Embedded Systems

CS 397

TRIMESTER 3, AY 2021/22

Controller Area Network (CAN) (3/3)

[Ref 03] RM0410, Reference Manual STM32F76xxx, Rev 4, Mar2018, ch 40, pp. 1531 – 1576.

Dr. LIAW Hwee Choo

Department of Electrical and Computer Engineering

DigiPen Institute of Technology Singapore

HweeChoo.Liaw@DigiPen.edu

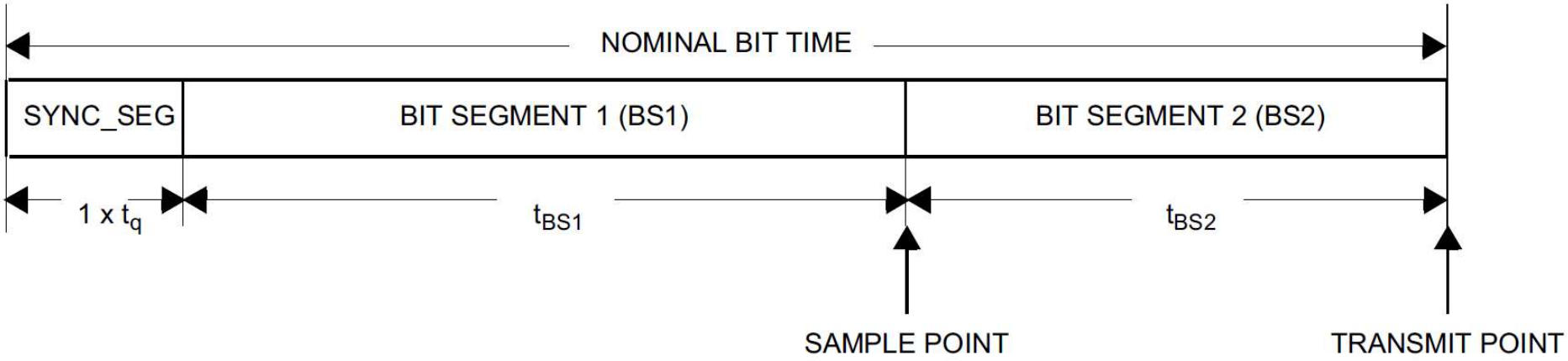
Contents

The bxCAN Bit Timing

- The **bit timing logic** monitors the serial bus-line and performs sampling and adjustment of the sample point and transmit point by synchronizing on the start-bit edge.

- The **nominal bit time** is divided into three segments:

  - **Synchronization segment (SYNC_SEG)**: a bit change is expected to occur within this time segment. It has a fixed length of one time quantum (1 x tq).

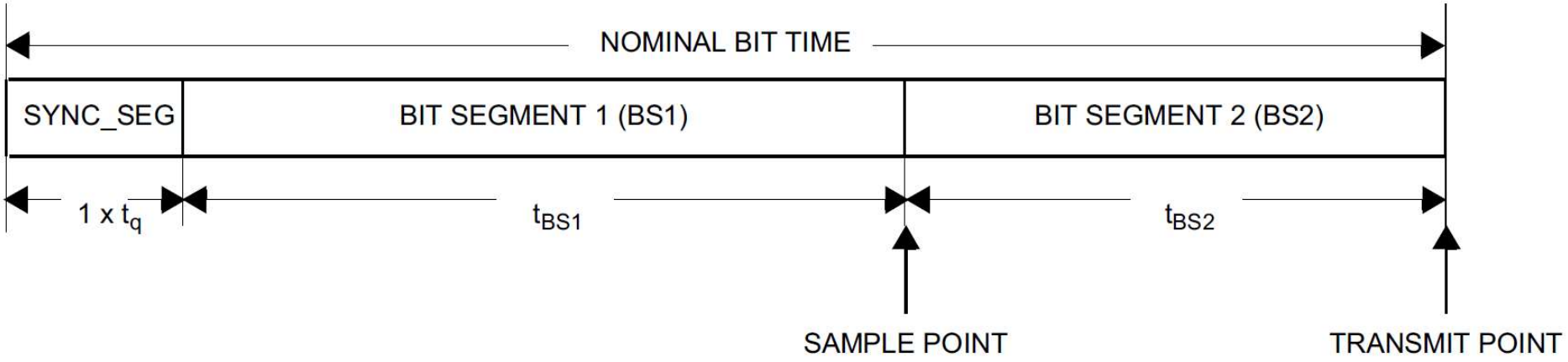


Note: In CAN standard, BS1 includes the PROP_SEG (Propagation Time Segment) and PHASE_SEG1 (Phase Buffer Segment Phase 1), and BS2 represents the PHASE_SEG2 (Phase Buffer Segment Phase 2).

- **Bit segment 1 (BS1)**: defines the location of the sample point. Its duration is programmable between **1 and 16 time quanta** but may be automatically lengthened to compensate for the physical delay times and positive phase drifts due to differences in the frequency of the various nodes on the network.

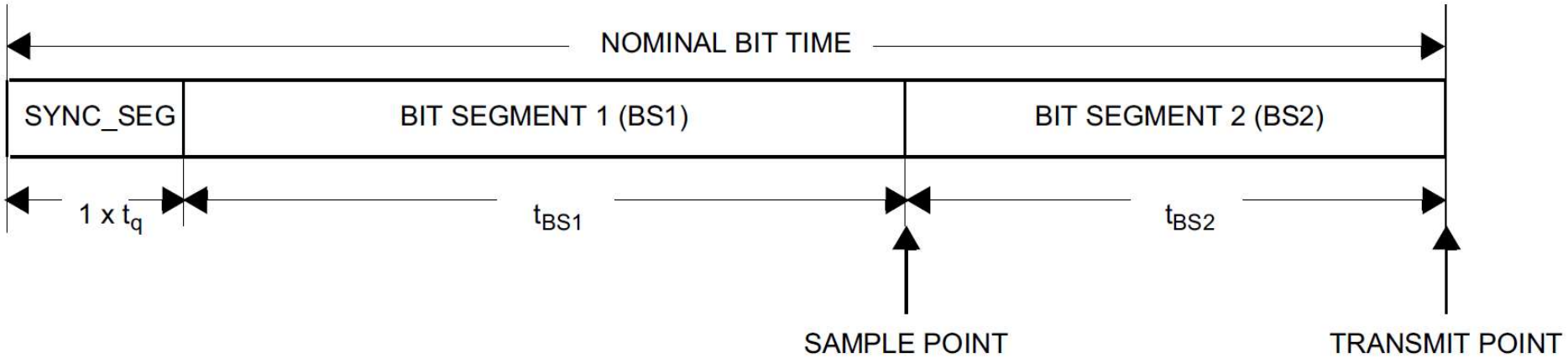
- **Bit segment 2 (BS2)**: defines the location of the transmit point. Its duration is programmable between **1 and 8 time quanta** but may also be automatically shortened to compensate for the negative phase drifts.

- The **Resynchronization Jump Width (SJW)** defines an upper bound to the amount of lengthening or shortening of the bit segments. It is programmable between **1 and 4 time quanta.**



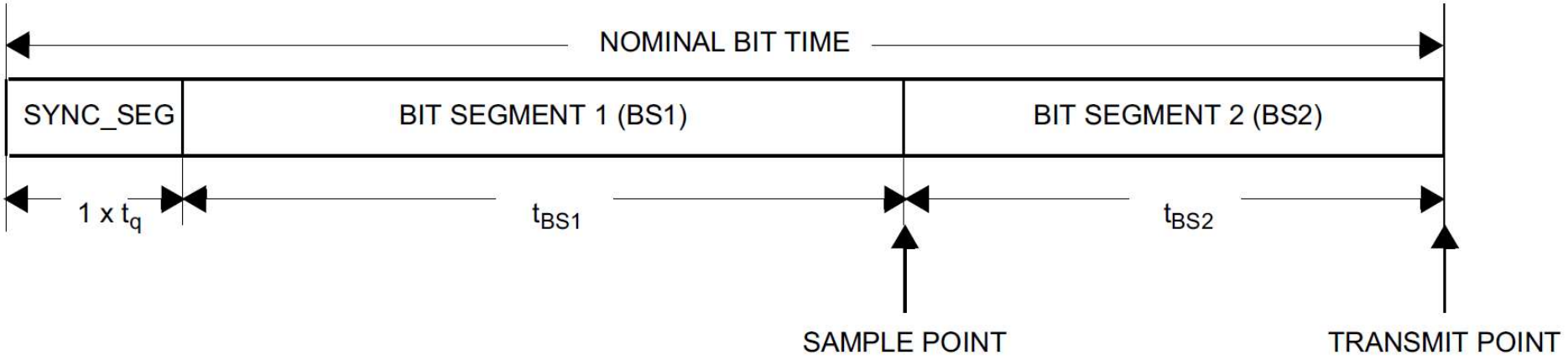
The bxCAN Bit Timing

- A valid edge is defined as the first transition in a bit time [SOF is (1) to (0)] provided the controller itself does not send the (0) bit.
- If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW so that the sample point is delayed.
- Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.
- Note: To prevent errors, the configuration of the CAN Bit Timing Register (CAN_BTR) is only possible when the device is in Initialization mode.



For a detailed description of the CAN bit timing & resynchronization mechanism, refer to the ISO 11898 standard.

Controller Area Network



(Nominal Bit Rate) $$\text{Baud Rate} = \frac{1}{\text{NominalBitTime}}$$

$$\text{NominalBitTime} = 1 \times t_q + t_{BS1} + t_{BS2}$$

with: (in the absence of resynchronization)

$$t_{BS1} = t_q \times (TS1[3:0] + 1), \quad (1\text{-}16)$$

$$t_{BS2} = t_q \times (TS2[2:0] + 1), \quad (1\text{-}8)$$

$$t_q = (BRP[9:0] + 1) \times t_{PCLK}$$

where t_q refers to the Time quantum

t_{PCLK} = time period of the APB clock,

1/PCLK1 for STM32F7

APB1 for STM32F7

The bxCAN bit timing is configured by the CAN Bit Timing Register (CAN_BTR) involving BRP[9:0] (baud rate prescaler), TS1[3:0] (time segment 1), TS2[2:0] (time segment 2), and SJW[1:0] (resynchronization jump width), $t_{SJW} = t_q \times (SJW[1:0] + 1)$. (1-4)

Controller Area Network

CAN bit timing register (CAN_BTR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SILM	LBKM	Res.	Res.	Res.	Res.	SJW[1:0]		Res.	TS2[2:0]			TS1[3:0]			
rw	rw					rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	BRP[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

SILM: Silent mode (debug)

LBKM: Loop back mode (debug)

SJW[1:0]: Resynchronization jump width — tRJW = tq x (SJW[1:0] + 1) (1-4)

TS2[2:0]: Time segment 2 — tBS2 = tq x (TS2[2:0] + 1) (1-8)

