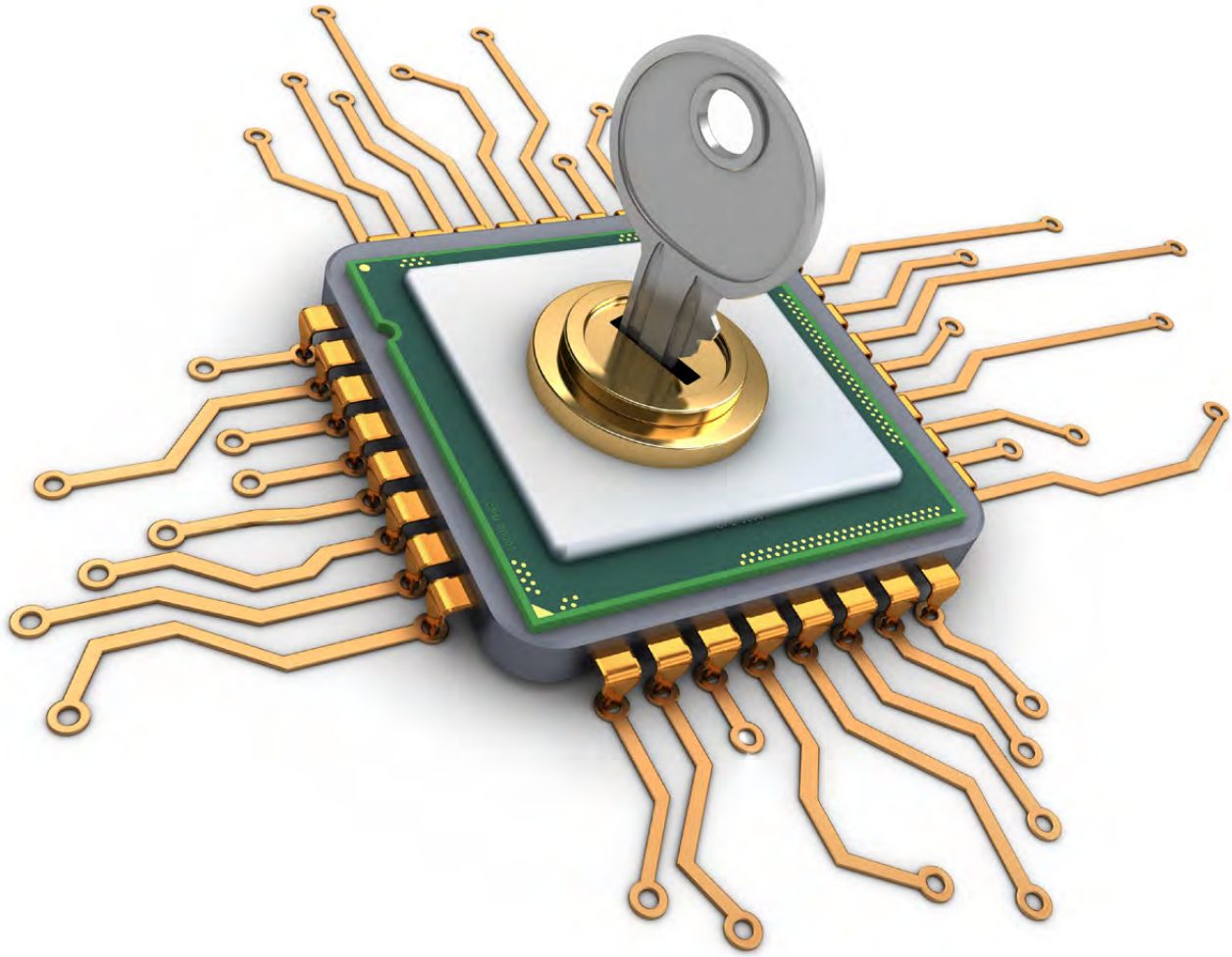
June 2017

CAN Newsletter

Hardware + Software + Tools + Engineering



Transceiver with cyber security functions

Scalable CAN security

Plug-and-secure communication for CAN

A house with unlocked doors is not secure!

Cyber security

www.can-newsletter.org

Mobile CAN FD Diagnosis

NEW

PCAN-Diag FD

Mobile Diagnostic Device for CAN and CAN FD Busses

The new PCAN-Diag FD is a handheld diagnostic device for CAN 2.0 and CAN FD busses that allows the examination on the physical and on the protocol layer. A funded analysis is done by the scope function and further measuring functions for voltage and termination. The examination of the CAN communication can be done by the display of CAN and CAN FD messages, a bus load measurement, or the recording and replay function for the CAN traffic.

Specifications

- High-speed CAN connection (ISO 11898-2)
  - Complies with the CAN specifications 2.0 A/B and FD (switchable support for ISO and Non-ISO)
  - CAN FD bit rates for the data field up to **12 Mbit/s**
  - CAN bit rates from 25 kbit/s up to 1 Mbit/s
  - CAN bus connection via D-Sub, 9-pin (in accordance with CiA® 303-1) with switchable CAN termination
- Display with 800 x 480 pixel resolution
- Device operation via a push dial and 4 buttons
- Memory card for saving projects, traces, and screenshots
- Power supply via the internal rechargeable batteries or the provided supply unit

Overview of functions

- Clear display of the CAN traffic with various information
- Configurable symbolic message representation
- Transmitting individual messages or CAN frame sequences
- Recording of incoming CAN messages
- Playback of trace files with optional loop function
- Automatic bit rate detection based on a fixed value list
- Switchable listen-only mode and silent startup function
- Measurement of the CAN bus load and termination
- Voltage measurement at the CAN connector for pin 6 and 9

Oscilloscope functions

- Oscilloscope with two independent measurement channels, each with a maximum sample rate of 100 MHz
- Display of the CAN-High and the CAN-Low signals as well as the difference of both signals
- Trigger configuration to various properties of CAN messages like frame start, CAN errors, or CAN ID
- Decoding of CAN frames from the recorded signal trace
- Display of various properties and of measuring data of the decoded CAN frame

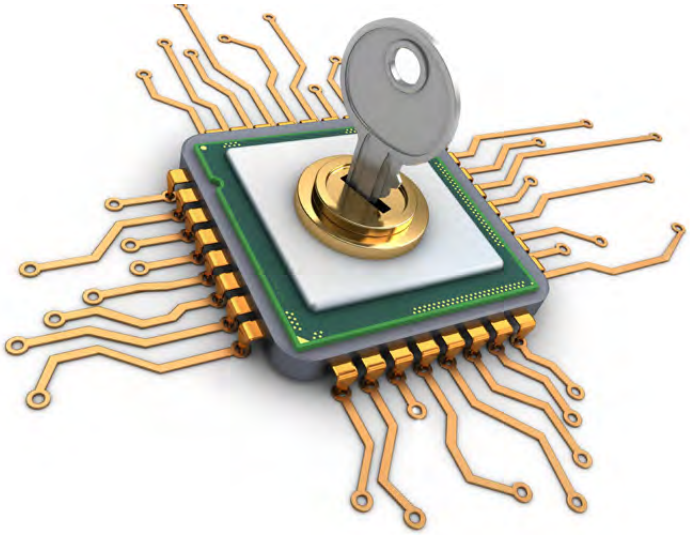
www.peak-system.com

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



Cyber security

Imprint

Publisher
CAN in Automation GmbH
Kontumazgarten 3
DE-90429 Nuremberg

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

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

CEO
Holger Zeltwanger
Reiner Zitzmann

AG Nürnberg 24338

Downloads March issue:
(retrieved May 11, 2017)
4121 full magazine

Editors
pr@can-cia.org

Cindy Weissmueller
Holger Zeltwanger
(responsible according
to the press law)

Layout
Nickel Plankermann

Media consultants
Gisela Scheib
(responsible according
to the press law)
Gertraud Weber

Distribution manager
Holger Zeltwanger

Engineering

Hardware

Market research

Software

Security is a big issue

If you like to secure your house, you put a lock on your doors and you protect also the windows. This is obvious. When we protect our car's electronic, we should be as serious as we are with our homes. But we are not. Permanently, we introduce additional "doors" and "windows". Some of them we are locking properly, but others we even do not equip with locks. Of course, in the past this was not necessary, because the in-vehicle networks were not accessible from outside. Nowadays, they are. Additionally, we introduce unintended bypasses, as for example the proposed JTAG interface option for CAN.
We should not look to single doors and windows; we should secure cars on the system level considering all interfaces. CAN is often blamed to be unsecure. But you cannot blame a door to be not locked, if you have not installed a lock. In this issue of our CAN Newsletter, you will find some ideas, how to equip the CAN-doors with locks. Perhaps one is not sufficient.

Holger Zeltwanger

Table of contents

Transceiver with cyber security functions

NXP, the market-leading CAN transceiver manufacturer, has introduced some ideas to secure the CAN lower layers. This can be implemented in smart transceivers.

The modern connected car with various internal and external communication interfaces, up to 150 electronic control units (ECUs) and 100 million lines of code, is a cyber-physical system rather than a simple mechanical system. One challenge of seamless connectivity to the Internet and end-user devices is the exposure of the vehicle to malicious exploitation of vulnerabilities, such as buffer overflow exploits, malware, and Trojans. The connected car's potential for attack (its "attack surface") is increasing as the amount of connectivity, electronics, and software continues to increase.

Figure 1: Transceiver with cyber security functions (Photo: NXP)

A common method to mitigate these risks is Defense-in-Depth (DiD). DiD is a concept in which multiple layers of security countermeasures are placed through a system to provide redundancy in the event a single security countermeasure fails or a vulnerability is successfully exploited. This is important as the attacker will need to circumvent multiple countermeasures to launch a successful attack.

The responsibility to define what level of security is required lies with the vehicle manufacturer. Current state of the art solutions are cryptographic-based with secure key exchange, authentication, and possibly encryption. Cryptographic checks of message authenticity are adding message latency and requiring considerable computing power. Thus, the disruption of applying these kinds of solutions can be prohibitive or lead to only partial implementation for protecting solely a low percentage of the CAN messages in a network. NXP therefore proposes an additional layer in the DiD concept, either complementing state of the art security solutions, or as a standalone solution for less critical, low cost ECUs, providing a basic-level of protection and hack containment.

Proposed is a distributed intrusion detection methodology, based on CAN network specific parameters, like identifiers of the CAN messages and the contribution to the overall network busload of an ECU. This method helps contain network attacks like spoofing, remote frame tampering and denial of service (flooding).

The method described is implemented solely in a smart CAN transceiver, operating fully independent and isolated from the micro-controller (MCU) – providing an inherent level of security, without neither impacting the message latency nor increasing the processor load. It can be introduced into a network in a stepwise approach, without impacting other ECUs. Such smart transceivers can be provided as drop-in replacements with today's standard CAN transceivers avoiding further hardware and software changes on the ECU and do not affect the operation of other ECUs. This makes the proposed approach a fast, low-effort and highly cost-effective way to introduce a basic level of security or fortify state of the art security solutions with a last layer of defense.

Spoofing, tampering, and flooding

Spoofing a CAN-ID means that a compromised ECU attempts to use an ID that it is not intended to be sent by this ECU. This can be useful to pretend to be another ECU. This technique has been used in practical attacks on modern cars, see Figure 2. Spoofing in the body and comfort domain can be become safety relevant, as sudden unexpected actions can distract the driver tremendously, e.g. set radio volume to maximum, or turn lights on/off.

For the tampering attack, the attacker aims to adjust a message, which another ECU is currently sending on the bus. The attacker must also adjust the cyclic redundancy ▷

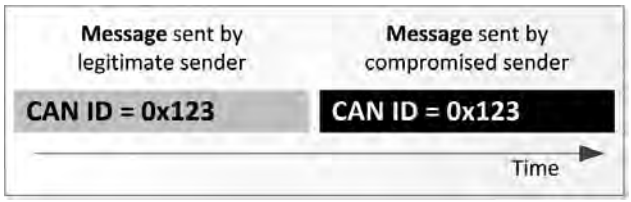


Figure 2: Spoofing attack (Photo: NXP)

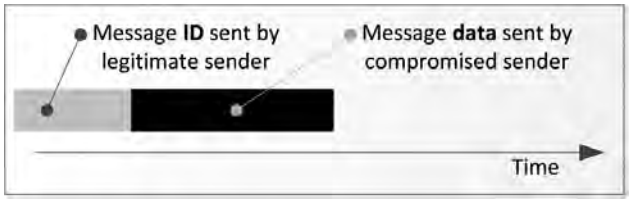


Figure 3: Tampering attack (Photo: NXP)

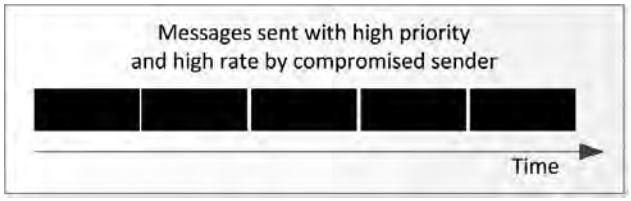


Figure 4: Flooding attack (Photo: NXP)

check (CRC) to match the tampered data. Before a successful tamper attack can be accomplished, the legitimate sender must be forced into the Error-passive state, or else it will publish an active error on the bus when the attacker causes a bit flip. The attacker can put the legitimate sender in Error-passive state by intentionally publishing errors on the bus for several times. The tampering attack is useful since it gives the attacker the power to tamper with the messages that are being sent on the bus, which may be of critical operation for the car. This kind of attack has been presented at several conferences, see Figure 3. The effects in the network are like caused by spoofing.

Flooding the bus by continuously pumping the bus full of messages is a way to deny service, see Figure 4. This makes the bus unusable for all other ECU, which forces the entire vehicle into an emergency operating mode.

Countermeasures

The methodology proposed by NXP can be implemented in smart CAN transceivers. All the countermeasures are based on parameters that the transceiver can perceive and are executed independently from the host, which might be compromised.

The first countermeasure, filtering messages based on CAN-IDs in the transmit path, is a way for the transceiver to protect the bus from a compromised ECU. If the ECU tries to send a message with an ID that is originally not assigned to it, the smart CAN transceiver can refuse to transmit this message on the bus by invalidating the message and deny subsequent transmissions. CAN ID-based filtering can be done using a white list of IDs that is user-configurable. For example, the IDs for Unified Diagnostic Services (UDS) as specified in ISO 14229 for off-board testers may be excluded from ▷

25 years CAN Newsletter

In June 1992, the first issue of the CAN Newsletter was released. In those days, it was a copied newsletter published in German language.

The March issue 1994 of the CAN Newsletter was already published in English language. Step-by-step the newsletter migrated to a magazine. In early days, the number of subscribers increased from 500 to about 3500. Still to today, it is a unique source of exclusive information on CAN technology.

The very first issue was published in June 1992 (left), the first English issue was released in March 1994 (right), both in black-and-white and just stapled

In 2012, the printed CAN Newsletter was accompanied by the CAN Newsletter Online. This online magazine provides the more product-oriented information. The CAN Newsletter magazine is not longer hard-copied, it is now published as PDF file. There are still released four issues per year with more technical in-depth articles.

In the last year, about 7600 visitors were looking into the online magazine per month. On average, they read about two pages per visit. The number of downloads of the four CAN Newsletter magazine issues is hard to count. Sometimes our website is attacked meaning someone downloads frequently PDF files. On average, there are more than 4000 regular readers of the magazine. In addition, there is a significant number of downloads of single articles. *hz*

the whitelist. This would prevent a compromised ECU from starting a diagnostic session with another ECU in the vehicle to, for example, manipulate calibration values.

The second countermeasure against spoofing is the monitoring and invalidating messages on the network based on the CAN-ID. This method enables every ECU to protect its own IDs in case a rouge ECU is not prevented from sending this ID; e.g. in case of an aftermarket device that is not under control of the car OEM and thus does not have a smart CAN transceiver with a configured transmission whitelist. When any ECU sends a message on the network, the smart CAN transceiver of the legitimate ECU can actively invalidate that message by writing an active error frame to the bus. It can do this based of the same white list as the filtering in the transmit path. The compromised sender will repeat the spoofed message 16 times before Suspend-transmission behavior kicks in, limiting the bus load contribution, and finally another 16 repetitions will occur before the attacking ECU enters Bus-off state.

Preventing spoofing makes transferring a stolen cryptographic key to a rogue ECU useless, as the ECU cannot send the CAN IDs of the messages that it could authenticate with the stolen key!

Invalidating messages on the CAN network can also be used to prevent tampering. The smart CAN transceiver can check whether there was a valid message on the network, for which the local node has won arbitration, but stopped transmission (due to receiving a dominant bit while sending recessive). This is a clear sign that a compromised ECU has stepped into the transmission.

Limiting the number of transmitted messages per ECU of time can prevent flooding the network, when implemented at the sender side. In certain applications, a burst of messages on the CAN network is desirable, but this should only last for a certain amount of time. To prevent flooding, a leaky bucket mechanism can be used. In order, not to hamper diagnostic services, e.g. for uploading data, the contribution of messages with low priority IDs is neglected when filling the bucket. Flooding protection increases the availability of the network, also in case of babbling idiots.

