Virtual CAN FD network

Using the "Virtual CAN Bus" it is possible to separate the control tasks from other bandwidth-consuming tasks by modifying the physical layer i.e. the transceiver and the communication network without any impact on the control task.

The bandwidth required for pure control tasks is governed by the dynamics of the machinery to be controlled. CAN is a well-proven network for control tasks in a distributed embedded control system. For example in cars, it is not well suited for requirements that relate to the security of messages, transfer of raw data from advanced sensors and cameras as well as the re-flashing of ECUs (electronic control units) that demand a high bit rate. The primary reason for this is that CAN uses a very simple bit transfer method and a low bit rate at the arbitration phase.

A solution to this dilemma is the "Virtual CAN Bus" where signals to and from ordinary CAN controllers are multiplexed by smart transceivers that support one or more modern high-speed communication protocols running on the same physical layer. In this way, CAN is kept for control tasks and continuously the advantages of the newest technology for bandwidth-hungry tasks are used.

Proposed solution principles

CAN is, in a sense, a unique protocol: The transmitter uses 100 % of the network bandwidth but the receivers use just a fraction of it. The proposed solution takes advantage of this peculiarity. It introduces a "Virtual CAN Bus" (VCB), executed in a smart transceiver unit "Virtual CAN Converter" (VCC) connected to an ordinary CAN controller. The smart VCC transceiver encodes and transmits the CAN information on a modern high-speed communication. CAN signals are transmitted from the CAN controller to the VCC transceiver using the TX connection, according to the CAN standard. However, the signals are reduced to only dominant edges and the respective bit values (i.e. the only things CAN receivers identify), which are then passed to the lower layers of the transceiver unit. The communication in the final system runs on a modern physical layer where such reduced CAN bits are multiplexed and modulated. At reception, the encoded CAN dominant edges and bit values are received and the receiving VCC transceiver restores the CAN bits and signals these to the CAN controller's RX connection.

By using the VCB, the control task is separated from other tasks. The distributed embedded control system can be developed using standard CAN controllers and transceivers in a traditional way with well proven tools. Other tasks such as encryption, transmitter authentication, re-flashing, etc. can be developed by experts in these fields and carried out by using other protocols. With modern technology, the different tasks can run in parallel and simultaneously communicate on the same physical layer. It is a great advantage to separate the control problems from other problems. The control problem can be solved once and for all by the control experts and other problems by experts in their respective technology fields. Any solution to problems in those fields can be implemented at a later date by modifying the physical layer i.e. the transceiver and communication network without any impact on the control task.

CAN background

CAN was designed purposely to fit the needs of a distributed embedded controller network. The most important properties are:

1. No addresses. All nodes receive all CAN frames and determine if the frame should be received or not.
2. All nodes participate in error handling. No application receives a frame if all nodes have not found it to be correct.
3. Bit-wise arbitration at frame collisions. No frame is lost, and the maximum latency of any frame can be calculated.

These three properties solve some general control design problems in an efficient and elegant way.

- A first problem is data consistency within a system. All nodes shall have the same information at any given time. CAN takes care of this.
- A second problem is a predictable latency time. CAN allows the maximum latency time to be calculated, even for unscheduled frames.
- On top of that, the "No-address" feature makes the frames short as only a CAN-Identifier and a value are transported during the runtime. The required network bandwidth is minimized.

When CAN was developed in the early 80s, it was very efficient. It made maximum use of the technology at hand. CAN remains an excellent protocol for control systems. However, new tasks have been assigned to it. The first one was flashing of ECUs. The ECU software is continuously growing and using CAN for this purpose proved too slow. To remediate this problem, Bosch started in 2011 developing the CAN FD (CAN with flexible data rate) resulting in the ISO 11898-1:2015 standard. With the arrival of CAN FD, flashing could be done faster, but more requirements were then brought to the table. Fear for the system hacking requires encryption of the data and authentication of the transmitter and this creates more message overhead, ▷

i.e. longer messages. Another problem is that some safety standards require a Hamming Distance of a minimum of 4. Initially, CAN was said to have a Hamming Distance of 6, but it was later shown that a data bit mistaken for a stuff bit and vice versa might result in the same frame length and the same CRC (cyclic redundancy check) value, in which case the Hamming distance is only 2. This is true also for CAN FD and this might disqualify CAN for use in some safety critical systems.

