Introducing CAN XL into CAN Networks

Florian Hartwich, Robert Bosch

The next generation of CAN communication is currently being specified inside the CiA's CAN XL Special Interest Group. The new frame format combines Ethernet style frames with the non-destructive collision-resolution of CAN arbitration. Newly developed CAN XL transceivers with symmetrical bit levels in the data phase will enable a net bit rate of 10 MBit/s, but existing CAN FD transceivers may also be used, at lower bit rate.
The new CAN XL frame format introduces a header CRC, a payload type describing the contents of the data field, and up to 2048 data bytes. After the arbitration is decided, the bit timing is switched from arbitration bit rate to data bit rate and the CAN XL transceivers are set into a high-speed operating mode. They return to arbitration mode before CAN style acknowledgement.
This paper explains the CAN XL frame format in detail, especially the differences and the compatibilities between CAN XL and CAN FD. Further emphasis is placed on the introduction strategy of CAN XL nodes into existing CAN FD systems and on new communication concepts. Finally, the paper shows the impact of the longer payloads on the hardware implementation of CAN controllers.

Introduction

The initial spark for CAN XL was a presentation at the ISO/TC 22/SC31/WG3 plenary meeting in September 2018, where Volkswagen started a discussion about a bus network that combines Ethernet style frames with CAN style arbitration at a net data rate of 10 MBit/s.

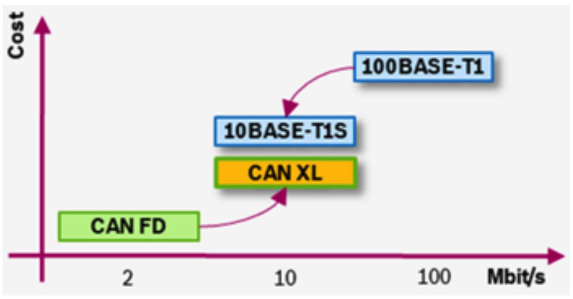

Figure 1: Filling the gap between CAN FD and Ethernet

While an automotive Ethernet bus network is also in development (10BASE-T1S), that requires a more complicated collision-avoidance method, compared to CAN's non-destructive collision-resolution.
The presentation found widespread interest in both the automotive as well as in the industrial automation industry and lead to the establishment of the CAN XL Special Interest Group (SIG) inside CAN in Automation for further discussion on this new concept.
The name CAN XL was chosen at the first SIG CAN XL meeting in December 2018 that started the specification of the CAN XL protocol and physical layer as parts of the forthcoming CiA610 document series.
The goal of the SIG CAN XL is a technically stable CAN XL specification for OSI layers 1 and 2 (known as CAN XL protocol) that merges CAN principles with higher bandwidth, low cost, and future proof features. As second step, the ISO standardization will be started. In addition, many other CAN related standards will be updated simultaneously to support CAN XL, namely the CAN Conformance Test according to ISO16845, time-synchronization, and the transport layer.

Key Protocol Features

- Net data rates up to 10Mbit/s
- Large data fields with up to 2048 byte enable the tunneling of complete Ethernet frames e.g. used for higher layer protocols like IP (Internet Protocol)
- Interoperability with CAN FD for mixed FD/XL networks

Physical Layer

CAN XL will be operable with four classes of CAN transceivers (see also [5]):

- Classical CAN up to 1Mbit/s
- CAN FD up to 2Mbit/s
- CAN FD-SiC up to 5-8Mbit/s
- CAN XL up to 10 Mbit/s and beyond (in development)

The actual bit rate of CAN XL can be adapted to the requirements of the communication system and the network topology.