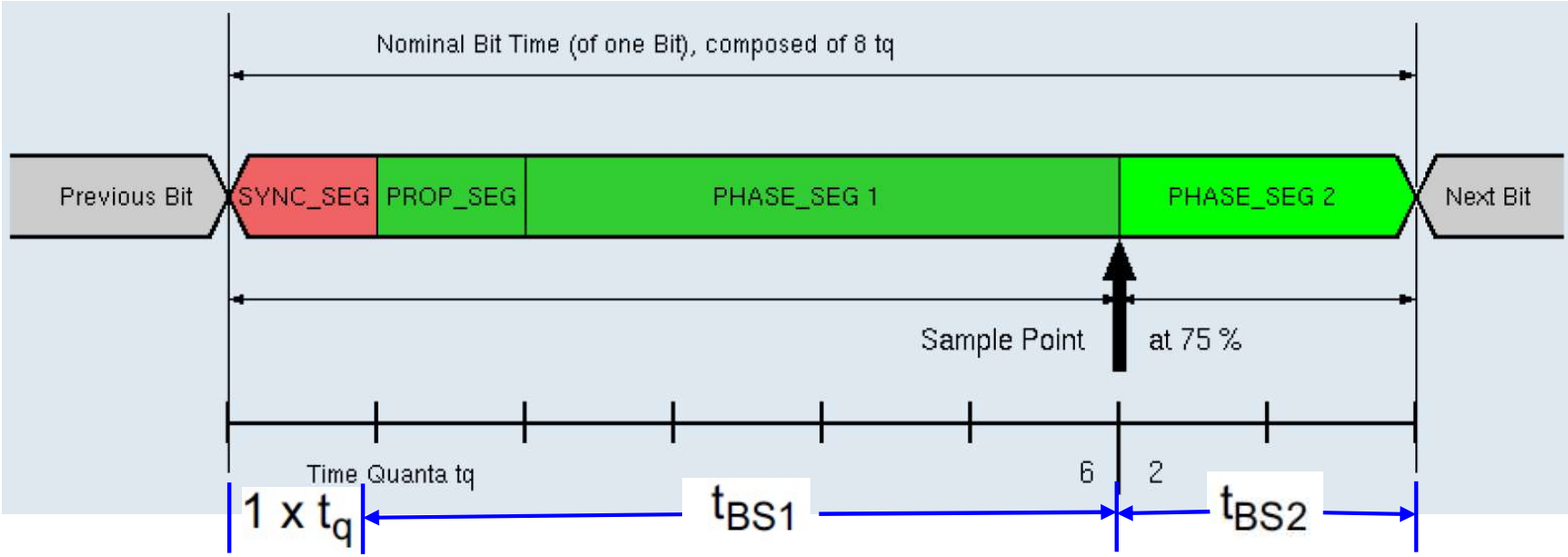
TS1[3:0]: Time segment 1 — tBS1 = tq x (TS1[3:0] + 1) (1-16)

BRP[9:0]: Baud rate prescaler — tq = (BRP[9:0]+1) x tPCLK

1/PCLK1 for STM32F7

The bxCAN Bit Time Calculation

- An example of CAN bit time calculation is presented at http://www.bittiming.can-wiki.info



Recommendation:

- A possible sample point (50% to 90%) is set at **75%**,

  i.e., $t_{BS2} = (t_q + t_{BS1}) \times (1/3)$ or $(t_q + t_{BS1}) = (3 \times t_{BS2})$

  $t_{BS1} = (3 \times t_{BS2}) - t_q$

- Nominal Bit Time = (**8 to 25**) x $t_q = (t_q + t_{BS1}) + t_{BS2} = 4 \times t_{BS2}$

  $t_{BS2} = ((\ 8 \text{ to } 25\ )/4) \times t_q$

- $t_{SJW} = 4 \times t_q$

  $(8, 12, 16, 20, 24) \times t_q$ <--> $t_{BS2} = (2, 3, 4, 5, 6) \times t_q$

75%

$t_{BS1} = (3 \times t_{BS2}) - t_q$

$t_{BS2} = ((\ 8 \text{ to } 25\ )/4) \times t_q$

The bxCAN Bit Time Calculation

(1) Example 1, APB1 = 50 MHz, we want to achieve the CAN Nominal Bit Rate (baud rate) = 50 kbit/s:

t_q = (baud rate prescaler) / APB1 = 50 (selected) / (50 MHz) = 1 μsec

Nominal Bit time = 1 / baud Rate = 1 / (50 kbit/s) = 20 μsec

20 μsec = **20** x t_q = t_q + t_{BS1} + t_{BS2} = 4 x t_{BS2}

$t_{BS2} = (20/4) \times t_q$

--> t_{BS2} = **5** x t_q, t_{BS1} **= 14** x t_q, baud rate prescaler = **50**, and t_{SJW} = **4** x t_q

--> CAN_BTR = 0x034d0031

(1) Example 2, find the CAN bit time settings to achieve CAN Nominal Bit Rate of 1 Mbit/s with PCLK1 = 54 MHz.

t_q = (baud rate prescaler) / PCLK1 = 3 (selected) / (54 MHz) = 1/18 μsec

Nominal Bit time = 1 / baud Rate = 1 / (1 Mbit/s) = 1 μsec = **18** x t_q = **4** x t_{BS2}

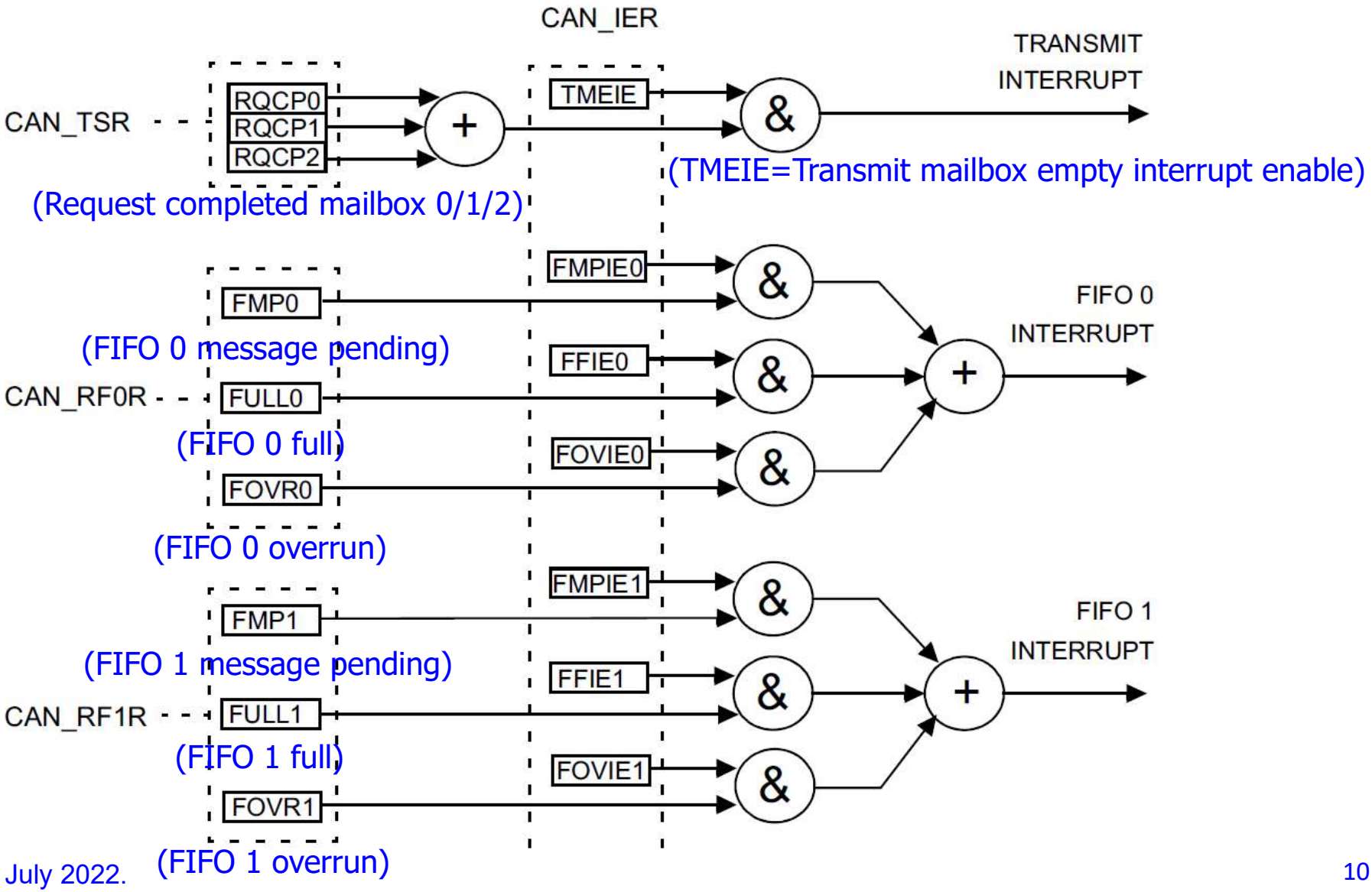
--> t_{BS2} = **4** x t_q, t_{BS1} = **13** x t_q, baud rate prescaler = **3**, and t_{SJW} = **4** x t_q

-- > CAN_BTR = 0x033c0002

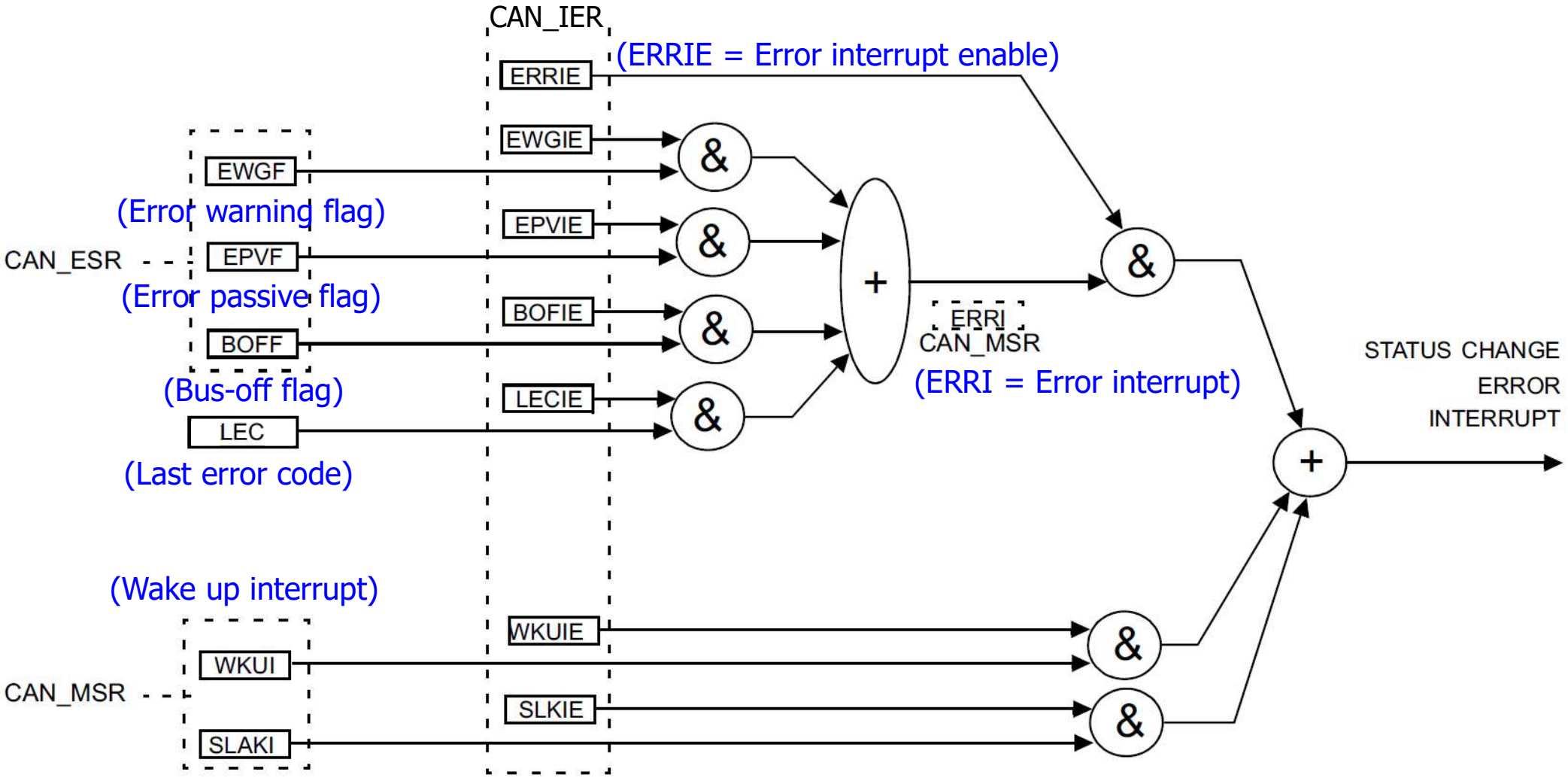
$$\frac{14}{18} \times 100\% = 77.78\%$$

The bxCAN Interrupts (1/2)

Four interrupt vectors are dedicated to bxCAN. Each interrupt source can be independently enabled or disabled by means of the CAN Interrupt Enable Register (CAN_IER).



