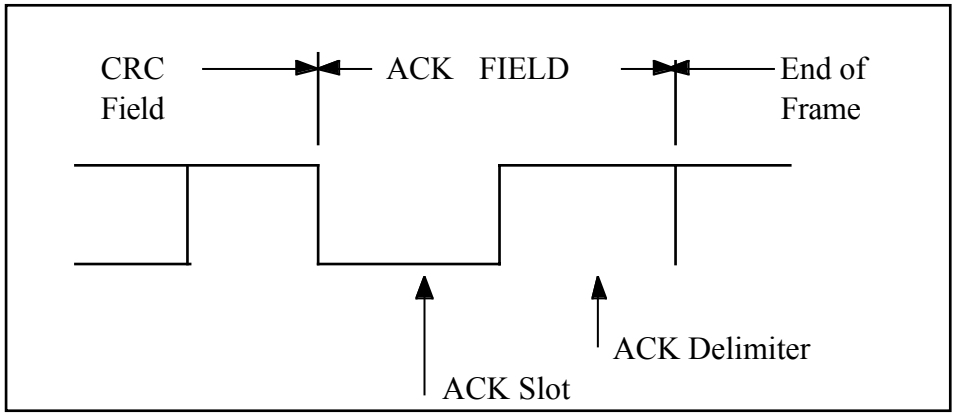
The remainder of this polynomial division is the CRC SEQUENCE transmitted over the bus. In order to implement this function, a 15 bit shift register CRC_RG(14:0) can be used. If NXTBIT denotes the next bit of the bit stream, given by the destuffed bit sequence from START OF FRAME until the end of the 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;                                  // 1 position
    IF CRCNXT THEN
            CRC_RG(14:0) = CRC_RG(14:0) EXOR (4599hex);
    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 contains the CRC sequence.

CRC DELIMITER
The CRC SEQUENCE is followed by the CRC DELIMITER which consists of a single 'recessive' bit.

ACK FIELD
The ACK FIELD is two bits long and contains the ACK SLOT and the ACK DELIMITER. In the ACK FIELD the transmitting station sends two 'recessive' bits.
A RECEIVER which has received a valid message correctly, reports this to the TRANSMIT-TER by sending a 'dominant' bit during the ACK SLOT (it sends 'ACK').



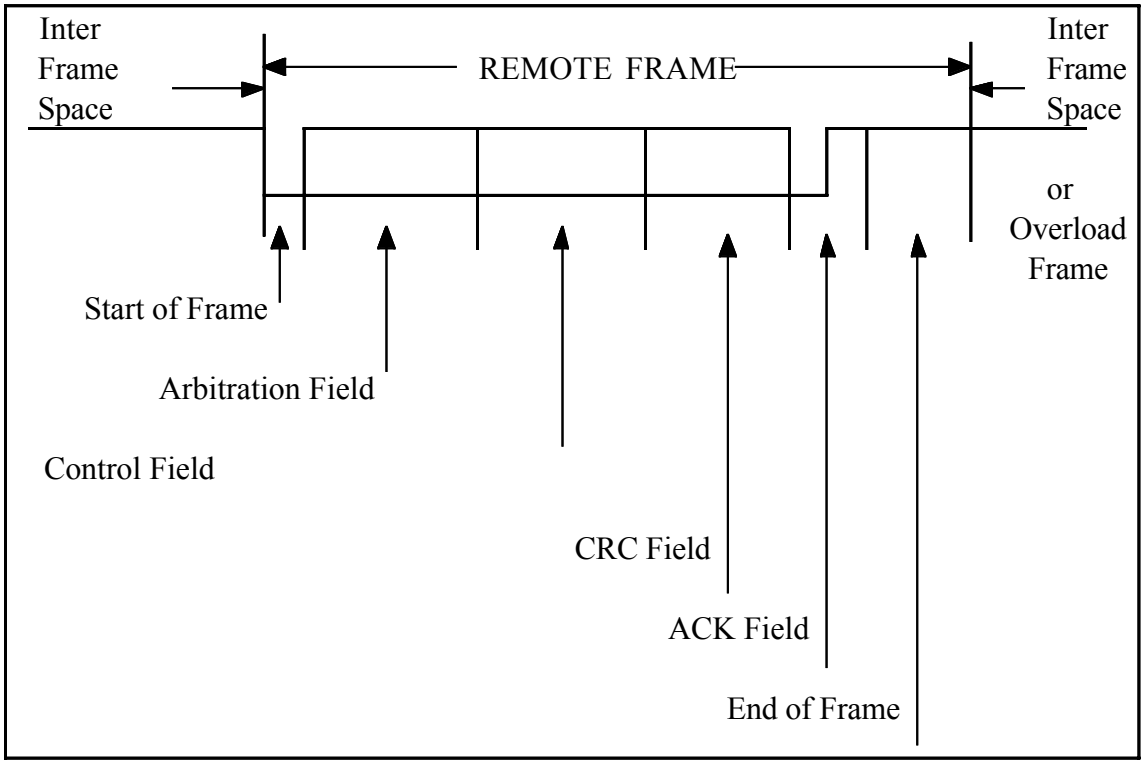
ACK SLOT
All stations having received the matching CRC SEQUENCE report this within the ACK SLOT by superscribing the 'recessive' bit of the TRANSMITTER by a 'dominant' bit.