Smart CAN transceivers with cyber security features are available

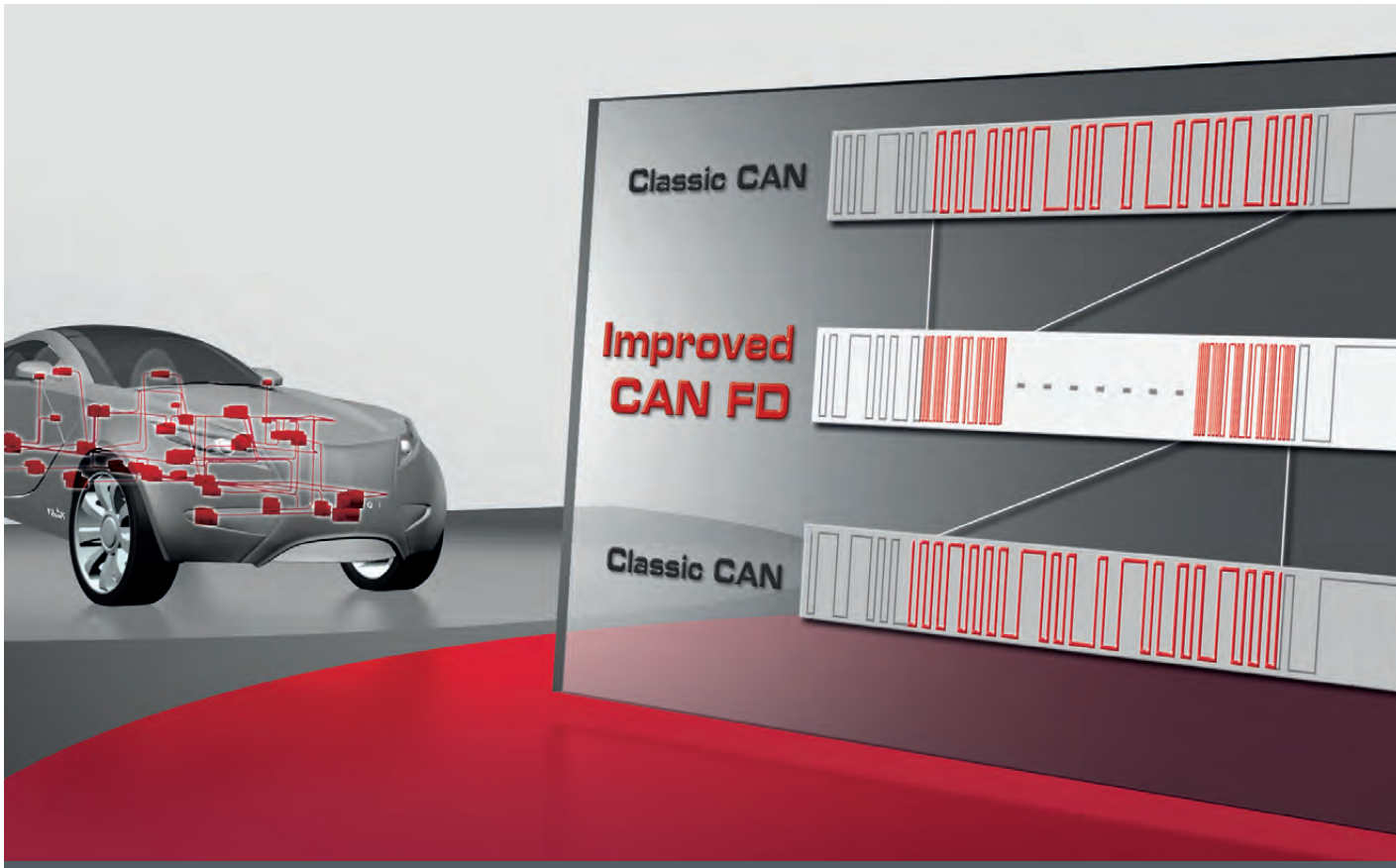
The proposed methodology is deployed on smart CAN transceivers, isolated from the host MCU and intended to be configured one-time at the Tier-1 production site and then locked, preventing future reconfiguration. An additional advantage of implementing in a CAN transceiver is it exploits the pervasiveness of the CAN transceivers in the in-vehicle network, enabling a fast and cost-effective security upgrade of existing ECUs without touching the MCU and/or software.

NXP has developed a demonstrator to prove the concept. It is based on demo silicon in an SO8 package with standard transceiver pin-out. It has been showed, for example, at the iCC 2017, see Figure 1. ◀

Authors

Bernd Elend
Tony Adamson
NXP Semiconductors
bernd.elend@nxp.com
tony.adamson@nxp.com
www.nxp.com

VECTOR



First class solutions
for your CAN and CAN FD based projects

Your complete and universal tool chain

Increase the efficiency of your projects with the use of the complete tool chain from Vector:

- Tools for testing, flashing and calibrating ECUs
- Flexible bus network interfaces
- High performance Scope for bit accurate signal analysis
- Easy to configure AUTOSAR basic software
- Worldwide engineering services and trainings

Information and downloads: **www.can-solutions.com**

More CAN power: benefit from over 25 years of networking experience.

Vector Informatik GmbH | Germany · USA · Brazil · France · UK · Italy · Austria · Sweden · Japan · Korea · India · China | www.vector.com/contact

Scalable CAN security

The CANcrypt framework is suitable for CANopen and other CAN-based application layers. It supports security methods for authentication and encryption/decryption.

The CANcrypt framework is described in more details in the book "Implementing scalable CAN security with CANcrypt". It adds different levels of security features to CAN. The CANcrypt system is higher-layer protocol independent. A manager/configurator is only required for the generation and exchange of keys, but not during regular operation. For key generation, CANcrypt uses a CAN feature that allows two devices to exchange a bit not visible to other CAN devices. This allows generating pairing keys that only the two participants know.

Per default, CANcrypt uses a dynamic 64-bit key to cover the longest possible secure data block, 8 byte. From this key, a pseudo one-time pad is generated and changes frequently. How often, new random bits are introduced to modify the shared key is configurable. 128-bit keys for AES-128 are also supported.

CANcrypt provides a security infrastructure for CAN, in which developers can still select or customize specific security functions. It can be integrated into existing code at the lower driver level, making it independent from higher-layer protocols or application layers above.

Limits of CAN specific security

When looking at security for existing CAN based communication then some cases can be excluded from further analysis. These are the cases that either we cannot protect a system from or cases for which there are already a number of solutions available.

If an intruder has access to a CAN system at a level where they can inject any CAN message at any rate, it allows them to run a denial of service style attack by flooding the CAN networj with high priority messages. It then becomes unusable for other participants which has a similar effect as physically cutting the CAN signal lines. In other words, once the door is open to an intruder, a complete shutdown of the system cannot be prevented. However, typically an intruder would not want to break a system but instead extract or manipulate status or control information. Safeguarding the integrity of a system against this type of attack therefore has a higher priority.

For larger blocks of data there are common end-to-end security and encryption standards used on the Internet such as SSL. They can be applied to CAN communications, too, but only in combination with a peer-to-peer transport protocol on top of CAN such as CANopen Segmented-SDO transfer, in which larger blocks of data are split into small segments that fit into single CAN messages.

One challenge securing "generic" CAN communications is to develop a security mechanism that can be applied to a single message, which includes for example sensor data of just a few bytes, or even single-bit commands that each control an individual switch like unlocking a door, and that also includes those messages that make use of the broadcast feature of CAN by having multiple receivers (one-to-many). Common Internet security protocols are not suitable for these scenarios.

Typical CAN attack vectors and security requirements

An attacker who has gained CAN access to a system was either able to physically install some sniffer device or achieved access to a CAN-connected device remotely. Either way, the attacker should be assumed to be able to receive and transmit CAN messages on the bus.

In many CAN systems, authentication is the only security requirement, addressing questions such as: How can we verify that a received message was really transmitted by the authorized sender and was not injected by some intruder? How can we detect if an intruder disabled an existing CAN device and tried to replace it by mimicking its communication behavior?

Often in CAN systems, encryption is less important. In an industrial or automotive system it is generally more important that a control command or sensor data is authenticated because the data itself is not regarded a "secret". A

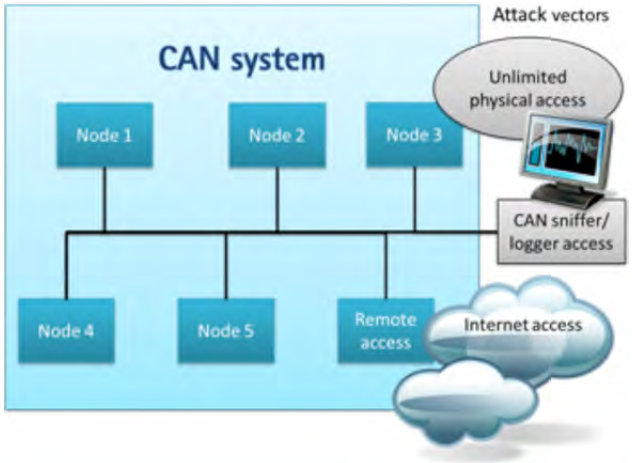


Figure 1: An attacker's access options (Photo: Esacademy)

use case for encryption might be configuration data, which however tends to be bigger than the average CAN message, allowing common Internet encryption methods to be used, as previously pointed out.

Manipulation detection

Already today, activities originating from possible attacks can sometimes be detected as a "side effect". The CiA 447 CANopen application profile for add-on car electronics for example has a built-in protection against "spoofing" of nodes. In CiA 447 with its highly dynamic nature, node IDs can be re-assigned upon every system wakeup. In order to avoid accidental duplicate node IDs, every device must monitor the network for CAN message IDs that it transmits itself. If such a message is detected, the device issues an emergency message. So if an intruder injects a message "owned" by another device an appropriate emergency message would immediately show up on the network, invalidating all its communication.

For a successful attack on such a system, an intruder would need to disable the node "owning" (transmitting) the CAN messages in question first, before introducing a device that mimics the behavior of the node.

Authentication

Any security feature will add some overhead to the communication. Such added security data includes a secure checksum and housekeeping values as well as messages to maintain the overall security mechanism. This eats up valuable bandwidth, and potentially slows down communication.

Thinking about a minimal authentication feature, all we will need is a secure heartbeat message though, as long as we add it to a system where all nodes monitor their own messages like in the CiA 447 example described above. Each individual message does not need to be protected if it is continuously monitored and manipulations are reported or cause a secure heartbeat timeout.

Devices participating in the secure communication scheme are grouped. The security foundation is a shared symmetric key from which a dynamically changing key is generated.

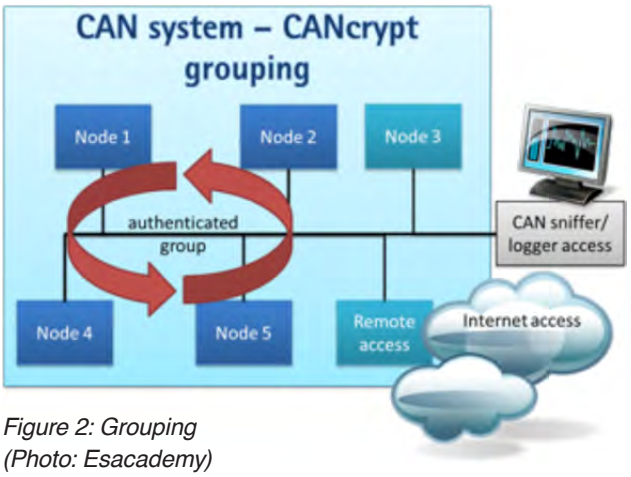


Figure 2: Grouping (Photo: Esacademy)

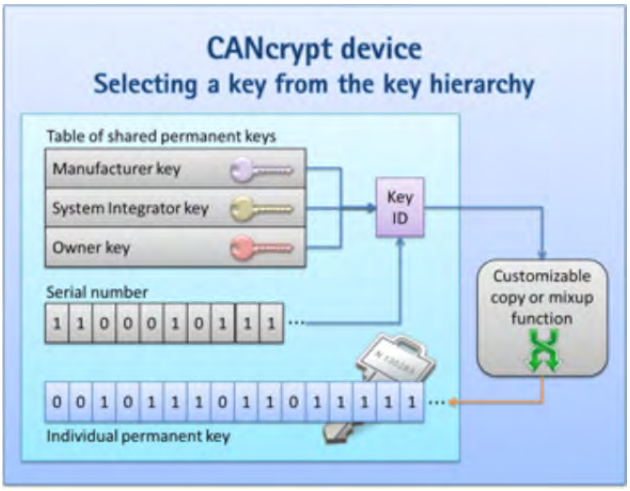


Figure 3: Key hierarchy (Photo: Esacademy)

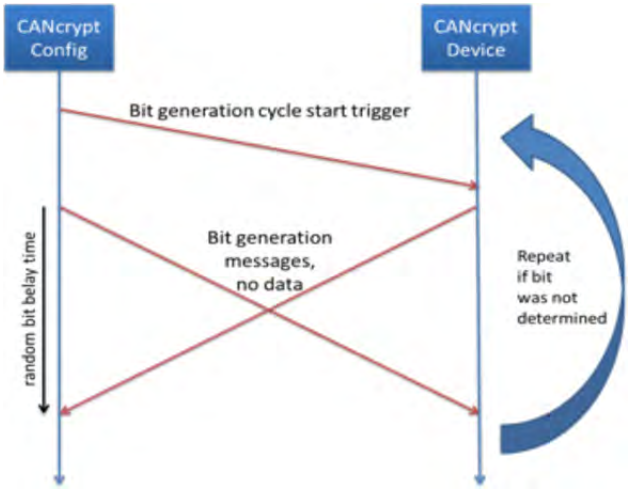


Figure 4: Bit generation cycle (Photo: Esacademy)

Each participating CAN device monitors the network for injected/duplicate messages that use a CAN ID that it itself uses for transmissions. As long as no injection or manipulation is detected, the device keeps producing a secure Heartbeat. Otherwise, it produces an emergency or alert message.

On the receiving side, secure messages are only authenticated with the reception of the following secure Heartbeat. The injection/manipulation detection alone is not enough, as we cannot guarantee that the emergency message is successfully transmitted; after all, an intruder could try to specifically block that message with collisions. Only a successfully received secure Heartbeat authenticates "all previous messages" from a device.

The disadvantage of such a system is that the secure Heartbeat cycle time directly impacts the system control cycle time and therefore needs to be faster than a typical heartbeat period such as the one used in CANopen. Depending on the authentication requirements of a system, a control unit must wait for the secure heartbeat timeout to decide if the previous messages are authenticated. This calls for a secure Heartbeat period as short as 100 ms or less.

Key management challenges

As with any security system, the management of the used keys can be a tougher challenge than applying the security methods to the communication. Assuming the use of a shared (symmetric) key for the main security features, the question arises when and how this key finds its way into the individual devices and where we keep copies of it. Another challenge is the dynamic update/modification of the shared key. If all participating devices continuously update the key, the specific security algorithms do not need to be very strong. A frequently updated key creates a one-time pad that even with a simple XOR algorithm produces a very high security level.

With a key hierarchy a device can have multiple authorization levels. The original manufacturer could use the highest priority key – only this key would allow a factory reset or to activate a possible boot-loader to re-program the flash memory in the device. The next priority level down could be the "system integrator" level, which would allow combining devices from multiple manufacturers in one system. The key for this level would allow restoring a system default configuration. And another priority level could be for the "owner" or technician that needs to be able to replace a single device within the system. This authorization level would support pairing and grouping of the components in a system.

Preferably, the lowest priority key-level used to pair/group multiple components would not be stored anywhere outside the system. Upon original pairing/grouping, a random shared (symmetric) key would be generated and exchanged between the participating devices and stored locally by each device. Figure 3 shows that optionally the device's serial number may also be included in the generation cycle for the initial permanent key, which would be used as a base for the dynamic one-time key pad used.

The method used to exchange keys described in the following does not use any "direct" CAN communication. Anyone just monitoring the exchanged CAN messages will not be able to determine the values exchanged. The method is loosely based on principals introduced by a paper from Bosch at the 15th iCC.

By monitoring CAN communications at the message (data link) level, an observer cannot determine the physical device that sent an individual message, because in CAN, any device may transmit any message. As an example, let us allow two nodes (named "configurator" and "device") to transmit messages with the CAN-IDs 0010_h and 0011_h and data length zero. The bits transmit within a "bit select time window" that starts with a trigger message and has a configurable length, for example 25 ms. Each node must randomly send one of the two messages at a random time within the time window.

At the end of the bit select time window, a trace recording of the CAN messages exchanged will show one of the following scenarios:

- Case 1: One or two messages of CAN-ID 0010_h
- Case 2: One each of CAN-ID 0010_h and 0011_h
- Case 3: One or two messages of CAN-ID 0011_h

Note that if two identical messages collide, they'll be visible just once on the network. If 0010_h and 0011_h collide, 0010_h is transmitted first followed by 0011_h. Let us have a closer look at case 2 – one each. If the messages are transmitted randomly within the bit response time window, ▷

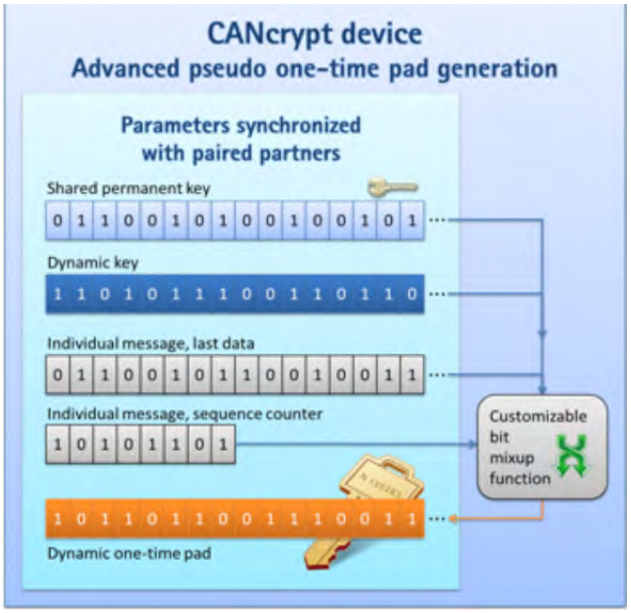


Figure 5: One-time pad generation (Photo: Esacademy)

an observer has no clue as to which device sent which message. However, the devices themselves know it! Now a simple "if" statement can determine the random bit for both participants:

```
IF I am the configurator device
 IF I transmitted 0010h and also saw a 0011h
  common bit is 0
 ELSE IF I transmitted 0011h and also saw 0010h
  common bit is 1
 ELSE
  both used same message, no bit determined
ELSE I am a device
 IF I transmitted 0010h and also saw a 0011h
  common bit is 1
 ELSE IF I transmitted 0011h and also saw 0010h
  common bit is 0
 ELSE
  both used same message, no bit determined
```

Unfortunately, we cannot use cases 1 and 3, so if those happen, both nodes need to recognize it and retry (try again in the next bit select time window). To prevent an observer from identifying individual device delays, each device should choose two good random values for each cycle.