The bxCAN Interrupts (2/2)



CAN_MSR = CAN master status register
CAN_TSR = CAN transmit status register
CAN_RF0R = CAN receive FIFO 0 register
CAN_RF1R = CAN receive FIFO 1 register
CAN_IER = CAN interrupt enable register
CAN_ESR = CAN error status register

The CAN Frame/Message

- A CAN network has two different message (or frame) formats.
- (a) the "CAN standard/base frame" supports a 11-bit identifier, and
- (b) the "CAN extended frame" supports a 29-bit identifier, which makes up of a 11-bit base identifier and an 18-bit extended identifier.
- The difference is made by using the IDE (identifier extension) bit, which is transmitted as dominant (0) in case of an 11-bit frame and transmitted as recessive (1) in case of a 29-bit frame.
- CAN controllers that support extended frame format messages can send and receive messages in CAN standard/base frame format.
- All frames begin with a start-of-frame (SOF) bit.

The CAN Frame Types

- CAN has four frame types:
  - **Data frame**: a frame containing node data for transmission
  - **Remote frame**: a frame requesting the transmission of a specific identifier
  - **Error frame**: a frame transmitted by any node detecting an error
  - **Overload frame**: a frame to inject a delay between data or remote frame

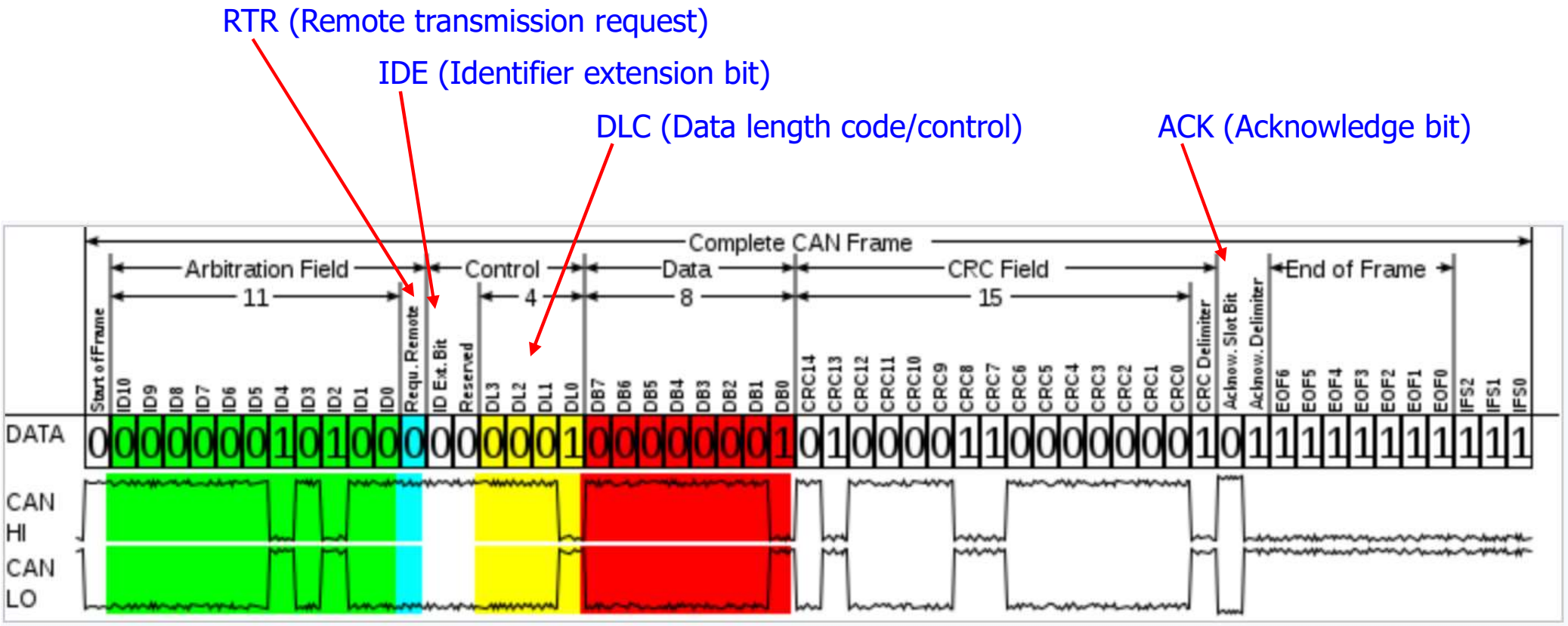
Data frame

- The data frame is the only frame for data transmission. There are two message formats:
  - Standard/Base frame format with a 11-bit identifier
  - Extended frame format with a 29-bit identifier

Controller Area Network

The CAN Standard Data Frame Format (11-bit Identifier)

Source: https://en.wikipedia.org/wiki/CAN_bus



The CAN standard/base data frame format with electrical levels without stuff-bits

The CAN Standard Data Frame Format

[a] It is possible for a value between 9–15 to be transmitted in the 4-bit DLC, but the actual data is always limited to 8 bytes.

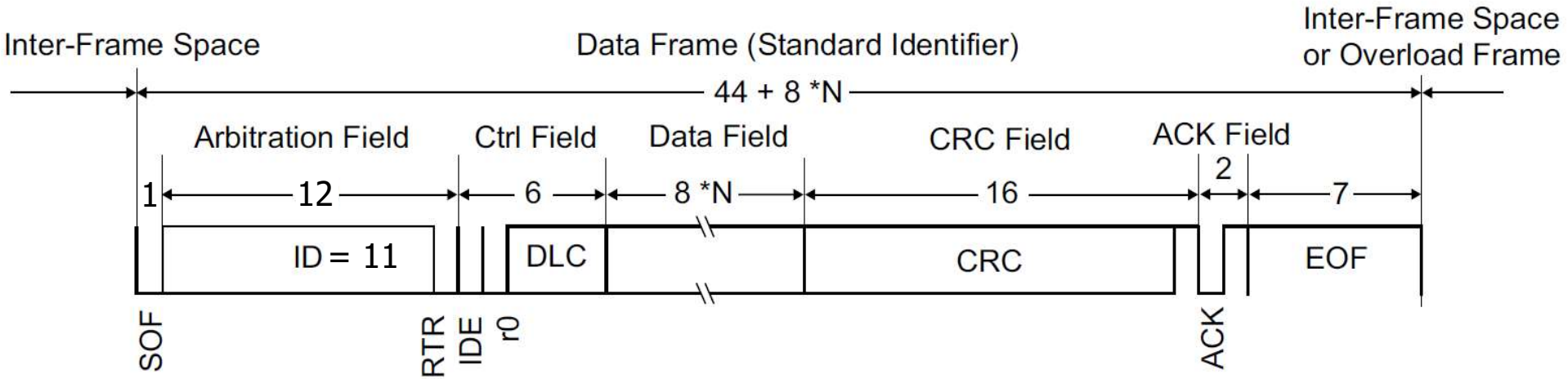
Field name	Length (bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier (green)	11	A (unique) identifier which also represents the message priority
Remote transmission request (RTR) (blue)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames (see Remote Frame, below)
Identifier extension bit (IDE)	1	Must be dominant (0) for base frame format with 11-bit identifiers
Reserved bit (r0)	1	Reserved bit. Must be dominant (0), but accepted as either dominant or recessive.
Data length code (DLC) (yellow)	4	Number of bytes of data (0–8 bytes)[a]
Data field (red)	0–64 (0-8 bytes)	Data to be transmitted (length in bytes dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)

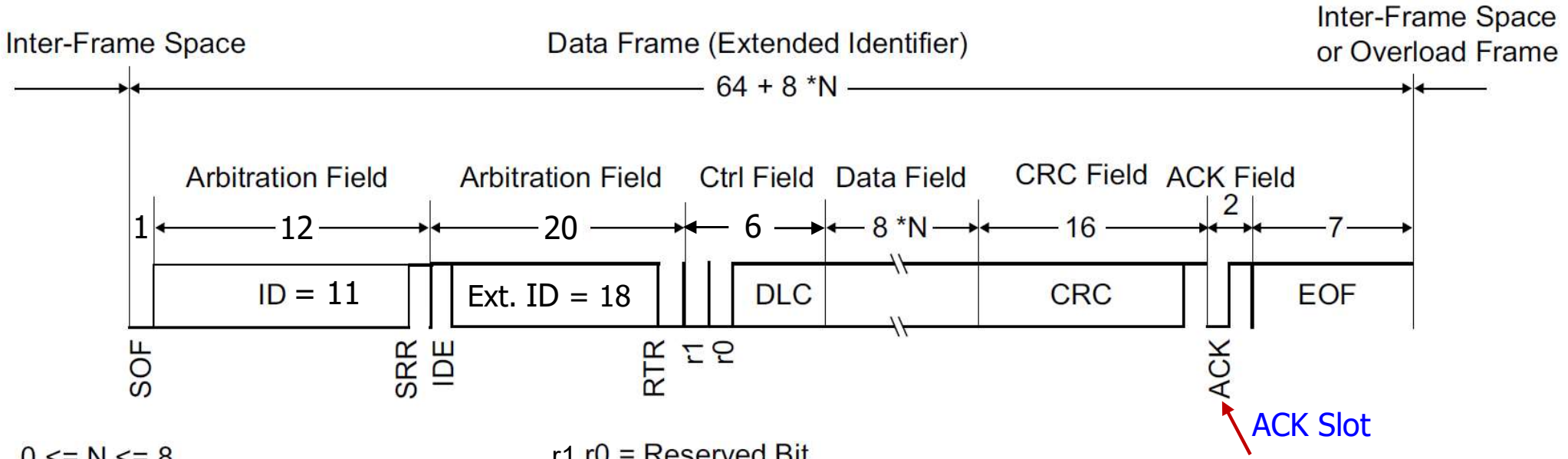
Note: The bit values are described for CAN-LO signal.