In addition to distribution of control data in an efficient and reliable way (solved already by Classical CAN), CAN FD should also be capable of:

1. Encryption of messages.
2. Authentication of the message transmitter.
3. Fast file transfer for ECU flashing.

CAN FD does not seem to solve the additional problems and the Hamming distance issue remains.

The solution

CAN is used for feedback loop controls of systems involving mass. The bandwidth needed for control tasks is governed by the dynamics of the controlled items: the lower the mass, the higher the bandwidth needed. Most devices with CAN are related to humans. Since we can expect these 'human-related' devices to remain roughly the same size forever, we can safely say that the dynamic requirements of the future will be the same as today. Notably, many fourth-generation jet fighters are controlled by MIL-STD-1553 systems running at 1 Mbit/s. Does a car really need a faster control system than a high-performance, inherently unstable jet fighter? MIL-STD-1553 is less efficient than Classical CAN, so we can expect CAN FD to match any control demand of the future.

Figure 1: Modern, fourth-generation jet-fighters use MIL-STD-1553 at 1 Mbit/s, which is less efficient than CAN (Source: Adobe Stock)

CAN is more than adequate for control tasks. The driving force for a higher bandwidth for CAN is instead due to non-control issues. We should find a way to run an "old-fashioned" CAN system with a low bit rate on a modern physical layer, multiplexed with another protocol with a high bit rate that carries all the other information needed to satisfy any requirement on top of the control task. A starting point is an efficient system architecture with features given in Figure 2.

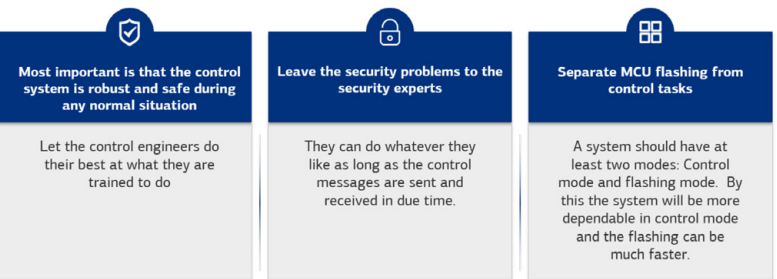


Figure 2: An efficient system architecture (Source: Kvaser)

Hence, let us:

- Separate control problems from other problems
- Use CAN for control tasks
- For other tasks, use better suited protocols
- Run multiplexed protocols on the same physical medium

This can be achieved by having the transceiver establishing a "Virtual CAN Bus" (see Figure 3).

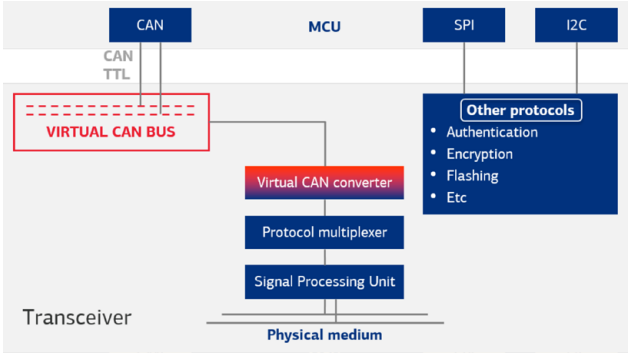


Figure 3: The transceiver establishes a "Virtual CAN Bus" (Source: Kvaser)

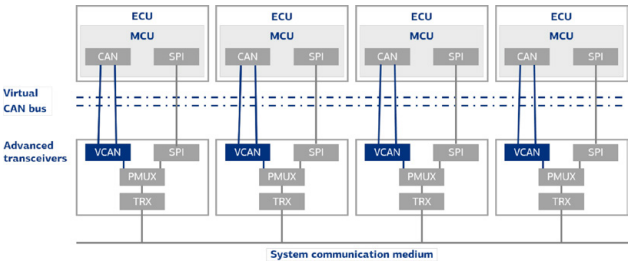


Figure 4: The respective CAN controller sends/receives the TTL signals according to the CAN FD standard (Source: Kvaser)

The physical layer carries two or more protocols in parallel and the transceiver multiplexes/demultiplexes the protocols. A "Virtual CAN Converter" in the transceiver processes CAN frames in a specific way.

Essential CAN features for a "Virtual CAN Converter"

The Figure 5 shows the construction of a CAN FD bit.