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).

END OF FRAME
Each DATA FRAME and REMOTE FRAME is delimited by a flag sequence consisting of seven 'recessive' bits.

3.1.2 REMOTE FRAME

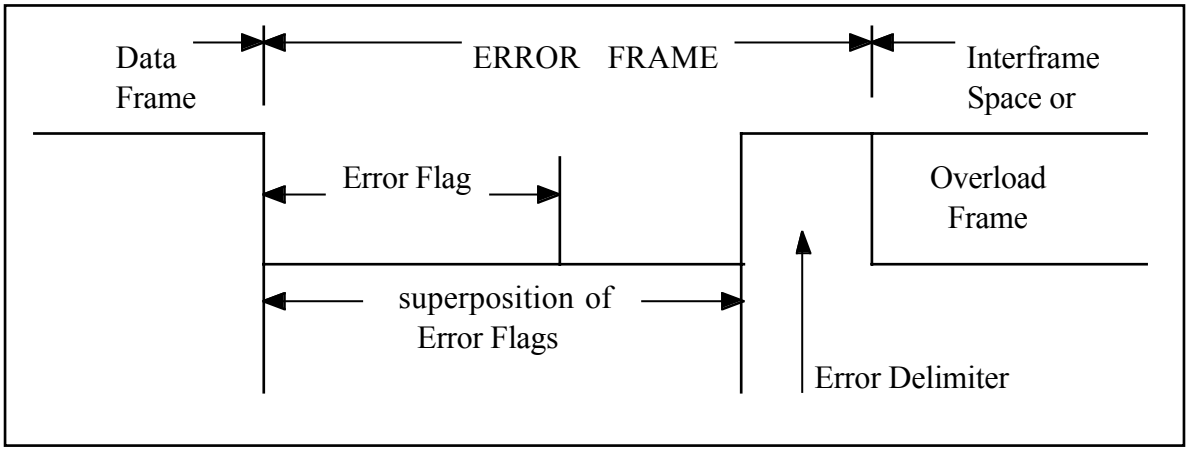
A station acting as a RECEIVER for certain data can initiate the transmission of the respective data by its source node by sending a REMOTE FRAME.
A REMOTE FRAME is composed of six different bit fields:
START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, CRC FIELD, ACK FIELD, END OF FRAME.
Contrary to DATA FRAMEs, the RTR bit of REMOTE FRAMEs is 'recessive'. There is no DATA FIELD, independent of the values of the DATA LENGTH CODE which may be sig-ned any value within the admissible range 0...8. The value is the DATA LENGTH CODE of the corresponding DATA 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’).

3.1.3 ERROR FRAME

The ERROR FRAME consists of two different fields. The first field is given by the superposition of ERROR FLAGs contributed from different stations. The following 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 3 bit times (if there is a local error at an 'error passive' receiver). Therefore the bus should not be loaded to 100%.

