BOSCH

CAN with Flexible Data-Rate

Specification
Version 1.0
(released April 17th, 2012)

can FD

Recital

The acceptance and introduction of serial communication to more and more applications has led to increasing demand for bandwidth in CAN communication and caused system developers to look for alternative communication options in certain applications. These applications can be realized more comfortably with the new protocol CAN FD that allows data rates higher than 1 MBit/s and payloads longer 8 bytes per frame.

CAN FD shares the physical layer, with the CAN protocol as defined in the BOSCH CAN Specification 2.0. The frame format however, is different. There are two new control bits in the CAN FD frame, the first enabling the new frame format with different data length coding and the second optionally switching to a faster bit rate after the arbitration is decided. New CRC polynomials are introduced to secure the longer CAN FD frames with the same Hamming distance as in the proven CAN protocol.

The CAN FD frame format has been defined so that messages in CAN frame format and in CAN FD frame format can coexist within the same network. The BOSCH CAN Specification 2.0 remains valid without any modification as an independent, self-contained CAN bus protocol specification. The coexistence is assured by the requirement, that in order to be compatible with this CAN FD specification it is required that a CAN FD implementation be compatible with this CAN FD specification as well as with the BOSCH CAN Specification 2.0.

In order to be compatible with this CAN FD specification it is required that a CAN FD implementation be compatible with this specification as well as with ISO 11898-1.

Note: CAN FD implementations that are designed according to this specification and CAN implementations that are designed according to the BOSCH CAN Specification 2.0 can communicate with each other as long as it is not made use of the CAN FD frame format. This enables CAN systems to migrate gradually into CAN FD systems. In the introductory phase, it is possible to use CAN FD only in specific operation modes, e.g. software-download at end-of-line programming, while other controllers that do not support CAN FD are kept in standby.

1 Introduction 3

2 Basic Concepts 5

3 Message Transfer 8

3.1 Frame Formats 8

3.2 Frame Types 8

3.3 Operation Modes 19

4 Message Validation 21

4.1 Message Filtering 21

5 Coding 22

6 Error Handling 23

6.1 Error Detection 23

6.2 Error Signalling 23

7 Fault Confinement 24

8 Bit Timing Requirements 26

8.1 Transceiver Delay Compensation 30

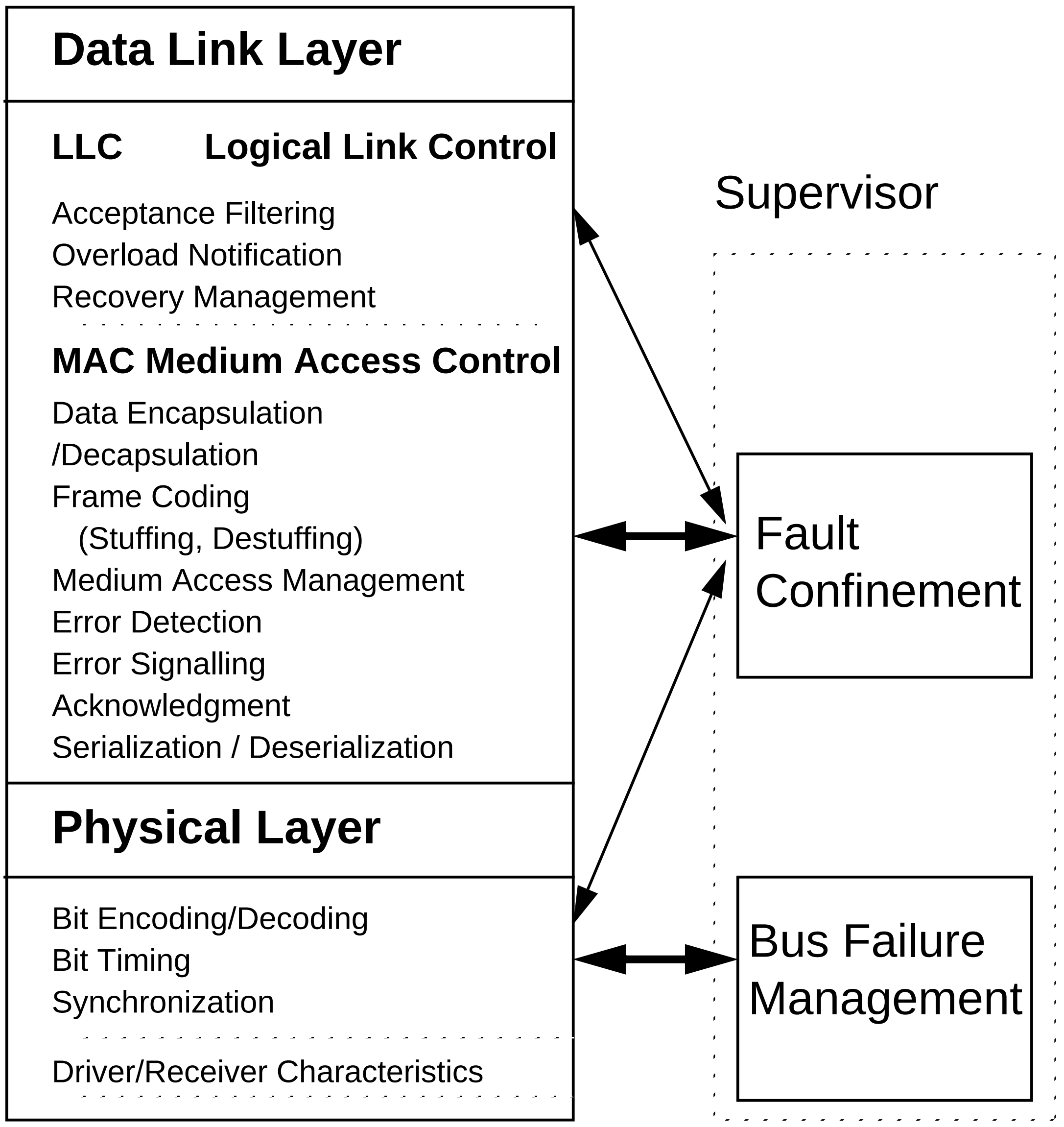
9 CAN FD Implementation 32

1 INTRODUCTION

CAN FD is a serial communications protocol which efficiently supports distributed real-time control with a very high level of security.

The intention of this specification is to achieve compatibility between any two CAN FD 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 FD has been subdivided into different layers according to the ISO/OSI Reference Model.

Layered Architecture of CAN FD according to the OSI Reference Model



The scope of this specification is to define the MAC sublayer and a small part of the LLC sublayer of the Data Link Layer as well a part of the Physical Layer and to describe the consequences of the CAN protocol on the surrounding layers.

Data Link Layer
The Data Link Layer handles frames and consists of the two sublayers:

- Logical Link Control (LLC)
- Medium Access Control (MAC)

LLC sublayer of the Data Link Layer
The LLC corresponds to the node's controller-host interface and is concerned with Message Filtering, Overload Notification and Recovery Management. Its scope is

- to decide which messages received by the MAC sublayer are actually to be accepted,
- to provide services for data transfer and for remote data request,
- to provide messages to the MAC sublayer for transmission,
- to provide means for recovery management and overload notifications.

There is much freedom in defining object handling.

MAC sublayer of the Data Link Layer
The MAC sublayer is responsible for Message Framing, Arbitration, Acknowledgment, Error Detection and Signalling. It is supervised by a management entity called Fault Confinement which is a self-checking mechanism for distinguishing short disturbances from permanent failures. Within the MAC sublayer it is decided whether the bus is free for starting a new transmission or whether a reception is just starting. The MAC sublayer represents the kernel of the CAN FD protocol. It is in the nature of the MAC sublayer that there is no freedom for modifications.

