June 2014 B 25361

CAN Newsletter

Hardware + Software + Tools + Engineering

System design

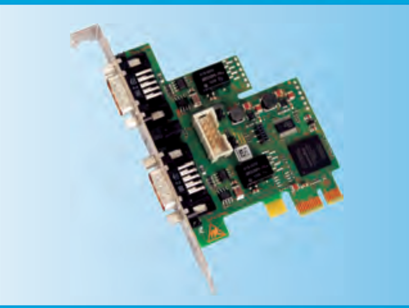
CAN FD from an OEM's point of view

The susceptibility of CAN to latent faults

CAN FD: Impact on system design

www.esd.eu

CAN-PCIe/402



CAN-USB/400

CAN-PCI/400

Ethernet

CAN-USB/2

Gateways

All you CAN plug

CAN-PCIe/402

- up to 4 high performance PCI Express CAN interfaces
- DMA busmaster
- Powered by esd Advanced CAN Core (esd-ACC)
- MSI (Message Signaled Interrupt) support
- Electrically isolated
- Provides high resolution hardware timestamp

CAN-USB/400

- 2 high performance CAN-USB interfaces
- Powered by esd Advanced CAN Core (esd-ACC)
- USB 2.0 with high speed data rates of 480 Mbit/s
- Electrically isolated
- Provides high resolution hardware timestamp
- Error injection for advanced diagnostic
- IRIB B timecode as option

CAN-PCI/400

- up to 4 high performance CAN interfaces
- Powered by esd Advanced CAN Core (esd-ACC)
- Electrically isolated
- Provides high resolution hardware timestamp
- Error injection for advanced diagnostic

CAN-PCI104/200

- PCI104-CAN interface
- One or two CAN interfaces for PCI104 bus

EtherCAN

- 10/100 BaseT ETHERNET-CAN Gateway
- Electrically isolated
- Configuration and Diagnostics by webbrowser

CAN-USB/2

- CAN-USB interface
- Intelligent CAN interface with ARM 7
- USB 2.0 with high speed data rates of 480 Mbit/s
- Electrically isolated
- Provides high resolution hardware timestamp

Gateways

- EtherCAT-CAN
- PROFINET-CAN
- PROFIBUS-CAN
- PROFIBUS-DeviceNet
- EtherNet/S7-CAN

Operating Systems

esd supports the real-time multitasking operating systems VxWorks, QNX, RTX, RTOS-32 and others as well as Linux and Windows 32/64Bit systems

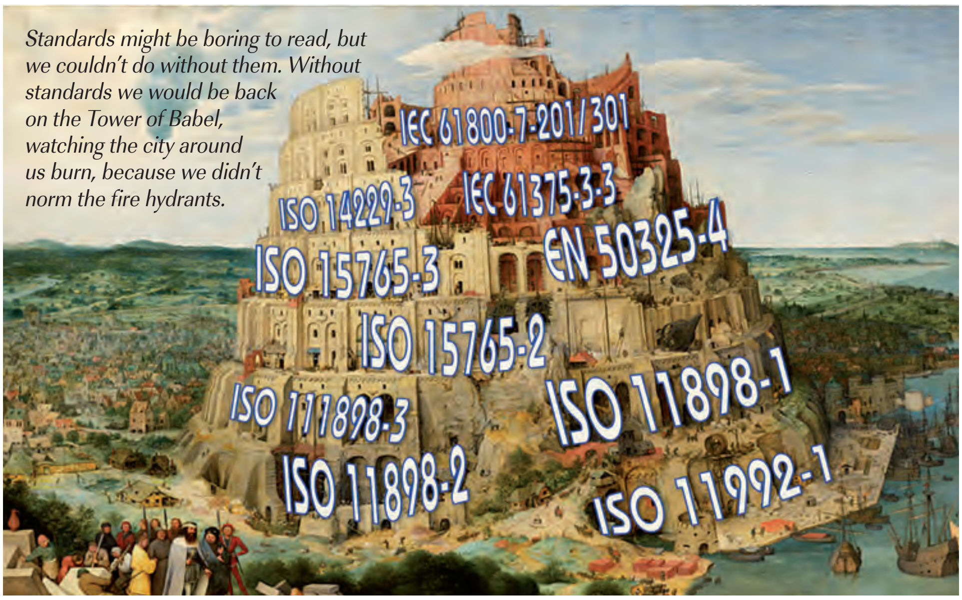
CAN Tools

- CANreal: Display and recording of CAN message frames
- CANplot: Display of online/offline CAN data
- CANrepro: Replay of pre-recorded CAN message frames
- CANscript: Python scripting tool to handle CAN messages
- COBview: Analysis and diagnostics of CANopen nodes

The tools are free of charge on the driver CD or can be downloaded at www.esd.eu

esd gmbh • Vahrenwalder Str. 207 • 30165 Hannover • Tel: (0511) 372 98-0 • Fax: (0511) 372 98-68 • info@esd.eu • www.esd.eu
esd electronics Inc. • 70 Federal Street • Suite 2 • Greenfield • MA 01301 USA • Phone: 413-7723170 • us-sales@esd-electronics.com • www.esd-electronics.us

The history of standardization and CAN in a nutshell



Standards might be boring to read, but we couldn't do without them. Without standards we would be back on the Tower of Babel, watching the city around us burn, because we didn't norm the fire hydrants.

The CAN protocol has been internationally standardized since 1993 in the ISO 11898 standard. These days, all basic CAN standards are under review. The ISO 11898-1 standard is going to be upgraded and to include the CAN FD protocol extension. The ISO 11898-2 high-speed transceiver standard and its younger sisters ISO 11898-5 (low-power mode) as well as ISO 11898-6 (selective wake-up transceivers) are going to be merged into one single ISO document, to make them consistent and easier to maintain.

We all know that standards are boring to read, but we can't work without them. It's especially dull to read standards that were not made for your needs. For example, the ISO 11898 series is written for chipmakers who want to implement the CAN protocol. Users should not read them. A better way to spend your time would be to buy one of the helpful books on the topic or to search for articles and basic information on the Internet.

We do need a shared way of communication, so that different implementations can exchange messages. This is the same with human languages: If we didn't agree on paper and envelope formats as well as character sets, we couldn't communicate through mail. There would be no normed keyboards and no letters that could be read without an interpreter.

Standardization

Standardization is as old as interactions within larger human communities. The first Chinese emperor, Qin Shi Huang (260 to 210 BC), standardized not just the Chinese characters, but also the system of units and measurements as well as the currency and the width of cart axles.

War has often driven standardization. More than 2500 years ago Heraclitus stated: "War is the father and king of all." In the American Civil War (1861 to 1865), one of the reasons for the victory of the Union against the South was the standardization of their rail tracks. The problem was the difference of track gauges: The Confederate rail network was mostly in the broad gauge format, only North Carolina and Virginia had mainly standard gauge lines. Southern railroads west of the Mississippi differed widely in gauge and were isolated and disconnected. During the Civil War, the Union government recognized the military and economic advantages of having a standardized track gauge. The government worked with the railroads to promote use of the most common railroad gauge in the U.S. at the time, which measured 4 feet and 8 ½ inches, a track size that originated in England. This gauge was mandated for use in the Transcontinental Railroad in 1864 and by 1886 had become the U.S. standard.

Another example for standardization during wars is standardized rifle parts. Standardized parts are interchangeable between guns. This was a revolutionary idea by Thomas Jefferson and Eli Whitney (mechanical engineer) in the late 18th century.

Likewise the foundation of the predecessor of the DIN (German standardization body) in 1917 had a military background: The German industry wanted to optimize production during World War I (1914 to 1918), which was mainly a material battle between Germany and France.

There are also civil examples of standardization benefits. In 1904, a fire broke out in Baltimore. To combat the flames, reinforcements from New York, Philadelphia and Washington (DC) came to Baltimore. After they arrived, they realized that their fire hoses could not be connected to the fire hydrants. Lesson learnt, the U.S. started a lot of standardization projects. In 1904, the ANSI (American National Standards Institute) was established. A few years earlier, the British Standardization Institute (BSI) had been founded.

An increase in international business demanded worldwide standardization. This was the birth of the IEC (International Electrotechnical Commission): The inaugural meeting was as early as 1906. Originally located in London (UK), the commission moved to its current headquarters in Geneva (CH) in 1948. One year before, 25 countries founded the ISO (International Standardization Organization) to deal with all "non-electrical" standards. The ISO predecessor had already been established in 1926, but it was suspended during World War II. Today the division of labor is more or less history, because electrical equipment is used in many industries and needs to be standardized. ISO standardizes electronics too, in particular for those industries that have non-electrical roots. That is why CAN is standardized by the automotive technical committees of ISO: cars were originally ▷

Editorial

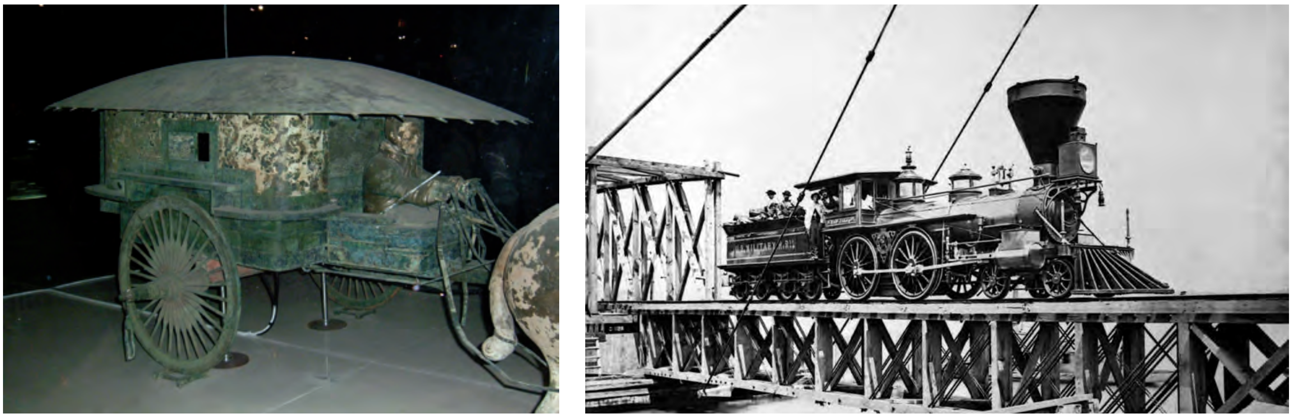


Figure 1: The war is the father of all things – the first Chinese emperor standardized among others the width of axles (left) and the Union government standardized rail tracks (right) (Photos: Wikipedia)

not defined as electric and electronics.

ISO is a voluntary organization whose members are recognized authorities on standards, each one representing one country. Members meet annually at a General Assembly to discuss ISO's strategic objectives. The Central Secretariat coordinates the standardization activities and publishes the ISO standards. There are over 250 technical committees and thousands of subcommittees, working groups, and task forces.

The IEC has a similar number of technical works. Some 10 000 electrical and electronics experts from industry, government, academia, test labs, and others with an interest in the subject develop the standards. IEC standards have numbers in the range from 60000 to 79999. The IEC is made up of members, called national committees (NC). Each NC represents its nation's electro-technical interests.

CAN-related standards in ISO and IEC

The CAN protocol was first described in a specification published by Bosch. Many people still use the terms CAN 2.0A and CAN 2.0B from back then. However, in 1993 the ISO 11898 standard was released, substituting all predecessors including the Bosch specification. The ISO standard comprised the CAN data link layer and CAN high-speed transmission. In 1995, the ISO 11898 standard was extended by an addendum describing the extended frame format using the 29-bit CAN identifier. Ten years after the publication of ISO 11898, the document was split into parts: The first part contained the data link layer and the physical signaling, while part two standardized high-speed transmission. Fault-tolerant, low-power transmission went into ISO 11898-3. At the same time, ISO started the standardization of CAN-based truck-trailer communication. The results are specified in the ISO 11992 series, which includes its on physical transmission solution and a higher-layer protocol based on J1939. This series also specifies dedicated parameter groups (signals assembled to CAN messages). All this standards are published under the roof of the Technical Committee (TC) 22. Within this TC there is also the ISO transport protocol standardized (ISO 15765-2), which is the base for several emission-related CAN-based diagnostic standards (ISO 15765-4). The ISO 14229-3 standard specifies unified diagnostic services (UDS) transmitted via CAN. Another CAN-related series is ISO 16844 standardizing tachograph communication for commercial vehicles. This standard is referenced by European regulations, but not much loved in the industry, especially not by truckers. They don't like to be watched electronically.

Just after the year 2000, Bosch started to extend the CAN protocol by a time-triggered protocol. It is an unanswered question if this is a session layer or if it doesn't fit in the OSI reference model at all. But it was standardized in ISO 11898-4 and is known as TTCAN. Up to now it has not made its way into the industry. Some chipmakers have implemented it, but it is not in use. The same happened to the ISO 15745-2-2 standard describing an XML-based framework for CAN-based networks, which has never been used in industrial automation systems. I was personally involved in its development, but understood little of what the IT experts were discussing. This is one of the standards that are just paperwork and eating memory space on computers. They are not really ecologically valuable, especially when considering the traveling. Anyway, standardization can sometimes be slow and eat a lot of work-time.

ISO TC 23 is the home of the ISO 11783 series (also known as Isobus), which standardizes the communication between tractor and agriculture equipment (so-called implements). This standard is based on the J1939 higher-layer protocol. The ISO 13628-6 standard describes the general requirements for subsea equipment using CAN networks to link sensors and meters to the subsea control unit. These CAN applications make use of the CANopen application layer and the related CiA 443 CANopen profile for SIIS level-2 devices.

In the IEC organization, there are also some CAN-related standards. First of all the IEC 61375-3-3 standard has to be mentioned: It describes the CANopen-specific implementation of CAN networks within rail vehicles; locomotives as well as coaches. This includes some physical layer specifications and some functions originally specified in CiA 301 and CiA 302. Another IEC standard, IEC 61800-7-201/301, specifies the CiA 402 CANopen profile for drives and motion controllers.

Of course there are also two European Cenelec standards related to CANopen: EN 50325-4 specifies the CANopen application layer and EN 50325-5 describes the CANopen Safety protocol extension. Cenelec and CEN, the European standardization bodies, were established in 1973 respectively in 1961. Today, they work in close cooperation with IEC and ISO, in order to avoid double standardization. Or in other words: the try not to re-invent the wheel.

No standards without sponsors and editors

To put it bluntly, all standardization activities are driv- ▷

en by interests of individuals or companies – sometimes both. Bosch backs the current activities regarding the ISO 11898 series. The C&S Group performing the related conformance testing of CAN silicon supports all conformance test plans. Of course, the market-leading semiconductor manufacturers and some carmakers spend a lot of effort on pursuing CAN standardization.

The editor of ISO 11898-1 is Florian Hartwich from Bosch. During the development of the document there were many contributions from different experts, in particular from GM, Mercedes, and Renesas. The related CAN conformance test plan, standardized in ISO 16845-1, was edited by Andreas Meidrodt from the C&S Groups. A college of him, Christoph Wosnitza, does the paperwork for ISO 16845-2, the conformance test plan for ISO 11898-6 (selective wake-up CAN transceivers). Bernd Elend from NXP volunteers as editor of the harmonized high-speed transceiver standard (ISO 11898-2). Without these editors, who also spend some of their free time on editing the documents, the standards would not meet the deadlines given by the rules of ISO.

Every ISO or IEC group needs a convener and a secretary. The secretary is normally a representative of a national standardization body. Volunteers for conveners are always welcome, because this position is unpaid. But sometimes, there are political discussions and arguments about who gets which position, because no country or company should dominate in a group. Traditionally, Germany is strongly represented in automotive-related committees. Standardization does not only have a technical dimension; it also has a political one.

> *“All standardization activities are driven by interests of individuals or companies.*

Standardization is slow

The process of standardization is sometimes slow compared to the development of technology. Because of this, technology is often introduced first and standardized later. Standardization is slow, because all parties should have the chance to comment on the provided drafts and proposals. On the other hand, the slow process avoids standardizing technology that disappears again soon. In the beginning, I found it harrowing to wait. In the meantime, I have learned to be more relaxed and patient. For example, when we started to standardize the CAN FD protocol, only Bosch and some other experts actually developed the protocol. But while we were preparing the committee draft for voting, we received a lot of valuable ideas and comments from other experts. In the end, this improved the CAN FD protocol and will increase its acceptance in the industry. Of course, the standardization process should not be so slow that the document is never published. Therefore, ISO and IEC establish project deadlines that have to be met.

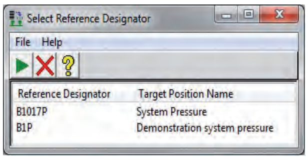
Still, it is boring to read standards. In particular, if you are not familiar with the specific standardization language. But we need these formal rules on how to write standards: we need a standard that tells us how to create standards. One of the reasons for the need is to avoid misunderstandings and misinterpretations when translating the standard into different languages than the two official ones: English and French. ◄

The history of standardization and CAN in a nutshell

Standards might be boring to read, but we couldn't do without them. Without standards we would be back on the Tower of Babel, watching the city around us burn, because we didn't norm the fire hydrants.

Application

System design



Tools

Business

Imprint

Publisher
CAN in Automation GmbH
Kontumazgarten 3
DE-90429 Nuremberg

publications@can-cia.org
www.can-cia.org
www.can-newsletter.org

Tel.: +49-911-928819-0
Fax: +49-911-928819-79

CEO Holger Zeltwanger
AG Nürnberg 24338

Printed in Germany
Hard copies: 3000
Soft copies: 3000

Editors
(pr@can-cia.org)

Annegret Emerich

Cindy Weißmüller

Holger Zeltwanger
(responsible according to the press law)

Layout
Nickel Plankermann

Advertising manager
Gisela Scheib
(responsible according to the press law)

Distribution manager
Julia Adolf

We mourn the passing of Arnulf Lockmann

On March 26, 2014, Arnulf Lockmann died unexpectedly. He was a member of Janz Tec's board of directors and he had been CiA's Business Director since 2005. Only a month ago he was re-elected unanimously during the annual CiA General Assembly. Arnulf Lockmann was born in 1956. He worked for Janz Tec for more than 25 years. I knew him since 1987, when he wrote articles about VMEbus products for the technical magazines I worked for. He was not just a business partner for me: We spent many evenings together after CiA meetings and business events, like the last international CAN Conference in Paris.

I appreciate him as an involved and dedicated member of the CiA Board of Directors. He was always reliable in all his contributions to the CAN community, but he was also companionable and humorous. With him, the CAN community lost one of its most committed fellows. I will miss him as competent business associate, but also as a friend. We had many inspiring talks, not just about CAN technology and markets.