ERROR FLAG
There are 2 forms of an ERROR FLAG: an ACTIVE ERROR FLAG and a PASSIVE ERROR FLAG.

1. The 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' station detecting an error condition signals this by transmission of an ACTIVE ERROR FLAG. The ERROR FLAG's form violates the law of bit stuffing (see CODING) 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 stations detect an error condition and on their part start transmission of 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 stations. The total length of this sequence varies between a minimum of six and a maximum of twelve bits.
An 'error passive' station detecting an error condition tries to signal this by transmission of a PASSIVE ERROR FLAG. The 'error passive' station 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 6 equal bits have been detected.

ERROR DELIMITER
The ERROR DELIMITER consists of eight 'recessive' bits.
After transmission of an ERROR FLAG each station sends 'recessive' bits and monitors the bus until it detects a 'recessive' bit. Afterwards it starts transmitting seven more ‘recessive' bits.

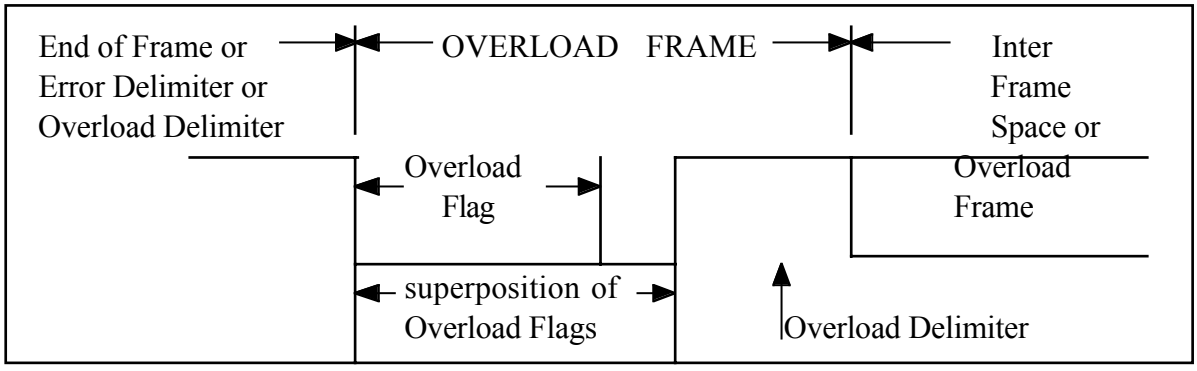
3.1.4 OVERLOAD FRAME

The OVERLOAD FRAME contains the two bit fields OVERLOAD FLAG and OVERLOAD DELIMITER.
There are two kinds of OVERLOAD conditions, which both lead to the transmission of an OVERLOAD FLAG:

1. The internal conditions of a receiver, which requires a delay of the next DATA FRAME or REMOTE FRAME.

2. Detection of a 'dominant' bit during INTERMISSION.

The start of an OVERLOAD FRAME due to OVERLOAD condition 1 is only allowed to be started at the first bit time of an expected INTERMISSION, whereas OVERLOAD FRAMEs due to OVERLOAD condition 2 start one bit after detecting the 'dominant' bit.



At most two OVERLOAD FRAMEs may be generated to delay the next DATA or REMOTE FRAME.
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 stations also detect an OVERLOAD condition and on their part start transmission of an OVERLOAD FLAG. (In case that there is a 'dominant' bit detected during the 3rd 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 causing an error condition).

OVERLOAD DELIMITER
consists of eight 'recessive' bits.

