Virtual CAN FD bus

Lars-Berno Fredriksson KVASER AB 181015

Summary

CAN is an excellent and well-proven protocol for the control task in a Distributed Embedded Control System. It is not well suited for requirements that relate to the security of messages and the re-flashing of ECUs that demand a high bit rate. The primary reason for this is that CAN uses a very simple bit transfer method and a low bitrate at the arbitration phase. The bit transfer during the arbitration phase is a remnant of the technology around at the time when CAN was conceptualized in the 80s.

The bandwidth required for pure control tasks is governed by the dynamics of the machinery to be controlled. CAN is good enough for controlling any car of the future, as a car will have roughly the same size and dynamics. Other tasks, however, such as encryption, transfer of raw data from advanced sensors and cameras, RADAR, LIDAR, etc. and flashing of ever-growing software will generate an ever-increasing request for bandwidth. A solution to this dilemma is the “Virtual CAN bus” where signals to and from ordinary CAN Controllers are multiplexed by ”intelligent” transceivers that support one or more modern high-speed communication protocols running on the same physical layer. In this way we can keep CAN for control tasks and continuously take advantage of the newest technology for bandwidth-hungry tasks.

CAN is, in a sense, a unique protocol: The transmitter uses 100% of the bus bandwidth but the receivers use just a fraction of it. The proposed solution to the problem of CAN’s low bandwidth takes advantage of this peculiarity. It introduces a “Virtual CAN Bus” (VCB), executed in an “intelligent” transceiver unit – a Virtual CAN Converter (VCC) – connected to an ordinary CAN Controller. The “intelligent” 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 thing 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 RX connection.

By using a Virtual CAN Bus, we separate the control task 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.

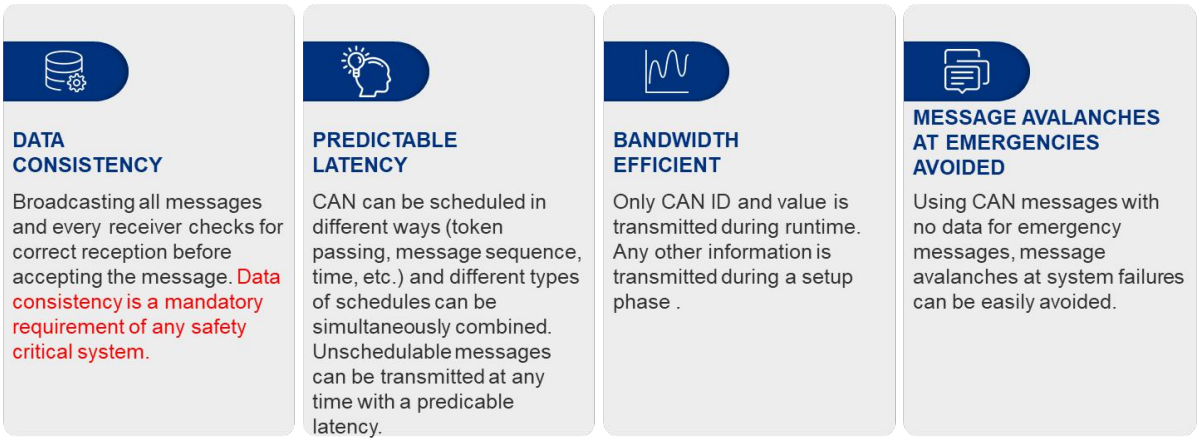
1. CAN Background

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 messages and determine if the message should be received or not.
2. All nodes participate in error handling. No application receives a message if all nodes have not found it to be correct.
3. Bitwise arbitration at message collisions. No message is lost, and the maximum latency of any message 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 messages.
- On top of that, the No Address feature makes the messages short as only a message identifier and a value are transported during runtime. The required bus 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 developing CAN Flexible Data rate in 2011, resulting in the ISO standard 11898-1:2015. With the arrival of CAN FD, flashing could be done faster, but more requirements were then brought to the table. Fear for 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 six, but it was later shown that a data bit mistaken for a stuff bit and vice versa might result in the same message length and the same CRC 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.

Since the birth of CAN, communication technology has developed tremendously, but CAN FD has not taken advantage of that. It is still using the most primitive amplitude modulation for bit transmissions.

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.

2. The solution

2.1 Overview

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 1553 systems running at 1 Mbps. Does a car really need a faster control system than a high performance, inherently unstable jet fighter? Mil 1553 is less efficient than Classical CAN, so we can expect CAN FD to match any control demands of the future.

Modern, fourth-generation jet fighters use MIL-STD-1553 at 1 Mbps, which is less efficient than CAN.

Does a car really need a faster control system than an inherently unstable jet?

CAN is more than adequate for control tasks. The driving force for a higher bandwidth for CAN is instead due to non-control issues.

CAN is good enough as it is

- Most trucks are successfully controlled using J1939 at 250 kbps and less than 50% bus load
- Scheduling, using SPN as key to CAN ID and increasing the bus load to 99% would extend the bandwidth at least five times

The driving force towards higher bit rates are non-control issues

- Authentication
- Encryption
- Flashing
- General safety demands
  - Bit error in every ten message has to be expected

We should find a way to run an “old-fashioned” CAN system with a low bitrate on a modern physical layer, multiplexed with another protocol with a high bitrate 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:

AN EFFICIENT SYSTEM ARCHITECTURE

Most important is that the control system is robust and safe during any normal situation

Let the control engineers do their best at what they are trained to do

Leave the security problems to the security experts

They can do whatever they like as long as the control messages are sent and received in due time.