Compatibility of CAN XL and CAN FD

When CAN FD was specified in ISO 11898-1:2015, one bit was reserved for the future expansion of the protocol. This is the reserved bit after the FDF bit, which is expected to be dominant in CAN FD frames. When it is seen recessive, a CAN FD node will detect a protocol exception. Software configuration decides whether the node treats this as a form error or whether it enters the Protocol Exception State, where it will remain until the bus is idle again.
The CAN XL frame format is, up to the FDF bit, identical to the FD base frame format FBFF. IDE is always dominant in CAN XL frames; they do not use 29-bit identifiers. In CAN XL format, both the FDF bit and the following XLF bit are transmitted recessive. XLF replaces the dominant reserved bit of the CAN FD for-mat, so a receiving node branches at this bit into CAN XL reception state, or into Protocol Exception State when it is a CAN FD node. The transmitter of a CAN XL frame may lose arbitration at the FDF or XLF bits against other nodes sending a Classical CAN or CAN FD frame.

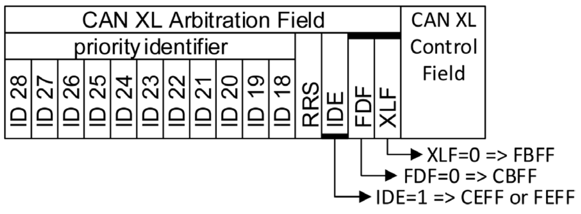

Figure 2: Decoding the Frame Format

The Protocol Exception State feature of CAN FD node allows introducing CAN XL nodes into CAN FD systems, providing an incremental upgrade path to CAN XL.

When a CAN XL frame is transmitted, the recessive XLF bit after the FDF bit sets the CAN FD nodes into the Protocol Exception State where they do not disturb the CAN XL frame. The CAN FD nodes cannot receive the CAN XL frame, but they do not detect an error and they are able to join the arbitration for the following frame.

CAN XL Frame Format

The CAN XL frame format also contains one bit that is reserved for the future expansion of the protocol. This is the resXL bit after the XLF bit, which is expected to be dominant in CAN XL frames. When it is seen recessive, a CAN XL node will detect a protocol exception, which is handled like in a CAN FD node.
The beginning of a CAN XL frame has the same bit rate and stuffing mechanism as in Classical CAN or CAN FD, but after arbitration is decided, the control field is expanded with additional sub-fields and fixed bit stuffing is used. Excluding the 29-bit identifiers enables a high net bit-rate for short payloads while additional addressing may be added to the payload.
The ADS bit sequence provides synchro-nization edges before and after switching from arbitration bit rate to data phase bit rate and optionally switching of the trans-ceiver's operating mode.
The Payload Type PT describes the formatting of the frame's data field. This enables classical CAN data fields, additional addressing, transparent Ethernet frame tunneling, use of TCP/IP, SOME/IP, and more. Details will be specified in a sepa-rate higher-layer standard document, also developed by the SIG CAN XL.
The DLC in the range of 0 to 2047 codes the data field length of 1 to 2048 bytes in byte granularity. The complete header is protected by the header CRC while the Stuff Bit Count SBC checks the number of dynamic stuff bits.

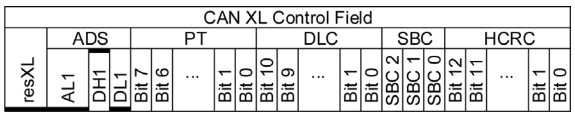

Figure 3: CAN XL Control Field

The data field is followed by the 32 bit long frame CRC sequence that secures the whole frame, including the header. The format check pattern FCP at the end of the CRC field checks for bit insertions and bit drops caused by mis-synchronizations.

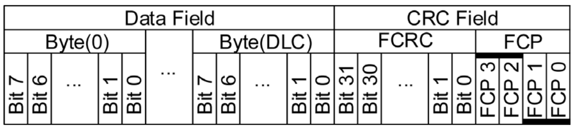

Figure 4: CAN XL Data Field and Frame CRC

After the data phase ends with the CRC field, the DAS bit sequence provides synchronization when switching back to arbitration bit rate. Acknowledge, EOF, and Intermission complete the frame and allow the re-integration of CAN FD nodes.