Holger Zeltwanger

The disabled set sail

Sailing is not an activity that can easily be done with a handicap. Sailors typically need mobility to steer a boat. An Arduino-compatible CAN architecture for sailing applications is set to change that.

Authors
Kévin Bruget
kevin.bruget@ensta-bretagne.org

Benoît Clement
benoit.clement@ensta-bretagne.fr

Olivier Reynet
oliver.reynet@ensta-bretagne.fr

Ensta Bretagne
2, Rue Francois Verny
FR-29200 Brest

Bernt Weber
bernt.weber@splashelec.com

Splashelec
18, Rue de Pont Louet
FR-29200 Brest

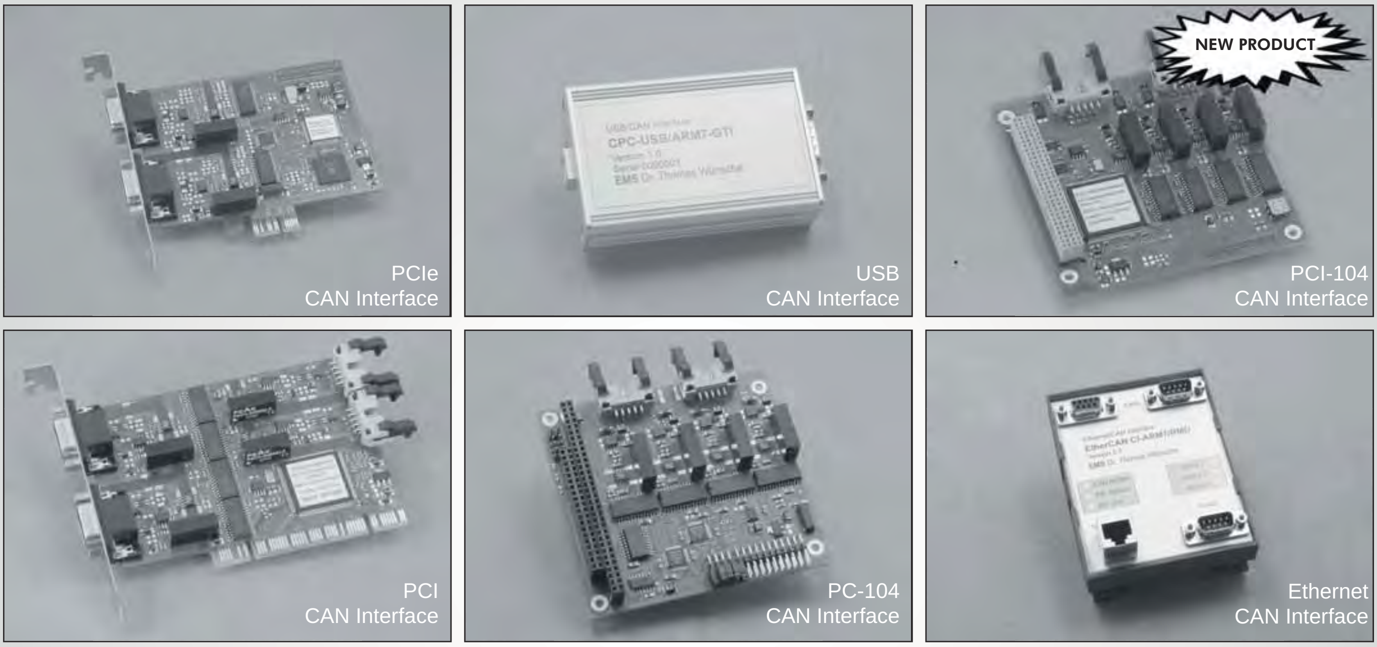
Links
www.ensta-bretagne.fr
www.splashelec.com

“We aim to give sailors with a disability access to mobile activities.

Robotics in the sailing field is now a reality as was demonstrated in 2008 at the World Robotic Sailing Championship and the associated conference [3]. Able to replace humans during competition or to realize autonomous measurements, its field of action has become larger than before. Our approach is not to remove human action, but to assist it during information acquisition, decision process and execution. Sailing globally has remained a field inaccessible for people with a disability, because of the extreme mobility the sailor needs to acquire information and to manipulate commands of a boat. Sailing requires significant efforts that not all disabled can put forth. We aim to give sailors with a disability access to those kinds of activities. The assistance system was initially composed of an electronic board and a joystick, which allow a person to steer a boat manually as helmsman, forming part of a crew. When needing free hands, the helmsman can activate compass guided steering.

An Android based HMI was developed to complete the system with a visual navigational aid for disabled people [2]. To provide a more complete view of the environment for the skipper and to counterbalance the lack of mobility, sensors were added to the system. Linked to an Arduino board, information was sent to a tablet in charge of the information display via Bluetooth. Interfacing and cabling became complex and interfaces to connect extensions became a rare resource. Users then asked for joystick-steering on smaller single-handed boats, which requires more actuators and sensors. It was time to think about a different system architecture, with fewer connections and better reliability.

The present part of the project therefore aims to develop a CAN interface board and the necessary software to allow communication using a more modular and flexible network system. This system needed to be easily adaptable to all sorts of situations and disabilities: developers should be able to plug in their own sensors and show data directly on the tablet. On the software side, an Arduino bootloader compatible with CAN allows the programming of nodes directly through the network. The final objective ▷


NEW PRODUCT
PCIe
CAN Interface
USB
CAN Interface
PCI-104
CAN Interface
PCI
CAN Interface
PC-104
CAN Interface
Ethernet
CAN Interface

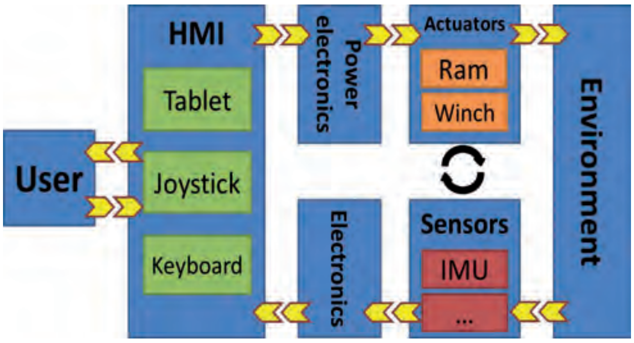


Figure 1: The system is based on a closed loop, where the user can be removed for autonomous behavior

is to provide sailors with a navigational assistance system able to perform automated tasks like heeling limitation, autopilot, etc. The system has to integrate the needed adaptability into the development process and to offer a solution for various kinds of disabilities. For example steering with a joystick compensates the lack of strength in an arm and the HMI centralizes sensor information to compensate a lack of mobility and sensitivity to wind and speed. Furthermore, due to its open-source nature, the software system is modifiable and developers can add functionality or change the HMI according to specific needs. Finally, the system should be adaptable to every kind of boat.

Displays and autopilots installed on today's sailing yachts mostly use multi-drop serial networks with vendor specific proprietary protocols. On some recent boats, the CAN-based NMEA2000 [1] network has replaced the proprietary protocols. For higher bandwidth requirements such as radar or echo sounder images, these networks are sometimes completed by Ethernet cabling (Furuno Navnet, Raymarine Seatalk-HS). Note that the Ethernet approach is power-hungry, costly and difficult to adapt to simple 8-bit microcontrollers, but NMEA2000 offers a good solution. The problem is that NMEA2000, and the J1939 protocol it is based on, are proprietary protocols. CAN looks promising, but the missing piece is an adapted open source protocol that would allow the design and integration of new hardware for different handicaps.

The number of connections in existing sailing assistance systems has reached a limit, where the system becomes impractical and expensive in an environment where every connector has to be waterproof and saltwater-resistant. Joystick box connections need up to 9 wires; adding sensors to individual actuators would complicate the cabling even more. Since bus systems are already employed in user-interfaces, we think that they should be extended to the entire system, including actuators and associated sensors. This would simplify installation, allow more advanced interactions, better user experiences, and make the systems more modular, flexible, and adaptable.

Figure 2: PSC of the Splashelec system in a waterproof case

System description

The system is composed of two main parts, the Programmable Servo Controller (PSC) and the HMI, which displays sensor information and allows the skipper to activate the joystick or to use the autopilot. The communication between the bus and the HMI tablet happens via Bluetooth, whereas internal communication uses the CAN protocol. As shown in Figure 1, it can be modified to work as a closed-loop system: information that comes from the environment, just like any human input, influences the actuators, which means the system will be able to perform automated tasks.

The programmable servo controller PSC (Figure 2) is a key component and can do the work of an autopilot course computer, which can steer to wind or compass when associated to the corresponding sensors. The PSC board (6 inch x 7,5 inch) contains a microcontroller, power electronics for an electric motor, an electric clutch and the power supply including filter circuits (Figure 3). Various interfaces for rudder angle sensor, joystick and control keyboard, and electric winches will be integrated directly into the main circuit board via the CAN interface. This PSC is based on open-source technology in order to allow the easy integration of new functions or to modify current ones. The microcontroller is compatible to Arduino boards and can be programmed with Arduino's IDE software, which gives access to the programming interface, existing libraries and various on-line examples. Wired to this box, multiple sensors such as an Inertial Measurement Unit (IMU) and a wind sensor or loch-speedo are linked into a CAN architecture. When using more than one actuator, each one uses its own dedicated PSC, with power electronics, local sensors, and a CAN interface. The whole system, including mechanical parts, is transportable and can be adapted to different boats.

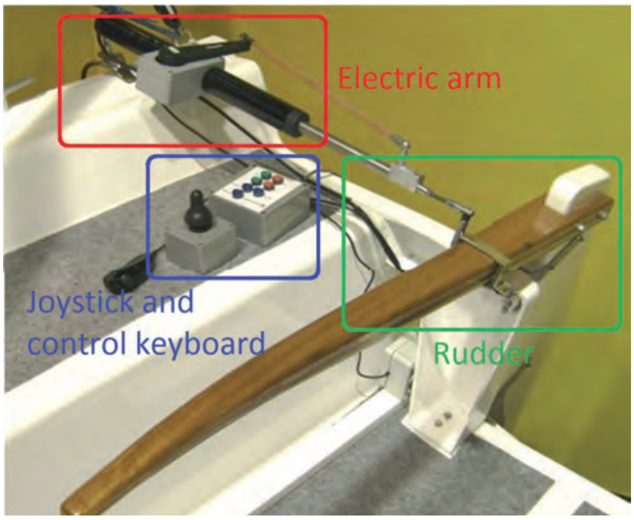


Figure 3: System composed of an electric ram (in red), a joystick and a control keyboard (in blue) and the rudder (in green)

The HMI, programmed for Android by using the Android Software Development Kit tool for Eclipse, contains different areas (Figure 4): in addition to textual information, some data, such as wind orientation and speed, compass and rudder angle, is also displayed in graphic form to allow a better understanding of the environment. A last area ▷

acts as a virtual keyboard to switch between autopilot and joystick mode and the user can modify the course. The graphical layout of the HMI is realized in XML language and could easily be modified to fit the needs of specific users. This results in a system (joystick, tablet display, extra keyboard, etc.), compatible with different disabilities, while still being usable by sailors without any disabilities.

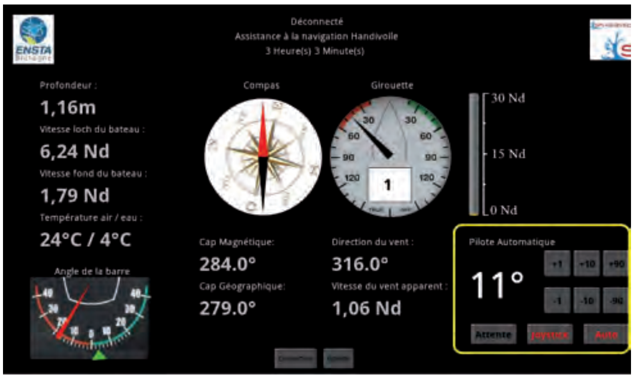


Figure 4: The human machine interface displays sensor information and contains a virtual keyboard (in yellow) to replace the physical one

CAN protocol

The choice of the CAN protocol for our architecture arose from the need of a broadcast communication mechanism, that had to be easy to use, able to work with multiple nodes, and which provides a reliable communication protocol to exchange information from sensors and actuators. Indeed CAN is able to detect errors with no less than three mechanisms: Cyclic Redundancy Check (CRC) to verify message integrity, Frame Check to verify that data is sent in the correct shape, and acknowledgments to guarantee reception. Besides, adding nodes to an existing CAN network can be done easily, which meets our needs for a modular architecture.

Adding nodes to the network is straightforward: adding a sensor with interfaces like for example NMEA183 (National Marine Electronics Association) or I2C (Inter Integrated Circuit) implies reading the data by the connecting nodes microcontroller and to put the data in a CAN frame. Arduino allows doing this in very few lines of code, using libraries for CAN and a plethora of choices for sensor interfaces.

Higher-layer protocol

At this point the need for a higher level CAN protocol to organize data in the CAN frames appears. Various higher-layer CAN protocols already exist, such as J1939 used in NMEA2000, CAN Kingdom, or CANopen, but it appears that non of these fits our need for a simple CAN protocol. Too complex or not adapted for 8-bit microcontrollers, CAN protocols are not widespread and their code is not always open-source. We decided to develop our own protocol, based on our needs, called SimpleCAN, into an Arduino library containing the essential functions to support our architecture. The protocol permits adding new features.

System architecture

Our architecture is based on Arduino compatibility and uses a CAN interface called the CANinterfacer, compatible to an Arduino with a CAN shield on top. This board [5] uses an ATMEGA32U4 microcontroller as do the Arduino Leonardo and Micro. It is small in dimension (slightly smaller than 5 cm x 5 cm) and can be programmed via USB using the Leonardo bootloader and the Arduino IDE. It can be powered through CAN by an on-board switching power supply accepting from 7 V to 32 V. Most of the I/O pins are available for local connections.

In the system, a group of elements (actuators, sensors, HMI elements, etc.), wired to a CANinterfacer, becomes a CAN node (Figure 5): each CANinterfacer uses Arduino libraries to convert input from various sources, for example analog inputs, NMEA183 or I^2C connected sensors, into CAN messages. Putting a CANinterfacer between the new hardware and the bus to integrate it as standard node increases the flexibility. In the same way, the PSC contains a CANinterfacer and functions as a native node in our bus system. This allows integrating servo-controlled actuators such as rams and winches with their associated sensors.

CAN bootloader

The CAN protocol implementation opens the way to simplified programming of every node through the bus with only one connection. To obtain that, an Arduino bootloader that accepts CAN programming commands for our CANinterfacer is needed. The Robotics Club of Aachen [4] has already worked on the subject with very similar hardware and built its own CAN bootloader that allows updating firmware and local code from CAN messages. Small modifications have been made in order to suit connections of our CANinterfacer [5]. The programming operation is initiated by a Python script that initiates the communication process between the PC connected programming node and the node which needs to be reconfigured. In practice, the programming node acts as an In-System Programming (ISP) interface: it receives the new program by USB or serial port and sends it encapsulated in CAN messages to reprogram the specified node. The whole process is detailed in Weber's The CANinterfacer [5]. ▷

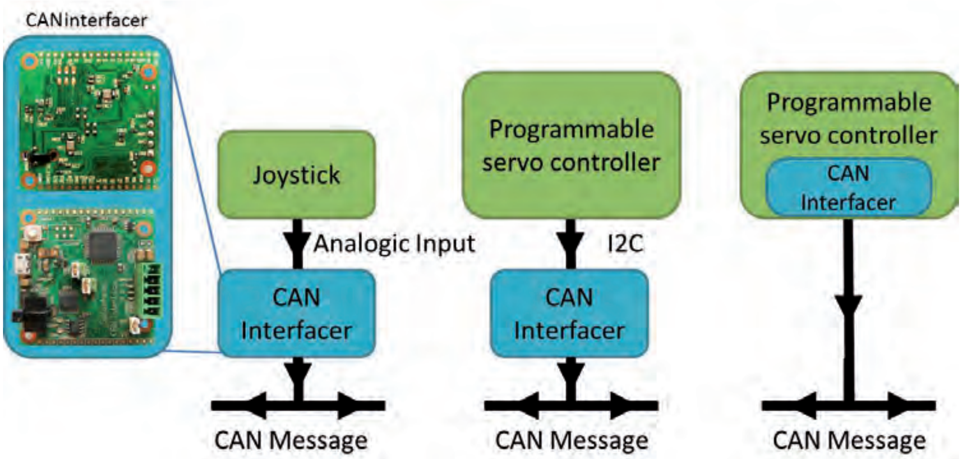


Figure 5: Examples of a CAN node: The joystick and the Programmable Servo Controller (PSC) in its first version; the next version will have a CANinterfacer integrated into the board

References

[1] National Marine Electronics Association. NMEA2000(TM) standard, In: National Marine Electronics Association website http://www.nmea.org/content/nmea standards/nmea 2000 ed3 00.asp, 2012.

[2] N. Brocheton, K. Bruget, A. Wibaux, O. Reynet, B. Clement, and B. Weber. Systeme d'assistance a la navigation handivoile. In Proceedings of Handicap 2012: 7th congress on technical assistances for disabled people, Paris, France, 2012.

[3] K. Bruget, B. Clement, O. Reynet, and B. Weber. CAN bus interface board for sailing applications. In Proceedings of the 5th International Robotic Sailing Conference (Springer Eds.), Brest, France, 2013.

[4] Fabian Greif. CAN Bootloader, In: Universal CAN library. Roboterclub Aachen e.V. http://www.kreatives-chaos.com/artikel/can-bootloader, 2010.

[5] B. Weber. The CANinterfacer, In: The Splashelec Wiki. http://wiki.splashelec.com/index.php/caninterfacer, Cited 13 May 2013.

Demonstrator

The WRSC 2013 demonstrator [3] proves the flexibility of such an architecture. The choice of boat fell on a Miniji from "Handivoile Brest", based on a small scale of a historic America's Cup hull. It is an inexpensive single-handed sailing boat, ordinarily steered by foot pedals or with a steering wheel. It offers vivid sensations to the sailor seated in a comfortable position in a bucket seat.

Furthermore, instructors working with disabled sailors demand an increase in autonomy and safety. To achieve this, a boat will be equipped with electric winches and a rudder system, all interconnected via CAN. Adding sensors to the bus, this boat will be able to perform automated tasks as a sailing robot. Indeed, to increase security, we can limit the heeling or restrict the navigational area. Instructors will also be able to take control of the boat for safety reasons or even activate the autopilot.