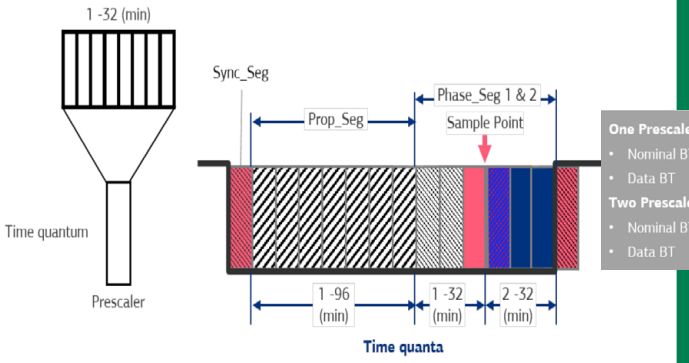


Figure 5: Construction of a CAN FD bit (Source: Kvaser)

According to the ISO 11898-1:2015, a CAN FD bit is constructed on time quanta. A time quantum (TQ) represents a number of clock cycles. A bit starts with a Sync_Seg of one TQ followed by a Prop_Seg, a Phase_Seg 1, and a Phase_Seg 2. The value of the bit is sampled at the sample point located between Phase_Seg 1 and Phase_Seg 2. The signal on the bus lines is the amplitude modulated in the simplest way: A zero is voltage, a one is no voltage. A CAN transceiver balances the outputs CAN high (CAN_H) and CAN low (CAN_L) around 2,5 V (see Figure 6).

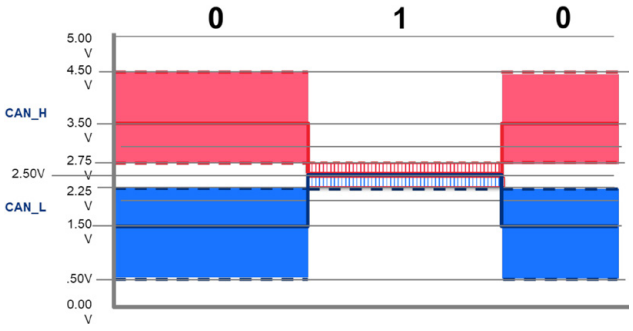


Figure 6: A CAN transceiver balances the outputs CAN high (CAN_H) and CAN low (CAN_L) around 2,5 V (Source: Kvaser)

At the sample point, the bit value is decided by the differential voltage between CAN_H and CAN_L (see Figure 7).

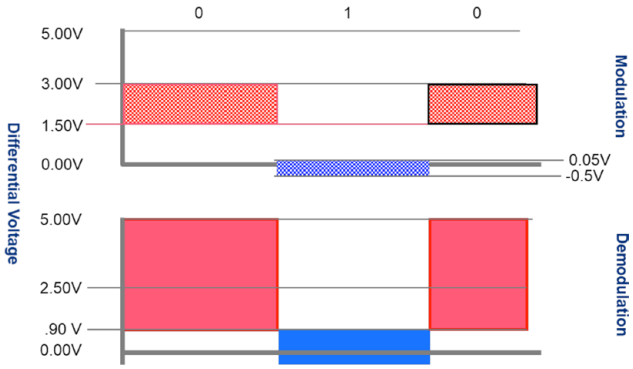
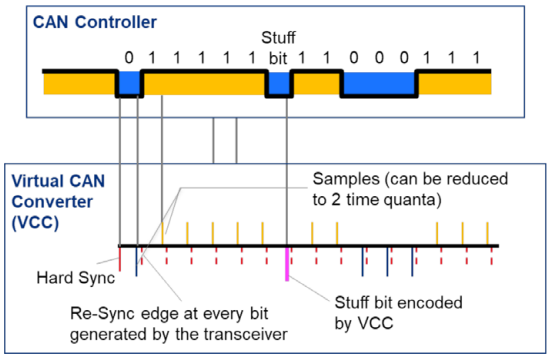


Figure 7: The bit value is decided by the differential voltage between CAN_H and CAN_L (Source: Kvaser)

A unique characteristic of CAN is that the transmitter occupies 100 % of the network bandwidth when transmitting but the receiver only a fraction of it at reception. The receiver only looks for flanks from an idle bus (recessive state, continuous value of 1) to dominant value (0), denoted as a dominant edge, where it makes a hard synchronization of the bit clock. If it samples the zero level at the sampling point, it regards the signal as a "start of frame" (SOF) and continues to look for dominant edges (where it resynchronizes its bit counter) and samples the bit value at each sample point. Any other signal is ignored. Thus, the receiver only uses two or three time quanta of every received bit. CAN uses a non-return-to-zero (NRZ) coding so consecutive bits of the same value are demodulated by dead reckoning of the sample points. Stuff bits with the opposite value are inserted when five bits of the value are transmitted in order to keep the bit clocks synchronized.