Controller Area Network

The CAN Standard and Extended Date Frame Formats

IFS consists of at least 3 consecutive recessive (1) bits





ACK Slot

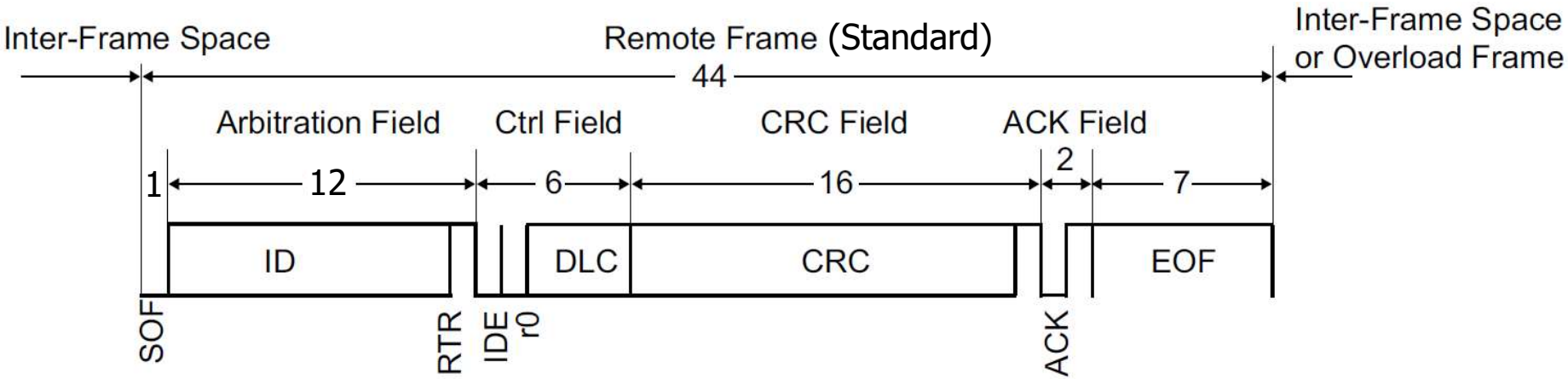
Transmitter sends recessive (1) and any receiver can assert a dominant (0)

0 <= N <= 8
SOF = Start Of Frame
ID = Identifier
RTR = Remote Transmission Request
IDE = Identifier Extension Bit
SRR = Substitute Remote Request

r1,r0 = Reserved Bit
DLC = Data Length Code
CRC = Cyclic Redundancy Code
EOF = End of Frame
ACK = Acknowledge bit
Ctrl = Control

Note: The 7 most significant bits of the 11-bit ID must not be all recessive (1).

The CAN Remote Frame Format



- A node is possible to request for data from another node on the CAN network by sending a Remote Frame.

- A Remote Frame exists in both Standard Format and Extended Format.

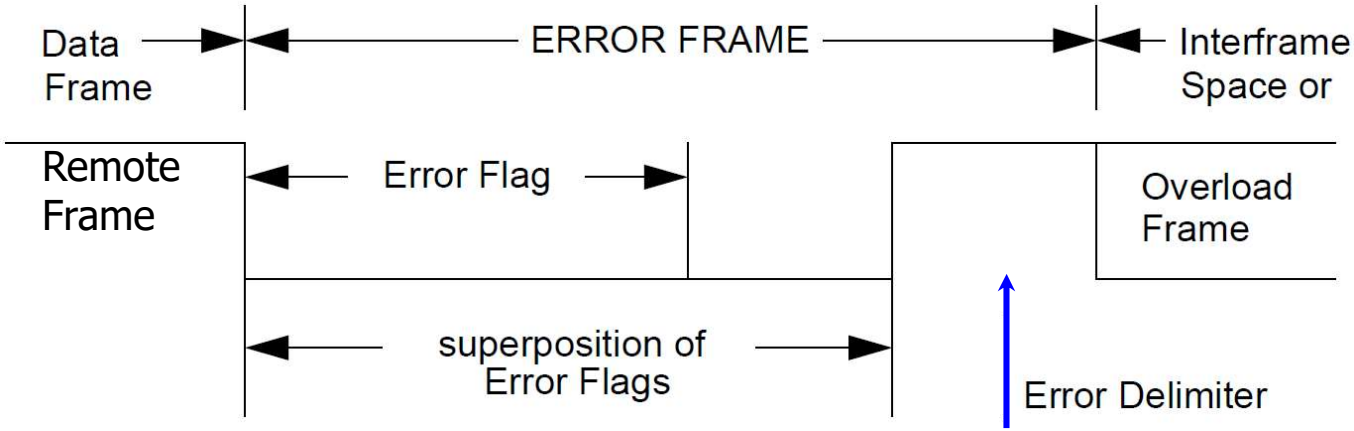
- In a Remote Frame, the RTR-bit is transmitted as a recessive bit, and there is no Data Field. The DLC field indicates the data length of the requested message.

  RTR = 0 : Dominant in Data Frame
  RTR = 1 : Recessive in Remote Frame

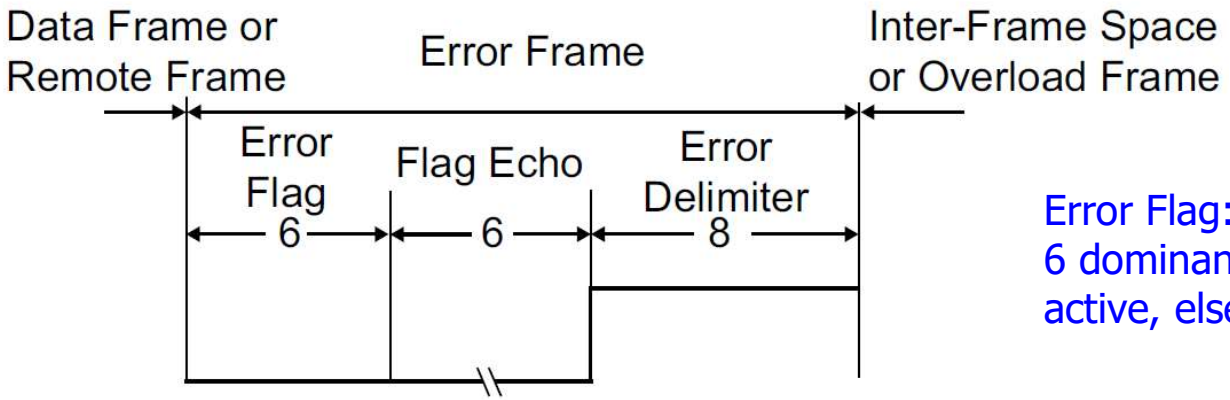
  A problem occurs when two CAN devices with different DLC sending request at the same time.

- If the Data and Remote Frames having the same identifier transmitting at a same time, the Data Frame will win the arbitration due to dominant RTR bit following the identifier.

The CAN Error Frame Format



- The Error Frame consists of two different fields:
  - The first field is given by the superposition of **Error Flags** (6 – 12 dominant / recessive bits) contributed from different stations/nodes.
  - The second field is the **Error Delimiter field** (8 recessive bits).



Error Flag:
6 dominant bits if node is error active, else 6 recessive bits

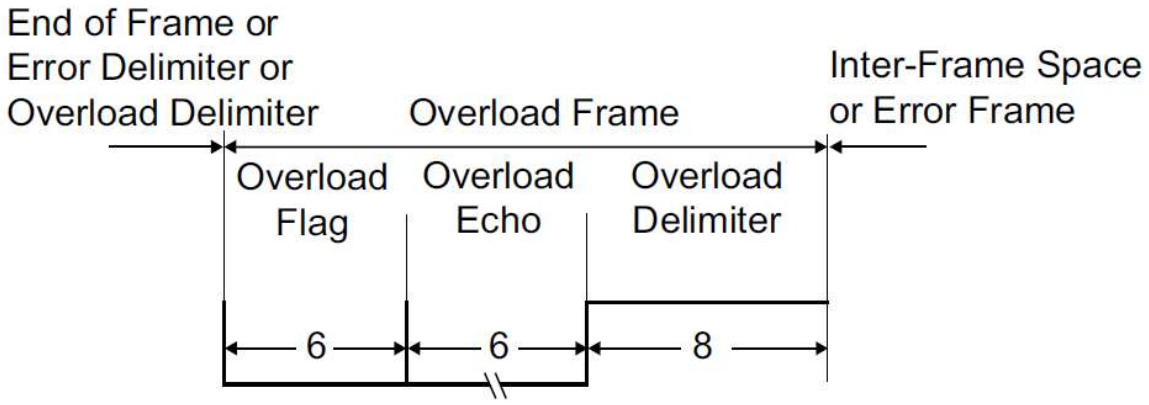
- There are two forms of an Error Flag:

  (1) The **Active Error Flag** consists of six consecutive dominant (b000000) bits. These are transmitted by an "Error Active" node when detecting an error condition.

  (2) The **Passive Error Flag** consists of six consecutive recessive (b111111) bits. These are transmitted by an "Error Passive" node when detecting an error condition. These bits may be overwritten by dominant bits from other nodes.

