GETTING STARTED WITH CAN FLEXIBLE DATA-RATE (FD)

STEVE MIHALIK
INGO KISSEL

AMF-AUT-T2788 | AUGUST 2017

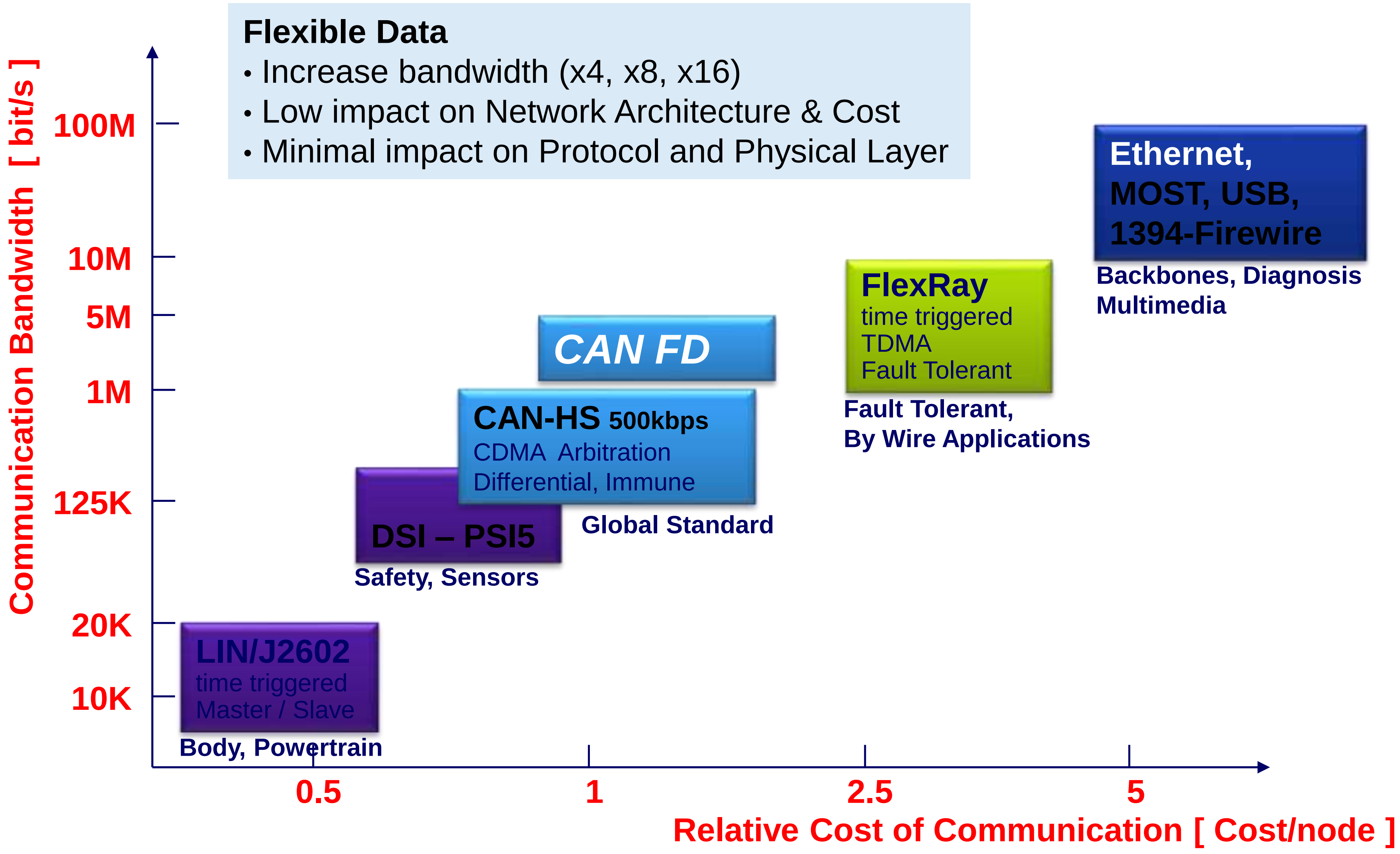
SECURE CONNECTIONS
FOR A SMARTER WORLD

AGENDA

- CAN FD Protocol Overview
- FlexCAN3 FD
- Hybrid Networks

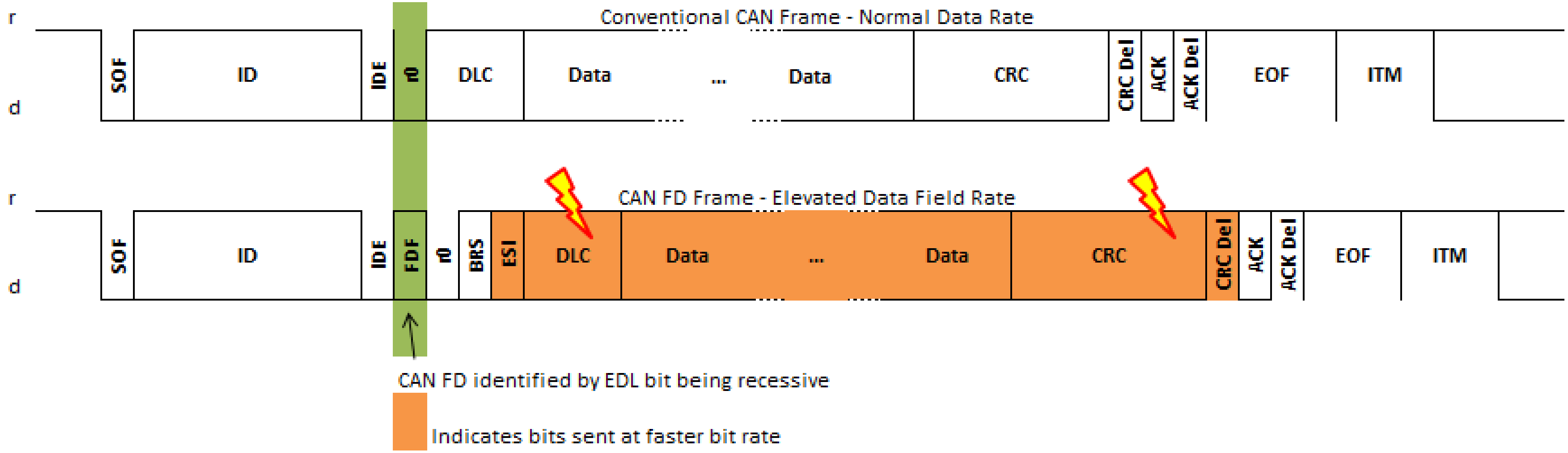
CAN FD Protocol overview

Increasing Bandwidth at Limited Cost



CAN FD Protocol at a Glance

- CAN FD stands for CAN with Flexible Data-Rate
- CAN FD was a proposal by Bosch to
  – Increase the baud rate of the data portion of a CAN message
  – Increase the number of data bytes that can be sent in a single CAN message to up to 64
  – No changes to arbitration field to allow existing physical layers to be used



Background: Conventional Standard CAN Frame

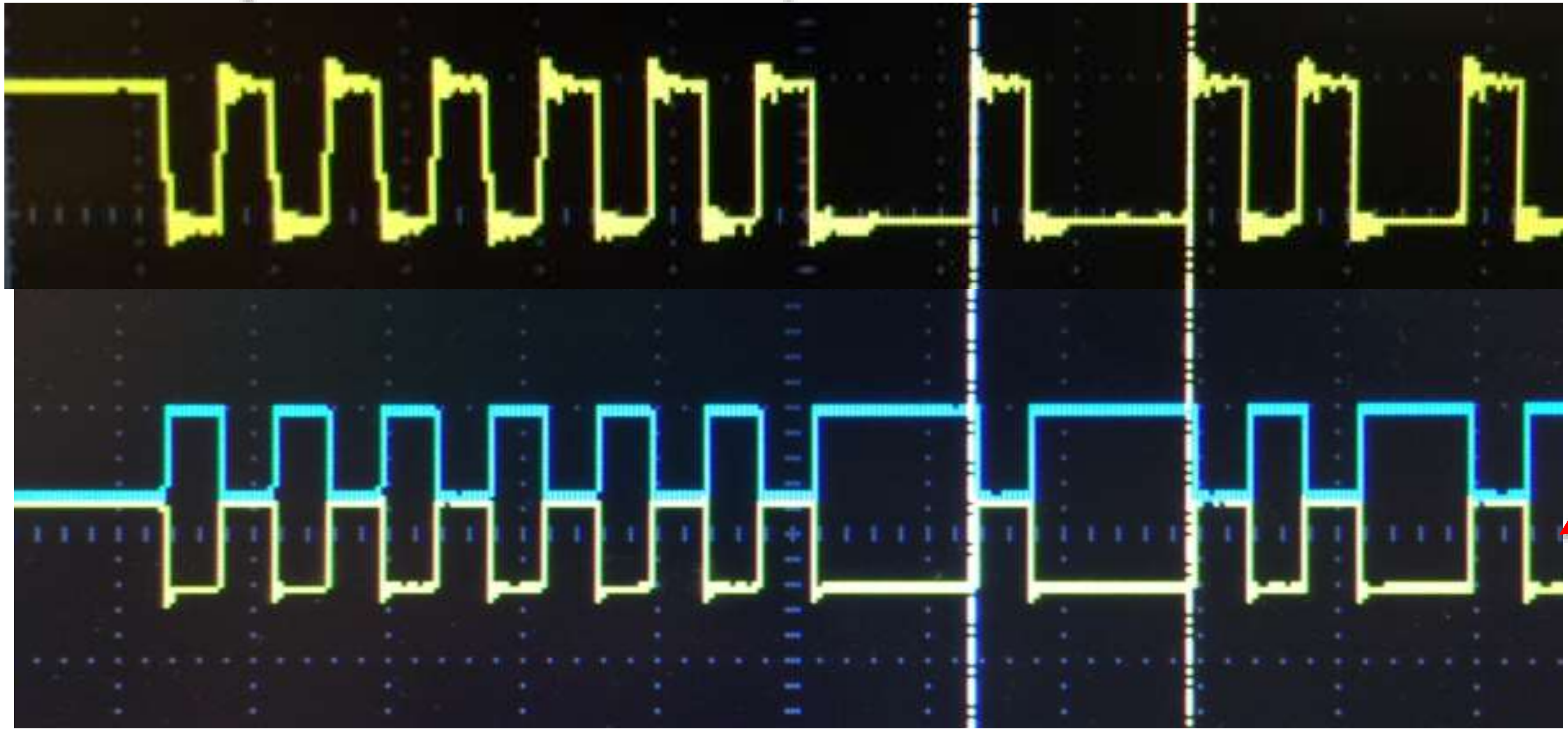
If reserved bit r0 is opposite state (recessive) then frame is CAN FD

11 bit ID
(automatic arbitration)

SOF

RDR
IDE
r0

MCU
CAN_TX
(normally
high)



CAN BUS
(normally
recessive)

Dominant
differential
voltage
typ. 2V

Gradual transition

- Classical CAN nodes can’t listen to CAN FD frames.
- CAN FD nodes can send and receive Classical CAN frames.

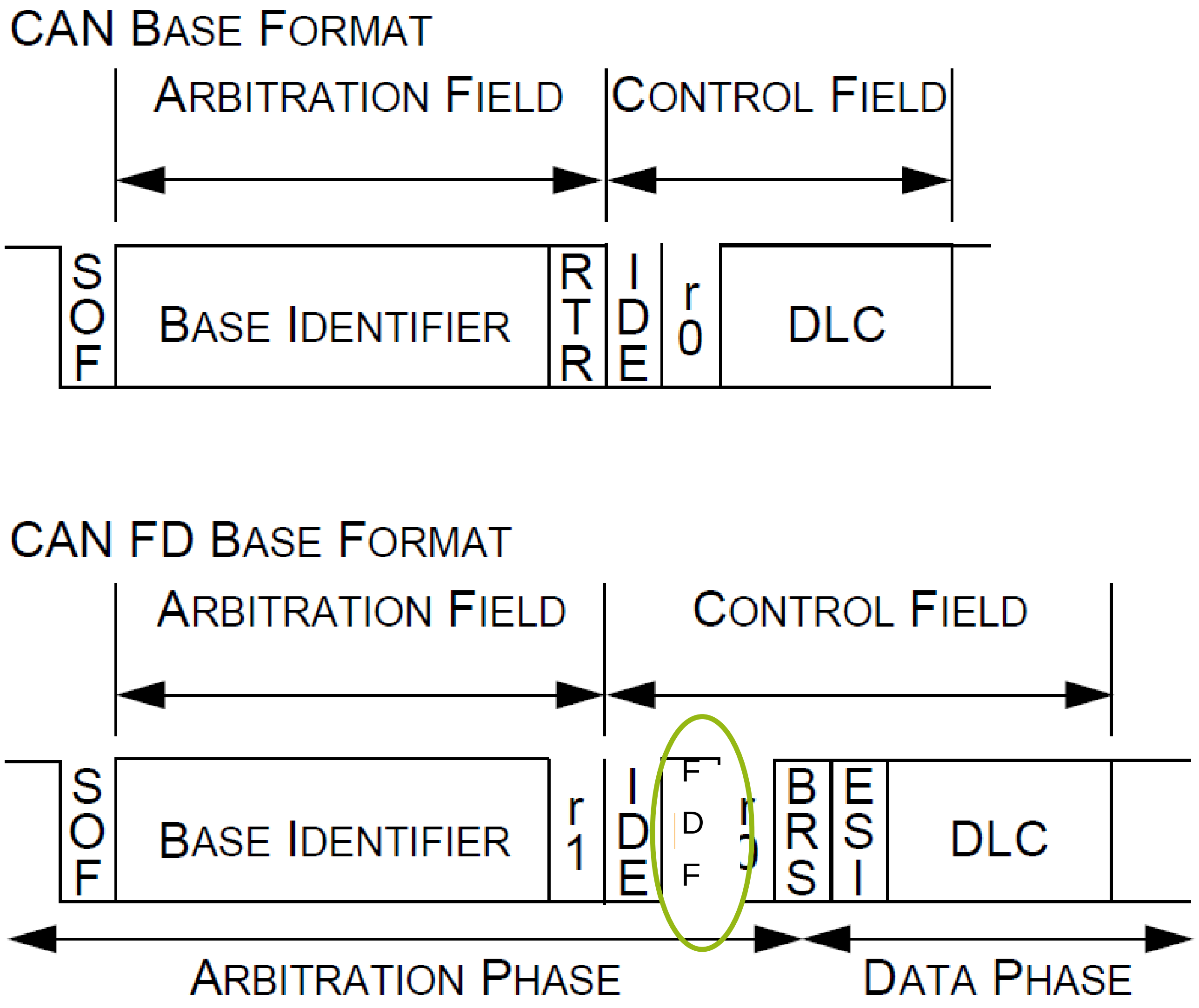
- Are Classical CAN and CAN FD nodes able to coexist?