Reduced CAN Protocol (RCP)

Implemented in the "Virtual CAN Converter" is a "Reduced CAN Protocol" (RCP) that creates dominant edges at each Sync_Seg and the bit value at each sample point. This would take only one TQ for the Sync_Seg and two for the bit value. According to the CAN standard, a stuff bit of the opposite value should be transmitted after five consecutive bits of the same value. As described earlier, this causes some problems with the Hamming Distance. The "Virtual CAN Converter" could modulate the stuff bits and send an error frame if a stuff bit is detected in the wrong place.



- Only 2Tq per CAN bit needed for the bit value
  - Remaining bit time can be used for other protocol(s)
- Ghost Sync_Segs keeps the CAN controller in perfect sync.
- Stuff bits can be distinguished from data bits

Figure 8: Reduced CAN Protocol principle (Source: Kvaser)

The RCP also generates a dominant edge at the end of every bit, i.e. a "Ghost Sync_Seg" (see Figure 8 and Figure 9) at each bit instance where the CAN controller is not generating a recessive bit followed by a dominant bit (1/0). This keeps the CAN controller in synchronization with the VCC.

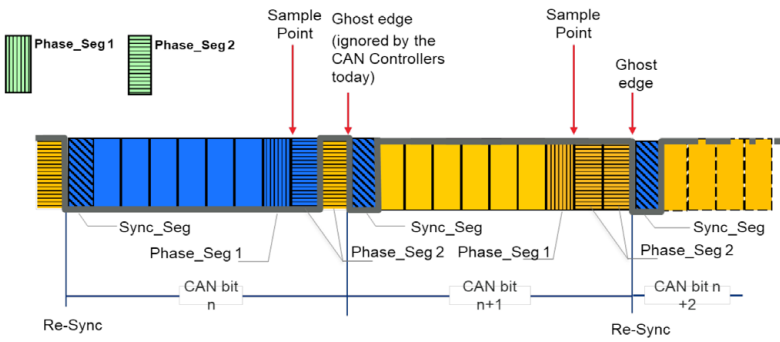


Figure 9: Injection of "Ghost edges" (Source: Kvaser)

According to the CAN protocol, Phase_Seg 2 should be no shorter than two TQ. Being so, the VCC can make a recessive TQ at the Phase_Seg 2 of a dominant bit and we have a dominant edge at every bit. However, the CAN controller ignores any dominant edge after sampling a dominant value, so such an edge will not cause a resynchronization. A worst-case scenario would then be six bits before the CAN controller resynchronizes (five consecutive 0 values and a stuff bit).

The VCC will receive full bits from the CAN controller but can reduce these to edges and bit-value signals at the sample points. The generation of "Ghost edges" after dominant bits could be of value to the "Protocol Multiplexer" (PMUX) but if not, this VCC feature can be omitted.

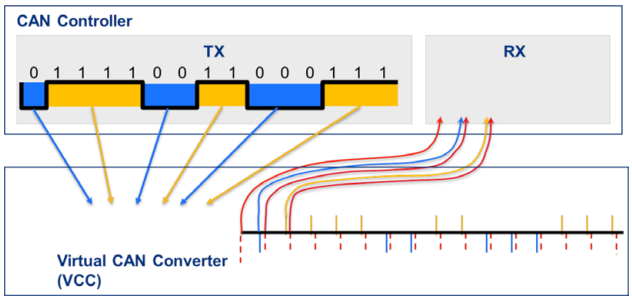


Figure 10: The VCC receives the full bits from the CAN controller on the TX line but transmits back only dominant edges and the bit values at the sampling point on the RX line (Source: Kvaser)

When the CAN controller transmits, the VCC receives full bits, i.e. differential voltage shifts when two consecutive bits have a different value, from the TX line of the CAN controller. The VCC creates at least a Sync_Seg and a bit value at the sample point at the respective bit and feeds them back to the RX line.