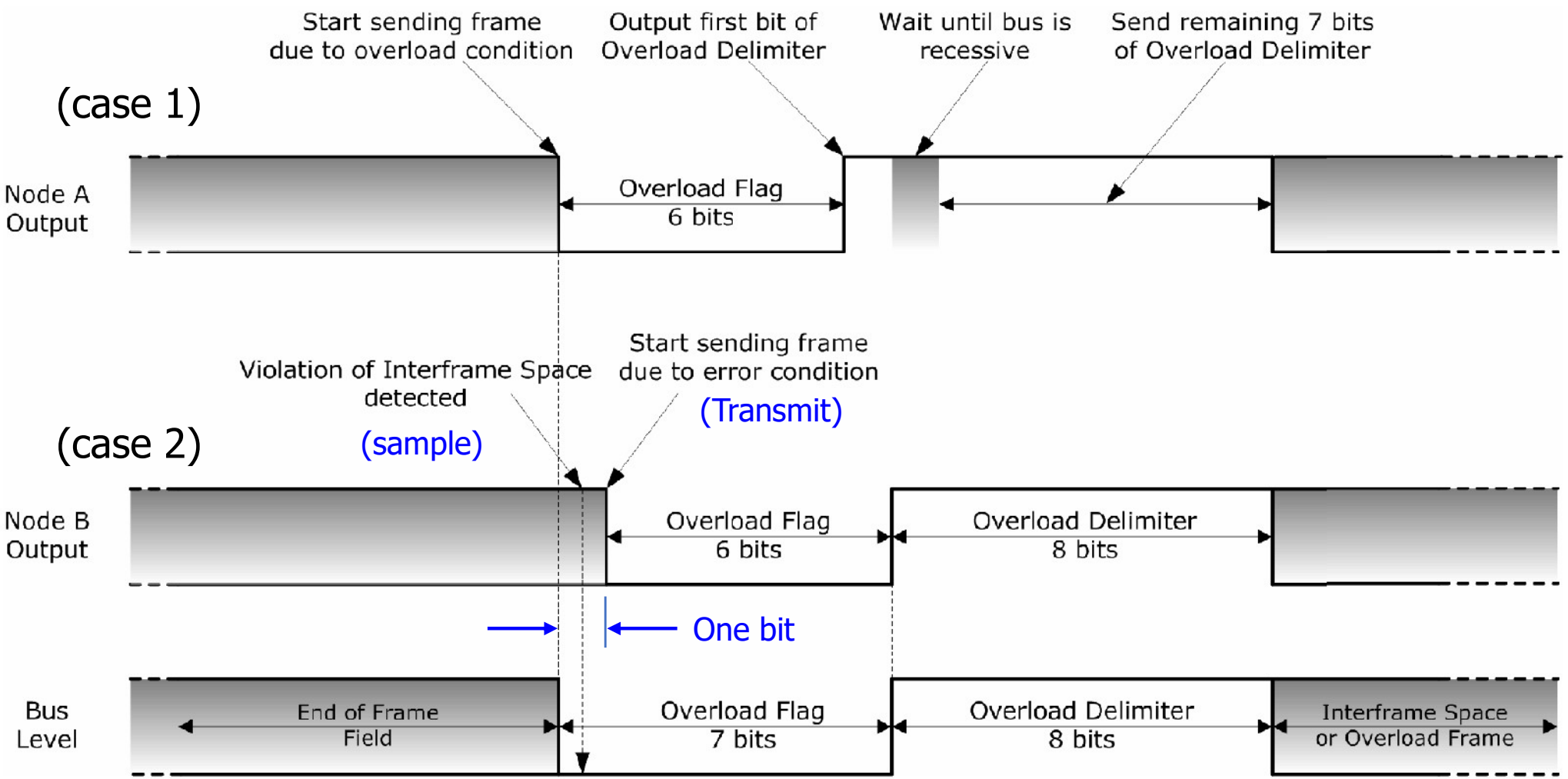
- The Error Flag's format violates the law of bit stuffing applied to all fields from Start Of Frame to CRC field (CRC Delimiter excluded) or destroys the fixed form Ack Field or End Of Frame field. As a result, all other stations/nodes detect an error condition and on their part start transmission of an Error Flag.
- An "Error Passive" node detecting an error condition tries to signal this by transmission of a Passive Error Flag. The "Error Passive" node **waits for six consecutive bits of equal polarity,** beginning at the start of the Passive Error Flag. The Passive Error Flag is complete when these 6 equal bits have been detected.
- After transmission of an Error Flag, each node sends the Error Delimiter bit field, which consists of eight recessive (b11111111) bits.

The CAN Overload Frame Format



- The Overload Frame contains two fields, Overload Flag (may include overload echo) and Overload Delimiter.
- Two overload conditions that 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/and second bit of Intermission (first 3 bits of Inter-Frame Space).
- The start of an Overload Frame due to case 1 is at the first bit time of an expected Intermission, whereas Overload Frames due to case 2 start one bit after detecting the dominant bit.

The CAN Overload Frame Format



Overload Frame Example

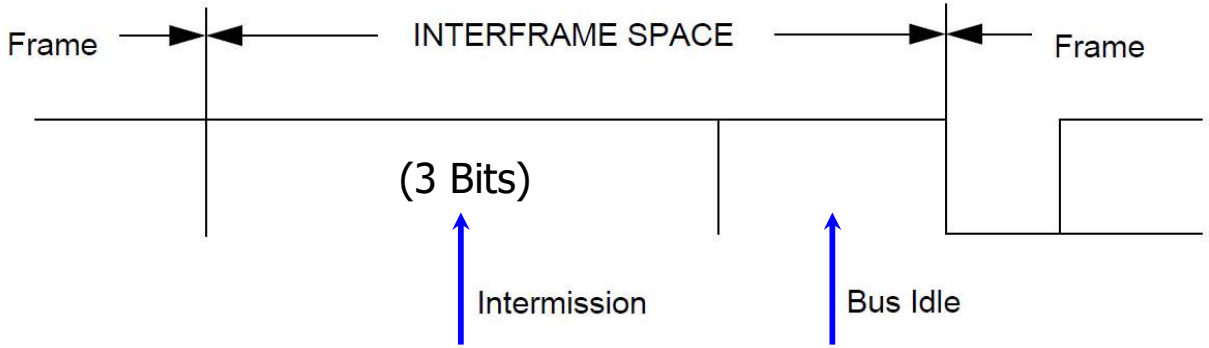
Remark

- If a CAN node samples a dominant bit at the eighth bit (the last bit) of an Error Delimiter or Overload Delimiter, it will start transmit an Overload Frame (start one bit after detecting the dominant bit, not an Error Frame). The Error Counters will not be incremented.
- Overload Flag consists of six dominant (b000000) bits.
- The overall form of an Overload Flag corresponds to that of the Active Error Flag.
- The Overload Flag's form destroys the fixed form of the intermission field. As a result, all other stations also detect an overload condition and on their part start transmission of an Overload Flag (overload echo).
- Overload Delimiter consists of eight recessive (b11111111) 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 recessive bit. At this point of time every bus station has finished its Overload Flag and all stations start transmission of seven more recessive bits in coincidence.

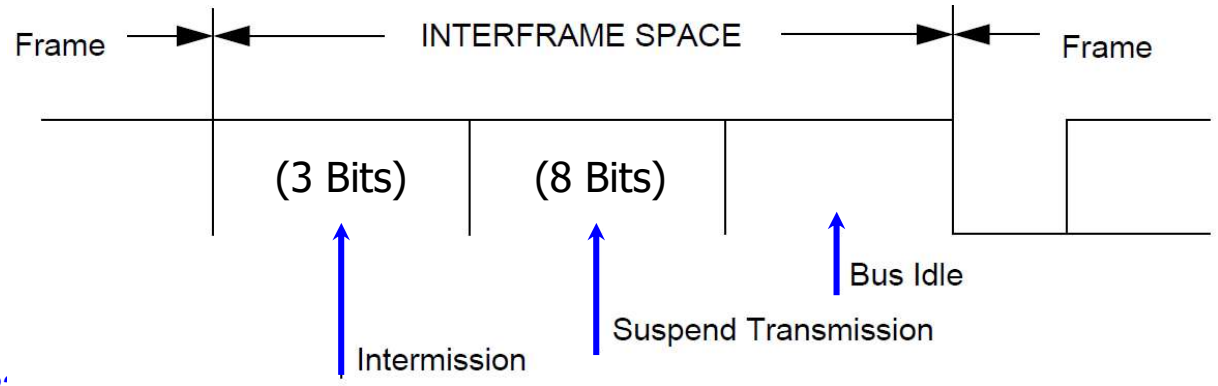
The CAN Inter-Frame Space Format

- Data Frames and Remote Frames are separated from preceding frames by a field called Inter-Frame Space, which contains either

  (1) Intermission and Bus Idle fields (for stations which are not "Error Passive" or have been "Receiver" of the previous message), or

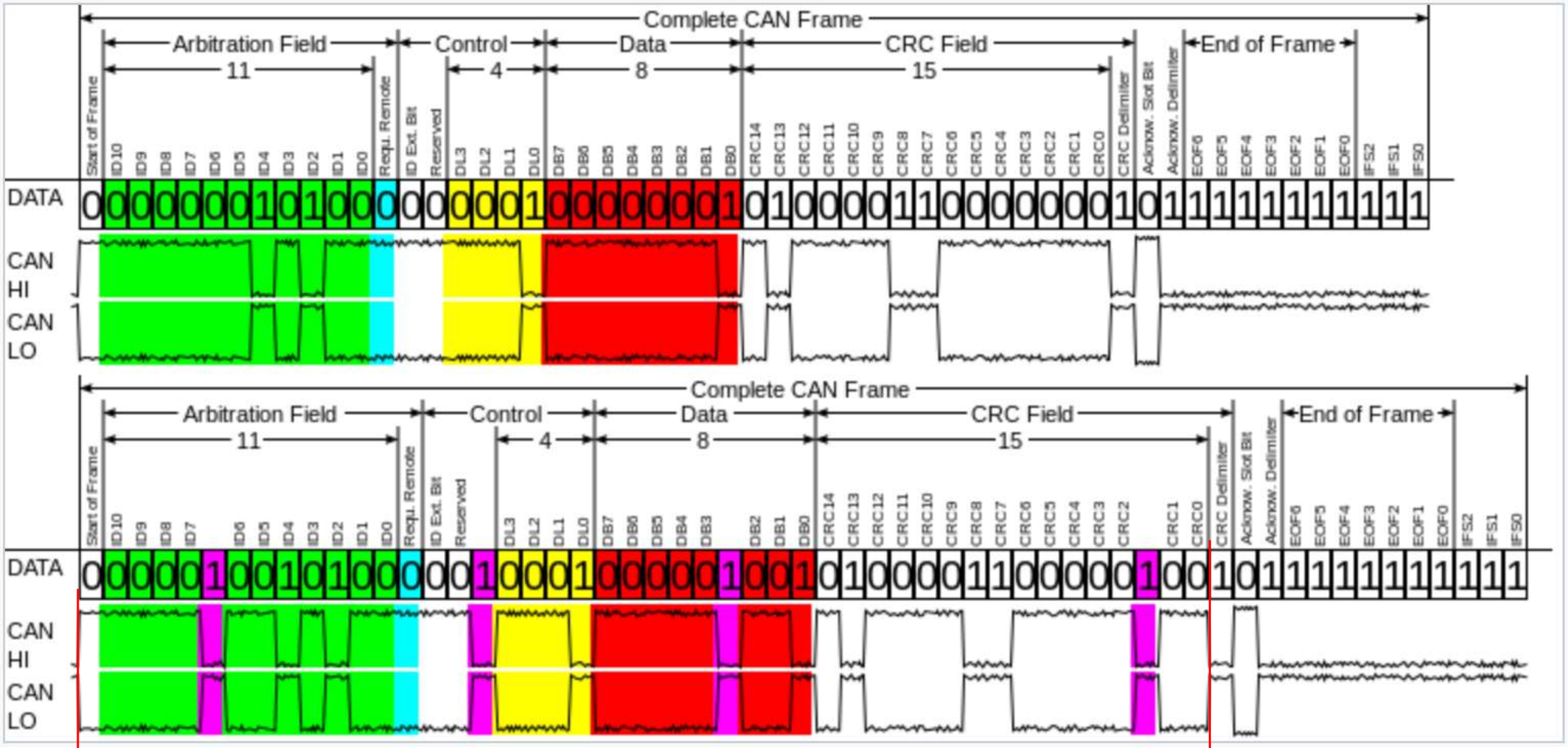


  (2) Intermission, Suspend Transmission, and Bus Idle fields (for "Error Passive" stations which have been "Transmitter" of the previous message).