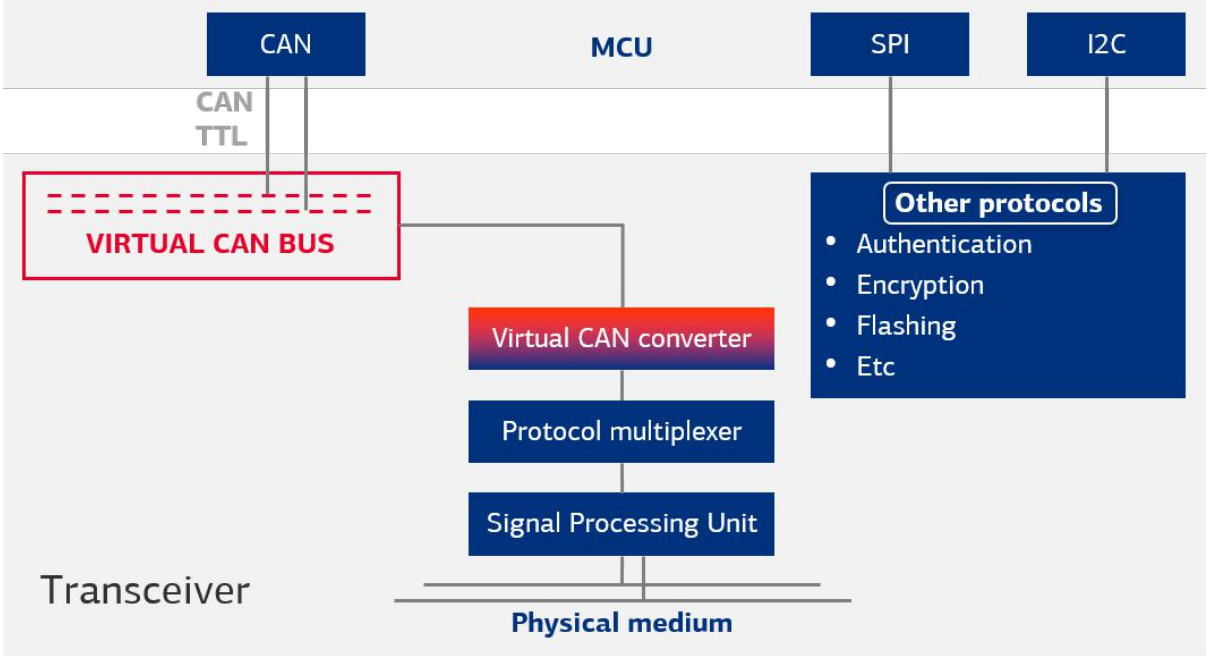
Separate MCU flashing from control tasks

A system should have at least two modes: Control mode and flashing mode. By this the system will be more dependable in control mode and the flashing can be much faster.

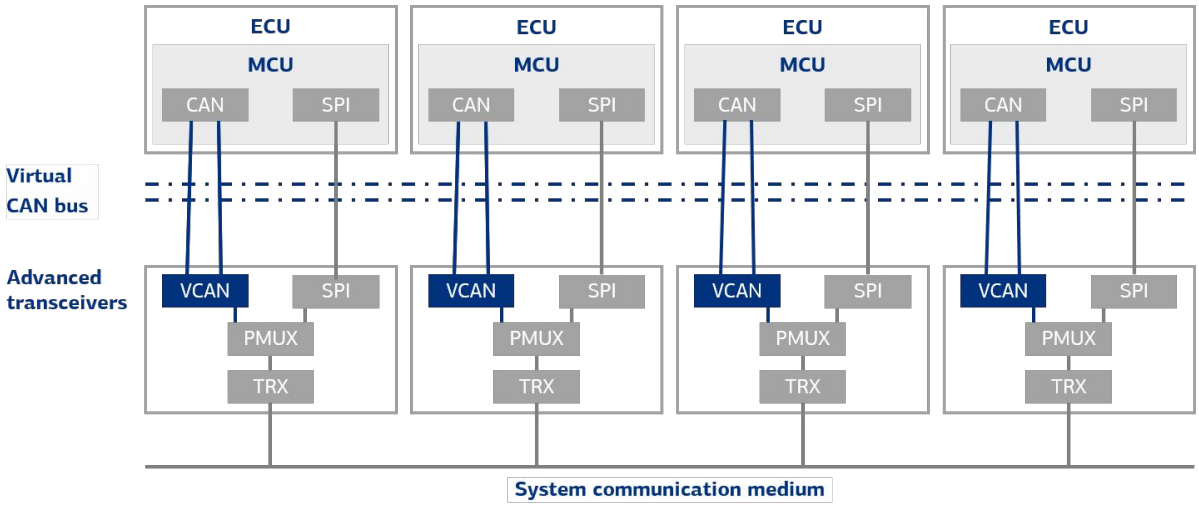
Hence, let us:

- 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 establish a Virtual CAN Bus.



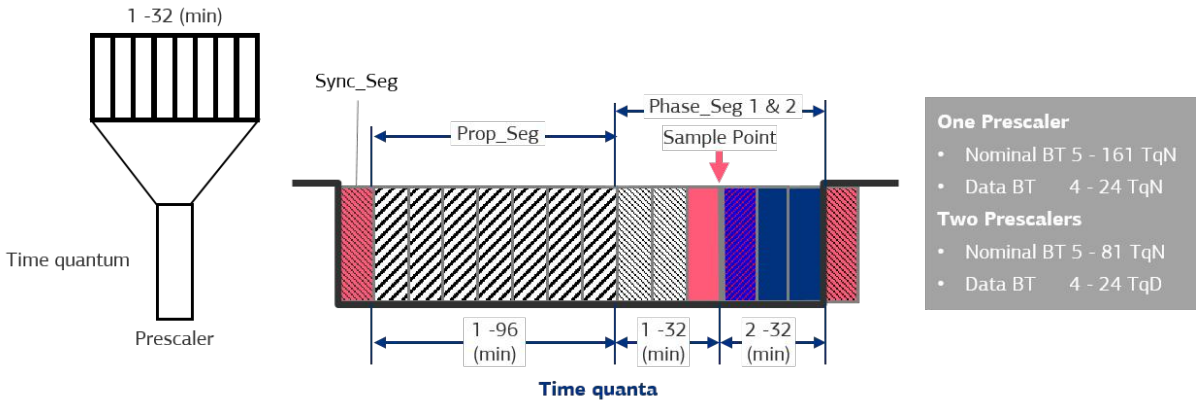
THE RESPECTIVE CAN CONTROLLER SENDS/GETS TTL SIGNALS ACCORDING TO THE CAN FD PROTOCOL