  - Option 1)
    Introduction phase: CAN FD use only in specific operation modes
    - E.g. software download at end of line programming with non CAN FD nodes in standby or sleep, managed by CAN FD tolerant partial networking transceivers.

  - Option 2)
    Use of CAN FD shield transceivers that make Classical CAN nodes deaf for CAN FD frames
    (-> discussed later)

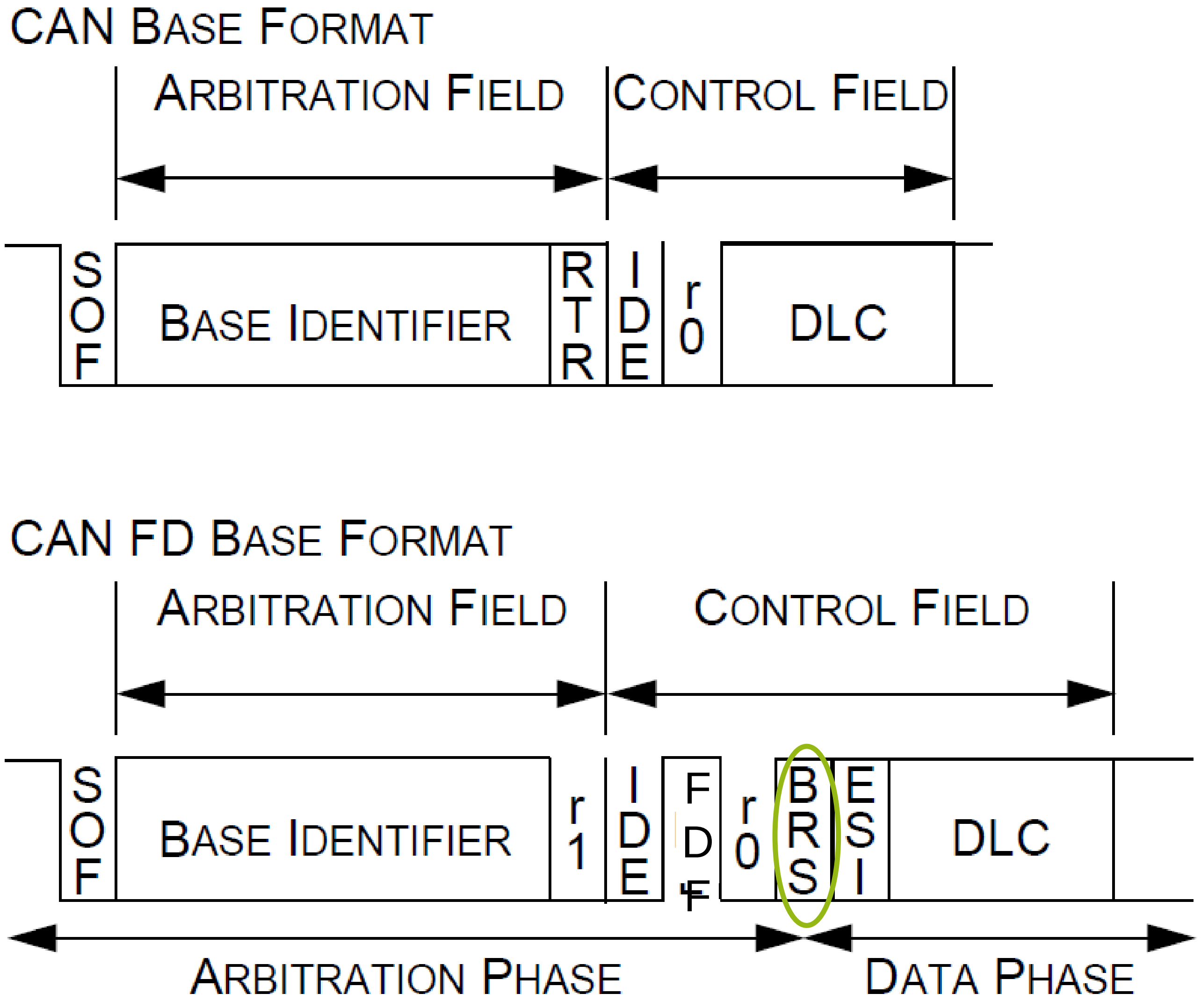
Frame Format – 1

- The FD FRAME FORMAT (FDF) bit is *recessive.*
- *It only exists in CAN FD format* frames, it distinguishes between CAN format and CAN FD format frames



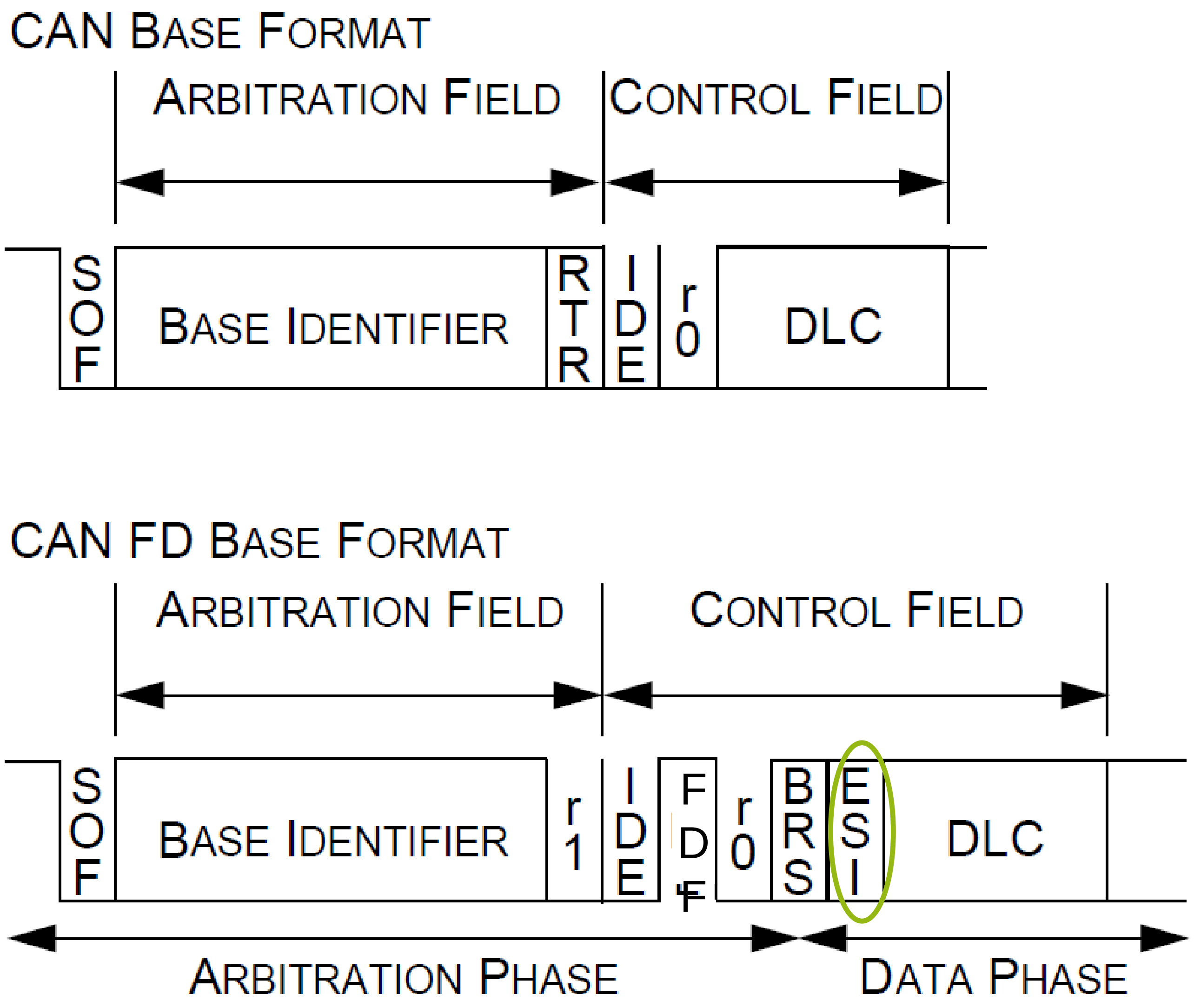
Frame Format – 2

- The BIT RATE SWITCH (BRS) bit decides whether the bit rate is switched inside a CAN FD format frame.
- If recessive, the bit rate is switched immediately at the sample point of the BRS
- Switching back to arbitration speed happens in this case at the sample point of the CRC delimiter.
  Thus the BRS and the CRC delimiter are so called “mixed-bits”.



Frame Format – 3

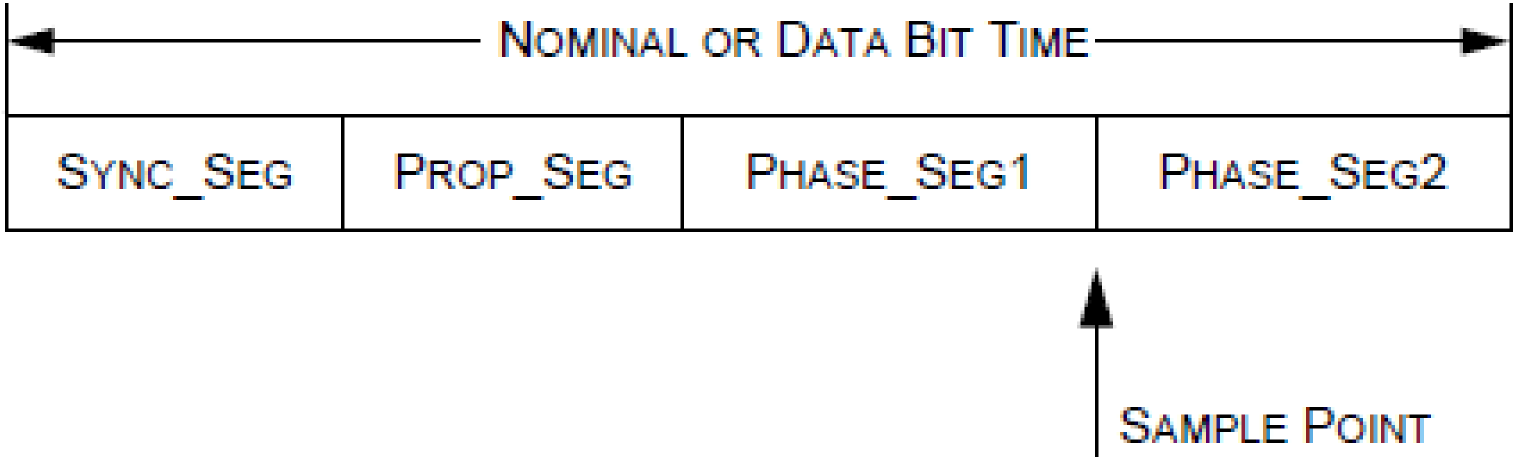
- The ERROR STATE INDICATOR (ESI) flag is transmitted *dominant by error active nodes, recessive by error passive nodes*.



Bit Rates

- Two bit rates:
  - ARBITRATION-PHASE
    - longer bit time = NOMINAL BIT RATE
  - DATA PHASE
    - same or shorter bit time <= NOMINAL BIT RATE

➔ Two bit rate register sets



Bit Time Segments

- Two prescalers ➔ two TIME QUANTUM
  - m(N) for the NOMINAL BIT TIME
  - m(D) for the DATA BIT TIME
- The number of TIME QUANTA in a bit time shall be programmable at least from 8 to 25.