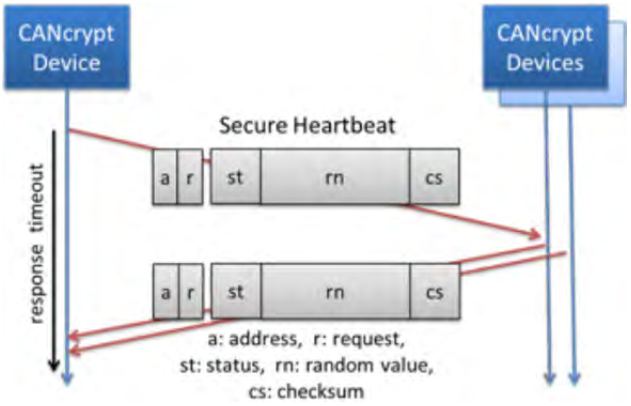


Figure 6: Secure Heartbeat contents (Photo: Esacademy)

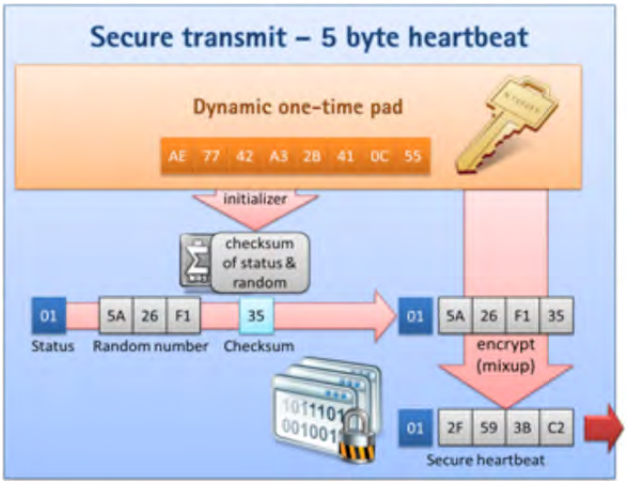


Figure 7: Secure heartbeat transmit (Photo: Esacademy)

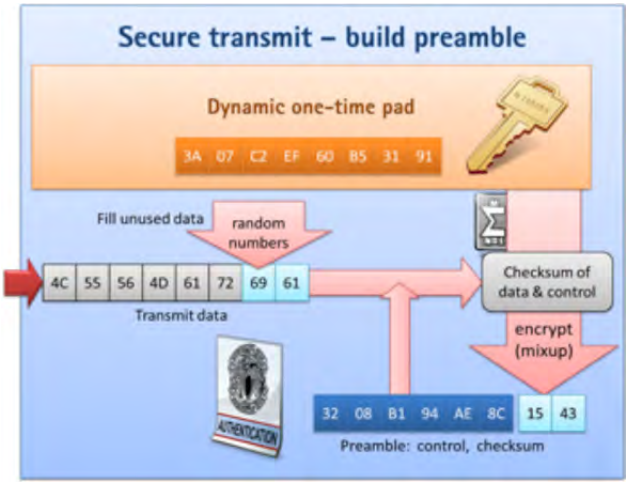


Figure 9: Secure transmit with preamble (Photo: Esacademy)

The devices should randomly pick one of the two messages (0010_h or 0011_h) and randomly select a delay from zero to two-third of the bit select time window.

There are several options to optimize this cycle as well as allowing one device to "enforce" a certain key to the other. Depending on version and timeout, this method can be used to exchange a key of 64 bits within about one second.

Dynamic one-time keypad

All devices participating in the secure communication use a locally stored symmetrical key as a basis. During initialization and detection of their communication partners, the participating devices also exchange random numbers.

The combination of all random numbers exchanged during the initial detection is used to generate the initial one-time keypad. This ensures a unique shared initial dynamic one-time keypad with every system start.

Once the devices are paired or grouped, the shared dynamic one-time keypad gets periodically updated, for example using random values from the secure heartbeat and an optional message counter.

All devices participating in the secure communication produce a secure Heartbeat. The secure Heartbeat is synchronized. In one cycle all participants transmit their own security Heartbeat once.

The main component of each secure Heartbeat is a three-byte random value with a one-byte checksum. All four bytes are encrypted based on the current shared dynamic key. All receivers decrypt the four bytes and verify if the checksum matches. If it does, the Heartbeat is considered "confirmed".

All (decrypted) random values of all participating nodes are used to update the shared one-time key pad which is then used for the next cycle. This ensures changes to the dynamic shared key with every use.

In CANopen, the secure Heartbeat fits into the manufacturer-specific fields of the Emergency (EMCY) message. This allows the implementation of the secure Heartbeat as a variation of the no error / emergency reset message already defined in CANopen.

Point-to-point communication

When using pairing instead of grouping, CANcrypt supports more advanced methods involving individual message authentication and encryption. In this case, individual messages are encrypted and authenticated with a preamble message that contains the security overhead information for the following message.

Figure 9 shows how a preamble is build. Unused data of the original message is filled with random values. The preamble contains a checksum covering both messages ▷

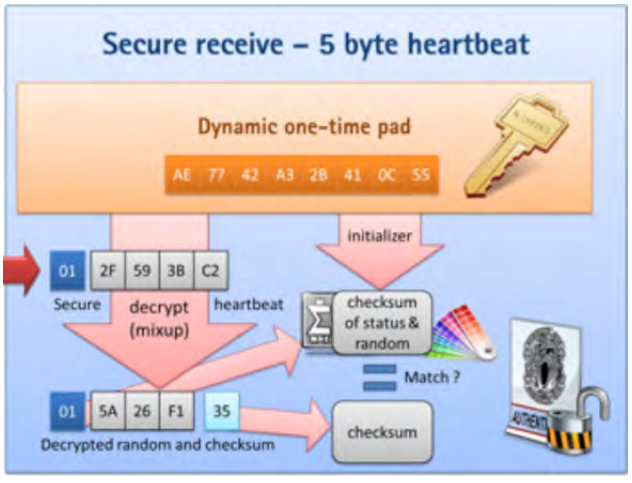


Figure 8: Secure Heartbeat receive (Photo: Esacademy)

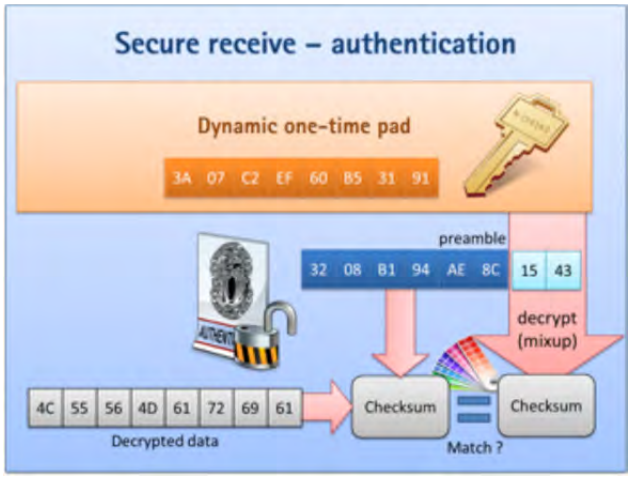


Figure 10: Secure transmit with preamble (Photo: Esacademy)

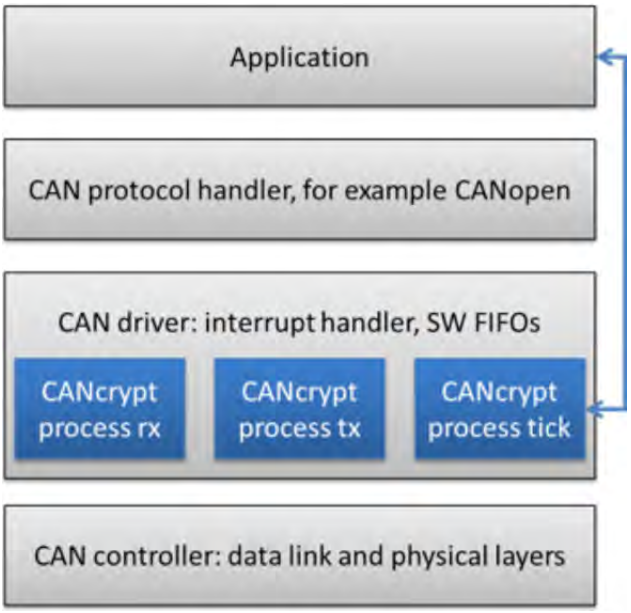


Figure 11: CANcrypt processes (Photo: Esacademy)

and control/status values. Per default, both the preamble with the checksum and the main message are encrypted. As the total data size is 128 bits, algorithms supporting a 128-bit key size may be used (for example AES-128)

Figure 10 illustrates the receiving side. Both the preamble and data message are decrypted and then analyzed. Only if the checksum matches is the message considered “authorized” and passed on to the receive FIFO.

Looking at existing systems already in use, one of the challenging questions is how security can be added with minimal changes to the software. Often, security is added to a higher communication layer. However, the higher up this is added, the more changes to existing software very close to the application level are needed.

The implementation for CANcrypt on the other hand can happen entirely at the driver level with the only requirement being that FIFO buffers are used. To all software layers above the driver, CANcrypt is largely transparent and no software changes are required on the application level. The application will simply not receive data that is not secure as the driver will only pass on messages that are authenticated.

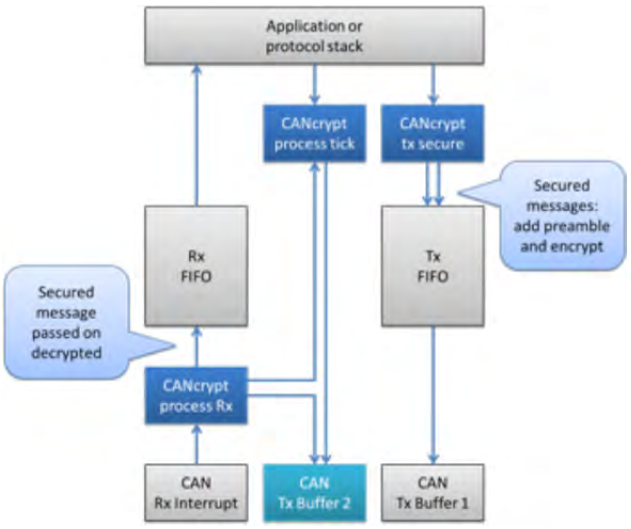
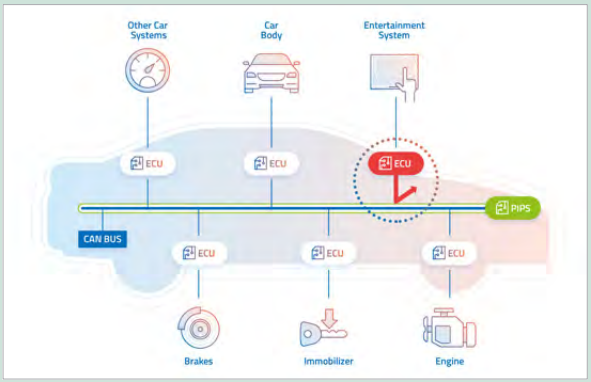


Figure 12: CANcrypt process integration (Photo: Esacademy)

Parallel Intrusion Prevention System

One of the methods most frequently used by hackers intending to take control of a vehicle is sending unauthorized commands that the car's system mistakenly takes as coming from an authorized ECU (electronic control unit). Filtering these malicious messages is key to preventing such impersonation attacks when one of the ECUs sends messages as if it was another. The by NNG Cyber Security introduced Parallel Intrusion Prevention System (PIPS) is a solution that analyzes not only the content and context of the communication on the CAN network between ECUs, but the source of it as well. This enables it to intercept malicious messages in real-time.



"We all know there aren't any vehicles that cannot be hacked, but our solution can effectively, and permanently rule out one of the methods most widely used by hackers. I could describe this as having discovered a vaccine for cars that fully protects them from a type of dangerous malicious attack. By applying our preventive technology, OEMs can make sure their cars are immune to such attacks" said Ziv Levi, CEO of Arilou, now part of NNG Cyber Security.

The PIPS technology analyzes the physical characteristics of the communication and determines its validity. As the analysis happens in real-time, it can stop the threats before they get to their destination, ensuring the security of the vehicle's ECUs. The technology is unique on the market because it can accurately track the origins of the communication, thus excluding the chance of impersonation attacks.

As opposed to available security approaches, which secure only the communication flow, the NNG's solution is integration agnostic. It can be connected anywhere on the CAN network, which ensures full network coverage from one single point of integration. The solution was revealed to partners at the CES 2017 in Las Vegas. Patent requests have been already filed. *hz*

Figure 12 shows in more detail how the CANcrypt processes are incorporated into a typical embedded system using one receive and one transmit FIFO.

The main entry points for CANcrypt are on the receive side, before a message is added to the receive FIFO (so CANcrypt can only insert the message if it is considered "secure", i.e. authenticated) and on the transmit side, also before it gets inserted into the transmit FIFO (to possibly add security overhead like a preamble).

Summary

The described security functions apply to both Classical CAN and CAN FD. The CANcrypt framework allows developers using it to select the individual security methods or algorithms used. These range from a simple grouping with authentication-only to a more secure pairing of two nodes with both authentication and encryption. CANcrypt security is based on symmetric 64-bit or 128-bit keys and data sizes of up to 128 bit. All security algorithms suitable for a 128-bit data/key length can be used, including AES-128 or comparable algorithms for the security. ◀


Authors

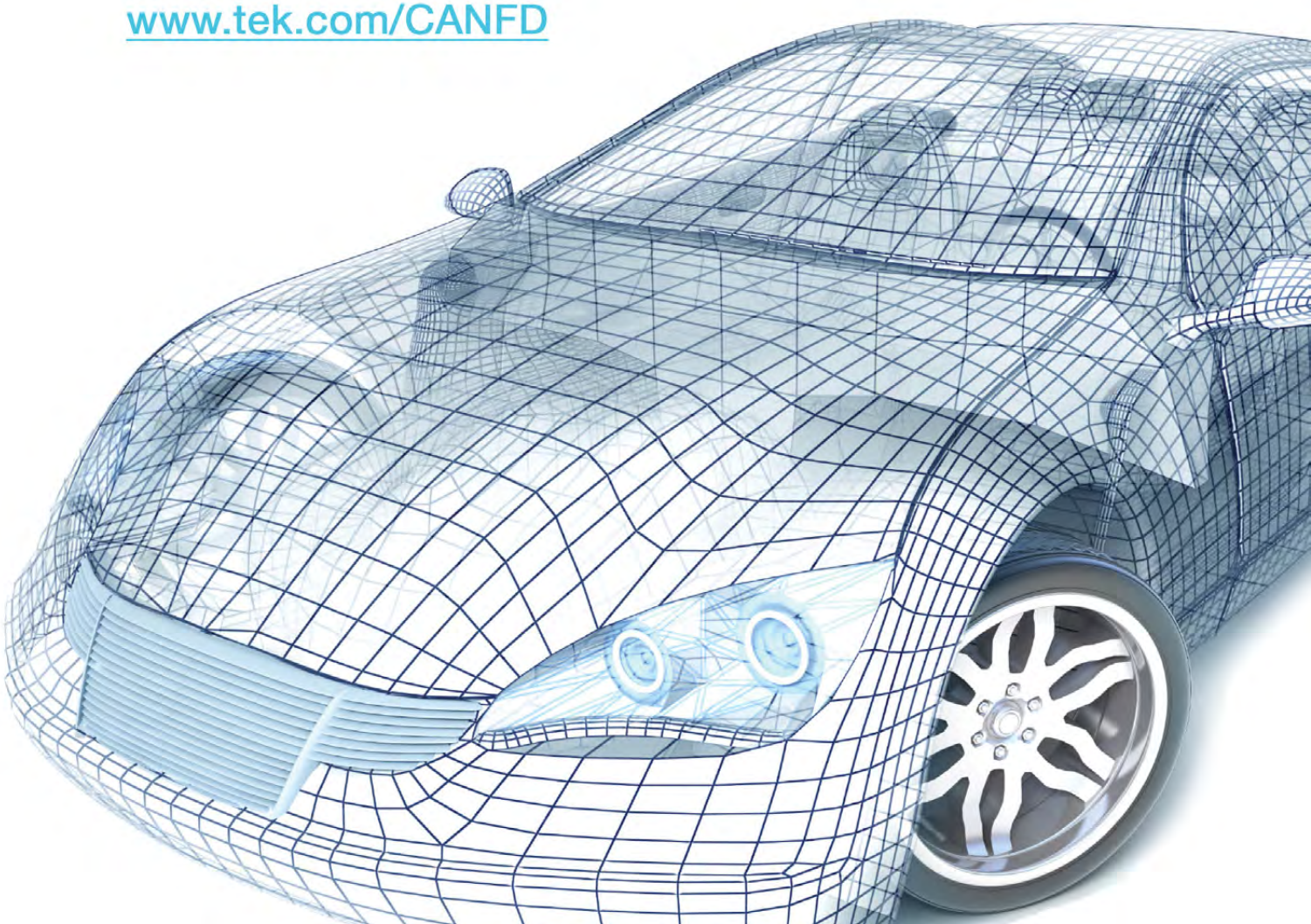
Olaf Pfeiffer
Christian Keydel
Embedded Systems Academy
opfeiffer@esacademy.de
ckeydel@esacademy.de
www.esacademy.com

Tektronix®

Drive to CAN FD faster

Capture more data with Tektronix Oscilloscopes

LEARN MORE AND
REQUEST A DEMO AT:
www.tek.com/CANFD



(Photo: Bosch)

Plug-and-Secure Communication for CAN

A novel approach from Bosch may greatly simplify the key management in CAN networks.

In a highly connected world, security plays an increasingly important role. Recently, various attacks have shown that the current trends and developments towards connected and automated driving lead to a higher vulnerability and susceptibility to corresponding security threats. Therefore, it is of utmost importance to implement suitable security measures. In this regard, a secure communication between different control devices plays a central role. However, securing CAN networks turns out to be more difficult compared to the securing of classical IT systems. This is because typically embedded devices are involved, which usually have only very limited compute, memory, and bandwidth resources and which usually are very cost-sensitive. While some appropriate schemes for encryption or authentication of CAN messages are already available, the distribution of the required cryptographic keys yet remains very challenging.