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 messages in a specific way (more about that later).

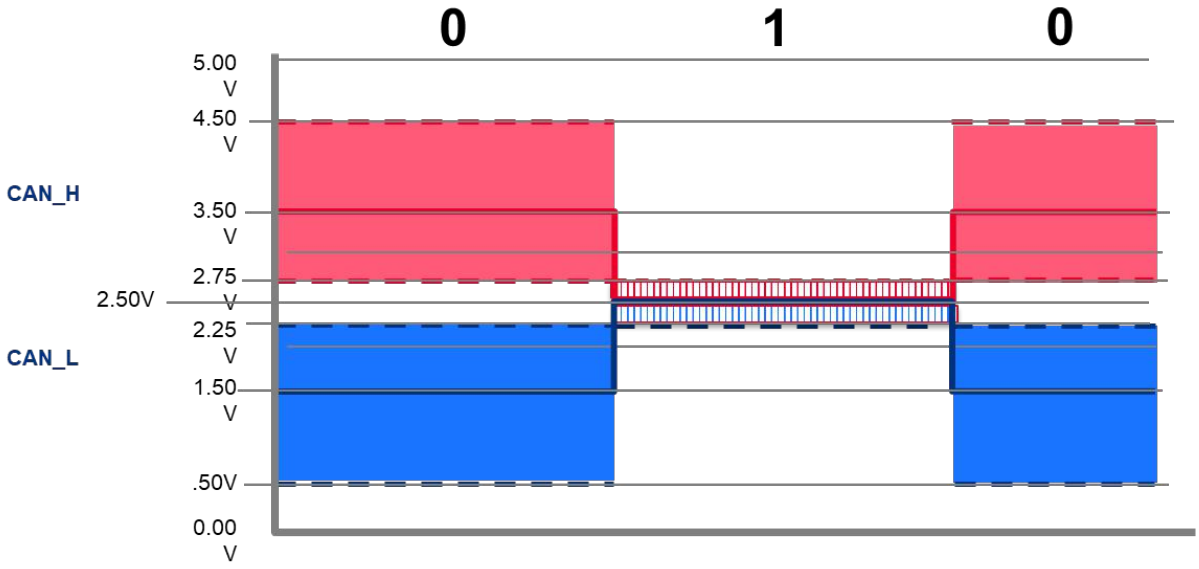
2.2 Essential CAN features for a Virtual CAN Converter

The figure below shows the construction of a CAN FD bit.

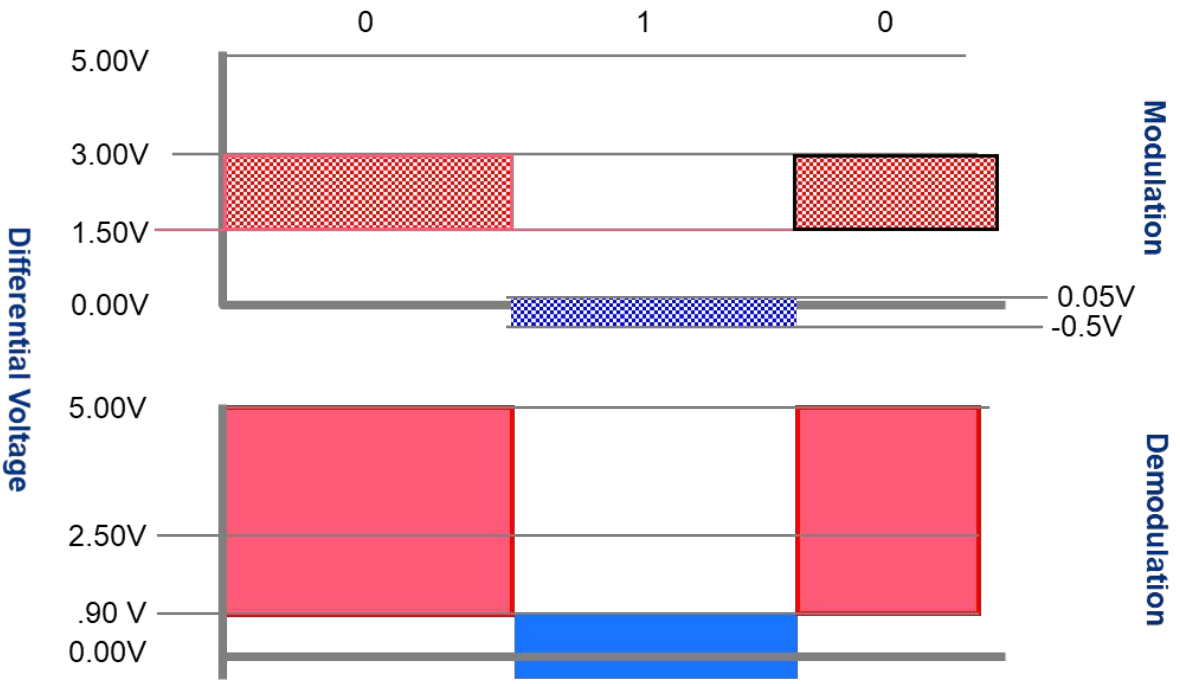


According to the ISO standard 11898-1:2015, a CAN FD bit is constructed on Time Quanta. A Time Quantum (TQ) represents a number of clock cycles. A bit starts with a Sync_Seg of one TQ followed by a Prop_Seg, a Phase_Seg 1 and a Phase_Seg 2. The value of the bit is sampled at the Sample Point located between Phase_Seg 1 & 2.

