CAN Send and Receive with Hardware Timestamping

Hauke Webermann, esd electronic system design gmbh

Modern CAN controllers, especially for FPGA integration, keep growing stronger and more versatile. Also the issues in Industrial Automation are increasingly complex. As a result the number of requirements, in particular for high precision timestamps, grows as well. The required accuracy can only be obtained through a hardware based solution. For this reason, the CAN controller has to be extended with a 64 bit timestamp counter and every received CAN frame gets a timestamp at the receiving time. In complex systems such as test benches or airplanes, global time networks are necessary. Via an external source (e.g. IRIG-B) a timestamp synchronisation of all CAN controllers and other nodes in the whole system is possible. Through the availability of an internal timestamp high precision CAN frame transmissions will be facilitated. The TX-FIFO of the CAN controller is expanded with a 64 Bit timestamp which contains the sending time. This enables approximate real-time behaviour in non-real-time operating systems. Additionally a precise feedback containing the transmission time of any given CAN frame is available. This is particular used by higher level protocols like ARINC825.

Introduction

In this paper, the possibility for hardware based timestamps in CAN controllers for high precision requirements and use cases in higher level protocols will be discussed

The exact time is becoming increasingly important for applications using CAN. Therefore, there is the need to break down this temporal accuracy to every CAN frame. For applications that do not have high requirements for accuracy, it may be sufficient to add a timestamp to the CAN frame in the interrupt service routine. If there are higher requirements for the accuracy or for less jitter, a hardware-based timestamp is indispensable. When the CAN frame is received, a timestamp is attached to the CAN frame directly in the CAN controller.

In addition, this paper also discusses further advantages and new applications for hardware Timestamps. For the timestamped TX technology, the TX-FIFO of the CAN controller is expanded with a timestamp which contains the sending time. This enables approximate real-time behaviour on non-real-time operating systems. Additionally there is available a precise feedback containing the transmission time of any given CAN frame.

Timestamps

The timestamp is captured the moment a CAN message was received or a CAN event occurred. Depending on the device capabilities the time stamping is performed either in hardware by the CAN controller or in software by the driver's interrupt handler using a high resolution counter of the host CPU.

For a software timestamp, the timestamp is added to the CAN message in the interrupt service routine. Depending on the reactivity of the host system, this can lead to the following problem. If several interrupts are present at the same time, the sequence of different events can no longer be distinguished. For example, if a receive interrupt and a transmit done interrupt or a CAN error interrupt are present at the same time, the implementation of the interrupt service routine decides in which order they are copied into the RX FIFO of the CAN driver. An additionally jitter will be added to the timestamp of the CAN message depending on the reaction time of the host system. This jitter depends on different factors, so it is difficult to calculate this afterwards.

If these and other imprecisions concerning accuracy and jitter are not acceptable, a

hardware based timestamp must be used. But first, it should be considered which properties this should have. The first point must be which timestamp frequency is needed to be able to map the order on the CAN bus by using the timestamp.

The shortest CAN frame has a length of 44 bits (ID: 0x111, DLC: 0x12, Data: none, Stuff bits: none). With additional interframe space, frames can be received within a distance of 47 µs. The minimum requirement at 1 Mbit/s would therefore be a timestamp with a frequency of 21.3 kHz, to store each frame with a new timestamp.

In order to be able to depict CAN errors exact in time and map them in correct order, a bit time exact timestamp is necessary. This results in a new minimum frequency of 1 MHz.

Now, the time quantum is the smallest unit that can occur with CAN and by resync and hard sync events, frames are shifted in the bit timing by this. This results in the fact that the required Timestamp frequency must be the CAN controller frequency. In order to be able to map all events and messages on the CAN bus exactly via the timestamp, this or a higher frequency should be selected.

The second question is the width of the timestamp. Various widths have already been used in the past. If the selected timestamp width is too small, the driver and all further applications have to cope with the fact that the timestamp can wrap and restart counting from zero. This can happen very quickly with modern CAN controllers.

Table 1: Maximum time count of different Timestamp widths at e.g. 80 MHz

TS Width	Time in Sec	Time in Years
16 Bit	0.8192 ms	2.6×10^{-11}
32 Bit	53.7	1.7×10^{-6}
64 Bit	2.306×10^{11}	7312

The table shows that even a 32 bit wide timestamp in a human readable time resolution is not sufficient for CAN applications, since it wraps every 54 seconds. In the CAN driver every incoming timestamp would have to be recalculated to an absolute time. In order to avoid these problems, a 64 bit wide Timestamp is recommended because it will not wrap in the foreseeable future. In addition, the 64 bit wide timestamp can be considered as an absolute time. Without this property, the applications described below would not be possible in some cases.

