NXP | FTF 2016 TECHNOLOGY FORUM

CAN FLEXIBLE DATA (FD)

REBECA DELGADO / BERND ELEND
FIELD APPLICATIONS ENGINEER / PRINCIPAL ENGINEER

FTF-AUT-N1784

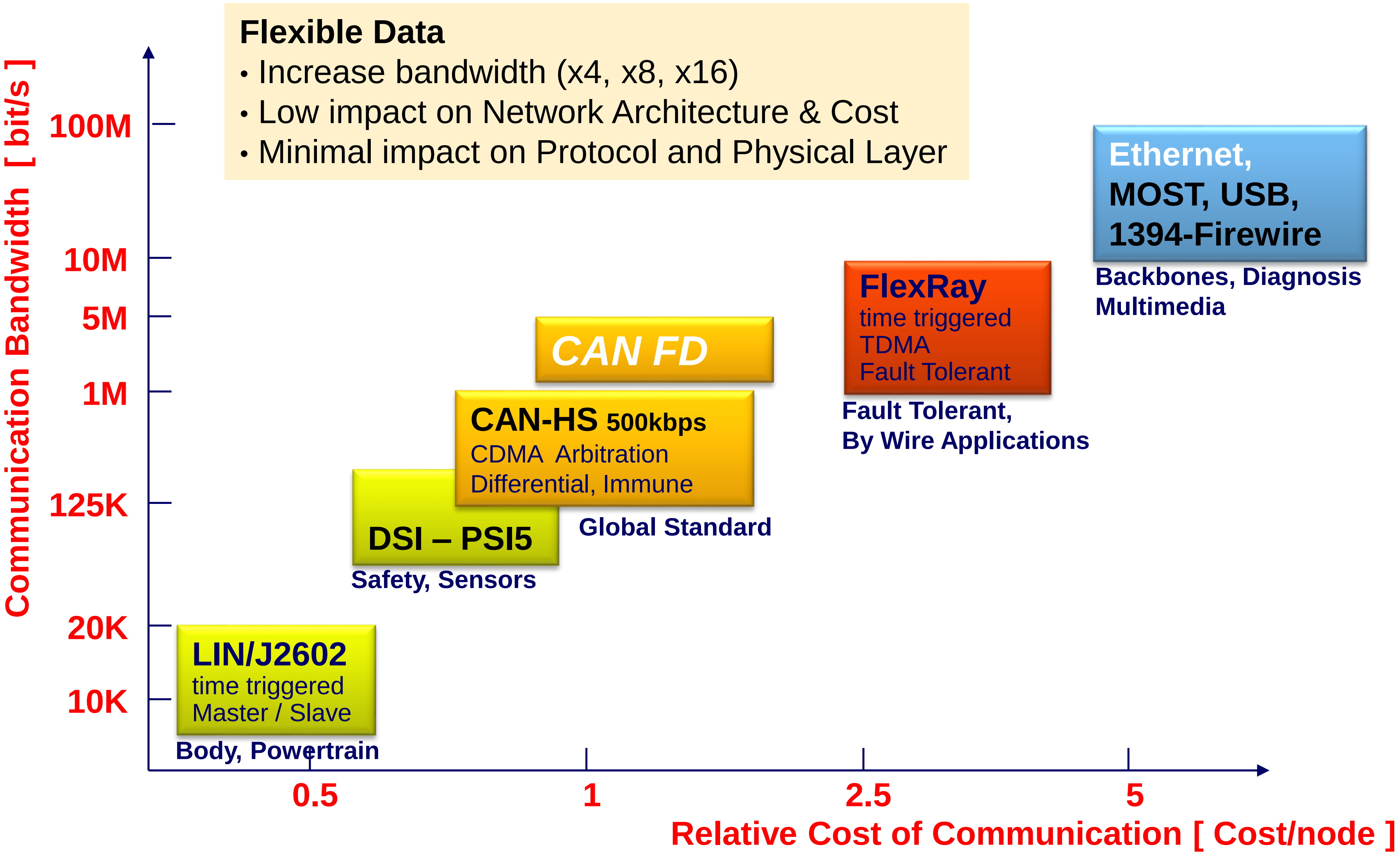
MAY, 2016

AGENDA

- CAN FD Protocol Overview
- FlexCAN3 FD
- MCAN FD
- Flexible Data and Partial Networking Impact on CAN Physical Layer

CAN FD

Increasing Bandwidth at Limited Cost

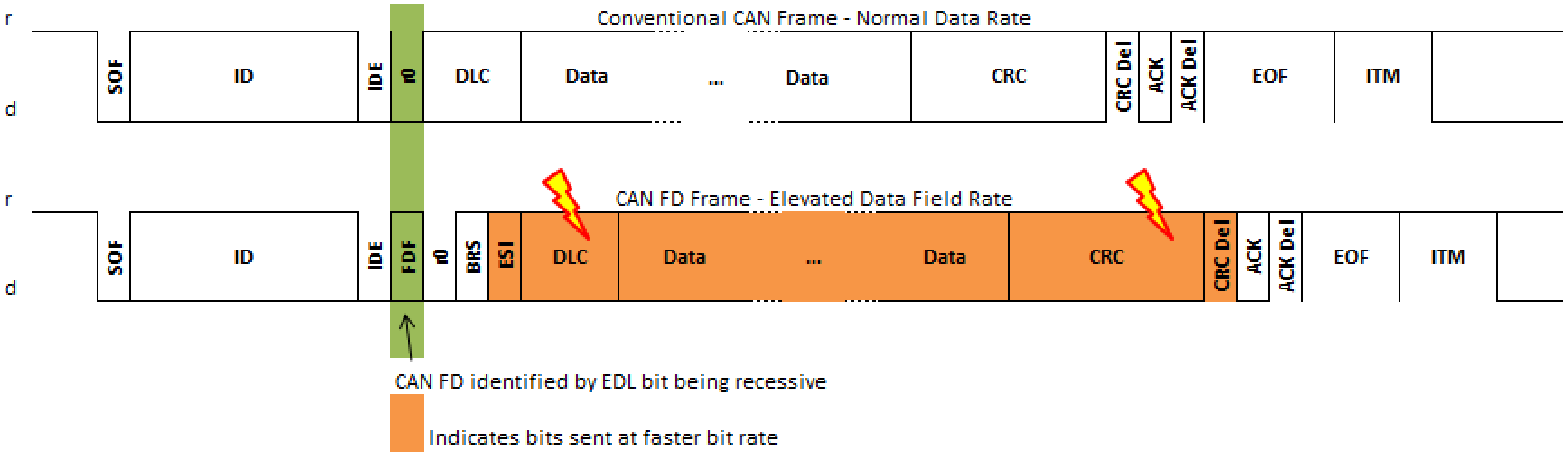


Flexible Data

- Increase bandwidth (x4, x8, x16)
- Low impact on Network Architecture & Cost
- Minimal impact on Protocol and Physical Layer

CAN FD

- 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 allow for existing physical layers to be used



CAN FD identified by EDL bit being recessive

🟧 Indicates bits sent at faster bit rate

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
    (-> details will be disclosed in “FTF-AUT-N1775 Enabling Hybrid CAN Network: FD shield”)

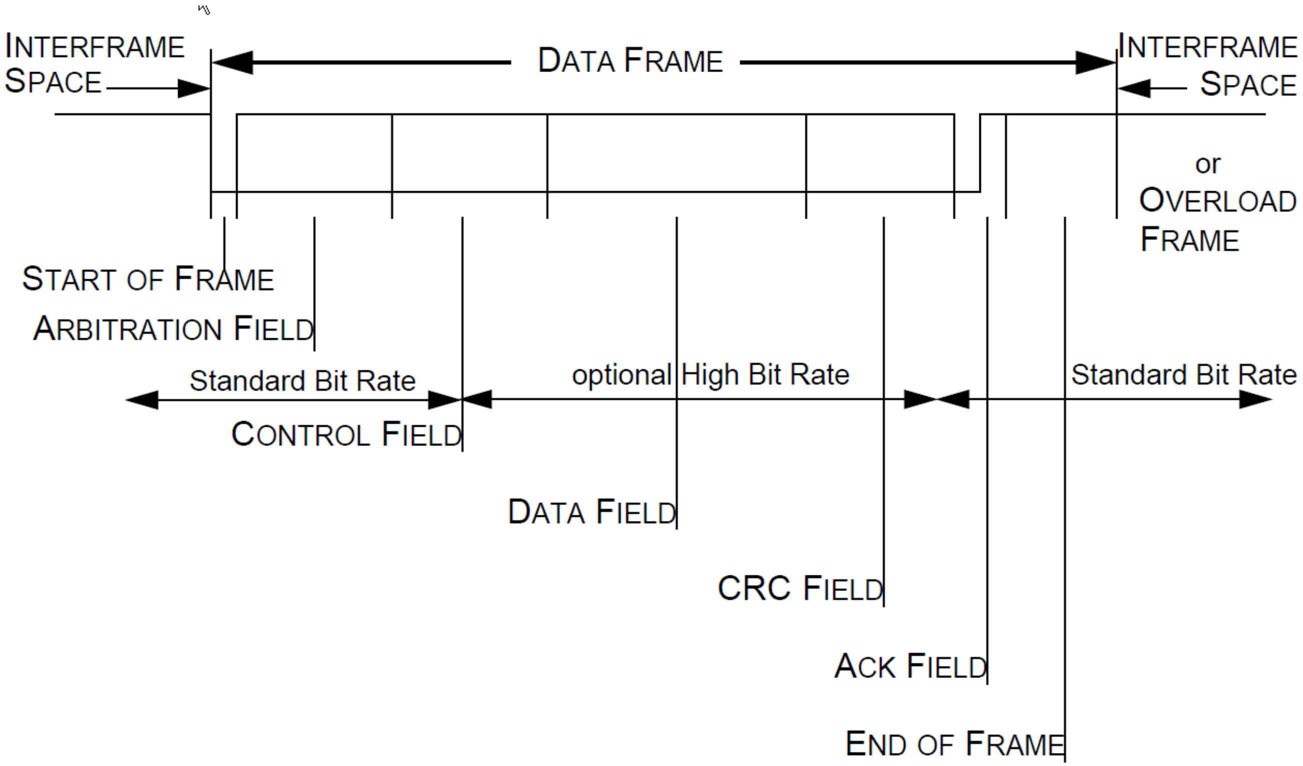
Basic Concepts

- Bit rate
  - Two bit rates in a CAN FD system,
    ne for the arbitration phase and one for the data phase.