The signal on the bus is 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) m and CAN low (CAN_L) around 2.5 V.

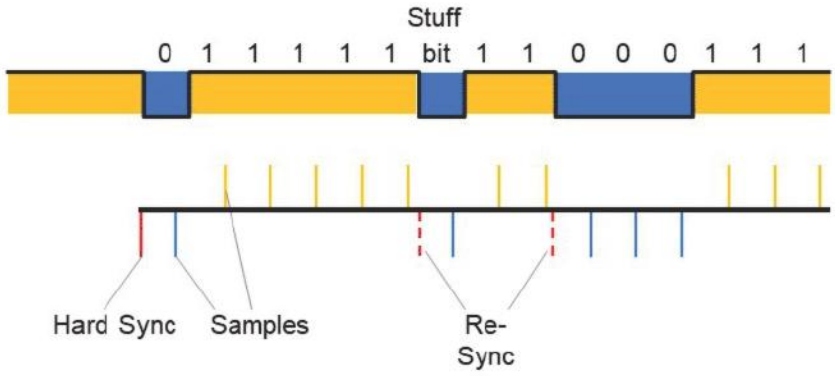


At the Sample Point, the bit value is decided by the differential voltage between CAN_H and CAN_L.



A unique characteristic of the CAN protocol is that the transmitter occupies 100% of the bus 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 is a Non Return to Zero (NRZ) type of protocol 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.

THE CAN TRANSMITTER OCCUPIES 100% OF THE BANDWIDTH WHEN TRANSMITTING

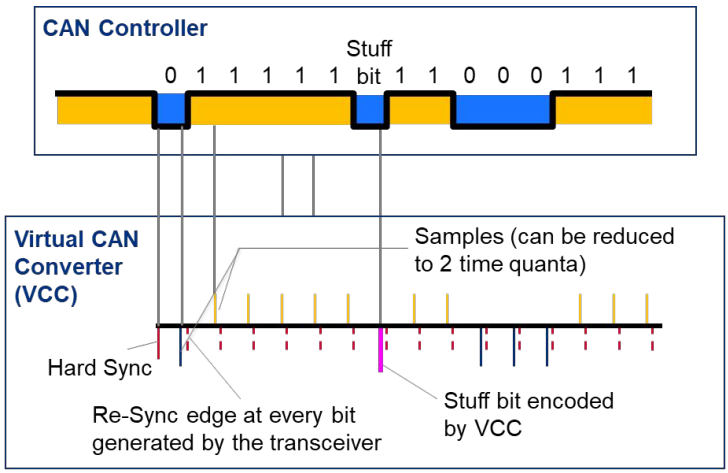


The CAN receivers use a fraction of the bandwidth, the Sync_Seg and the Sample Point. The rest of the bandwidth is ignored.

KVASER

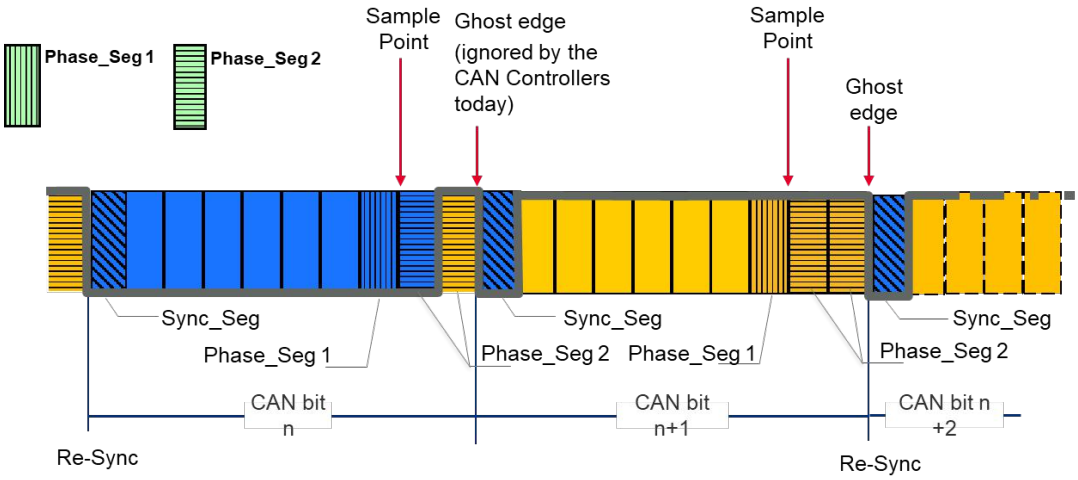
2.3 The reduced CAN Protocol

Implemented in the Virtual CAN Controller 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 protocol, 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 Controller 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