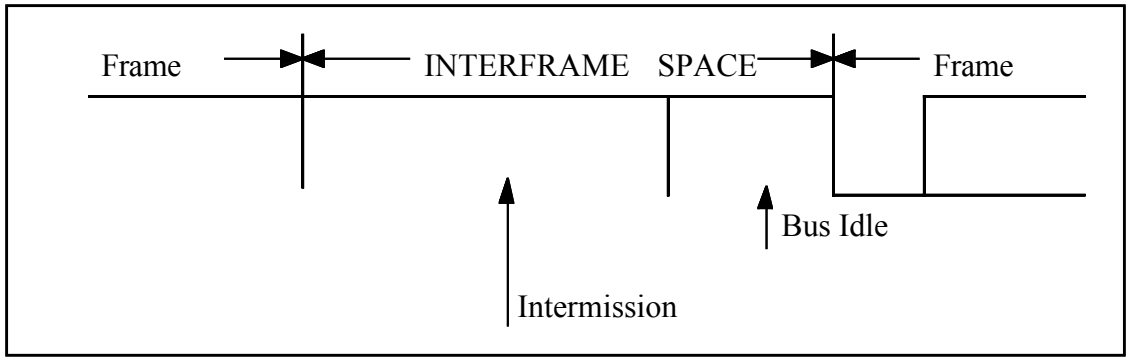
The OVERLOAD DELIMITER is of the same form as the ERROR DELIMITER. After transmission of an OVERLOAD FLAG the station monitors the bus until it detects a transition from a 'dominant' to a 'recessive' bit. At this point of time every bus station has finished sending its OVERLOAD FLAG and all stations start transmission of seven more 'recessive' bits in coincidence.

3.1.5 INTERFRAME SPACING

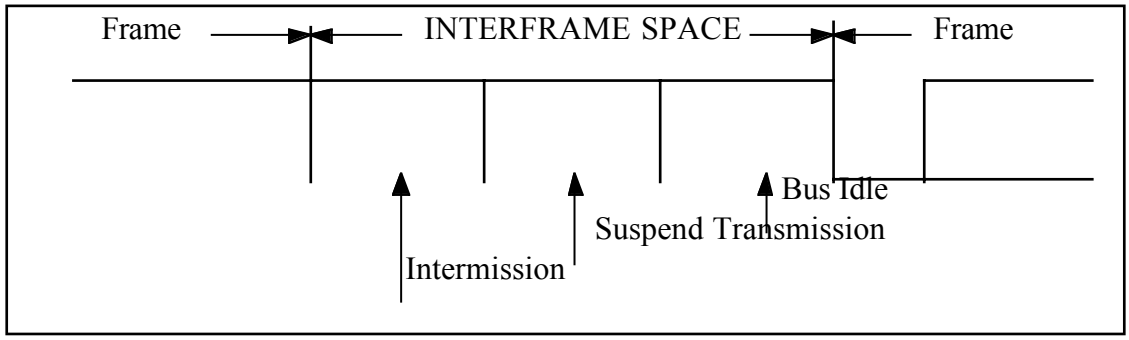
DATA FRAMEs and REMOTE FRAMEs are separated from preceding frames whatever type they are (DATA FRAME, REMOTE FRAME, ERROR FRAME, OVERLOAD FRAME) by a bit field called INTERFRAME SPACE. In contrast, OVERLOAD FRAMEs and ERROR FRAMEs are not preceded by an INTERFRAME SPACE and multiple OVERLOAD FRAMEs are not separated by an INTERFRAME SPACE.

INTERFRAME SPACE
contains the bit fields INTERMISSION and BUS IDLE and, for 'error passive' stations, which have been TRANSMITTER of the previous message, SUSPEND TRANSMISSION.

For stations which are not 'error passive' or have been RECEIVER of the previous message:



For 'error passive' stations which have been TRANSMITTER of the previous message:



INTERMISSION
consists of three 'recessive' bits.

During INTERMISSION no station is allowed to start transmission of a DATA FRAME or REMOTE FRAME. The only action to be taken is signalling an OVERLOAD condition.

BUS IDLE
The period of BUS IDLE may be of arbitrary length. The bus is recognized to be free and any station having something to transmit can access the bus. A message, which is pending for transmission 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.

SUSPEND TRANSMISSION
After an 'error passive' station has transmitted a message, 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 station) starts, the station will become receiver of this message.

3.2 Definition of TRANSMITTER / RECEIVER

TRANSMITTER
A unit originating a message is called "TRANSMITTER" of message. The unit stays TRANSMITTER until the bus is idle or the unit loses ARBITRATION.

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

4 MESSAGE VALIDATION

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

Transmitter:
The message is valid for the transmitter, if there is no error until the end of END OF FRAME. If a message is corrupted, retransmission will follow automatically and according to 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.