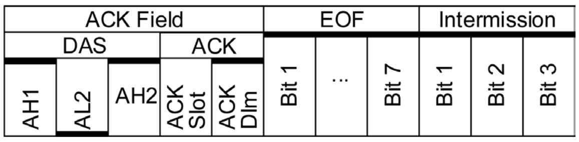

Figure 5: CAN XL Acknowledge and End of Frame

CAN XL Transceiver's Operating Modes

Like in CAN FD, CAN XL frames include a data phase where a faster bit rate is used than in the arbitration phase. The bit rate of the arbitration phase has the same dependency on signal delay times as in CAN FD or Classical CAN, so the three frame formats can arbitrate with each other.

In the data phase, the bit rate is not limited by signal delay time, but by asymmetry of the bit edges and by ringing caused by bus topology. Two new types of transceivers have been developed for CAN FD, "FD" with improved symmetry and "FD-SIC" with a ringing-suppression function. These transceivers enable bit rates up to 8 MBit/s, but they still operate with the usual bit levels specified in ISO 11898-2, "dominant" (transceiver drives differential voltage V_{Diff} to 2 V at twisted pair bus line) and "recessive" (termination resistors shorten V_{Diff} to 0 V).

A new transceiver type with three operat-ing modes is being developed for CAN XL. In the arbitration phase, it operates with the ISO11898-2 levels, but in the data phase, the transmitter drives both bit levels push/pull with the same, inverse strength while the receiver does not drive at all.

The (currently not finalized) specification sets the V_{Diff} levels to ±1V with a receiver threshold T_d at 0V. Target is a gross bit rate of more than 12 MBit/s, resulting in a net bit rate of at least 10 MBit/s.

Switching the transceiver from arbitration phase mode into data phase transmitter mode or receiver mode and then back requires a change in the interface between protocol controller and transceiver.

The CAN XL protocol controller signals the mode switches to the transceiver during the ADS and DAS bit sequences. The signaling method uses only the existing RxD and TxD signals, avoiding the cost of additional pins. The (currently not finalized) specification signals via a pulse on the RxD signal, driven by the protocol controller; details are shown in Figure 7 and Figure 9.

In current transceivers, TxD is output of the protocol controller and RxD is output of the transceiver. The protocol controller gets the ability to switch the direction of its RxD pin for mode signaling, while the transceiver reduces its driving strength on the RxD signal after each change of the signal; details are shown in Figure 6.

When a CAN IP module is integrated into a μC, two of the General Purpose Input/-Output (GPIO) pins are selected for CAN communication. Generic GPIO pins have three register bits to control their direction and their output value or to read their input value, accessible via their peripheral bus, see top of Figure 6. The pin selected as output CAN_Tx needs a multiplexer that enables the output driver to be directly controlled by the protocol controller, which operates in its own CAN clock domain, not in the peripheral bus clock domain. The pin selected as input CAN_Rx needs a direct path to the protocol controller that bypasses the synchronization to the peripheral clock.

For CAN XL mode switching, two multiplexers allow the protocol controller to control direction and output value of the RxD pin.