While asymmetric approaches known from the classical IT domain (e.g., the Diffie-Hellman key exchange protocol) come along with a rather high computational complexity as well as high bandwidth requirements, the distribution of cryptographic keys in a secure environment (e.g., at the end of a production line) lacks flexibility and requires a high organizational overhead. Therefore, Bosch has developed a novel approach, which makes the establishment and refreshment of symmetric cryptographic keys in CAN networks more efficient and flexible at rather low costs.

From an overall perspective, the most important and threatening attacks on cars and other vehicles include those that can be performed remotely – for example by remotely gaining access to a control unit or a CAN network. This is because these attacks often easily scale (in the sense that if an attacker manages to successfully attack one car, he may often use the same procedure to successfully attack many other cars as well) and may be hard to be traced back. When performing such attacks, the hardware integrity of a network generally remains unchanged, but the software is manipulated by a malicious (external) intervention (e.g., through a car's connectivity interface). In the worst case, remote attackers could even take over control of a large number of vehicles with a specific vulnerability and intentionally cause a crash of the complete traffic infrastructure. A simple CAN network illustrating a remote attack scenario is shown in Figure 1. Here, Alice and Bob are two legitimate (i.e., unmodified) CAN nodes while the software running on Eve has been manipulated by an attacker. For simplicity, it is assumed that all nodes are connected to the same CAN segment. Now the challenge that is addressed by the novel approach of Bosch is how Alice and Bob can agree on a symmetric cryptographic key without letting Eve gain any knowledge of it.

Key establishment by means of public discussion

The basic idea of "Plug-and-Secure Communication for CAN" is based on the simultaneous transmission of CAN frames by Alice and Bob with random bit strings in the payload field. The CAN frames sent by both nodes interfere with each other on the CAN network, so that Eve can only see the superposition of both messages. However, by just knowing the superposition, Eve is generally not able to tell which message has been sent by Alice and which one by Bob. Alice and Bob themselves see the superimposed message as well, but in contrast to Eve they additionally know the messages they have transmitted themselves. With this additional information (which is not accessible to Eve), they are able to determine which message must have been sent by the corresponding other node and thus establish a shared secret, from which a symmetric cryptographic key pair may be derived.

Figure 2 shows the detailed procedure of the core idea. First of all, Alice and Bob generate independently of each other a random bit string of a certain length. In the second step, these bit sequences are extended by inserting the corresponding inverse bit after every bit. This is necessary in order to make sure that Alice and Bob later on can really determine which bit string has been transmitted by the respective other node, as will become more obvious below. In the third step, the extended random bit strings are simultaneously transmitted by Alice and Bob in the payload field of a CAN frame, which results in the previously mentioned superposition of these two messages. This superposition can be observed by all nodes connected to the CAN network, including Alice, Bob and Eve. Due to the special properties of the CAN physical layer, a CAN network basically resembles a wired AND-function, i.e., if two (or multiple) nodes simultaneously transmit a certain bit, the effective bit on the CAN network will always be '0' (dominant bit), except for the case when all nodes transmit a '1' (recessive bit), in which case also the effective bit on the CAN network would be '1'. Clearly, this property represents also the basis for the bus arbitration in CAN. Different to the classical bus

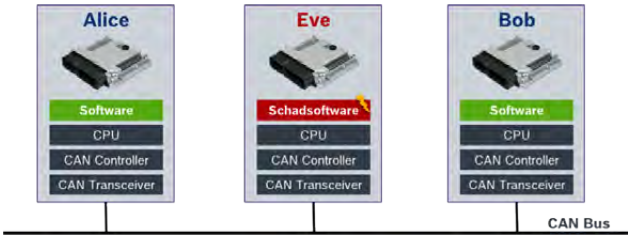


Figure 1: Considered setup for the establishment of a symmetric key pair over CAN between Alice and Bob in presence of the attacker Eve (Photo: Bosch)

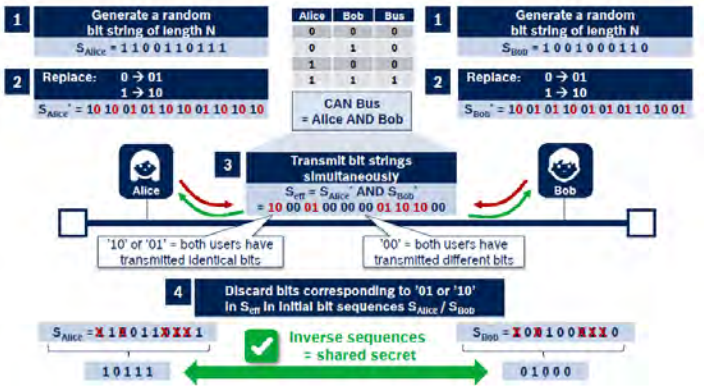


Figure 2: Basic idea for easy key agreement in CAN networks by means of the novel "Plug-and-Secure Communication for CAN" approach (Photo: Bosch)

arbitration, however, "Plug-and-Secure Communication for CAN" relies on the superposition of different bits also in the payload field of a CAN frame. As can be seen from Figure 2, the effective (superimposed) bit string S_{eff} on the CAN network is consequently simply given by a logical AND function applied to the corresponding extended bit strings transmitted by Alice and Bob.

As already mentioned above, this superimposed bit sequence S_{eff} can be also received by Eve. Nevertheless, Alice and Bob have basically already achieved their goal. They just have to interpret the contained bits in a proper way. This is done by considering always tuples of two bits each, where every tuple corresponds to a bit of the original random sequence of Step 1. If a bit-tuple contains a '1' (i.e., the effective bit on the CAN network is recessive), the corresponding bit in S_{Alice} resp. S_{Bob} is discarded (Step 4), because in this case Alice and Bob must have transmitted exactly the same bit-tuple – which can be concluded by Eve as well. Hence, Alice and Bob have no advantage over Eve in this case.

The more interesting case happens when an effective bit-tuple on the CAN network corresponding to '00'. This occurs if and only if Alice and Bob transmit different bits. Hence, in such a case Eve cannot tell which tuple has been sent by which node whereas Alice and Bob know, of course, what they have sent themselves. In addition, they also can see the '00' on the CAN network and thus they are able to conclude that the other node must have transmitted the corresponding inverse bit-tuple. This is then exactly the secret that Alice and Bob share in this case. In order to extract this secret, the bits in S_{Alice} and S_{Bob} corresponding to the '00' bit-tuples are not discarded (while the ones including a '1' are), such that Alice and Bob finally obtain two shortened bit sequences, which are exactly inverse to each other and unknown to Eve. Hence, Alice and Bob have established a common secret and by incorporating the described procedure into an appropriate protocol, symmetric keys of arbitrary length can be derived.

For realizing the described procedure in practical systems, some additional concepts and mechanisms have to be taken into account in order to meet the special demands and requirements of the automotive industry. Therefore, Bosch has developed an overall system concept based on the presented approach, which includes procedures for the generation of group keys between more than two devices, key verification, as well as key activation.

No disturbance of the regular CAN communication

The simultaneous transmission of CAN frames by Alice and Bob in the payload field leads to some challenges if full backwards compatibility with standard CAN should be guaranteed. Two major challenges in this respect arise from the possible introduction of so-called stuff bits as well as the calculation of the cyclic redundancy check (CRC) field contained in every CAN frame. This is due to the fact that the superposition of two (or more) valid CAN messages usually does not result in a valid superimposed CAN frame. This problem, however, can be addressed in such a way that even the superimposed CAN frames on the bus are always valid ones according to the CAN specification – thus assuring the previously mentioned backwards compatibility. For that purpose, the value of the CRC field as well as the decision whether or not to introduce a stuff bit are determined dynamically based on the effective bit sequence on the CAN bus (resulting from the superposition of the CAN frames simultaneously transmitted by Alice and Bob) rather than on the basis of the respective individually transmitted bit sequences. The proposed approach can be easily embedded into regular CAN traffic, since the CAN frames transmitted by Alice and Bob compete for the bus access via the well-known CAN bus arbitration, just like any other node.

Plug-and-Secure Communication for CAN" comes along with a variety of different advantages: First of all, the approach exhibits a very low complexity, has very low resource requirements and may be implemented at rather low costs. In fact, in order to generate symmetric cryptographic keys, Alice and Bob simply have to transmit and receive CAN messages and interpret the received bits in the right manner. In addition, very few messages are sufficient to let nodes agree on a key. For example, on average a single 64 byte CAN FD frame is sufficient to establish a 128-bit key between two nodes, thus resulting in very low bandwidth requirements. Besides, the procedure is universally applicable to all sorts of devices and also suitable for efficiently refreshing already existing keys, which generally is a recommended practice in order to ensure a high level of security. To this end, one can generate a few new secret bits between Alice and Bob every now and then and combine these with the old key in order to obtain a refreshed (and thus more secure) key.

How secure is the procedure?

It can be shown that – considering the previously described attacker model – Eve is not able to determine or manipulate the established key between Alice and Bob. Eve can merely prevent the successful key establishment between Alice and Bob by means of a "Denial-of-Service" attack, but this is a threat that generally exists in any CAN network and that generally cannot be avoided. Furthermore, it is interesting to note that in contrast to alternative key establishment procedures, such as the Diffie-Hellman key agreement protocol, the security of "Plug-and-Secure Communication for CAN" is not based on the hardness of solving certain mathematical problems, but on the difficulty to differentiate between two superimposed physical signals by just knowing their superposition. Therefore the procedure would also not be affected if, for example, with the ▷

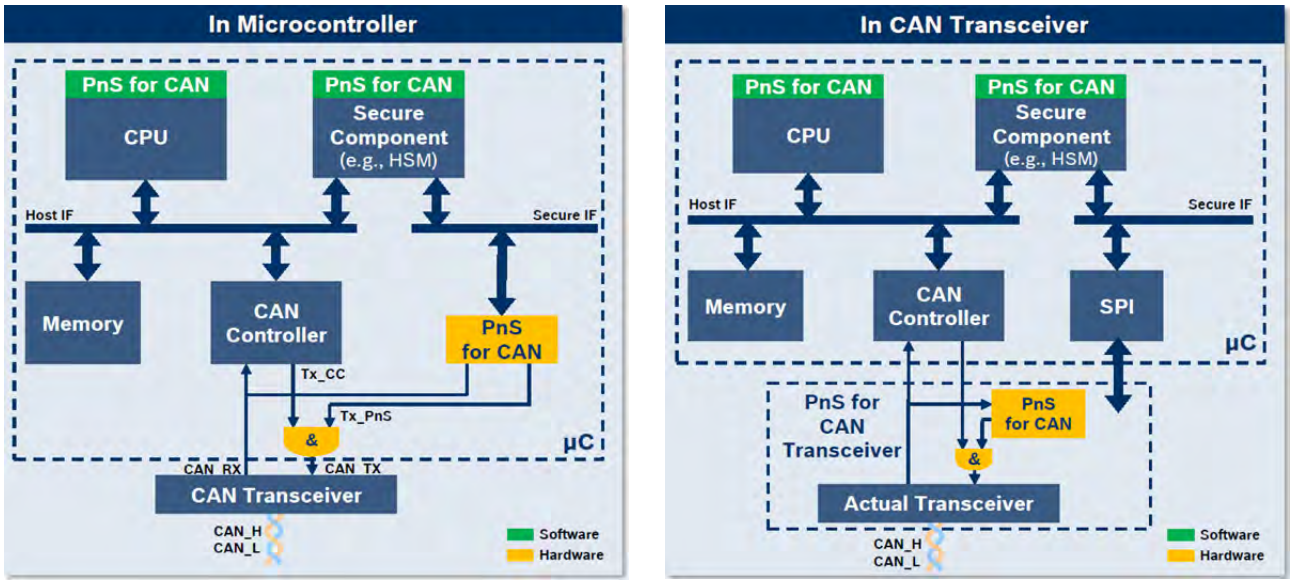


Figure 3: Two possible implementation options (Photo: Bosch)

introduction of efficient quantum computers many of the alternative schemes (including Diffie-Hellman) would suddenly become vulnerable or even insecure.

In case that an attacker has physical access to the CAN network, the situation becomes a bit more challenging. This is because in this case the attacker may have access to the detailed physical signals on the bus, which may help her to actually separate the individual CAN frames simultaneously transmitted by Alice and Bob. However, the relevance of such a scenario is rather questionable, because these attacks do not scale and with physical access to a car, an attacker could also manipulate the car with much less effort, e.g., by manipulating the brake pipe. Moreover, appropriate countermeasures against attacks with physical access to the CAN network are available and may be implemented if needed. This may involve, for example, the modulation of the voltage levels of the transmitted CAN frames in an appropriate way. Besides, it should be noted that in general also a combination of "Plug-and-Secure Communication for CAN" with other approaches, such as the Diffie-Hellman key exchange protocol, are possible, by which the best of the two worlds may be combined.

For the realization of the proposed procedure, two different implementation options are available, which are outlined in Figure 3. In particular, a dedicated hardware module may be integrated either directly into a microcontroller or into a CAN transceiver. In both cases, this hardware module is responsible for the simultaneous transmission of the CAN frames as well as for the low level signal processing in order to assure full backwards compatibility with standard CAN. In addition, a suitable software component is required for the higher-layer protocol mechanisms, which can run on a hardware security module (HSM) or the CPU. The complexity of the hardware module is very low. It requires less than 10 000 gates. The main advantage of a realization in the CAN transceiver is the easy upgradability of legacy devices compared to a direct integration into a microcontroller. In both cases a combination with any CAN controller is possible, without the necessity of any further modifications.

The approach has been successfully implemented and demonstrated by Bosch already and was presented at different fairs and other events. Figure 4 shows a basic setup of a demonstrator, which implements the scenario according to Figure 1, including the attacker Eve. The setup supports bit rates of up to 1 Mbit/s. On average, only eight Classical CAN messages with an 8-byte payload field or just a single CAN FD message with a 64-byte payload field have to be exchanged in order to generate a 128-bit key, thus reflecting the very low bandwidth requirements.

Conclusion

With "Plug-and-Secure Communication for CAN", a novel and innovative approach for the establishment and refreshment of symmetric cryptographic keys on CAN networks is available. It is characterized by its low complexity, high efficiency, as well as cost-effective implementation options and it is well-suited for the protection of CAN networks against remote attacks. Therefore, the approach has high potential to become an important building block for secure communication in CAN networks in future. ◀

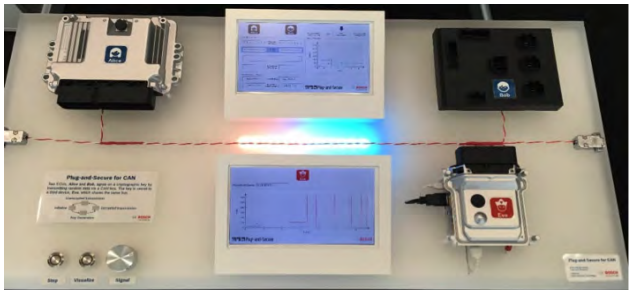


Figure 4: Existing demonstrator setup (Photo: Bosch)

Authors

Dr. Andreas Müller, Timo Lothspeich, René Guillaume, Dr. Arthur Mutter
Robert Bosch GmbH
info@de.bosch.com
www.bosch.com

FAULHABER

FAULHABER BP4

The flyweight that packs a heavyweight punch

NEXT ROUND

70 YEARS OF DRIVE SYSTEMS

NEW

WE CREATE MOTION

FAULHABER BP4 Brushless DC-Servomotors

In the battle for high performance with minimum possible weight, the BP4 series from FAULHABER has proven its champion quality. As well as having an outstanding torque-to-size ratio, the drives have integrated sensor systems as well as a wide range of speeds, and they are less than half of the weight of conventional motors with comparable performance capability.
Also available as a complete Motion Control System with integrated control electronics for complex positioning tasks in the direct automation environment.

www.faulhaber.com/bp4/en

LASER World of PHOTONICS

München 26. – 29.06.2017
Hall A2 · Stand 426

A house with unlocked doors is not secure!

In the last years, CAN has been criticized often to be not secure. In the meantime, several ideas have been discussed to add authentication and encryption to CAN-based networks.

When our ancestors lived in caves: their homes were not locked. I remember from U.S. movies made in the 50ties and 60ties, that the cars were not locked. Blaming a house not to be secure, because the doors have no locks, does not make sense. The not secure gateways are the problem.

CAN is a data link layer protocol developed before the World Wide Web was invented by Tim Berners-Lee, a British scientist at Cern in 1989. In the 80ties, the car's electronic had no external interfaces, no "doors" to the outside world. There were no gateways at all. It was like a permanent "prison", a castle without gates and backdoors.

One of the first external interfaces based on CAN was the OBDII connection. It was designed for diagnostic purposes in the repair station. In the meantime, people connect Bluetooth and other dongles (available for just a few dollars) for different private applications. This was originally not intended and is a kind of misuse. The door to the in-vehicle networks is nowadays wide open. All electronic control units (ECU) in the vehicle can be hacked. This includes also functions, which we have not on our screen as critical, such as window wipers, fog lights, etc. Switching them off when it strongly rains or when it is foggy is a real safety issue. Another thing is the airbag: A hacker should not be able to inflate it. Just faking the sensor values could be sufficient.

The ODBII interface is just one of the unlocked doors to the in-vehicle networks. Hundreds of articles have been written about cyber security for cars and autonomous driving cars. IT security experts published white papers, and offered already their solutions and services to the car-makers. The third VDI conference on IT security for vehicles scheduled for July 2017 in Berlin will provide further papers and ideas. The most important message is always, we need a holistic approach. If you lock securely all doors of your house, anything is okay; but, if new tailgates without locks are introduced, you can use the best security practice in the front doors, it doesn't help. Last year, the CAN community walled-up the door introduced by the IEC 62969-5 new work item proposal. It was intended to use one configurable CAN-ID for testing the attached micro-controller. The new work item proposal was withdrawn. We may have the same problem, when the micro-controller uses the CAN interface for JTAG access. This time not internationally standardized. It is also a not locked door by default, which the ECU or device designer needs to equip with a proper lock. It is not a good idea to launch such not secured tailgates. ARM has introduced to support JTAG access by several interfaces. Of course, the ECU designer may not support those options. But, if a not so experienced spare part supplier does it, you have an open tailgate in your car. And you don't know it.