The information from the VCC to the "Protocol Multiplexer" can be further reduced. Already at the Sync_seg of bits from the CAN controller, the VCC knows the value of the bit. By applying some of the CAN specification rules, it can also know if it is a start of frame, any of the fixed value bits, end of frame (etc.) and add this information to the "Protocol Multiplexer" by a modulated signal. This can be done in a fraction of the CAN bit time.

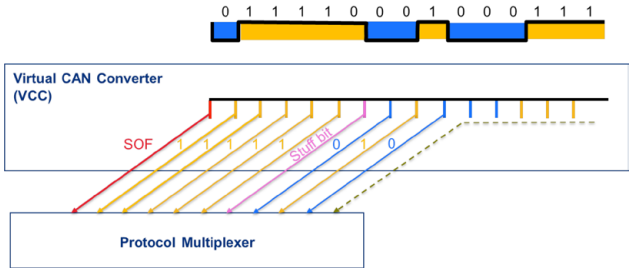
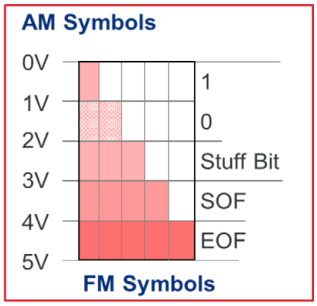


Figure 11: The VCC transmits only encoded Sync-Segs to the PMUX (Source: Kvaser)

The length of a CAN frame can be very important. It is therefore an advantage if the VCC in communication with the "Protocol Multiplexer" generates an encoded SOF bit at the beginning of a frame and an encoded EOF bit at the end of the frame. The RCP should then be capable of generating three specific encoded bits: SOF, EOF, and stuff bits. The encoding can be done in many ways, e.g. by amplitude modulation or phase modulation.

The Sync_Segs are the essential connection signal between the VCC and the PMUX. They can be encoded or modulated in different ways. Most important is that the receivers gets the bit value fast enough to meet the arbitration requirement.



Error and stuff-bit checking can be done by the VCC.

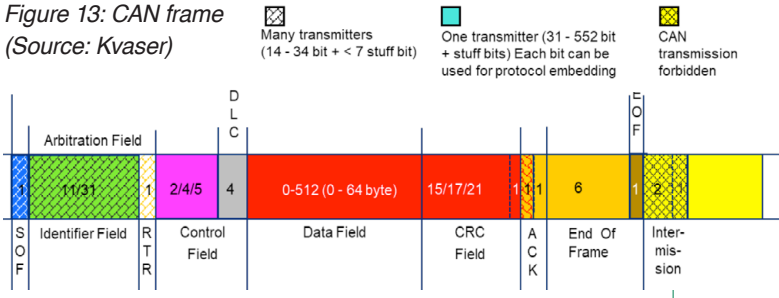
Figure 12: CAN bit modulation (Source: Kvaser)

The TQ on the CAN controller side can be different (longer) from the PMUX side (shorter), as the CAN controller side is limited by old technology but the PMUX can use modern and faster technology. Each Sync_Seg signal at the CAN controller side can be modulated on the PMUX side to immediately also carry the bit value. As the connection between the CAN controller and the VCC is a short point-to-point connection, the risk of disturbances is very low and the signal quality can be constantly supervised and acted upon.

Protocol multiplexing

The basis for the protocol multiplexing is the CAN frame generated by the RCP at the VCC. Such a frame is initiated either by the CAN controller or the "Protocol Multiplexer". It is essential that the timing of the RCP signals is kept by and through the "Protocol Multiplexer" and the "Signal Processing Units", so the VCCs can accurately create the "Virtual CAN Bus". A CAN frame (see Figure 13) starts with the dominant SOF bit followed by the CAN-Identifier field and one more bit (RTR bit in Classical CAN and RRS bit in CAN FD). These bits can be sent simultaneously from two or more CAN controllers. The ACK bit is sent from all receiving CAN controllers.

Figure 13: CAN frame (Source: Kvaser)



This can cause some difficulties for the "Protocol Multiplexer" and the RCP. When a bit value is received when more than one CAN controller is transmitting, the bit value can appear anywhere in the Prop_Seg of the CAN bit. The RCP has to catch it and transmit it to the CAN controller at the sample point. There are (at least) two ways to solve the problem:

1. The bit value is sent continuously on the communication in the part of the CAN frame where multi-transmissions are allowed. Any second protocol signal is blocked.
2. The CAN bit values and dominant edges are modulated in a way that they can be filtered out at the right time and position of the CAN frame.