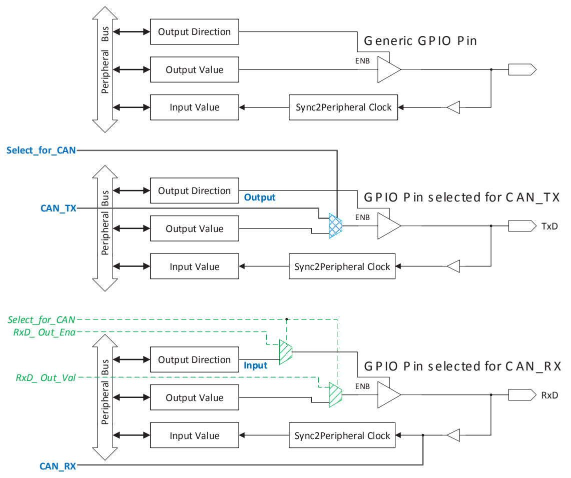

Figure 6: GPIO pins for CAN Controllers

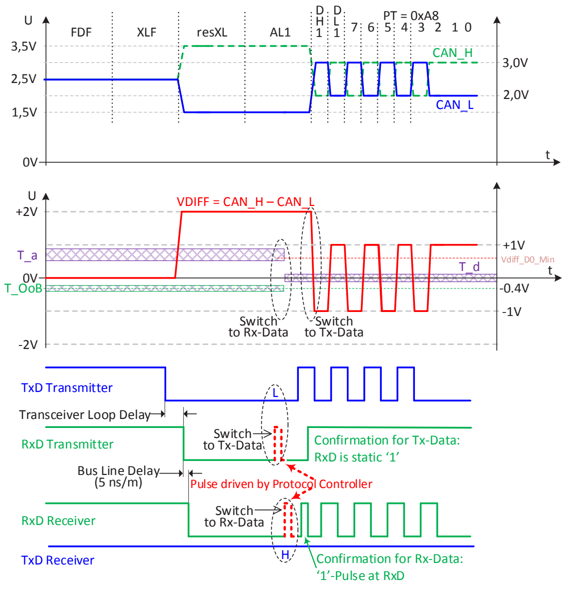

Figure 7: Arbitration to Data Sequence ADS

Entering Data Phase Modes

The transceiver mode is switched from arbitration phase mode to data phase mode during the ADS bit sequence. Both resXL and AL1 bits are driven dominant by the transmitter and recessive by the receivers, so all nodes see $V_{Diff} >$ T_a, setting their RxD pin to low level. During AL1, the protocol controllers set their RxD pins, for a time of 3 tq, into output mode and drive a highpulse, see Figure 7. The transceivers detect the pulse and switch their operating mode into Tx data phase mode (when TxD = L) or into Rx data phase mode (when TxD = H).
The mode-switch is confirmed to the protocol controllers by a static high level at the RxD signal for transmitters or by a level in the rest of the AL1 bit for receivers.

The AL1 bit is the last bit of the arbitration phase, the DH1 bit is the first bit of the data phase, the bit rate changing at the bit boundary. All protocol controllers ignore the edges and the value sampled at the AL1 bit, it is excluded from CRC calculation (see [4]). The receivers perform a hard synchronization at the edge from DH1 to DL1, minimizing their accumulated phase shifts.
In the data phase, the transmitter's transceiver drives the TxD signal symmetrically in push/¬pull mode with VDiff levels at ±1V, beginning with the edge to the DH1 bit. The receivers' transceivers switch their input comparator threshold from the usual T_a (0.7 V) to T_d at 0V, latest at the DH1 bit.
Error flags that overwrite data bits are not possible while the transmitter sends in push/pull mode, so the receivers will not send an error flag when they detect an error. Instead, they will switch their transceivers back into arbitration phase mode and wait for the bus idle detection condition, similar to the restricted operation mode as specified in chapter 10.15 of [2]. When the receivers cannot send error flags, the transmitter dispenses with bit error checks; it treats a transmission as valid when it is acknowledged. For CAN XL error detection capabilities, see [3].

Integrating into CAN XL communication

While a transceiver in data phase mode can decode a '0' bit send by another transceiver in arbitration phase mode and transceiver in arbitration phase mode can decode a '1' bit send in data phase mode, the inverse bit levels cannot be decoded unambiguously.
This could prevent a node that is started while CAN XL communication is already in progress from integrating into the system. A newly started node waits for the idle condition (detection of a sequence of eleven consecutive sampled recessive bits) before it can receive or transmit frames. When the starting node cannot reliably decode the CAN bus signal because its transceiver is not in the same operating mode as the actual transmitter's transceiver, it could detect idle to fast or could not detect it at all. In the first case, it could disturb communication; in the second case, it would not integrate. This

problem can be solved by the introduction of an additional input comparator threshold, T_OoB. This threshold is used during arbitration phase mode, together with T_a. It compensates for ‘0’ bits of the data phase that may reach a receiver with a level that does not exceed T_a and that may be decoded as ‘1’ bits, see Figure 8.