A system based on a CAN architecture allows developers to connect new sensors or actuators, and to reprogram the system using Arduino technology. Such a system assists the sailor during navigation and can automate complex tasks. It helps disabled people by easing the access to sailing activities and gives navigational assistance to anyone. Based on the open-source approach, electronics and software can be modified according to specific requirements. Numerous possibilities for development exist. The final aim is to obtain products with new features derived from robotic sailing, encouraging people to develop their own system modifications ◄

Electric boost for hydraulic transmission

A combination of hydrostatic and electric drives is used in mobile machinery. It allows downsizing the combustion engine, reduces consumption and exhaust emissions, and increases cost-effectiveness.

Authors
Ralf Schrempp
Sauer Bibus GmbH
Lise-Meitner-Ring 13
DE-89231 Neu-Ulm
Tel.: +49-731-1896-0
Fax: +49-731-1896-199
info@sauerbibus.de

Richard A. Zahoransky
Barbara Schwarz
Heinzmann GmbH & Co. KG
Am Haselbach 1
DE-79677 Schoenau
Tel.: +49-7673-8208-0
Fax: +49-7673-8208-188
info@heinzmann.de

Links
www.heinzmann.com
www.sauerbibus.de

An integrated electric drive from Heinzmann is used as an “e”-booster to support the ICVD (integrated continuously variable drive) hydraulic gearbox by Sauer Bibus (Figure 1). The hydraulic gearbox is equipped with what is known as wide-angle technology. The gearbox, which was developed in cooperation with GKN Walterscheid, facilitates continuous drive speeds of up to 40 km/h due to the 45-degree wide-angle adjustment ("45° swiveling angle") with maximum hydro-mechanical efficiency produced by the bent axis design. As a result, this gearbox is suited for self-propelled agricultural and construction machinery. In combination with a primary adjustment of the hydraulic pump, the hydrostatic conversion range is increased in such a way that this technology represents a convenient alternative to manual gearboxes and power-shift torque converters.

The strengths of the ICVD gearbox include improved energy efficiency, optimized driving characteristics for the machinery and increased operator convenience. With the ICVD, it is possible to deliver drive throughout the entire speed range without any interruption of the torque. In addition, the variable drive facilitates the simple reversal of travelling direction and torque as well as the automatic adaptation of the power requirement. The drive is easy to operate due to its precise controls and highly dynamic acceleration. This increases productivity and significantly reduces the driver's workload, meaning that the operators can concentrate on important work processes without being distracted by selecting the drive mode or changing gears.

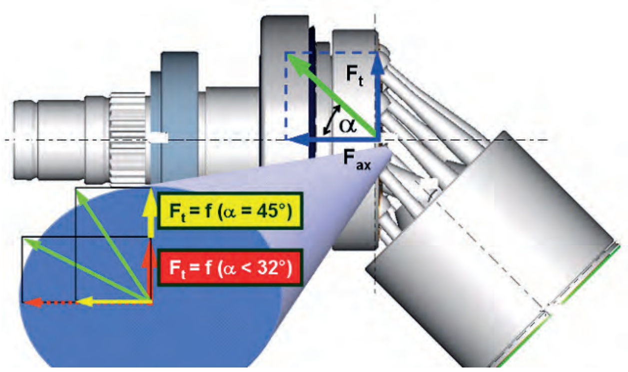


Figure 1: Schematic diagram of a wide angle unit

The additional integrated electric motor supports the hydraulic gearbox as a booster on the drive side in the hybrid version. The unit can be integrated in the CAN-based control network, so the drive can be activated as needed. This hybrid version created by integrating the electric motor into the hydraulic gearbox provides sought-after independence from the diesel engine. The diesel engine and the electric part of the hybrid can be installed in the vehicle as independent structures. The "decentralized" hybrid drive therefore offers a great amount of freedom with respect to its design, which makes it an option for virtually any mobile machine, as the expense of redesigning the existing, conventional mobile machines to form hybrid drive trains is kept within reasonable limits. Figure 2 shows the principal arrangement of the decentralized hybrid

The dynamic drive concept does not simply provide support during performance peaks; it also simultaneously recovers energy during downhill drives in its recuperation mode. This means that less overall performance is required from the combustion engine. This facilitates the "downsizing" or more accurately the "right-sizing" of engine performance because its operation is permanently kept in the optimum consumption window. This means that a small combustion engine can be used, resulting in lower fuel consumption and lower exhaust emissions. Additional cost savings may be realized by these investments with regard to the required after-treatment of exhaust gases. Thus, the total installation space needed for the entire hybrid drive train can be reduced.

With a corresponding power split between hydrostatic and electric drive ▷

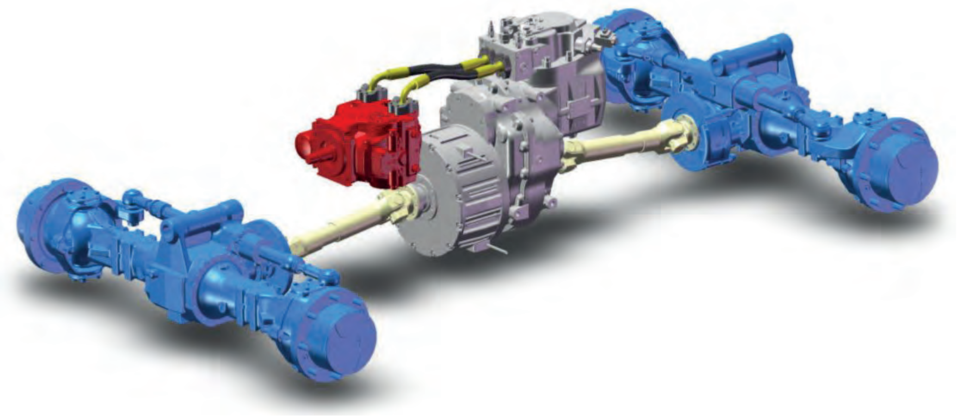


Figure 2: Example of the structural arrangement of the decentralized, hybrid drive train (ICVD hydraulic gearbox with integrated electric motor: "e"-boost) in the chassis of a mobile machine

energy, the driving comfort is exceeded compared to conventional systems. Simulations of the two traction drive concepts demonstrated particularly significant fuel and emissions advantages for a parallel hybrid in wheeled loaders and telescopic loaders, both with rapid, short load cycles, as well as in municipal vehicles [2, 3].

Electric motor requirements

The electric motor for the e-booster must meet the following requirements:

- Highest power and torque density,
- High efficiency for motor and generator operation,
- High torque with minimum torque ripple and minimum detent torque,
- Robustness and resistance to harsh environmental influences,
- Good controllability in coordination with combustion engine and gearbox,
- High dynamic performance.

The permanent magnet synchronous motor with rare-earth magnets proved to be the most suitable solution in a recent comparison of the motor types in question [1]. Heinzmann's hybrid e-machine fulfills all requirements: the single-tooth winding design meets the requirements of cost-effective, automated production and provides minimum axial dimension.

Literature

[1] S. Zhitkova, B. Riemer, D. Franck, K. Hameyer, R. Zahoransky: Hochdrehzahlmotoren für mobile Arbeitsmaschinen (High-speed engines for mobile machinery), 4th symposium on hybrid drives for mobile machinery, 20 February 2013, M. Geimer (ed.), Karlsruher Schriftenreihe Fahrzeugsystem-technik, KIT Scientific Publishing, 2013

[2] F. Böhler, P. Thiebes, M. Geimer, J. Santoire, R. Zahoransky: Hybrid Drive Systems for Industrial Applications. SAE-NA Paper Series, Paper 2009-24-0061, 2009

[3] Rebholz, W.: Neues hydrostatisch-mechanisch leistungsverzweigtes CVT-Getriebe für den Einsatz in Baumaschinen (New hydrostatic/ mechanical power-split CVT for use in construction machinery). Symposium on gearboxes in vehicles. Friedrichshafen 2010 (conference proceedings)

The "e"-boost ICVD hydraulic gearbox is a parallel hybrid. The electric motor and combustion engine have to be operated in a coordinated manner. The hybrid applications in mobile machinery require an electrical voltage between 200 V and 800 V, but can also be designed for 80 V or 96 V if required. Li-ion batteries are recommended, particularly to allow high power recuperation phases at high efficiencies. The degree of electrification can be varied to enable the electric operation of secondary and auxiliary drives and external electric power units.

The hybrid strategy must ideally be adapted to the typical load cycles, taking the battery capacity into account. For the shown individual load case shown in Figure 5, i.e. for a required torque at a specific speed, the combustion engine is ideally driven at its optimum operating point in terms of fuel consumption at the appropriate speed. The torque generated by the combustion engine in this operation mode will either be above or below the required target torque, as demonstrated in Figure 4 in the torque-speed diagram of a diesel engine. The difference between the torque values is covered by the electric motor: if more torque is required, the electric motor operates in motorized, i.e. in boost mode, whereby the required torque is supplied in total by the two motors. The electric motor power is drawn from the battery. If less torque is needed, the electric motor switches to generator mode and charges the battery. The optimum battery size can be found through simulations for the relevant load cycles. Naturally, the control system outlined above still has to respect constraints such as battery state of charge and the temperatures of the electric motor and battery. In addition to fuel consumption, this control strategy also minimizes pollutant emissions and is therefore the preferred option.

Other control strategies are also possible, such as phlegmatization of the combustion engine, to reduce noise emissions due to high speeds or high transients of speed and power. This type of strategy is of advantage for the operation of mobile machinery in densely populated urban districts. This operation mode is feasible because the electric motor delivers high torque levels even at low speeds. Ideally, the combustion engine operates at a constant output, as outlined in Figure 4. As previously described, the electric motor covers the changes in output power. The braking energy, which is indicated by a negative output in Figure 4, can be recovered in generator mode, irrespective of the control strategy. The diagram also shows the enhanced dynamic performance of the hybrid motor compared to the combustion engine. Power increases are implemented faster by the electric motor so that the hybrid drive exhibits a significantly higher dynamic output.

Purpose of CAN

Engine manufacturers generally integrate the hybrid control system in their own engine control unit, i.e. the sketched separate units are normally united in one sin-

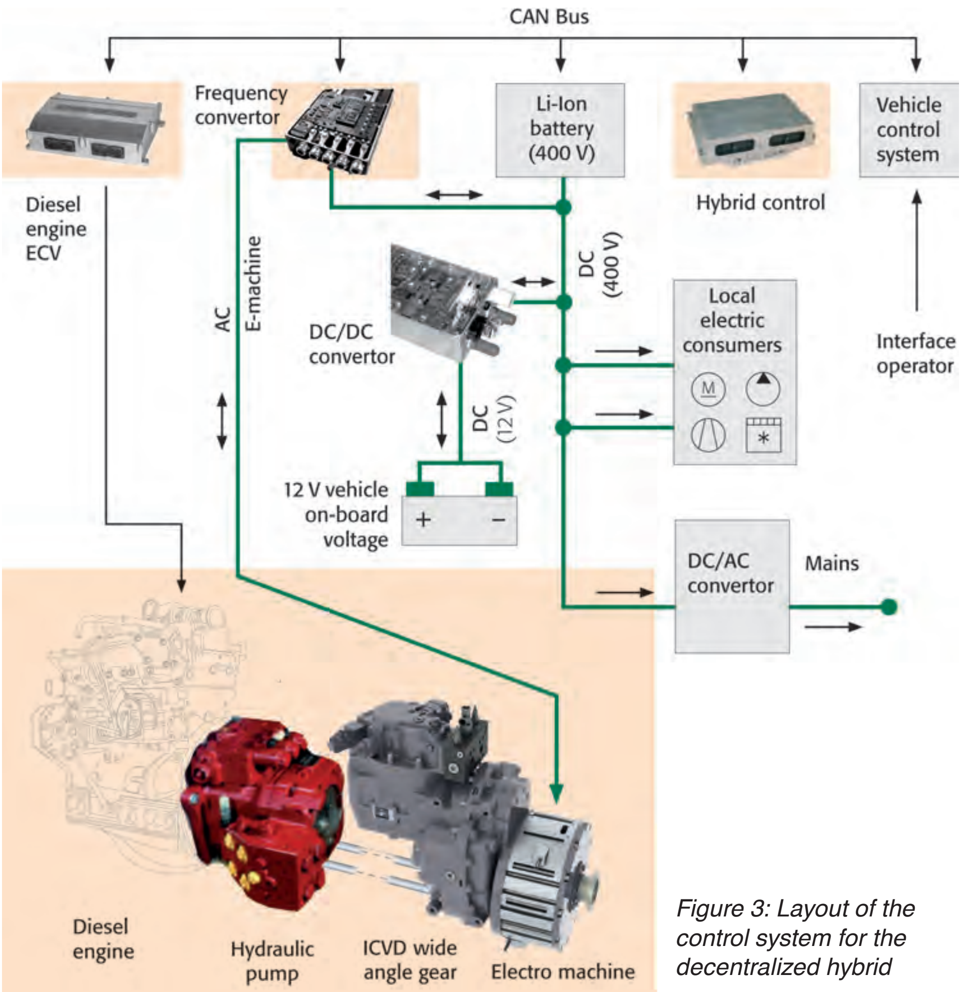


Figure 3: Layout of the control system for the decentralized hybrid

gle control device for diesel engine and electro motor. The presented hybrid system may additionally be a retrofit solution for manufacturers of mobile working machines who want to downsize diesel engines with the aim to reduce emissions at the same power output but at a higher dynamic, particularly at low speeds. The latter case requires the hybrid control system to establish not only the connection to the inverter drive control and to the battery control, but also the vehicle, gear and diesel engine control – considering the fact that these components are generally not optimized for hybrid characteristics. Figure 3 presents the selected system layout for the hybrid control. Thus, the demanded improvements are determined by the functionality, complexity and flexibility of the add-on hybrid control if no modifications can be done at the other individual controls within the complete vehicle.

The CAN network is required to receive all necessary data out of the multitude of individual electronic control units. Such CAN interfaces are integrated in all modern electronic control units (ECU). ECU manufacturers use different CAN-based higher-layer protocols (HLP) in contrast to the hardware, which is universally defined by a two-wire connection. There is the SAE J1939 approach, which is supported by most controls designed for vehicle, gear and diesel engines. But inverter and battery manufacturers may not use this HLP. Therefore, the hybrid ECU must be designed to allow adjustments without excessive effort to different HLPs to fulfill the requirements of the specific applications of the wide range of mobile working machines. The hybrid control must comprehend several physically separated CAN channels, as not all of the various protocols are compatible. All necessary actual operational and status data of vehicle, gear, diesel engine, battery and inverter of the e-machine are read out from the individual control boxes and used as input for the hybrid control. This concerns start or stop demands of the vehicle control, shifting information of the gear control, speed of revolution, torque and cooling temperature of the diesel engine, state of charge of the battery and finally speed of revolution, torque and temperature of the electro machine. All this information determines the actually required control strategy and imposes the operational value for the electro machine. If the electro machine takes over part of the power load (working in its drive mode), the diesel engine control reacts automatically by reducing the fuel injection mass. If the electro machine operates in its generator mode (e.g. during braking periods), the ▷

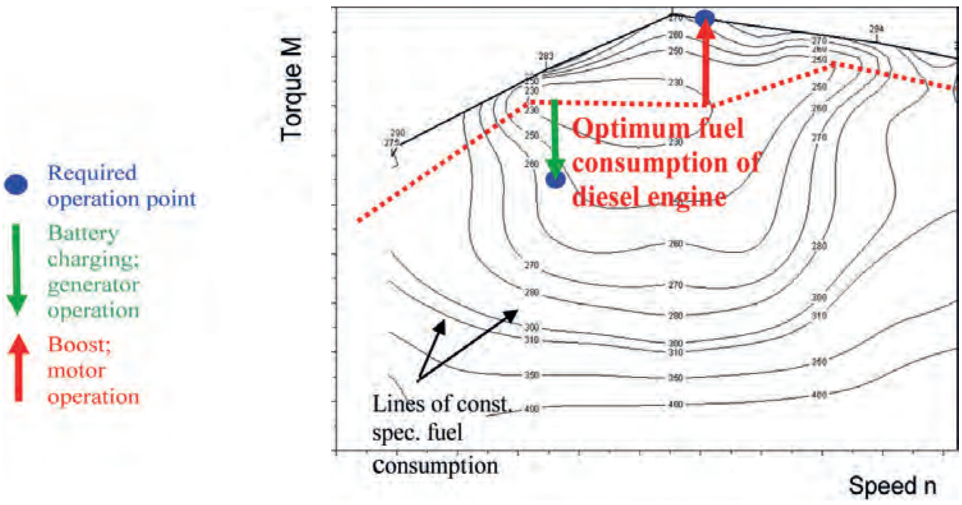


Figure 4: Hybrid control strategy of the hydraulic gearbox for minimum fuel consumption and minimum pollutant emissions

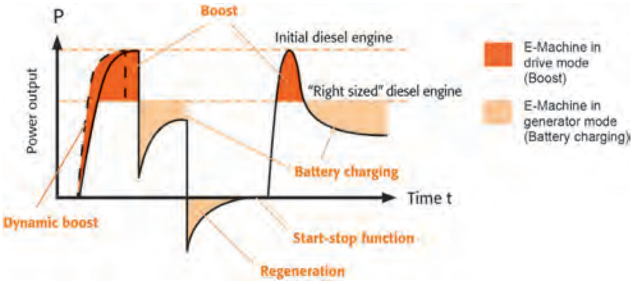


Figure 5: Idealized load cycle

excess power is used for charging the battery. However, all this is limited by the boundary conditions like allowable temperatures, allowable torques, state of charge of the battery etc. This data flood can only be mastered by CAN networks.

First experiences

Until now, the manufacturers of combustion engines have provided the diesel/electric hybrid systems. With the "e"-boost gearbox described here, hybrid drive trains can be installed irrespective of the brand of the combustion engine. The "e"-boost ICVD hydraulic gearbox combines the advantages of wide-angle technology with those of a hybrid drive. The combination represents a parallel hybrid system.

The following advantages are derived from studies of the various types of mobile machinery (wheeled loaders, multifunctional excavators, mining machinery, forklift trucks, municipal multi-purpose vehicles):

- Reduction in fuel consumption: A fuel consumption reduction of approx. 20 % was determined for typical load cycles of mobile machinery. The developed simulation model was verified in experimental terms in real driving cycles [2]. Further potential from the electrification of operating, secondary and auxiliary power units was not considered.
- Associated reduction in emissions: For the pollutants CO2, CO, NOx, CmHn.
- Increase in productivity: The enhanced dynamic performance of the drive train at low speeds significantly increases the productivity of mobile machinery. In one application case, an astonishing 40 % increase in productivity was reported – these increases in productivity depend strongly on the load cycles of the applications. A more in-depth analysis is still to be completed.