- The Intermission consists of three consecutive recessive (b111) bits.
- During Intermission, the only action to be taken is signaling an Overload condition and no station is allowed to actively start transmission of a Data Frame or Remote Frame.
- 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 transmitting the Start-of-frame bit.
- Overload Frames and Error Frames are not preceded by an Inter-Frame Space and multiple Overload Frames are not separated by an Inter-Frame Space.
- Bus Idle: The period of Bus Idle may be of arbitrary length.
- A message, which is pending for transmission can be started in the bit following the Intermission.
- Suspend Transmission consists of eight consecutive recessive (b11111111) bits following the Intermission

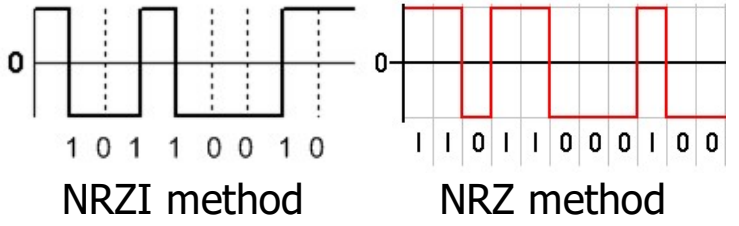
The CAN Bit Stuffing



CAN Data Frame before and after the addition of Stuff Bits (in purple)

Apply bit stuffing

The CAN Bit Stuffing and Bit Stream Coding



NRZI method NRZ method

- The frame segments: Start Of Frame, Arbitration Field, Control Field, Data Field and CRC Sequence/Field (excluded CRC delimiter) 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.
- A stuffed data frame will be de-stuffed by the receiver. (not mentioned in CAN Spec)
- 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 (0) or recessive (1).

The CAN 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.

Receivers:

- The message is valid for the receivers, if there is no error until the last but one bit (i.e., the last 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.

The CAN Error Handling – Error Detection (1/2)

- 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 must 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 (1) bit during the stuffed bit stream of the Arbitration Field or during the ACK Slot. Then no Bit Error occurs when a dominant (0) bit is monitored.

- A transmitter sending a Passive Error Flag and detecting a dominant (0) bit does not interpret this as a Bit Error.

• Stuff Error

- A Stuff Error must 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.

The CAN Error Handling – Error Detection (2/2)

- CRC Error

- The CRC sequence consists of the result of the CRC calculation by the transmitter. The receivers calculate the CRC in the same way as the transmitter.

- A CRC Error must be detected, if the calculated result is not the same as that received in the CRC sequence.

- Form Error

(CRC Delimiter, ACK Field, and End Of Frame)

- A Form Error must be detected when a fixed-form bit field contains one or more illegal bits. (Note that, for a receiver, a dominant bit during the last bit of End Of Frame is not treated as Form Error).

- Acknowledgment Error

- An Acknowledgment Error has to be detected by a transmitter whenever it does not monitor a dominant (0) bit during the ACK Slot.

The CAN Error Handling – Error Signalling

- A node, detecting an error signal, will transmit an Error Flag.
- For an "error active" node, it is an Active Error Flag, and 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 node 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.

The CAN Fault Confinement

- A CAN unit may be in one of three states:
  - "error active"
  - "error passive'"
  - "bus off"
- An "error active" unit can take part in bus communication and sends an Active Error Flag when an error is detected.
- An "error passive" sends Passive Error Flag when an error is detected.
- After a transmission, an "error passive" unit will wait (Suspend Transmission) before initiating a further 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 (TEC), and (2) Receive Error Count (REC)

The CAN Fault Confinement – TEC and REC Changes (1/4)

The Transmit Error Count (TEC) and Receive Error Count (REC) 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 REC 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. (refer Rule 5)

2. ~~When a receiver detects a dominant (0) bit as the first bit after sending an Error Flag the REC will be increased by 8. (refer Rule 5)~~

3. When a transmitter sends an Error Flag, the TEC is increased by 8.

   Exception 1: If the transmitter is "error passive" and detects an Acknowledgment Error because of not detecting a dominant (0) ACK, and does not detect a dominant (0) bit while sending its Passive Error Flag.

   Exception 2: If the transmitter sends an Error Flag because of a Stuff Error occurred during Arbitration, and should have been recessive (1), and has been sent as recessive (1) but monitored as dominant (0).

   In exceptions 1 and 2, the TEC is not changed.

The CAN Fault Confinement – TEC and REC Changes (2/4)

4. If a transmitter detects a Bit Error while sending an Active Error Flag or an Overload Flag the TEC is increased by 8.

5. If a receiver detects a Bit Error while sending an Active Error Flag or an Overload Flag the REC is increased by 8. (refer Rule 1 & 2)

6. Any node tolerates up to 7 consecutive dominant (0) bits after sending an Active Error Flag, Passive Error Flag or Overload Flag. After detecting the 14th (6 + 8)consecutive dominant (0) bit (in case of an Active Error Flag or an Overload Flag) or after detecting the 8th consecutive dominant (0) bit following a Passive Error Flag, and after each sequence of additional eight consecutive dominant (0) bits, every transmitter increases its TEC by 8 and every receiver increases its REC by 8.

7. After the successful transmission of a message (getting ACK and no error until End Of Frame is finished), the TEC is decreased by 1 unless it was already 0.

The CAN Fault Confinement – TEC and REC Changes (3/4)

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 REC is decreased by 1, if it was between 1 and 127. If the REC was 0, it stays 0, and if it was greater than 127, then it will be set to a value between 119 and 127 (120 for STM32F7).

9. A node is "error passive" when the TEC equals or exceeds 128, or when the REC equals or exceeds 128. An error condition letting a node become "error passive" causes the node to send a Passive Error Flag.

10. A node is "bus off" when the TEC is greater than or equal to 256.

11. An"error passive" node becomes "error active" again when both the TEC and the REC are less than or equal to 127.

12. A node which is "bus off" is permitted to become "error active" (no longer "bus off") with its error counters (TEC and REC) both set to 0 after 128 occurrence of 11 consecutive recessive (1) bits have been monitored on the bus.

The CAN Fault Confinement – TEC and REC Changes (4/4)

Note 1: 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.
[STM32F7 implements this error condition using the EWGF (error warning flag) in the CAN Error Status Register (CAN_ESR)]

Note 2: Start-up / Wake-up:
If during start-up only 1 node is online, and if this node transmits some message, it will get no acknowledgment, detect an error and repeat the message. It can become "error passive" but not "bus off" due to this reason. (Rule 3, Exception 1)

Note 3: The rules for increasing and decreasing the error counters are simple: **transmit errors give 8 error points,** and **receive errors give 1 error point**. Correctly transmitted and/or received messages causes the counters/counter to decrease 1 error point unless the counters/counter = 0.

The CAN Oscillator Tolerance

- A maximum oscillator tolerance of 1.58% is given and therefore the use of a ceramic resonator at a bus speed of up to 125 Kbits/s as a rule of thumb

- For a more precise evaluation refer to

  Dias, S; Chapman, M;
  "Impact of Bit Representation on Transport Capacity and Clock Accuracy in Serial Data Streams", SAE Technical Paper Series 890532, Multiplexing in Automobiles SP-773, March 1989.

- For the full bus speed range of the CAN protocol, a quartz oscillator is required.

- The chip of the CAN network with the highest requirement for its oscillator accuracy determines the oscillator accuracy which is required from all the other nodes.

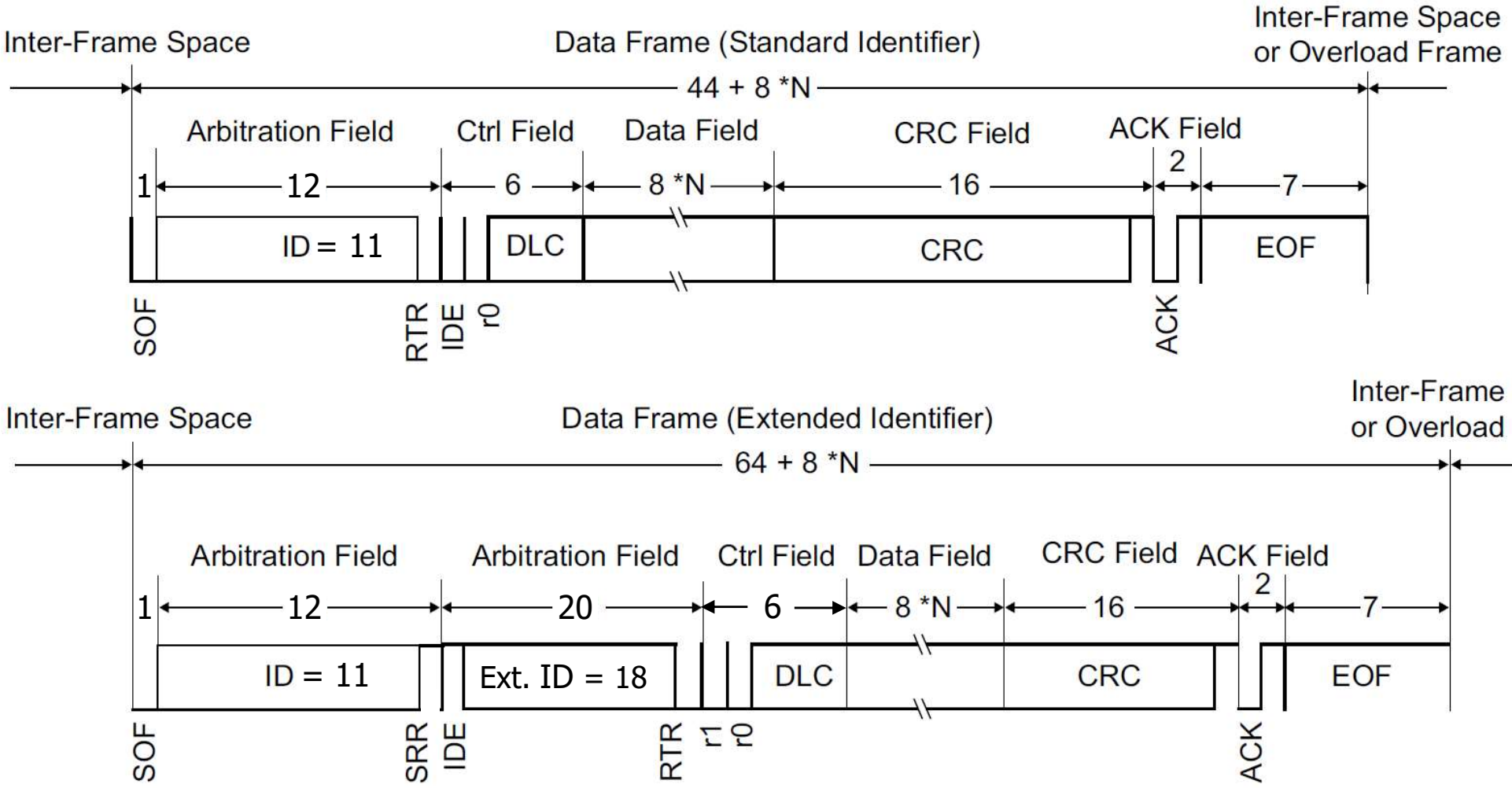
- Quartz crystals are more accurate and temperature stable than ceramic resonators.

  Note: The ceramic resonators can only be used in a network with all the nodes of the network following CAN Protocol Specification versions 1.2 or later.

The CAN Arbitration

- The arbitration field of a standard format CAN message consists of an 11-bit identifier, a RTR (remote transmission request) bit, and an IDE (identifier extension) bit.
- The arbitration field of an extended format CAN message consists of an 11-bit base identifier, a SRR (substitute remote request) bit, an IDE (identifier extension) bit, an 18-bit extended identifier, and a RTR bit.
- The CAN arbitration scheme is called "carrier sense multiple access with collision detection" or CSMA/CD and assures that the highest priority message is broadcasted.
- Message priority is determined by the numerical value of the identifier (include SRR and IDE) in the arbitration field, with the lowest numerical value having the highest priority.
- Non-destructive, bit-wise arbitration resolves conflicts among competing transmitters.
- If any node writes a dominant (0) bit on the bus, every node will read a dominant (0) bit regardless of the value written by that node.