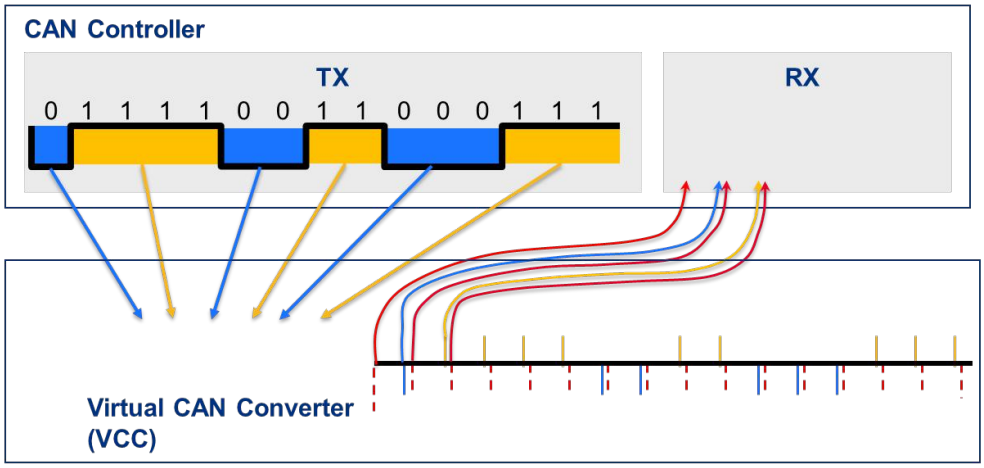
The RCP also generates a dominant edge at the end of every bit, i.e. a “Ghost Sync_Seg” at each bit instance where the CAN Controller is not generating a recessive bit followed by a dominant bit (1/0). This keeps the CAN Controller in synch with the VCC.



According to the CAN protocol, Phase_Seg 2 should be no shorter than two TQ. That 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 0s 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 but if not, this VCC feature can be omitted.

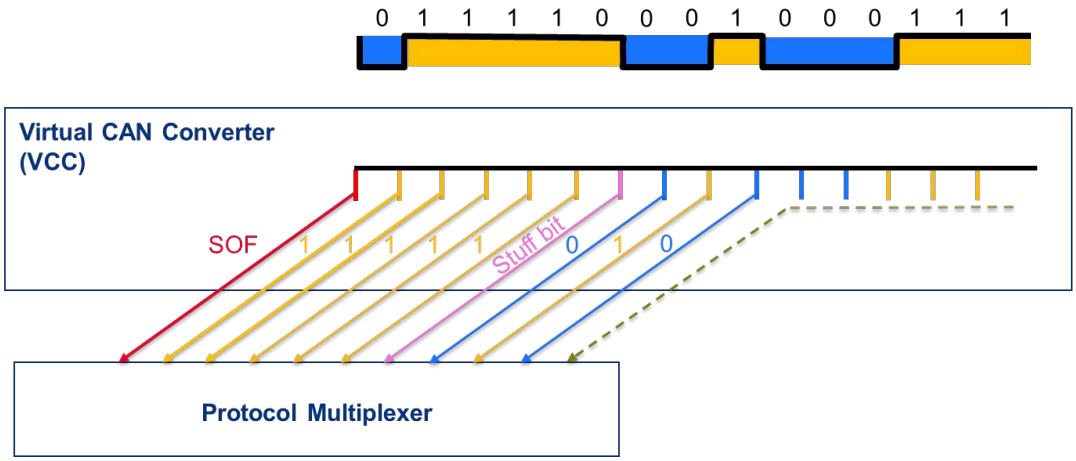
THE VCC RECEIVES THE FULL BITS FROM THE CAN CONTROLLER ON THE TX LINE BUT TRANSMITS BACK ONLY DOMINANT EDGES AND THE BIT VALUES AT THE SAMPLING POINT ON THE RX LINE



When the CAN Controller transmits, the VCC receives full bits i.e., a differential voltage shifts when two consecutive bits have 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 O 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.

THE VCC TRANSMITS ONLY ENCODED SYNC_SEGS TO THE PMUX.

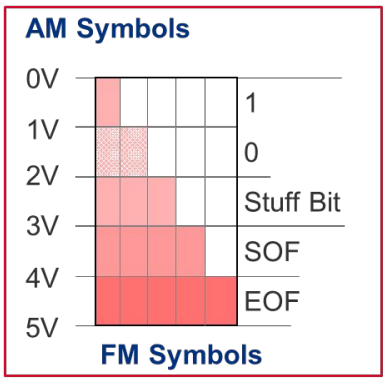


As it will be shown later, the length of a CAN message can be very important. It is therefore an advantage if the VCC in communication with the Protocol Multiplexer generates an encoded SOF bit at the beginning of a message and an encoded EOF bit at

the end of the message. 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.

CAN BIT 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.



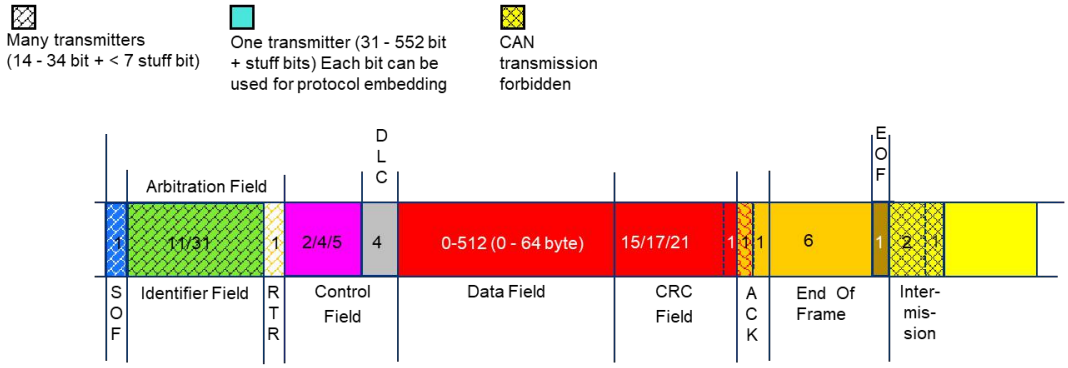
Error and stuff-bit checking can be done by the VCC.