Converting mobile machinery to the "e"-boost ICVD hydraulic gearbox in the manufacturing phase of the mobile machine is generally possible without the need of major design changes to the vehicle.

This is due to the fact that a smaller combustion engine with lower output can be selected, providing more space for the electric motor. In conjunction with this, the smaller combustion engine is selected so that it falls in a different emissions class, reducing the investment and installation expense for the exhaust gas after-treatment technology. The hydraulic gearbox is suitable for any mobile machinery, but especially for applications where value is placed on optimizing the driving characteristics of the machinery and increasing operator convenience. This allows the driver to concentrate on complex work processes without being distracted by having to select the drive mode or changing gear. ◄

Systematic re-use of information

Eliel Saarinen, a Finnish architect, said: "Always design a thing by considering it in its next larger context – a chair in a room, a room in a house, a house in an environment, an environment in a city plan." That also applies to system design.

Author

Dr. Heikki Saha
TK Engineering
P.O. Box 810
FI-65101 Vaasa
Tel.: +358-6-35763-00
Fax: +358-6-35763-20
info@tke.fi

Links
www.tke.fi

References

Design processes defined by the selected system integration framework, such as CANopen, should be understood as an integral part of companies' design, assembly and service operations, instead of "just another communication protocol" in individual control systems. There are many use cases for such kinds of system integration framework in system design, assembly and service activities.

The core of the design process is a set of standardized file formats for storing design information in different phases of the projects, enabling various levels of re-use [13] [14] [15]. There is also a comprehensively defined application programming interface (API) for accessing CANopen communication services from IEC 61131-3 applications to enable the use of device-independent applications [4]. The communication database (DBC) is not standardized, but it is the de-facto format for describing network communication, mainly for network analyzers to enable the visualization of communication in a format that can be understood by humans.

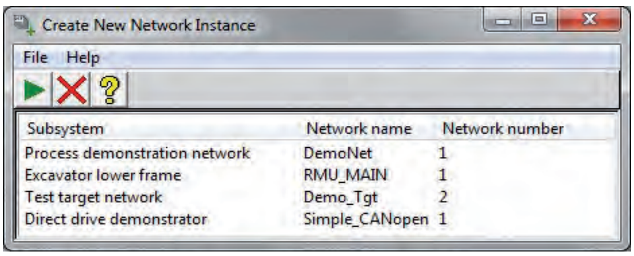


Figure 1: An example user interface for selecting the target subsystem instance from a list updated in conceptual design phase

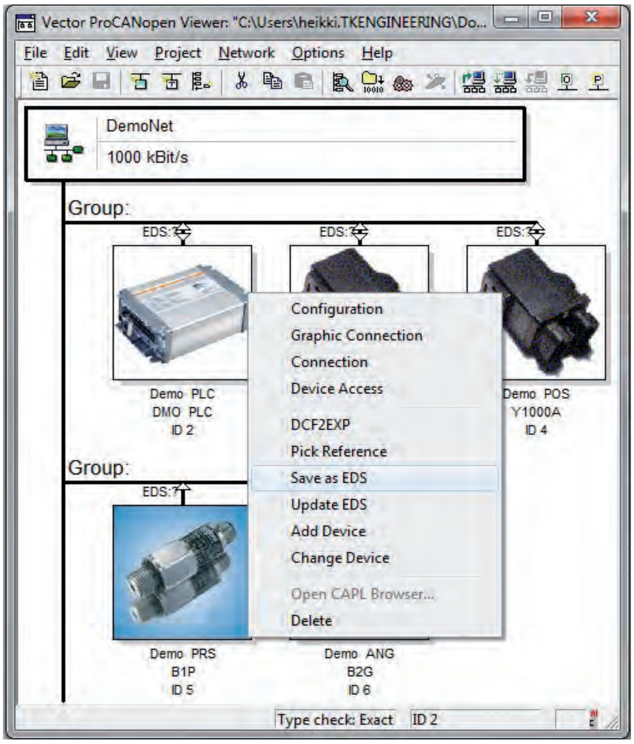


Figure 2: Example system tool with "Save as EDS" command highlighted

Of course, it is recommended to design CANopen systems by following the standardized design process [13]. But systems can also be configured device by device, stored in individual device settings as DCF files, and used during assembly and service. After designing the first system, the question arises how to efficiently use the information during assembly and service. Another question is how to re-use this information in the following design cycles and system projects.
The main focus of this article is on the re-use of design information within one single CANopen project or between CANopen projects. It is also considered how design information of other disciplines may be systematically used in CANopen projects and vice versa. This approach also emphasizes the possibilities of linked reference data, programmable system data management, and validation. This is in line with the mainstream SW development principles. Systematic re-use means not only the reduction of design efforts but also the reduction of failure costs.

Exporting from CANopen projects

Re-using information from CANopen projects is straightforward, thanks to the thoroughly standardized file formats. Various use-cases are ▷

published and generating communication abstraction layers for applications is the most widely used export [4]. The generic approach utilizes all information existing in CANopen projects, including signal scaling and units [7]. Further improvements, such as management of parameter access paths [4] and error codes [6], have also been investigated and proposed as future versions of corresponding standards. In addition to the generic approach, there are also vendor specific frameworks on the market, mainly supporting application development only and excluding device configuration management.

Though configuration downloads from DCF files to CANopen devices have been included in the scope of CANopen [13], generic CANopen tools do not commonly support a completely automated approach with layer setting services (LSS) and program download support. There are few details that cannot be solved by using DCF files only. Templates with most of the required information can be generated directly from the CANopen projects, but in some cases manual work might be required. Systematic approaches integrated into CANopen system design processes and tools have already been published [5]. It can be concluded that the better the latest CANopen features are supported, the less manual work is required.

It has also been investigated how structural information of systems can be composed from multi-disciplinary projects [3]. The use of additional, system level mechanisms, such as reference designators, is required to enable the recognition of common components between disciplines. Thanks to its standardized project structure, CANopen was the easiest discipline, but e.g. for electric design, standardized interfaces do not exist. Thus, specific export plugins for each CAD program need to be developed [9].

Re-use of entire subsystems

It is efficient to re-use entire subsystems in larger systems. In practice, only one version needs to be designed by always using the network-ID and network name of the first version. Following versions can be generated from the first one, by modifying the network name and network-ID. Several members of a product line may share the subsystems without modification, when network name and network-ID remain the same. The systematic process expects the definition of each version in the conceptual design phase. An example tool shown in Figure 1 lists the possible target subsystems, defined in a central repository.

The procedure of re-use is simple. After selecting a source project, a target project a and target folder, each DCF file of the source project has to be copied to the target folder first. Next, the network-ID and network name must be updated into each target DCF file. Finally, the nodelist.cpj project file has to be created in the target folder. If alternative devices are supported, all DCF files in the project folder must be copied, not only files listed in nodelist.cpj [10]. With an example tool the whole copying process takes only a few seconds. If an older project is used as a source, after the copying process some devices may need to be changed into new models, for which a systematic approach already exists [10].

Re-use of device parameters

Quite often there is a need to re-use configurations between devices. If a device becomes obsolete or needs to be changed for a cheaper or better one, the rest of the project typically remains unchanged. It means that the part of the system integration interface that is in use, needs to be supported by potential alternative devices [10]. Various compatibility levels require different approaches.

In the best-case scenario, the new device has an identical system integration interface. A good example can be found in single-coil valve drivers, where the only difference is in the coil connector. In this case, communication related parameters can automatically be copied from the original device [10]. After setting the correct communication, the device settings must be set. Because of the one-to-one compatibility, settings can be imported by a device editor directly from the DCF file of the original device into the DCF file of the new device [8]. Human mistakes can be avoided when settings are imported with a tool instead of manual copy and paste.

Devices following device profiles are so similar that in most cases the most essential parameters can be imported. The first obstacle may arise when special features of the original device are in use. A second obstacle may arise when rarely supported optional parameters are used. For example, most vendors and models support the most common interface of pressure transmitters. As long as default pressure data type is used and no other parameters than measurement unit and number of decimal digits, one transmitter can be replaced by another independent of their configuration.

The most difficult case occurs when the old and new devices share only the signals. Then the alternative devices need to be configured to support equal signal scaling and units. Parameters are often at least partially different and for compatibility reasons, it is not a good idea to access device specific parameters in the target system, because it efficiently decreases the drop-in replaceability [2]. Table 1 provides an example for this approach, consisting of three different I/O devices with fully compatible CANopen communication and M12 connector interfaces.

Table 1: Comparison of three drop-in replaceable general purpose I/O-devices with completely different off-line adjustable parameters

Objects	Device A	Device B	Device C
Digital inputs	6000_h		
Digital outputs	6200_h		
Analog inputs	6401_h		
Off-line adjustable parameters	6431_h 6432_h	2000_h	5240_h

Re-use of devices

Changes to the system integration interfaces during system design may be required for many reasons. Generally, transformation from DCF back to EDS mostly applies to application programmable devices, whose EDS files contain application specific signal and parameter objects. So-called Dynamic Channels may be used for signals, which means that network variables are defined during the system design phase [13] [15]. Alternatively there might be missing or erroneously defined signal or parameter objects in the profile databases created from specifications and in the EDS composed of the profile databases. Due to the constraints of the system ▷

design tools, it might be faster to make small changes directly to the DCF files in the system project and synchronize the changes into the original EDS file. Synchronization is required to keep the project structure consistent and to maintain re-usability. The translation from DCF to EDS is simple. The attribute LastEDS from the FileInfo section and the whole DeviceComissioning section must be removed [13]. The attributes ParameterValue, Denotation [13], and ClientX [4] of each object must also be removed. Additionally, the attribute FileName in the FileInfo section must be updated [13].

The translation from DCF to EDS can be performed manually by opening a DCF file with an EDS file editor and saving it as an EDS file. A more systematic approach for translation has also been evaluated. An automatic translation from DCF to EDS can be automated by using the CANopen tool's integration mechanism [15] and the existing information of the EDS file, from which the DCF file has been generated. A major risk in the fully automated translation is an unintended change of the original EDS file if it is named automatically for each DCF.

A safer approach is to use the CANopen tool integration mechanism but let a designer trigger the translation manually from the system design tool, as shown in Figure 2. The safest but also most inefficient approach is to only enable transformation into a new EDS file. CANopen tool calls are included in the EDS and DCF files, which enables limiting the availability of the translation only to the devices requiring such a feature. Based on the measurements, translation speed-up is at least in factor of 100 – even with a small DCF, manual translation may take at least 15 minutes while a tool makes it in less than 1 second.

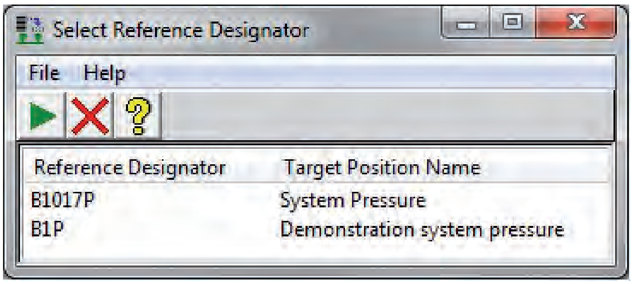


Figure 3: Example of a tool for picking reference designators and position names from central repository into a CANopen project

Re-use of applications

CANopen profile databases (CPD) enable the description of parts of the devices object dictionary. Profile databases may be used as a standardized mechanism for the management of device interfaces in a modular way [14]. Signal and parameter definitions may be managed manually, in requirement management systems, or in various kinds of models. A commitment to the standardized transfer format enables starting with a document-based approach and moving further into model based design, without a need for modifications in the detailed design.

In case of changes made during later phases of the project, it is important to avoid additional workloads by automatically synchronizing changes backwards. When the changes in the system integration interfaces are synchronized back into profile databases, interface descriptions may be re-used in the application or function level. If a model-based design is used, changes may be synchronized from the profile databases back to the model to keep it up-to-date.

The translation from an EDS file into profile databases is technically straightforward: It is simply a conversion from one description format of selected parameters into another format. However, there are some open issues on how to handle the original modular structure. First, a standardized mechanism for indicating from which profile databases the EDS file has been constructed and how the objects have originally been divided into modules does not exist [13] [14]. Parameter objects are easily divided into dedicated groups, but signals are located into sub-objects so that the main object depends on the data type [12]. Therefore, other criteria than the object index must be used. One option may be an attribute PP-Offset [15], which defines the absolute offset of the signal object in the process image. Another option could be the use of additional entries in the EDS and DCF files, but there is a compliance risk because such objects are not standardized.

The use of object dictionary areas in compliance with CANopen is well defined. Signals to and from applications are updated through so-called network variables [12]. There are signal data type and direction specific objects defined, providing access to the same memory area. The value of the attribute PP-Offset defines how the data type specific network variable areas are organized in the process image [15]. The first problem is that multiple types of network variables may point to the same area of the process image. The second problem is that PP-Offset attributes may not be available because network variables are assigned into objects already in the EDS or profile database.

While fully automatic DCF to EDS translation introduces risks, automatic EDS to CPD conversion introduces comparable risks. Therefore only manually triggered translations with object index based filtering have been implemented. They provide significant speed-up and do not lead to complex and proprietary design rules. Based on these measurements, computer-aided translation according to an example in Figure 4 increases the efficiency by a factor of 60 or more, depending on the size of the EDS file and the number of resulting profile databases.

Import to CANopen projects

Parameter and signal descriptions for application-programmable devices can be transferred from specifications into designs as profile databases [1]. Profile databases are an excellent entity for such purposes, because each one ▷

```
C:\>python eds2cpd.py demo _ plc.eds plc _ params.cpd 2000 5FFF
Translating objects 0x2000..0x5FFF
  From: demo _ plc.eds
  To: plc _ params.cpd
C:\>
```

Figure 4: Example of extracting manufacturer specific objects into a separate profile database

defines a subset of an object dictionary and thus does not have any dependencies on the target HW- or SW-platform [14]. The profile database format is not the most convenient one to be edited manually. Profile databases can be viewed but not edited with spreadsheets, because comment lines may reserve whole lines and will not contain separator characters. Furthermore, standardized file headers have to be maintained manually.

Reference designators are commonly used in the industry for identifying system components [9]. Each logical position has a unique identifier, a reference designator. Because they are technologically independent and exist in design documents of each relevant discipline, they can be used for combining discipline-specific design documents into a common, multi-disciplinary design entity [3] [9]. In addition to creating design documents, reference designators can be used to validate the consistency of the documents. Reference designators are currently only used for certain subsets of components and disciplines, but the actively used approaches may easily be improved. Furthermore, reference designators are manually maintained and written into each design document. To avoid inconsistencies caused by human mistakes and to increase efficiency of design work, reference designators should be stored into a central repository, where they can be picked for the designs, instead of being written manually [9]. Many design tools require the use of their internal mechanisms, but due to the standardized project files, a system design tool independent pick-up tool, such as presented in Figure 3, can be used with CANopen projects.

Compiled application software can be linked into DCF [13]. Download objects are of the data type domain. The name of the file to be downloaded may be given as a DownloadFile attribute. When the compiled application is linked to the configuration, download tools can first download the application and then the corresponding parameter values as a single operation from the user's point of view.

Summary

Standardized storage formats of design information enable almost unlimited possibilities for the programmable management and re-use of information from within and to CANopen projects. These notions are in line with the approach to Industrial Internet, where machine-understandable and enriched design data is the key to successful and modern services. In addition, EDS and DCF files enable seamless integration of the supporting tools into the design process. The rich information contents of the standard files serve well as a multi-disciplinary design approach by supporting architecture level information, in addition to the communication description. Current design tools do not provide all features, but they support open, standardized interfaces for filling the gaps.

Systematic design information management significantly improves the efficiency of each phase in the process, moving focus from the management of the raw files to the management of the system level design issues. Further improvement can be achieved by re-using various kinds of design entities, which are stored by using standardized information contents. Information exchange based on standardized contents enables easy crossing of geographical and organizational borders. Instead of the improvements achieved in the design process, the most significant improvement comes from reduced failure costs caused by inconsistent or outdated information, especially in assembly and service. ◄

CAN FD from an OEM’s point of view

CAN FD provides bit-rates higher than 1 Mbit/s and payloads larger than 8 byte. Nevertheless, it is a proven technology: robust and reliable. With these characteristics CAN FD meets the requirements of carmakers.

Authors
Dr.-Ing. Marc Schreiner
Daimler AG – Research and Development
Wilhelm-Runge-Straße 11
DE-89081 Ulm

Haytham Elwy Mahmoud
Martin Huber
Daimler AG – Research and Development
Bela-Barenyi-Str. 1
DE-71063 Sindelfingen

Serhan Koç
Dr.-Ing. Jan Waldmann
Daimler AG – Truck Development
Mercedesstr. 123
DE-70372 Stuttgart

Tel.: +49-711-170
Fax: +49-711-1722244
dialog@daimler.com

Link

Keynote speech of Dr. Schreiner at the 14th iCC on Youtube

References
[1] CAN with Flexible Data-Rate - Florian Hartwich, CAN in Automation, iCC 2012
[2] Safeguarding CAN FD for applications in trucks - M. Schreiner, H. Leier, M. Zerzawy, T. Dunke and J. Dorner, CAN Newsletter March 2013

Because vehicle networking is not an infrastructure that is directly perceptible by the customer, it is a balancing act between economical pressure and technical innovation that requires the adequate use of specific technologies. Along the lines of “as few as possible, as much as necessary” Figure 2 shows a simplified schematic of the E/E-architecture of the current S-Class (launched in 2013) and Figure 3 the current Actros truck (launched in 2011).

Both architectures follow the idea of grouping communicating systems of the same domain in their own network systems to reduce overall busload. Especially in truck systems a large amount of inter-domain communication requirement (e.g. brake and powertrain system) remains, which cannot be satisfied by introducing just another CAN network system. On the other hand the introduction of non-CAN communication would require a big change of today’s software and communication implemented within truck systems. The introduction of CAN FD [1] for these busload-critical CAN-networks seems to be a perfect solution to achieve higher network capacity without large changes in the existing systems like brake systems or engine controllers.

The next evolutional step, automotive Ethernet, will have an impact on CAN. On the one hand there will be a need for more bandwidth in architectures like Figure 4. Such an architecture and especially Ethernet itself requires sub-networks with payloads of more than 8 byte.