Physical Layer
The Physical Layer handles bits and defines how signals are actually transmitted and therefore deals with the description of Bit Timing, Bit Encoding, and Synchronization. Within this specification the electrical driver/receiver characteristics of the Physical Layer are not defined so as to allow transmission medium and signal level implementations to be optimized for their application.

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.

2 Basic Concepts

CAN FD 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 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
- compatibility with CAN protocol, every CAN FD node is able to receive and to transmit CAN messages according to ISO 11898-1.

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

Bit rate
There may be two bit rates in a CAN FD system, one for the ARBITRATION-PHASE and one for the DATA-PHASE. The speed of CAN FD may be different in different systems. However, in a given system the two bit rates are uniform and fixed.

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

<u>Remote Data Request</u>
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. There is no REMOTE FRAME in the CAN FD format. Each CAN FD node however is able to transmit a REMOTE FRAME in the standard CAN format.

<u>Multimaster</u>
When the bus is free any unit may start to transmit a message. The unit with the started message of higher priority gains bus access.

<u>Arbitration</u>
Whenever the bus is free, any unit may start to transmit a message. If two 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.

<u>Safety</u>
In order to achieve the utmost safety of data transfer, powerful measures for error detection, signalling and self-checking are implemented in every CAN FD 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 CRC Sequence 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

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

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 restart of the disturbed message is at most 31 bit times, if there is no further error.

Fault Confinement
CAN FD nodes are able to distinguish short disturbances from permanent failures. Defective nodes are switched off.

Connections
The CAN FD 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 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* levels, 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.

Acknowledgment
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 FD device may be set into sleep mode without any internal activity and with disconnected bus drivers. The sleep mode is finished with a wake-up by 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.

3 MESSAGE TRANSFER

3.1 FRAME FORMATS

There are four different formats which differ in the length of the ARBITRATION FIELD and in the CONTROL FIELD:

CAN BASE FORMAT:	11 bit long identifier and constant bit rate
CAN EXTENDED FORMAT:	29 bit long identifier and constant bit rate
CAN FD BASE FORMAT:	11 bit long identifier and dual bit rate
CAN FD EXTENDED FORMAT:	29 bit long identifier and dual bit rate

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*. There are four subtypes of DATA FRAME in CAN FD:

DATA FRAME in CAN BASE FORMAT

DATA FRAME in CAN EXTENDED FORMAT

DATA FRAME in CAN FD BASE FORMAT

DATA FRAME in CAN FD EXTENDED FORMAT

A REMOTE FRAME is transmitted by a bus unit to request the transmission of the DATA FRAME with the same IDENTIFIER format. A CAN FD mode shall support two subtypes of REMOTE FRAME:

REMOTE FRAME in CAN BASE FORMAT

REMOTE FRAME in CAN EXTENDED FORMAT

There are no REMOTE FRAMES in CAN FD format.

An ERROR FRAME is transmitted by any unit on detecting a bus error.

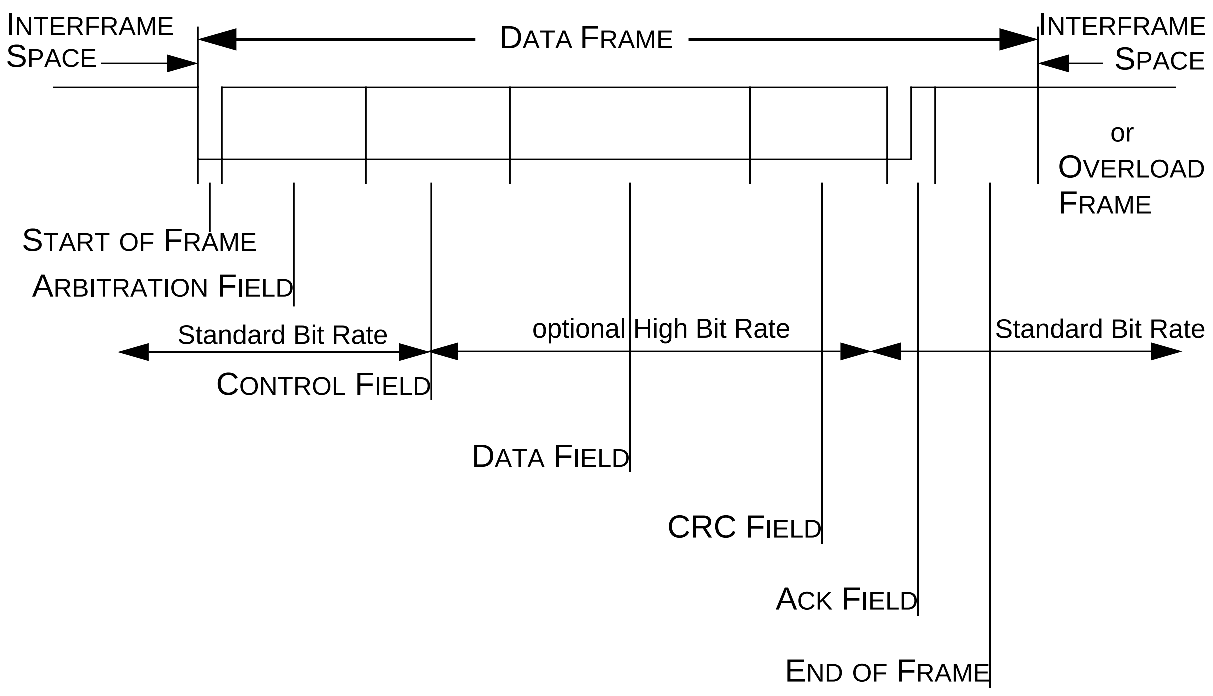
An OVERLOAD FRAME is used to synchronize idle detection and to provide for an extra delay between the preceding and the succeeding DATA or REMOTE FRAMES.

DATA FRAMES and REMOTE FRAMES are separated from preceding frames by an INTER-FRAME SPACE.

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 may be of length zero.



START OF FRAME

The START OF FRAME (SOF) marks the beginning of DATA FRAMES and REMOTE FRAMES. It consists of a single *dominant* bit.

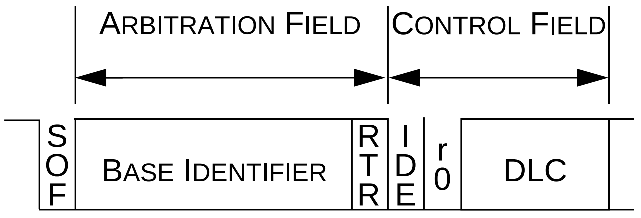
A station is only allowed to start transmission when the bus is idle (see INTERFRAME SPACE). All stations have to synchronize to the leading edge caused by START OF FRAME (see HARD SYNCHRONIZATION) of the station starting transmission first.

ARBITRATION FIELD

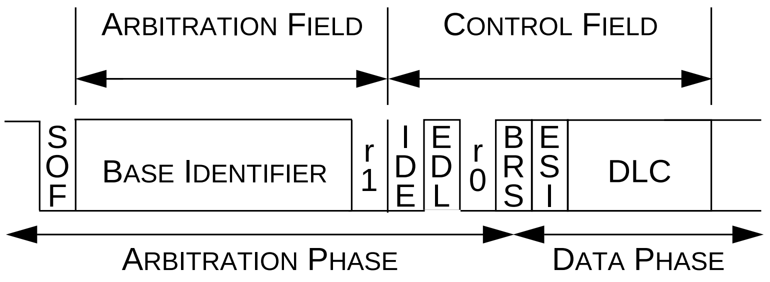
The structure of the ARBITRATION FIELD is different for BASE FORMAT and EXTENDED FORMAT frames, but there is only one different bit in CAN format and CAN FD format.