- Frame format:

– CAN BASE FRAME FORMAT (CBFF):	11 bit long identifier and constant bit rate
– CAN EXTENDED FRAME FORMAT (CEFF):	29 bit long identifier and constant bit rate
– CAN FD BASE FRAME FORMAT (FBFF):	11 bit long identifier and optional dual bit rate
– CAN FD EXTENDED FRAME FORMAT (FEFF):	29 bit long identifier and optional dual bit rate

Frame Format

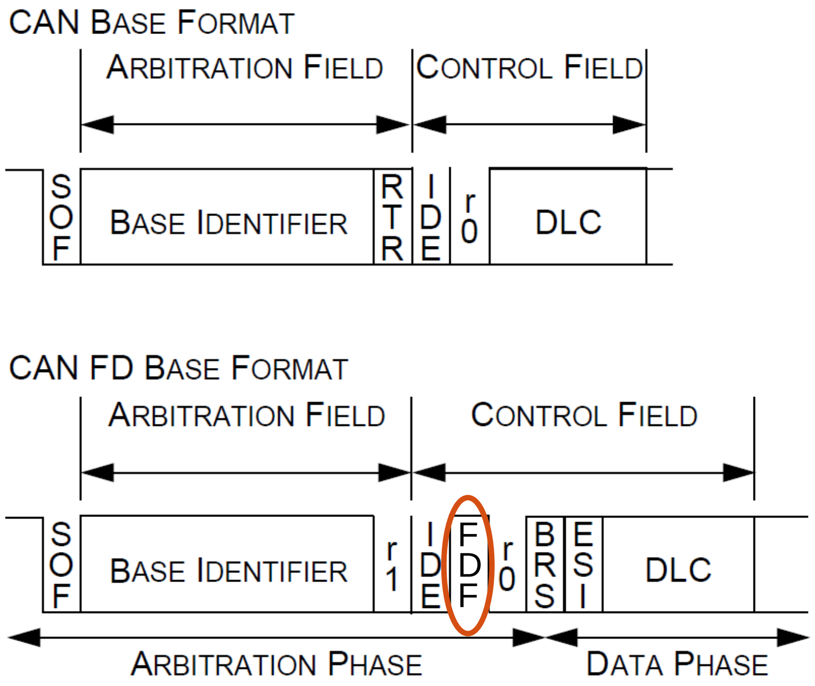


Frame fields

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

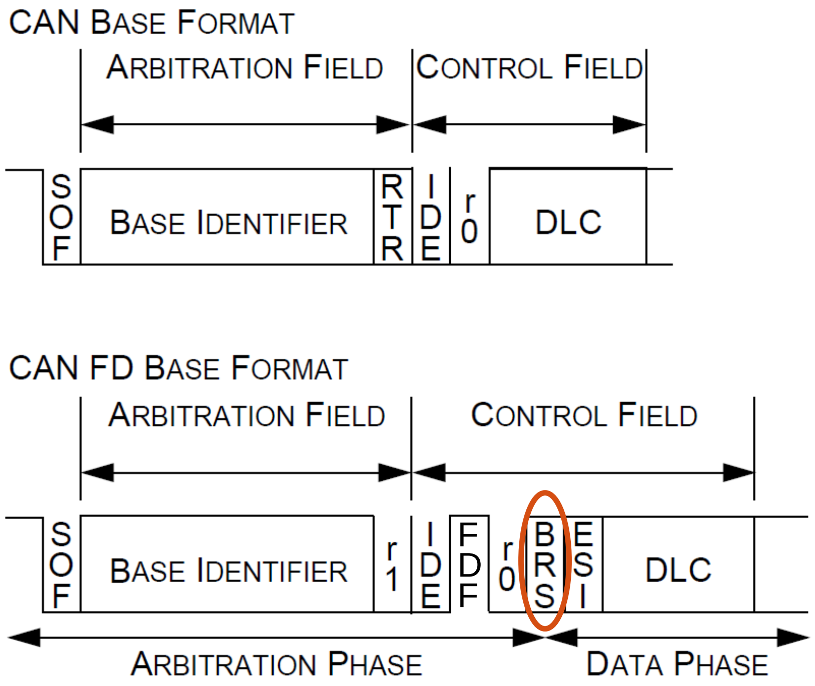
Frame Format

- 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



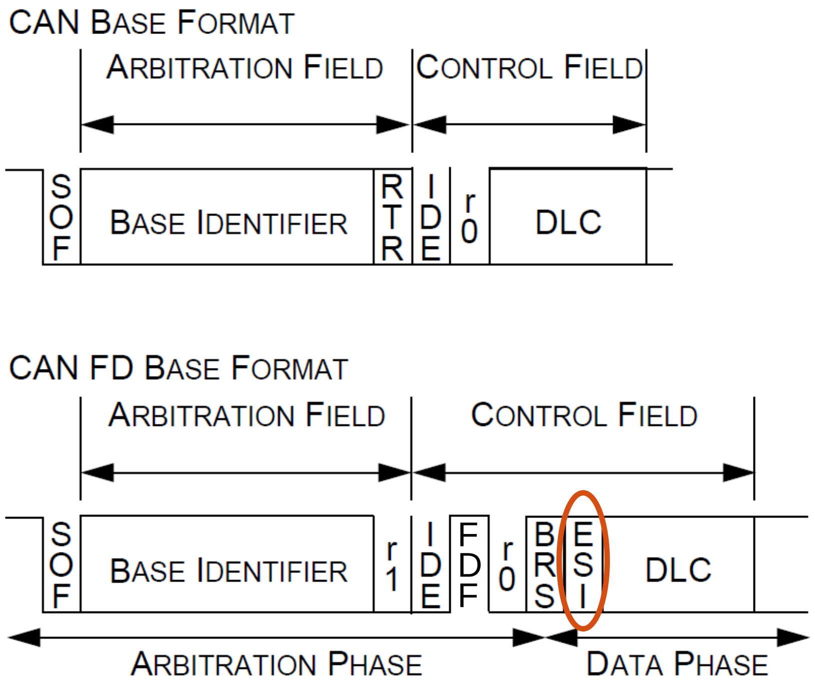
Frame Format

- 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

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

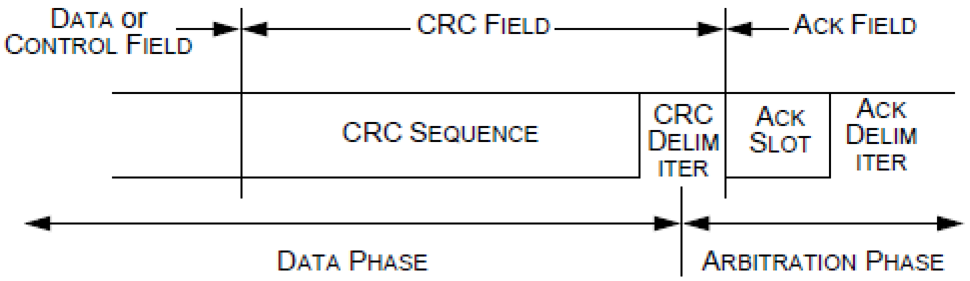


Data Payload

- CAN:
  - 0 to 8 bytes
- CAN FD
  - 0 to 64 bytes

	# of Data Bytes	DLC3	DLC2	DLC1	DLC0
DLC in CAN Format 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	x	x	x
DLC 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

Frame Format – CRC



- CAN: CRC_15 0xC599

$$(x^{15}+x^{14}+x^{10}+x^{8}+x^{7}+x^{4}+x^{3}+1) = (x+1) \cdot (x^{7}+x^{3}+1) \cdot (x^{7}+x^{3}+x^{2}+x+1)$$

- CAN FD < 16 bytes of data: CRC_17 0x3685B

$$(x^{17}+x^{16}+x^{14}+x^{13}+x^{11}+x^{6}+x^{4}+x^{3}+x^{1}+1)$$
$$= (x+1) \cdot (x^{16}+x^{13}+x^{10}+x^{9}+x^{8}+x^{7}+x^{6}+x^{3}+1)$$

- CAN FD > 16 bytes of data: CRC_21 0x302899

$$(x^{21}+x^{20}+x^{13}+x^{11}+x^{7}+x^{4}+x^{3}+1)$$
$$= (x+1) \cdot (x^{10}+x^{3}+1) \cdot (x^{10}+x^{3}+x^{2}+x^{1}+1)$$

- All CRCs with Hamming Distance of HD = 6.