Timestamps with higher resolution also mean increased hardware requirements. In consequence more logic and memory resources are needed. But this should not be a big problem with modern FPGA based CAN interfaces. Since even the smallest FPGA types already offer considerable possibilities and logic resources.

Using a 32 bit timestamp has another major drawback. Each timestamp must be converted to an absolute time in the host system. As a result, a long word can be saved during data transmission to the host. But there are extensive 64 bit operations which cost a lot of time in the kernel or are not even available in all embedded systems.

CAN controllers can run at different frequencies, so the timestamp has no default resolution and offers more flexibility. The timestamp is realized as 64 bit free running counter with the most accurate available timestamping source. The application can query the frequency of the time stamping source in order to scale the timestamps online or offline and can query the current timestamp to link them to the absolute system time.

Classically, the timestamp is recorded at the end of the CAN message. This causes an unwanted jitter, by the length of the frame and according to the stuff bits. If a CAN record is to be replayed again, this is distorted by the jitter. Therefore, another advantage of a hardware based timestamp is that the timestamp can also be taken at the Start of Frame (SOF).

In order to keep independent systems synchronized, it should be possible to feed an external timestamp. Various concepts can be presented here, for example

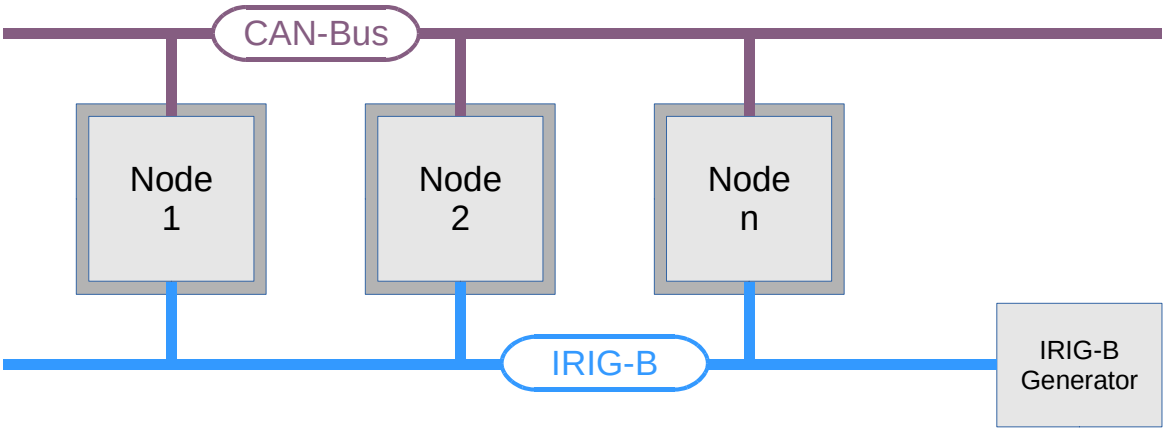

Figure 1: Example configuration for a System with IRIG-B Time Synchronization

IRIG-B [3], in which the time information can be transmitted analogue or digitally by a separate link. Among other, this time synchronisation is popular in the aviation industry. With this extension all devices can be operated with the same time base.

In Figure 1 is shown an example for a System with several nodes that are connected via a CAN bus. For time synchronization the nodes are connected by an additional link which is provided by an IRIG-B generator.

Timestamped TX

Through the availability of an internal timestamp high precision CAN frame transmissions will be facilitated. The idea is to add a timestamp to the CAN Frame with the time information when it should be sent. The TX-FIFO of the CAN controller is expanded with a 64 Bit timestamp which contains the sending time. This enables approximate real-time behaviour on non-real-time operating systems. CAN hardware supporting this mechanism reaches an accuracy of one bit time, when sending onto a free CAN bus.

In normal case a CAN frame is sent with “canSend” from application level to the CAN driver. The frame is added to the TX queue and if there is a free Element in the TX FIFO in Hardware, the CAN driver moves this Frame into the TX FIFO.

In case of the new Timestamped TX the CAN frames are transferred to second TX queue in the CAN driver. This queue is parallel to the normal TX queue. This creates a second channel for transferring CAN frames in the CAN controller. In this queue, the CAN frames are at first collected and sorted. The sorting is arranged chronologically according to the transmitted transmission times. CAN frames with the same timestamp remain in the order as they were passed to the CAN driver. Subsequent jobs with the same timestamp are sorted down as follows.

At cyclic intervals, the driver moves CAN frames from the TX queue into a TX window. No reordering is possible from this point onwards. The size or duration of this transmission window depends on the used hardware (CAN card, host architecture, host CPU), the used data bus (e.g. USB, PCI Express) and maybe the used operating system. Typically this value is approximately 100 ms before the actual planned time of transmission. In this time the CAN driver will move this CAN frames into the so called TX-TS window.