CAN FD physical layer

The average bit-rate depends on payload length and identifier length (11-bit or 29-bit). The correlations are given in Figure 5, where the average bit-rate can directly be compared with today’s CAN bit-rates, e.g. 500 kbit/s. Please note that this estimation does not include stuff bits. The upper graphs in Figure 5 are plotted for arbitration speeds between 500 kbit/s and 800 kbit/s. The x-axis represents the bit-rate in the fast data phase of a CAN FD frame, while on the y-axis the resulting average bit-rate is plotted, assuming that only 8 byte payload frames are used. Figure 5a shows that the average bit-rate could be nearly doubled by an arbitration speed of 500 kbit/s and 2 Mbit/s for the data phase using only 8-byte data frames and 29-bit IDs. There is more gain in average bandwidth using only 11-bit IDs.

The lower graphs in Figure 5 show the effect of the extended payload length, assuming that all transmitted frames make complete use of the respective payload. It is evident that the gain in average ▷

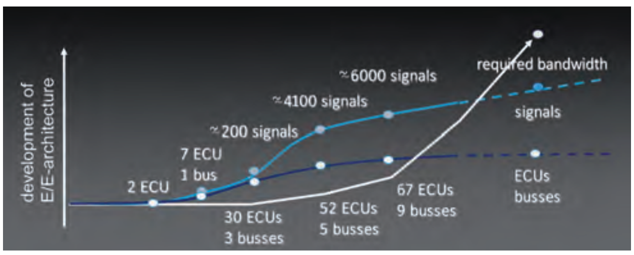


Figure 1: Evolution of in-vehicle networking: different evolutional steps in terms of in-vehicle networks from its early days to the present using the example of the E-Class.

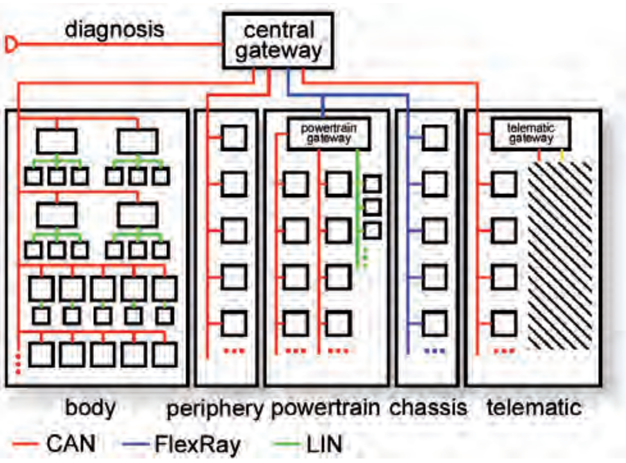


Figure 2: Current passenger car architecture

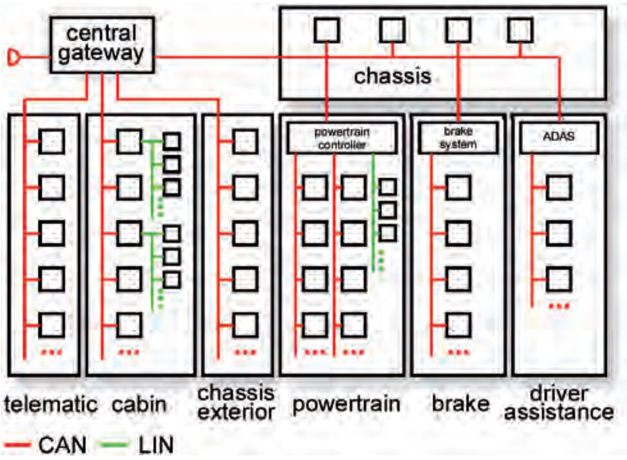


Figure 3: Current truck architecture

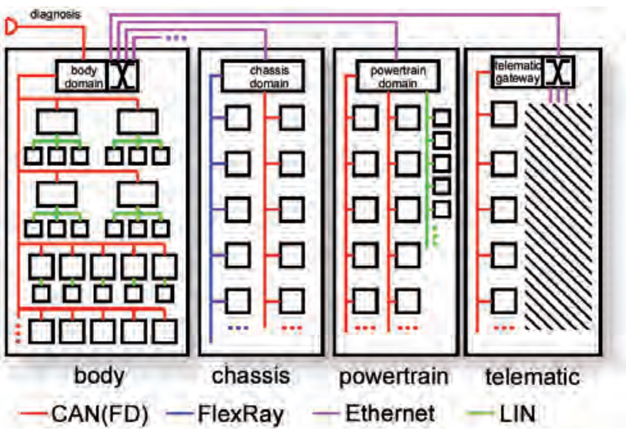


Figure 4: Future passenger car architecture

bit-rate is maximized when frames with long payloads are used: e.g. in a network with an arbitration speed of 500 kbit/s and a speed of 2 Mbit/s in the data phase using 8 bytes payload would give just a little less than a 1 Mbit/s average bit-rate. However using only 64 byte of payload yields a little more than 1,5 Mbit/s average data-rate as shown in Figure 5d. This means an increase of approximately 50 % in an average data-rate.

Designing CAN FD networks

The most important key parameter for evaluating CAN networks is the propaga- ▷

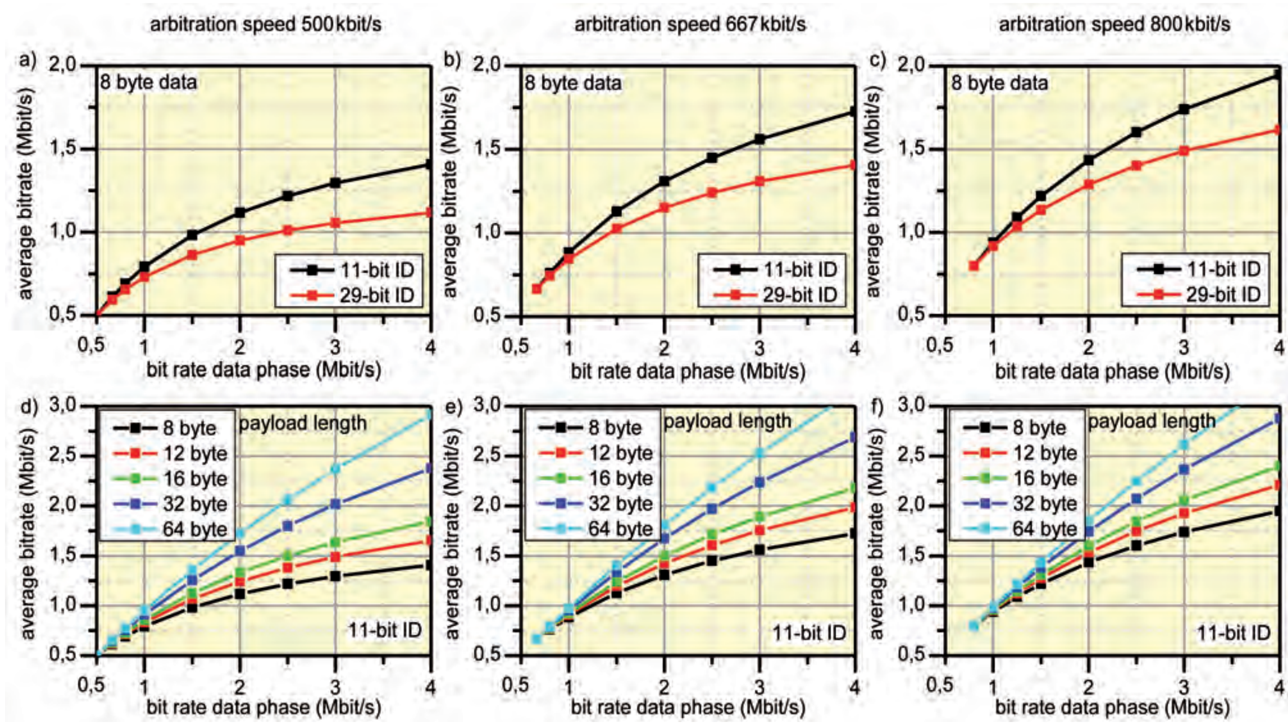


Figure 5: CAN FD average bit rate depending on data phase speed and ID length

tion delay in the network. This value is limited by the CAN protocol mechanisms and the respective bit time settings. In simple terms all nodes within a network need to receive the response of all other nodes to their own signal within a bit time. If the delay is too long, CAN-arbitration and acknowledge mechanisms fail and as a consequence the communication in the CAN-network breaks down completely. To make sure that this does not happen under any circumstance, all communication relationships in a network between all nodes are assessed, e.g. by means of measurement or physical layer network simulation. The maximum delay time in the network (TX to RX) is extracted and the signal integrity of the network is checked as well in order to make sure that the predicted values are stable. If there is ringing in the network that could further enlarge the delay time, the predicted delay values have to be adopted. Figure 6 shows how the evaluation can be done by means of a signal integrity chart. Of course, there will always be an adequately defined safety margin to account for tolerances, EMC or temperature influences.

In the phase of accelerated data transmission of a CAN FD frame, the delay values are not relevant as all other nodes are already synchronized and just listen to the transmitted data. However other key parameters can be identified for CAN FD frames that have not been considered for CAN even though the effects are present in CAN-networks as well. Most important is the asymmetric delay of the received signals in the network that becomes relevant especially for higher bit-rates. This effect is due to the fact that the rising and the falling edges of a dominant signal have different physical preconditions, i.e. the recessive to dominant edge is driven actively whereas the dominant to recessive edge is just released. In the end, depending on the transceiver, used dominant or recessive bits shrink or grow. The exact value may even be dependent on the previously transmitted signals. Figure 7 gives an example of bit asymmetry measured in a real network.

Depending on the bit time settings, bit asymmetry will cause communication errors due to erroneous sampling of the bits. The total asymmetry is a combination of the intrinsic asymmetry of the transceivers and the specific characteristics of the topology. Up to now there is no official tolerance range of the intrinsic asymmetry of the transceivers themselves. Just like symmetric delay values in CAN implementations there has to be an adequately defined safety margin for the asymmetric delay to account for tolerances, EMC or temperature influences.

Integrating CAN FD into E/E architecture

Especially passenger cars use the Autosar (automotive open system architecture) software stack in their ECU software. Autosar follows the principle: cooperate on standards, compete on implementation. In the following, three introduction scenarios for CAN FD into ECU software are discussed. In addition, the impact on the ECU software is explained exemplarily for the Autosar software stack used in passenger cars in which the principle is also valid for other ECU software solutions. All Daimler vehicles (trucks, buses or passenger cars) make use of several network systems that are interconnected via gateways. Thus not only the communication within one network but also the routing between several CAN networks (which might be CAN FD and CAN) or between other networks systems like Ethernet or Flexray has to be considered for the introduction of CAN FD.

Scenario 1

The first scenario considers an increased communication speed, while maintaining an 8-byte payload per frame. This scenario, called CAN FD 8 (payload remains limited to 8-byte), will be introduced for the Autosar release 4.1.1. Figures 8, 10 and 14 show the Autosar software stack. Blue shaded boxes indicate that the respective component has to be adopted. In Scenario 1 only the communication speed is increased, all other communication software mechanisms are maintained. As shown in Figure 8 the only software component that is af- ▷

fected is the CAN driver program:

- Expansion of the CAN-driver to enable the configuration of the second bit-rate,
- Additional attributes required in the system description (bit time settings).

Low busload reserves do not occur on all CAN networks simultaneously which might result in a mixed CAN FD / classic CAN structure inside passenger cars or trucks. As long as a payload length of 8-byte is used in all CAN networks, routing between CAN FD and classic CAN networks is easy, as shown in Figure 9.

PDUs (protocol data unit = bare payload of an application / CAN-frame without any additional control information) can be routed from CAN FD to CAN and vice versa without further considerations and even the same CAN-IDs could be used on both networks. This type of routing called "frame routing" is easy to implement and contained in communication standard software stacks. The routing mechanism to other network systems such as Ethernet or FlexRay would be maintained as well without changes.

It has been shown that CAN FD 8 would approximately double the average bit-rate compared to today's CAN. Thus, Scenario 1 might be a first step towards the introduction of CAN FD.

Scenario 2

The second scenario assumes an increased communication speed and a 64-byte payload per frame. Frames with extended payload (CAN FD 64) provide significant additional bandwidth, which is why the extension of the payload to 64 bytes will be addressed in Autosar 4.2.1. As shown in Figure 10, this scenario has an extensive impact on sev- ▷

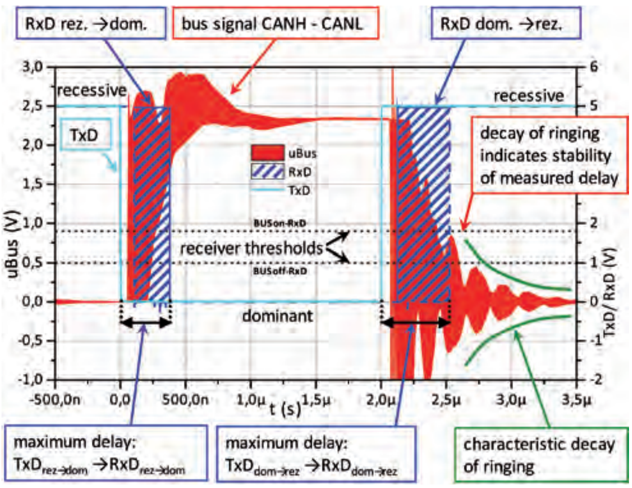


Figure 6: Signal integrity diagram

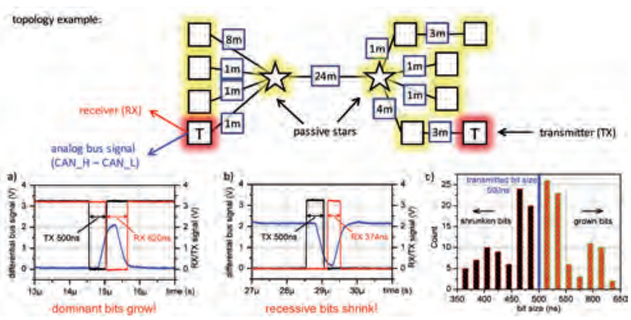


Figure 7: Example for bit asymmetry in a topology with CAN FD 2 Mbit/s

eral software components in the ECU software stack.

In detail the following changes have to be applied:

- Expansion of the CAN driver to enable the configuration of the second bit-rate,
- Expansion of the CAN modules, PDUR, COM and RTE to support 64-byte payload,
- Expansion of the System Description / ECU extracts the ECU to support 64-byte payload,
- Expansion of the configuration tools and generators to support 64-byte payload.

As for CAN the payload length is limited to 8 bytes, generally multiple messages have to be used to transport the original PDU from a CAN FD frame. Unfortunately this type of routing is not implemented in current communication standard software stacks, which means that every single CAN-signal contained in the original PDU has to be treated separately by the routing mechanisms. On the other hand, the possibility to realize separate and independent transmission cycles for the CAN-ID1 to CAN-IDn frames can be achieved in the CAN network, bought dearly by an increased router processor load. Ongoing standardization tends to implement this routing scheme within Autosar specification based on a signal routing scheme in the router.

A slightly different situation occurs if the PDU definition is changed in CAN FD networks. Originally, the PDU identifies the frame content of a CAN-message minus the control information like e.g. message ID, acknowledge bit or stuff bits. For the simplification of the CAN FD to CAN routing, the basic idea is a CAN FD frame containing multiple PDUs with a DLC (data length code) of 8 if compatible to CAN messages.

As shown in Figure 12, the routing scheme looks the same as in Figure 11, routing a CAN FD frame to multiple CAN frames. But here a fixed relation between the PDUs in the CAN FD frames and the routed CAN PDUs/frames can be realized. This PDU routing scheme can be implemented within communication standard software stacks with a high reduction of routing processer interrupt load, resulting in an increased routing capacity. Furthermore, there is a static relation between CAN FD frames and their contained PDUs. This procedure could be used as a migration strategy for vehicle architectures with CAN and CAN FD networks.

Currently the Autosar software has the limitation that only one single PDU can be mapped to one single CAN-frame. If this restriction is maintained, it will be very difficult to use CAN FD effectively. In passenger cars, most applications currently using the CAN network only generate a comparatively small amount of data that is designed to effectively use 8-byte PDUs today. Only a certain amount of applications could really generate large PDUs efficiently using 64-byte CAN FD frames.

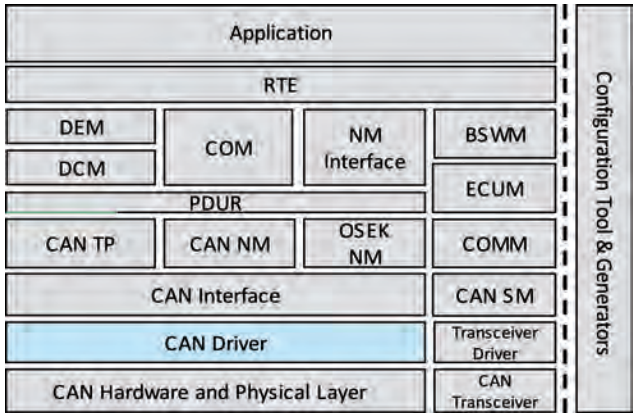


Figure 8: Software stack changes using CAN FD 8

Figure 9: Routing scenario with CAN FD 8

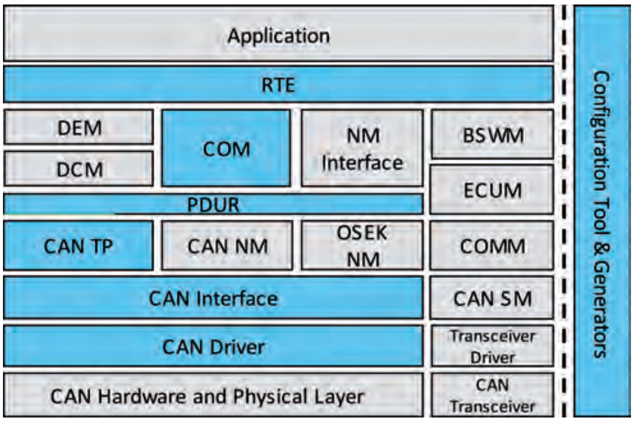


Figure 10: Software stack changes using CAN FD 64

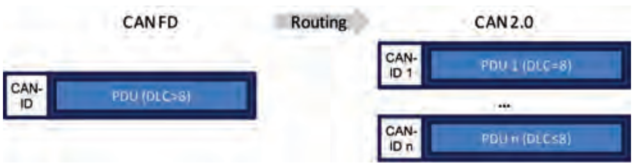


Figure 11: Routing scenario with CAN FD 64

Generally, PDUs should be filled with signals that are transmitted with identical cycle times and not by different software components in order to be able to map the relocatability of functions on a network level. Due to different senders, transmission types, cycle times etc. there is no point in combining arbitrary signals of an ECU into large PDUs in applications.