CAN FD CRC Issue

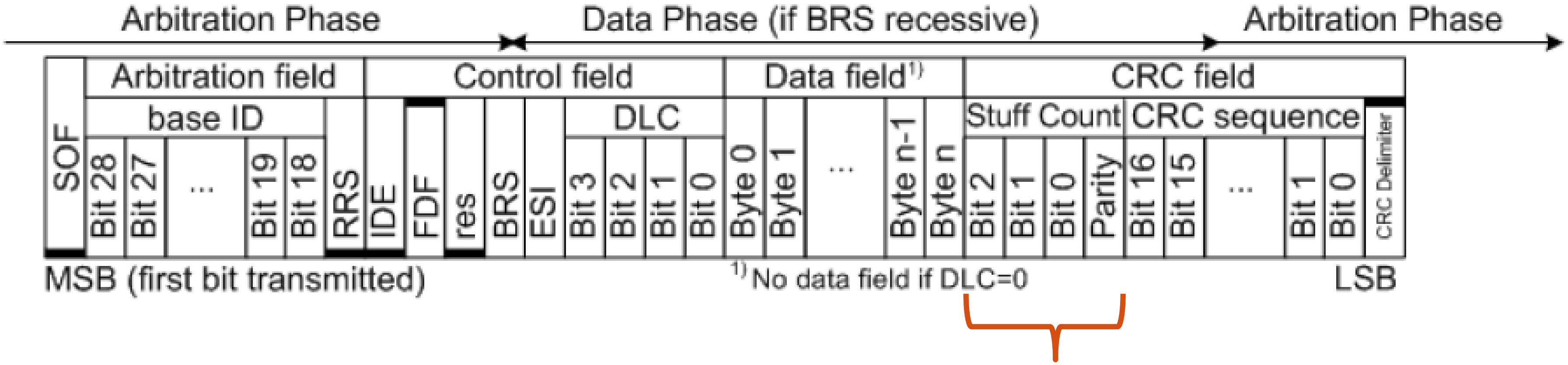
- When Bosch invented CAN, the CRC calculation did not consider bit stuffing.
- The initialization vector was chosen to be “0” (zero)

- Later it was recognized that the bit stuffing impacts the Hamming distance
- In Classical CAN : HD=2 and not HD=6

- With CAN FD Bosch included the stuff bits in the CRC calculation to overcome this problem
- However, in conjunction with the initialization vector of “0” (zero) it became worse!

CAN FD CRC Fixes

- **Fix1:**
- Change CRC initial value to use “1” followed by “0”s as the CRC polynomial seed
- **Fix2:**
- Add stuff bit 3-bit count , gray coded with additional parity bit at LSB, in front of CRC



[1] No data field if DLC=0

4 added bits: 3-bit Modulo-8 stuff bit counter + 1 parity bit calculated over the previous 3 bits

Implications of CAN FD Fixes

- There are now two versions of CAN FD

- non ISO CAN FD (original Bosch proposal)
- ISO CAN FD (ISO11898-1:2015) including the two fixes

CAN-FD Errata - CAN FD Errata Background

- Errata will be released shortly for a number of products that have the “non ISO CAN FD” capabilities on them.
- These errata include:
  - ERR008759 (for devices with FlexCAN3 (non ISO FD CRC))
  - ERR008860 (for devices with MCAN and a FSL wrapper (non ISO FD CRC))
  - ERR008923 (for devices with MCAN and a ST wrapper (non ISO FD CRC))
- This errata was driven by this flaw in the CRC definition of the original CAN FD specification that resulted in a need for a late change in the ISO 11898-1.

- This is not a bug in the controller silicon!

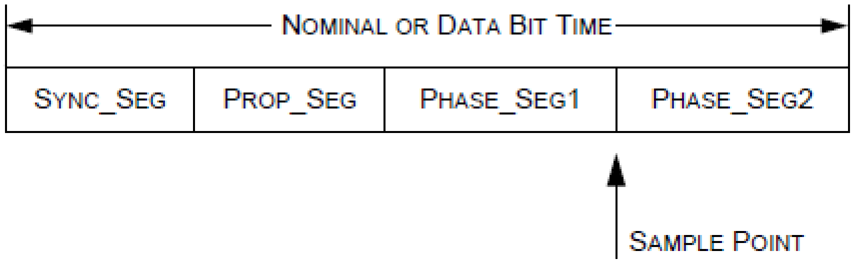
Other OEM Requirements

- 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.

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



NOMINAL OR DATA BIT TIME

SYNC_SEG	PROP_SEG	PHASE_SEG1	PHASE_SEG2

SAMPLE POINT

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)

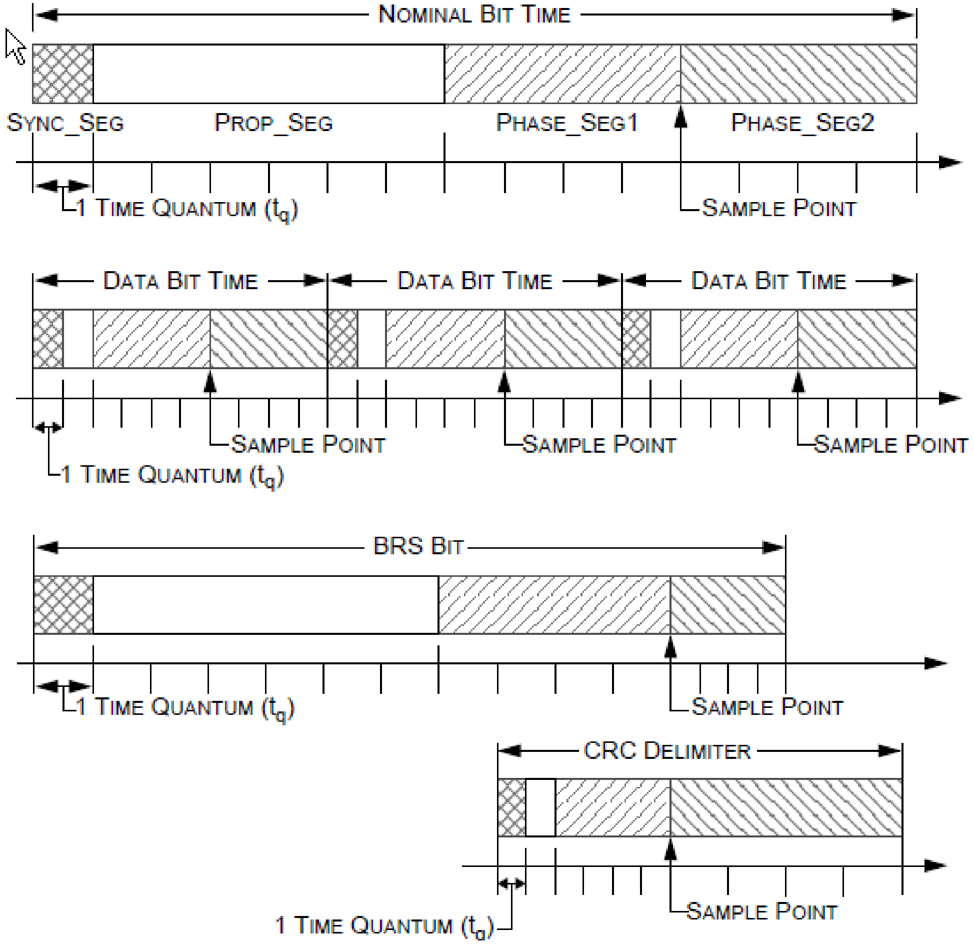
CAN FD Bit Rate Example

- NOMINAL BIT TIME prescaler m(N) = 2
- DATA BIT TIME prescaler m(D) = 1

Keep both prescaler values equal whenever possible!!

Make SJW in data phase not bigger than mandated by the precision of the clock sources.

Make SJW in arbitration phase big to allow for phase shifts at loss of arbitration



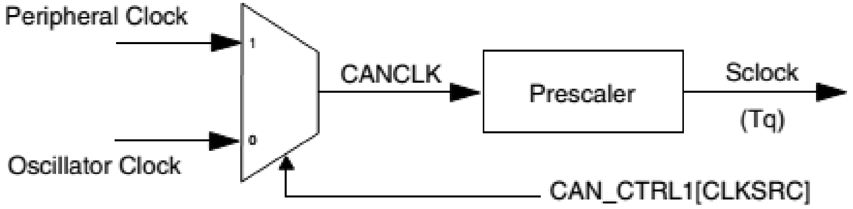
USING FLEXCAN3_FD

FEATURES

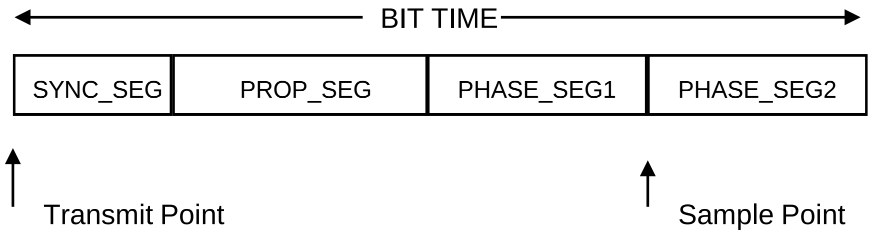
- FlexCAN3-FD Module Features:
  - Up to 128 Message buffers configurable as Tx or Rx
  - Standard and Extended ID frames and Remote Frames* (not for FD)
  - Payload: 0 to 64 bytes data length
  - Programmable bit rate up to 8 Mb/sec
  - Individual Rx Mask Registers per Message Buffer
  - Full featured 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, optional source
  - Low Power operation via pretended networking feature

Protocol Bit Timing

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



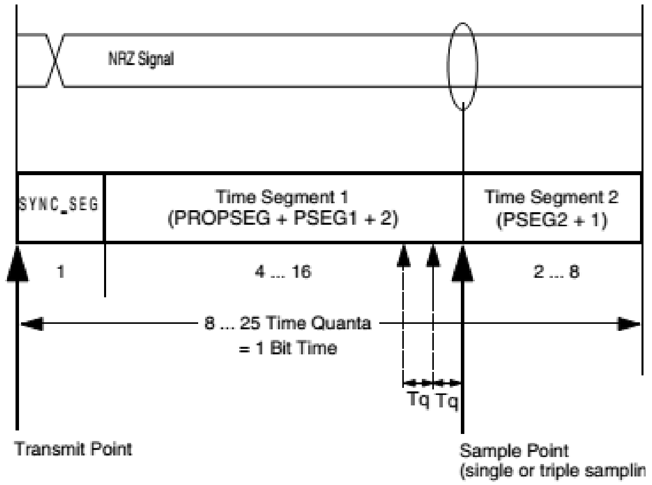
- Recall, FD frame requires two bit rates – Arbitration and data phase
- Bit timing parameters are achieved by setting up the following registers:
  - Control register 1 (CTRL1): “Classical” CAN (equivalent FlexCAN2)
  - CAN bit timing reg (CBT): Extended version of above register
  - CAN FD bit timing 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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRESDIV								RJW		PSEG1			PSEG2		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