Bit embedding

The CAN controller transmits a CAN frame to the VCC that reduces the bits to dominant edges, bit values, and bit type. When the "Protocol Multiplexer" receives an SOF Sync_Seg, it synchronizes the embedded protocol to the CAN bit timing. The time between the encoded Sync_Seg from the VCC is used for transmitting the second-protocol information. The symbol stream is sent to the "Sig-nal Processing Unit" (SPU) and transmitted on the physical medium.

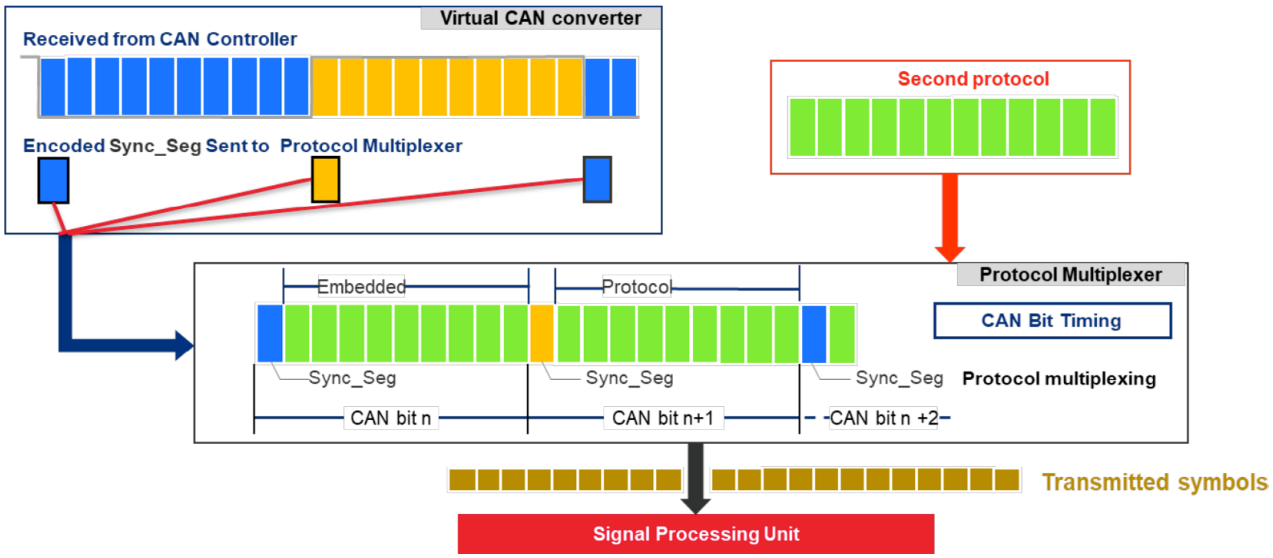


Figure 14: Integrated protocol multiplexing bit embedding transmission (Source: Kvaser)