- In BASE FORMAT the ARBITRATION FIELD consists of the BASE IDENTIFIER and the RTR bit (CAN format) or the r1 bit (CAN FD format). The BASE IDENTIFIER is 11 bits long, denoted ID-28…ID-18.
- In EXTENDED FORMAT the ARBITRATION FIELD consists of the EXTENDED IDENTIFIER, the SRR bit, the IDE bit, and the RTR bit (CAN format) or the r1 bit (CAN FD format). The EXTENDED IDENTIFIER consists of two sections, the first section is the BASE IDENTIFIER (denoted ID-28…ID-18), the second section is the IDENTIFIER EXTENSION (18 bits long, denoted ID-17…ID-0).

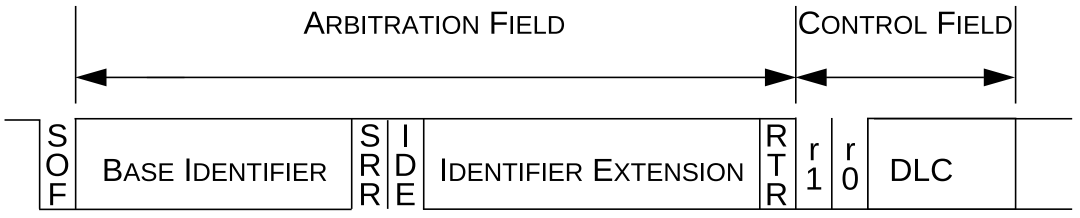
CAN Base Format



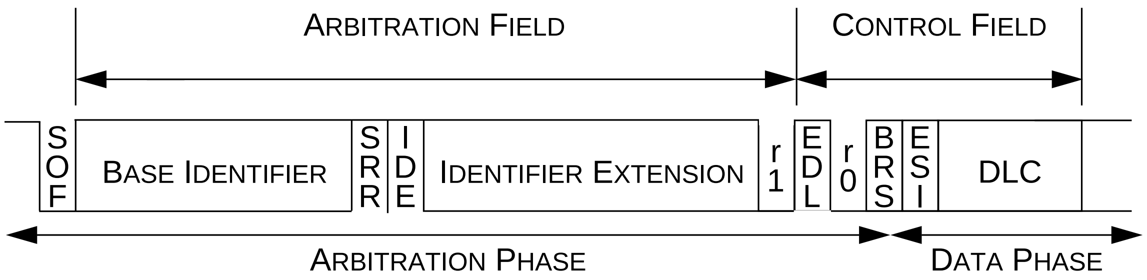
CAN FD Base Format



CAN Extended Format



CAN FD Extended Format



IDE
The Identifier Extension Flag (IDE) belongs to

- the Arbitration Field for the Extended Format
- the Control Field for the Base Format,

it distinguishes between the formats. The IDE bit is transmitted *dominant* in the Base Format, whereas in the Extended Format the IDE bit is *recessive*.

SRR
The SUBSTITUTE REMOTE REQUEST (SRR) bit is *recessive*. It is transmitted in EXTENDED FORMAT at the position of the RTR bit in BASE FORMAT. Therefore, collisions of a frame in BASE FORMAT and a frame in EXTENDED FORMAT, the BASE IDENTIFIER of which is the same in both frames, are resolved in such a way that the frame in BASE FORMAT prevails the frame in EXTENDED FORMAT.

RTR
The REMOTE TRANSMISSION REQUEST (RTR) bit only exists in CAN format frames. It has to be *dominant* within DATA FRAMES and has to be *recessive* within REMOTE FRAMES. In CAN FD format frames, it is replaced with the *dominant* reserved bit r1. There are no REMOTE FRAMES in CAN FD format.

CONTROL FIELD
The structure of the CONTROL FIELD is different for CAN BASE FORMAT, CAN FD BASE FORMAT, and CAN EXTENDED FORMAT, and CAN FD EXTENDED FORMAT frames.

- In CAN BASE FORMAT the CONTROL FIELD consists of the bits
  IDE, r0, and the 4 bits wide DATA LENGTH CODE (DLC).
- In CAN FD BASE FORMAT the CONTROL FIELD consists of the bits
  IDE, EDL,r0, BRS, ESI, and the 4 bits wide DATA LENGTH CODE (DLC).
- In CAN EXTENDED FORMAT the CONTROL FIELD consists of the bits
  r1, r0, and the 4 bits wide DATA LENGTH CODE (DLC).
- In CAN FD EXTENDED FORMAT the CONTROL FIELD consists of the bits
  EDL, r0, BRS, ESI, and the 4 bits wide DATA LENGTH CODE (DLC).

EDL
The EXTENDED DATA LENGTH (EDL) bit is *recessive*. It only exists in CAN FD format frames, it distinguishes between CAN format and CAN FD format frames. In a CAN format frame, the *dominant* bit r0 is transmitted instead of EDL. In frames with 11-bit identifiers, EDL comes after the IDE bit, in frames with 29-bit-identifier, it comes after the r1 bit. EDL is always followed by the *dominant* bit r0, which is reserved for future expansion of the protocol.

BRS
The BIT RATE SWITCH (BRS) bit decides whether the bit rate is switched inside a CAN FD format frame. If the bit is transmitted *recessive*, the bit rate is switched from the standard bit rate of the ARBITRATION PHASE to the preconfigured alternate bit rate of the DATA PHASE. If it is transmitted *dominant*, the bit rate is not switched. BRS does not exist in CAN format frames.

ESI
The ERROR STATE INDICATOR (ESI) flag is transmitted *dominant* by *error active* nodes, *recessive* by *error passive* nodes. ESI does not exist in CAN format frames.

The reserved bits r1 and r0 have to be sent *dominant*. *Receivers* accept *dominant* and *recessive* bits in all combinations. *Receivers* also accept *dominant* SRR bits.

DATA LENGTH CODE

The number of bytes in the DATA FIELD is indicated by the DLC, its coding is different in CAN and in CAN FD. The first nine codes are the same, but the following codes, that in CAN all specify a DATA FIELD of eight bytes, specify longer DATA FIELDS in CAN FD.

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

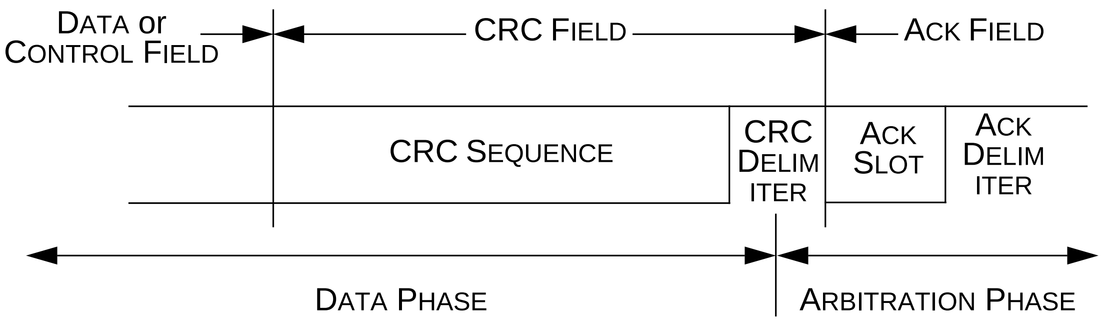
	Number of Data Bytes	Data Length Code			
		DLC3	DLC2	DLC1	DLC0
Codes in CAN and CAN FD Format	0	0	0	0	0
	1	0	0	0	1
	2	0	0	1	0
	3	0	0	1	1
	4	0	1	0	0
	5	0	1	0	1
	6	0	1	1	0
	7	0	1	1	1
CAN Format	8	1	0/1	0/1	0/1
Codes in CAN FD Format	8	1	0	0	0
	12	1	0	0	1
	16	1	0	1	0
	20	1	0	1	1
	24	1	1	0	0
	32	1	1	0	1
	48	1	1	1	0
	64	1	1	1	1

DATA FIELD

The DATA FIELD consists of the data to be transferred within a DATA FRAME. It can contain from 0 to 8 bytes in CAN Format and 0 to 64 bytes in CAN FD format. The bytes each contain 8 bits which are transferred MSB first. When the DLC is zero, or in REMOTE FRAMES, there is no DATA FIELD.