According to most of the published cyber security concepts, it is necessary to establish a multi-level authentication, and depending on the application to encrypt the data. Security guards in software are preferred, because if a "lock" is hacked you can simply substitute the no more secure program by means of a secure software update. Exchanging hardware is more costly. But on the other hand, security functions in hardware require generally some kind of physical access to the networked devices. A combination of both hardware and software security could be the best practice.

Security becomes increasingly important with communication channels to the outside world. Self-driving cars may need car-to-infrastructure, car-to-car, and car-to-x interfaces. The house gets more doors and windows. And we all know, there is no error-free software. This means, one day your car will be hacked. In order to limit the damage, the inner doors should provide firewalls, too. Between each in-vehicle network segment we should establish some security mechanism. My wife always looks all inner doors, when we leave our home for vacations ("I want to make it as difficult as possible for the thefts"). She would like to put all our valuables in rooms without exterior doors, but our home is not a castle.

The "Cybersecurity Best Practices for Modern Vehicles" released last year by the NHTSA (National Highway Traffic Safety Administration) of the U.S. Department of Transportation contains a lot of recommendations and guidelines to built secure and functional safe road vehicles. "Logical and physical isolation techniques should be used to separate processors, vehicle networks, and external connections as appropriate to limit and control pathways from external threat vectors to cyber-physical features of vehicles. Strong boundary controls, such as strict white list-based filtering of message flows between different segments, should be used to secure interfaces." It is recommended to avoid sending safety signals on CAN or other networks. Instead, the ECUs should provide dedicated inputs from critical sensors, because this would eliminate the spoofing problem. "If critical safety information must be passed across a communication bus, this information should reside on communication buses segmented from any vehicle ECUs with external network interfaces." A segmented network may also mitigate the potential effects of interfacing insecure aftermarket devices to vehicle networks. Critical safety messages, particularly those passed across non-segmented networks, should employ a message authentication scheme to limit the possibility of message spoofing.

Author

Holger Zeltwanger
CAN Newsletter
headquarters@can-cia.org
www.can-cia.org

Debug over CAN

ST Microelectronics (STM) provides for its SPC57xx/SPC58xx family of 32-bit micro-controllers the option to debug the software via the on-chip CAN module.

The SPC5 series of micro-controllers is intended for automotive applications featuring functional safety (ASIL-d) and cyber security functions. The Debug-over-CAN is a new feature of the multi-core MCU. It is intended to enable debugging of ECUs in the field, where standard debug interfaces like JTAG or Nexus are usually not readily available, but access to the CAN interface is possible. Use of the CAN interface for debug purposes requires the use of the following resources to be exclusively used for that application:

- MCAN resources: Three CAN filters need to be configured for the three CAN debug messages. Furthermore two dedicated TX elements are needed for the transmission of the debug messages that are to be sent back to the external tool
- DMA resources: A total of six DMA transfers is required per Debug-over-CAN cycle. When using the scatter/gather mechanism of the DMA module only one DMA channel is needed, plus 192 byte of flash memory (32 byte for each of the six required Transfer Control Descriptors)

Initialization of both MCAN and DMA via software is required, but once this initialization has completed, debugging over CAN is possible without any further software overhead. The initialization can be performed in the boot code.

As the Debug-over-CAN scheme generates internal JTAG messages based on the received CAN data, all JTAG clients, and included debug resources are accessible. Basic trace capability is also possible by configuring the trace hardware to stream to an overlay/trace RAM, which can be read later using debug over CAN.

No other (external) JTAG tool/debugger can be connected when using Debug-over-CAN. If a JTAG tool is connected, the JTAGM is not able to access the DCI resources.

In a typical application in the field, an external debug tool sends debug CAN messages to one of the MCAN ▷

Figure 1: Debug-over-CAN sequence overview (Photo: STM)

modules of the micro-controller. The debug messages have specific CAN-IDs and have to be sent/received in a specific order but may be interleaved with non-debug CAN messages. A mechanism internal to the MCAN module consisting of debug message filtering and a debug message state machine handles the incoming CAN debug messages, stores them correctly in a specified RX Buffer and triggers DMA data transfers between MCAN and JTAGM.

The JTAGM acts as a JTAG master within the device and is used to generate internal JTAG messages. Three CAN messages are necessary to generate the JTAG messages. This means three MCAN message filters need to be configured for the debug messages and the DMA channel needs to be enabled for MCAN triggering. The MCAN modules are part of the Controller Area Network (CAN) controller alongside the Time-Triggered CAN (TTCAN) modules and the CAN RAM controller. The CAN implementation complies with ISO 11898-1:2015 and supports Classical CAN as well as CAN FD.

The RX Handler and the TX Handler provide all functions concerning the handling of messages. The RX Handler manages message acceptance filtering, the transfer of received messages from the CAN core to the Message RAM as well as providing receive message status information. The TX Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN core as well as providing transmit status information. Acceptance filtering is implemented by a combination of up to 128 base filter elements or 64 extended filter elements. Each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The MCAN modules 1 and 2 can be used for the Debug-over-CAN applications. Their role in this application is to receive and handle the incoming debug messages from an external tool, trigger the DMA transfer sequence, and to send internal debug messages back to the external tool. Incoming messages are identified as debug messages through a unique CAN-ID and then stored in a specified location in the receive buffer. After the three CAN debug messages have been received correctly and stored in the debug receive buffer, the MCAN triggers a DMA transfer to send the data to the JTAGM. The MCAN also receives data back from the JTAGM via the DMA. This data is to be sent back to external tool. The CAN reception and transmission of these debug messages has to be configured by the user.

Description of the debug sequence

Typically debugging over CAN takes several individual Debug-over-CAN cycles to accomplish a given task, e.g. a Nexus Read/Write Access. In one Debug-over-CAN cycle ▷

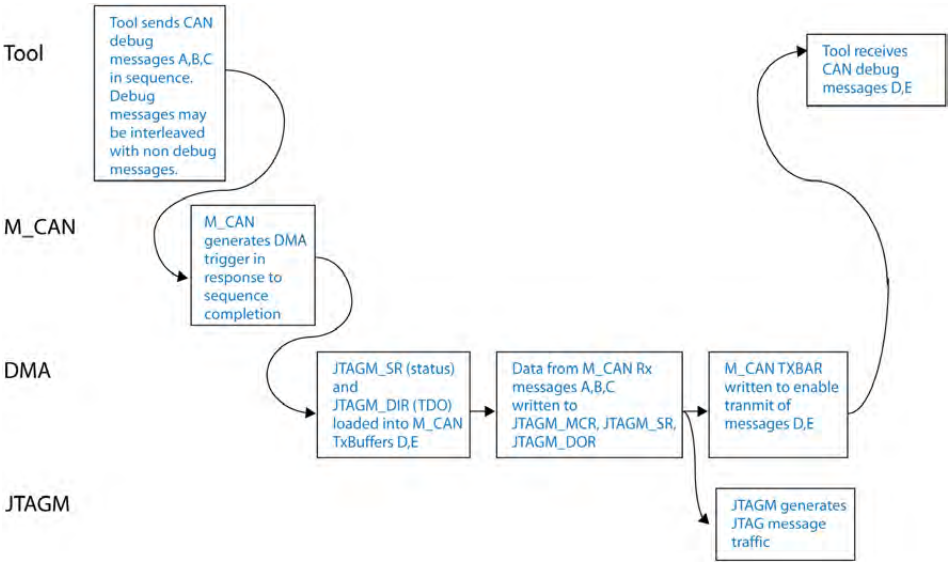


Figure 2: Detailed Debug-over-CAN sequence (Photo: STM)

an external tool sends three CAN debug messages A, B, and C sequentially to the MCAN of the micro-controller. The debug messages may be interleaved with none debug messages but must be in the correct order. The CAN debug messages are identified by a unique CAN-ID. When a debug message has been identified it is stored in the user defined RX Buffer in the CAN message RAM.

After the correct reception of debug messages A, B, and C, the MCAN triggers a DMA transfer. First, the content of the JTAGM status register (JTAGM_SR) and the JTAGM DATA IN 0 (JTAGM_DIR0) and JTAGM DATA IN 1 (JTAGM_DIR1) registers are transferred to the user defined MCAN TX buffer in the CAN message RAM. JTAGM_DIR0 and _DIR1 hold the debug data from the previous Debug-over-CAN cycle and hence needs to be read back before starting a JTAG traffic generation sequence.

After reading back the data from the JTAGM Status and Data-In registers, the relevant data from the debug messages A, B, and C in the RX Buffer of the CAN message RAM is transferred to the JTAGM's module control register (MCR), status register (SR) and data out registers (DOR0-3).

Writing the LSB in DOR3 sets the JTAGM_DOR3_SEND bit and internal JTAG traffic is generated depending on the data written to the JTAGM registers.

Parallel to that the MCAN sends the data that has just been stored in TX Buffers back to the external as debug messages D and E. The MCAN transmission is initiated using a DMA transfer to write transmission start bit of the given TX element.

Before starting the next Debug-over-CAN cycle the external tool waits for the two CAN messages D and E coming back from the MCU's MCAN. This provides enough time for the internal JTAG traffic to be finished and the data in JTAGM_DIR0 and _DIR1 has been updated.

MCAN debug message handling

Generally, the CAN modules on the SPC57xx/SPC58xx MCUs share 16 KiB of common CAN message RAM space. Within this CAN message RAM the user defines an individual area for each of the CAN modules. This area contains the message filters, Rx FIFO blocks, Rx Buffers, a Tx Event FIFO block and Tx Buffers.

The CAN modules accept CAN messages with matching CAN-ID. Any CAN message that is accepted by the CAN module will be stored as an element in either Rx FIFO_0, Rx FIFO_1 or Rx Buffer depending on the message filter that is configured for a given CAN-ID.

The eight data bytes (DB0 to DB7) contain the data that will be transferred to the JTAGM by means of the DMA module. If an extended filter is used the XTD bit is set to '1' and all 29 bits of the CAN-ID are used. For a base ID, the bits [28 to 18] are used for the 11-bit CAN-ID. In order to receive/accept messages and to handle debug messages as desired the correct filter settings have to be programmed into the filter blocks of the CAN Message RAM Configuration area.

As mentioned above, the three debug messages need to be received in the correct order. The DMS bit field in the JTAGM status register indicates the status of the FSA. Debug messages that arrive in the wrong order are rejected, and the state machine is reset to its initial state. The correct reception of debug message C triggers a DMA transfer (if configured correctly). However, before the debug data is transferred to the JTAGM, the status and the debug data from the previous Debug-over-CAN cycle must be read back from the JTAGM and transferred to the MCAN to be sent to the external tool.

DMA transfers between MCAN and JTAGM

The DMA transfer sequence is triggered by the MCAN when debug message C has been received successfully. A total of six DMA transfers are required to complete one sequence of debug message transfer between MCAN and JTAGM.

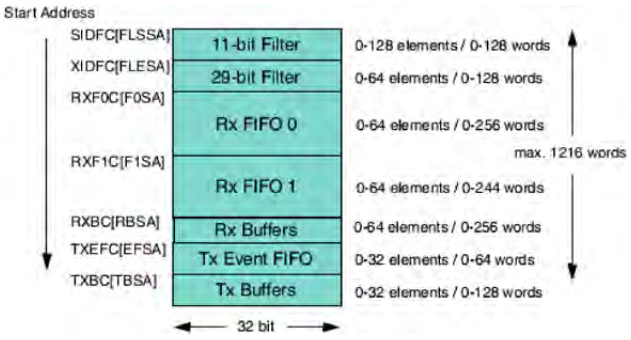


Figure 3: Debug Message Handling FSA (Photo: STM)

Reference

[1] Application Note AN4353: Debug over CAN. ST Microelectronics, 2017.

- First, two transfers are required to send data from the JTAGM to the transmit buffers of the MCAN
- Thereafter, three transfers are required to send the new debug messages from the MCAN debug message RX buffer to the JTAGM
- One transfer is required to trigger the MCAN transmission to send data back to the external CAN debug tool

One additional transfer is required when using the Debug-over-CAN feature with SPC5744K. Due to an errata the new data flags in the MCAN_NDAT registers have to be reset by a DMA transfer at the end of each Debug-over-CAN cycle. It is recommended to do that before triggering the MCAN transmission with the final DMA transfer. ◀

Author

Holger Zeltwanger
CAN Newsletter
headquarters@can-cia.org
www.can-cia.org

CANopen IoT integration via CiA 309-3

Many protocols are already in use for the Internet of Things. This article explains how to implement CANopen Internet of Things (IoT) via a CiA 309-3 gateway.

One of the oldest and most often used protocols is MQTT (MQ Telemetry Transport or Message Queue Telemetry Transport). The main advantage is its simplicity and the availability especially of the Client software for different programming languages and platforms. Ranging from C to Python, Javascript to Shell programming. And the protocol specification is open, with the same degree of openness as the CANopen specification. MQTT was developed as OASIS standard and has been approved by a joint technical committee of ISO and IEC and it has been given the designation 'ISO/IEC 20922' in July 2016.

Version 3.1.1 of MQTT was balloted through the Joint Technical Committee on Information Technology (JTC1) of ISO and IEC and given the designation 'ISO/IEC 20922'. MQTT is a Client Server publish/subscribe messaging transport protocol. Clients, in our case the CANopen gateway, can publish process values collected on the CANopen network, like PDOs, emergency messages, or network events. These values are delivered to the MQTT Broker. It stores these values for other Clients subscribed for these data. These Clients, once have subscribed for data, receive it as soon as the CANopen gateway publishes it.

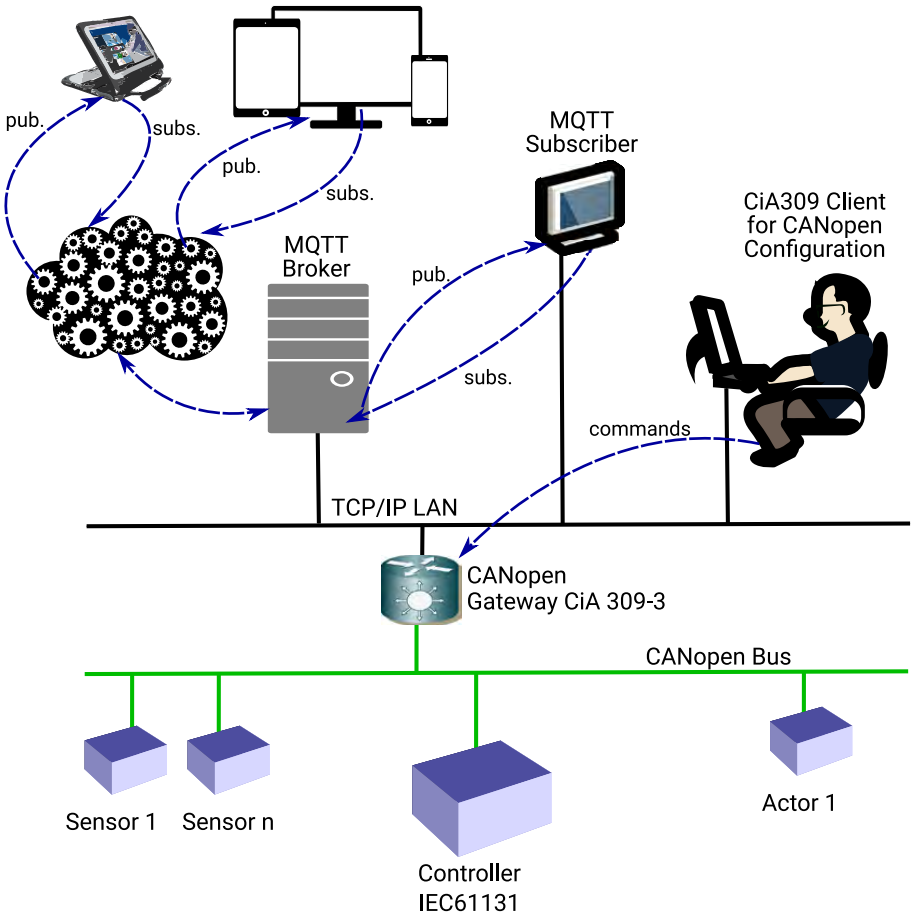


Figure 1: Illustration of connecting the gateway (Photo: Emtas)

The CiA 309-3 already follows some of same principles on the Ethernet (or simple serial console) side with its command based client interface. In the following we refer to a CiA 309 client as a client connected via TCP/IP. The commands allow that a let's say TCP/IP client subscribes for CANopen events like the Heartbeat or for PDO's on the CANopen network it is interested in. In the following it is assumed that the reader is familiar with CiA 309-3 and the MQTT standard. This command interface can be extended to extend the subscription in the way that subscribed data is published on the IoT side using the MQTT protocol. Let's look as an example. The CiA-309 command to register, or subscribe for a PDO:

```
[12] set rpdo 1 0x181 event 3 u8 u8 u16
```

This subscribes the CiA-309 client to a PDO which is sent by some (unknown) device in the CANopen network with a CAN-ID 0f hex 0x181. If this is one of the default PDOs, it could be the first transmit PDO of a CANopen device with Node-ID 1. To enable the same PDO to be published as MQTT data by the gateway it is only necessary to specify this behavior by a flag.

```
[12] set rpdo 1 0x181 event 3 u8 u8 u16 publish
[13] 1 start
```

In order to not violate the current specification of the `rpdo` command, the flag is appended as optional at the ▷

end of the command. The flag `publish` can be abbreviated by the letter `p` only. The correct syntax has to be specified by the CiA special interest group. The CiA 309 gateway will publish the data as soon as the specified PDO, a CAN frame with CAN-ID 0x181, is received on the CANopen network. The published data is the same as in the answer to the CiA 309 client.