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.

3 Protocol multiplexing

The basis for the protocol multiplexing is the CAN message 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 Multiplexers and the Signal Processing Units, so the VCCs can accurately create the Virtual CAN Bus. A CAN Message Frame starts with the dominant SOF bit followed by the 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.

CAN MESSAGE FRAME



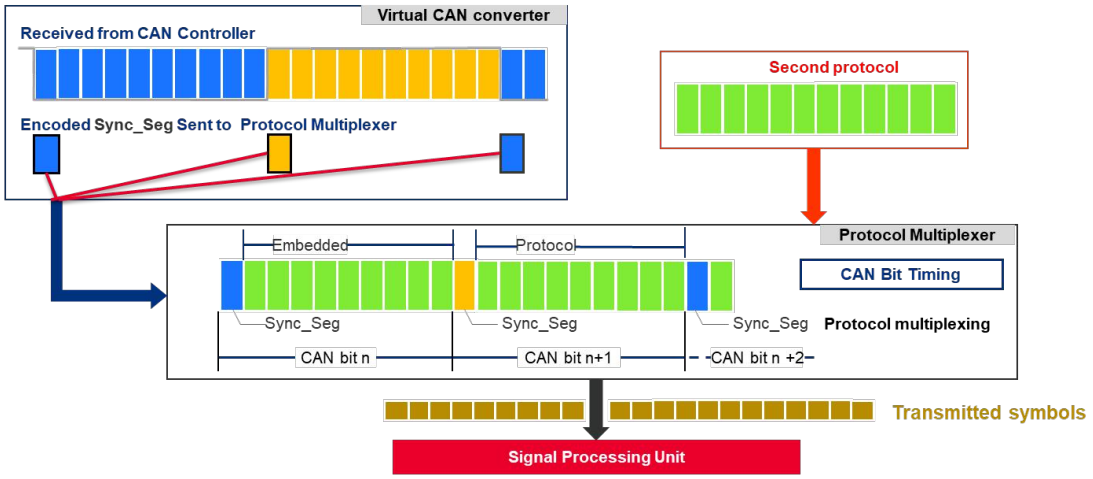
This can cause some difficulties for the Protocol Multiplexer and the RCP. When a bit value is received when more than one CAN Controller is transmitting, the bit value can appear anywhere in the Prop_Seg of the CAN bit. The RCP has to catch it and transmit it to the CAN Controller at the Sample Point. 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.

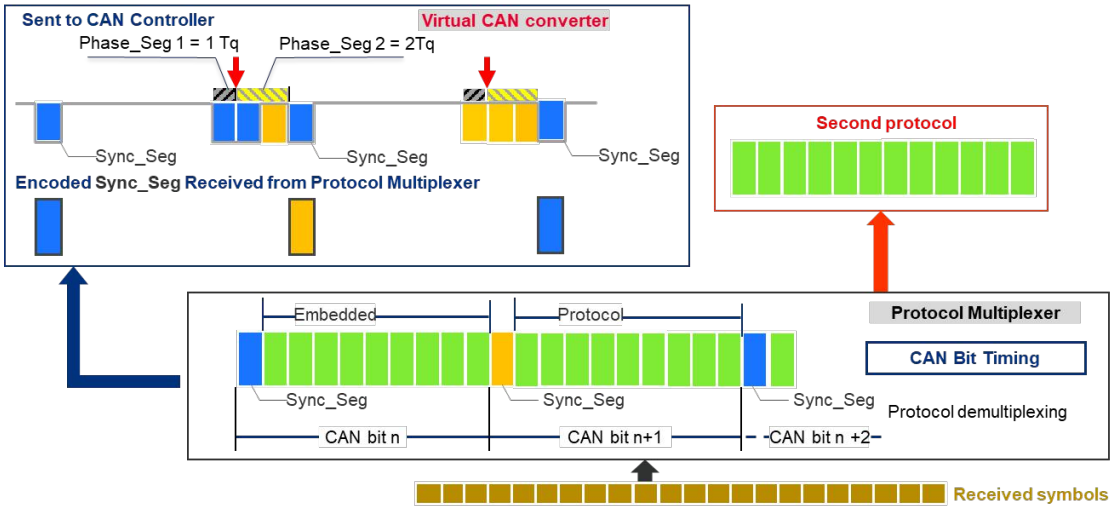
3.1. Bit Embedding

The CAN Controller transmits a message to the VCC that reduces the bits to dominant edges, bit values and bit type. When the Protocol Multiplexer receives a 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 Signal Processing Unit and transmitted on the physical medium.