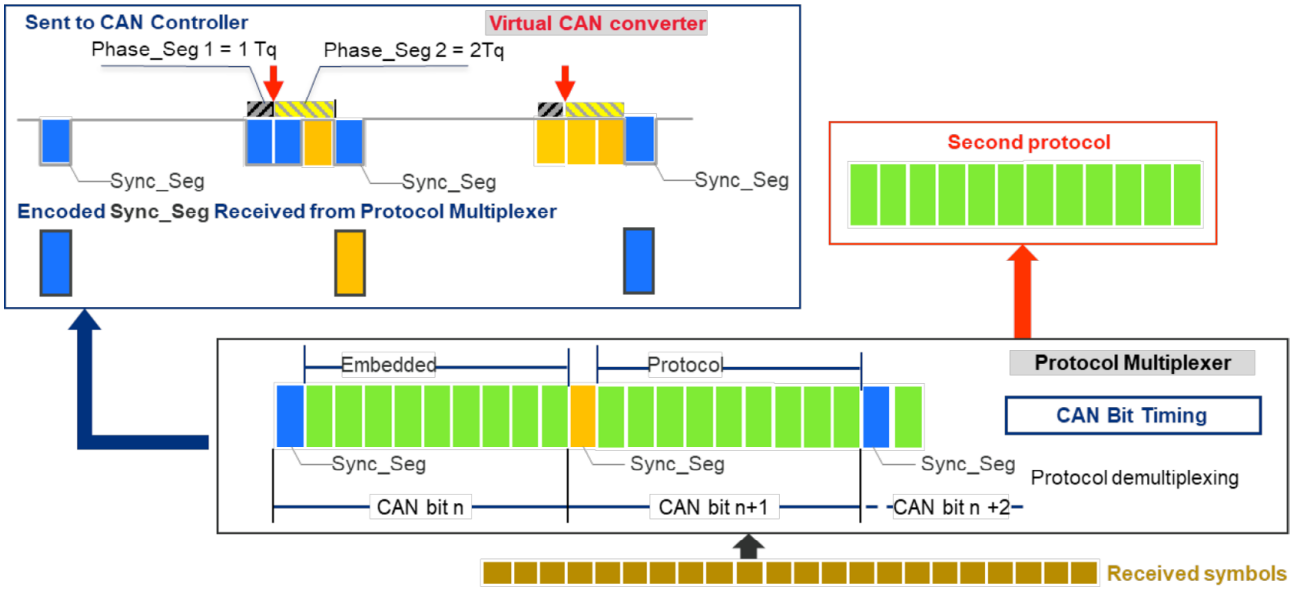


Figure 15: Receiving modules process the signals in reverse order (Source: Kvaser)

Receiving modules process the signals in reverse order (see Figure 15). A symbol stream is received from the SPU and demultiplexed by the "Protocol Multiplexer". The CAN symbols according to the RCP are fed to the VCC and the symbols of the second protocol to the second protocol handler. The VCC recreates the CAN bits by decoding the received Sync_Segs. A point-to-point connection between the CAN controller and the VCC (which is generating ghost edges) ensures that the CAN controller and the VCB are in perfect synchronization. This being the case, the sample point can be moved as far as possible to the end of the bit, i.e., the Phase_Seg 1 is one TQ and the Phase_Seg 2 two TQ long.

Embedding fake frames

Another method to embed a second protocol is to send a fake CAN frame. One CAN identifier is reserved for this purpose and known by the PMUX and the VCC. When the PMUX wants to transmit something from the second protocol, it starts the transmission as a CAN frame with the reserved CAN-Identifier and a DLC (data length code) that will create a time slot until the EOF that can be occupied by second protocol bits.

All VCCs will receive the first part and transmit it to their respective CAN controller (bit by bit). If the fake frame wins the arbitration, the respective VCC will create a fake frame and send it to its CAN controller. The PMUX will use the time until EOF for transmission of the second-protocol bits. More than one CAN-Identifier can be reserved and used to distinguish frames of specific kinds and addresses and/or to identify a third or fourth protocol, etc.

Longer time slots for second protocol frames

Sometimes the control system uses just a fraction of the available bandwidth. One way to make a window for the second protocol is to set up several dummy frames. These dummy frames are sent back-to-back to the CAN controller. If the dummy frame has the CAN-Identifier 0, it will block the CAN controller from any attempt to transmit a frame. During runtime of a control system, i.e. the CAN system, some frames require the highest priority ▷

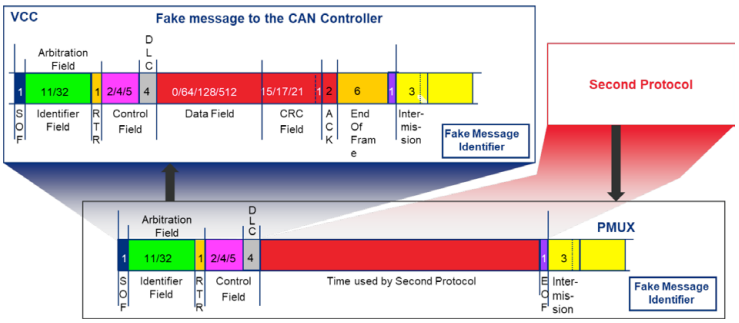


Figure 16: Integrated protocol multiplexing fake frame embedding transmission (Source: Kvaser)