CRC FIELD

The CRC FIELD contains the CRC SEQUENCE followed by the *recessive* CRC DELIMITER bit.



CRC SEQUENCE
The frame check sequence is derived from a cyclic redundancy code (BCH Code).

A CAN FD node uses different CRC generator-polynomials for different frame formats. The first polynomial, CRC_15, is used for all frames in CAN format. The second, CRC_17, is used for frames in CAN FD format with a DATA FIELD up to sixteen byte long. The third, CRC_21, is used for frames in CAN FD format with a DATA FIELD longer than sixteen byte. Each polynomial results in a Hamming Distance of HD = 6.

- CRC_15 0xC599 $(x^{15}+x^{14}+x^{10}+x^8+x^7+x^4+x^3+1)$
  = $(x+1) \cdot (x^7+x^3+1) \cdot (x^7+x^3+x^2+x+1)$
- CRC_17 0x3685B $(x^{17}+x^{16}+x^{14}+x^{13}+x^{11}+x^6+x^4+x^3+x^1+1)$
  = $(x+1) \cdot (x^{16}+x^{13}+x^{10}+x^9+x^8+x^7+x^6+x^3+1)$
- CRC_21 0x302899 $(x^{21}+x^{20}+x^{13}+x^{11}+x^7+x^4+x^3+1)$
  = $(x+1) \cdot (x^{10}+x^3+1) \cdot (x^{10}+x^3+x^2+x^1+1)$

The length of the CRC SEQUENCE (n_{CRC}, the order of the generator-polynomial) is set to 15 for CRC_15, to 17 for CRC_17, and to 21 for CRC_21.

At the start of the frame, all three CRC SEQUENCES shall be calculated concurrently; in all nodes including the *Transmitter*. The node that wins the arbitration sends the CRC SEQUENCE selected by the values of the frame's EDL bit and DLC. The *Receivers* shall consider only the selected CRC polynomial to check for a CRC-ERROR.

The relevant bit stream for CRC calculation is the bit stream consisting of START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, and (if present) DATA FIELD, supplemented with n_{CRC} bits of '0'. In CAN FD format frames, STUFF-BITS are included in the relevant bit stream for CRC calculation, in CAN format frames, STUFF-BITS are not included.

In order to carry out the CRC calculation, the polynomial to be divided is defined by the coefficients of the relevant bit stream. This polynomial is divided (the coefficients are calculated modulo-2) by the generator-polynomial.

The remainder of this polynomial division is the CRC SEQUENCE transmitted over the bus. In order to implement this function, a n_{CRC} bit shift register CRC_RG(n_{CRC}-1:0) can be used. Each CRC SEQUENCE is calculated in a separate shift register block. If NXTBIT denotes the next bit of the bit stream, given by the relevant bit stream from

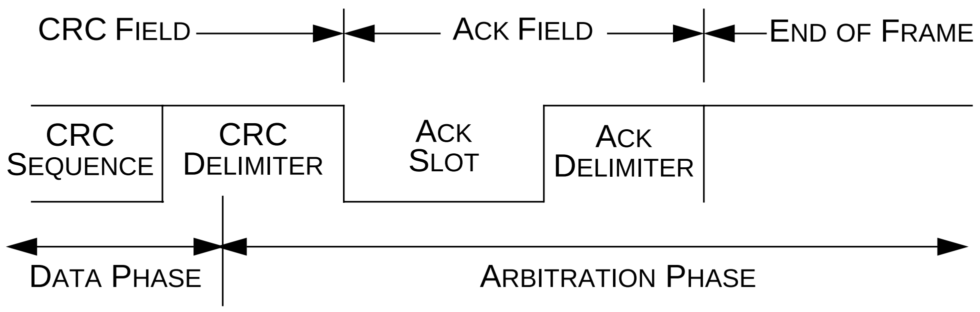
START OF FRAME until the end of the DATA FIELD, the CRC SEQUENCES are calculated as follows:

```
CRC_RG = 0;                                  // initialize shift register
REPEAT
 CRCNXT = NXTBIT EXOR CRC_RG(nCRC-1);
 CRC_RG(nCRC-1:1) = CRC_RG(nCRC-2:0);        // shift left by 1 position
 CRC_RG(0) = 0;
 IF CRCNXT THEN
  CRC_RG(nCRC-1:0) = CRC_RG(nCRC-1:0) EXOR (CRC polynomial);
 ENDIF
UNTIL (CRC SEQUENCE starts or there is an ERROR condition)
```

After the transmission / reception of the last bit of the relevant bit stream, each CRC_RG contains one of the three CRC SEQUENCES.

CRC DELIMITER
The CRC SEQUENCE is followed by the CRC DELIMITER. In CAN format, the CRC DELIMITER is one single *recessive* bit. In CAN FD format, the CRC DELIMITER may consist of one or two *recessive* bits. A *Transmitter* shall send only one *recessive* bit as CRC DELIMITER, but it shall accept two *recessive* bits before the edge from *recessive* to *dominant* that starts the ACKNOWLEDGE SLOT. A *Receiver* will send its ACKNOWLEDGE bit after the first CRC DELIMITER bit.

Note: CAN FD protocol controllers switch back from the DATA-PHASE to the ARBITRATION PHASE when they reach the SAMPLE POINT of the (first bit of the) CRC DELIMITER.



ACK FIELD
The ACK FIELD contains the ACK SLOT and the ACK DELIMITER. In the ACK FIELD, the transmitting station sends *recessive* bits.

The phase-shift between the nodes in a CAN network is defined by the delay times in the transceivers and the propagation time on the CAN bus line. The phase-shift is the same in CAN and in CAN FD, but it is proportionally larger in the phase with the shorter bit time. All *Receivers* in the network may have a different phase-shift to the *Transmitter*, depending on their distances from the *Transmitter*, since they see the transmitted edges at different times. To compensate for these phase-shifts when the bit rate is switched back from the shorter to the longer bit time, one additional bit time tolerance is allowed before and after the edge from *recessive* to *dominant* that starts the ACKNOWLEDGE SLOT.

A *Receiver* which has received a valid message correctly, reports this to the *Transmitter* by sending one *dominant* bit at the start of the ACK SLOT.

ACK SLOT
All stations having received the matching CRC SEQUENCE report this within the ACK SLOT by superscribing the *recessive* bit of the *Transmitter* with one *dominant* bit (they send ACK). In CAN FD format, all nodes shall accept a two bit long *dominant* phase of overlapping ACK bits as a valid ACK, to compensate for phase shifts between the *Receivers*. In CAN format, a *dominant* bit following the single ACK SLOT bit is a FORM-ERROR.

ACK DELIMITER
The *recessive* ACK DELIMITER is the last bit of the ACK FIELD. 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.