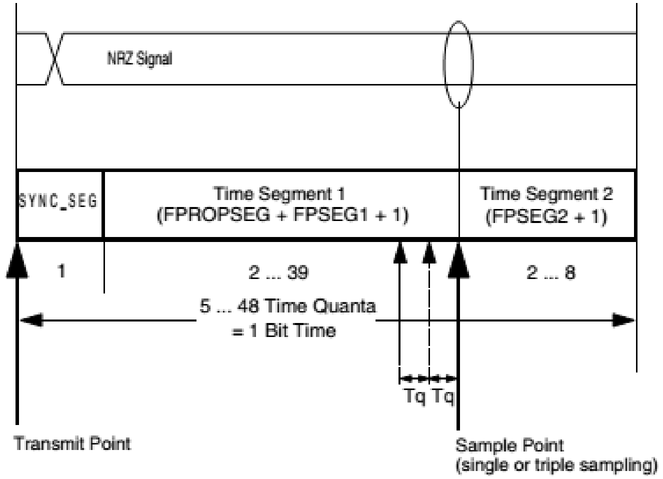
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BOFFMSK	ERRMSK	CLKSRC	LPB	TWRNMSK	RWRNMSK	Reserved	Reserved	SMP	BOFFREC	TSYN	LBUF	LOM	PROPSEG		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



CAN_CTRL1

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		FPRESDIV										Reserved		FRJW	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	FPROPSEG					Reserved		FPSEG1			Reserved		FPSEG2		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



CAN_FDCBT

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

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

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

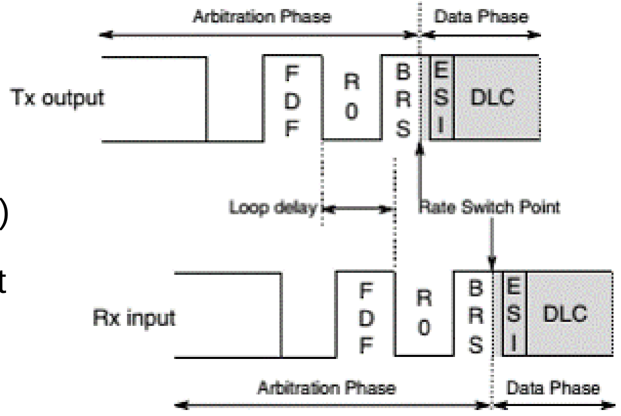
Transceiver Delay Compensation

- 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

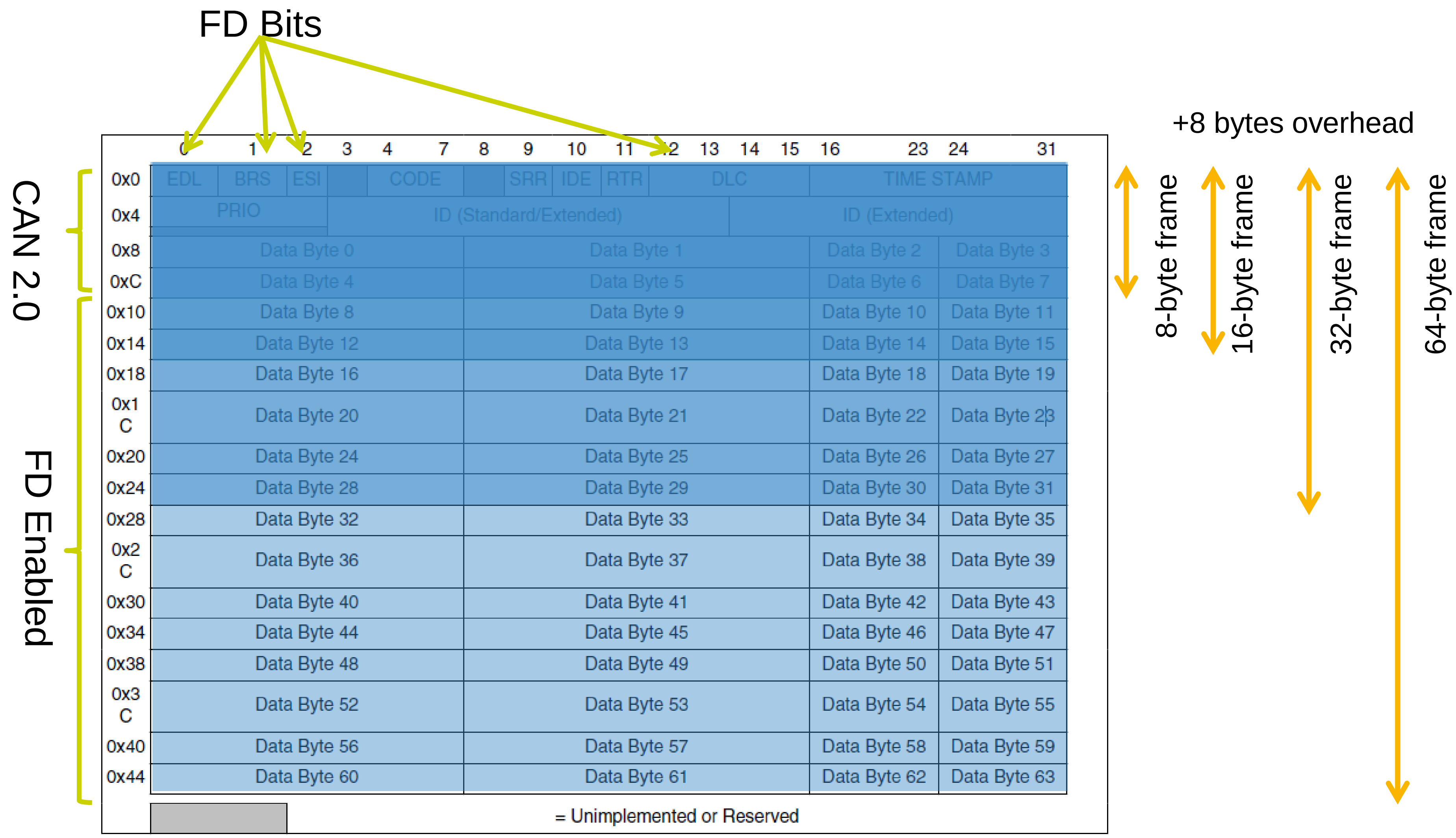
- When higher bit rate data phase occurs in FD frame, the bit time that elapses until the sample point can be shorter than the transceivers loop delay impeding correct comparison.

- 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



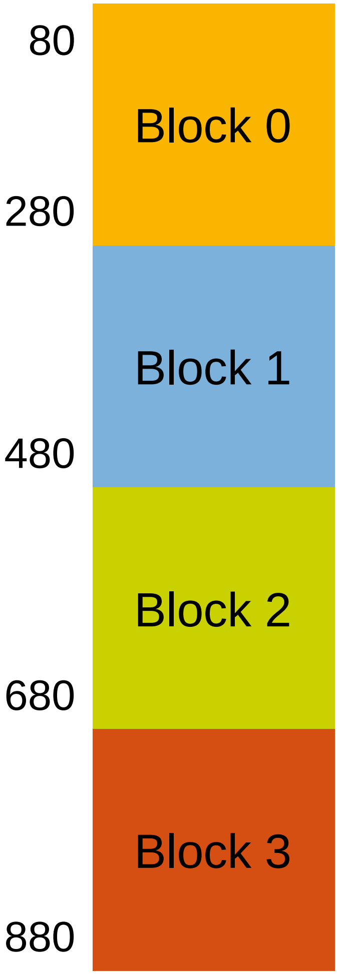
FlexCAN3 FD Message buffer structure



<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>23</th><th>24</th><th>31</th></tr>
<tr><td>0x0</td><td>EDL</td><td>BRS</td><td>ESI</td><td></td><td colspan="2">CODE</td><td></td><td>SRR</td><td>IDE</td><td>RTR</td><td colspan="4">DLC</td><td colspan="4">TIME STAMP</td></tr>
<tr><td>0x4</td><td colspan="3">PRIO</td><td colspan="9">ID (Standard/Extended)</td><td colspan="6">ID (Extended)</td></tr>
<tr><td>0x8</td><td colspan="6">Data Byte 0</td><td colspan="8">Data Byte 1</td><td colspan="2">Data Byte 2</td><td colspan="2">Data Byte 3</td></tr>
<tr><td>0xC</td><td colspan="6">Data Byte 4</td><td colspan="8">Data Byte 5</td><td colspan="2">Data Byte 6</td><td colspan="2">Data Byte 7</td></tr>
<tr><td>0x10</td><td colspan="6">Data Byte 8</td><td colspan="8">Data Byte 9</td><td colspan="2">Data Byte 10</td><td colspan="2">Data Byte 11</td></tr>
<tr><td>0x14</td><td colspan="6">Data Byte 12</td><td colspan="8">Data Byte 13</td><td colspan="2">Data Byte 14</td><td colspan="2">Data Byte 15</td></tr>
<tr><td>0x18</td><td colspan="6">Data Byte 16</td><td colspan="8">Data Byte 17</td><td colspan="2">Data Byte 18</td><td colspan="2">Data Byte 19</td></tr>
<tr><td>0x1C</td><td colspan="6">Data Byte 20</td><td colspan="8">Data Byte 21</td><td colspan="2">Data Byte 22</td><td colspan="2">Data Byte 23</td></tr>
<tr><td>0x20</td><td colspan="6">Data Byte 24</td><td colspan="8">Data Byte 25</td><td colspan="2">Data Byte 26</td><td colspan="2">Data Byte 27</td></tr>
<tr><td>0x24</td><td colspan="6">Data Byte 28</td><td colspan="8">Data Byte 29</td><td colspan="2">Data Byte 30</td><td colspan="2">Data Byte 31</td></tr>
<tr><td>0x28</td><td colspan="6">Data Byte 32</td><td colspan="8">Data Byte 33</td><td colspan="2">Data Byte 34</td><td colspan="2">Data Byte 35</td></tr>
<tr><td>0x2C</td><td colspan="6">Data Byte 36</td><td colspan="8">Data Byte 37</td><td colspan="2">Data Byte 38</td><td colspan="2">Data Byte 39</td></tr>
<tr><td>0x30</td><td colspan="6">Data Byte 40</td><td colspan="8">Data Byte 41</td><td colspan="2">Data Byte 42</td><td colspan="2">Data Byte 43</td></tr>
<tr><td>0x34</td><td colspan="6">Data Byte 44</td><td colspan="8">Data Byte 45</td><td colspan="2">Data Byte 46</td><td colspan="2">Data Byte 47</td></tr>
<tr><td>0x38</td><td colspan="6">Data Byte 48</td><td colspan="8">Data Byte 49</td><td colspan="2">Data Byte 50</td><td colspan="2">Data Byte 51</td></tr>
<tr><td>0x3C</td><td colspan="6">Data Byte 52</td><td colspan="8">Data Byte 53</td><td colspan="2">Data Byte 54</td><td colspan="2">Data Byte 55</td></tr>
<tr><td>0x40</td><td colspan="6">Data Byte 56</td><td colspan="8">Data Byte 57</td><td colspan="2">Data Byte 58</td><td colspan="2">Data Byte 59</td></tr>
<tr><td>0x44</td><td colspan="6">Data Byte 60</td><td colspan="8">Data Byte 61</td><td colspan="2">Data Byte 62</td><td colspan="2">Data Byte 63</td></tr>
</table>