Receivers:
The message is valid for the receivers, if there is no error until the last but one bit of END OF FRAME.

5 CODING

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 of fixed form as well and 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'.

6 ERROR HANDLING

6.1 Error Detection

There are 5 different error types (which are not mutually exclusive):

- BIT ERROR
  A unit that is sending a bit on the bus also monitors the bus. A BIT ERROR has to be detected at that bit time, when the bit value that is monitored is different from the bit value that is sent. An exception is the sending of a 'recessive' bit during the stuffed bit stream of the ARBITRATION FIELD or during the ACK SLOT. Then no BIT ERROR occurs when a ‘dominant' bit is monitored. A TRANSMITTER sending a PASSIVE ERROR FLAG and detecting a 'dominant' bit does not interprete this as a BIT ERROR.

- STUFF ERROR
  A STUFF ERROR has to be detected at the bit time of the 6th consecutive equal bit level in a message field that should be coded by the method of bit stuffing.

- 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 has to be detected, if the calculated result is not the same as that received in the CRC sequence.

- FORM ERROR
  FORM ERROR has to be detected when a fixed-form bit field contains one or more illegal bits.

- ACKNOWLEDGEMENT ERROR
  An ACKNOWLEDGEMENT ERROR has to be detected by a transmitter whenever it does not monitor a 'dominant' bit during ACK SLOT.

6.2 Error Signalling

A station detecting an error condition signals this by transmitting an ERROR FLAG. For an 'error active' node it is an ACTIVE ERROR FLAG, for an 'error passive' node it is a PAS-SIVE ERROR FLAG. Whenever a BIT ERROR, a STUFF ERROR, a FORM ERROR or an ACKNOWLEDGEMENT ERROR is detected by any station, transmission of an ERROR FLAG is started at the respective station at the next bit.
Whenever a CRC ERROR is detected, transmission of an ERROR FLAG starts at the bit following the ACK DELIMITER, unless an ERROR FLAG for another error condition has already been started.

7 FAULT CONFINEMENT

With respect to fault confinement a unit may be in one of three states:

- 'error active'
- 'error passive'
- 'bus off'

An 'error active' unit can normally take part in bus communication and sends an ACTIVE ERROR FLAG when an error has been detected.
An 'error passive' unit 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' unit will wait before initiating a further transmission. (See SUSPEND TRANSMISSION)
A 'bus off' unit is not allowed to have any influence on the bus. (E.g. output drivers switched off.)
For fault confinement two counts are implemented in every bus unit:

1) TRANSMIT ERROR COUNT
2) RECEIVE ERROR COUNT

These counts are modified according to the following rules:
(note that 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:
If the TRANSMITTER is 'error passive' and detects an ACKNOWLEDGEMENT ERROR

because of not detecting a 'dominant' ACK and does not detect a 'dominant' bit while sending its PASSIVE ERROR FLAG.

Exception 2:
If the TRANSMITTER sends an ERROR FLAG because a STUFF ERROR occurred during ARBITRATION whereby the STUFFBIT is located before the RTR bit, and should have been 'recessive', and has been sent as 'recessive' but monitored as 'dominant'.

In exceptions 1 and 2 the TRANSMIT ERROR COUNT is not changed.

4. If an TRANSMITTER detects a BIT ERROR while sending an ACTIVE ERROR FLAG or an OVERLOAD FLAG the TRANSMIT ERROR COUNT is increased by 8.

5. If an 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, PASSIVE ERROR FLAG or OVERLOAD FLAG. After detecting the 14th con- secutive 'dominant' bit (in case of an ACTIVE ERROR FLAG or an OVERLOAD FLAG) or after detecting the 8th consecutive 'dominant' bit following a PASSIVE ERROR FLAG, 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 let- ting 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. An 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 occurrances 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 of advantage to provide means to test for this condition.

Note:
Start-up / Wake-up:
If during system start-up only 1 node is online, 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.

8 BIT TIMING REQUIREMENTS

NOMINAL BIT RATE

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

NOMINAL BIT TIME

NOMINAL BIT TIME = 1 / NOMINAL BIT RATE

The Nominal Bit Time can be thought of as being divided into separate non-overlapping time segments. These segments

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

form the bit time as shown in figure 1.