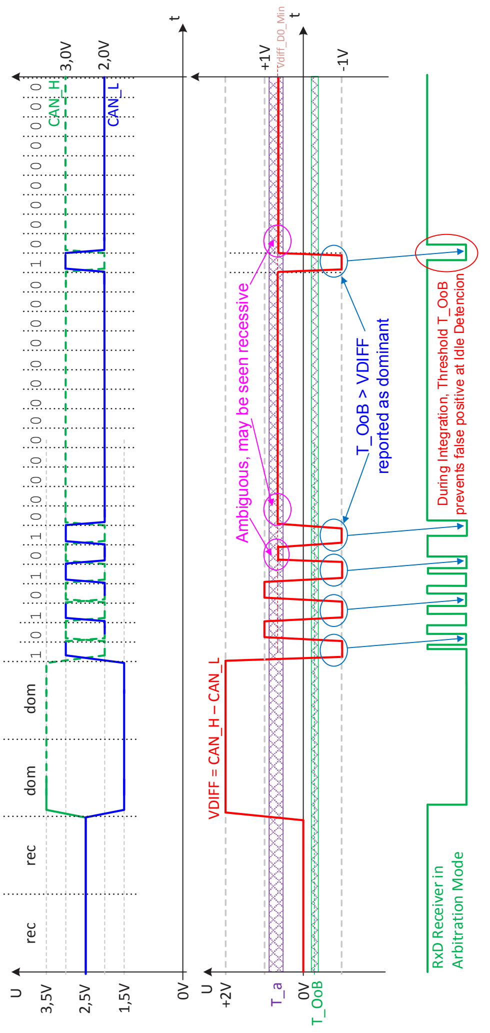
CAN_H
CAN_L
3,0V
2,0V
3,5V
2,5V
1,5V
0V
U
t
rec
rec
dom
dom
+1V
Vdiff_D0_Min
-1V
+2V
T_a
T_OoB
VDIFF = CAN_H – CAN_L
Ambiguous, may be seen recessive
T_OoB > VDIFF reported as dominant
During Integration, Threshold T_OoB prevents false positive at Idle Detencion
RxD Receiver in Arbitration Mode

Figure 8: Integration into CAN XL Communication

To compensate for the missing ‘0’ bits, T_OoB causes ‘1’ bits of the data phase to be decoded as ‘0’ bits, preventing a false positive in idle detection for a starting node with a transceiver operating in arbitration phase mode. When the actual transmitter switches back to arbitration phase mode, the recessive bits enable idle detection.
Idle detection is also used by receivers that detect an error in a CAN XL frame. Instead of starting an error frame, they treat it as a protocol exception and wait until the bus is idle again.