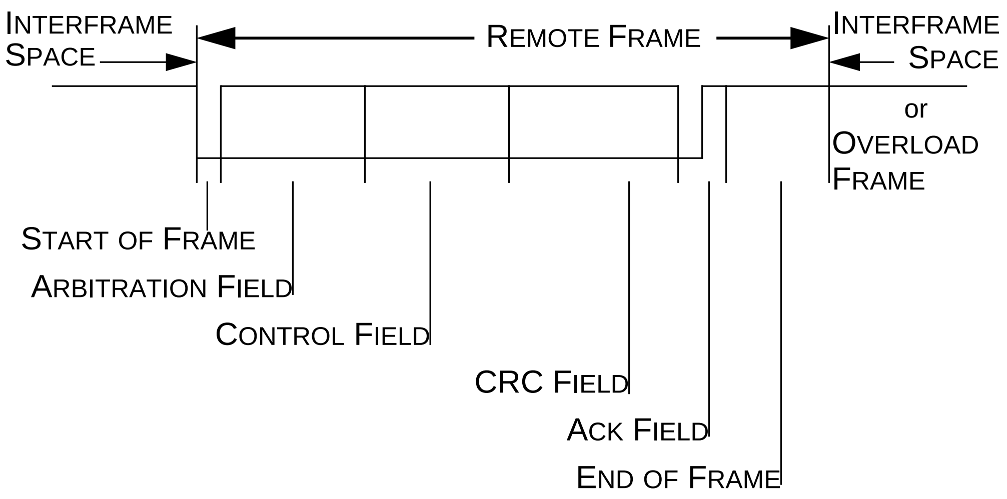
3.2.2 REMOTE FRAME

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:

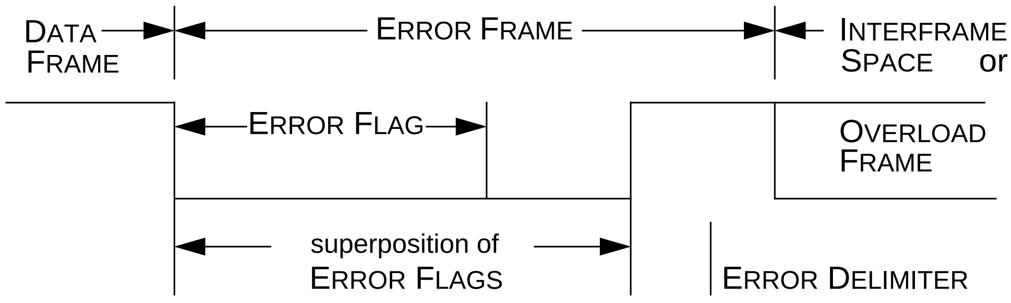
START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, CRC FIELD, ACK FIELD, END OF FRAME.

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 signed any value within the admissible range 0...15. The value shall be the DATA LENGTH CODE of the corresponding DATA FRAME. The REMOTE FRAME is only defined in the CAN format, neither REMOTE FRAME nor RTR bit exist in the CAN FD format.



3.2.3 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 two 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 section 5: 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.

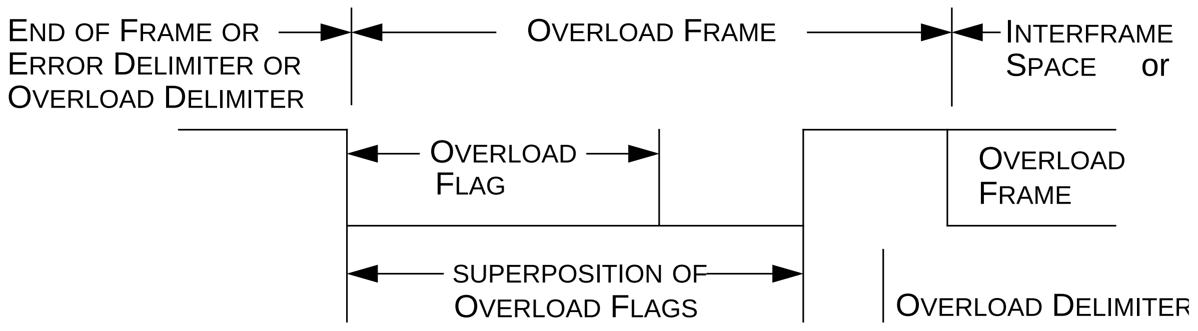
3.2.4 OVERLOAD FRAME

The OVERLOAD FRAME contains the two bit fields OVERLOAD FLAG and OVERLOAD DELIMITER.

There are three kinds of OVERLOAD conditions, which 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 at the first or second bit of INTERMISSION.
3. If a CAN FD node samples a dominant bit at the eighth bit (the last bit) of an ERROR DELIMITER or OVERLOAD DELIMITER, or if a CAN FD *Receiver* samples a dominant bit at the last bit of END OF FRAME, it will start transmitting an OVERLOAD FRAME (not an ERROR FRAME). The Error Counters will not be incremented.

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 or condition 3 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.

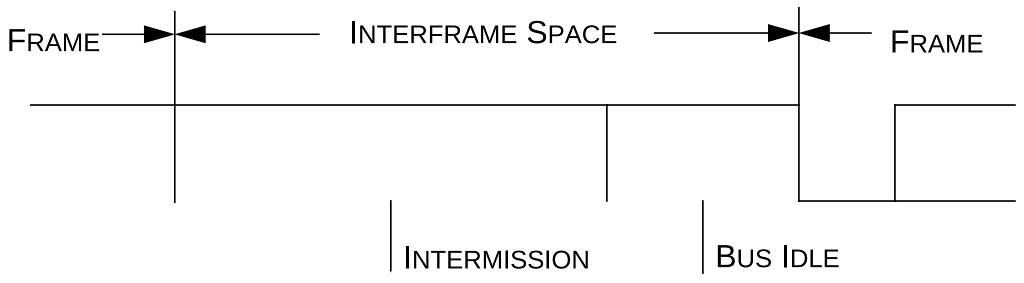
The OVERLOAD DELIMITER is of the same form as the ERROR DELIMITER. After transmission of an OVERLOAD FLAG the station monitors the bus until it detects a transition from a *dominant* to a *recessive* bit. At this point of time every bus station has finished sending its OVERLOAD FLAG and all stations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

3.2.5 INTERFRAME SPACE

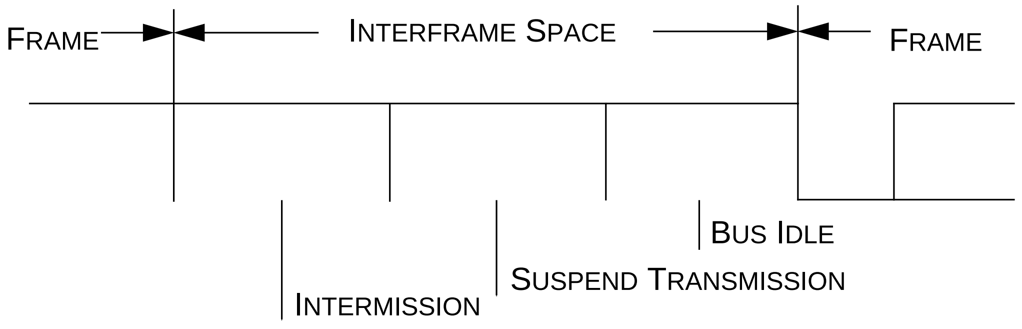
DATA FRAMES and REMOTE FRAMES are separated from preceding frames whatever type they are (DATA FRAME, REMOTE FRAME, ERROR FRAME, OVERLOAD FRAME) by a bit field called INTERFRAME SPACE. In contrast, OVERLOAD FRAMES and ERROR FRAMES are not preceded by an INTERFRAME SPACE and multiple OVERLOAD FRAMES are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

The detection of a *dominant* bit on the bus at the third bit of INTERMISSION shall be interpreted as START OF FRAME.