MQTT communicated data is organized in a hierarchical manner. For a PDO it could be specified like:

```
/canopen/net1/rpdo1
/canopen/net1/rpdo2
/canopen/net2/rpdo4
```

Besides the CANopen gateway, other IoT connected devices can use the same MQTT broker. Therefore the data tree starts here with `/canopen`, which is followed by the network ID, a CiA 309 gateway can be connected to more than one CANopen network. The endpoint is the PDO number. Assuming now on the CANopen network CANopen with node number 2 sends its first TPDO on CAN-ID 0x182 and four data byte:

```
1492070538.675177      386/0x00000182 : bD ( 4): 01 02 03 0
```

An MQTT client subscribed to `canopen/net1/rpdo1` will receive the following in case of an RPDO1 event.

```
$ mosquitto_sub -t canopen/net1/rpdo1
pdo 1 3 1 2 1027
```

The used broker must be known, by the CiA 309 gateway and by all interested parties in order to subscribe at the broker for a specific topic specified by -t. The above and following examples use the defaults, the broker is at the local host, reachable at the default port (check mosquitto_sub(1) for details). With MQTT it is possible to use wildcards, the '#', to subscribe for data matching a pattern. Imagine that the CiA 309 client has subscribed for three PDOs

```
[1] set rpdo 1 0x181 event 3 u8 u8 u16
[2] set rpdo 2 0x182 event 2 u16 u16
[3] set rpdo 3 0x183 event 2 u16 u32
[4] 1 start
```

and on the bus are the following PDO's

```
1492082052.444469 385/0x00000181 : bD ( 4): 37 64 10 27
1492082085.220261 386/0x00000182 : bD ( 4): 00 01 00 02
1492082092.804541 387/0x00000183 : bD ( 6): 11 27 41 42 0f 00
```

The MQQT subscriber using the wildcard for all data from `canopen/net1` receives:

```
$ mosquitto_sub -t /canopen/net1/#
pdo 1 3 55 100 10000
pdo 2 2 256 512
pdo 3 2 10001 1000001
```

Of course the hierarchical schema could be extended to something like:

```
canopen/net1/rpdo1/mapping1
canopen/net1/rpdo1/mapping2
canopen/net2/rpdo4/mapping1
```

In this case subscribing `/canopen/net1/rpdo1/mapping1` would only receive a single signal mapped at the first position of a TPDOx in network 1, and `/canopen/net1/rpdo1/#` will receive all single mappings of this ▷

PDO, and again `/canopen/net1/#` will receive all events happening on net1, not only PDOs. Other useful events for listeners are network events on the CANopen bus, like EMCY or HB failures, as defined in CiA 309-3.

```
canopen/net1/emergency/node1
canopen/net1/heartbeat/node10
canopen/net1/emergency/node2
canopen/net1/heartbeat/node11
```

These are following the same rules as for PDOs. These events are presented to MQTT subscribers in the same way as for CiA 309 clients. In addition to the well known addressing scheme – network – node- index – subindex an additional addressing schema for functional addressing can be used. *ReferenceDesignators* can be used to address a specific network, node, or parameter of a device. Assuming CANopen is used as default network and CANopen net1 is installed in the ware house connecting some sensors and actors, using this schema a value could be addressed as `/warehouse/AC/temperature-sensor42`. This schema shall also be useable with MQTT. Another interesting application is connecting publicly available city Pedelecs. They can provide data like:

```
/citybike/bike155/location
/citybike/bike155/reach
```

To provide most interesting information on it's current location as GPS data and estimated drive range in km for a mobile App users can use to get a Pedelec in the city. Dream of more applications…

Configuration of the MQTT publisher

The configuration is divided in two parts. The first configures the amount and presentation of the data to be published. Examples have been given above.

The second part is the configuration of the access to the MQTT broker, the instance holding the data for subscribers of the data. There is a minimum of configuration parameters the MQTT broker's IP address and the used IP port number for the service. This can be done by using the CiA 309 client interface the same way as defining the RPDOs. In this case changes of the 309-3 standard are necessary. An other possibility is implementation specific for the gateway, e.g. passing these data on the command line at startup. And of course, when using commands for the CiA 309 client, all these configuration commands can be stored in a file and be executed at start up or at any time by a Gateway proprietary command. In addition of using CiA 309-3 for the configuration, the newer CiA 309-5 will define additional methods to provide web services to access the data in CANopen networks. Basically it is a mapping of CiA 309-3 commands into HTTP requests.

The MQTT protocol allows to use secure connections. Security and IoT is still an underestimated topic. Often devices are connected to the internet without any additional security protections. But connecting objects like machines exposes lots of sensitive data. There are different kinds of data, which are not meant for the public. For this kind of data standard security measures: confidentiality, integrity, and availability should be ensured. Security in MQTT is solved at different levels. Common implementations are using protocols like SSL/TLS for transport security. On the network level using a physically secure network or VPN are ways to establish a trustworthy connection. At the application level communication is encrypted and the identity is authenticated in different ways. These MQTT specific topics are not in scope of the article and can be read at the official MQTT 3.1.1 specification. ◀

Reference

- OASIS (http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html)
- MQTT Security (http://www.hivemq.com/blog/introducing-the-mqtt-security-fundamentals)

Author

Heinz-Jürgen Oertel
emtas GmbH
oe@emtas.de
www.emtas.de

Decentralised signal detection and processing

WARRANTY 5 years on ifm products

I/O module designed for mobile applications with integrated PLC

The ioControl module can either be used as a configurable I/O CAN slave in a decentralised control system or as a compact PLC in the field.
The high protection rating and robust housing make it suitable for installation in wet and dirty areas of mobile machines. Programmed with CODESYS.
Practical solutions for automation by ifm – close to you!

www.ifm.com/de/en/iocontrol
Phone +49 800 16 16 16 4

Time-stamping of CAN frames

CiA 603 specifies an Autosar-compliant time management for CAN networks. CAN controllers will support this.

The currently used Autosar (Automotive Open System Architecture) compliant time-base synchronization is implemented in software. In order to achieve higher time accuracy, a hardware implementation is needed. The CiA 603 document specifies a hardware time-stamping concept to be implemented in future CAN controllers. This hardware approach is independent of interrupt response times and results in higher accuracy of the time-base synchronization.

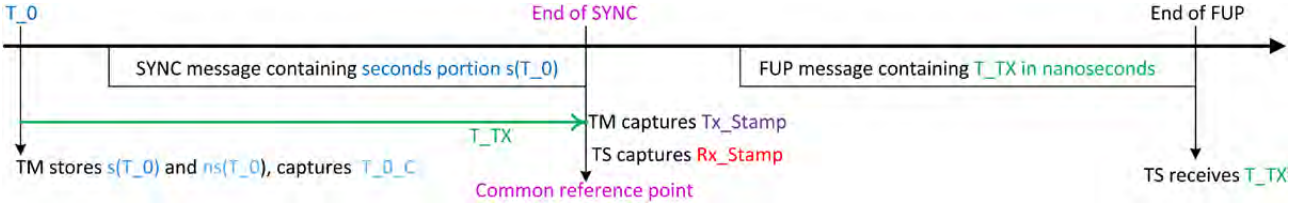


Figure 1: Time synchronization process (Photo: Bosch)

In Autorsar systems, time is generally represented as a 64-bit number. The actual time value in an ECU is given by adding the value of a 32-bit free-running timer counter to the value of a 64-bit time-base register. An ECU may be linked to several time domains, with different time-base registers, but sharing the same timer counter. Each time domain has one time master and several time slaves. The time master synchronizes the time slaves by propagating, over the communication network, the time-base value to the time slaves.

A time-base can be distributed between networks that are connected by a time gateway. The time gateway receives the time-base as time slave from one network and propagates it as time master to the other networks. In the following only the synchronization of the time-base over the CAN network is regarded.

Synchronization over CAN

In the first step of the synchronization procedure, the time master saves the actual time T_0 at the beginning of the procedure in seconds-portions s(T_0) and nanoseconds-portions ns(T_0), as well as the actual value T_0_C of its timer counter (a 32-bit number). The time master writes s(T_0) into the transmit buffer for the Sync message and requests the CAN controller to transmit it. When the CAN controller has successfully transmitted it, this event triggers the capture of the actual timer counter value as Tx_Stamp. From this the time master calculates T_Tx, which is the time from s(T_0) to the end of Sync's transmission: T_Tx = ns(T_0) + ns(Tx_Stamp - T_0_C).

In the second step of the synchronization procedure, the time master writes T_Tx (a 32-bit number representing nanoseconds) into the transmit buffer for the FUP (follow-up) message. The data of the FUP message is complemented by two additional bits that signal whether there was an overflow of the timer counter or in the calculation of T_Tx. For the time master, the synchronization procedure ends when the CAN controller has transmitted the FUP message.

The Sync and FUP messages are transmitted using the same CAN-ID; additional coding in the data field distinguishes the both messages, identifies the time domain, and enables error checking. While an ECU uses only one CAN-ID if it is time master for different time domains, the CAN protocol requires that other time masters (of other time domains) on the same CAN network use different CAN-IDs.

A time slave starts the synchronization procedure at the reception of the Sync message, which triggers the capture of its timer counter value as RX_Stamp and provides the seconds-portion of the time master's T_0. The capturing of Tx_Stamp in the time master and Rx_Stamp in the time slaves is triggered in all nodes by the end of the same CAN data frame (Sync message). The different nodes see this event with a phase shift of less than one CAN bit time.

The time slaves enter the second step of the synchronization procedure at the reception of the FUP message. This message enables the time slave to calculate, based ▷

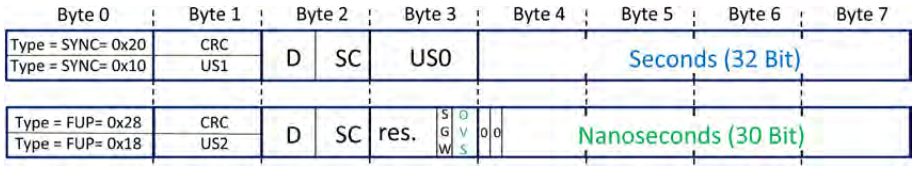


Figure 2: Autosar CAN synchronization messages (Photo: Bosch)

the value of its timer counter TC, the received s(T_0), and its Rx_Stamp, the actual time T_a: T_a = s(T_0) + T_Tx + ns(TC – Rx_Stamp).

Repeated synchronizations allow the time slaves to adjust their local clock speeds. It is not necessary to increment the timer counter in all nodes at the same speed, because in the Sync or FUP messages: all time information is transformed into real time units, seconds in the Sync message, and nanoseconds in the FUP message.

Advantage of time-stamping in hardware

Software implementations for CAN are based on Sync messages that trigger interrupts at frame transmission (time master) and reception (time slaves). The interrupt service routines capture and compare the values of free running counters and calculate, with the help of a FUP message, the actual time offset between time master and time slaves. The accuracy depends on the interrupt response times after the Sync message. The synchrony between the time-stamps Tx_Stamp and Rx_Stamp is worsened by latency jitter.

When the timer counters are captured in hardware, directly triggered by the CAN controllers, instead of being captured by the interrupt service routines, latency jitter is avoided and the accuracy of the synchronization is improved.

The purpose of the CiA 603 is to specify, beyond the functions already specified in ISO 11898-1, which functions CAN controllers should provide to support the Autosar-compliant synchronization method.

In ISO 11898-1, time-stamping is specified for the support of Time-triggered CAN (TTCAN) as standardized in ISO 11898-4. These time-stamps are captured at the SOF (start of frame) bit and they are 16-bit numbers, using the CAN bit time as time steps. In current Autosar-compliant systems, time-stamps are captured at the EOF (end of frame), by the message's transmission or reception interrupt service routines. They are 32-bit numbers, using smaller time steps.

The gradual, non-disruptive integration of new nodes with CAN time-stamping in hardware into existing systems requires that the hardware time-stamps are also captured at EOF. To achieve the necessary precision, the time-stamps need to be 32-bit numbers, captured from timer counters with time steps of less than one CAN bit time. These features enable hardware-based time-stamping nodes to participate in the synchronization procedure with software-based time-stamping nodes in the same network.

CiA 603 specifies that time-stamps are captured at EOF, when the data frame becomes valid according to the CAN protocol. That is the last-but-one bit of the EOF field for the received Sync messages and the last bit of the EOF field for the transmitted Sync messages. These are the same conditions that trigger the message's transmission or reception interrupt flags. The one CAN bit time difference (plus the signal delay from the receiver's ACK to the transmitter) between the two triggers is well known and can be considered in the time slave's calculations.

Time-stamps are captured from a free-running 32-bit counter that is incremented in steps of at least 1 ns and at most 1 µs; it counts upwards and overruns to zero. The counter may be inside the CAN controller or outside. The software can read anytime its value. Several CAN modules may share the same timer counter.

It is not necessary to store a time-stamp for each message transmitted on the CAN network. The time master needs a time-stamp only for the transmitted Sync messages, its capture can be controlled by that message's transmit buffer configuration. A time slave also needs to store time-stamps only for the Sync messages, but storage is needed for two time-stamps since the CAN controller's acceptance filtering cannot distinguish between Sync and FUP messages that use the same CAN identifier.

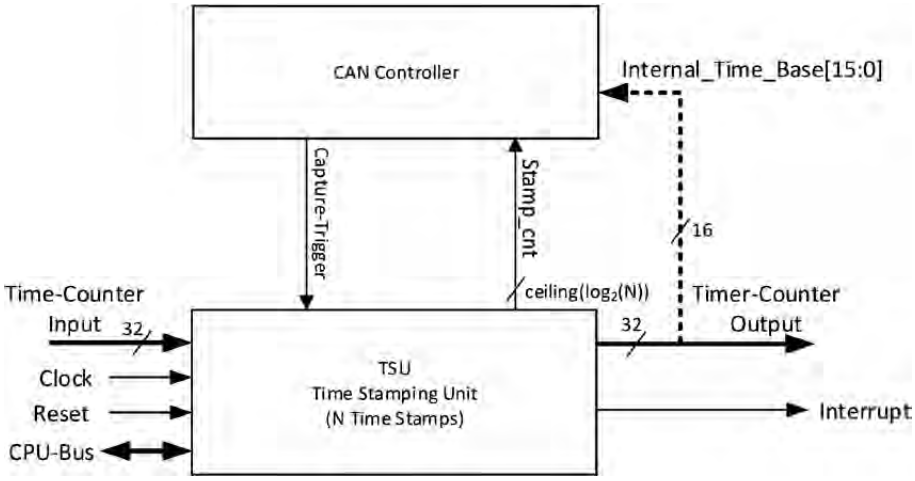


Figure 3: A separate TSU simplifies the CAN controller re-design, when the time-stamping is added (Photo: Bosch)

In CiA 603, it is mandatory to provide storage for at least two Rx_Stamps and at least one Tx_Stamp, or at least two time-stamps if storage is shared between them. In order to be able to support multiple time-bases concurrently, it is recommended to provide at least four times the mandatory minimum storage. Autosar systems may have up to 16 synchronized time-bases.

Separate time-stamping unit

Not all existing CAN controllers support time-stamping of messages. If they do, time-stamps are usually 16-bit wide and are stored inside the message buffer structure. If the position is not configurable, the time-stamps are captured at the start of frame.

Changing the width of the stored time-stamps to 32 bit (half of a Classical CAN data field) would require restructuring and enlarging the CAN message storage area. The CAN driver software would need to be adapted to the new structure. The solution to this problem is to implement the new hardware time-stamping function not into the CAN controller itself, but into a separate module, the Time-stamping Unit (TSU). The CAN controller is only minimally modified, keeping its controller host interface unchanged.

The interface between the CAN controller and the TSU can be kept simple. The CAN controller provides trigger signals to capture the time-stamps and the TSU provides information that indicates which time-stamps belong to which messages. If there is more than one CAN controller, they may share one TSU, otherwise each CAN controller is connected to a dedicated TSU.

The TSU has its own controller host interface (CHI), to configure and to control its function, and to read the captured time-stamps. The TSU may include the free running timer counter with an optional prescaler, alternatively, an external timer counter may be connected. The timer counter value may be cascaded from one TSU to the next and it may be used as time-base for legacy time-stamping with less resolution.

The time-stamps are stored inside the TSU in a circular buffer, addressed by a counter. The elements of the circular buffer can also be read by via the CHI. Each time a capture is triggered, the address counter is incremented; the counter over-flows to zero. The address counter value is provided to the CAN controller where it is stored with the message buffer, instead of the 32-bit time-stamp itself.

The number of time-stamps stored in the TSU can be decided by a generic parameter, changing the size of the module. The interface signals of the TSU are, with the exception of the address counter width, not changed by the size of the circular buffer.

The TSU may optionally include software debug support, flags that show whether a time-stamp register contains new data or whether unread data was overwritten.

The timer counter input vector is not needed when the TSU implements an internal timer counter. If several TSUs are cascaded, they share the same timer counter. The TSU's interrupt output may optionally be used to signal the capture of a new time-stamp or when a time-stamp register was overwritten before it was read. It is not needed for time-base synchronization.

The CAN controller activates the capture trigger for relevant messages, e.g. when a message is received that is recognized as Sync message by CAN's acceptance filtering or when it is transmitted from a correspondingly configured transmit buffer.

The 3-bit cyclic stamp counter value for storage of eight time-stamps shows into which time-stamp register the currently triggered time-stamp is stored. In CAN controllers designs that already support (shorter) message time-stamps, this counter value can be stored instead of the time-stamp, generally for all messages or only for Sync messages.