Bit Time Segment	**NOMINAL BIT RATE TIME QUANTA(N)**	**DATA BIT RATE TIME QUANTA(D)**
SYNC_SEG	1	1
PROP_SEG	1, 2,…, 32 or more	0, 1, 2,…, 8
PHASE_SEG1	1, 2,…, 32 or more	0, 1, 2,…, 8
PHASE_SEG2	Max(PHASE_SEG1(N), IPT)	Max(PHASE_SEG1(D), IPT)

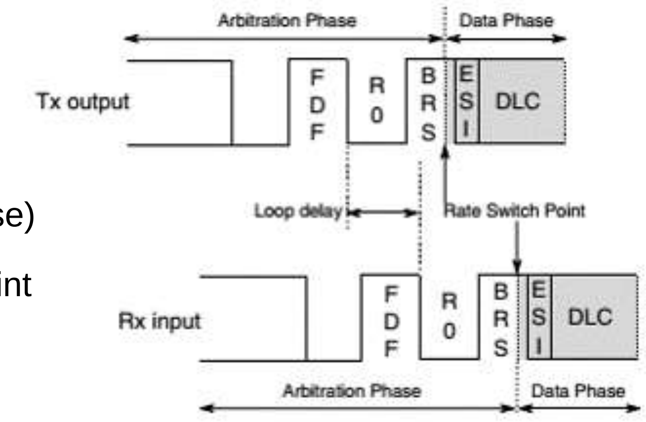
Transceiver Delay Compensation

- CAN transmitters are required to check for errors in their transmitted bits between their CAN_Tx and CAN_Rx pins
  - ISO11898 defines this delay at maximum of 255ns

- When higher bit rate data phase occurs in FD frame, the bit time that elapses until the sample point can be shorter than the transceivers loop delay impeding correct comparison.

- A secondary sample point is defined where the transmitted bit is correctly compared with the received bit to check for bit errors.

- Register to enable:
  - CAN_FDCTRL[TDCEN] – enable, TDC only available once BRS is set (i.e. data phase)
  - CAN_FDCTRL[TDCOFF] – offset value to assist in position of secondary sample point



Data Payload Size

- Data Length Control (DLC) field in the frame specifies the data payload size
- CAN:
  - 0 to 8 bytes
- CAN FD
  - 0 to 64 bytes

Format	# of Data Bytes	DLC3	DLC2	DLC1	DLC0
CAN and CAN FD	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
	8	1	x	x	x
CAN FD	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

CAN FD Fixes

- There are two versions of CAN FD
  - non ISO CAN FD (original Bosch proposal)
  - ISO CAN FD (ISO11898-1:2015) including two fixes
    - For ISO CAN FD, set FlexCAN register bit CTRL2[STFCNTEN] = 1

OEM Requirements

- OEM Requirements
  - Enhanced CRC that includes stuff bit count (ISO 11898-1:2015)
    - GM, Ford, VW, Daimler AG, Renault, PSA have documented that they require full ISO conformance
  - Interleaving Classic CAN frames and CAN FD frames
    - GM, Ford, VW have stated that they require ability to interleave classical and FD frame for series productions.

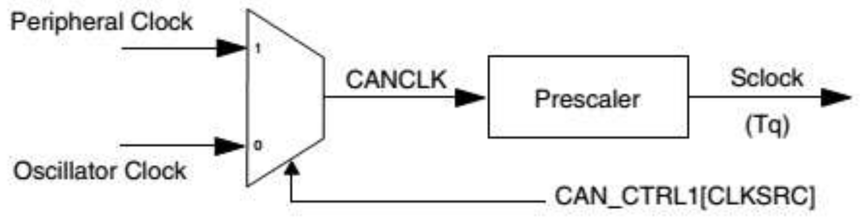
Using FlexCAN3_FD

Features

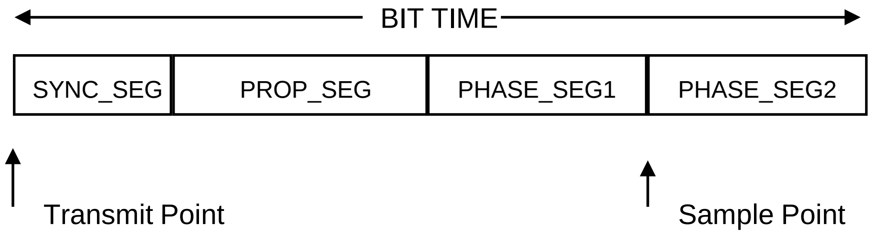
- FlexCAN3-FD Module Features:
  - Up to 128 Message buffers configurable as Tx or Rx
  - Standard and Extended ID frames and Remote Frames* (not for FD)
  - Payload: 0 to 64 bytes data length
  - Programmable bit rate up to 8 Mb/sec
  - Individual Rx Mask Registers per Message Buffer
  - Full featured Rx FIFO
    - Storage capacity for 6 frames and internal pointer handling with DMA support
  - Powerful Rx FIFO ID filtering
    - Capable of matching incoming IDs against 128 extended, 256 standard or 512 partial (8-bits) IDs, with individual masking capability
    - Programmable acceptance filters for receive message buffers
  - 16-bit time Stamp

Protocol Bit Timing

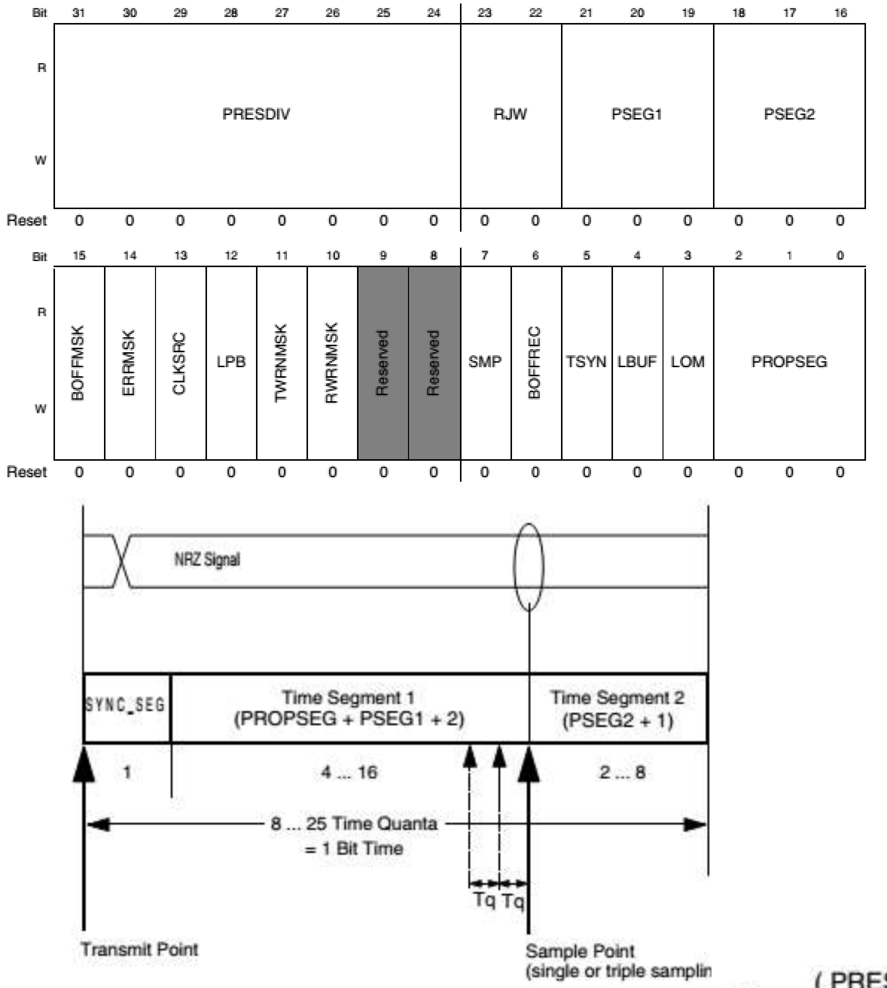
- FlexCAN supports either oscillator or peripheral clock as CAN timing source



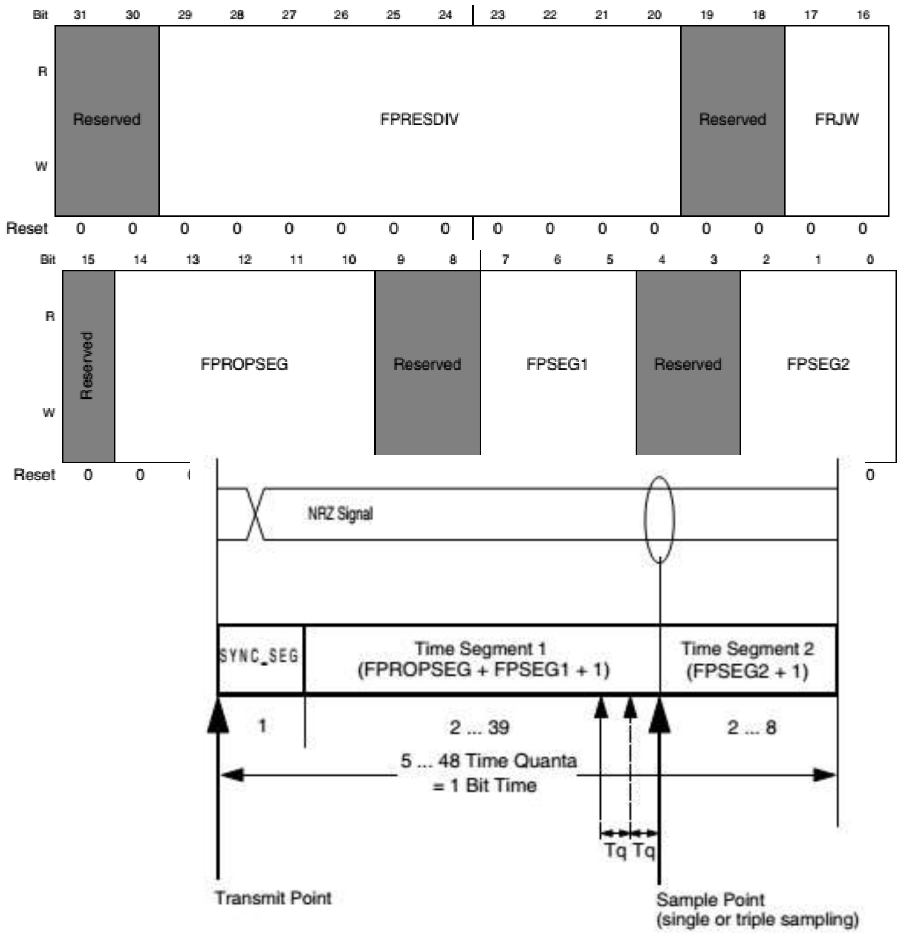
- FD frame requires two bit rates – Arbitration and data phase
- Bit timing parameters are achieved by setting up the following registers:
  - Control register 1 (CTRL1): “Classical” CAN (equivalent FlexCAN2)
  - CAN bit timing reg (CBT): Extended version of above register
  - CAN FD bit timing reg (FDCBT): Bit time associated with FD data phase
- Bit time consists of following elements which are comprised from a variable number of time quanta



FlexCAN FD Bit Timing Examples



CAN_CTRL1



CAN_FDCBT

$$Tq = \frac{(\text{PRESDIV} + 1)}{f_{\text{CANCLK}}}$$

CAN Bit Time = (Number of Time Quanta in 1 bit time) * Tq

$$\text{Bit Rate} = \frac{1}{\text{CAN Bit Time}}$$

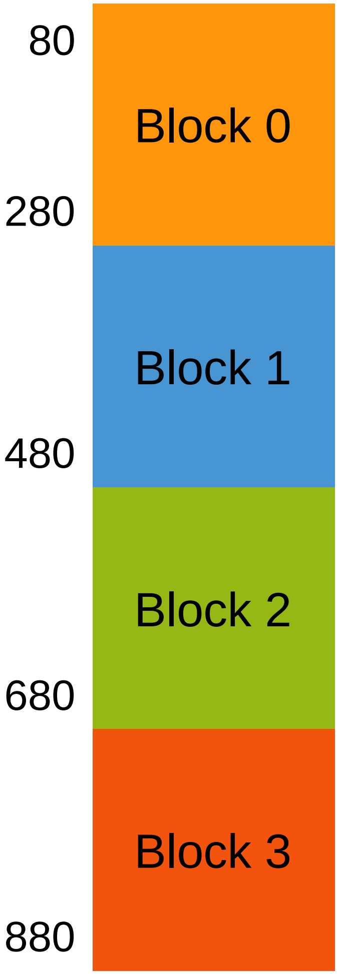
Classic CAN vs CAN FD initializations (FlexCAN examples)