▨ = Unimplemented or Reserved

FlexCAN3 FD Memory Partitioning

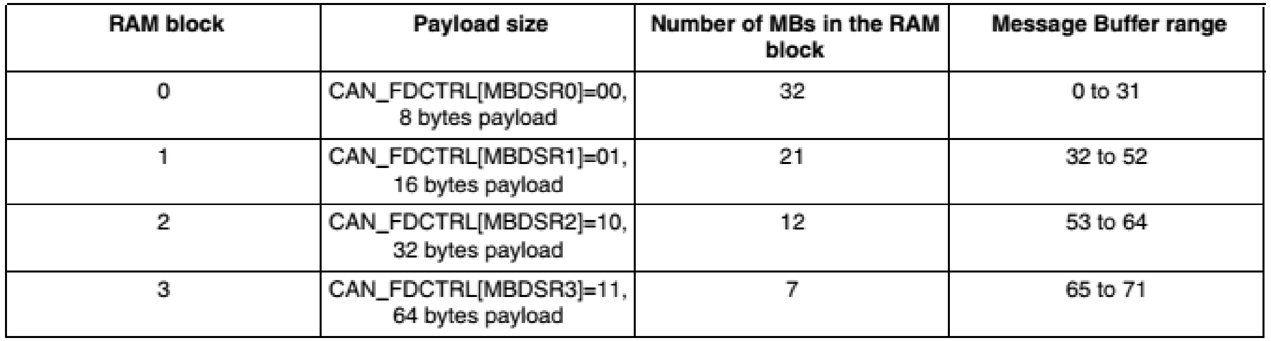


- FlexCAN3 FD memory is partitioned in 4 blocks
  - Essentially 128 x 8-byte payload buffers can be supported
  - SoC can instantiate 1 to 4 blocks

- Each block is composed by the memory space equivalent to 32 x 16-byte MBs or 512-bytes (0x200)

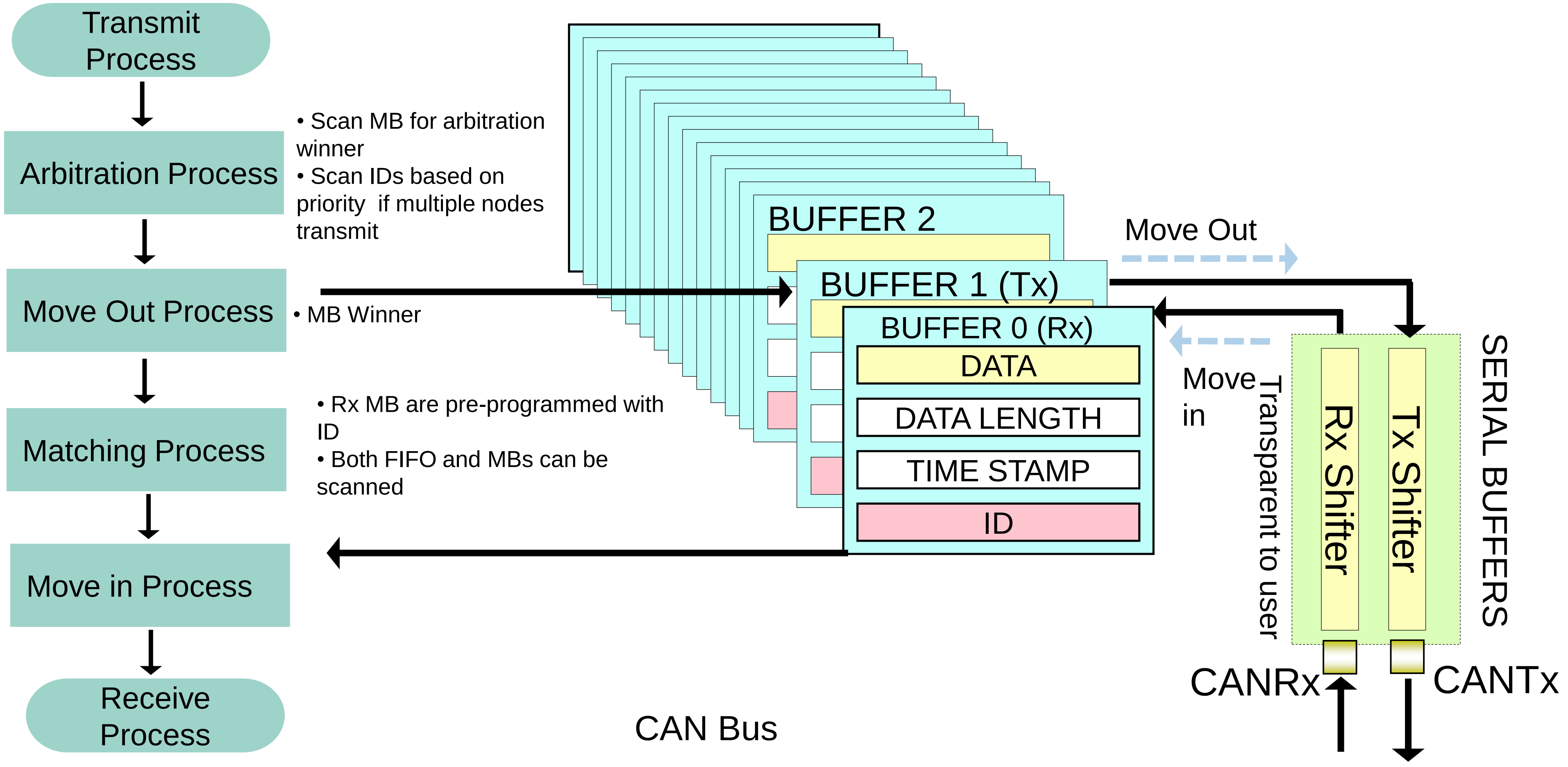
- Each block can be configured to have message buffers (MBs) of a defined size in order to store messages with payload of 8-byte, 16-byte, 32-byte or 64-bytes. Message buffer data size region bit field:
  - **FDCTRL[MBDSRn] = 0bxx**

- 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

Transmit, Arbitrate, Move, Match and Receive!



- Scan MB for arbitration winner
- Scan IDs based on priority if multiple nodes transmit

- MB Winner

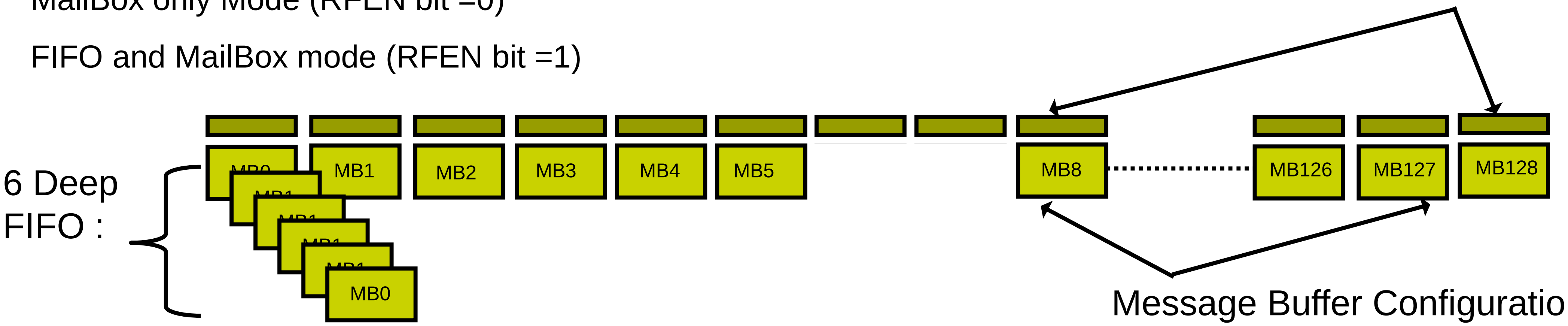
- Rx MB are pre-programmed with ID
- Both FIFO and MBs can be scanned

FlexCAN Rx FIFO

Rx FIFO cannot be used for CAN FD frames

MailBox only Mode (RFEN bit =0)

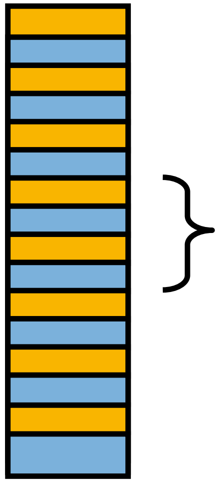
FIFO and MailBox mode (RFEN bit =1)



Individual masks per MB

6 Deep FIFO :

Message Buffer Configuration



Filter Table :

8-128 entry filter table and masks

Filter on ID, IDE and RTR

Filter Mask Options:

Individual 32bit mask per table entry

Filter Table Options:

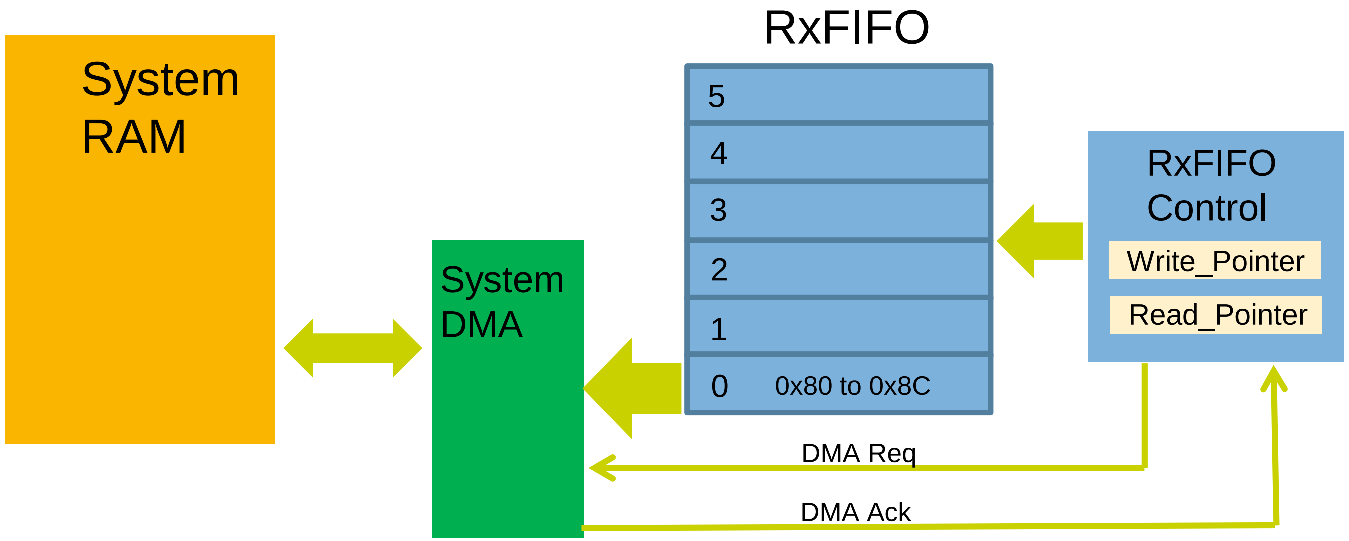
RXIDA: 128 x Standard or Extended IDs (includes IDE and RTR)

RXIDB: 256 x Standard IDs or Extended 14-MSBit IDs slices (includes IDE and RTR)

RXIDC: 512 x Standard or Extended 8-MSBit slices (includes IDE and RTR)

Rx FIFO Under DMA Operation

- Enhanced Feature
  - **DMA Request added to FlexCAN RxFIFO**
    - Allows CPU longer time to service the received CAN messages. With the DMA connected the depth of the FIFO should also be able to be relatively small
    - Enabled via the MCR[DMA]
    - DMA Request asserted when any frame in FIFO
      - DMA reads head of FIFO (0x80 to 0x8C). 8C must be read last to update for next FRAME



USING MCAN FD

MCANs Features and user point

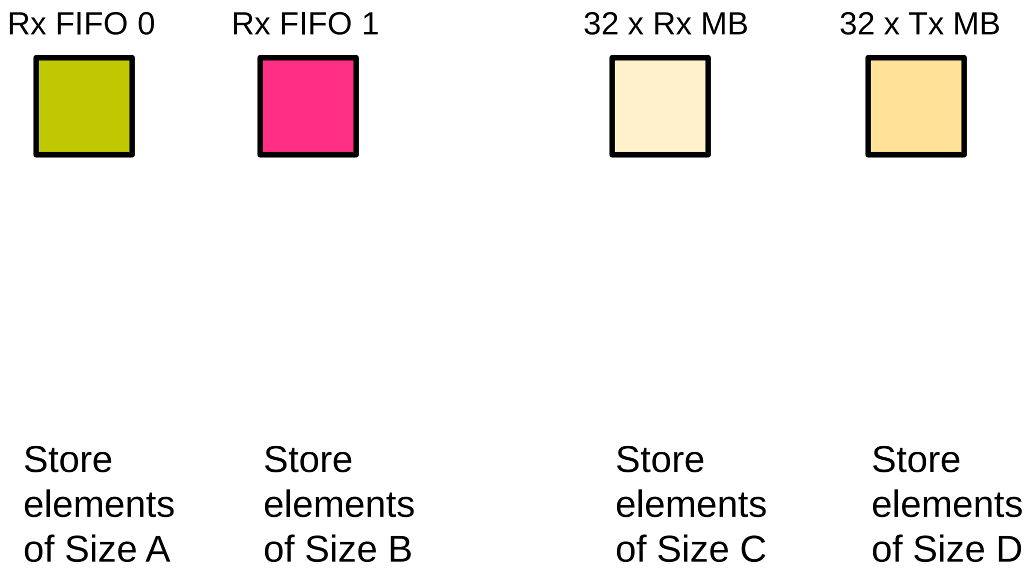
- Debug over M_CAN
- FD-Enabled
- 20kB shared message Random Access Memory (RAM)
- 128 Filter elements
- 2 configurable receive First In First Out registers (FIFO) for each module
- 64 dedicated receive and 32 dedicated transmit buffers
- 11 bit or 29 bit ID lengths

MCAN FD Memory Partitioning

- The MCAN FD provides several data structures to store Messages, such as FIFOs and MBs (Message Buffers). The data payload size for each data structure is fixed for a given module configuration (configured by the software).

- For example: if the Rx FIFO 0 is defined to store 32-byte payload messages, it only stores messages with size of up to 32-bytes payload.

MCAN memory partitioning


Rx FIFO 0 | Rx FIFO 1 | 32 x Rx MB | 32 x Tx MB

Store elements of Size A

Store elements of Size B

Store elements of Size C

Store elements of Size D

Different sizes can be configured but only 1 size for each data structure

FIFOs and MBs data structures may store elements of one of the following sizes (configured by the software) : 8, 12, 16, 20, 24, 32, 48 or 64

Since there is only one Tx structure, only one message size is allowed.

CAN 2.0 and FD messages cannot be interleaved. If FD mode is selected only FD messages are treated.

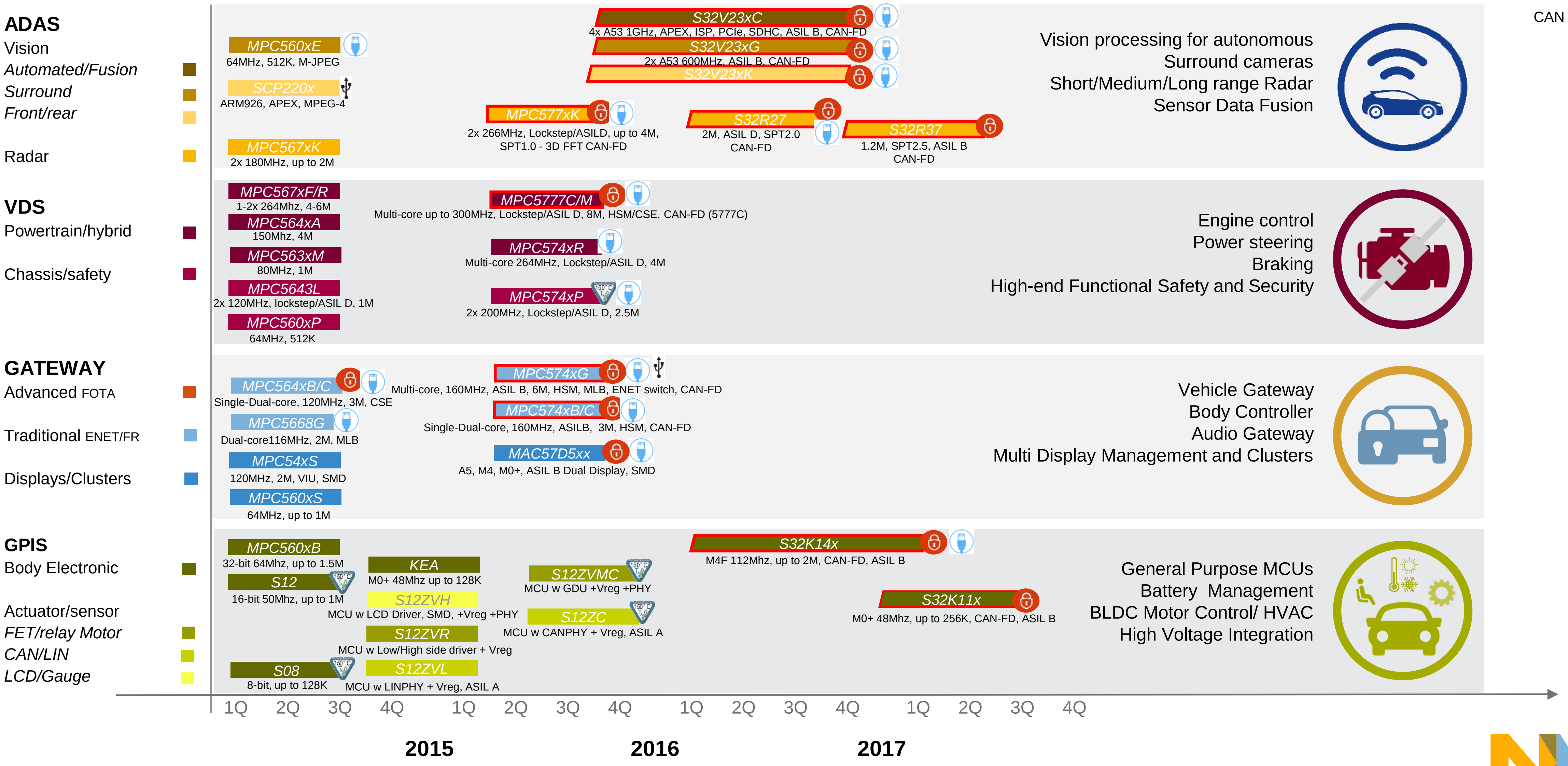
CAN Module Comparison

	FlexCAN3_FD	MCAN_3	TTCAN_2
Functional Differentiator	Flexible Data Structure (mail box + FIFO)	Shared RAM	Shared RAM + Message Time scheduling
FD Support	Yes	Yes	Yes/Partial
MBs and FIFO	Yes	Yes	Yes
Message RAM	2k RAM: Partitioned into 4 x 512 byte blocks**	Shared 20k RAM block*	
ID Acceptance Filtering	Yes	Yes	Yes
Debug over CAN	No	Yes	Yes
DMA support	Yes	No	No
Pretended Networking (low pwr support)	Yes	No	No
Optional ext timestamp with clock	Yes	No	No