INTEGRATED PROTOCOL MULTIPLEXING
BIT EMBEDDING TRANSMISSION

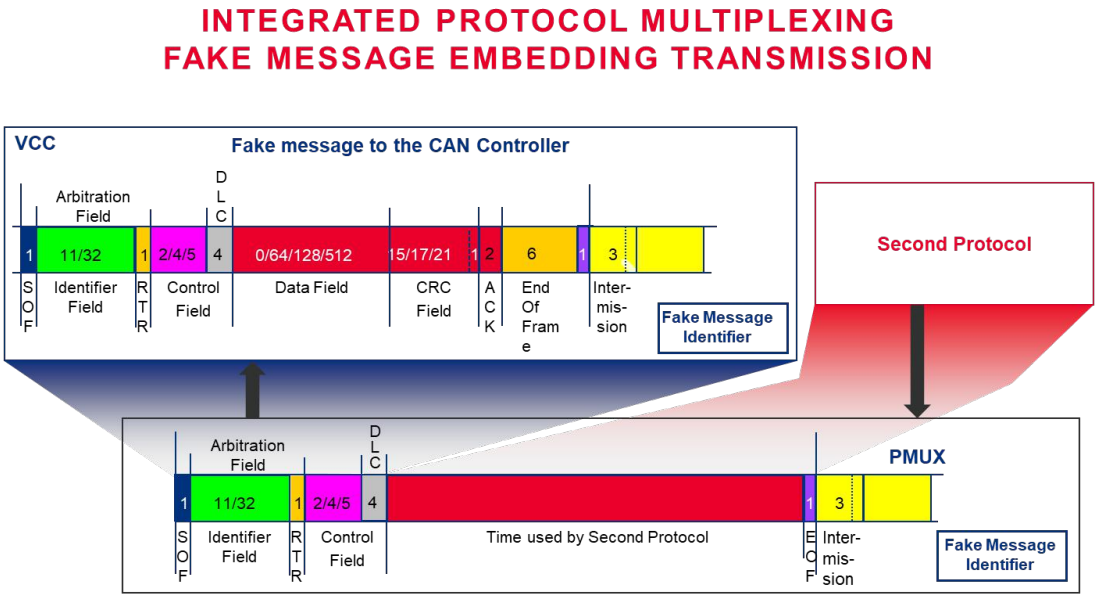


Receiving modules process the signals in reverse order. A symbol stream is received from the SPU and demultiplexed by the Protocol Multiplexer. The CAN symbols according to the RCP are fed to the VCC and the symbols of the second protocol to the second protocol handler. The VCC recreates the CAN bits 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 Virtual CANbus are in perfect synchronisation. 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.

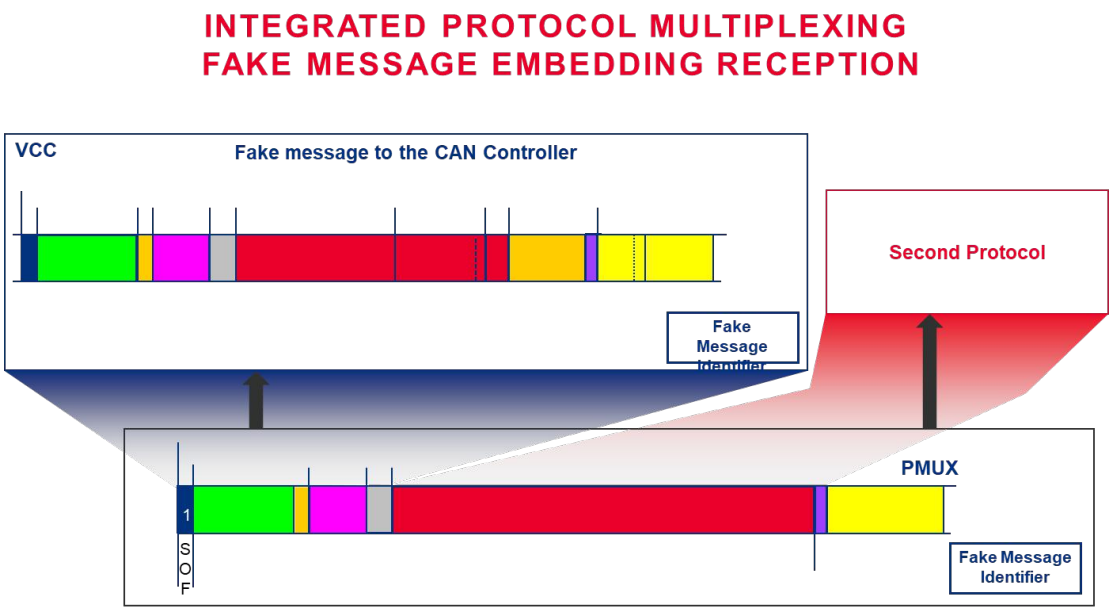


3.2. Embedding Fake Messages

Another method to embed a second protocol is to send a fake CAN message. 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 message with the reserved identifier and a DLC that will create a time slot until the EOF that can be occupied by second protocol bits.

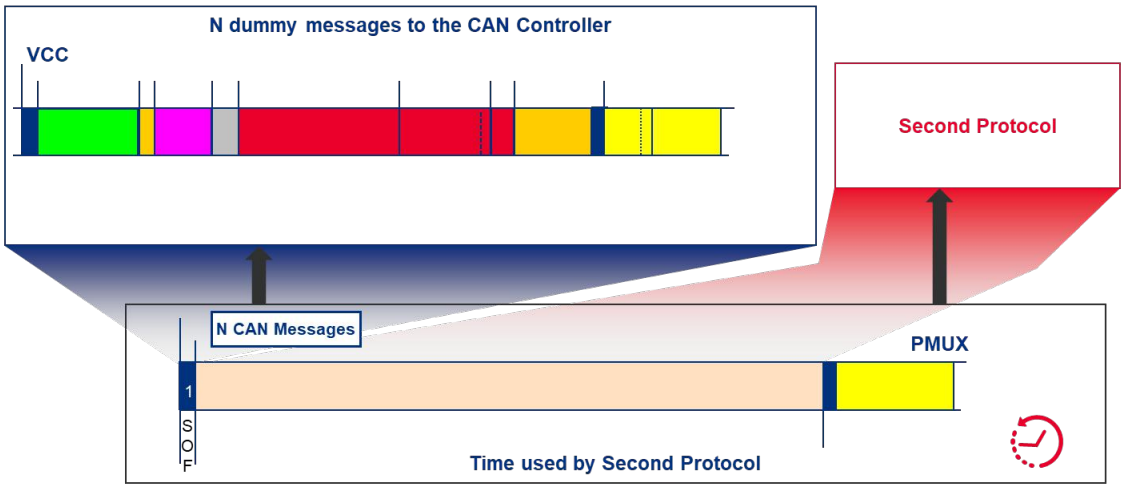


All VCCs will receive the first part and transmit it to their respective CAN Controller (bit by bit). If the fake message wins the arbitration, the respective VCC will create a fake message and send it to their CAN Controller. The PMUX will use the time until EOF for transmission of second protocol bits.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