Initialization	Classic CAN registers, memory	CAN FD registers, memory
Clock source selection	CAN_CTRL1, CAN_MCR	CAN_CTRL1, CAN_MCR
CAN bit timing	CAN_CTRL1	Nominal phase: CAN_CBT Data phase: CAN_FDCBT
Transceiver Delay Compensation	-	CAN_FDCTRL
Payload (message buffer data) size	-	CAN_FDCTRL
Message buffers inactivation	RAMn	RAMn (optional larger sizes)
Tx and Rx message buffer configuation	RAMn	RAMn (optional larger sizes)
Enable CRC fix for ISO CAN FD	-	CAN_CTRL2 [STFCNTEN]
Enable CAN FD	-	CAN_MCR
Negate halt state	CAN_MCR	CAN_MCR

Message Buffer Organization

<table>
<tr><th rowspan="2">RAM Index (words, 4B each)</th><th colspan="2">8 Bytes Data (MSBDRx = 0)</th><th colspan="2">16 Bytes Data (MSBDRx = 1)</th><th colspan="2">32 Bytes Data (MSBDRx = 2)</th><th colspan="2">64 Bytes Data (MSBDRx = 3)</th></tr>
<tr><th>MB # (32 max.)</th><th>Frame</th><th>MB # (20 max.)</th><th>Frame</th><th>MB # (11 max.)</th><th>Frame</th><th>MB # (6 max.)</th><th>Frame</th></tr>
<tr><td>0x0</td><td rowspan="2">MB0</td><td>8B Hdr.</td><td rowspan="3">MB0</td><td>8B Hdr.</td><td rowspan="5">MB0</td><td>8B Hdr.</td><td rowspan="9">MB0</td><td>8B Hdr.</td></tr>
<tr><td>0x2</td><td>8B Data</td><td rowspan="2">16B Data</td><td rowspan="4">32B Data</td><td rowspan="8">64B Data</td></tr>
<tr><td>0x4</td><td rowspan="2">MB1</td><td>8B Hdr.</td></tr>
<tr><td>0x8</td><td>8B Data</td><td rowspan="3">MB1</td><td>8B Hdr.</td></tr>
<tr><td>0xA</td><td rowspan="2">MB2</td><td>8B Hdr.</td><td rowspan="2">16B Data</td></tr>
<tr><td>0xC</td><td>8B Data</td><td rowspan="5">MB1</td><td>8B Hdr.</td></tr>
<tr><td>0xE</td><td rowspan="2">MB3</td><td>8B Hdr.</td><td rowspan="3">MB2</td><td>8B Hdr.</td><td rowspan="4">32B Data</td></tr>
<tr><td>0x10</td><td>8B Data</td><td rowspan="2">16B Data</td></tr>
<tr><td>0x12</td><td rowspan="2">MB4</td><td>8B Hdr.</td></tr>
<tr><td>0x14</td><td>8B Data</td><td rowspan="3">MB4</td><td>8B Hdr.</td><td rowspan="5">MB1</td><td>8B Hdr.</td></tr>
<tr><td>0x16</td><td rowspan="2">MB5</td><td>8B Hdr.</td><td rowspan="2">16B Data</td><td rowspan="4">MB2</td><td>8B Hdr.</td><td rowspan="4">64B Data</td></tr>
<tr><td>0x18</td><td>8B Data</td><td rowspan="3">32B Data</td></tr>
<tr><td>0x1A</td><td rowspan="2">MB6</td><td>8B Hdr.</td><td rowspan="2">MB5</td><td>8B Hdr.</td></tr>
<tr><td>0x1C</td><td>8B Data</td><td>16B Data</td></tr>
<tr><td colspan="9">etc.</td></tr>
</table>

FlexCAN3 FD Memory Partitioning



- FlexCAN3 FD memory is partitioned in 4 blocks
  - Essentially 128 x 8-byte payload buffers can be supported
  - SoC can instantiate 1 to 4 blocks
- Each block is composed by the memory space equivalent to 32 x 16-byte MBs or 512-bytes (0x200)
- Each block can be configured to have message buffers (MBs) of a defined size in order to store messages with payload of 8-byte, 16-byte, 32-byte or 64-bytes. Message buffer data size region bit field:
  - **FDCTRL[MBDSRn] = 0bxx**
- The number of message buffers in a block depends upon payload size. Each block can be configured (by software) as follows:

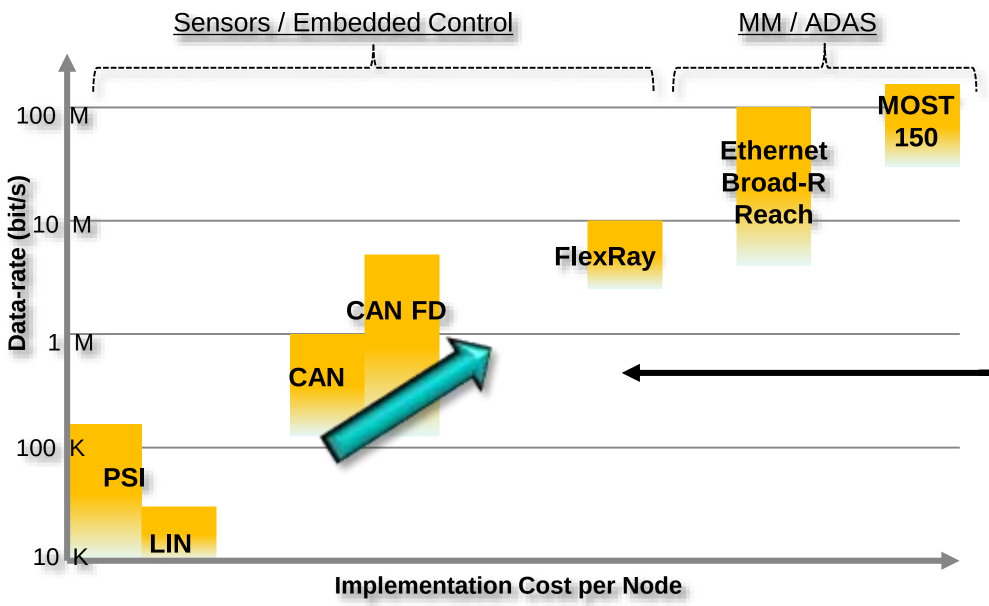
RAM block	Payload size	Number of MBs in the RAM block	Message Buffer range
0	CAN_FDCTRL[MBDSR0]=00, 8 bytes payload	32	0 to 31
1	CAN_FDCTRL[MBDSR1]=01, 16 bytes payload	21	32 to 52
2	CAN_FDCTRL[MBDSR2]=10, 32 bytes payload	12	53 to 64
3	CAN_FDCTRL[MBDSR3]=11, 64 bytes payload	7	65 to 71

Hybrid Networks

Today´s HS CAN Network - Issues

- **Bandwidth for normal communication:**
  Close to the edge
  - New and increasingly complex functions in the car creating more nodes in the network and an increasing amount of data exchange, requiring more bandwidth.
- **Cable Length vs. Bit rate:**
  Limitations due to network topology
  - Large cable length esp. in big vehicles (commercial vehicles, busses, trucks) put a strict limit on the (arbitration) bit rate due to reflections, ringing, jitter and mainly loop delay.
- **Bandwidth for Fast Flashing:**
  Size of flash memory increased rapidly
  - ECU´s need to be flashed as quick as possible, to keep production/standing/service time as short as possible.
  - Current systems run at max practical bit rates of HS-CAN

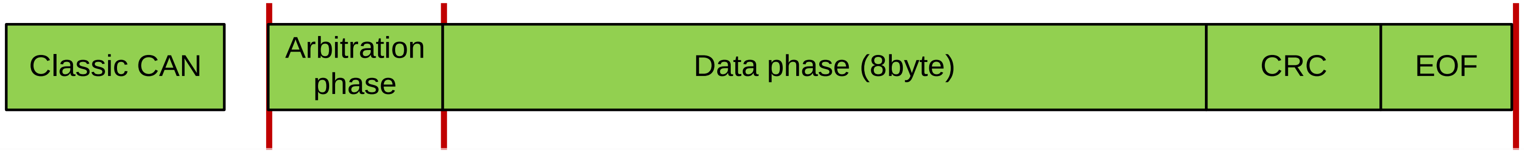
CAN FD – Boosting traditional CAN



- Frames with up to 64 data bytes can be transmitted
- Different bit times in arbitration and data phase
- Average bit rate significant above 1Mbps can be achieved
- Arbitration mechanism unchanged
- Known CAN protocol handling remains the same
- Timing requirements on Acknowledge slot relaxed
- Going to be standardized in ISO11898-1

25

CAN FD – Boosting traditional CAN

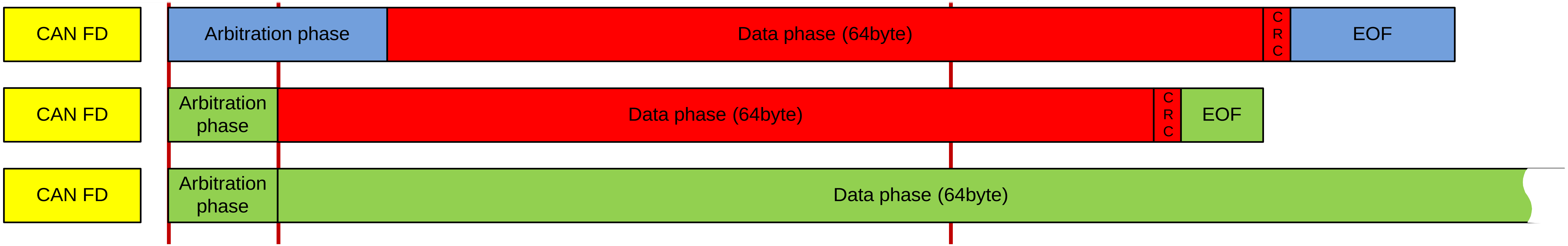


Single bit rate, 8-byte data frame

CAN FD decouples bit rates in arbitration and data phase

Slower arbitration allows for more cable length, faster data phase keeps net bit rate unchanged!

CAN FD allows up to 64 data bytes

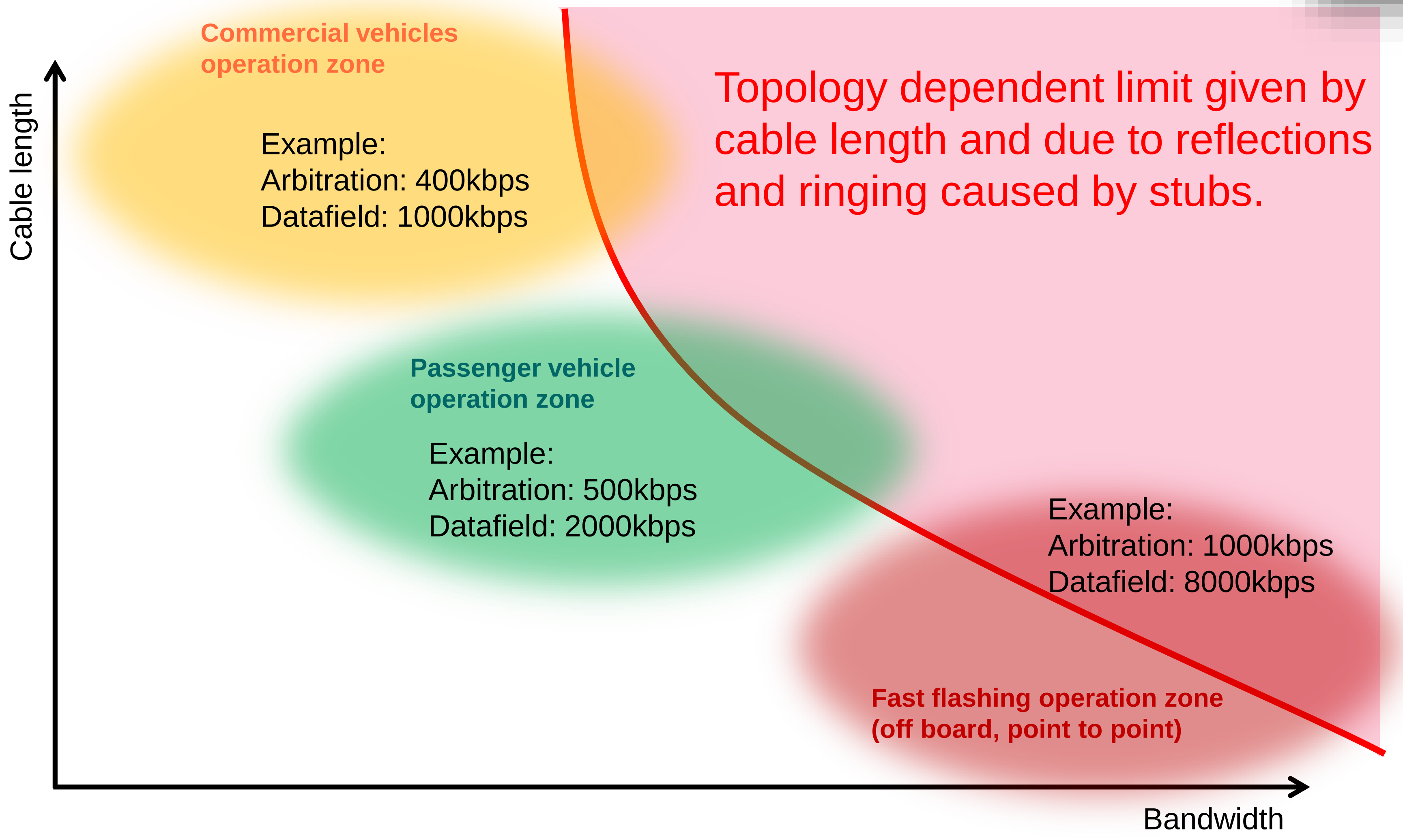


You have the freedom to choose independently:

- Bit rate in data phase higher than in arbitration: YES / NO
- Number of data bytes per frame: 0, 1, 2, 3, 4, 5, 6, 7, 8, 12, 16, 20, 24, 32, 48, 64

„Using CAN FD with 64bytes per frame allows to use the same PDU (Protocol Data Unit) routing strategy in CAN Networks as planned to be used in Ethernet networks"

CAN FD Use Cases



CAN FD Functions: Terminology

Functions being developed:

- **CAN FD “Active”:**
  - Send & Receive CAN FD frames
  - **NXP support: Speed-improved transceivers >1 Mbps**

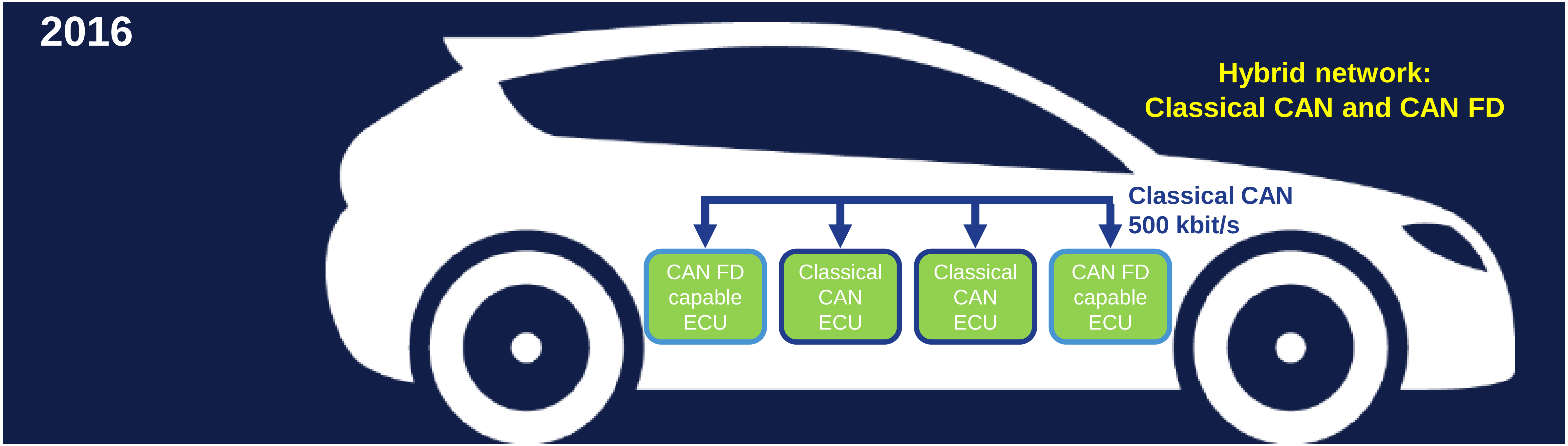
    (proven up to 5 Mbps on some products so far)
- **CAN FD “Passive”:**
  - New CAN Partial Networking transceiver IP, ignores CAN FD frames while in PN Low power mode.
  - Independent of CAN FD data phase speed - arbitration speed remains max 1Mbps
  - **NXP support: Improved Partial Networking decoder tolerates CAN FD frames**

    Examples:
    - **TJA11xxFD/GT** **Allows > 1Mbps active speed, PN decoder ignores CAN FD frames**
    - **TJA11yyFD** **Allows ≤ 1Mbps active speed, PN decoder ignores CAN FD frames**

Other functions being discussed:

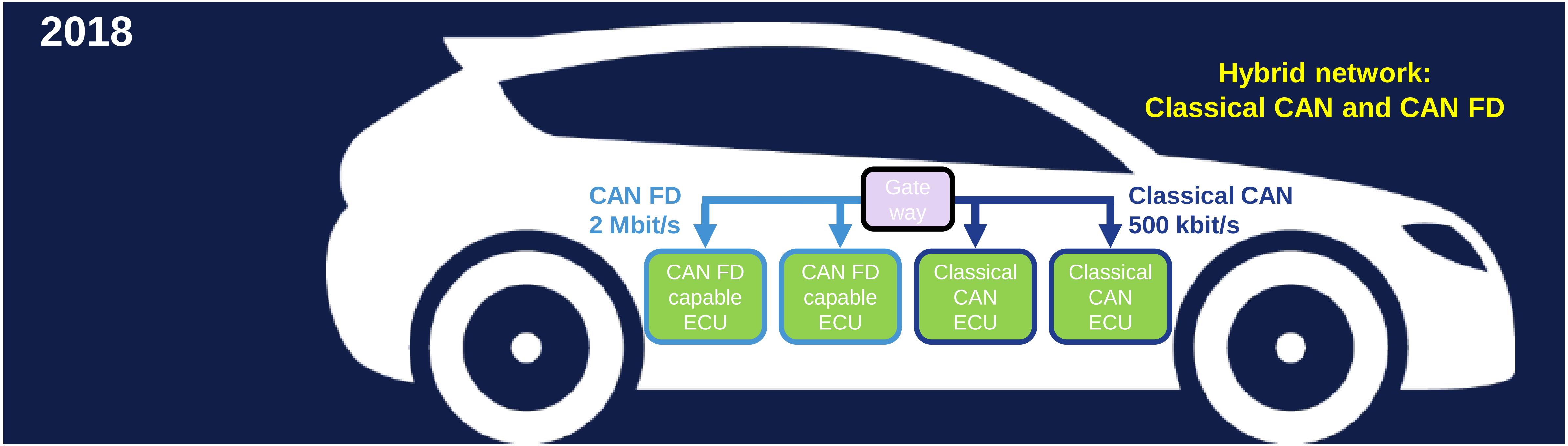
- **CAN FD “Tolerant”:**
  - New CAN-Controller IP in µC being able to accept CAN FD frames without creating errors.

Integrating CAN FD with todays Networks



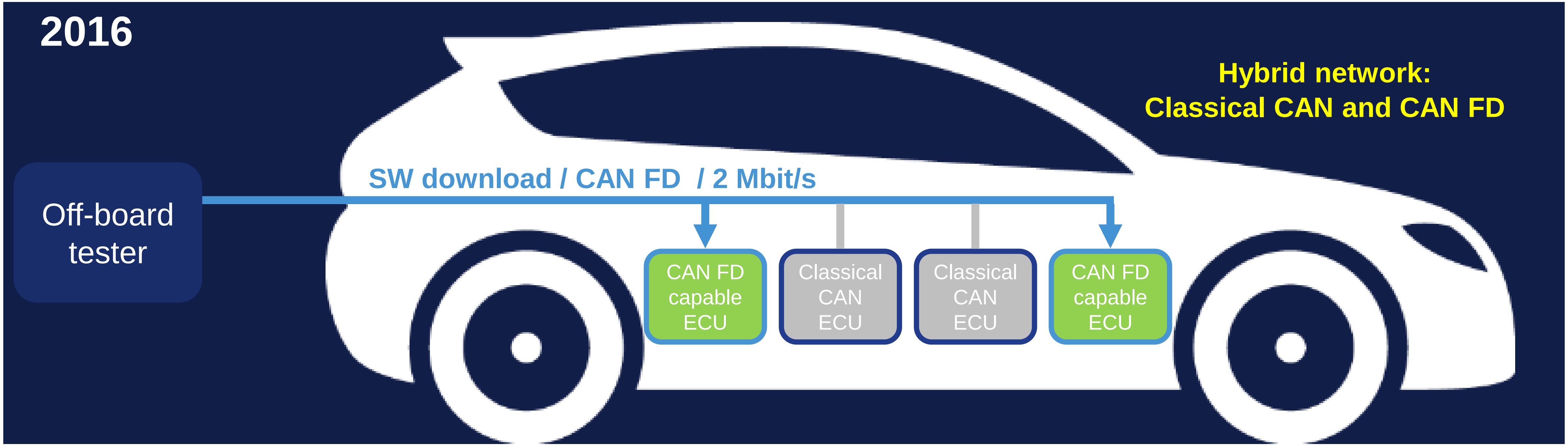
Challenges of Mixing Classical CAN and CAN FD	For some modules the µC family is already available with CAN FD	Modules with 8 and 16 bit µC are often not available with CAN FD	Classical CAN modules destroy CAN FD frames with error frames	At least SW flashing at the end of line should be made available	Classical CAN modules need to be “deaf” when CAN FD frames occur

Hybrid Networks for ECU Flashing



Mixing Classical CAN and CAN FD using a new network layout	One solution is adding a gateway; which is costly	Modules are now grouped by CAN FD capabilities and not by vehicle domains	Major changes in the wire harness required	Inflexible solution when further modules can be changed to CAN FD

Hybrid Networks for ECU Flashing

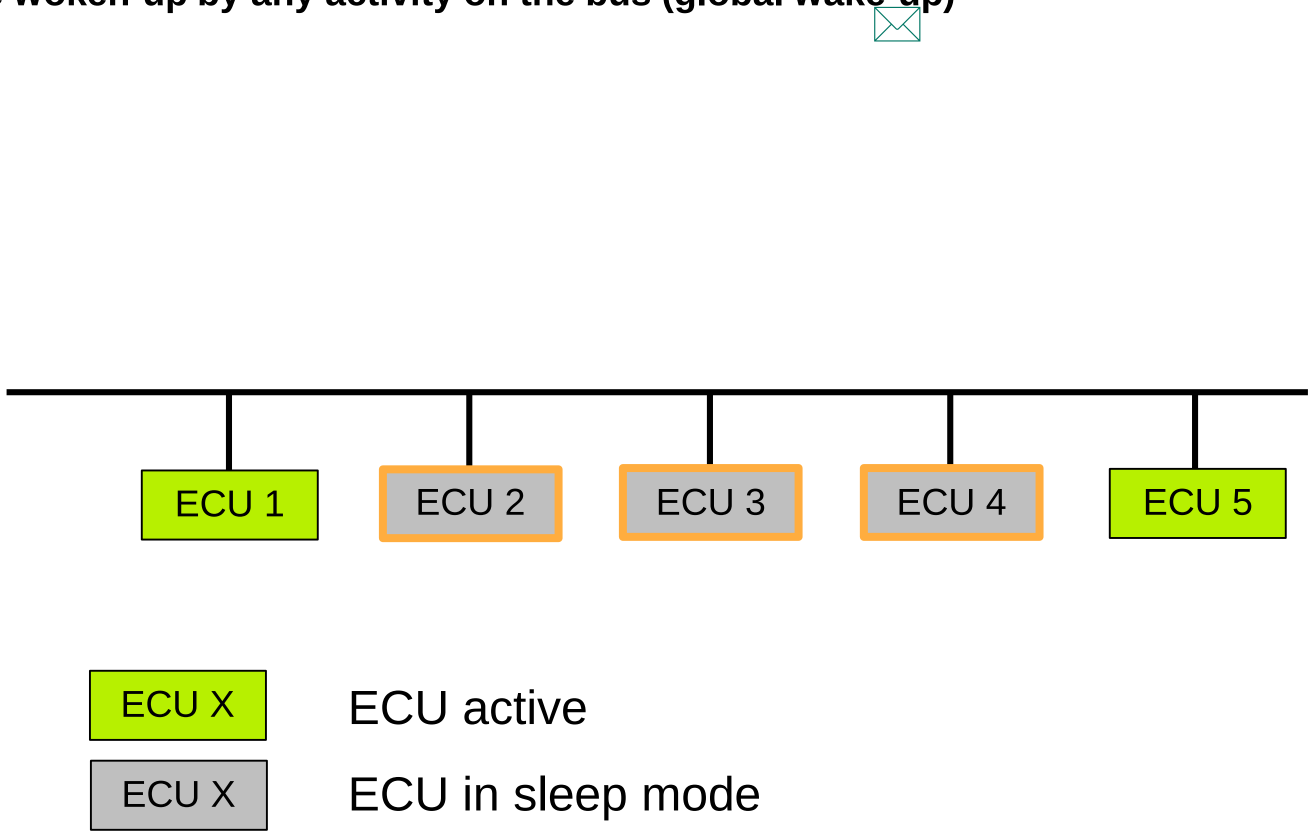


Mixing Classical CAN and CAN FD – managed by partial networking transceivers (FD passive)	Only update ECUs that have a huge demand on SW download as CAN FD MCU become available	ECUs that do not have CAN FD capabilities remain in low-power modes during CAN FD communication	Transceiver wake-up initiated by dedicated Classical CAN frame after flashing is complete	Step-wise introduction of CAN FD	CAN FD communication only during flashing or other special vehicle operation modes