In trucks and busses, communication is based to a greater extent on cyclic signals and due to technical or legislative requirements, more data has to be exchanged between systems (e.g. exhaust relevant communication between engine controller and exhaust after treatment control unit to ensure Euro IV conformity).

Thus, Scenario 2 theoretically offers the possibility to make use of the extended payload frames CAN FD offers, however depending on the grown structure of the vehicles' software applications and the needed communication relations between systems, only a certain increase of network capacity can be achieved. Therefore, the ability to map multiple PDUs dynamically into one CAN FD frame appears to be an additional requirement.

Senario 3

The third scenario involves expanded communication facilities with flexible PDU mapping. As a first step, multiple 8-byte PDUs could be mapped statically onto one single CAN FD frame. However the efficiency of such a solution would be poor with the transmission mechanisms specified in the present Autosar software package. As an example: In case that four PDUs are statically mapped to one CAN FD frame and only one of the PDUs has been updated, this solution would imply that the entire CAN FD frame has to be transmitted

Einhefter

Vector

Einhefter

Vector

including three PDUs without new information. Like Scenario 2 this solution would make ineffective use of CAN FD possibilities. Therefore, as a second step, a so-called PDU-Header (similar to what is currently specified in Ethernet) could be introduced, see Figure 13.

The introduction of this PDU header allows a dynamic mapping of PDUs onto CAN FD frames. In this case only those PDUs are transmitted in a CAN FD frame whose contents have actually changed. There is no redundant transmission of unchanged PDUs. The PDUs that are contained in a current CAN FD frame can be identified clearly by means of the PDU header. Furthermore the length of the CAN FD frame can be adapted dynamically depending on the current communication needs. This method allows using the possibilities of the CAN FD technology in a quite effective manner, even though some bandwidth gets lost for the additional PDU headers. Secondly, it fits into the grown structures of the ECU software structure. Figure 14 again highlights the software components that have to be adopted.

It has to be mentioned that the PDU header concept also implies the loss of bandwidth due to the PDU headers in the CAN-message's payload. E.g. if five 8-byte PDUs are transmitted in an 64-byte CAN FD frame, 60 bytes of the frame are really used and another 20 bytes get lost for the PDU headers resulting in an effective usage of 62,5 % compared to the complete usage of 64-byte payload. This would especially apply to the grown applications in the E/E-architecture, whereas new applications could make use of larger PDUs where the loss is much less, e.g. approximately 93 % efficiency for a single 60-byte PDU and 4-byte header.

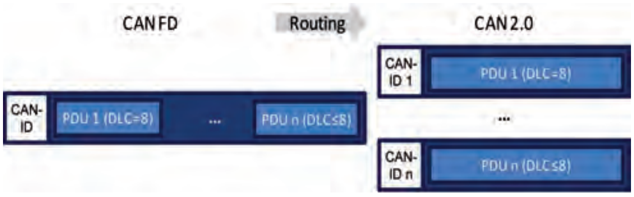


Figure 12: Routing scenario with CAN FD 64

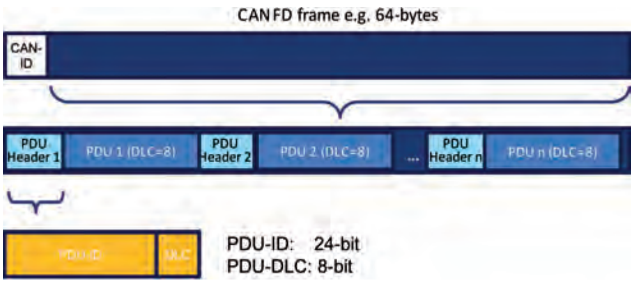


Figure 13: PDU header concept

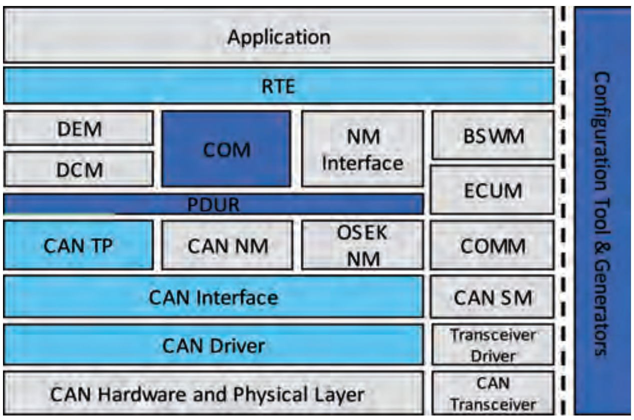


Figure 14: Software stack changes using CAN FD 64 and PDU routing Routing of CAN-frames between networks containing CAN FD and classic CAN using the PDU header concept is shown in Figure 15.

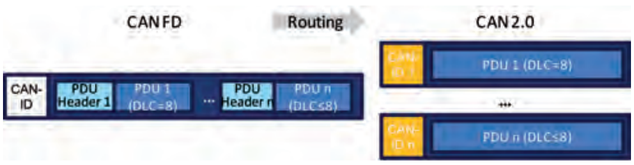


Figure 15: Routing scenario with CAN FD 64 and PDU header concept

Figure 15 shows the routing of a CAN FD frame with a DLC > 8 from a CAN FD network towards a classic CAN network. In this case, the router gains more flexibility and reduces lookup table memory requirements as the CAN FD frame can carry the classic CAN destination CAN-ID information within the PDU header. Only in case of a multi-router with more than two CAN-networks, the router needs a lookup table for the selection of the destination network (e.g. CAN-ID1 should be routed to CAN network No. 1, while CAN-ID2 should be routed to classic CAN network No. 2 only). Of course this destination network information also has to be available at the router in the previous discussed routing schemes. Discussing the routing schemes from a CAN network to a CAN FD network, the principle stays the same. A CAN FD frame is combined from multiple CAN frames depending on the PDU arrangement and the router scheme (PDU routing or signal routing).

But another aspect rises up as for the case of DLC > 8 or multiple PDUs on CAN FD: the timing of the arriving CAN frames needs to be considered. Multiple approaches may be used varying in routing latency for the different PDUs. Depending on the specific application of the routed signals, the appropriate routing timing scheme has to be selected (e.g. starting routing process at first incoming CAN frame, starting after arrival of last incoming CAN frames). In every case, buffer memory for single signals or complete PDUs has to be provided within the router. ◄

Yes we can ... do without CAN

Unfortunately CiA (CAN in Automation) has currently only 29 members located in the US. This article briefly introduces their CAN-related business activities.

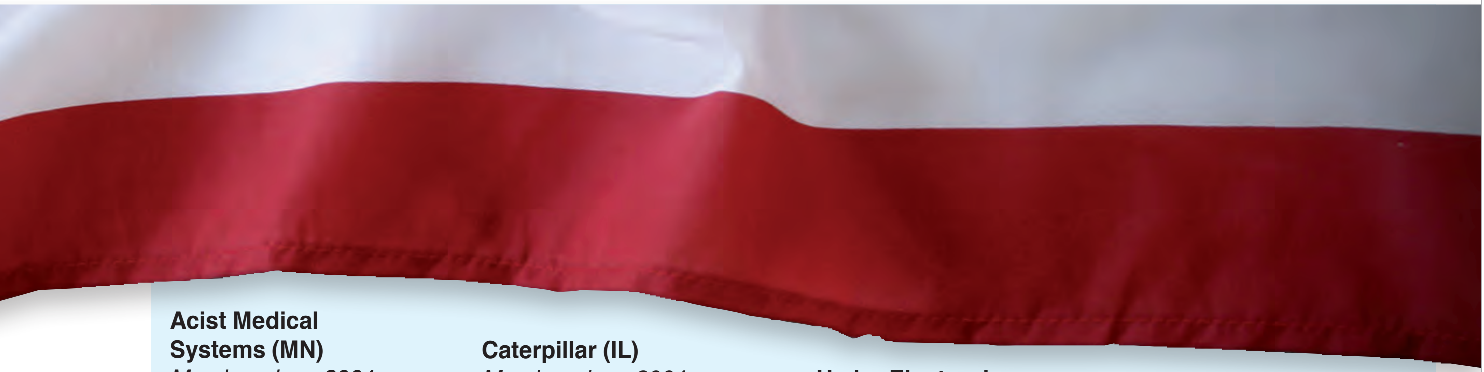
The number of CiA members headquartered in the US is quite small compared to the importance of the US industries. Allen-Bradley (AB) and Honeywell started early in the 90s to introduce CAN-based networks in industrial automation. In the very beginning, they worked together. But at one point of time, both enterprises went their separate ways: AB developed Devicenet while Honeywell developed SDS (Smart Distributed System). While both protocols were standardized internationally, SDS has disappeared in the meantime. Devicenet is still in use, but not heavily promoted by the originator and the international organization "Open Devicenet Vendor Association" (ODVA), which focuses more on Ethernet/IP. Completely independent from industrial automation, the SAE (Society of Automotive Engineers) specified the CAN-based J1939 application profile, used mainly in diesel engine powered vehicles.

Then there is CANopen. It is the internationally standardized (EN 50325-4) CAN-based higher-layer protocol for embedded control systems. The set of CANopen specifications comprises the application layer and communication profile as well as application, device, and interface profiles. CiA's CANopen application layer did not fly in the US though – with some exceptions, of course. Medical equipment manufacturers were the first to adapt CANopen. Other CANopen users were builders of small machines such as industrial printing machines and laboratory automation. Today, CANopen is also used in various US industries. Most of the North American CiA members are device suppliers. It is unusual for US companies to join a non-profit association located in Germany. Therefore, a couple of years ago CiA established an office in the US. Unfortunately, this office was not as successful as expected: it achieved only a very small increase in membership. After two years, CiA gave up and closed the office.

Figure 1: The bridge that CAN needs to reach the US is not finished yet

The number of current CiA members in the US is relatively stable. However, fluctuation is high. There are a few US members who understand that investment in a non-profit association will not return within a quarter. Hans J. Kuffer, vice president of business development at Inicore, which is CiA's oldest US member (since 2002) says: "When we first opened up our office here in the US (1997), we noticed that CAN was not as popular as in Europe. Where do potential applicants of CAN go first when they are looking into using it, to your [CiA's] website. […] Being a member of CiA was a marketing decision, to become visible to new and existing CiA members and future applicants of CAN. At that time it was also important to show our commitment to the CAN standard and to the product. […] Being a member gives us access to all the standards; we even can participate in various work groups and help define new specifications and standards. We can publish news about our CAN products and its application. Also, the CiA organization helps drive the market for the use of CAN in different markets outside the original automotive and helps to promote the product through exhibition, through educational seminars and publications all over the world."

Besides healthcare, the mobile machine industry increasingly uses CANopen. During the Conexpo, US suppliers presented a range of CANopen products. Another hot topic is CAN FD, which is driven by General Motors and heavily supported by CiA in respect to international standardization as well as the development of user guidelines and design recommendations.



Acist Medical Systems (MN)
Member since 2004
Produces contrast media injectors
www.acistmedical.com

Air-Weigh (OR)
Member since 2012
Provider of on-board weighing products including CAN J1939 interfaces
www.air-weigh.com

Altronic (OH)
Member since 2013
The manufacturer of ignition and control systems for industrial engines offers products featuring CANopen and J1939
www.altronicinc.com

American Science & Engineering (MA)
Member since 2011
The provider of X-ray technologies for ports, borders and military uses a CANopen vendor-ID
www.as-e.com

Angio Dynamics (NY)
Member since 2013
Focus on improving patient care through the innovation of medical devices
www.angiodynamics.com

Arcus Technology (CA)
Member since 2014
Provides a range of motion control products with CAN/CANopen interfaces
www.arcus-technology.com

Bayer Healthcare Medical Care (PA)
Member since 2004
Healthcare equipment manufacturer with German roots
www.medrad.com

Caterpillar (IL)
Member since 2004
Offers construction machines and stationary power Generators using CAN networks
www.cat.com

Copley Controls (MA)
Member since 2010
Supplies CAN/CANopen tools, among other things
www.copleycontrols.com

Curtis Instruments (NY)
Member since 2008
Offers CANopen devices for mobile machines
www.curtisinst.com

Danlaw (MI)
Member since 2014
Provides a CAN physical layer tester that helps to test ECU to OEM standards for CAN connectivity, among other things
www.danlawinc.com

Divelbiss (OH)
Member since 2014
Designs and manufactures industrially hardened electronic control solutions such as Devicenet controllers
www.divelbiss.com

FTI Flow Technology (AZ)
Member since 2012
Subsea equipment manufacturer implements CANopen
www.ftimeters.com

Glentek (CA)
Member since 2009
Manufactures servo amplifiers which communicate via CANopen
www.glentek.com

Hunting Subsea Technologies (TX)
Member since 2010
Provides CAN-based products and services to the upstream oil and gas industry
www.hunting-intl.com

Hydro Electronic Devices (WI)
Member since 2013
Supplier of off-highway vehicle equipment based on J1939 and CANopen
www.hedonline.com

Inicore (CA)
Member since 2002
System design house providing CAN IP cores and FPGAs
www.inicore.com

Javad GNSS (CA)
Member since 2009
Provides for example OEM board-level products featuring CANopen
www.javad.com

Macro Sensors (NJ)
Member since 2014
Designs and manufacture position sensors for linear position & displacement measurements
www.macrosensors.com

Mallinckrodt Pharmaceuticals (OH)
Member since 2004
Makes contrast media injectors
www.mallinckrodt.com

OEM Controls (CT)
Member since 2009
Manufacturer of electrohydraulic controllers and control systems
www.oemcontrols.com

Olympic Controls (CA)
Member since 2014
Vendor of electric fuel pumps
www.occorp.com

PG Trionic (MA)
Member since 2008
Vendor of CAN-based devices for mobile machines
www.trionicusa.com

Quicksilver Controls (CA)
Member since 2005
Manufactures motion controllers with CANopen interfaces
www.quicksilvercontrols.com

Real Time Automation (WI)
Member since 2006
Offers a range of gateways and software solutions for Devicenet
www.rtaautomation.com

Red Lion Controls (PA)
Member since 2004
Produces Devicenet and CANopen industrial control devices
www.redlion.net

Skoflo Industries (WA)
Member since 2008
Implements CANopen in their subsea electronic products
www.skoflo.com

Stellar Technology (NY)
Member since 2007
Provides CAN-based sensors for different kinds of applications
www.stellartech.com

Teledyne CDL (TX)
Member since 2013
Designs and manufactures a variety of sensors including gyrocompasses, attitude and heading reference systems, and inertial navigation systems
www.cdltd.net

The susceptibility of CAN to latent faults

A qualitative evaluation of the susceptibility of CAN communication to latent faults can extend the scope of functional safety assessment of in-vehicle embedded systems.

Author
Lukusa Didier Kabulepa
Institute for Embedded Systems
Mannheim University of Applied Sciences
John-Deere-Straße 85
DE-68163 Mannheim
Tel.: 0621-292-6854
Fax: 0621-292-6-6854-2

Links
www.hs-mannheim.de

References

Additional information

- B. Gaujal, N. Navet, "Fault Confinement Mechanisms on CAN: Analysis and Improvements", IEEE Transactions On Vehicular Technology, 2001.
- N.Navet, Y.-Q. Song, "Performance and Fault Tolerance of Real-Time Applications Distributed over CAN (Controller Area Network)", Submission for the CiA Awards 1997.
- E. Tran, "Multi-Bit Error Vulnerabilities in the Controller Area Network Protocol", Carnegie Mellon University, Pittsburgh PA, USA, 1999.

The ISO 26262 standard for functional safety has replaced the IEC 61508 generic standard in the development process of automotive embedded systems. ISO 26262 has brought the concept of latent multiple-point faults in addition to the commonly considered single-point faults. Therefore, it is worthwhile to assess the robustness of previously specified automotive E/E (Electric/Electronic) systems against latent faults.

Nowadays automotive manufacturers are facing the big challenge of integrating a growing amount of software and electronics commonly delivered by numerous suppliers. This increasing complexity of automotive embedded systems and related networks calls for a more rigorous approach to system development. With regards to minimizing the risk of physical injury or environmental damage, stringent functional safety requirements must be stipulated and considered in conjunction with other specific automotive requirements (e.g. high quality, target cost and delay objectives). The increasing importance of safety systems in modern cars strengthens the need for more robust and dependable electronic and software components. Risks resulting from hardware, random failures, or software systematic errors must be reduced so far as reasonably practicable throughout the foreseeable lifetime of the products.

The automotive industry now applies the ISO 26262 standard as best practice for functional safety. According to ISO 26262, car manufacturers and suppliers are expected to agree on producing safety cases for automotive embedded systems. Every safety case must provide a clear, comprehensible and defensible argument, supported by qualitative and quantitative evidence that a system is acceptably safe to operate in a particular environment. Typically based on engineering judgment rather than on formal logic approach, the argument of a safety goal must show that steps have been appropriately taken to deal with the hazards caused by malfunctioning behavior of a system. ISO 26262 defines a process framework and a procedure model along with necessary activities and methods to be considered throughout the life-cycle of safety critical systems in road vehicles.

A key concept of ISO 26262 relies on the specification of the "Automotive Safety Integrity Level (ASIL)" as one out of four discrete levels, each reflecting a set of necessary requirements, recommended diagnostic techniques and architectural constraints for avoiding an unreasonable residual risk. ASIL-D represents the most stringent level and ASIL-A indicates the least stringent level. After risk assessment and hazard analysis of operational situations, an ASIL is assigned to each safety goal. Hence the required ASIL-specific values of the single-point fault metric and latent fault metric must be met for the functional path allocated to a given safety goal. Data transmission of safety-critical data over CAN does belong to various functional paths targeting ASIL-C or -D in modern cars.