3.3. Longer time slots for Second Protocol messages

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 messages. These dummy messages are sent back-to-back to the CAN controller. If the dummy message has the CAN Id Std0, it will block the CAN Controller from any attempt to transmit a message. During runtime of a control system, i.e. the CAN system, some messages require the highest priority e.g., in catastrophic situations. The dummy message should then have a lower priority. The VCC would then detect an attempt to transmit the alarm message but too late for the PMUX to transmit it. The VVC then creates a bit fault to the CAN Controller. The CAN Controller will respond with an error frame and retransmission of the alarm message. The PMUX will now get a SOF, abort the Second Protocol message and continue transmitting the alarm message.



.

4 Summary of the RCP

The Reduced CAN Protocol generates the position and values of the essential bit quanta in a CAN message 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 :

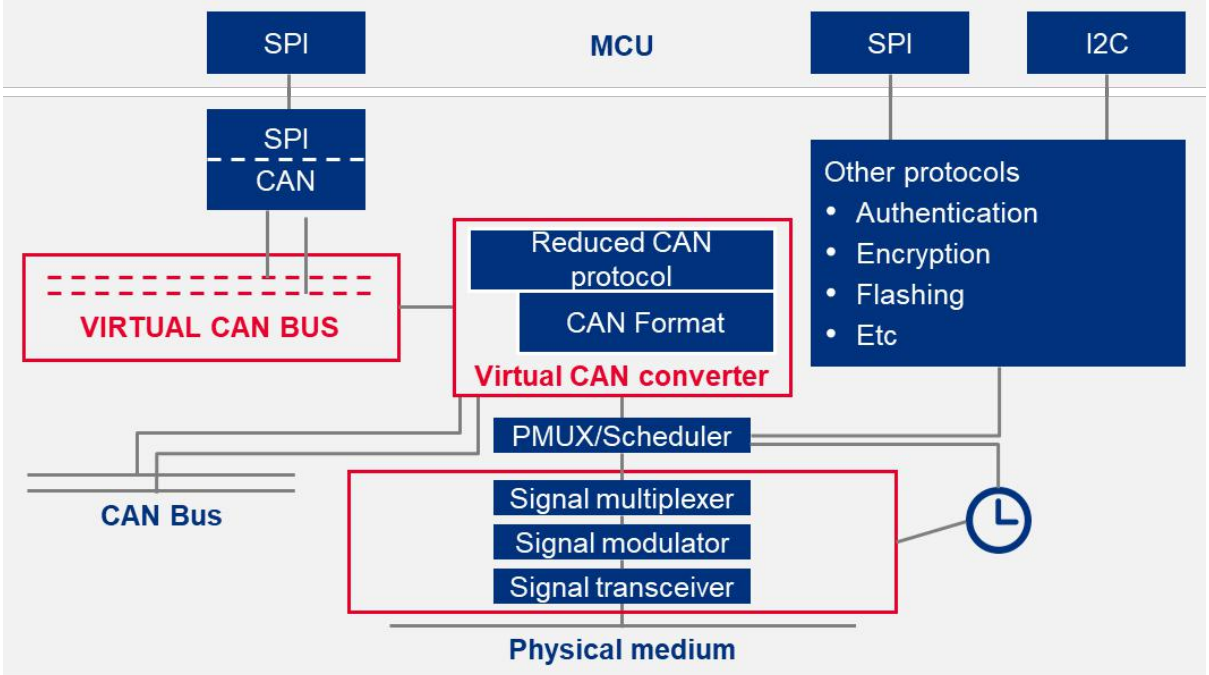
1. The Sync_seg of every bit.
2. The bit value by one or more of the alternatives below
   a) modulating the Sync_Seg
   b) modulating the first TQ of the Prop_Seg
   c) last TQ of Phase_Seg 1 and first TQ of Pase_Seg 2
3. Encoded Stuff Bits
4. Encoded SOF
5. Encoded EOF

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 feed 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.

5 A further enhancement

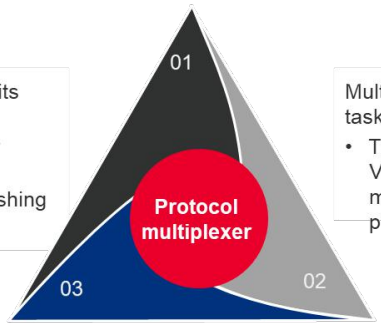
Until now it has been assumed that the ECU has a CAN Controller. This makes it easy to apply the invention as old ECUs can be used without any modification. For complete new designs, it could be advantageous to move the CAN Controller to the transceiver unit. There are already many such designs both 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. 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 bus, 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 message schedulers, system clocks, etc.



THE PROTOCOL MULTIPLEXER IS THE KEY TO AN EFFICIENT USE OF THE PHYSICAL LAYER

- The physical layer can be used to its maximum
- Different protocols can be used for different tasks and situations
  - High speed point-to-point for flashing
  - CAN FD for control tasks



Multiplexed protocols for different parallel tasks

- The Reduced CAN protocol and the Virtual CAN bus makes it possible to multiplex CAN into another high-speed protocol

Dynamic switching of protocols at specific situations

- Relocating bandwidth and tasks
- Maintenance
- Troubleshooting

KVASER