Partial Networking

Technical Principle

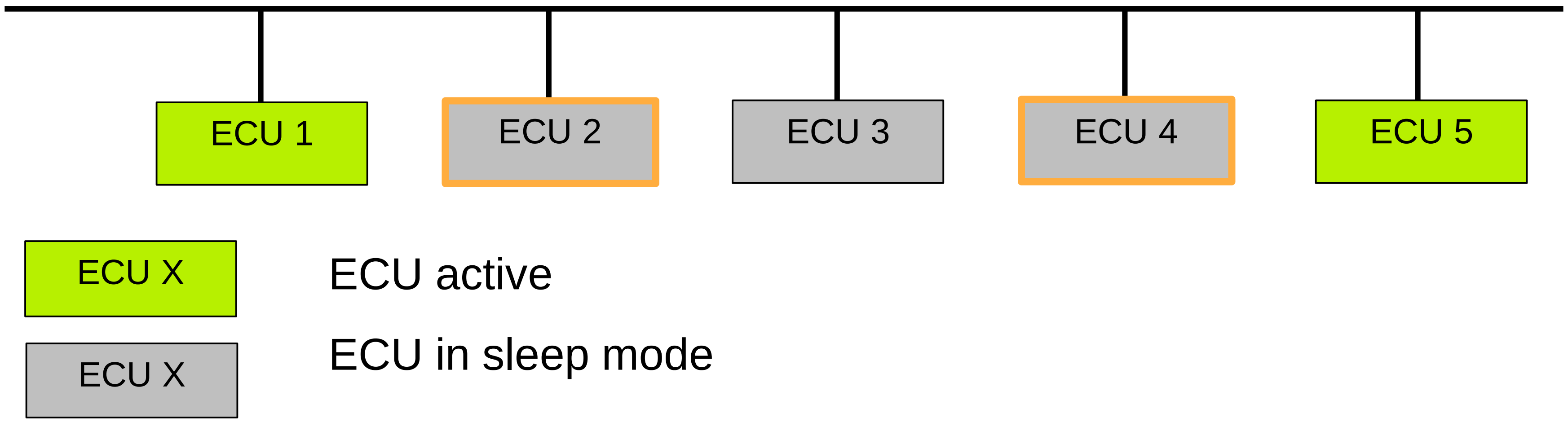
- Wake up behavior according **ISO 11898-5**
  - **All ECUs are woken-up by any activity on the bus (global wake-up)**



Partial Networking

Technical Principle

- Wake up behavior according **ISO 11898-6**
- Partial Networking is the ability to operate certain parts of a network while others remain inactive
  - **Normal Bus traffic <u>does not</u> wake up a partial network enabled ECU**
  - **PN enabled ECU wakes up only when a defined wake-up message is received**

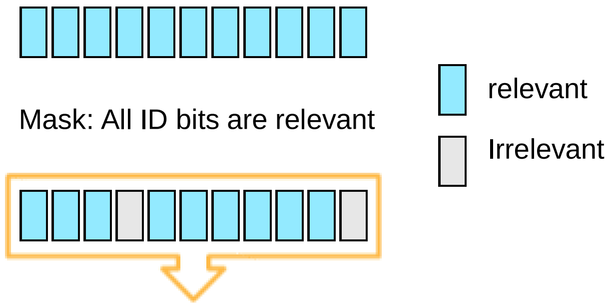


Partial Networking – Wake Up

Identifier

- **Wake Up Message is defined by CAN Identifier and DLC**
- **Identifier:**
  - Use of 11-bit or 29-bit Identifier
  - Single identifier (all 11/29 bits are defined)
  - Group of identifiers (min one bit don´t care)

Example: 11 bit Identifier



Mask: Certain ID bits are irrelevant

Example configuration

11-bit ID

ID Range: 0x1A0 – 0x1A7

Binary: 001 1010 0xxx

This configuration allows up to 8 different nodes
in the system to send wake-up messages

Partial Networking – Wake Up

Data Length Code

- **Data Length Code (DLC)**
  - DLC code 0 … 8: Defines the length of the wake message in Byte‘s
  - Expected Data length and received data length have to be equivalent to identify a valid Wake message

Example configuration

Data length: 1 Byte = 8 Bit

This configuration allows up to 8 different groups of nodes to be woken-up simultaneously with one wake-up message.

8 Bytes of data allow for 64 groups

Partial Networking – Wake Up

Addressing Groups (Data Field)

- **(Data Field)**
  - Is irrelevant if DLC=0
  - Every single bit is used to address a dedicated group of CAN nodes (can address only 1 or up to 64 groups in one message)
  - Individual Nodes can be members of 1 or more groups

Example configuration

Group 1: Node 1, Node 2

Group 2: Node 5

Group 3: Node 1, Node 7, …

1 Group = 1 Use Case !

Partial Networking – Wake Up

Wake-Up process/workflow

- **Configuration node 1:** ID range: 0x1A0 – 0x1A7, DLC=1, Groups 1, 3

Identifier											DLC				Groups							
0	0	1	1	0	1	0	0	X	X	X	0	0	0	1	1	0	1	0	0	0	0	0
															1	2	3	4	5	6	7	8

- **Configuration node 2:** ID range: 0x1A0 – 0x1A7, DLC=1, Groups 1, 4

Identifier											DLC				Groups							
0	0	1	1	0	1	0	0	X	X	X	0	0	0	1	1	0	0	1	0	0	0	0
															1	2	3	4	5	6	7	8

- **Received message:**

Identifier											DLC				Datafield								Datafield							
0	0	1	1	0	1	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0

match

☒ **No valid wake-up message**

Partial Networking – Wake Up

Wake-Up process/workflow

- **Configuration node 1:** ID range: 0x1A0 – 0x1A7, DLC=1, Groups 1, 3

Identifier											DLC				Groups							
0	0	1	1	0	1	0	0	X	X	X	0	0	0	1	1	0	1	0	0	0	0	0
															1	2	3	4	5	6	7	8

- **Configuration node 2:** ID range: 0x1A0 – 0x1A7, DLC=1, Groups 1, 4

Identifier											DLC				Groups							
0	0	1	1	0	1	0	0	X	X	X	0	0	0	1	1	0	0	1	0	0	0	0
															1	2	3	4	5	6	7	8

- **Received message:**

Identifier											DLC				Datafield							
0	0	1	1	0	1	0	0	0	1	0	0	0	0	1	0	1	1	1	1	0	0	0

End of ID and control field:

☑ **Complete match ⇨ Valid wake-up message !**

Partial Networking – Wake Up

Wake-Up process/workflow

- **Configuration node 1:** ID range: 0x1A0 – 0x1A7, DLC=1, Groups 1, 3

Identifier											DLC				Groups							
0	0	1	1	0	1	0	0	X	X	X	0	0	0	1	1	0	1	0	0	0	0	0
															1	2	3	4	5	6	7	8

Wake-up

- **Configuration node 2:** ID range: 0x1A0 – 0x1A7, DLC=1, Groups 1, 4

Identifier											DLC				Groups							
0	0	1	1	0	1	0	0	X	X	X	0	0	0	1	1	0	0	1	0	0	0	0
															1	2	3	4	5	6	7	8

Wake-up

- **Received message:**

Identifier											DLC				Datafield							
0	0	1	1	0	1	0	0	0	1	0	0	0	0	1	0	1	1	1	1	0	0	0

Wake-up actually happens when the CRC has been received correctly.

End of ID and control field:

☑ **Complete match ➩ Valid wake-up message !**

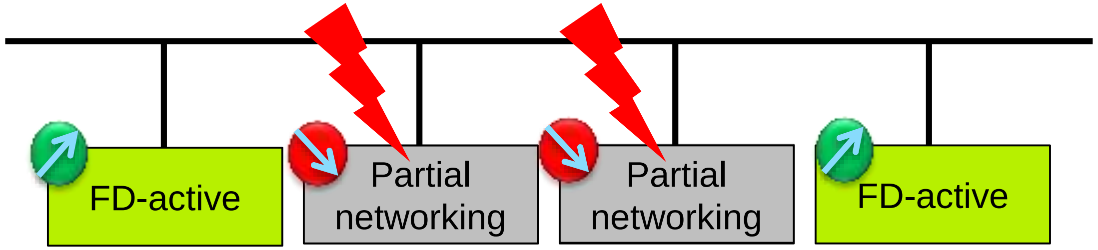
FD Passive: Enhancing Partial Networking for hybrid networks

- Partial networking functionality according to ISO11898-6
  - **CAN FD frames will be seen as coding errors** and thus TJA1145 will wake-up with error counter overflow when in partial networking sleep mode

- **FD Passive** offers additionally the option to remain in partial networking sleep mode, when CAN FD frames occur on the bus
  - **CAN FD frames are recognized by their recessive FDF bit and judged as 'valid CAN frames'** regardless of what comes after the FDF bit