In the next step the CAN driver will move the CAN Frames from this window into a special TX-TS FIFO of the CAN controller. This is parallel to the existing TX FIFO. The size of this TX FIFO has to be that a "back-to-back" transmission of scheduled frames can be ensured. Thus there is only a subset of the frames of the TX-TS window in the TX-TS FIFO. In this case, the CAN frames can no longer be resorted in their ordering.

As soon as the transmission object of the CAN controller is empty, a new CAN frame

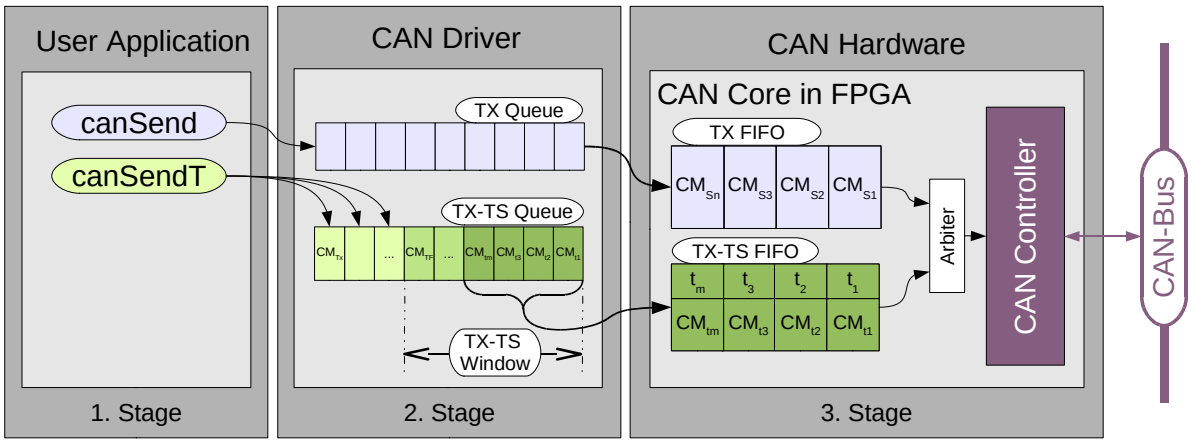

Figure 2: Timestamped TX structure, from User Application to CAN Hardware

is obtained from the FIFOs. An arbiter will prioritize CAN frames from the TX-TS path which are ready to send over frames from the normal TX path. Alternatively, other arbitration modes could also be selected, such as CAN priority or round robin.

Figure 2 describes the structure of the Timestamped TX concept parallel to a standard CAN FIFO implementation. The CAN frames are sent via the API functions to the CAN driver. In normal case, the CAN frame is added ("canSend") to the TX Queue. The CAN driver checks the TX FIFO in FPGA. If there is space for new messages the driver copies the data in the FIFO otherwise it waits for completed jobs.

Frames to be sent with "canSendT" are added to the TX-TS queue in order of the attached timestamp. In the area of the TX-TS Window CAN frames are moved into the TX-TS FIFO if there is space for new elements the way it was already described.
Now the arbiter decides from which FIFO the next CAN frame is to be sent. If the timestamp of the scheduled frame is expired, this frame will be the next CAN frame to be sent. Otherwise the next frame of the TX FIFO will be sent on the free bus.

If the configured timeout for the CAN frame from the TX-TS FIFO has expired, this frame is discarded at this moment. A precalculation does not take place if a timeout will occur during the transmission. Similarly, it could happen that a frame is taken from the normal TX FIFO and even if the CAN controller is still transmitting, the transmit time for the next scheduled frame from TX-TS FIFO is reached.

In figure 3 the timeline of an example of CAN messages is shown. At time t_1, the CAN bus is free and the arbiter takes the CAN message CM_{t1} from the TX-TS FIFO and sends this as soon as possible to the next bit time. The resulting delay is indicated by t_D. The frame is directly followed by a frame CM_{S2} from the TX FIFO. At the time of transmission from CM_{t2} to time t_2, the CAN controller is busy and sends CM_{S3}. Thus, the scheduled CAN frame CM_{t2} is delayed, but this Frame is sent directly in the following of CM_{S3}. At time t_3 the frame CM_{t3} has been

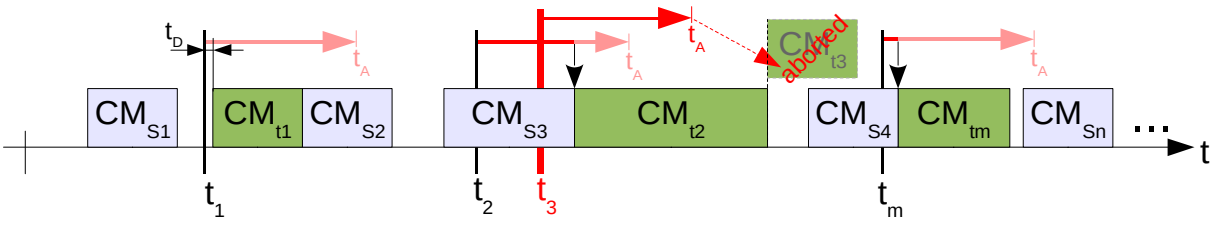

Figure 3: CAN message send timeline

scheduled. With the configured abort timeout t_A, the CAN frame could not be sent and is aborted after CM_{t2}. The CAN driver and the user application are informed by an event message.