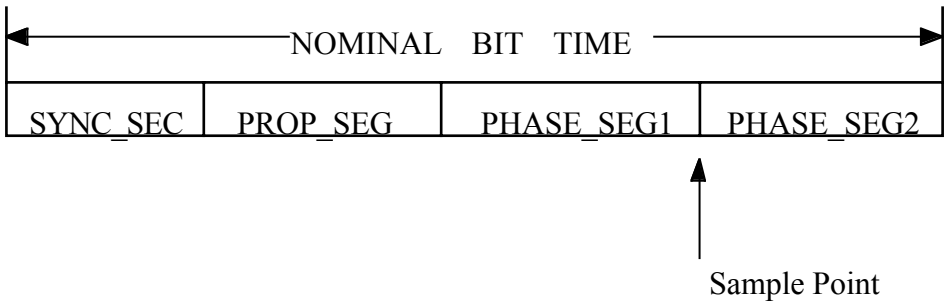


Fig. 1 Partition of the Bit Time

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.

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.

PHASE SEG1, PHASE SEG2
These Phaes-Buffer-Segments are used to compensate for edge phase errors. These segments can be lengthened or shortened by resynchronization.

SAMPLE POINT
The SAMPLE POINT is the point of time at which the bus level is read and interpreted as the value of that respective bit. It's location is at the end of PHASE_SEG1.

INFORMATION PROCESSING TIME
The INFORMATION PROCESSING TIME is the time segment starting with the SAMPLE POINT reserved for calculation the subsequent bit level.

TIME QUANTUM
The TIME QUANTUM is a fixed unit of time derived from the oscillator period. There exists a programmable prescaler, with integral values, ranging at least from 1 to 32. Starting with the MINIMUM TIME QUANTUM, the TIME QUANTUM can have a length of

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

with m the value of the prescaler.

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 has to be programmable at least from 8 to 25.

SYNCHRONIZATION

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 SYNCH-RONIZATION to lie within the SYNCHRONIZATION SEGMENT of the restarted bit time.

RESYNCHRONIZATION JUMP WIDTH
As a result of RESYNCHRONIZATION PHASE_SEG1 may be lengthened or PHASE_ SEG2 may be shortened. The amount of lengthening or shortening of the PHASE BUFFER SEGMENTs has an upper bound given by the RESYNCHRONIZATION JUMP WIDTH. The RESYNCHRONIZATION JUMP WIDTH shall be programmable between 1 and min(4, PHASE_SEG1).
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 resynchronizing a bus unit to the bit stream during a frame. The maximum length between two transitions which can be used for resynchronization is 29 bit times.

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 within SYNC_SEG.
- e > 0 if the edge lies before the SAMPLE POINT.
- e < 0 if the edge lies after the SAMPLE POINT of the previous bit.

RESYNCHRONIZATION
The effect of a RESYNCHRONIZATION is the same as that of a HARD SYNCHRONIZA-TION, when the magnitude of the PHASE ERROR of the edge which causes the RESYN-CHRONIZATION is less than or equal to the programmed value of the RESYNCHRO-NIZATION 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.

- and if the PHASE ERROR is negative, then PHASE_SEG2 is shortened by an amount equal to the RESYNCHRONIZATION JUMP WIDTH.

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 case of low bit rates) fulfilling the rules 1 and 2 will be used for RESYNCHRONIZATION with the exception that a node transmitting a dominant bit 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.

9 INCREASING CAN OSCILLATOR TOLERANCE

This section describes an upwards compatible modification of the CAN protocol, as specified in sections 1 to 8.

9.1 Protocol Modifications

In order to increase the maximum oscillator tolerance from the 0.5% currently possible to 1.5%, the following modifications, which are upwards compatible to 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 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.

This modifications allow a maximum oscillator tolerance of 1.58% and the use of a ceramic resonator at a bus speed of up to 125 Kbits/second. For the full bus speed range of the CAN protocol, still a quartz oscillator is 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, have all to be 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.