Leaving Data Phase Modes

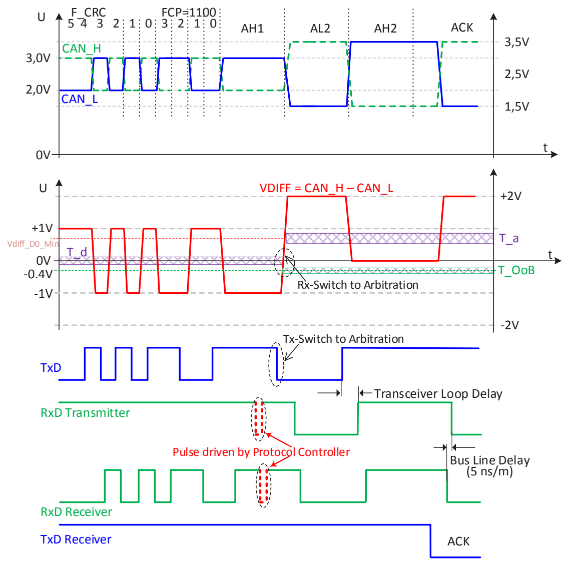

Figure 9: Data to Arbitration Sequence DAS

The transceiver mode is switched from data phase modes to arbitration phase mode during the DAS bit sequence. After the CRC field, all nodes change from data bit rate to arbitration bit rate at the bit boundary between FCP0 and AH1. In AH1, the transceivers stay in data phase mode, so the transmitter’s transceiver drives a negative VDiff, causing the RxD pins to high level.
During AH1, the protocol controllers set their RxD pins, for a time of 3 tq, into output mode and drive a low-pulse, see Figure 9. The transceivers detect the pulse and switch their operating mode into arbitration phase mode at the edge from AH1 to AL2. The transmitter’s transceiver waits for the edge of its TxD signal, the receivers’ transceivers wait for edge on the CAN bus signal.
The edge from AH1 to AL2 synchronizes all nodes to the changed bit rate; it is followed by the recessive AH2 bit. The following bits ACK, End of Frame, and Intermission are again compatible to Classical and CAN FD frame format and allow other nodes to reintegrate when they detected an error or are restricted to CAN FD proto-col frames.
Apart from this regular mode switch during DAS, there are additional mechanisms to return a transceiver into its default arbitration phase mode.
A transceiver in Rx data phase mode switches back into arbitration phase mode when it detects a low signal at its TxD

input or when the distance between two edges on the CAN bus exceeds a timeout limit.
A transceiver in Tx data phase mode switches back into arbitration phase mode when the distance between two edges on its TxD input exceeds a timeout limit.
These two mechanisms allow receivers that detected an error in the frame to reintegrate and prevents a transmitter's transceiver from disturbing the communication when its protocol controller is stopped.

Implementation of CAN XL into Hardware

There are currently several different IP modules available for Classical CAN and for CAN FD, integrated into a wide range of μCs and as standalone protocol control-lers. Future CAN XL IP modules will still be able to handle Classical CAN and CAN FD communication, but updating IP modules from CAN FD to CAN XL is a complex task.
Updating the protocol FSM that encodes and decodes the frames is a straightfor-ward process, it needs additional CRC generators, different stuffing mechanism, new bit time configuration, control func-tions for the RxD pin, and enhanced time stamping function.
The main difference of CAN XL modules compared to CAN FD modules is their message handling. In classical CAN standalone implementations (e.g. CC770), the messages can be stored locally in a dedicated CAM/RAM array that allows very fast acceptance filtering and atomic message handling. In classical CAN IPs (C_CAN), atomic message handling is still possible with a dedicated wide RAM and indirect CPU access. Atomic message handling, where the protocol controller moves a whole CAN message in one RAM access, had to be abandoned with the introduction of CAN FD, where the messages can be up to 64 bytes long. The M_CAN IP module buffers only a part of a received message locally, until acceptance filtering decides in which part of its dedicated message RAM it is to be stored. The M_CAN's message handling concept works well for CAN FD, but has disadvantages for longer CAN XL messages. Critical factors are the space needed for the storage of long messages and the time needed to copy large data packages between system RAM and message RAM. Another factor are the divergent require-ments of different applications. While some applications require a conventional CAN style message handling, others, especially those where Ethernet frames are tunneled (see [6]), require Ethernet style message handling. Gateway implementa-tions will need an interface to e.g. an Enhanced Data Engine (see [7]).
The solution is a separation between the message handling and the protocol controller, where different Message Handler IPs can be connected to the same protocol controller, using standardized interfaces, e.g. AXI interfaces enhanced with user signals.