Author

Florian Hartwich
Robert Bosch
florian.hartwich@de.bosch.com
www.bosch.com

TTControl
HYDAC INTERNATIONAL

Highly Robust Operator Interfaces

Usability

- Excellent sunlight readability
- Ability to display videos and PDF documents
- Programming and debugging facilitated by CODESYS® V3

HY-eVision² Family

Performance

- Best-in-class CPU performance
- OpenGL graphics with hardware acceleration
- Fast boot-up time
- Sleep mode, wake-up pin and wake-up timer (<0,5s)

Connectivity

- Optionally GPS and GSM enabled
- Two camera interfaces with picture-in-picture functionality
- Up to 4 CAN interfaces
- Interface for Ethernet cameras
- WLAN interface

www.ttcontrol.com/HY-eVision2-Family


Safety Certified ECUs

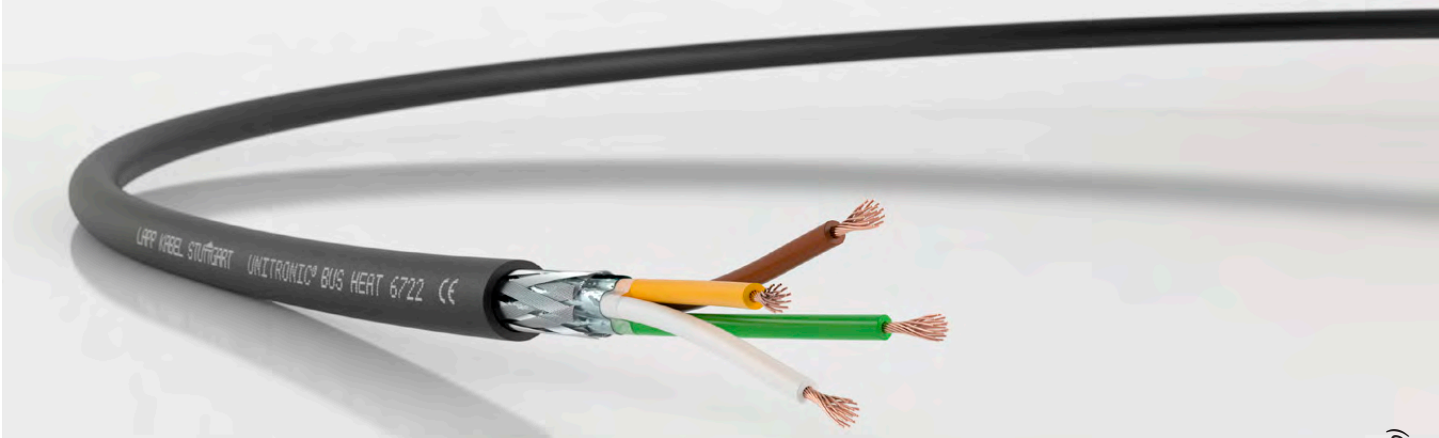

General Purpose ECUs


I/O Modules


Safe I/O Modules


Operator Interfaces

It's the cables that count



Lapp came up with the idea of developing their own CAN cable specifically designed for commercial vehicle bodies.

(Photo: Lapp)

Commercial vehicles are now also being fitted with CAN technology for data transmission yet the cables are often not custom-made for this type of vehicle. The Lapp Group, a manufacturer of cables and connection solutions, plans to provide a remedy with a cable that satisfies all requirements in terms of robustness, fire safety, and flexible applications. Bosch introduced Controller Area Network (CAN) in 1986. This standard has taken root as the central "nervous system" for transferring data to vehicle electronics in cars, and later in commercial vehicles, ever since. Especially in the automation sector CAN is used. Perhaps less well-known is the fact that CAN in mobile applications is not just featured in the vehicle itself but also in its bodies. There is a vast range of potential applications for the CAN network, including bodies for pick-up trucks, panel vans, tank trucks, car carriers, concrete mixers, glass and logging trucks, dump trucks, low loaders, cattle trucks, road sweepers, dustbin lorries, and snow-clearing vehicles.

The cables in a vehicle are extremely long. According to a VDE study, 1 800 m of data cables are tucked away in the BMW 3 Series. And a Claas combine harvester features a staggering 3 000 m of cables in four CAN networks with 350 connectors. FireCAN, a group of 25 manufacturers of fire-fighting equipment, is a supporter of fast mobile data buses. A few years ago, it launched a standardized system for managing electronic applications in fire engines that uses CAN. Its standard connectors enable components made by various manufacturers to be connected in line with the plug and play principle. Nevertheless, a little more attention should have been paid to the cables used to transfer the data as no requirements have been defined for them. So members of FireCAN and the manufacturers of vehicles with other bodies have been using cables that were actually developed for other uses, e.g. automating industrial machinery. Lapp products have also been used here as they are robust, even though they are not designed to handle the specific stresses of fire-fighting equipment.

Specialist not generalist

Standard CAN cables can of course be used for vehicle bodies, and Lapp's cables, which are renowned for their robust properties, can master this task confidently. But this is not an ideal situation as the requirements for a cable in a factory are different to cables used outside and in vehicles, as is the case with fire fighters and many of the aforementioned vehicle types. There are currently no cables specifically designed for this purpose and no other manufacturer has yet looked into this. This is all the more astonishing as around 250 000 new commercial vehicles are registered every year in Germany alone. German manufacturers of commercial trailers have a market share of around 50 % in Europe. The export share is also around 50 %. As such, there is huge potential for CAN cables in commercial vehicle bodies. Talks between Lapp and potential customers revealed that they would be glad to see cable types adapted to this use. So Lapp came up with the idea of developing their own cable specifically designed for commercial vehicle bodies.

The list of requirements for this cable type is long: it needs to withstand temperatures of -40 °C to +105 °C in line with DIN/ISO 6722 class A+B and it needs to be resistant to oil, petrol, diesel, lubricants, and many other chemicals. As the cable is used outdoors, it needs to be resistant to UV light and weather conditions. As it is also sometimes laid in areas where people are present, certification under ECE R118 (the burning behavior of materials used in interior construction) is mandatory. According to this regulation, the sheath material must be halogen-free so that, in the event of a fire, a person's airways are not chemically burnt when the blazing plastic comes into contact with extinguishing water. As a result, only a sheath made of special polyurethane can be used here.

Lapp was able to build on its expertise when it developed the Unitronic CAN cable. Its portfolio already included cables for various uses in commercial vehicles, for instance:

- Ölflex Truck for the electrical wiring in truck trailers, which is also approved for hazardous materials transportation as a result of the special ADR approval. The cable features an outer sheath made of either special PVC or special PUR. The latter is also microbe resistant;
- Etherline Heat 6722 is designed for data transmission inside buses, e.g. security cameras or entertainment systems in luxury coaches. The cable is halogen-free, flame retardant and tested in line with the ECE R118 standard;
- Unitronic Bus IS is an Isobus cable that complies with ISO 11783-2. It is based on the CAN standard and is used to transfer data in agricultural vehicles.

Specialized cables for CAN

Unitronic Heat 6722 is now the fourth cable in this group. It features the aforementioned properties, in particular the temperature range and robustness to all possible chemicals, and is certified in line with ECE R118. To a certain extent, it is a combination of the excellent features of the three existing cables. But the smaller diameter and thus lower weight is new. The outer diameter of the Unitronic Heat 6722 is 2,4 times the diameter of the four individual cores within this cable, making it 40 % thinner than conventional cables. The Lapp engineers managed to achieve this by forming a star quad, also known as a twisted quad, in the cable. According to experts, this means a cable with four cores where the opposite cores are twisted together. This saves space and weight and allows for narrow bending radii. The cable has a highly flexible sheath and fine-wired strands, so it is ideal for installation in moving applications, e.g. a drag chain in an extendible fire ladder.

Author

Juergen Greger
Lapp Group
juergen.greger@lappgroup.com
www.lappgroup.com

CAN FD is set, but still new ideas are popping up

In the automotive industry, CAN FD is well accepted. Nevertheless, there are some new ideas to increase the throughput and to improve the physical layer.

The success story of CAN FD continues. Pre-developed by Bosch and standardized in ISO 11898-1:2015, the CAN FD protocol has been implemented by all market-leading automotive chipmakers. The only drop of bitterness was the so-called CRC issue. But in the meantime it is fixed by means of the additional stuff-bit counter integrated into the CRC field. Nevertheless, there are still some non-ISO CAN FD controllers on the market. They are already designed-in in some local, deeply embedded interfaces in ECUs (electronic control units). This hurts the toolmakers, because they have to support ISO CAN FD and non-ISO CAN FD in their network interfaces and their analyzing software. For carmakers they are invisible.

The good news: Nearly every major OEM (original equipment manufactures) including BMW will migrate step-by-step to CAN FD. In the beginning, the carmakers were thinking to run Classical CAN and CAN FD communication on the same network segment. Except for a few Chinese brands, this is not a real requirement anymore. The migration to CAN FD will be in most cases a complete one or the OEM strictly separates Classical CAN and CAN FD network segments.

This is, why the idea of Satheesh K. Kini (Mercedes-Benz India) has not been really discussed in the CAN community. He proposed a small change in the CAN FD protocol. His idea was to mute Classical CAN only controllers during the transmission of CAN FD data frames. But he was to late, the CAN FD protocol was already standardized in the ISO working group and first chipmakers had implemented already the CAN FD protocol. So his proposal was not considered.

By the way, there are other solutions to integrate Classical CAN only nodes into networks running CAN FD without impacts on the protocol level. One of them is the hiding of CAN FD frames by means of a “smart” transceiver. NXP will sample its CAN Shield transceivers by end of this year [2]. The main market for such a product seems to be non-automotive applications. Kvaser has developed a smart approach [3].

CAN FD is by some means a new protocol. Of course, the current implementations support both Classical CAN and CAN FD, because this is required by ISO 11898-1:2015. CAN FD has many similarities with Classical CAN. This makes the migration easy and also the acceptance. Nevertheless, CAN FD is a new protocol. All other statements are marketing. The still simple data link layer protocol, the still inexpensive controller and transceiver chips, and the low-power consumption of CAN FD nodes made the new protocol so successful. It has been adapted by the automotive industry very quickly. Robustness and reliability are also key features of the success. Experiences made in several CAN FD plugfests organized by CiA have proved this.

TurboCAN: More than 100 Mbit/s

This is the promise of South Korean scientists. Suwon Kang, founder and CEO of VSI, and his team have developed the TurboCAN approach, which is backwards compatible with Classical CAN supporting. It supports bitrates higher than 100 Mbit/s. This would be a real competitor to 10-Mbit/s Ethernet, even knocking at the door of 100-Mbit/s Ethernet. Suwon Kang presented this solution on an IEEE conference and published it last year also in the IEEE Communications Magazine. The research project was partly supported by the Institute for Information & Communications Technology Promotion (IITP) financially funded by the Korean government.

Suwon Kang sees the primary cause of the data rate limitation of the CAN system coming from three factors. The first one is the constraint of the bus characteristics, which limits the minimum clock pulse width, which then limits maximum clock rate. Secondly, due to the attenuation at higher frequency, higher clock pulse suffers from severe edge degradation that could render received waveform hard to detect properly. Finally, only binary signaling is allowed in the standard with very low bandwidth utilization.

Suwon Kang wrote: “The proposed scheme overcomes these limitations by adding carrier modulation to the CAN frame along with higher bandwidth utilization. One of the biggest advantages of using carrier modulation for data transmission is that the proposed system is no longer dependent on edge detection using bit transitions. It also enables the use of higher bandwidth modulation sending multiple bits for each transmit symbol, providing higher data rate without transmission bandwidth increase.”

In contrast to the binary signaling of CAN, the use of multi-level modulation can increase the throughput. The transmitted CAN bits are used to form a complex symbol to be modulated and transmitted on the network. The complex symbol can be constructed using any modulation scheme, including quadrature phase shift keying (QPSK), 16-quadrature amplitude modulation (QAM,) 64-QAM, etc. According to Kang, higher order modulation is preferred to achieve higher data rates. However, the choice of modulation scheme is related to the signal-to-noise ratio, frequency attenuation characteristics of the channel, and receiver complexity.

The team of Suwon Kang has not yet implemented the described multi-level modulation approach. But they have simulated it. The results are discussed in detail in the mentioned IEEE paper and article [1]. The proposed carrier modulation on top of the Classical CAN signals allows significant higher bit rates. The performance of the scheme in the CAN environment has been evaluated in terms of BER (bit error rates) and net throughput to show that the proposed scheme can provide higher data rate, while keeping backwards compatibility with the CAN protocols (Classical and CAN FD). With the use of a longer frame supported by CAN FD, the net throughput can be increased to 161,8 Mbit/s claims the researchers. "The proposed scheme can easily be applied to the existing CAN network without additional deployment of cabling to support the need for high data rate links between devices, resulting in significant reduction of the cost and weight of the vehicle," stated Kang in his paper. The team of Suwon Kang is working on the implementation of the multi-level modulation scheme now and the key components including controller and transceiver are said to be available next year.

It seems that the proposed multi-level modulation approach can be implemented on top of the existing CAN protocols. Nevertheless, it is not proven that it can be used on the topologies used by the OEMs. It looks like that the simulation results are based on a strict bus-line topology. But even if just half of the simulated bandwidth can be achieved, this would be an interesting approach, because the existing wiring harnesses for Classical CAN or CAN FD can be re-used. The next step would be to prove the concept by means of a prototype implementation. This could be submitted to one of the CAN FD plugfests to see, if it has really no impacts on the CAN communication and which throughput can be achieved in practice.

Re-using is the key. Since years, new network architectures have been discussed, for example, the so-called domain architecture, but none of the OEMs has made already the paradigm change. Still improving existing solutions is on the agenda.

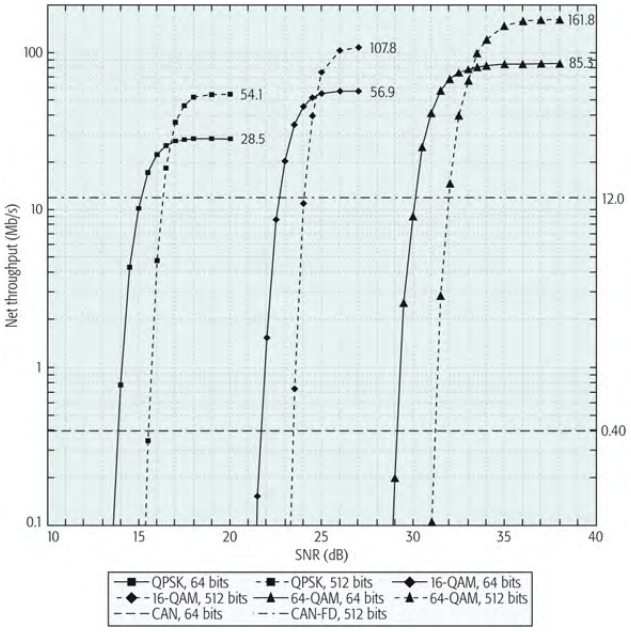


Figure 1: Simulated net throughput of the proposed multi-level modulation scheme on CAN signals (Photo: IEEE Communications Magazine)

Ringing suppression

Another option to improve the data throughput in CAN FD networks is the suppression of ringing. Today we can achieve in bus-line topologies 2 Mbit/s in the data phase without problems. In point-to-point links, 5 Mbit/s are realistic. This could be used for software download, for example. Star and hybrid topologies are more challenging. Even 2 Mbit/s is sometimes a challenge. Ringing suppression as proposed by Denso and specified in CiA 601-4 is a possibility [4]. In Detroit at the WCX17 conference organized by SAE (Society of Automotive Engineers), Denso presented simulation results for 5-Mbit/s CAN FD networks. Still this is just simulation.

Recently, NXP pre-announced another ringing suppression solution. In opposite to the Denso approach based on circuitry suppressing the ringing on the receivers, NXP proposed a ringing suppression on the transmitting node. This saves some time, because you don't have to detect the ringing; you can just avoid it. But NXP has published just a few details in a presentation by Tony Adamson during the iCC 2017 in Nuremberg. He promised to provide more information in this summer, when discussions with OEMs have made some progress.

References

[1] Kang, S. and others: High Speed CAN Transmission Scheme Supporting Data Rate of over 100 Mb/s. IEEE Communications Magazine, June 2016, page 128 to 135.

[2] Bernd Elend, "Transceiver with cyber security functions", CAN Newsletter 2/2017

[3] Kent Lennartsson, "CAN FD filter for Classical CAN controllers", iCC 2015

[4] CiA 601-4, CAN FD

Author

Holger Zeltwanger
CAN Newsletter
headquarters@can-cia.org
www.can-cia.org

Implementing a program flow using hooks

Kvaser supports the t-script language for its CAN interface products. The t-scripts are based on "hooks". Hooks are like interrupts. The runtime engine waits for these hooks.

(Photo: Kvaser)

The hooks invoke *t*-programs, which are entry points that are executed at the occurrence of certain events. These events include reception of dedicated CAN messages, timer expirations, or external inputs. When such an event occurs, the runtime engine executes the code that is inside the hook's code block. An example: You need to respond to the CAN-ID 200_h. The *t*-script code would look something like:

```
On CanMessage 0x200{   replyFunc(); }
```

When a CAN message is received with an ID of 200_h, the *t*-script will raise replyFunc() to send a CAN response message, for example. Then it will leave the code block when complete. However, each time a message is received with the ID 200_h, this is repeated. This can make creating a program flow challenging.

Judson Brundage from Exclusive Origin recommends to use a switch/case statement with a global incrementor. Now, you can enter the hook and reference the incrementor – just as you would use a bookmark. The software will execute some process, and increment to bookmark before it leaves. See the following code example:

```
On CanMessage 0x200{   switch(bookmark){   case 0:
replyFuncStepOne();   bookmark++;   break;   case 1:
replyFuncStepTwo();   bookmark++;   break;   case 2:
replyFuncStepThree();   bookmark++;   break;   } }
```

The *t*-language is a C-like, event-oriented script language. The *t*-program runs on the CAN interface device and can be loaded and started either from a PC or autonomously on the device. To be able to test and run a *t*-program you need a Kvaser Professional product. The software needed is installed with CANlib SDK and called Kvaser TRX. To get up and running and write your first *t*-program, start by downloading and installing CANlib SDK then start TRX. It will look something like this:

```
on start {   printf("Hello World!\n"); }
View sourceCopy to clipboard
```

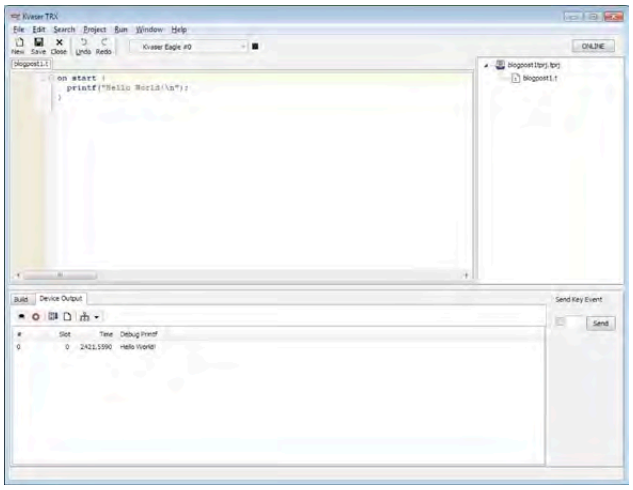


Figure 1: Screenshot of the t-program window (above) and the output device (below) (Photo: Kvaser)

This is a very basic program that reacts on an 'on start' event and prints "Hello World!" in the 'Device Output'. It is the actual device that sends the "Hello World!" to the computer and that TRX reads. Then choose `Run->Compile` from the menu. To be able to compile you first have to save the *t*-program, TRX will prompt you for this. It will also prompt you to save a project file.