Note: A CAN FD node with a pending transmission that is *error active* or has been *Receiver* of the previous frame shall, if it samples a *dominant* bit at the third bit of INTERMISSION, start transmitting its message with the first bit of its BASE IDENTIFIER at the next bit, without first transmitting a START OF FRAME bit and without becoming *Receiver*.

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 a 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 BUS IDLE. If meanwhile a transmission (caused by another station) starts, the station will become *Receiver* of this message.

3.2.6 DATA CONSISTENCY

Messages to be transmitted are prepared by the host and are transferred via the node's controller-host interface and LLC sublayer of the Data Link Layer to the MAC sublayer that is responsible for Message Framing. Messages may be stored in a shared memory Data consistency of transmitted messages from a shared memory shall be ensured by at least one of two methods:

- The MAC sublayer shall store the whole message to be transmitted in a temporary buffer that is filled before the transmission is started.
- The LLC sublayer shall check for data errors while the message to be transmitted is transferred to the MAC sublayer. If a data error is detected, the transmission shall not be started. If it is already started when the data error is detected, the node shall be switched into Bus Monitoring Mode, see section 3.3.1. Receiving nodes will not see a valid message.

Note: Data errors are e.g. parity errors in a RAM word, data not provided in time, or data partially updated during a transmission.

3.3 OPERATION MODES

A CAN FD unit is in one of four operation state , *Integrating*, *Idle*, *Receiver*, or *Transmitter*. They are defined as follows:

Integrating
A unit is *Integrating* while it waits to detect eleven consecutive *recessive* bits after the start of the controller or during *bus_off* recovery, then switching to *Idle*.

Idle
A unit is *Idle* if it is ready for a START OF FRAME, switching to either *Receiver* or *Transmitter*.

Receiver
A unit operates as *Receiver* if it detects activity on the CAN bus and if it is not *Transmitter*.

Transmitter
A unit originating a message is operating as *Transmitter*. The unit stays *Transmitter* until the bus is idle or the unit loses ARBITRATION.

3.3.1 BUS MONITORING MODE

In an optional Bus Monitoring Mode, the CAN FD node shall be able to receive valid DATA FRAMES and valid REMOTE FRAMES, but it sends only *recessive* bits on the CAN bus and cannot start a transmission. If the CAN FD protocol controller is required to send a *dominant* bit (ACK SLOT, OVERLOAD FLAG, ACTIVE ERROR FLAG), the bit is rerouted internally so that the CAN FD protocol controller monitors this *dominant* bit, although the CAN bus may remain in *recessive* state.

3.3.2 RESTRICTED OPERATION MODE

In an optional Restricted Operation Mode, a CAN FD node is able to transmit and to receive DATA FRAMES and REMOTE FRAMES and it gives ACKNOWLEDGE to valid frames, but it does not send ACTIVE ERROR FRAMES or OVERLOAD FRAMES. In case of an error condition or overload condition, it does not send *dominant* bits, instead it waits for the occurrence of BUS IDLE condition to resynchronize itself to the CAN communication. The error counters are not incremented.

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. The number of retransmission attempts may be limited (by configuration) to a specific value. By default, the number of retransmissions is not limited

Receivers:
The message is valid for the *Receivers*, if there is no error until the last but one bit of END OF FRAME. The value of the last bit of END OF FRAME is treated as ’don’t care’, a *dominant* value does not lead to a FORM-ERROR see section 6.1: Error Detection). A *Receiver* that detects a *dominant* bit at the last bit of END OF FRAME responds with an OVERLOAD FRAME.

4.1 MESSAGE FILTERING

Message filtering is based upon the whole Identifier, it decides whether a received message is discarded or is stored inside the node. 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.

5 CODING

BIT STREAM CODING

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*.

In order to limit the maximum distance between edges available for synchronization, 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 (called STUFF-BIT) into the actual transmitted bit stream.The *Receiver* shall recognize a sequence of five consecutive bits of identical value and discard the following STUFF-BIT.

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.

In CAN FD format frames, the CAN bit stuffing method is changed for the CRC SEQUENCE. Here, the STUFF-BITS shall be inserted at fixed positions. There shall be a fixed STUFF-BIT before the first bit of the CRC SEQUENCE, even if the last bits of the preceding field do not fulfill the CAN stuff condition. A further STUFF-BIT shall be inserted after each fourth bit of the CRC SEQUENCE. The value of such a fixed STUFF-BIT shall be the inverse value of the bit preceding the fixed STUFF-BIT. A *Receiver* shall discard the fixed STUFF-BITS from the bit stream for the CRC check, it shall detect a STUFF-ERROR if the fixed STUFF-BIT has the same value as its preceding bit. The number of fixed STUFF-BITS in the CAN FD format CRC SEQUENCE is equal to the maximum number of stuff bits that would result from applying the CAN Format stuffing method. The fixed STUFF-BITS in the CRC SEQUENCE shall be discarded by the *Receivers*.

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 interpret this as a BIT-ERROR.

STUFF-ERROR
A STUFF-ERROR shall 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 shall be detected if the calculated result is not the same as that received in the CRC SEQUENCE.

FORM-ERROR
A FORM-ERROR shall be detected when a fixed-form bit field contains one or more illegal bits. (Exception is the detection of a *dominant* bit during the last bit of END OF FRAME by a RECEIVER, or the detection of a *dominant* bit during the last bit of ERROR DELIMITER or OVERLOAD DELIMITER by any node). When the value of a fixed STUFF-BIT in the CAN FD format CRC SEQUENCE is equal to its preceding bit, this shall also be detected as a FORM-ERROR.

ACKNOWLEDGMENT-ERROR
An ACKNOWLEDGMENT-ERROR has to be detected by a *Transmitter* whenever it does not monitor a *dominant* bit during the 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 PASSIVE ERROR FLAG. Whenever a BIT-ERROR, a STUFF-ERROR, a FORM-ERROR or an ACKNOWLEDGMENT-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 condition has already been started.

A CAN FD node operating in the DATA-PHASE shall switch back to the ARBITRATION-PHASE when starting an ERROR FLAG.

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 ACKNOWLEDGMENT-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, 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.

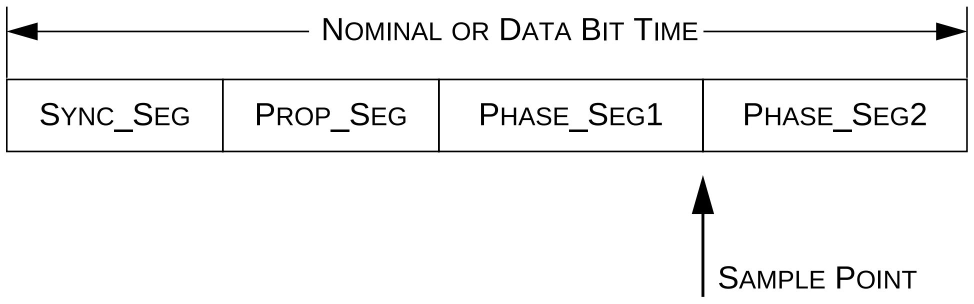
6. Any node tolerates up to 7 consecutive *dominant* bits after sending an ACTIVE ERROR FLAG, PASSIVE ERROR FLAG or OVERLOAD FLAG. After detecting the 14th consecutive *dominant* bit (in case of an ACTIVE ERROR FLAG or an OVERLOAD FLAG) or after detecting the 8th consecutive *dominant* bit following 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 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. 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 occurrence 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 start-up only 1 node is online, and if this node transmits some message, it will get no ACKNOWLEDGE, detect an error and repeat the message. It can become *error passive* but not *bus_off* due to this reason.