Prior to the specification of ISO 26262, the functional safety analysis of a CAN network was based on the consideration of single-point faults [4, 5, 6, 7]. To cope with ISO 26262, it is highly desirable to extend the related works by addressing the effects and the detection capability of latent faults in automotive networking systems. ISO 26262 defines a latent fault as a multiple point fault whose presence is not detected by a safety mechanism nor perceived by the driver within the multiple point fault detection interval. After a time span representing this interval, an undetected latent fault may allow another fault to cause a hazard. This work aims at qualitatively assessing the susceptibility of CAN communication to latent faults. ▷

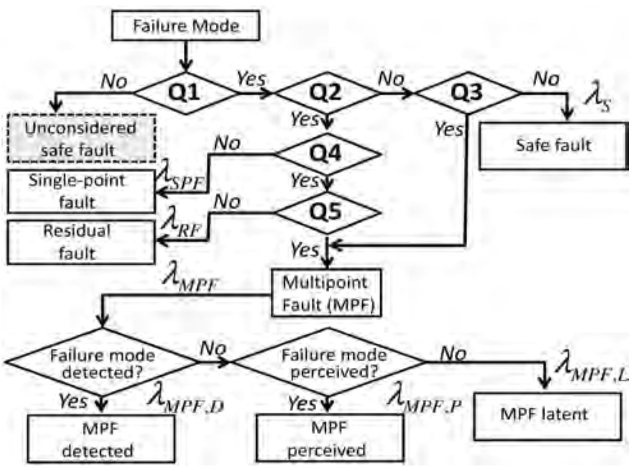


Figure 1: Classification of failure modes [2]

CAN built-in safety measures

Distributed automotive embedded systems mainly use a CAN network to transfer messages to various ECUs (Electronic Control Units). As a CAN network is also intended to support data exchange amongst safety-critical functions, robust safety measures are anchored in the related protocol. In light of the transmission errors listed in IEC 61508 and ISO 26262, the intrinsic CAN safety measures achieve a high error detection coverage. In some cases, this coverage of hardware safety mechanisms can be improved by means of software-based error detection methods.

known by the transmitter and the intended receiver.

Frame form check: Predefined bit positions (i.e. CRC delimiter, ACK delimiter, EOF bit) within a CAN frame must always be transmitted as recessive bits. If a receiver detects a dominant bit in one of these positions a form error will be flagged. Additionally the RTR and IDE bits must always exhibit a predefined value consistent with the actual format of the frame transmitted over the serial bus. The frame form check detects some data corruption errors, partial message deletion, and timing errors such as excessive jitters.

This error detection mechanism detects also erroneous signal activities on

Table 1: LFM (latent fault metric) related requirements

Safety levels	ASIL D	ASIL C	ASIL B
Safety requirements	LFM > 90 %	LFM > 80 %	LFM > 60 %

The CAN protocol implements five error detection mechanisms acting either at message level or at bit level:

- Cyclic redundancy check (CRC)
- Frame form check
- Acknowledge check
- Bit-monitoring
- Bit-stuffing

Cyclic redundancy check (CRC): A 15-bit CRC sequence is computed by the transmitting node for each message and verified upon reception. The selected hardware CRC polynomial guarantees a minimum Hamming distance of 6 in the unstuffed bit sequence of a CAN frame. The application of the CRC code represents a sufficient measure against data corruption. Thus the CAN hardware CRC is not a sufficient safety measure against masquerade errors. An additional CRC implementation in software may help overcome this limitation of the basic CAN CRC if, for instance, the software CRC computation is not limited to the transmitted frame, but also applies to unsent data (e.g. a special key) a-priori

the CAN network if they are not compliant with specified CAN frame forms.

Acknowledgment check: Every CAN transmit node expects an acknowledgement at the ACK bit position. An acknowledgement error is flagged if a transmitter does not sample a dominant bit during the ACK slot. As a feedback signal, the acknowledge bit allows the transmitter to detect a partial message deletion or insertion occurred after the SOF bit. Moreover, some types of data corruption, delays and excessive jitters can be detected by means of the acknowledgment check.

Bit-monitoring: Every sending node monitors the transmitted bit level on the bus. A bit error is flagged if the monitored bit level is different from the transmitted bit level. Bit-monitoring facilitates the rapid detection of data bit corruption and timing errors.

Bit-stuffing: The bit-stuffing method states that a bit of opposite polarity must be inserted after every five consecutive bits of the same polarity. A stuff error is indicated if six consecutive bits of the same polarity are detected between the start of frame (SOF) and the CRC delimiter. The bit-stuffing method detects some types of data bit corruption on CAN networks.

These error detection mechanisms are accompanied by an efficient error signaling scheme in CAN networks. Each node starts sending an error frame to other nodes upon error detection. Since the error message does not obey the bit-stuffing rule, it can be notified by any node that should have missed to identify a previous error condition. Therefore a CAN network error remains undetected only if it escapes the error detection mechanisms of all active nodes within the network. The scope of error detection is not limited to the transmitter and the intended receiver.

In safety-critical applications, the built-in safety measures of the CAN protocol are typically expected to be complemented by software-based error detection methods in the safety layer in order to achieve a better qualitative coverage of the transmission errors mentioned in IEC 61508 and ISO 26262. An evaluation of software-based safety mechanisms is beyond the scope of this work since they are commonly based on proprietary implementations of software measures such as alive counter, time stamp, or software-CRC. In contrast to hardware components, software implementations are only prone to systematic errors.

Assessment of latent faults

A latent fault is defined in ISO 26262-1 as a multiple-point fault whose presence is not detected by a safety mechanism nor perceived before the underlying combination of faults can lead to a violation of a safety goal. The evaluation of all possible latent multiple-point faults in a safety-relevant system may result in a search space beyond any reasonable range. A realistic interpretation of latent faults consists of considering faults of safety mechanisms as contributors to plausible multiple-point faults. A fault within a safety mechanism does lead to a violation of a safety goal only in combination with another fault.

Basically a latent fault scenario is assessed to be realistic if a fault in a safety mechanism remains undetected for a long time beyond the fault tolerance time of the system. In accordance with the CAN protocol specification [3], CAN safety mechanisms are not required to be tested neither at start time nor periodically. Hence a fault affecting a CAN safety mechanism will remain undetected in normal operational mode, although modern functional safety standards such as ISO 26262 implicitly require to only rely on safety mechanisms that can be tested at run-time. However any combination of a CAN safety mechanism fault with a transmission error will not necessarily result in a violation of a safety goal since CAN safety mechanisms offer overlapping coverage ranges of error detection. Different safety mechanisms may detect the same error types. In case a safety mechanism is unavailable, the targeted error may be detected by another safety measure. Therefore the susceptibility of CAN communication to latent faults must be assessed by considering the cooperation of the safety mechanisms.

Unavailable CRC error detection: The CRC is the most efficient protection scheme against data corruption. If the CRC error detection is not operating in a CAN node, data corruption occurring along the transmission medium (e.g. twisted pair cables) or inside the transmitting node will be detected by at least one of the remaining nodes within the CAN network. All CAN nodes belonging to the same network do verify the consistency of the CRC checksum of every transmitted message. Upon CRC error detection, each node generates and sends an error message. In a CAN network embodying N nodes, N-1 nodes participate to the CRC error detection process regardless of the targeted receiver node. Hence the unavailability of CRC error detection is compensated by other nodes. However a data corruption inside the receiving node (e.g. in a shift register prior to the CRC module) cannot be detected by another node.

Missing frame form check: The a-priori known values of the delimiters (e.g. CRC delimiter) in a CAN network frame are used for internal synchronization purposes. An undetected erroneous delimiter may cause unreliable behavior of a CAN protocol engine. A frame form error taking place on the transmission medium can be detected by bit monitor of the transmitting node.

Missing acknowledge check: The transmitting node will tend to repeatedly send the same messages after assuming that no message has been properly acknowledged. This behavior does impede the availability of the CAN networks while not directly impacting the safety integrity. The node will enter the bus-off state if the number of transmission error if the number of transmission errors exceeds the predefined threshold.

Unavailable bit-monitoring: The transmitter cannot detect bus errors. However the CRC and frame form checker safety measures fill this gap at receiver side. Bit errors are detected by the CRC and delimiter errors are indicated by the frame form checker.

Unavailable bit-(de) stuffing monitoring: If the received bits are not properly destuffed, the data bits will exhibit a lack of consistency with regards to the checksum. Hence a CRC error will be indicated.

Quantitative Considerations

Failure modes in a hardware element can be classified as shown in the figure below [1]. Accordingly the failure rate λ of a safety relevant hardware element can be expressed as the sum of the contribution of different failure modes:

$$\lambda = \lambda_{SPF} + \lambda_{RF} + \lambda_{S} + \lambda_{MPF}$$

where a multipoint failure (MPF) can be detected, perceived or latent:

$$\lambda_{MPF} = \lambda_{MPF,D} + \lambda_{MPF,P} + \lambda_{MPF,L}$$

The robustness against latent faults is quantitatively assesses by computing the latent fault metric (LFM):

$$LFM = 1 - \frac{\sum_{SR\,HW}(\lambda_{MPF,L})}{\sum_{SR\,HW}(\lambda - \lambda_{SPF} - \lambda_{RF})} \quad (1)$$

As stated in Table 1, a high latent fault metric and hence a low contribution of latent faults are required for a high ASIL. According to ISO 26262, the LFM must be evaluated for a specified safety goal. Commonly, a safety goal applies to a whole functional signal processing path starting for instance with the collection of sensor input data and ending with the actuator commands. A detailed LFM computation is beyond the scope of this paper since data transmission over CAN networks does represent only a portion of a signal path considered for a safety goal.

As shown in the qualitative analysis, the occurrence of latent faults in a CAN node represents a ▷

realistic scenario commonly underestimated in the state-of-the-art functional safety assessment of automotive systems. As a failure of a safety mechanism cannot be detected in the current implementation, a higher contribution of latent faults to the overall failure should be taken into account.

Let the relative variation in the contribution of latent faults be represented by ΔMPF,L as shown in the following equation:

$$LFM = 1 - \frac{(1+\Delta_{MPF,L})\sum_{SR\,HW}(\lambda_{MPF,L})}{\sum_{SR\,HW}(\lambda - \lambda_{SPF} - \lambda_{RF})} \qquad (2)$$

where -1 ≤ ΔMPF,L. A negative relative variation can be considered only if additional mechanisms are implemented to detect latent faults. Otherwise this relative variation is positive if a failure mode of a safety mechanism is later identified to be undetectable at run-time. The same range of relative variations ΔMPF,L causes a change of up to 12 % in the LFM of ASIL-B safety goals. Hence, if the susceptibility of CAN communication to latent faults is not considered in the functional safety analysis of automotive systems, the LFM value of the related safety goal may be in fact lower than the computed value. Latent faults such as a CRC failure can cause a significant reduction of the LFM of a safety goal.

A failure of a CAN safety mechanism cannot be detected in the current implementation of a CAN controller since the functional correctness of a CAN safety mechanism is not testable at run-time. Although CAN safety mechanisms are complementary to a given extent, the susceptibility of CAN communication to latent faults remains a realistic issue, which is commonly ignored in the

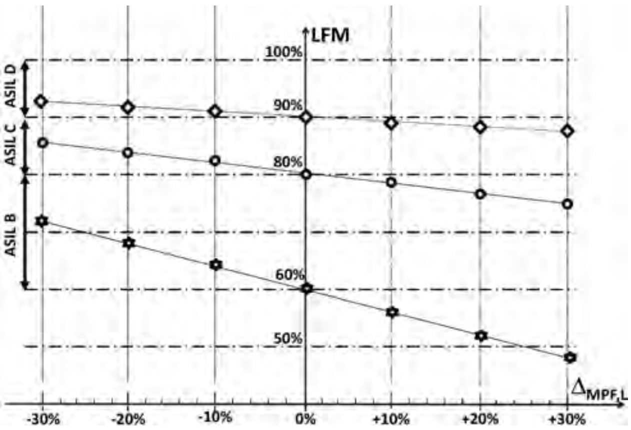


Figure 2: Dependency between the relative variations $\Delta_{MPF,L}$ in the contribution of latent faults and the latent fault metric (LFM)

By considering the LFM boundaries allocated to ASIL categories (see Table 1), variations in the contribution of latent faults have been evaluated with regards to their impact on the LFM. Relative variations ΔMPF,L of up to 30 % are illustrated in Figure 2. The higher the LFM, the lower the impact of the relative variation ΔMPF,L. For ASIL-D safety goals, LFM changes are lower than 3 %. The same state-of-the-art functional safety analysis. This work points out that a valuable discrepancy between the computed and the actual values of the latent fault metric (LFM) may occur if failures of CAN safety mechanisms such as CRC are not taken into account. ◄

Tips and tricks for the use of CAPL

CAPL is a programming language available in the software tools CANoe and CANalyzer. In three consecutive articles, CAPL fundamentals will be discussed as well as tips for all levels of user knowledge.

Authors

Marc Lobmeyer

Roman Marktl

Vector Informatik GmbH
Ingersheimer Str. 24
DE-70499 Stuttgart
Tel.: +49-711-80670-0
Fax: +49-711-80670-111

Links
www.vector.com

This first part focuses on the basics of CAPL. It is primarily intended for those who are new to this language; however, it also offers a few insights for well-informed users into the motivation behind individual CAPL constructs. The second part will discuss advanced functionalities of CAPL. Finally, the third part will address performance and memory needs and offers tips and tricks on using databases and associative arrays.

For over 20 years – it was initially used in CANalyzer for DOS – CAPL has been used to implement a broad bandwidth of tasks: from simple stimuli to the simulation of complex bus nodes. In the following, CANoe is illustratively named for the two products CANoe and CANalyzer. The goal of CAPL has always been to solve specific tasks as simply as possible. Typical tasks are reacting to received messages, checking and setting signal values and sending messages. A program should restrict itself to precisely these things and not require any additional overhead.

Many programming tasks that CANoe users typically perform might actually be as brief and trivial as the example presented below – of course many other tasks are not so trivial. That is why CAPL has been continually extended over the years to be a tool that can also solve complex tasks according to the principle “as simple as possible”.

“CAPL” is an acronym for Communication Access Programming Language. The original focus on CAN has long been extended to all automotive bus systems such as LIN, Flexray, Most, J1587, as well as a few others like Arinc and CANopen.

As in many other languages, the syntax of CAPL is closely based on the syntax of the C language. Those who are familiar with C, C#, or various modern script languages will quickly be quite comfortable using CAPL. However, a few unique aspects distinguish a CAPL program from a C program:

CAPL programs are event-driven. This means that they consist of individual functions, each of which reacts to an event within the system under analysis: receipt of a message, change of a signal, expiration of a timer, or even a change in the environment. To react to the message “EngineState”, for example, you would use: “On message EngineState” (Figure 1).

CAPL programs use specific databases for the concepts of the system under analysis. Messages and signals get their names there, and these names can be used directly in the program code. In Figure 1, they are the names “EngineState” for a message and “EngineSpeed” for a signal in this message.

CAPL programs do not give the user any pointer types to use. Right from the outset this excludes numerous potential programming errors and causes of program crashes, such as those that frequently happen in C programming. Nonetheless, since pointers – aside from their susceptibility to errors – also represent a very powerful concept, CAPL provides a substitute for some things, e.g. associative arrays as a substitute for dynamic memory.

One important property that CAPL shares with C should be mentioned: CAPL is always compiled, i.e. it is converted to efficiently executable and flexible machine code.

Example: a simple CAPL program

In Figure 1 a simple CAPL program is presented, which ▷

CAPL

CAPL is a procedural programming language similar to C, which was developed by Vector Informatik. The execution of program blocks is controlled by events. CAPL programs are developed and compiled in a dedicated browser. This makes it possible to access all of the objects contained in the database (messages, signals, environment variables) as well as system variables. In addition, CAPL provides many predefined functions that support working with the CANoe and CANalyzer development, testing and simulation tools.

```
1.   variables
2.   {
3.     const long kOFF = 0;
4.     const long kON = 1;
5.   }
6.
7.   on message EngineState {
8.     @sysvar::Engine::EngineSpeedDspMeter = this.EngineSpeed / 1000.0;
9.   }
10.
11.  on message LightState {
12.    if (this.dir == RX) {
13.      SetLightDsp(this.HeadLight,this.FlashLight);
14.    } else {
15.      write("Error: LightState TX received by node %NODE_NAME%");
16.    }
17.  }
18.
19.  SetLightDsp (long headLight, long hazardFlasher) {
20.    long tmpLightDsp;
21.
22.    tmpLightDsp = 0;
23.    if(headLight == kON)
24.      tmpLightDsp = 4;
25.    if(hazardFlasher == kON)
26.      tmpLightDsp += 3;
27.    @sysvar::Lights::LightDisplay = tmpLightDsp;
28.  }
```

Figure 1: A simple example of a CAPL program

performs one of the basic tasks of a bus monitoring tool: it listens to traffic on the bus and prepares a couple of events on the bus for observation/monitoring by the user. This is a shortened, sample CANoe program: “Display.can” from the sample “Easy.cfg”. In the following, first the overall functionality is briefly summarized, and then the individual sections are described in more detail.

Task description

The task is to observe a CAN network whose elements – e.g. bus nodes, messages and transported signals – are described in a database. When the “EngineState” message is received, then the Engine-Speed signal it contains is conditioned for display on a display panel, and it is routed to the panel. When the “LightState” message is received, the “HeadLight” and “FlashLight” signals it contains are conditioned for display on a panel, and they are routed to the panel for graphic display.

Description of the program

The line numbers are not part of the CAPL program and are only inserted here to make it easier to reference individual lines or sections. To achieve the most compact representation possible, opening brackets were not placed on a separate line. In a CAPL program, it is possible to define global variables and constants. This is done in the “variables” section (lines 1 to 5). These constants and variables are globally defined for this program: they can be used anywhere in the program, but not in other programs within the same CANoe application. The other sections define reactions to events (lines 7 to 17) and an auxiliary function (lines 19 to 28).

Lines 7 to 9 show a minimal form of a message event procedure. This function is called whenever this message has been transmitted on the bus. In reference to CAN, this means that the precise time point is the TX or RX interrupt of the CAN controller, i.e. immediately after correct transmission of the message. The message that triggers the call of the particular function is referenced by “this” syntax.

In line 8, the value of the “EngineSpeed” signal is read out from the received message (“this”) and is assigned to a system variable with a conversion (/1000.0).

Lines 11 to 17 show a message event procedure for the “LightState” message, which transmits the information for a turn signal flasher. Its processing is similar to that of the “EngineState” message with the following unique aspects: In line 12, the direction flag (.dir) is now checked in the message (“this”) that has been just transmitted. Only received messages should be considered in this program (value RX), because a message sent by the node itself would also trigger an event procedure (value TX). In this case, an error message would be output in line 15.

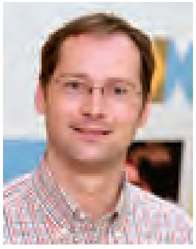
Since conditioning of the signal for display on the user interface (a panel on which different states are shown by different bitmaps) is somewhat more complex, its implementation is outsourced to a separate function: In line 13, “SetLightDsp” is called with the two message signals that are needed as parameters.

Finally, lines 19 to 28 define a separate function, which writes different values to the system variable “LightDisplay” in the “Lights” name space according to the value of the transmitted signal. In this demo configuration, this variable then selects the appropriate bitmap on a display panel. ◀

CAN FD: Impact on system design

The development of CAN FD is by no means at its end. Bernd Elend takes a look at issues that arise when CAN FD and the physical layer interact.

