# 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.

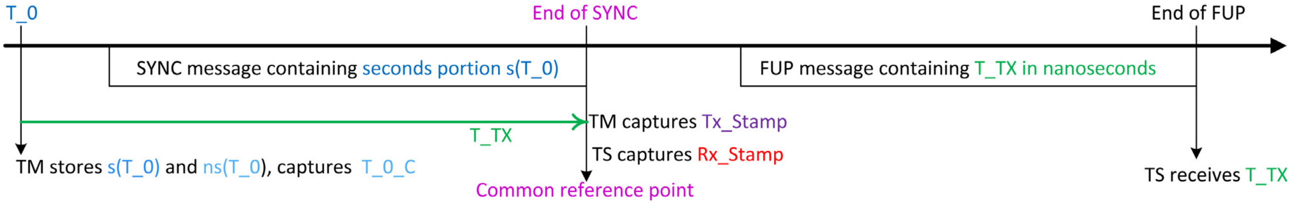

T_0
End of SYNC
End of FUP
SYNC message containing seconds portion s(T_0)
FUP message containing T_TX in nanoseconds
TM captures Tx_Stamp
T_TX
TS captures Rx_Stamp
TM stores s(T_0) and ns(T_0), captures T_0_C
Common reference point
TS receives T_TX

*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 ▷

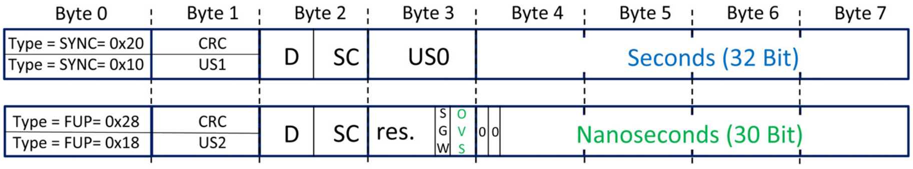


*Figure 2: Autosar CAN synchronization messages (Photo: Bosch)*

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.

▷

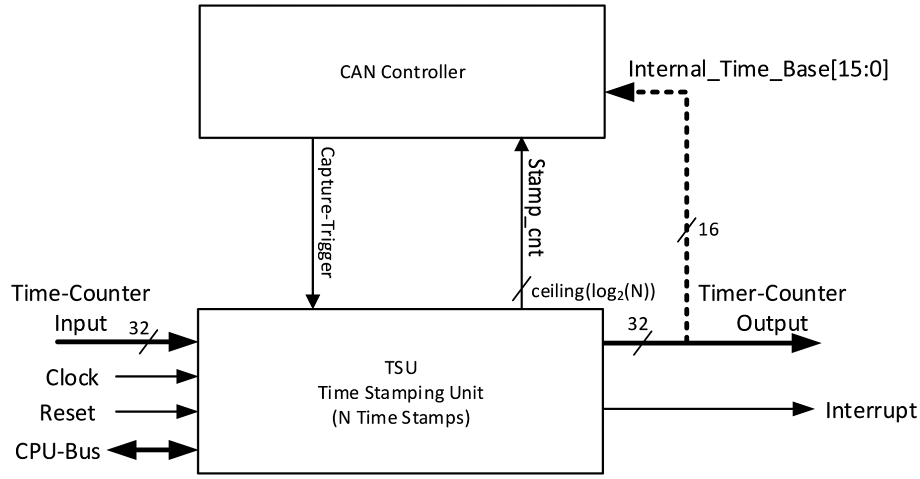


*Figure 3: A separate TSU simplifies the CAN controller re-design, when the time-stamping is added (Photo: Bosch)*

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 $\mu$s; 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.

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

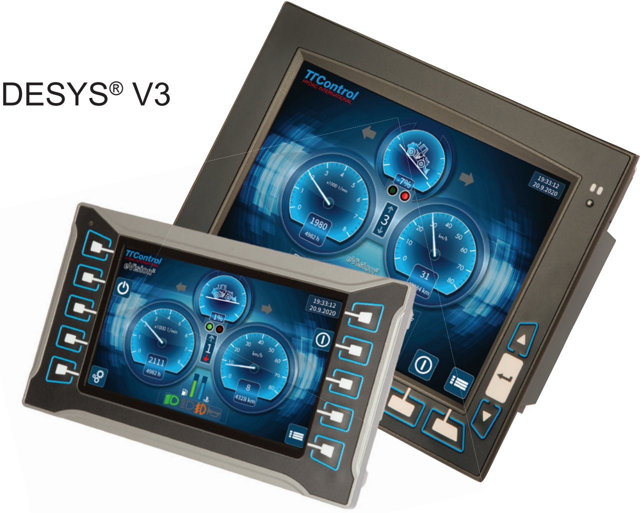
## Performance

- 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

HY-eVision$^2$ Family



www.ttcontrol.com/HY-eVision2-Family

Safety
Certified
ECUs

General
Purpose
ECUs

I/O
Modules

Safe I/O
Modules

Operator
Interfaces