8 BIT TIMING REQUIREMENTS

The CAN FD protocol defines two bit rates, the first for the ARBITRATION-PHASE with a longer bit time and the second for the data phase with the same or with a shorter bit time. The definition for the first bit rate is the same as for the NOMINAL BIT RATE and the NOMINAL BIT TIME in the CAN protocol specification. The definition for the second bit rate, the DATA BIT RATE WITH the DATA BIT TIME, requires a separate configuration register set. Both bit times consist of separate non-overlapping time segments, these segments form the bit time as shown in this figure:



SYNCHRONIZATION SEGMENT (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.

PROPAGATION TIME 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 BUFFER SEGMENT1 (PHASE_SEG1)
PHASE BUFFER SEGMENT2 (PHASE_SEG2)
These PHASE-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.

The time segments for the two bit rates of the CAN FD protocol are defined by two sets of configuration registers.

INFORMATION PROCESSING TIME
The INFORMATION PROCESSING TIME is the time segment starting with the SAMPLE POINT reserved for calculation the subsequent bit level, its length is determined by the CAN controller implementation.

The length of the time segments is defined in integer multiples of the TIME QUANTUM, with 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

TIME QUANTUM(n) = m(n) * MINIMUM TIME QUANTUM

with m(n) the value of the prescaler. Two values for the prescaler, m(N) for the NOMINAL BIT TIME and m(D) for the DATA BIT TIME, are defined for the CAN FD protocol, one for each bit rate, resulting in two different lengths of the TIME QUANTUM.

The number of TIME QUANTA in a bit time shall be programmable at least from 8 to 25.

Length of Time Segments for the NOMINAL BIT RATE

- SYNC_SEG(N) is 1 TIME QUANTUM(N) long.
- PROP_SEG(N) is programmable to be 1,2,…,32 or more TIME QUANTA(N) long.
- PHASE_SEG1(N) is programmable to be 1,2,…,32 or more TIME QUANTA(N) long.
- PHASE_SEG2(N) is the maximum of PHASE_SEG1(N) and the INFORMATION PROCESSING TIME
- The INFORMATION PROCESSING TIME is less than or equal to 2 TIME QUANTA(N) long.

The first part of a CAN FD frame, until the BRS bit, is transmitted with the NOMINAL BIT RATE. The bit rate is switched if the BRS bit is *recessive*, until the CRC DELIMITER is reached or until the CAN FD controller sees an error condition that results in the starting of an ERROR FRAME. CAN FD ERROR FRAMES, as well as ACK FIELD, END OF FRAME, OVERLOAD FRAMES, and all frames in CAN format are transmitted with the NOMINAL BIT RATE.

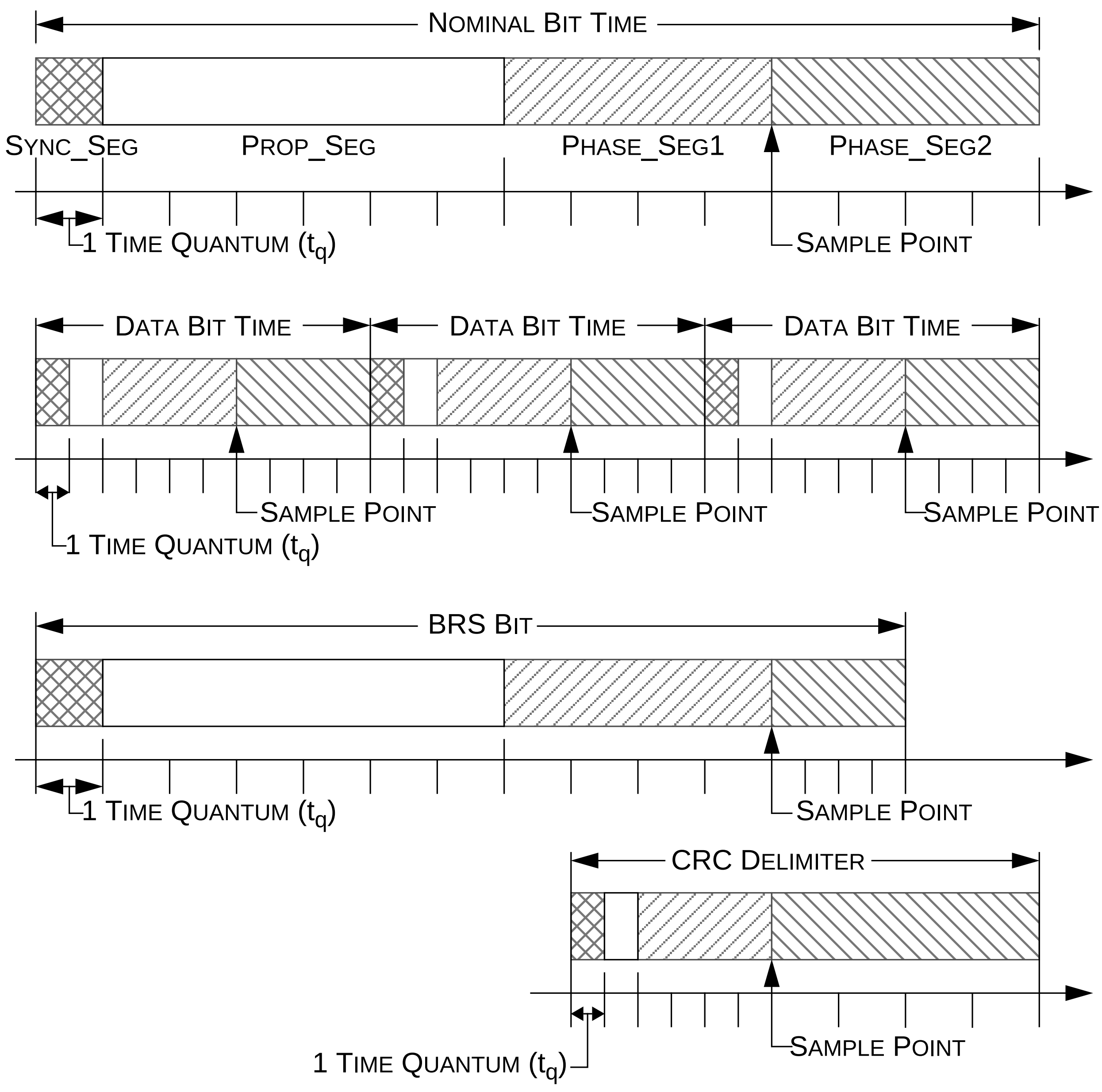
Length of Time Segments for the DATA BIT RATE

- SYNC_SEG(D) is 1 TIME QUANTUM(D) long.
- PROP_SEG(D) is programmable to be 0,1,2,…,8 TIME QUANTA(D) long.
- PHASE_SEG1(D) is programmable to be 1,2,…,8 TIME QUANTA(D) long.
- PHASE_SEG2(D) is the maximum of PHASE_SEG1(D) and the INFORMATION PROCESSING TIME
- The INFORMATION PROCESSING TIME is less than or equal to 2 TIME QUANTA(D) long.

The position of the SAMPLE POINT may differ in the two bit timing configurations, the length of the PROP_SEG may be reduced in the configuration for the DATA BIT RATE. When the bit rate is switched at the BRS bit or at the CRC DELIMITER bit, it shall be switched immediately after the SAMPLE POINT, causing the length of these two bits to be intermediate. The sum of the length of these two bits shall be the same as the sum of one bit of the NOMINAL BIT TIME and one bit of the DATA BIT TIME. When the bit rate is switched because an error condition is detected, the switching time may be shifted after the SAMPLE POINT, by the length of the INFORMATION PROCESSING TIME.