If the compilation succeeds, the next step will require an actual Kvaser professional product, for example the Eagle. You can see that TRX can use your device, if the combo-box just under the menu-bar has the name of the used device. To test the *t*-program, just go to `Run->Download`, this will download the *t*-program to the device. Next step will be to run the program. `Run->Run` will start the application, thus hopefully you will see the `printf` printout in the device output window. ◀

Reference

The Kvaser t Programming Language, February 2015

Author

Holger Zeltwanger
CAN Newsletter
headquarters@can-cia.org
www.can-cia.org

Rotary actuator for positioning operations

RD6 is a compact rotary actuator from Lika Electronic that integrates a BLDC motor, an absolute encoder, and control electronics in a single package. It is available in two sizes with 157-W and 250-W power ratings and CANopen.

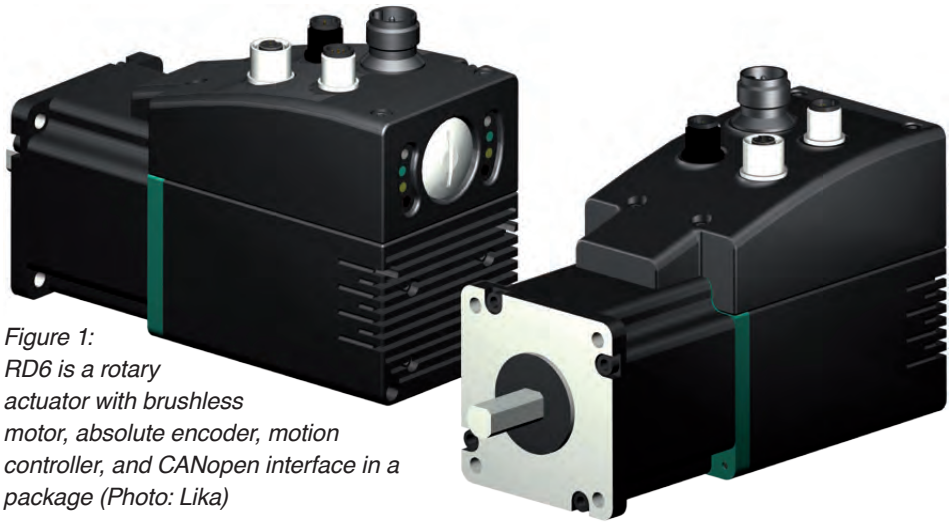


Figure 1: RD6 is a rotary actuator with brushless motor, absolute encoder, motion controller, and CANopen interface in a package (Photo: Lika)

Increasingly, modern industries require automated production processes with little downtimes to ensure efficiency, provide precise control and repeatability, increase productivity, and improve product quality. Meanwhile, the "large batch, long run" philosophy is becoming obsolete because volatile demands call for quick responses today. Small batches, one-off items, just-in-time production, and the acceleration of cycle times often drive businesses.

The RD6 rotary actuator is Lika Electronic's most complete, powerful, and up-to-date solution developed to help the modern industries solve these tasks. It is designed to drive positioning systems, changeover applications and linear guides, and allow to reduce set-up and change-over times, as well as ensure maximum efficiency and precise repeatability.

The positioning actuator without gears was developed to make positioning operations more efficient and powerful, so production processes can benefit from faster adjustments, shortened downtimes, and reduced risks of errors and waste. Its "all-in-one" configuration with embedded intelligence provides the user with a simplification of design and integration.

The actuator features a space-saving and rugged anodized aluminum housing capable of IP54 protection for installation in typical industrial environments. The 70 mm size square flange and the 14 mm shaft are designed for coupling with standard planetary gearboxes. Thus the unit can be integrated into custom applications to meet specific torque and speed requirements. The compact enclosure integrates a BLDC motor, a real multi-turn absolute encoder, a position and torque closed-loop controller, and a fieldbus interface. Its advantage is its simple installation while the cabling operations are reduced to a minimum, saving time and money. The 24-V_{DC} brushless motor is available in two sizes: 157-W rated power, 0,5 Nm-rated torque, 3000 RPM; or 250-W rated power, 0,8-Nm rated torque, 3000 RPM. The built-in encoder for accurate motor feedback provides an overall resolution of 28 bit (4096 counts per revolution and 65 536 revolutions) with a position accuracy of ±0.9° and features a real multi-turn that requires neither battery nor counter.

The actuator is versatile and open thanks to the integration of the CANopen standardized interface in compliance with the CiA 301 profile. The CANopen interface provides functionalities for motion-oriented machine control systems and especially for time-critical processes. Among the implemented features are the position and velocity readout, full scaling, jog, preset, software limit switches, extensive diagnostics, and network settings. The internal trajectory generator (boasting a 64-bit double precision) allows the operator to set a new target position on-the-fly. In addition, the integrated EiA-232 service interface and the free software tool allow easy set-up and configuration. Using this program, the operator can set the working parameters of the device, control some movements and functions manually, and monitor the work cycles of the unit even before installation. RD6 further includes safety controls of over-temperature, over-current, over-voltage, under-voltage and bus communication failure, and implements the boot-loader feature so the firmware can be upgraded at each new release.

The rotary actuator is designed to drive positioning systems, change-over applications and linear guides.

Typical uses are packaging lines, filling and bottling plants, food processing and pharmaceutical industries, material handling equipment, conveyor systems, wood and metalworking machinery, paper machinery, bending machines, printing machines, mold changers, mobile stops, tool changers, spindle positioning devices, and any equipment where you need to cut machine set-up times and reduce downtime.

The actuator can be complemented by the LDT10 HMI touch panel. The panel is designed to interface, set up, and operate the whole series of RD rotary actuators equipped with an EiA-485 interface: RD1A, RD4, RD5, RD6. The ▷

touch panel allows to control and configure all actuators connected to the network through the recipes stored in its memory. A single command starts the process, which makes change-over operations faster and easier, cutting set-up time and reducing downtime. The LDT10 comes in a 7-inch, 16:9-format LCD display with a resistive touchscreen panel. Its rugged construction complies with NEMA4 and IP65 protection ratings and allows for use in typical industrial environments.

Author

Paolo Giordan
Lika Electronic
paolo.giordan@lika.it
www.lika.biz

This proverb is often credited to Winston Churchill, but even this is not validated. Anyway, users need to interpret CAN market research studies, if they like to get some values from them.

In the early days of CAN, CAN in Automation (CiA) counted the number of annually installed CAN nodes. This was easy to do: Just a few CAN controller manufacturers were asked to provide their sales figures. CiA accumulated them and published only the total number. This was double-checked with the number of sold CAN transceiver chips. In those days, there were just two respectively three suppliers.

But times have changed. Today, there are many chipmakers providing CAN controllers embedded in their micro-controllers. The number of CAN transceiver suppliers is also much higher. Nowadays, the chipmakers do not know anymore, how many CAN interfaces they have sold. Still they may count micro-controllers, but the number of MCUs does not match with the number of CAN modules implemented. Additionally, a significant number of customized ASICs implement the CAN protocol. This is why CiA stopped counting CAN node installations beginning of the millennium.

Still you can estimate the number of annually installed CAN nodes by a simple calculation. There are quite good market figures for produced cars: about 76,86 millions in 2016. High-end cars have about 50 to 100 nodes, and even the very low-end cars comprise five nodes. In average each vehicle is equipped with 12 CAN nodes. Of course, this is a conservative estimation. This results in about 922 millions of CAN nodes used in passenger cars. Using the 80/20 rule (also known as the Pareto principle), the total available market is about 1,15 billion nodes. However, the figure of 20 percent for non-automotive nodes seems fairly high. Let us be conservative and assume that the non-automotive market is just half of this general estimation. Even then, the total number of installed CAN nodes is still about one billion. Double-checking this amount with the sales figures of CAN transceivers, CiA comes nearly to the same result: In 2016, about one billion CAN nodes have been installed. Of course, there is some uncertainty of ±10 percent.

Interpretation is necessary

Any market research not just counting nodes needs some interpretation. Most of the published studies provide figures based on revenue in US dollars or any other currency. But what is counted: The price for the entire electronic control unit (ECU) or device, just the price for the CAN interface hardware with or without the communication-related software. More critical: It seems that some studies are double-counting things. They count the price for the chips and for the board-level products. Any market research study needs to disclose the counting method in detail, so that the reader can interpret correctly the results.

A typical example is the study on in-vehicle networks (IVN) by Markets-and-Markets. The research company has published a global forecast to 2022 about IVN. The study ▷

reports IVN figures by vehicle type, network technology, application, and geography. It identifies and analyzes the market dynamics such as drivers, restraints, opportunities, and industry-specific challenges for the market. It also profiles the key players operating in the market. The demand for IVN in automobiles is expected to increase owing to the increasing vehicle production and rising trend of vehicle electrification.

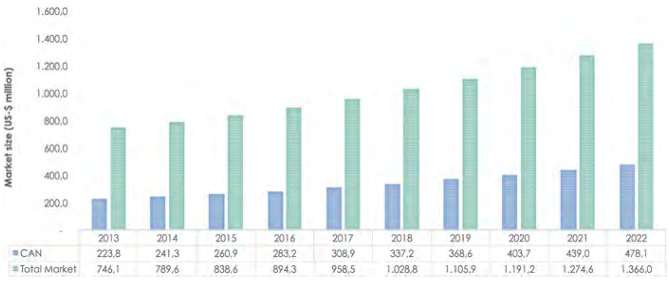


Figure 1: CAN turnover forecast for 2016 to 2022 (Photo: Markets-and-Markets)

The base year considered for the study is 2015, and forecast period is from 2016 to 2022. It values the IVN market at US-$ 836,6 million in 2015 and expects to reach US-$ 1,366 billion by 2022, at a CAGR of 7,14 % between 2016 and 2022. The value comprises just the hardware costs for network controllers and transceivers, said the authors in a telephone conversion. It was not clear, how the price per node for the CAN controllers were evaluated, when they were integrated in a micro-controller: With or without partial costs for the housing.

The market share for CAN as shown in the figure is surprisingly low. On the opposite, the Flexray revenue looks quite high. According to CiA, in 2016, there have been installed about 1 billion CAN nodes. Each CAN port comprises a CAN controller and a CAN transceiver. The prices for these high-volume applications are not publically available. But even with conservative estimations, they sum up to more than US-$ 500 million including the price for enclosures. Just the one billion of CAN transceivers costs more than US-$ 200 million.

The research methodology used to estimate and forecast the in-vehicle networking market began with capturing data on key vendor revenues through secondary research. Some of the secondary sources include associations such as Organisation Internationale des Constructeurs d'Automobiles (OCIA), International Council on Clean Transportation, and International Organization of Motor Vehicle Manufacturers, among others. The vendor offerings have also been taken into consideration to determine the market segmentation. Primary source were interviews with OEMs and automotive suppliers. The Flexray figures given for 2016 are questionable – in particular, because just a few high-end and medium cars implement Flexray nodes today. The only explanation is that the costs for Flexray controllers are more than ten-times higher as for CAN.

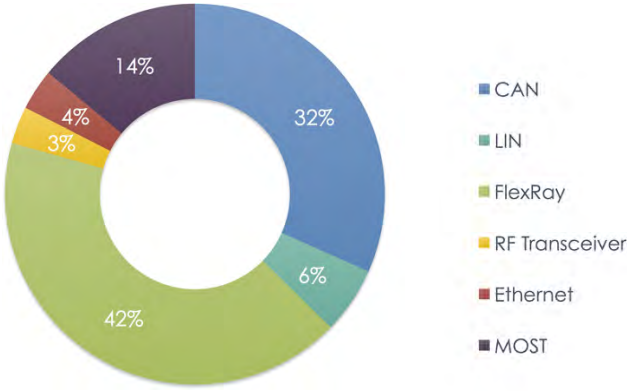


Figure 2: CAN market shares in IVN in 2016 (Photo: Markets-and-Markets)

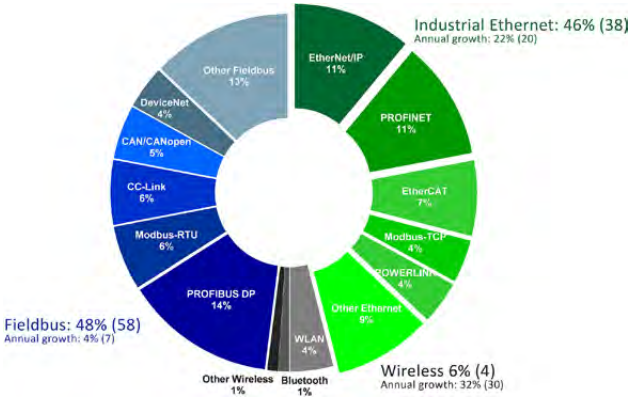


Figure 3: The shown figures are related to factory automation and not industrial electronics (Photo: HMS)

Another Market-and-Market study on data busses estimated the revenue in 2021 to about US-$ 19,5 billion. One year before, the researchers predicted the revenue in 2020 to just US-$ 8,5 billion. Could be that prices have increased within one year. Just counting the number of node respectively interfaces, makes market research studies more comparable.

Transparency Market Research (TMR) has also released an in-vehicle network survey. TMR estimated the revenue on transceiver chips from 2016 to 2025. Butt no detailed results have been given to the editors of the CAN Newsletter.

Studies on dedicated regional markets should be questioned, too. What has been considered: the micro-controller with CAN on-chip produced in Malaysia, the ECU produced in USA, installed in a car made in Germany, and sold in Egypt. The risk of double counting and comparing of apples to oranges are very high. Just count the number of interfaces on the chip level and you avoid double counting.

Absolute Reports has published recently the "United States CAN Market Report 2017". This 99-pages report provides expected revenues, annual growth rates, etc. for different CAN markets including automotive electronics, industrial control, and healthcare. The companies interviewed include some market-leading chipmakers, but not all of them. Surprisingly, two board-level manufacturers have also been consulted: ESD (Germany) and National Instruments (USA). Both are CiA members. Detailed results are not available publically. The study covers the time from 2012 to 2022 (forecast). For all three evaluated markets, a further increase is predicted.

Spotlights on dedicated markets

Detailed CAN market figures for non-automotive markets do not exist with some exceptions. You can estimate from some market-leading companies to the overall market of a specific application field. An example: Bromma, a brand of Cargotec, has a market share of about 70 percent in spreaders used for cranes. The spreaders are equipped with embedded CAN networks. Just count the spreaders sold by the Swedish company (2000 per year) multiplied with the average number of CAN nodes (six to ten) used in the embedded network and you have a figure for this market. You may add a seventh for competitors also implementing CAN-based networks in their products. This should be done for each sub-market in which CAN networks are used. This is a Sisyphean task. So-to-say, it is a never-ending story, because CAN is implemented in countless applications.

HMS (Sweden) has published recently figures on market shares of industrial networks. The good news: The market share figures are based on number of nodes, not on revenue. In a very first glance, it looks like CAN would play a minor role in industrial networking (just 5 % for CAN/CANopen and 4 % for Devicenet). But the study covers according to the authors only the factory automation market in which CAN is really not one of the mainstream communication technologies. The study doesn't cover embedded machine control networks, the high-volume application in industrial automation. It also doesn't include the captive factory automation applications – in those the products are made by the OEMs. The captive markets are normally not visible and hard to count.

Summary

Trust only statistics that you have falsified by yourself. Counting just the absolute numbers of the total available market (TAM) is not statistics. When you are evaluating sub-markets, you need to specify clearly the scope. Most of the market research studies are questionable. Nevertheless, some of them may contain valuable figures. But the price is normally too high. It is always good to ask, who initiated the study. Often one customer has requested a market research study. To share the price with others, those reports are made public. Of course, the initiator will never be blamed or fall into bad light. He should be always satisfied with research results. ◀

Author

Holger Zeltwanger
CAN Newsletter
headquarters@can-cia.org
www.can-cia.org

The Ultimate CAN FD Tool

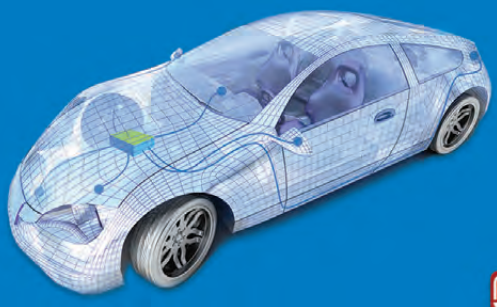
neoVI FIRE 2

Vehicle Interface & Data Logger

Standalone gateway functionality between DoIP, CCP/XCP, CAN FD, CAN, LIN, and Automotive Ethernet.

Device includes:

- 8x ISO CAN FD
- 4x LIN
- Ethernet: DoIP/XCP
- Hardware Cybersecurity
- Standalone Logging, Scripting, & Simulation
- Full Vehicle Spy Software Support



Find out more at www.intrepidcs.com

INTREPID CONTROL SYSTEMS GMBH

USA Germany Japan Korea China India Australia

+49 (0)721 6633703 -4 icsgermany@intrepidcs.com

CAN in Automation

The non-profit CiA organization promotes CAN and CAN FD, develops CAN FD recommendations and CANopen specifications, and supports other CAN-based higher-layer protocols.

Join the community!

- Initiate and influence CiA specifications
- Receive information on new CAN technology and market trends
- Have access to all CiA technical documents also in work draft status
- Participate in joint marketing activities
- Exchange knowledge and experience with other CiA members
- Get the CANopen vendor-ID free-of-charge
- Get credits on CANopen product certifications
- Get credits on CiA training and education events
- Benefit from social networking with other CiA members
- Get credits on advertisements in CiA publications

For more details please contact CiA office at headquarters@can-cia.org

www.can-cia.org