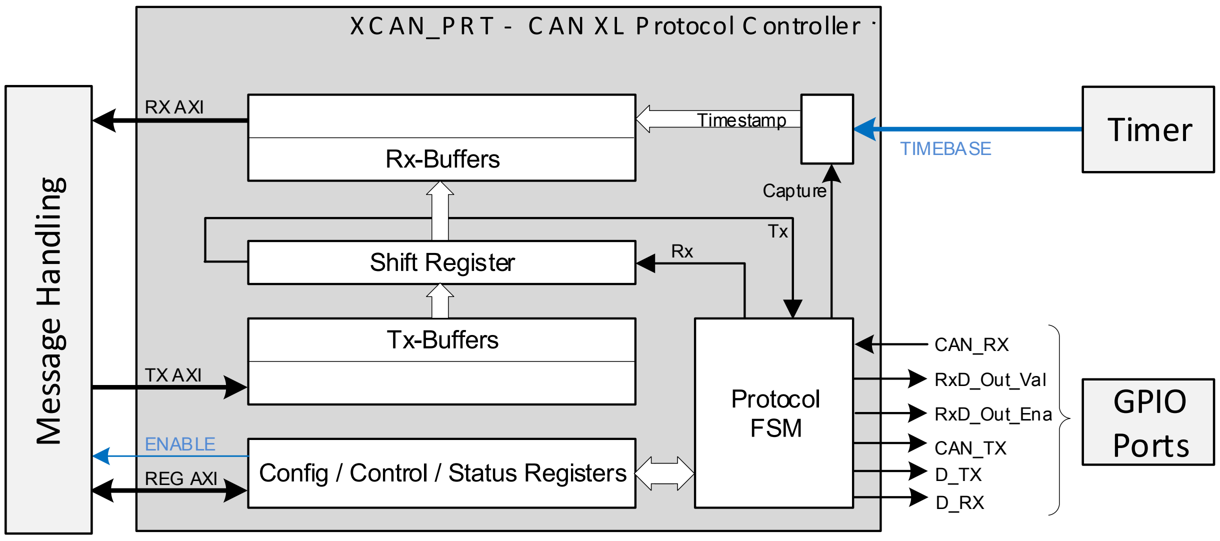

Figure 10: Example of a CAN XL Hardware Implementation

Figure 10 shows an example for such an implementation. XCAN_PRT provides all CAN XL protocol features specified in [1], with interfaces to control the GPIO pins and to capture a time base for time stamps.
Received messages are transferred as a sequence of words via RX_AXI to the Message Handler, while messages to be transmitted are transferred via TX_AXI in the opposite direction.

Conclusion

The introduction of CAN XL into CAN systems not only increases the transmission bandwidth, it also allows the introduction of Ethernet style IP communication. Since CAN XL implementations also support Classical CAN and CAN FD protocol and can be used with existing physical layers, they can be introduced incrementally into CAN applications.
To achieve the full advantages of CAN XL, all nodes need to be upgraded to the new protocol and to the switchable physical layer with a dedicated data phase mode.

Florian Hartwich
Robert Bosch GmbH
AE/EIY4
Postfach 13 42
DE-72703 Reutlingen
www.can.bosch.com

References

[1] CiA 610-1, CAN XL, Data link layer and physical signaling, working draft, December 2019.
[2] ISO 11898-1:2015 Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signaling, 2015
[3] A. Mutter, „CAN XL error detection capabilities“, in Proceedings of the 17th international CAN Conference, Baden-Baden, Germany, 2020.
[4] C. Senger, „CRC error detection for CAN XL“, in Proceedings of the 17th international CAN Conference, Baden-Baden, Germany, 2020.
[5] M-M. Hell, „The physical layer in the CAN XL world“, in Proceedings of the 17th international CAN Conference, Baden-Baden, Germany, 2020.
[6] P. Decker „IP concepts on CAN XL”, in Pro-ceedings of the 17th international CAN Confer-ence, Baden-Baden, Germany, 2020.
[7] A. Lock, “Trends in Future In-Vehicle Communication Networks”, in Proceedings of the 6th Automotive Ethernet Congress, Munich, Germany, 2020