Author



Bernd Elend
NXP Semiconductors
Stresemannallee 101
DE-22529 Hamburg
Tel.: +49-40-5613-2663
Fax: +49-40-5613-3524

Links
www.nxp.com

CAN Newsletter (print)

1/2014: Magnus-Maria Hell (Infineon) - Physical layer in the CAN FD world

In the previous edition of the CAN Newsletter, Magnus Hell discussed the new CAN physical layer parameter's "propagation delay symmetry". This parameter covers a new and important aspect when applying CAN FD with bit-rates that exceed 1 Mbit/s in the data field. Many of the existing transceivers meet this new parameter for bit-rates up to 2 Mbit/s. Updated datasheets of several well-known CAN transceivers are expected soon. The NXP Mantis series does meet this requirement even up to 5 Mbit/s. However, there are more interesting issues in the interaction of the CAN FD protocol and the physical layer than just this parameter. These items will be discussed in this article.

CAN FD and cable length

Lowering the bit-rate for arbitration allows for longer cable length between the most distant nodes in a CAN system. With classic CAN, this results in less bandwidth. However, with CAN FD the bandwidth can be constant or even increased by compensating the loss of bandwidth during arbitration by a higher bit-rate in the data field. Figure 1 illustrates a classic CAN frame at 500 kbit/s and a CAN FD frame with a bit-rate of 400 kbit/s in arbitration and 800 kbit/s in data field. For this example we can roughly calculate: A classic CAN frame with an 11-bit ID and 8-byte data lasts 200 μs at 500 kbit/s. A CAN FD frame with an 11-bit ID and 8-byte data lasts around 160 μs (in both cases plus stuff bits, acknowledgement, and end-of-frame field).

This example allows for roughly 25 % more bandwidth and some 10 m additional cable length.

Figure 1: CAN FD allows higher bandwidth, even with slower arbitration

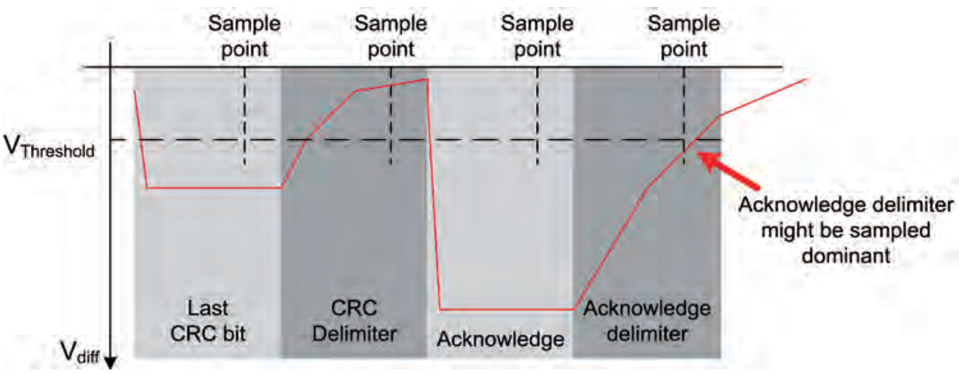


Figure 2: Differential bus voltage during acknowledge slot

CAN FD and acknowledgement

Classic CAN networks, with long overall cable lengths, sometimes suffer from the fact that the dominant-to-recessive transition at the end of the dominant acknowledge bit can be extremely slow. This is caused by the effect that all but one node (the transmitter) are concurrently sending the dominant acknowledgement bit, which results in the highest differential voltage that ever occurs on the bus, as depicted in Figure 2. The higher the differential voltage is, the longer the transition from dominant to recessive lasts. Due to this effect, some nodes may sample the acknowledge delimiter as dominant, which is seen as a form error in classic CAN. The consequence is the sending of an error flag. Thus the message is invalidated and has to be repeated. This problem is removed by CAN FD as receiving a dominant acknowledge delimiter is accepted after a CAN FD frame and is not flagged as an error.

CAN FD and EMC

Increasing the bit-rate in the data field has of course an effect on the EMC behavior of the entire system. Emissions are defined by the signal shape and by the number of signal edges per time. So the envelope of the emissions rises by 6 dB when the number of edges per time is doubled. This is a matter of physics and thus independent of the transceiver and independent of measure- ▷

The WAGO-I/O-SYSTEM 750 –
One System for All Applications!

CAN be connected to any fieldbus

CAN


750-337


750-338


750-347


750-348


750-837


767-2501

750-658

NEW

750-658 CAN Gateway:

- Gateway for all CAN protocols
- CAN 2.0A, CAN 2.0B
- Supports all CAN baud rates and autobaud
- Operating modes: sniffer, transparent, mapped I/O

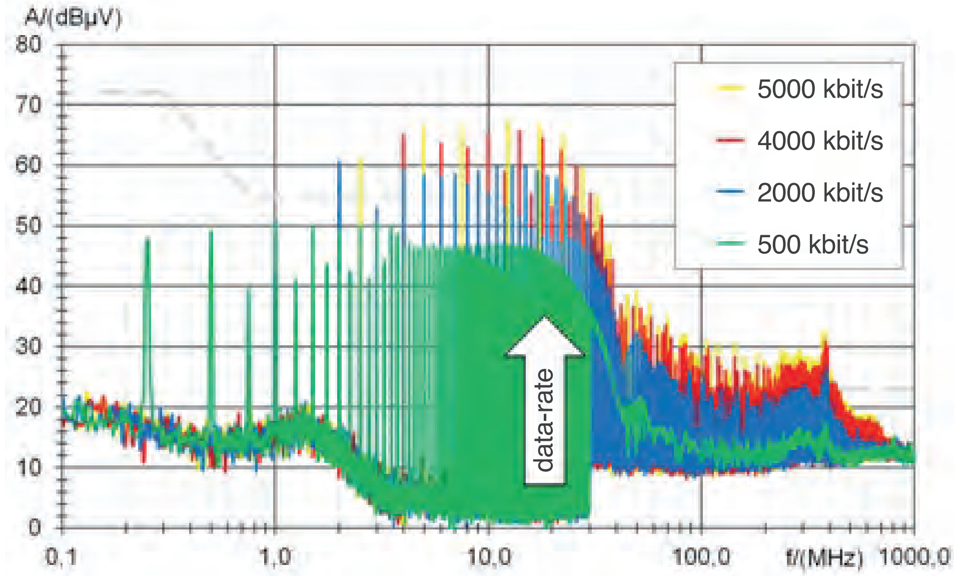


Figure 3: RF emission in dependence of the bit rate

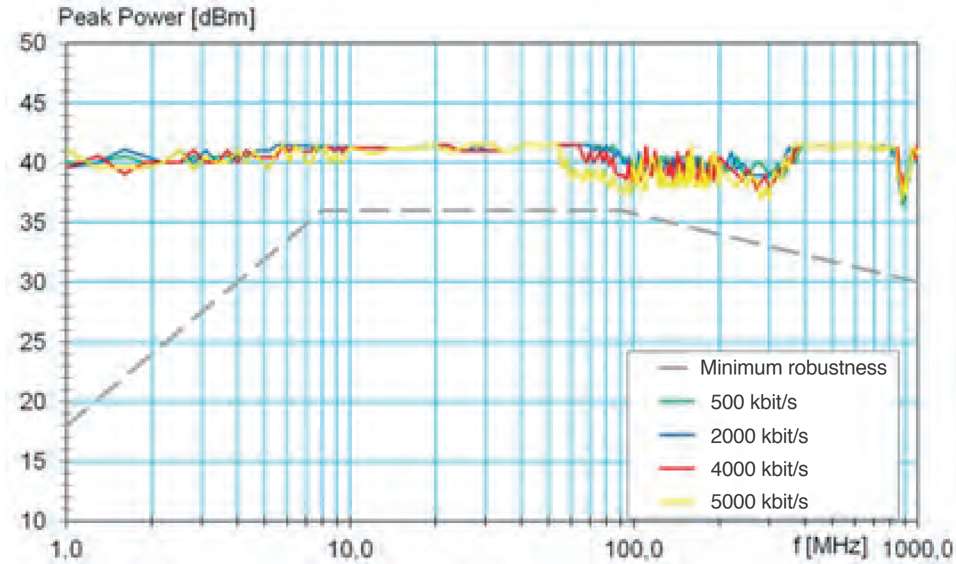


Figure 4: RF immunity of CAN FD communication

ment method (conducted or radiated measurement). Figure 3 shows a conducted emission measurement made with TJA1044GT (a member of NXP's Mantis series) without common mode choke at 500 kbit/s, 2000 kbit/s, 4000 kbit/s, and 5000 kbit/s.

The classic CAN protocol, same as the CAN FD protocol, is fairly robust against jitter on the received signal edges. Using state of the art transceivers shows that an increased bit-rate has a small impact on RF immunity. Figure 4 shows the level of RF injection (DPI) that is needed to disturb CAN communication on an EMC test board in dependence of the fast phase bit rate. Arbitration bit-rate was tuned to 500 kbit/s in each case. The maximum forward power applied during the tests was 40 dBm (10 W). Mantis TJA1044GT transceivers without common mode chokes have been used to achieve the results shown in Figure 4.

CAN FD and partial networking

Partial networking is standardized in ISO11898-6. At the time when this part of ISO11898 was developed, CAN FD was still in its infancy and thus not considered. In order not to disturb or delay the development of ISO11898-6, NXP published a proposal how to solve the challenge of selective wake-up in CAN FD systems. This publication was made as a so-called “defensive publication”, so that any attempt to patent such a compatibility mode is blocked. The technical idea behind this publication is simple: CAN FD frames are recognized by the recessive FDF bit and are just ignored. This disengages the CAN decoder in the physical layer chips from the need to decode CAN FD frames. Wake-up frames still have to be classic CAN frames.

The FDF bit is sent before the frame format and bit-rates are switched. Thus receiving a CAN FD frame does not cause a classic CAN decoder embedded in a partial networking transceiver to detect decoding errors prior to the FDF bit. After receiving a recessive FDF bit (or r0 according to the 'old' definition), the classic CAN decoder in the physical layer stops decoding and waits for idle on the bus. This new behavior will be added to ISO11898-2 in the course of the currently ongoing review (and merge) of parts 2, 5 and 6. With the TJA1145FD and the System Basis Chip UJA1168FD first products are available that feature “CAN FD Passive” partial networking.

Bear in mind that "passive" here is not the opposite of "active", since being ▷

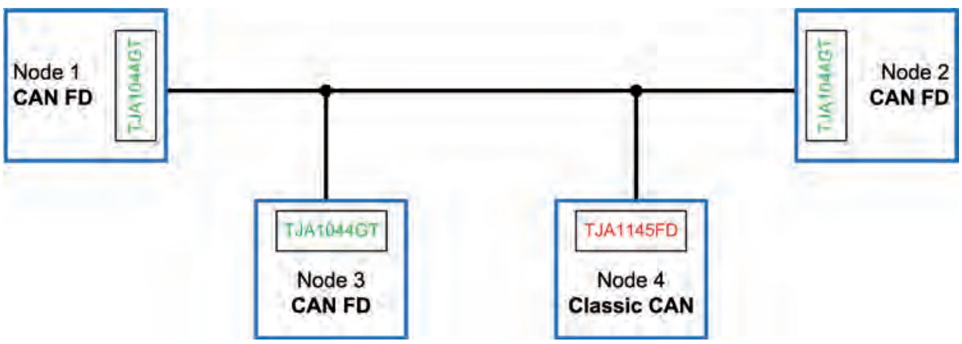


Figure 5: Exemplary system set-up for mixing CAN FD and classic CAN nodes in a network

Necessary considerations when designing a CAN FD network

Propagation delay symmetry:
This parameter is important only for CAN FD systems that exceed 1 Mbit/s. Watch out for updated datasheets that guarantee the required timings for the targeted bit-rate in the fast phase.

Backward compatibility to classic CAN:
This is solved by transceivers featuring partial networking which is CAN FD tolerant.

EMC:
Emission increases with the bandwidth by physical law, while it is different for immunity. Only transceivers that have "best in class" EMC performance should be used for systems exceeding 500 kbit/s in the fast phase.

Topology:
Slower arbitration allows for more cable length, not necessarily resulting in less bandwidth.

Transceivers:
For up to 1 Mbit/s in the fast phase all existing transceivers can be used, many of them even at bit-rates above 1 Mbit/s. Look out for updated datasheets that guarantee operation at higher bit-rates, if needed.

"CAN FD Active" means being able to receive and transmit with bit-rates above 1 Mbit/s in the fast phase. Unfortunately, this odd wording has become common over the last months. Attributes like "silent" and "tolerant" are also used in conjunction with "CAN FD". Unfortunately, the usage of all these attributes is currently not consistent.

Compatibility of CAN and CAN FD

Due to the fact that classic CAN controllers are unable to tolerate CAN FD frames, “CAN FD Passive” partial networking is currently the only way to allow a mix of classic CAN and CAN FD controllers connected on the same network. As long as one or more classic CAN controllers are active, the CAN FD controllers have to resign from CAN FD frames and send only classic CAN frames. After commanding the physical layers of all classic CAN nodes into a CAN FD passive partial networking sleep or standby mode, CAN FD frames can be transmitted over the bus lines. This is advantageous in special operating situations like SW-download to dedicated modules, etc. Such CAN FD communication phases end by a wake-up-frame (WUF) in the classic CAN format.

CAN FD and FT-CAN physical layer

Changing from classic CAN to CAN FD can also be attractive for low bit-rates, for example when using the increased number of up to 64 bytes of data per frame. As the physical layer does not care about the protocol, CAN FD can also be used in combination with the FT-CAN (fault tolerant) physical layer.

These are, of course, not the only possible application scenarios where CAN FD (with or without being mixed with classic CAN by means of partial networking) will bring advantages. The first implementations of CAN FD networks will hopefully encourage creativity for more. ◀

Subscription

Please use the following classifications for filling in your subscription form:

A Position in company
0. Other position
1. Director
2. Technical manager
3. Marketing manager
4. Sales manager
5. System designer
6. Device designer
7. Purchasing manager

B. Company's CAN business
0. Other business
1. Semiconductor manufacturer
2. Device manufacturer
3. Software house
4. System integration
5. Service provider
6. End-user
7. Research

C. CAN application interest
1. Passenger cars
2. Heavy-duty vehicles
3. Rail vehicles
4. Maritime vehicles
5. Aircraft/Aerospace vehicles
6. Power generation systems
7. Factory automation systems
8. Process automation systems
9. Industrial machine control systems
10. Construction machine control systems
11. Embedded control systems
12. Building automation
13. Door control systems
14. Lift control systems
15. Medical devices and systems
16. Science and research systems

D. Company size
1. 1 - 9 employees
2. 10 - 49 employees
3. 50 - 99 employees
4. 100 - 499 employees
5. 500 - 999 employees
6. 1 000 - 4 999 employees
7. 5 000 - 9 999 employees
8. 10 000 - 99 999 employees
9. more than 100 000 employees

CAN Newsletter

I hereby subscribe to the free-of-charge CAN Newsletter for the next four editions (published in March, June, September, and December of every year).

☐ Print version ☐ PDF version

Company

Name

Address

City, ZIP

Phone (Country code - Area code - Number)

Fax (Country code - Area code - Number)

E-mail

URL

Position (see A) Application (see C)

Business (see B) Size (see D)

☐ I like to receive CiA's Weekly Telegraph

Please send your subscription form to CAN in Automation (CiA) GmbH, Kontumazgarten 3, DE-90429 Nuremberg, Germany, or fax it to +49-911-928819-79 or e-mail it to headquarters@can-cia.org. You may also subscribe online at www.can-cia.org.

MICHAELIDES
electronics

Record CAN-messages and GPS-data using only one device: CANlogger® GPS provides easy handling and unlimited use.

CANlogger® series

CANlogger® 5001

- Record CAN messages on SD cards with up to 32 GB
- Record CAN messages without using a PC
- Especially suitable for long-term recordings
- Power management

CANlogger® 5002

- Record CAN messages on SD cards with up to 32 GB
- Record CAN messages from 2 CAN-Interfaces without using a PC
- Especially suitable for long-term recordings
- Power management
- Intelligent interface management: 1 each: analog input, digital input and digital output

CANlogger® 5102 (GPS)

- Record CAN messages on SD cards with up to 32 GB
- Record CAN messages from 2 CAN-Interfaces without using a PC
- Especially suitable for long-term recordings
- Power management
- Intelligent interface management: 1 each: analog input, digital input and digital output
- Record position data

DE RM MICHAELIDES SOFTWARE & ELEKTRONIK GMBH
Headquarters | Donaustr. 14 | 36043 Fulda, Germany
Phone +49 661 9490-0 | info@rmcan.com

US RM MICHAELIDES SOFTWARE & ELECTRONIC CORP.
US Subsidiary | 711 E. Monument Ave., Suite 310 | Dayton, Ohio 45402-1490, USA
Toll-Free +1 877 RMCAN-US | Phone +1 937 558-2211 | info@rmcan.com

Find more about our services and products at www.rmcan.com

Hardware
& Software for CAN / LIN Bus Applications

■ PCAN-Diag 2

Handheld device for CAN bus diagnostics

- Clear CAN traffic representation in lists with configurable symbolic message representation
- Transmission of individual CAN frames or CAN frame lists
- Recording and playback of CAN traffic
- Optional automatic bit rate detection
- Bus load and termination measurement
- Internal memory with USB connection for saving projects, screenshots, traces, and CSV files

Oscilloscope functions

- Oscilloscope with two independent measurement channels, each with a maximum sample rate of 20 MHz
- Display of the CAN-High, CAN-Low, and difference signal
- Configurable trigger events: frame start, frame end, CAN errors, or individual CAN frames based on their CAN ID

■ PCAN-Explorer 5

Universal CAN bus monitoring software

- Simultaneous connections to multiple networks (connected via the same hardware type)
- Easy message transmission
- Clear display of the CAN traffic with various information
- Configurable symbolic message representation
- Data logging with tracers and the 4-channel Line Writer
- Multiple flexible filters
- VBScript interface for macros
- Functionality upgrades with add-ins, e.g. an instruments panel or J1939 support
- Supports Windows 8, 7, Vista (32/64-Bit)

www.peak-system.com

Take a look at our website for the international sales partners. Scan the QR code on the left to open that page.

PEAK-System Technik GmbH

Otto-Roehm-Str. 69, 64293 Darmstadt, Germany
Phone: +49 6151 8173-20 - Fax: +49 6151 8173-29
E-mail: info@peak-system.com

PEAK System