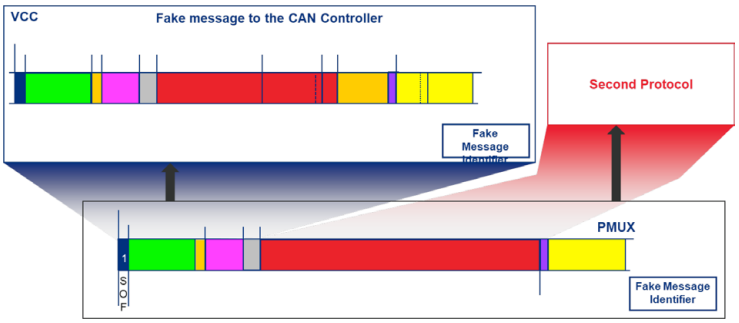


Figure 17: Integrated protocol multiplexing fake frame embedding reception (Source: Kvaser)

e.g., in catastrophic situations. The dummy frame should then have a lower priority. The VCC would then detect an attempt to transmit the alarm frame but too late for the PMUX to transmit it. The VCC then creates a bit fault to the CAN controller. The CAN controller will respond with an error frame and retransmission of the alarm frame. The PMUX will now get an SOF, abort the second protocol frame and continue transmitting the alarm frame.

Summary of the "Reduced CAN Protocol"

The RCP generates the position and values of the essential bit quanta in a CAN frame and the different frame structures from start of frame (SOF) to end of frame (EOF) according to the chosen CAN format for the actual system.

When the CAN controller transmits, the VCC mirrors the TX signal to the RX connection and conveys the following reduced CAN signals to the PMUX:

1. The Sync_Seg of every bit.
2. The bit value by one or more of the alternatives below
   a) modulating the Sync_Seg
   b) modulating the first TQ of the Prop_Seg
   c) last TQ of Phase_Seg 1 and first TQ of Pase_Seg 2
3. Encoded stuff bits
4. Encoded SOF
5. Encoded EOF

At reception, the PMUX transmits the CAN primitives according to the RCP to the VCC. The VCC converts the primitives to CAN bits on the go and feeds the signals to the CAN controller. The VCC also checks the bit flow according to the stuff bit rules. In case of a mismatch, it transmits an error flag both to the CAN controller and the PMUX.

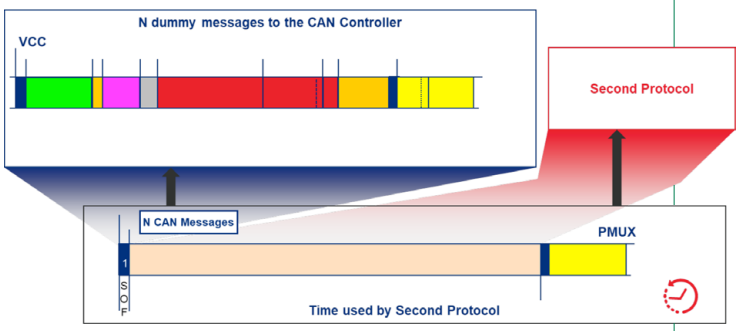


Figure 18: Integrated protocol time multiplexing (Source: Kvaser)

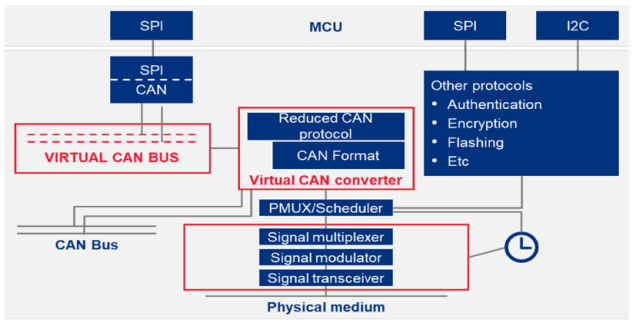


Figure 19: Further possible enhancements (Source: Kvaser)

A further enhancement

Until now it has been assumed that the ECU has a CAN controller. This makes it easy to apply the invention as old ECUs can be used without any modification. For completely new designs, it could be advantageous to move the CAN controller to the transceiver unit. There are already many such designs both for Classical CAN and CAN FD, denoted as standalone CAN controllers. In this case, the transceiver unit will have two modes, a "CAN only mode" and a "CAN embedded mode." In CAN only mode, the PMUX is bypassed and the signals according to the RCP are sent directly to the bus lines. In this way the kernel of the control system can be developed in a straight forward way using well proven CAN tools. As only the essential signals are transmitted on the CAN, detailed time analysis of the communication can be made. In a later stage of the development, when other features are added to the communication, the CAN control part can be verified by analyzing the "Virtual CAN Bus" by examining the RCP signals from the PMUX. Other features can be added such as frame schedulers, system clocks, etc. (see Figure 19).

Author

Lars-Berno Fredriksson
Kvaser
lbf@kvaser.com
www.kvaser.com