Timestamped TX is fully compliant with [1] and does not break any CAN rules of transmission. A scheduled frame will only be transmitted in time, if the bus is idle and the CAN priority (CAN-ID) qualifies for transmission at that certain moment. Otherwise the transmission is delayed until correct conditions are given or the transmission is aborted (e.g. by timeout).

Use Cases

Hardware based timestamps can be used in many areas. In more and more industrial sectors, there are increasing demands on accuracy, for example in aircraft or test benches. Validations can only be carried out meaningfully if the corresponding tools also have the required accuracy.

The use cases shown here are only an example for the industry areas and applications in which hardware based timestamps can be used.

Use Case 1:
Due to the high resolution timestamps, highly accurate CAN logging is possible. This can be used for documentation, but also for further offline error search. A bit-accurate image of the CAN bus can be analysed afterwards. The combination of the hardware based timestamp in send and receive direction makes it possible to replay this exact CAN log again.

Use Case 2:
In the area of residual bus simulation a higher accuracy is possible. The scenarios can be calculated in exact CAN bit time. For the validation of the system under test the CAN frames with hardware timestamp give an exact image of the CAN bus and can also be evaluated with bit time accurately.

Use Case 3:
In some systems, an exact sync signal is required. This can be ensured with the Timestamped TX mode. Herewith can be programmed a precise sync generation.

Use Case 4:
Other higher level protocols are also possible, for example the ARINC 825 [2] protocol. The available CAN bus time is divided into several time slices and in this time slices CAN frames can be sent cyclically. It must be ensured that the CAN frames are only sent within the time slice. CAN Frames that cannot be sent in case of a busy CAN bus or an overloaded time slice must be aborted.

Use Case 5:
A welcome side-effect of the Timestamped TX is an integrated high priority TX FIFO, which lets CAN frames bypass the queued TX jobs. Many users feel most comfortable while working in FIFO mode. Hereby it is possible to transmit emergency CAN frames deterministically with as little delay as such frames deserve, yet without changing the preferred programming paradigm. The Timestamped TX is used in the same way as described before, but instead of using timestamps in the future it is used a timestamps in the past. The frames will be transmitted as soon as possible with the advantage of having a higher priority.

Use Case 6:
A hardware based timestamp results in a smaller jitter in comparison to a software timestamp. With this advantage a timestamp synchronisation via CAN messages in a CAN network (CAN-To-CAN Synchronisation) results in a higher accuracy. For this application it is necessary to reduce the interrupt latency to get the best results.

Summary

The high precision hardware based timestamps can help to improve the accuracy of complex systems in areas where CAN is used. This may be in aerospace, automotive, medical or general industrial automation.

Hardware timestamps usually result in higher accuracy compared to software timestamps as the jitter does not depend

on real-time capabilities or the CPU load of the host system. Hardware based timestamps with 64 bit width and CAN controller frequency are the best choice for a high precision timestamp resolution. In order to reduce the jitter even further, timestamps can be recorded on the Start of Frame (SOF) instead of the end.

A 64 bit hardware based timestamp needs more hardware resources in the CAN controller, but offers an absolute time in hardware and so a variety of advantages. In addition the TX-FIFO of the CAN controller is expanded with a 64 Bit timestamp which contains the sending time. This enables approximate real-time behaviour in non-real-time operating systems.

Hauke Webermann
esd electronic system design gmbh
Vahrenwalder Str. 207
D-30165 Hannover
+49-511-37298-0
+49-511-37298-68
hauke.webermann@esd.eu
https://esd.eu

References

[1] ISO 11898-1: Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signaling, December 2015

[2] ARINC Specification 825-3: General Standardization of CAN (Controller area network) Bus Protocol for Airborne use, July 2015

[3] IRIG Standard 200-04: IRIG Serial Time Code Formats, September 2004

[4] esd electronic system design gmbh: NTCAN Part 1: Application Developers Manual, 11. November 2016

[5] esd electronic system design gmbh: ARINC 825 Library Software Manual, 23. November 2015

[6] esd electronic system design gmbh: CPCI-CAN/400. 4x CAN with ARINC Protocol and IRIG-B, 11. November 2015

[7] Voss, Wilfried: A comprehensible guide to Controller area network: Copperhill Technologies Corporation, 2005