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.

An example for the Nominal Bit Time configuration based on a prescaler of m(N) = 2 and the time segments Prop_Seg(N) = 6, Phase_Seg1(N) = 4, Phase_Seg2(N) = 4 combined with the Data Bit Time configuration based on a prescaler of m(D) = 1 and the time segments Prop_Seg(D) = 1, Phase_Seg1(D) = 4, Phase_Seg2(D) = 4, as well as the resulting bits of intermediate length BRS and CRC Delimiter is shown in the following figure:



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.

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(N) shall be programmable between 1 and min(16, Phase_Seg1(N)), the Resynchronization Jump Width(D) shall be programmable between 1 and min(4, Phase_Seg1(D)).

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 the Sync_Seg.
- e > 0 if the edge lies between the Sync_Seg and the Sample Point.
- e < 0 if the edge lies between the Sample Point and the following bit's Sync_Seg.

Resynchronization
The effect of a Resynchronization is the same as that of a 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.
- 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 between two Sample Points is allowed.
2. An edge shall 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, Suspend Transmission, and second or third bits of Intermission. Hard Synchronization is also performed at the *recessive* to *dominant* edge from EDL to r0 in CAN FD format frames.
4. All other *recessive* to *dominant* edges fulfilling the rules 1 and 2 shall be used for Resynchronization with the exception that a node transmitting a *dominant* bit shall not perform a Resynchronization as a result of a *recessive* to *dominant* edge with a positive Phase Error.
5. A *Transmitter* shall not resynchronize while it transmits in the CAN FD Data-Phase.

Oscillator Tolerance
The tolerance range df for an oscillator's frequency f_{osc} around the nominal frequency f_{nom} with $(1 - df) \bullet f_{nom} \le f_{osc} \le (1 + df) \bullet f_{nom}$ depends on the proportions of PHASE_SEG1, PHASE_SEG2, SJW, and the bit time. The maximum tolerance df is the defined by five conditions. Conditions I and II shall be met in CAN operation, all five conditions shall be met in CAN FD operation:

$$\text{I: } df < \frac{SJW_N}{20 \cdot \text{bit time}_N}$$

$$\text{II: } df < \frac{\min(Phase\_Seg1_N, Phase\_Seg2_N)}{2 \cdot (13 \cdot \text{bit time}_N - Phase\_Seg2_N)}$$

$$\text{III: } df < \frac{SJW_D}{20 \cdot \text{bit time}_D}$$

$$\text{IV: } df < \frac{\min(Phase\_Seg1_D, Phase\_Seg2_D)}{2 \cdot \left([6 \cdot \text{bit time}_D - Phase\_Seg2_D] \cdot \frac{m_D}{m_N} + 7 \cdot \text{bit time}_N\right)}$$

$$\text{V: } df < \frac{SJW_D - \left(\frac{m_N}{m_D} - 1\right)}{2 \cdot \left([2 \cdot \text{bit time}_N - Phase\_Seg2_N] \cdot \frac{m_N}{m_D} + Phase\_Seg2_D + 4 \cdot \text{bit time}_D\right)}$$

It has to be considered that SJW may not be larger than the smaller of the PHASE BUFFER SEGMENTS and that the PROPAGATION TIME SEGMENT limits that part of the bit time that may be used for the PHASE BUFFER SEGMENTS.

The combination PROP_SEG(N)= 1 and PHASE_SEG1(N)= PHASE_SEG2(N)= SJW(N)= 4 allows the largest possible oscillator tolerance of 1.58% in CAN operation. This combination with a PROPAGATION TIME SEGMENT of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 kbit/s (bit time = 8 µs) with a bus length of 40 m. This NOMINAL BIT TIME may be combined with a shorter DATA BIT TIME.

8.1 TRANSCEIVER DELAY COMPENSATION

Without a transceiver delay compensation, the bit rate in the DATA-PHASE of a CAN FD frame is limited by the fact that the *Transmitter* detects a BIT-ERROR if it cannot receive its own transmitted bit latest at the SAMPLE POINT of that bit.

CAN FD nodes shall support an optional transceiver delay compensation mechanism, to be used in applications where the length of the CAN bit time in the DATA-PHASE is shorter than the limit required by the transceiver's internal delay time.

The transceiver delay compensation mechanism defines a SECONDARY SAMPLE POINT where the transmitted bit is compared with the received bit in order to check for BIT-ERRORS. This mechanism shall only be used by *Transmitters* in the DATA-PHASE of a CAN FD frame. When it is used, the *Transmitter* shall ignore the bit value received at the SAMPLE POINT. The delay to be compensated, TRV_DELAY, shall be measured in each transmitted frame at the edge from the EDL bit to the following reserved bit r0, between the edge of the transmitted bit and the edge of the received bit.

The position of the SECONDARY SAMPLE POINT shall be TRV_DELAY plus an offset (e.g. half of the bit time in the DATA-PHASE), rounded down to the next integer number of TIME QUANTA. The resulting SECONDARY SAMPLE POINT may be placed after the end of the transmitted bit. If a BIT-ERROR is detected at the SECONDARY SAMPLE POINT, the *Transmitter* will react to this BIT-ERROR at the next following SAMPLE POINT.

9 CAN FD IMPLEMENTATION

CAN FD protocol implementations shall provide the same controller-host interfaces as CAN protocol implementations, to provide an easy migration path for existing CAN applications. The minimum required differences are new configuration registers for the CAN FD operation.

The CAN FD protocol allows frames with more than eight data bytes. It is not required that all CAN FD implementations support longer frames, CAN FD implementations may be limited to a subset of DATA FIELD length. A CAN FD implementation that supports only up to e.g. eight data bytes in a frame shall not treat longer received frames as an error, fault-free longer frames shall be acknowledged and shall take part in acceptance filtering. Received data bytes that exceed the CAN FD's data handling capacity shall be discarded. A such limited CAN FD implementation that is requested to transmit a longer frame shall padding up the rest of the DATA FIELD that exceeds its data handling capacity with a constant byte pattern. This pattern shall be chosen so that it does not cause the insertion of STUFF BITS, e.g. 0xCC.

The following optional interface registers provide an extended analysis of the ongoing communication:

- Double set of status registers to distinguish between messages and errors occurring while operating in the first or in the second bit rate.
- Dedicated error counter to compare error rates in the two operating modes.
- Counters for successful frames to measure the bus traffic.
- Detection and signalling of potential collisions on the CAN bus when a message is received with the same IDENTIFIER as a transmit message, optionally disabling the transmit message.
- Disabling of transmit messages with a DATA LENGTH CODE above a configured value.
- Per message status flag indicating whether a message was received using only the NOMINAL BIT RATE or also the DATA BIT RATE.
- Per message configuration flag controlling whether a message is to be transmitted using only the NOMINAL BIT RATE or also the DATA BIT RATE.
- Communication management state machine that enables or disables the use of the DATA BIT RATE according criteria like e.g.: Relative error rates in the two bit rates, reception of message in specific bit rate, control message received from external bus master, command written by local host.
- Automatic retransmission of failed frames may be limited to a pre-configured value.
- A transmitting node may be required to suspend following transmissions for a pre-configured time after each successful transmission.
- Trigger output for the synchronization of an external time base to the SAMPLE POINTS of START OF FRAME bits.
- Capturing of time stamps at the SAMPLE POINTS of the START OF FRAME bits of received and transmitted frames.