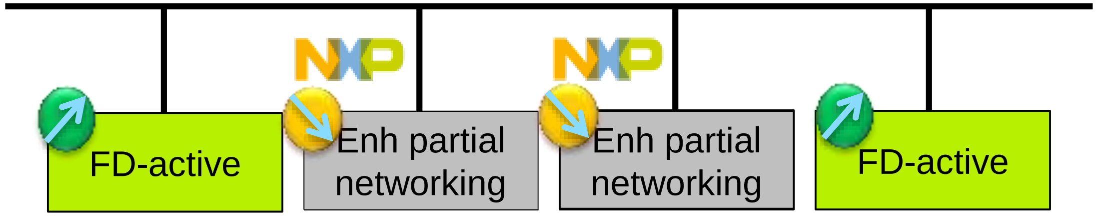
Partial Networking today

Wake-up due to CAN FD traffic → CAN Error Frames

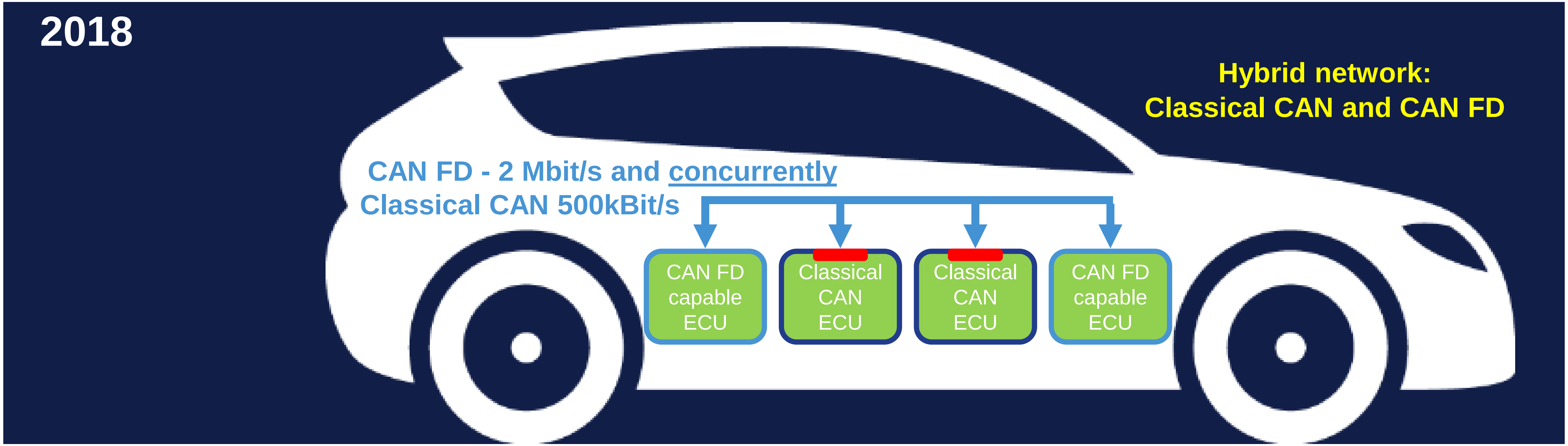


FD-passive Partial Networking

No Wake-up due to CAN FD traffic

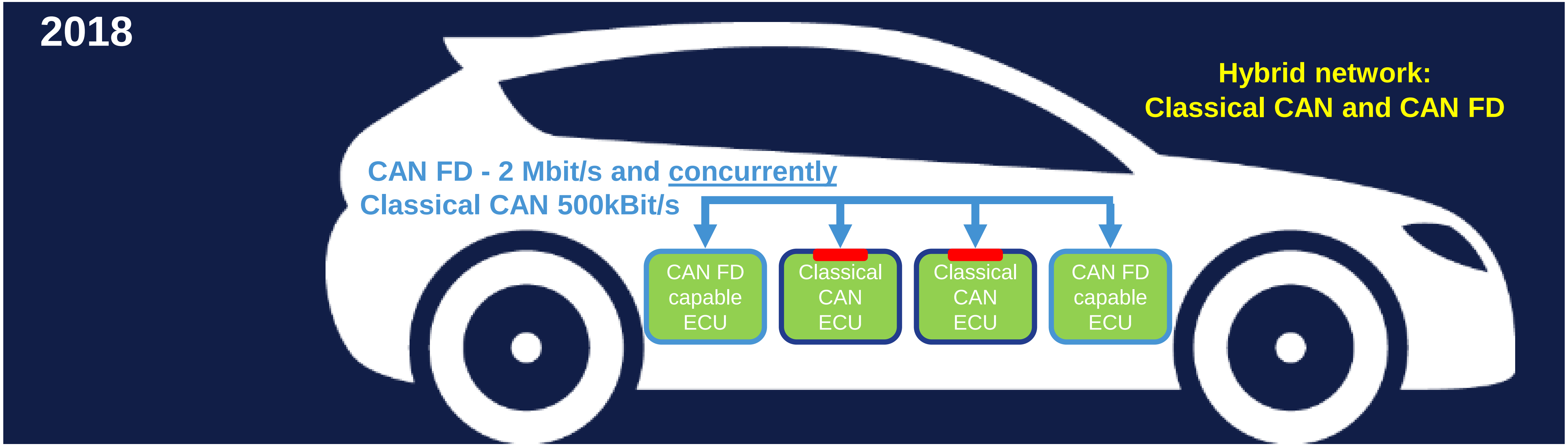


Hybrid Networks for Full Operation



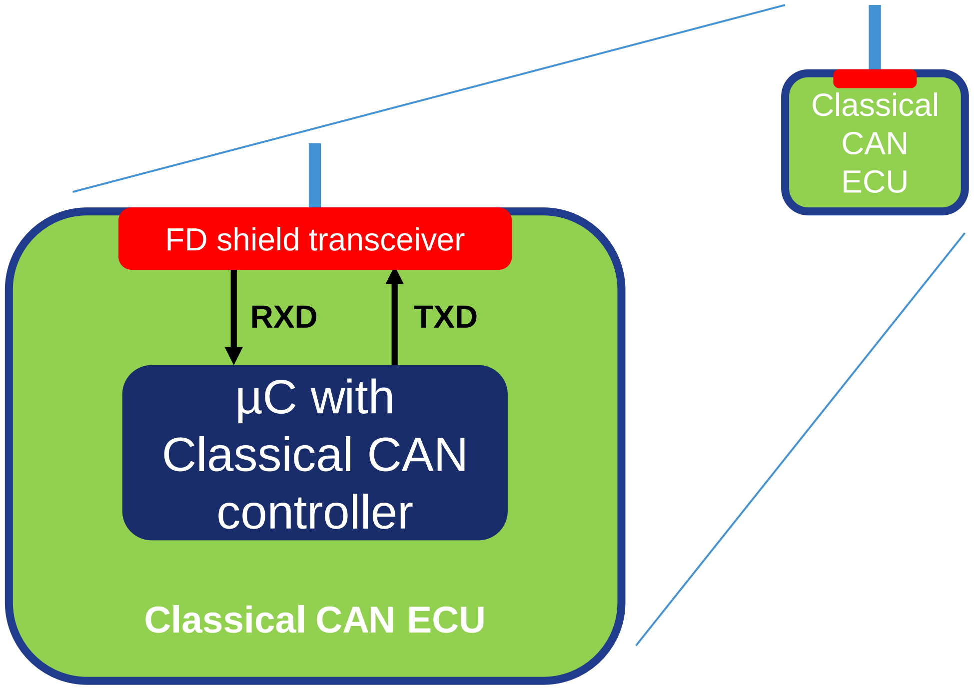
Mixing Classical CAN and CAN FD using FD shield	Classical CAN frames shall have the possibility to equally arbitrate against CAN FD frames	CAN FD frames will not be destroyed by error frames from Classical CAN controllers	A "FD shield" transceiver keeps CAN FD frames away from Classical CAN controllers

Hybrid Networks for Full Operation



Mixing Classical CAN and CAN FD – FD shield	Autosar compatible – no additional SW efforts	Transceiver drop in replacement for standard transceivers	Data consistency shall be guaranteed – no frames lost	Error management shall work as defined in ISO11898-1	The combination of a Classical CAN controller and FD shield shall behave like a “CAN FD tolerant node” as defined in ISO11898-1

FD Shield Implementation



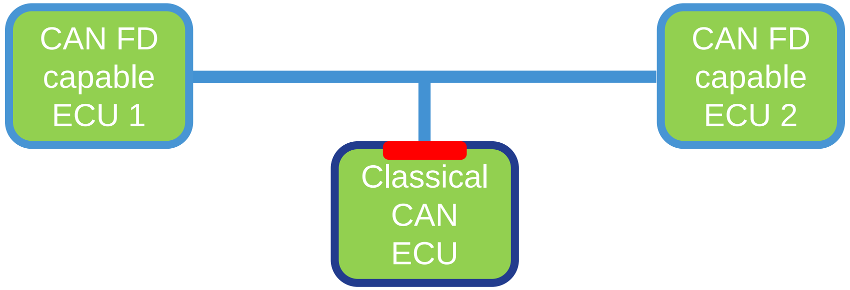
Standard transceiver in SO8 or SO14 replaced by FD shield transceiver

No change of µC

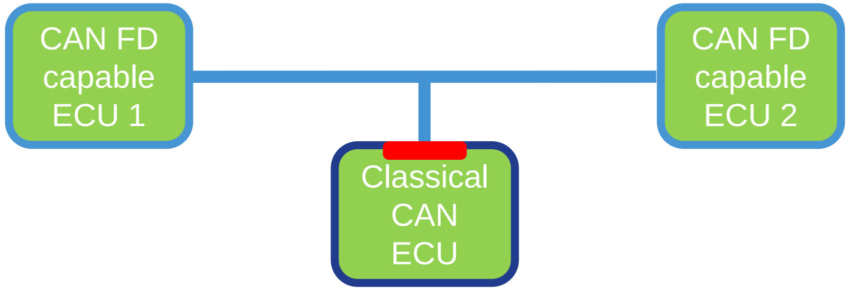
No change of printed circuit board

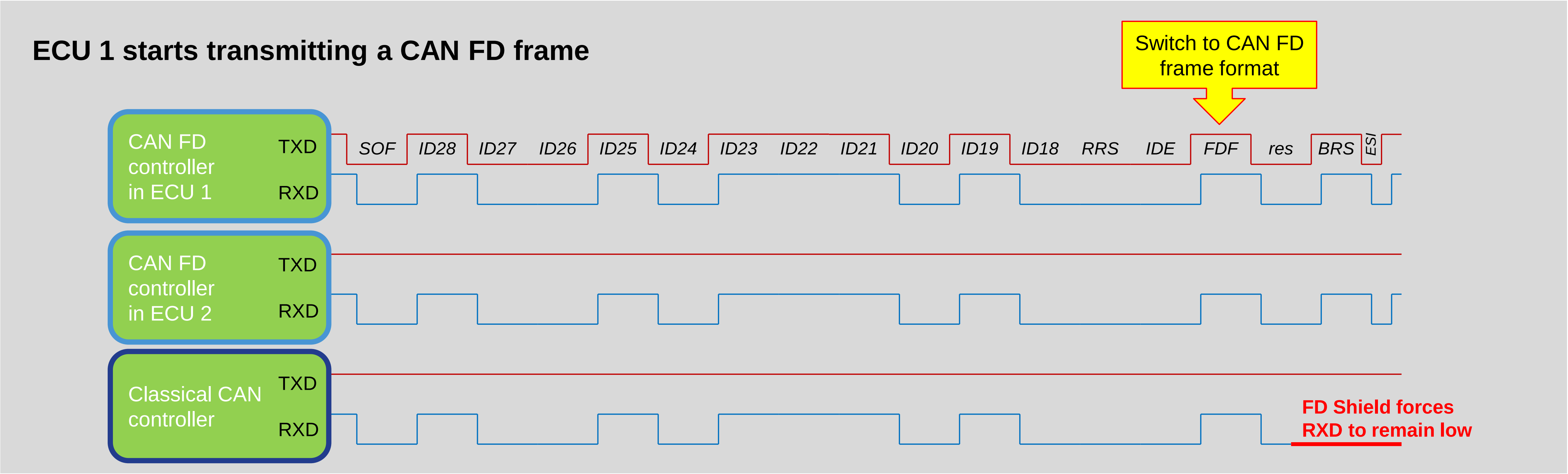
No change of software

FD Shield Operation

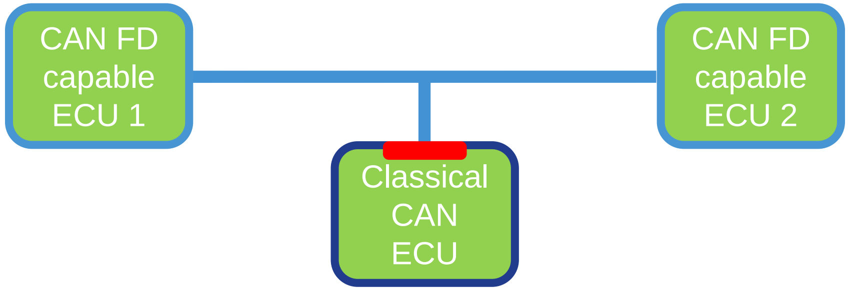


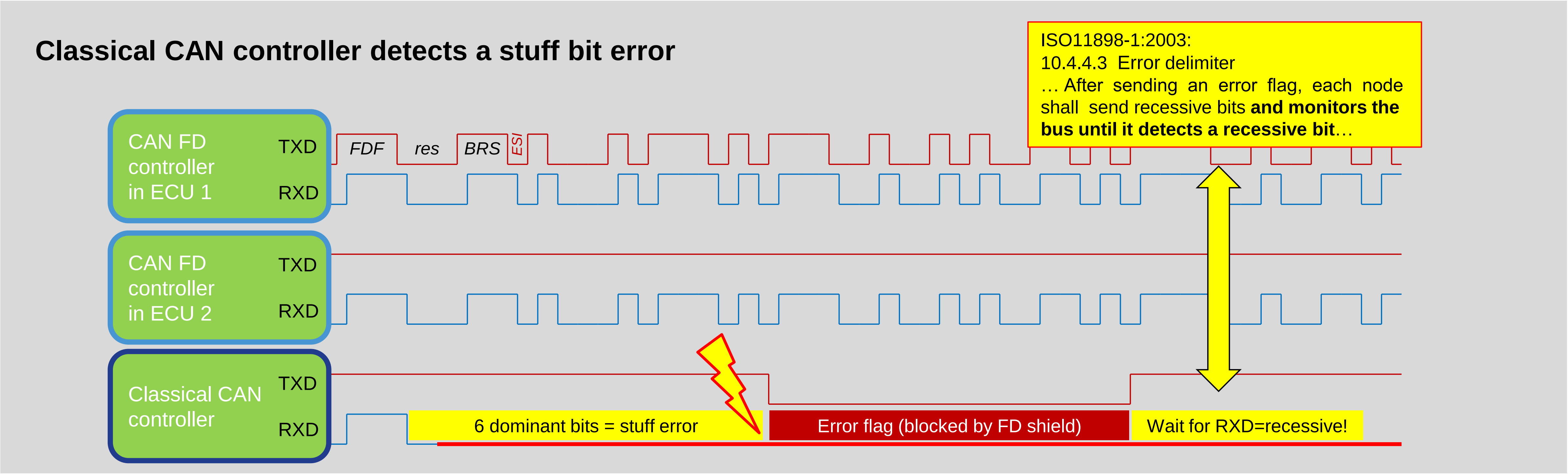
FD Shield Operation



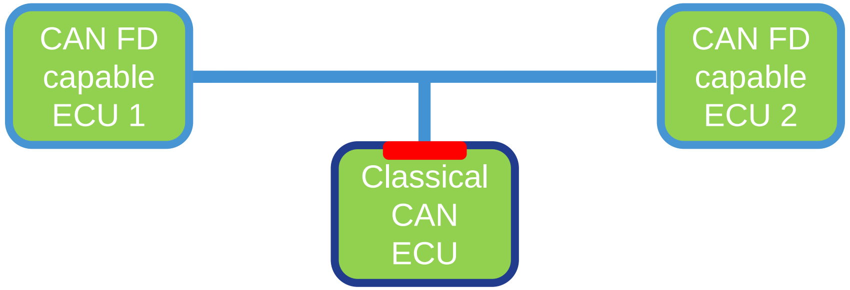


FD Shield Operation



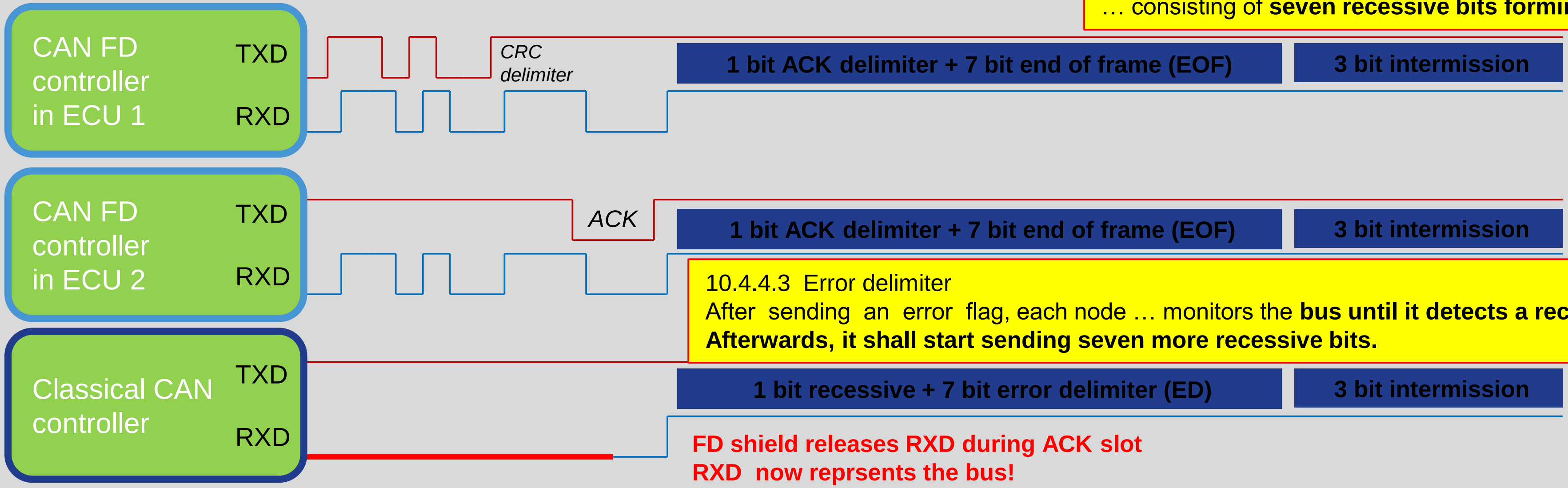


FD Shield Operation



All controllers are re-synchronize at “intermission”

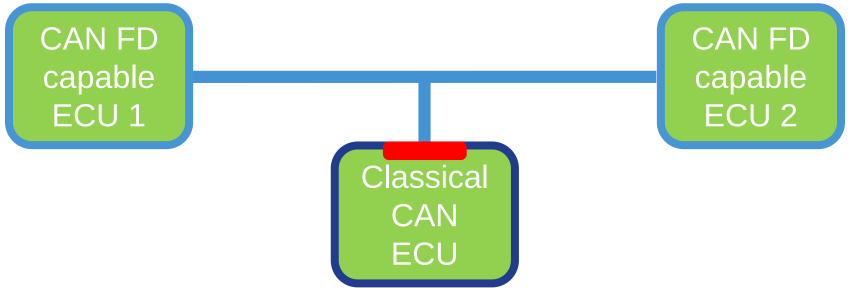
ISO11898-1:2003:
10.4.2.7 ACK field
…**The ACK delimiter, …, shall be a recessive bit.** …
10.4.2.8 EOF
… consisting of **seven recessive bits forming the EOF.**



10.4.4.3 Error delimiter
After sending an error flag, each node … monitors the **bus until it detects a recessive bit. Afterwards, it shall start sending seven more recessive bits.**

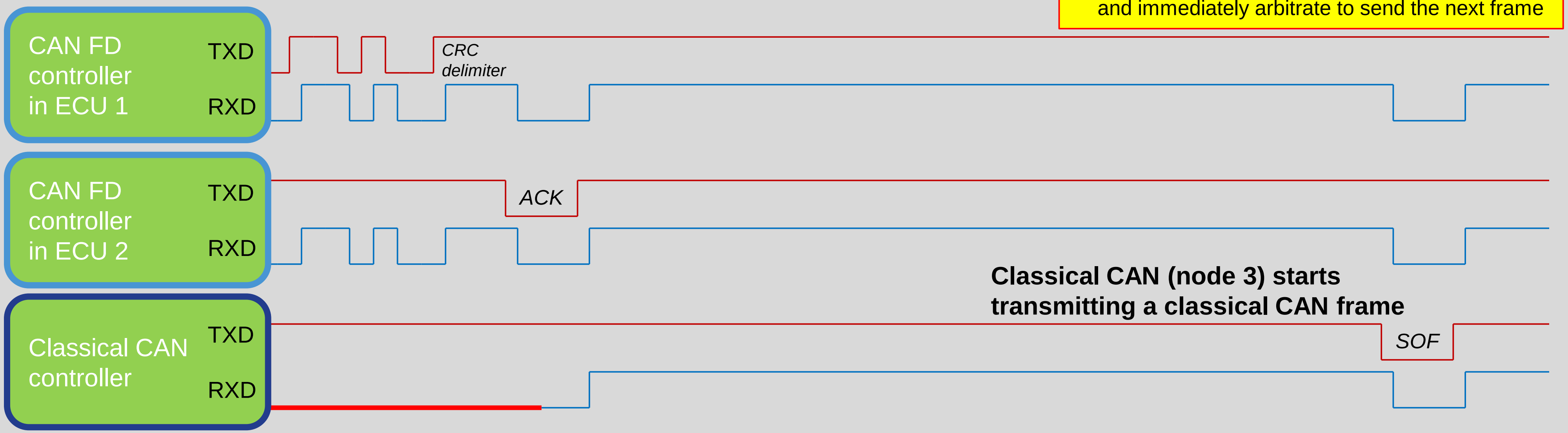
FD shield releases RXD during ACK slot
RXD now reprsents the bus!

FD Shield Operation



Equal arbitration between Classical CAN and CAN FD is possible

All nodes remain constantly synchronized
No interruption between messages
No complicated bit synchronization
Classical CAN and CAN FD ECUs can equally and immediately arbitrate to send the next frame

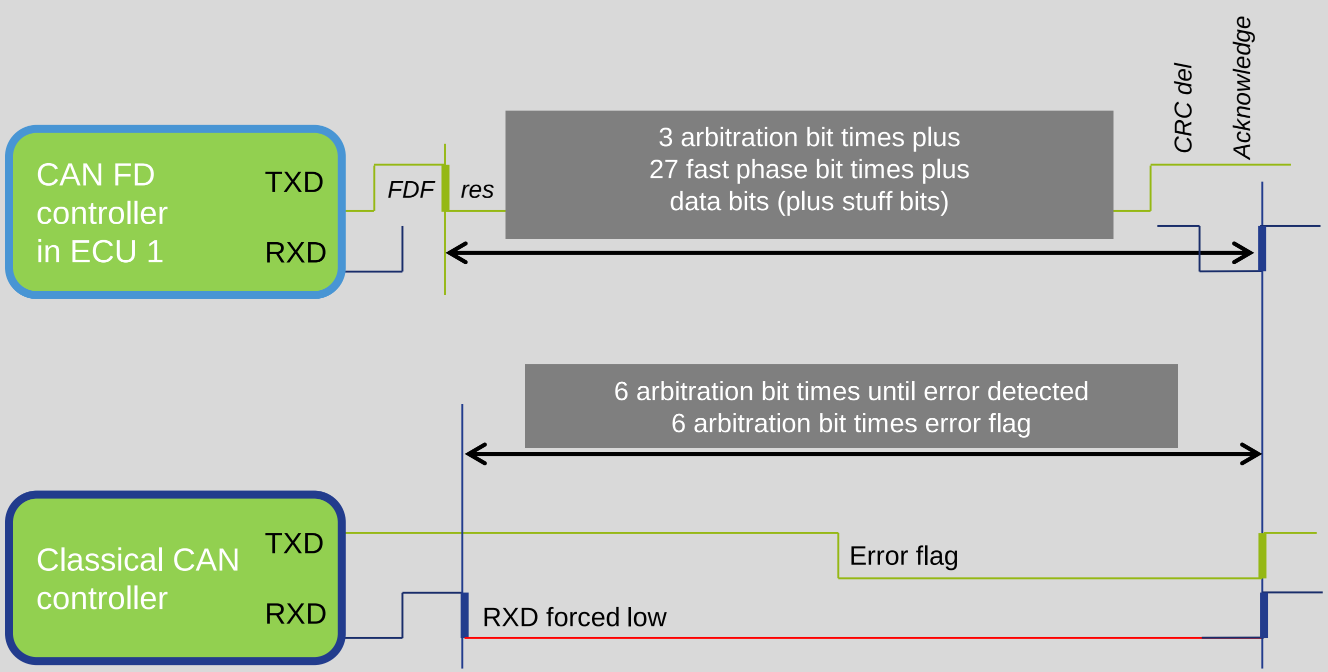


FD Shield Operation – Constraints

The CAN FD frame needs to be long enough to allow the Classical CAN node to complete its error handling

The Error flag shall not end earlier than the Acknowledge

Minimum number of data bytes > (fast phase bit rate / arbitration bit rate) - 3



With 500 kBit/s in arbitration phase

and 2000 kBit/s in fast phase:

Min {number of data bytes} > (2000/500) - 3

Min {number of data bytes} = 2

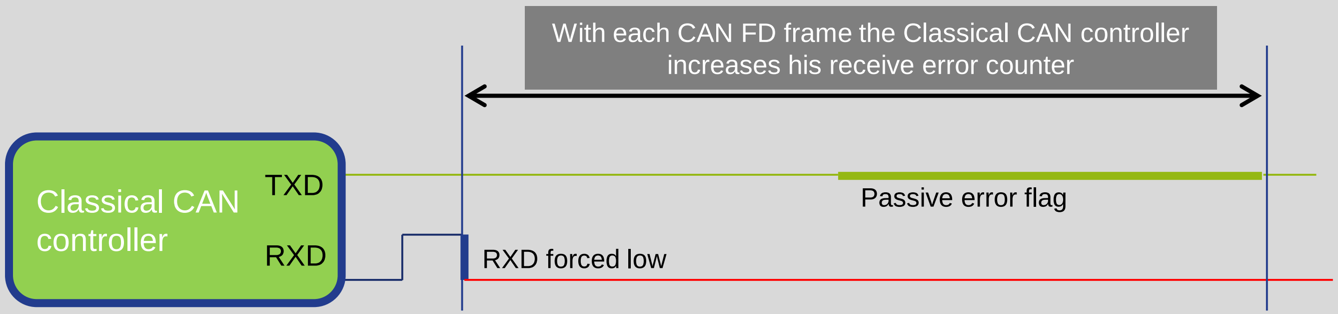
FD Shield Operation – Error Management

The Classical CAN controller will toggle between being error passive and error active, depending on the ratio of CAN FD to Classical CAN frames

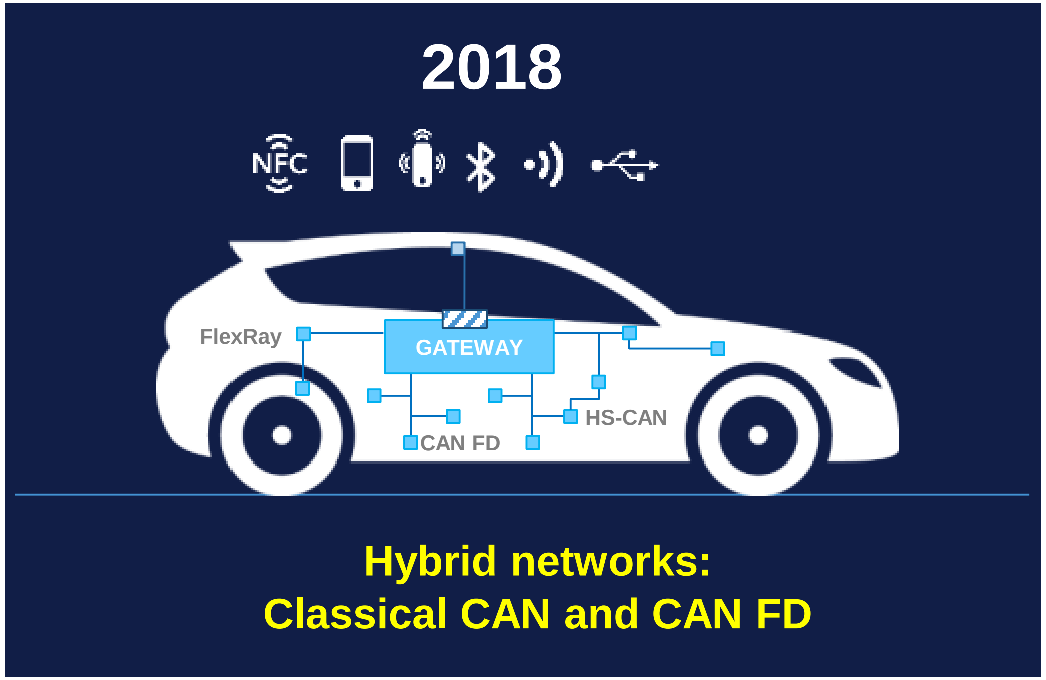
In case the Classical CAN controller has problems receiving a Classical CAN frame while being error passive it cannot enforce the repetition of that Classical CAN frame. Therefore FD shield takes over the error management!

FD shield detects: No active error frame sent and also no acknowledge given

-> Classical CAN frame not received & controller is error passive
-> Active error frame will be issued by FD shield!



Summary and Key Messages



Enabling Networks with Classical CAN and CAN FD ECUs to co-exist can save significant amounts of work/cost and ease the transition to higher bandwidth in vehicles

Hybrid networks provide more elegant solutions considering scalability, security and investments

FD Shield enables this co-existence, as a drop-in replacement CAN transceiver, making Classical CAN ECUs “CAN FD tolerant”, without any further hardware or software changes

SECURE CONNECTIONS
FOR A SMARTER WORLD