Controller Area Network



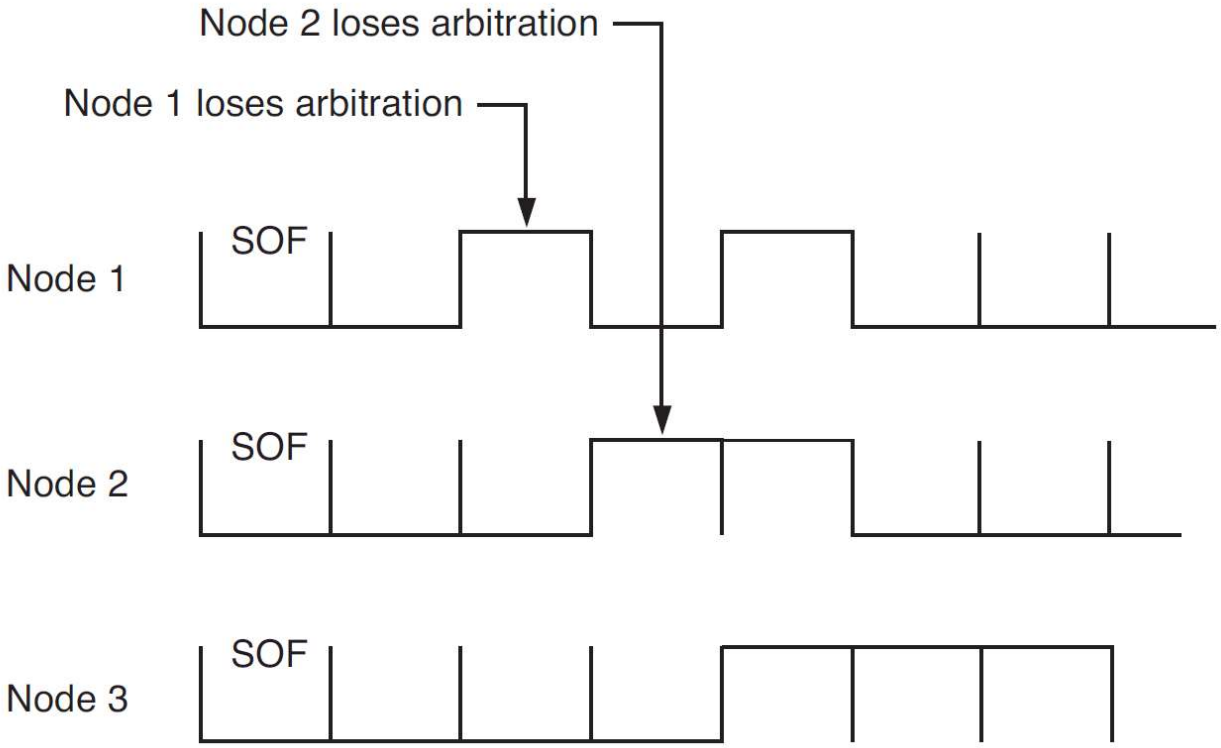
- The SRR (Substitute Remote Request) is a recessive bit. It is transmitted in an extended frame at the position of the RTR bit in a standard frame. In arbitration between standard and extended frames, recessive SRR guarantees the standard message frame prevails.
- The IDE must be recessive for an extended frame and dominant for a standard frame.

The CAN Arbitration

- Every transmitting node always reads back the bus value for each bit transmitted. If a node transmits a recessive (1) bit and reads back a dominant (0) bit, it immediately stops transmitting.

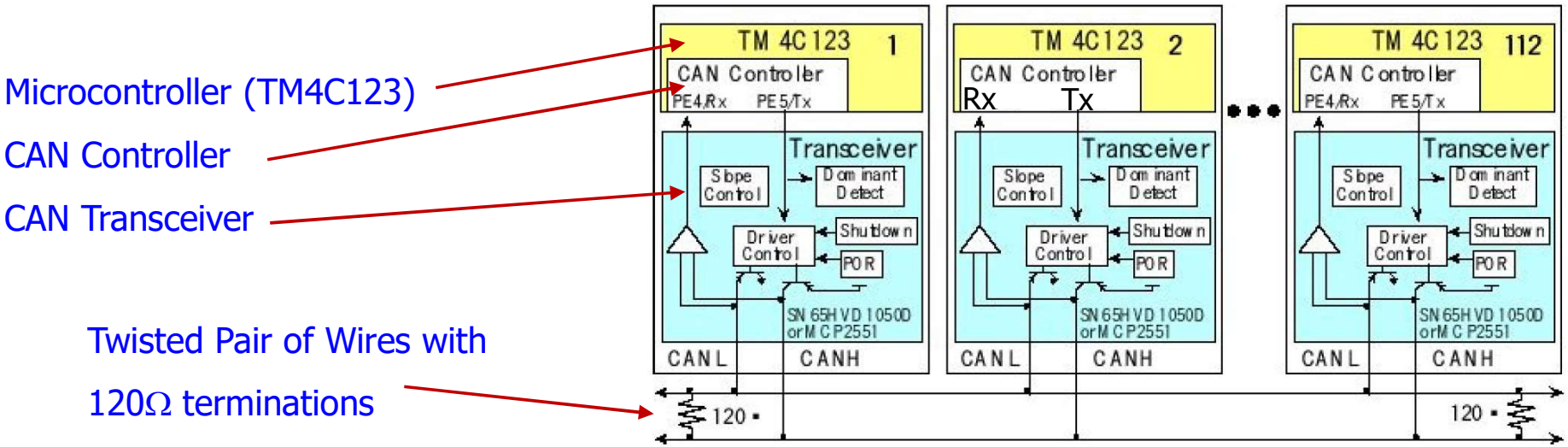
- The RTR bit simply distinguishes between data frames and remote frames. In data frames, the RTR bit must be dominant (0); in remote frames it must be recessive (1).

CAN Arbitration:
Node 3 has highest, and
Node 1 the lowest,
priority messages.



The CAN Transceiver

- There are four major components in a CAN system:
  - **A twisted pair of wires,** CANH (CAN High) and CANL (CAN Low), with a **120Ω** termination resistor on each end of the CAN bus.
  - **CAN transceiver,** which handles the voltage levels and interfacing the separate receive (Rx) and transmit (Tx) signals onto the CAN bus.
  - **CAN controller**, which is the hardware built into the microcontroller, and it handles message timing, priority, error detection, and transmission of CAN messages, and
  - **Embedded software** that implements the functions for CAN controller to generate data for transmission and to process data received from other node(s).



A Microcontroller-Based CAN system

The CAN Transceiver

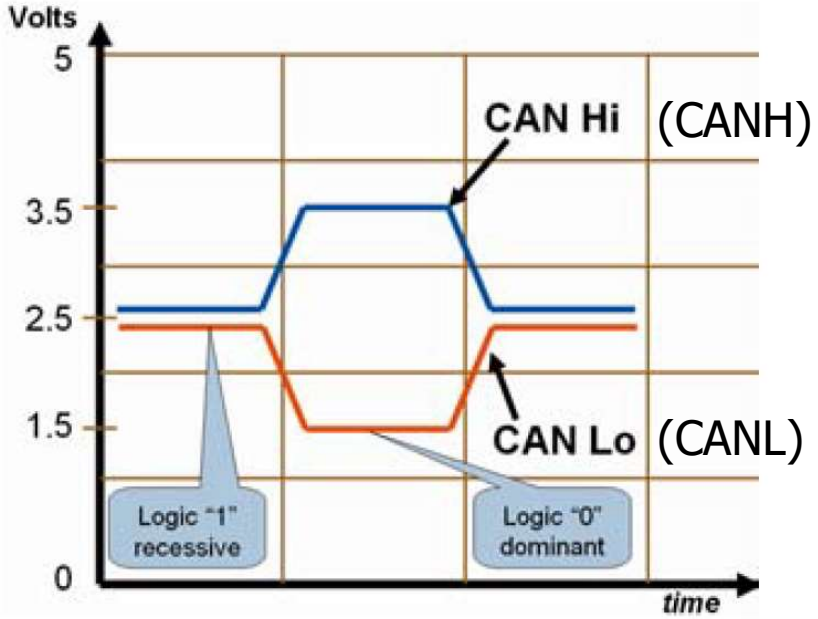
- Each node consists of a microcontroller (with an internal CAN controller), and a CAN transceiver that interfaces the CAN controller to the CAN bus.
- A **CAN transceiver** is a device capable of transmitting and receiving on the same physical channel.
- The CAN is based on the "broadcast communication mechanism", which follows a message-based transmission protocol rather than an address-based protocol.
- The CAN provides two communication services: the sending of a message (data frame transmission) and the requesting of a message (remote transmission request).
- All other services such as error signaling, automatic retransmission of erroneous frames are user-transparent, which implies that the CAN interface automatically performs these functions.
- Note: The Texas Instruments' microcontrollers, TM4C123 and TM4C1294, have two CAN devices on each microcontroller.

The CAN Transceiver

- The CAN transceiver physical channel consists of two wires containing one digital logic bit in differential mode. There are dominant (0) and recessive (1) states on the CAN transceiver bus, see diagram.

- The CAN transceiver outputs follow a wired-AND mechanism in such a way that if one or more nodes are sending a dominant (0) state, it will override any nodes attempting to send a recessive (1) state.

- CAN Hi voltage with respect to ground changes between 2.5 to 3.5 V.
- CAN Lo changes from 2.5 to 1.5 V.
- The difference between the two lines is either 0 volts (logical "1", recessive) or 2 volts (logical "0", dominant).