AMP CAN FD ROADMAP

AMP Product Offering and Roadmap

Hardware Security
Ethernet
USB
135-150C Ambient
CAN FD



ADAS
Vision
Automated/Fusion
Surround
Front/rear
Radar

MPC560xE — 64MHz, 512K, M-JPEG
SCP220x — ARM926, APEX, MPEG-4
MPC567xK — 2x 180MHz, up to 2M
S32V23xC — 4x A53 1GHz, APEX, ISP, PCIe, SDHC, ASIL B, CAN-FD
S32V23xG — 2x A53 600MHz, ASIL B, CAN-FD
S32V23xK
MPC577xK — 2x 266MHz, Lockstep/ASILD, up to 4M, SPT1.0 - 3D FFT CAN-FD
S32R27 — 2M, ASIL D, SPT2.0 CAN-FD
S32R37 — 1.2M, SPT2.5, ASIL B CAN-FD

Vision processing for autonomous
Surround cameras
Short/Medium/Long range Radar
Sensor Data Fusion

VDS
Powertrain/hybrid
Chassis/safety

MPC567xF/R — 1-2x 264Mhz, 4-6M
MPC564xA — 150Mhz, 4M
MPC563xM — 80MHz, 1M
MPC5643L — 2x 120MHz, lockstep/ASIL D, 1M
MPC560xP — 64MHz, 512K
MPC5777C/M — Multi-core up to 300MHz, Lockstep/ASIL D, 8M, HSM/CSE, CAN-FD (5777C)
MPC574xR — Multi-core 264MHz, Lockstep/ASIL D, 4M
MPC574xP — 2x 200MHz, Lockstep/ASIL D, 2.5M

Engine control
Power steering
Braking
High-end Functional Safety and Security

GATEWAY
Advanced FOTA
Traditional ENET/FR
Displays/Clusters

MPC564xB/C — Single-Dual-core, 120MHz, 3M, CSE
MPC5668G — Dual-core116MHz, 2M, MLB
MPC54xS — 120MHz, 2M, VIU, SMD
MPC560xS — 64MHz, up to 1M
MPC574xG — Multi-core, 160MHz, ASIL B, 6M, HSM, MLB, ENET switch, CAN-FD
MPC574xB/C — Single-Dual-core, 160MHz, ASILB, 3M, HSM, CAN-FD
MAC57D5xx — A5, M4, M0+, ASIL B Dual Display, SMD

Vehicle Gateway
Body Controller
Audio Gateway
Multi Display Management and Clusters

GPIS
Body Electronic
Actuator/sensor
FET/relay Motor
CAN/LIN
LCD/Gauge

MPC560xB — 32-bit 64Mhz, up to 1.5M
S12 — 16-bit 50Mhz, up to 1M
S08 — 8-bit, up to 128K
KEA — M0+ 48Mhz up to 128K
S12ZVH — MCU w LCD Driver, SMD, +Vreg +PHY
S12ZVR — MCU w Low/High side driver + Vreg
S12ZVL — MCU w LINPHY + Vreg, ASIL A
S12ZVMC — MCU w GDU +Vreg +PHY
S12ZC — MCU w CANPHY + Vreg, ASIL A
S32K14x — M4F 112Mhz, up to 2M, CAN-FD, ASIL B
S32K11x — M0+ 48Mhz, up to 256K, CAN-FD, ASIL B

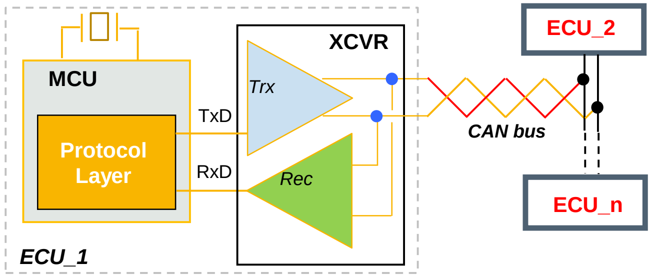
General Purpose MCUs
Battery Management
BLDC Motor Control/ HVAC
High Voltage Integration

Product Idea
Concept
Development
Production

FLEXIBLE DATA AND PARTIAL NETWORKING IMPACT ON CAN PHYSICAL LAYER

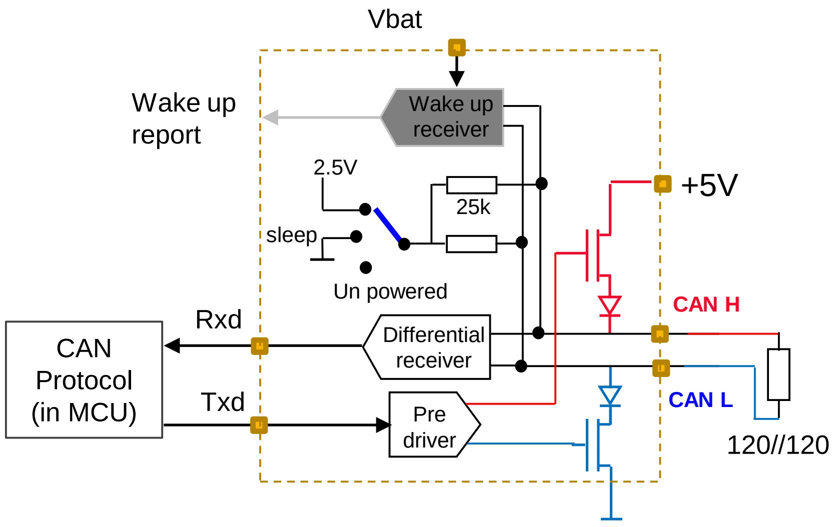
ISO Standards Applicable to CAN Physical Layer

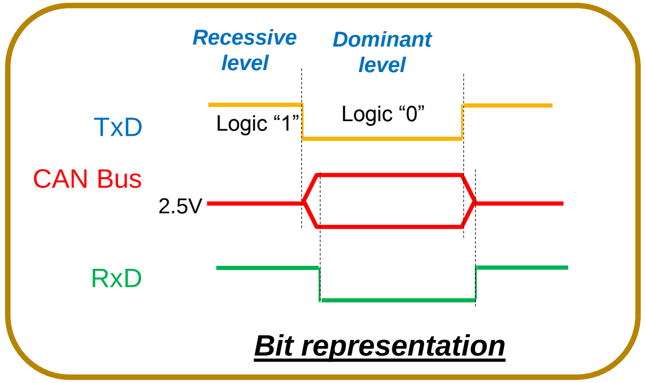
- CAN Flexible Data and Partial Networking
  - Flexible data has been incorporated into the new ISO11898-1 specification
  - ISO11898 -2, -5, and -6 will be merged into a new ISO11898-2 specification
  - EMC/ESD target levels and limits may get tighter
  - Additional testing will be required for Partial Networking
  - Requires better intrinsic Physical Layer performance and functionality



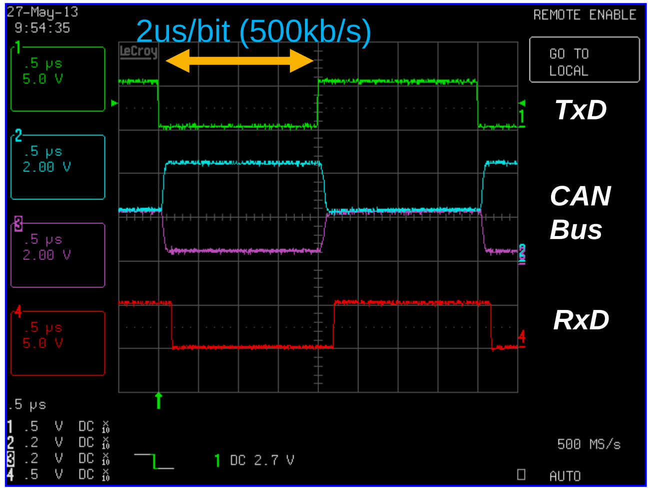
Standards	XCVR mode	Operation
ISO11898-2	Transmit-Receive (Normal mode)	Bi-directional interface to the physical bus
ISO11898-5	Low power mode and wake up	Wake up on any CAN frame
ISO11898-6	Partial Networking, selective wake up (frame detect mode)	Wake up on a dedicated frame (ISO11898-1). Error management.