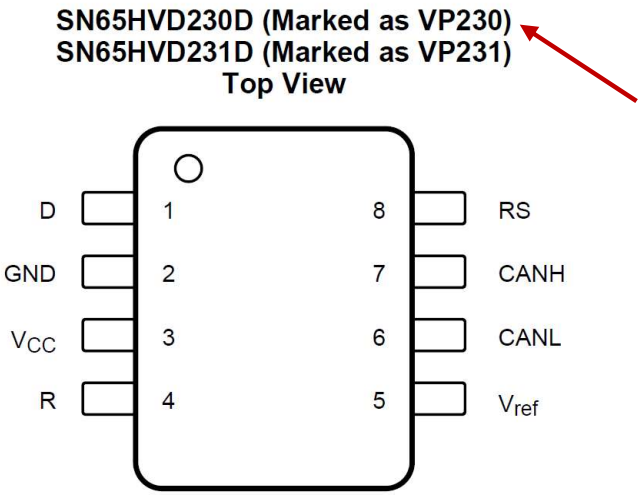


Voltage Specifications for the Recessive and Dominant States

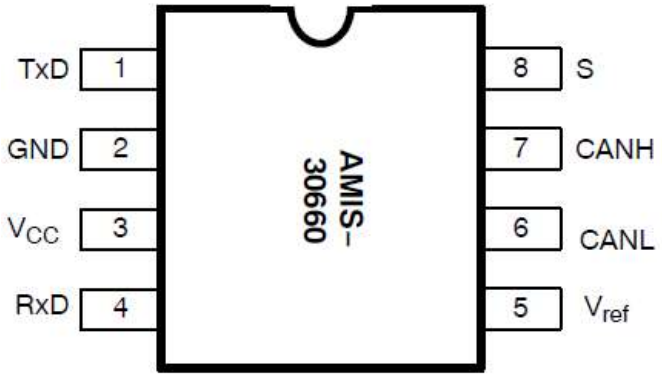
The CAN Transceiver

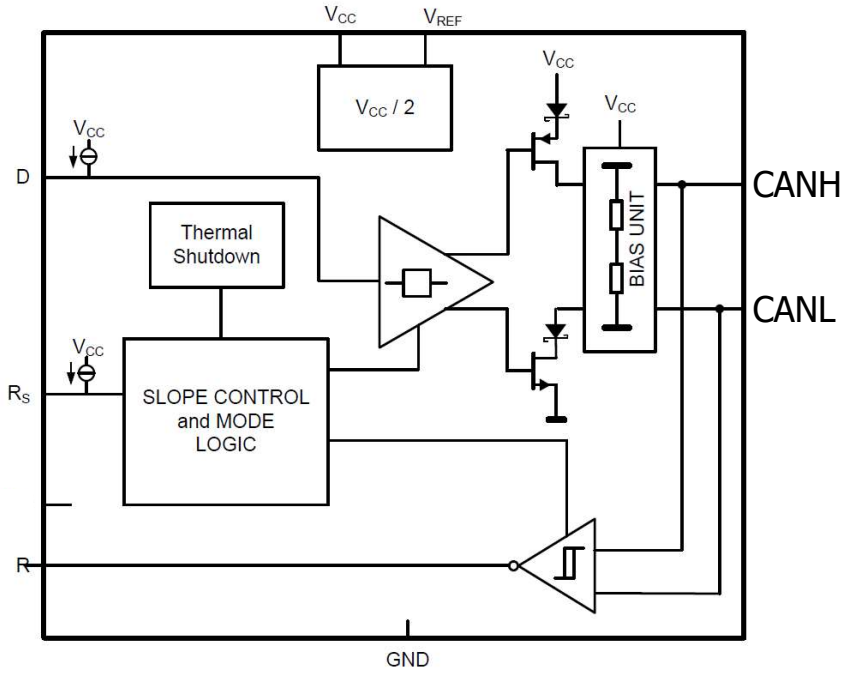
- The CAN transceiver is a high-speed, fault-tolerant device that serves as the interface between a CAN protocol controller (located in the microcontroller) and the physical bus.
- The CAN transceiver can drive the large current needed for the CAN bus and has electrical protection against defective stations.
- Typically, each CAN node must have a device (CAN transceiver) to convert the digital signals generated by a CAN controller to signals suitable for transmission over the bus cabling.
- The CAN transceiver also provides a buffer between the CAN controller and the high-voltage spikes that can be generated on the CAN bus by outside sources.
- Many CAN transceivers on the market today. They are having the similar characteristics and would be equally suitable for implementing a CAN system.

The Block Diagram of CAN Transceivers

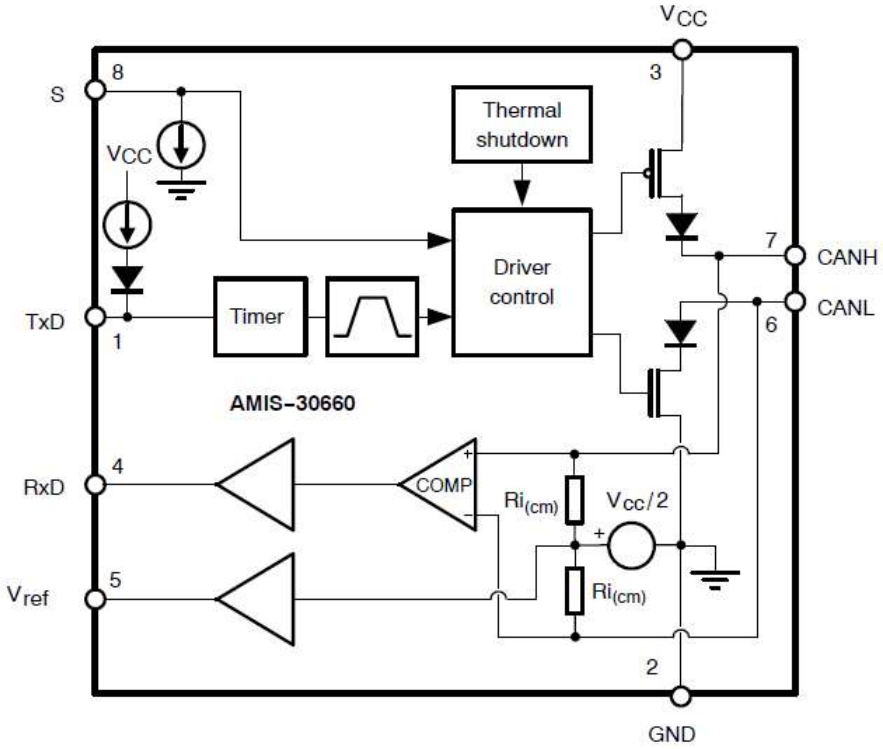


The Pin Assignment





The Block Diagram of SN65HVD230



The Block Diagram of AMIS-30660

Examples of CAN Transceivers

- Texas Instruments SN65HVD230 CAN transceiver, Vcc = 3.0V to 3.6V,
  http://www.ti.com/product/SN65HVD230
- AMIS-30660 high speed CAN transceiver, Vcc = 4.75V to 5.25V,
  https://www.onsemi.com/PowerSolutions/product.do?id=AMIS-30660
- ST Microelectronics L9615 CAN transceiver, Vcc = 4.5V to 5.5V,
  https://www.st.com/en/automotive-analog-and-power/l9615.html
- Philips Semiconductors (NXP) TJA1050 CAN transceiver, Vcc = 4.75V to 5.25V,
  https://www.nxp.com/docs/en/data-sheet/TJA1050.pdf
- Microchip MCP2551 CAN transceiver, Vcc = 4.5V to 5.5V,
  https://www.microchip.com/wwwproducts/en/MCP2551
- Maxim MAX13054A (Vcc = 4.5V to 5.5V) or MAX14878 (Isolated, Vcc_controller = 1.71V to 5.5V, Vcc_bus_side = 4.5V to 5.5V) CAN Transceiver,
  https://wBusww.maximintegrated.com/en/products/interface/transceivers/controller-area-network-transceivers.html

https://www.renesas.com/sg/en/products/space-harsh-environment/rad-hard-analog/rh-can-bus-transceivers.html

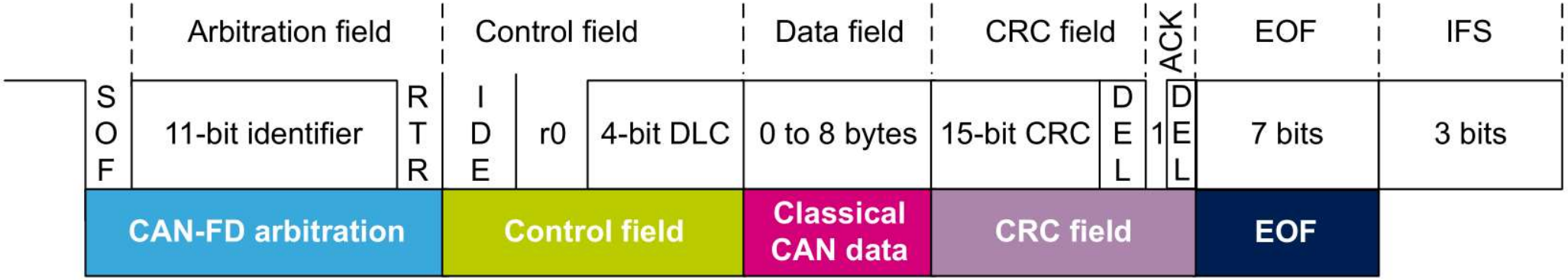
Controller Area Network

(CAN with flexible data-rate)
ISO 11898-1:2015

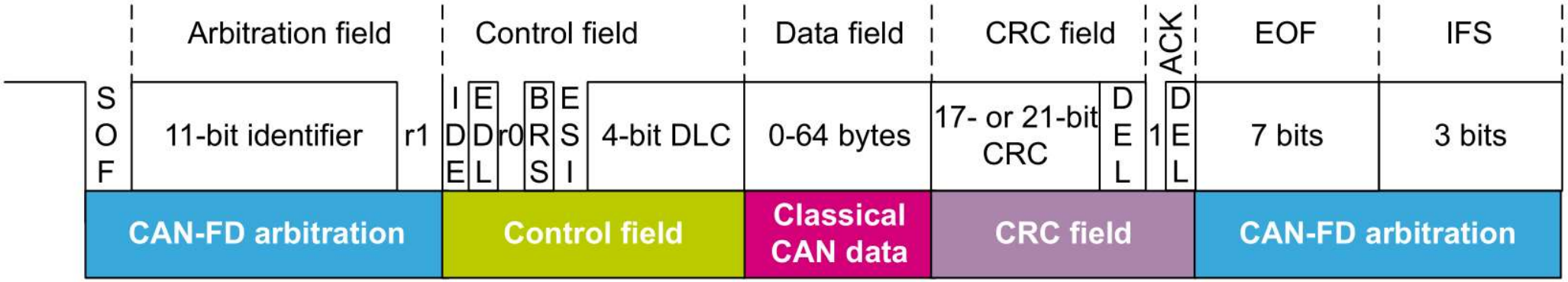
Frame architecture comparison between CAN 2.0 and CAN-FD

AN5348: Implementation of CAN-FD in STM32 devices

CAN 2.0: Classical base frame format



CAN-FD: CAN flexible data rate base frame format



ACK = Acknowledgment
BRS = Bit rate switching
CRC = Cyclic redundancy check
EDL = Extended data length
EOF = End of frame
ESI = Error state indicator
DEL = Deliminator
DLC = Data length code
IDE = Identifier extension
IFS = Interframe space
RTR = Remote transmission request
SOF = Start of frame

Payload data length codes (bytes)

DLC (Dec)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CAN 2.0	0	1	2	3	4	5	6	7	8	8	8	8	8	8	8	8
CAN-FD	0	1	2	3	4	5	6	7	8	12	16	20	24	32	48	64

Main differences between CAN-FD and CAN 2.0

Features	CAN 2.0	CAN-FD
Compatibility	Does not support CAN-FD	Supports CAN 2.0 A/B
Maximum bit rate (Mbit/s)	Frame bitrate: up to 1	Arbitration bitrate: up to 1 Data bitrate: up to 8
DLC field (4 bits) code	Coded in 0 to 8	Coded in 0 to 64
Maximum data bytes in one message	8 bytes of data	64 bytes of data
BRS support	No	Yes
EDL support	No	Yes
ESI support (error active or passive bit)	No	Yes
CRC bits check codes	Bits not included in CRC calculation	Bits included in CRC calculation
Remote frame support	Yes	No

Note: The 29-bit identifier frame is like the standard CAN-FD frame when adding an 18-bit identifier after the bit IDE in the first arbitration phase.

BRS = Bit rate switching
EDL = Extended data length
ESI = Error state indicator