High Speed CAN Physical Layer
Simplified Architecture – ISO11898-2 and -5



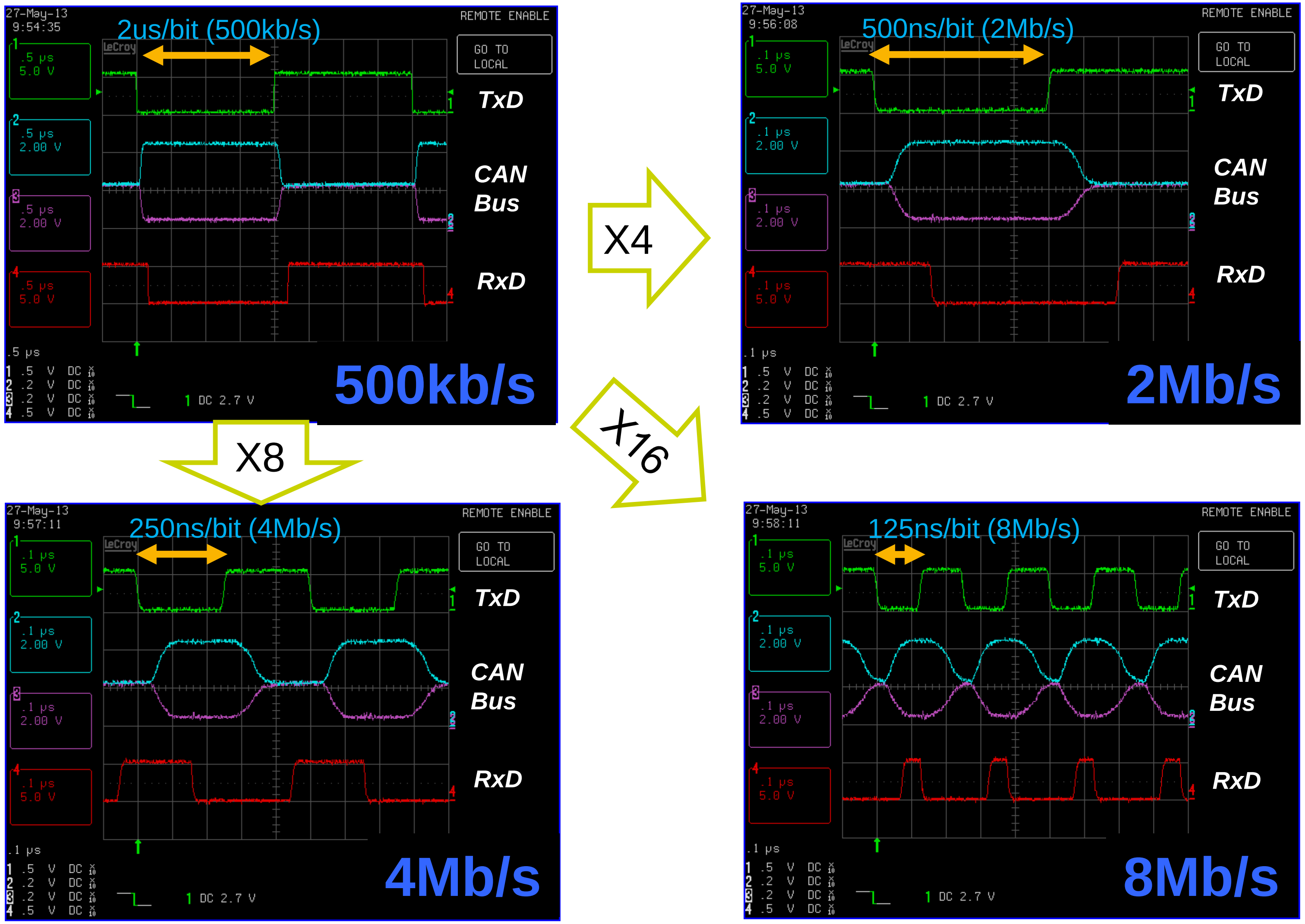


Bit representation



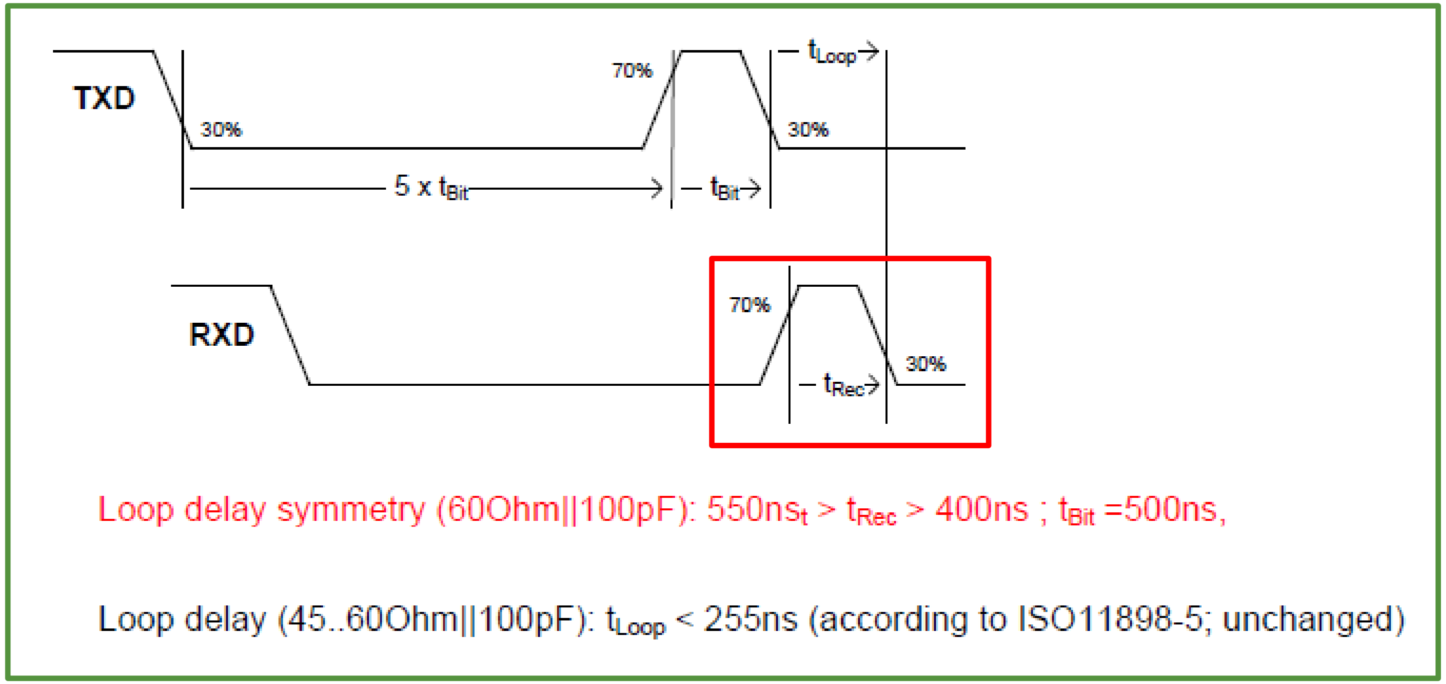
Signals at 500 kb/s, 2, 4 and 8 Mb/s

(ISO11898-2 Test Conditions: 60 ohms / 100 pF – “lab conditions” according to ISO11898-2)



New Timing Definition

Ensure a minimum duration of the recessive level

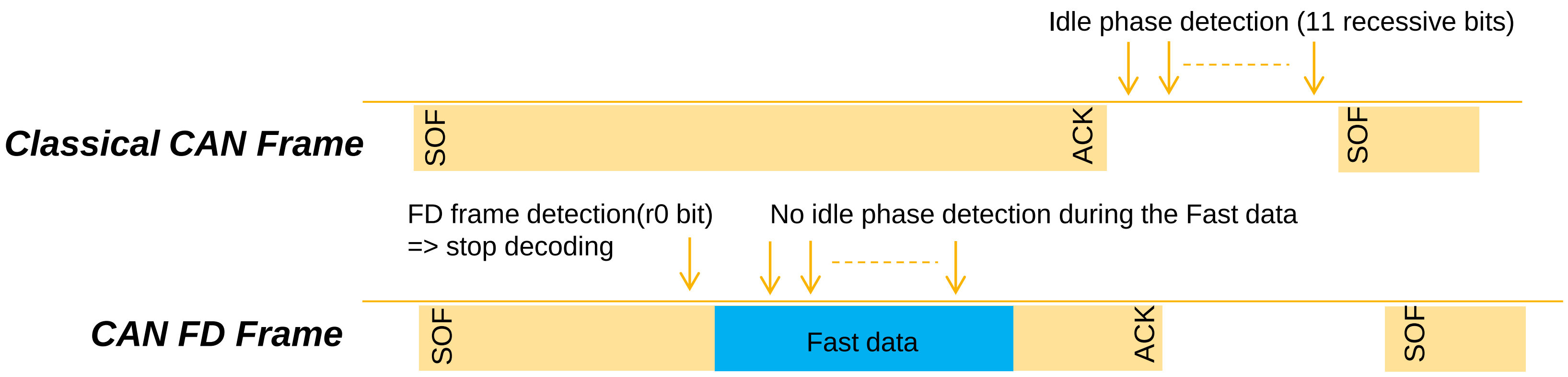


Loop delay symmetry (60Ohm||100pF): 550ns$_{t}$ > t_{Rec} > 400ns ; t_{Bit} =500ns,

Loop delay (45..60Ohm||100pF): t_{Loop} < 255ns (according to ISO11898-5; unchanged)

More bit timing requirements are given in ISO11898-2 for 2Mbit/s and 5MBit/s

CAN Flexible Data Impact on ISO11898-6 Physical Layer


Idle phase detection (11 recessive bits)
Classical CAN Frame
SOF
ACK
SOF
FD frame detection(r0 bit)
=> stop decoding
No idle phase detection during the Fast data
CAN FD Frame
SOF
Fast data
ACK
SOF

Partial networking begin FD passive means:

- CAN Flexible Data frame shall not “disturb” Partial Networking transceiver:
- The circuitry must be designed to avoid “Error Detection” and to prevent the Frame Error Counter from increasing and falsely waking up the transceiver
- Proper detection of “End of frame / Idle detection”

Conclusion

- Industry trend is increased **bandwidth** and EMC **robustness** combined with **energy savings**
- CAN Network is evolving with emergence of **CAN Partial Networking** and **CAN Flexible Data** standards to support these needs, alone or combined together
- At Physical Layer there are conflicting challenges:
  - Improving bandwidth, reduce immunity to power injection and increase noise emission level
  - Reducing quiescent current, improving noise immunity, EMC, and ESD continue to be extremely important.
  - Need to sustain performance without need for common mode choke
- NXP is focusing analog innovations to support High Speed **CAN, CAN FD and Partial Networking**

SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT