Embedded World 2004
Nürnberg, 17.–19.02.2004
pages 235–252

Comparison of Event-Triggered and Time-Triggered Concepts with Regard to Distributed Control Systems

Amos Albert
Robert Bosch GmbH
Corporate Research and Development
Robert-Bosch-Str. 2
71701 Schwieberdingen
Tel. ++49(0)711/811-43607
amos.albert@de.bosch.com

keywords: distributed control systems, event-triggered, time-triggered, real-time performance, latency, jitter, delay compensation, bus communication, scheduling, CAN, TTCAN

Abstract

This paper compares event-triggered and time-triggered concepts from the control theory point of view. For that purpose, the requirements of distributed control systems are summarized and advantages and disadvantages of both concepts are elaborated by hand of different criteria. These are, for instance, the ability to react to asynchronous events, the system design and scheduling, the synchronization problem, and the compensation of latency and jitter in the control loop. The results are highlighted with the aid of theoretical examinations and real measurements based on the event-triggered CAN and its time-triggered version TTCAN.

1 Introduction

The event-triggered CAN bus [CAN90] has established itself as a de facto standard for today's chassis control systems and power train communication. Recently, the demand for time-triggered communication has intensified and it is quite evident that busses with time-triggered operation modes will play a major role in future automotive networks. Therefore, it is necessary to investigate the impacts of such busses.

Within a time-triggered communication the permission to access the bus is controlled by predefined time windows (TDMA, time division multiple access), leading to the advantage of a quasi-deterministic behavior during regular operation. Therefore, time-triggered concepts potentially provide a higher dependability, since e.g. missing messages are immediately detected. Other important properties are the possibility to guard the bus against non authorized bus accesses (bus guardian) and to realize synchronously working busses in order to take care for redundancy (fault-tolerant systems) [Kop97]. A very interesting property from the point of view of the automotive field concerns the so called composability. Since the time windows to access the bus are predefined, the behavior along the timeline is decoupled from the actual bus load. Thus, it is possible to develop different subsystems independently (e.g. by the car manufacturers and suppliers), to exactly simulate the final time behavior of the subsystems and subsequently to integrate the subsystems into the complete system.

Although time-triggered concepts initially have been demanded by X-by-wire systems and their necessity for fault-tolerance, they are now also of interest for other applications. For instance, the (restrictive) framework of time-triggered architectures enables the system designers to disentangle the high complexity of today's large scale systems, making the system more controllable due to more determinism.

One drawback of time-triggered bus concepts is the lack on flexibility and the restrictive design process. All processes and their time specifications must be known in advance. Otherwise, an efficient implementation is not possible. Furthermore, the communication and the task scheduling

on the control units have to be synchronized during operation in order to ensure the strict timing specifications of the system design. Since the different oscillators jitter, the subsystems would otherwise run out of phase.
The main advantage of event-triggered systems is their ability to fastly react to asynchronous external events which are not known in advance [AG03]. Thus, they show a better real-time performance in comparison with time-triggered systems. In addition, event-triggered systems possess a higher flexibility and allow in many cases the adaptation to the actual demand without a redesign of the complete system.

In general, reality is neither black nor white but rather gray. Thus, it depends on the application whether a time-triggered or an event-triggered behavior is more suitable. Some bus concepts try to combine the advantages of both concepts (event- and time-triggered) as, for instance, TTCAN [LH02, FMD+00] and FlexRay [BBE+02]. It is not within the scope of this paper to intensively compare the different properties of event- and time-triggered systems. Works which address such a comparison are, for instance, [Kop00, APF02]. The comparison here is carried out merely from the control theory point of view.
The remaining part of this paper is organized as follows:
Section 2 summarizes the requirements of distributed control systems. Section 3 deals with the ability to react to asynchronous external events and section 4 is concerned with the behavior during regular operation in distributed control systems. Further, the origin of latency and jitter of event- and time-triggered concepts are elaborated. Finally, section 5 gives some annotations concerning the design process. The paper ends with a summary and a short discussion in section 6.

2 Requirements of distributed control systems

In modern upper class vehicles up to 2500 signals are exchanged by up to 70 ECUs, making nowadays cars highly distributed control systems. In order to reduce costs for cabling the nodes of the distributed control system are connected to each other via a network. Network systems offer the opportunity of modularity and scalability which allow a flexible design of variants. Further, (intelligent) sensors and actuators are used in a multiple way to realize high level functions and services.
There is a huge number of possible network topologies, like for instance, the currently frequently used bus communication, topologies with gateways, active stars or cascaded star structures. If the application requires fault-tolerance, the classic redundant network or a mixed redundant network which efficiently maps the system safety structure into a system architecture is utilized [MFH+02]. To simplify matters, this paper merely considers the bus topology[1].
Unfortunately, communication networks introduce delays. Furthermore, these delays may be varying in a random fashion, making the control system time varying [Hus97, Nil98].

If one is concerned to formulate the requirements of distributed control systems two different situations should be distinguished:

1.) Occurrence of a critical situation, like for instance, the excess of a critical temperature and
2.) the regular operation of the control system.

The former situation requires an as fast as possible reaction to the asynchronous event in order to start emergency mechanisms. For that purpose, section 3 compares event- and time-triggered communication by hand of their ability to react to asynchronous events. The 'frequency response' of the envisaged communication system is evaluated by a measuring method based on an orthogonal Walsh correlation. The measurement yields the average latency and the jitter when reacting to the event.

[1] In order to verify some statements of this paper, measurements based on the CAN bus have been carried out. Due to its arbitration mechanism, the only appropriate bus topology for CAN is the bus structure.

The requirements for the second situation arise from control theory. Since the control is implemented on digital computers, the most important requirements can be summarized as follows:

- The control design in general is carried out on the assumption that the sampling periods are constant and that the control algorithm is executed with perfect periodicity. Otherwise, a time varying algorithm may become necessary if one does not want to risk the degradation of the control performance or even instability of the system. Such a time varying algorithm presupposes that time information are available, like for instance, the actual sampling period, the instants at which the measurement and the realization of the control variable are carried out or the time demand for the control algorithm.
- Ideally, there is neither sensor delay nor actuator delay. The former denotes a delay between the measurement of the controlled variable and the reading of the input by the controller. The actuator delay denotes the latency between the writing of the output by the controller and the actual realization of the control variable at the plant. If latency is inevitable it should be at least constant in order to minimize the jitter and to use a time invariant control law [FPW97]. For the compensation of sensor delays generally observers are used which deliver the state variables at the interesting time instants. The original control algorithm can be used without modifications. For the compensation of actuator delays a new design of the controller becomes necessary.
- The Realization of small delays is always advantageous, since the overall control performance will degrade with increasing delays, no matter whether the delay is compensated or not. For instance, in the meantime occurring disturbances are not observed and hence not considered by the algorithm. Furthermore, also the exact compensation of a known and constant actuator latency leads to a delay between the real and the desired tracking of the controlled variable (see section 4.4).

The target of this paper is to illustrate the pro/cons of event- and time-triggered concepts from the control point of view. Illustrative examples are carried out here by hand of the well known CAN and its time triggered version TTCAN which has been specified by the International Standardization Organization in ISO 11898-4 [Org]. Silicon implementations of the protocol are available, e.g. from Bosch [Har02] or Infineon [LKK03]. Due to its non-destructive arbitration mechanism the bandwidth of a CAN network is limited by its cable length. The maximum data rate for a 40m network is 1Mbit/s which is not a technological but a physical limitation. This limitation also holds true for TTCAN, since on the base level still the CAN arbitration is carried out. For future applications with higher needs the FlexRay [BBE+02] standard is expected to play a major role.

3 Reaction to asynchronous external events

As explained in section 2, one of the elementary requirements of all real-time systems is the ability to react to an asynchronous event within a predefined period of time. In order to evaluate the quality of the system, frequently the average response time (latency) and its jitter is considered. Obviously, for such a comparison event-triggered busses have an advantage over time-triggered busses. Thus, the following examination should also show qualitatively and quantitatively which price one has to pay for the higher determinism of time-triggered concepts in terms of the ability to react to asynchronous events.

3.1 Test scenario

For the following examination the cooperative communication between two control units A and B is considered as depicted in figure 1. The inspected test scenario looks as follows:
A critical situation occurs at an arbitrary instant of time and corresponding sensor signals reach control unit A (in the following ECU A). Now ECU A informs another control unit B (in the following ECU B) about the critical situation and waits for its reply. Thus in total the cycle A → B → A is examined. Figure 2 illustrates the explained scenario along the timeline.

Embedded World 2004
Nürnberg, 17.–19.02.2004
pages 235–252

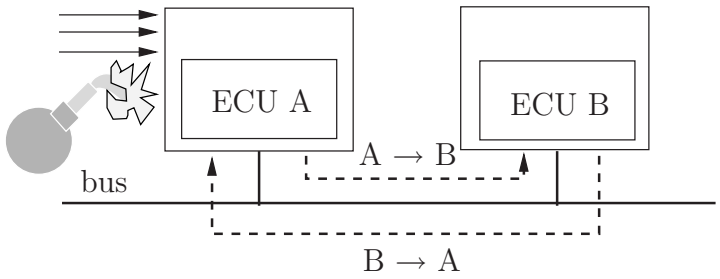
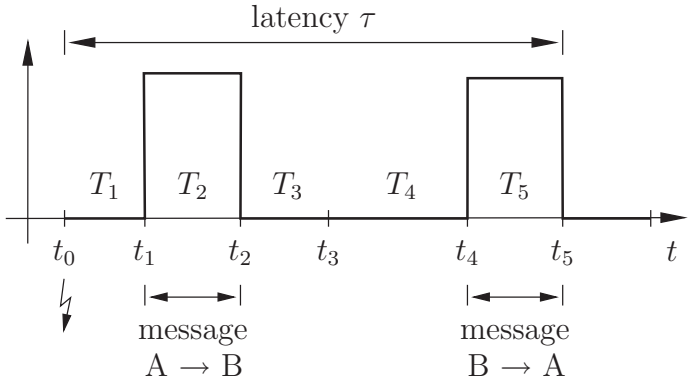


Figure 1: System configuration with two control units ECU A and ECU B



t_0: appearance of critical situation
t_1: transmission of message by ECU A
t_2: reception of message by ECU B
processing of information at ECU B
t_3: transmission request of ECU B
t_4: start of response by ECU B
t_5: reception of message by ECU A

Figure 2: Test scenario along the timeline and definition of the latency τ

At $t = t_0$ the critical situation occurs and is immediately available to ECU A. The transmission of the corresponding message to ECU B can take place not until t_1. Reasons for this latency $T_1 = t_1 - t_0$ are manifold. On the one hand there is a time demand for the computation at ECU A. On the other hand one has to wait for the permission to access the bus. Subsection 3.2 will explain the different compositions of the latency T_1 for event- and time-triggered systems. The duration T_2 is the transmission time on the bus which depends on the data rate and the length of the message. The reception of the message by ECU B is finished at t_2. The information processing takes place until t_3. Afterwards, a response for A is required. The permission to access the bus is received at t_4. The scenario ends at t_5 when ECU A receives the response from ECU B.
Since external (environmental) events occur asynchronously, the time at which the critical situation appears is not known in advance. Therefore, T_1 and T_4 are quite susceptible for jitter.

3.2 Origin of jitter

As shown above the overall latency τ is composed of the latencies $T_1, \ldots, T_5$. Particularly, the latencies T_1 and T_4 depend on the bus concept. Therefore, a more detailed examination is carried out for T_1 and T_4.
Figure 3 illustrates qualitatively the behavior along the timeline for an event-triggered and a time-triggered bus concept. The upper part shows a situation for CAN, where the critical situation occurs at an arbitrary instant of time. Particularly, the bus can be occupied if a transmission is currently in progress. Then ECU A has to wait for the next arbitration and receives in the best case the permission to the bus in this next arbitration. The cycle A → B → A then starts. For this situation to happen it is assumed that the message of ECU A possesses the highest priority compared to all other messages during the arbitration. Summarizing, the following influences are of importance:

- *bus work load and message priority*
- *maximum length of message and data rate*

The lower part of figure 3 shows the qualitative behavior of a time-triggered bus when reacting to an asynchronous event. For a time-triggered architecture the time instant at which the message is transmitted in the cycle is well defined. In the worst case, after the occurrence of the critical

Embedded World 2004
Nürnberg, 17.–19.02.2004
pages 235–252

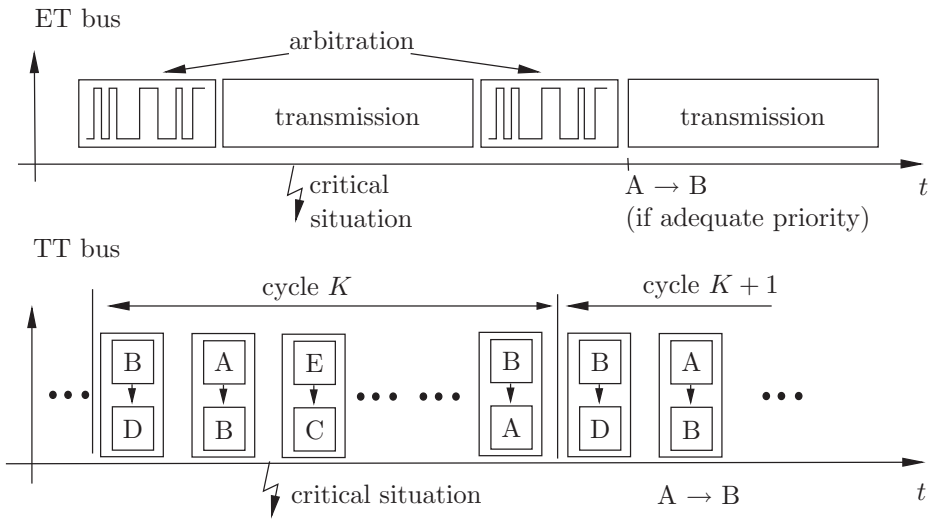


Figure 3: Qualitative response along the timeline for an event-triggered (in this example shown for the CAN bus) and a time-triggered bus

situation one has to wait an entire cycle, if the respective time slot has just passed. After this idle time it is guaranteed that the transmission will take place. Therefore, for the inspected scenario at least a guaranteed upper bound can be given. Summarizing, the following influences are of importance:

- *cycle structure, cycle time*
- *position and counts within cycle*
- *data rate*

As already mentioned, TTCAN and FlexRay provide also some limited event-triggered properties. For instance, TTCAN also allows the definition of free arbitration windows and FlexRay provides a scalable dynamic part of the communication cycle. These features are not considered here on purpose in order to assure a clear comparison of event-triggered and time-triggered bus concepts. If one is concerned to evaluate the real-time performance on the basis of the shown scenario, usually three questions arise:

1.) Does the system react to all critical situations?
2.) Of what magnitude is the average delay τ of the system?
3.) How reliable is the system's response? I.e., of what magnitude is the jitter?

To all three questions, the 'Distinctness of Reaction' (DoR) is able to give a quantitative answer [Wol02, WAG03, AWG03]. The method is based on an orthogonal Walsh correlation and yields a reliability measure given by the average latency response time and the jitter when reacting to asynchronous external events. The measurement of the DoR originally was developed for the evaluation and comparison of different real-time operating systems. As presented in [AWG03] it is possible also to evaluate communication systems by this method. Due to lack of space a detailed description of the measuring procedure is not possible in the written paper. More information can be found in the literature, e.g. [WAG03, AWG03].
In order to measure the DoR, the communication system is excited by a square wave signal $i(t)$ with an adjustable frequency as shown in figure 4. This excitation simulates the occurrence of the critical situation. After the described cycle A → B → A the system reacts in a predefined manner with its response $x(t)$. The signals $i(t)$ and $x(t)$ are processed by a digital circuit, implemented on a CPLD (Complex Programmable Logic Device) which allows to quantify the DoR. The result is delivered via a serial interface. The DoR is a measure for the jitter in the system's response and takes on values from 100% (no jitter) to 0% (at least sporadic loss of excitations).
Not only the determination of a solely value is carried out (constant frequency of the excitation $i(t)$), but the recording of an entire frequency response. A frequency response is known to consist of an amplitude response and a phase response. The DoR determines the amplitude response. The

Embedded World 2004
Nürnberg, 17.–19.02.2004
pages 235–252

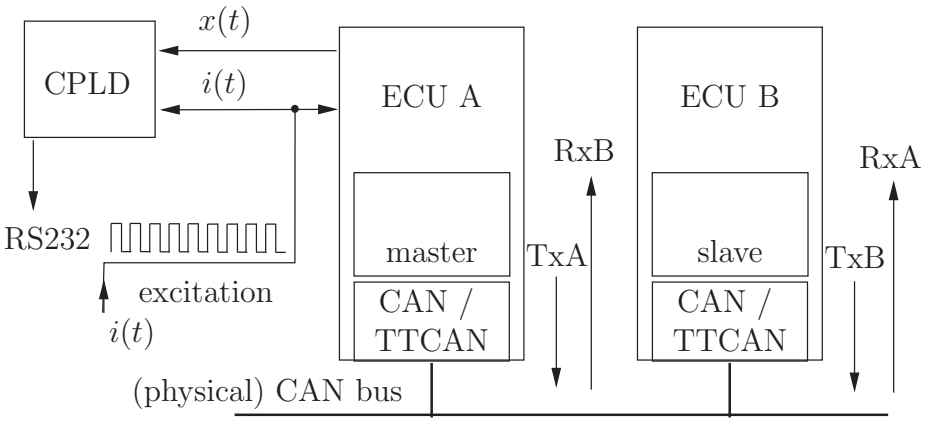


Figure 4: System configuration for the evaluation of the ability to react to asynchronous events

phase response is determined by the average response time τ which is a direct measure of the average latency of the system's response. In order to achieve a standard of comparison a normalization is carried out and the 'average skew' $s = -\tau/T$ is introduced. The skew s is scaled downwards from 0% to -100%. This choice provides the advantage that it allows to evaluate the system's quality from the plot by the simple rule 'the higher, the better' which holds true in the same manner also for the comparison by hand of the DoR plots.

3.3 Results

This article compares CAN as an example for an event-triggered bus protocol with TTCAN which serves as an example for a time-triggered concept. The basic differences where explained in section 3.2 and should now be verified with the aid of real measurements.
Figure 5 illustrates CAN results for different loads and also includes one result for TTCAN. For all measurements the data rate is fixed to 250 kbit/s and the message length equals 2 byte user data. Three load assumptions where examined for CAN:
Scenario Sc1 For scenario Sc1 there is no load. Measuring the time demand for the cycle A → B → A gives 0.774ms which corresponds to a maximum realizable excitation frequency of 1291Hz. As can be seen on the left side in figure 5, the system reaches very closely this limit. This result can also be deduced from the skew (right side in figure 5) which reaches -100%. At this frequency the system reacts exactly at the instant of time of the next trigger, meaning that the bandwidth of the system is fully utilized. The curve of the skew is nearly linear, which indicates a highly regular behavior. Deviations are due to measurement uncertainties and background functions of

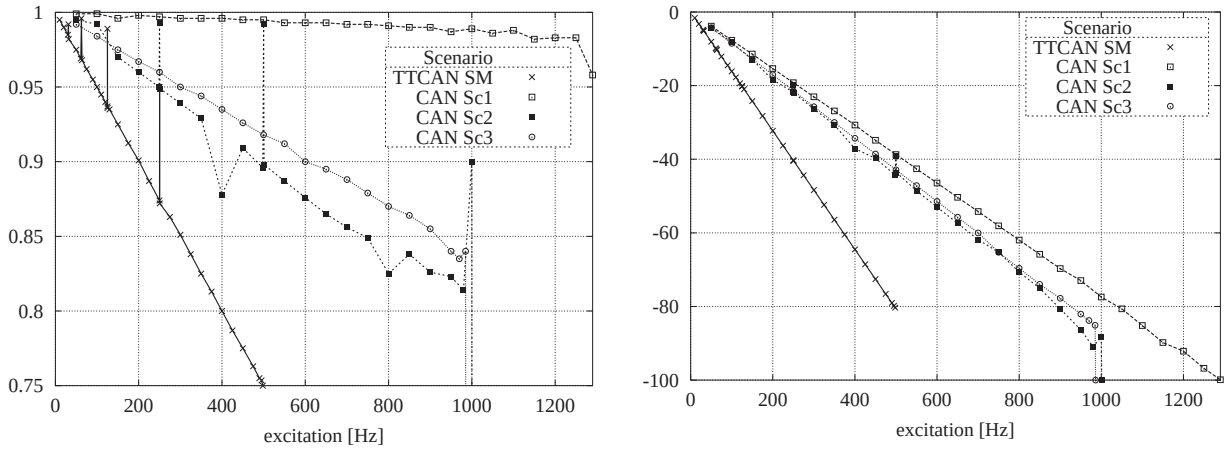


Figure 5: DoR (left) und skew (right) for CAN and TTCAN (both at the rate 250 kbit/s).

the operating system, which are more noticed at higher excitation frequencies of $i(t)$. An example of an essential background function is the timer interrupt of the operating system.

Scenario Sc2 For scenario Sc2 there is a burden of the bus by low priority messages which are send by CAN node A cyclically every millisecond. There are distinct resonances for the frequencies 250, 500 and 1000Hz (in fact, there are such resonances for every integer factor of 1000Hz, i.e. also for 125Hz, 62.5Hz etc.). The reason for this phenomenon is explained in figure 6. Since background load and excitation frequency are synchronous, there is no delay due to the background load and thus there is no jitter.

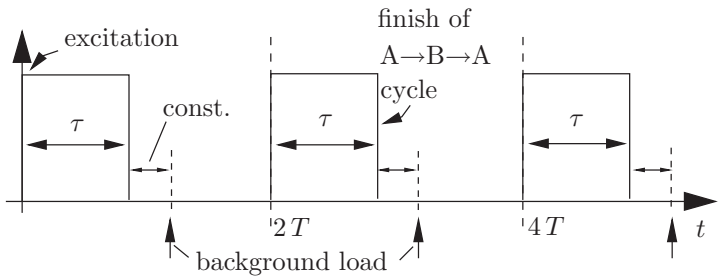


Figure 6: Synchronous background load and excitation

Scenario Sc3 Finally, for scenario Sc3 the micro controller is burdened by its serial interface. For that purpose the interface is parameterized such that an interrupt is generated after the reception of each single character (1 byte). For the serial data rate of 9600 baud and a continuous data stream there is a load of approximately 1000 IR/s. As can be recognized in figure 5 again jitter arise, since the micro controller is interrupted during the processing of the messages of the A → B → A cycle. Thus, not only load on the bus but also on the micro controller generates jitter.

For time-triggered bus concepts the communication structure is defined in advance and generally not modified during operation. The communication controllers are initialized at start-up; afterwards, they operate autonomously. Merely the data of the messages may be modified during operation. Figure 7 shows a simple communication structure with two messages (besides the

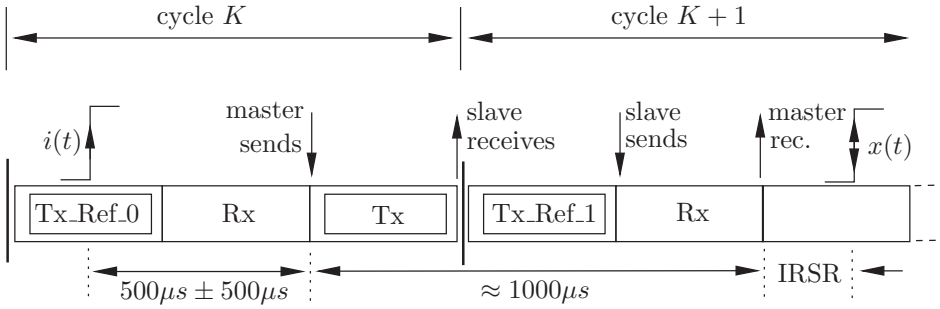


Figure 7: TTCAN cycle SM seen by the master and expected latencies

mandatory reference message of the TTCAN communication) in every cycle. This cycle is illustrated from the point of view of the master where double framed boxed mark actions of the master node. Within the cycle at first the message of the slave (node B) is defined and afterwards the message of the master (node A). Thus, from the point of view of the master, the cycle consists first of a receive message and then of a transmit message. In the following this constellation is called the scenario SM. The cycle time equals 1ms which for the chosen message length gives a bus load of 95.6% if the reference message is considered in the calculation and 66.4% otherwise. The result was already presented in figure 5. As could be expected beforehand, the maximum excitation frequency is 500Hz, since the inspected cycle A → B → A at least requires two communication cycles. Higher frequencies lead to the missing of triggers.

Again resonances are detected but here for every integer factor of 500Hz. At this frequencies the excitation and the cycle structure are synchronous. Both, the DoR as well as the skew show a linear characteristic, which indicates a regular behavior. Figure 7 allows the estimation of the expected latencies for the scenario SM as follows:

- The excitation $i(t)$ can occur at any time instant within the cycle. Therefore, the average response time of the master equals $500\mu s \pm 500\mu s$.
- There is a demand of an entire cycle until the master receives the answer message. This additionally generates a latency of $1000\mu s$ (corresponds to the cycle time).
- Finally, one has to wait for the completion of the interrupt service routine (IRSR), until the expected reaction at the output $x(t)$ is recognized.

Neglecting the run time of the IRSR one has to expect in total a latency of $1500\mu s \pm 500\mu s$. For $f = 500Hz$ one could expect an average latency of -1500/2000 = - 75%. The difference to the measured value of -80% is due to the time demand for the IRSR.

It can be summarized that event-triggered bus concepts are more efficient for small bus loads, since they generate lower latencies and less jitter when reacting to asynchronous events. But as has been shown a bus load or a load on the micro controller arbitrarily worsen the behavior of the CAN bus. For TTCAN the result is almost independent of the actual load and the DoR as well as the skew show a linear characteristic with respect to the frequency of the excitation. Thus in a sense, TTCAN is deterministic (compared to CAN), since the limit of the (worst case) time behavior can easily be determined in advance.

4 Regular operation in control systems

4.1 Sampled-data systems

Only a minority of today's control systems are implemented with analogue technique. Thus, control system generally are sampled-data systems, since computer algorithms work at certain instants of time. Figure 8 shows the structure of a sampled-data system. The system output $y(t)$

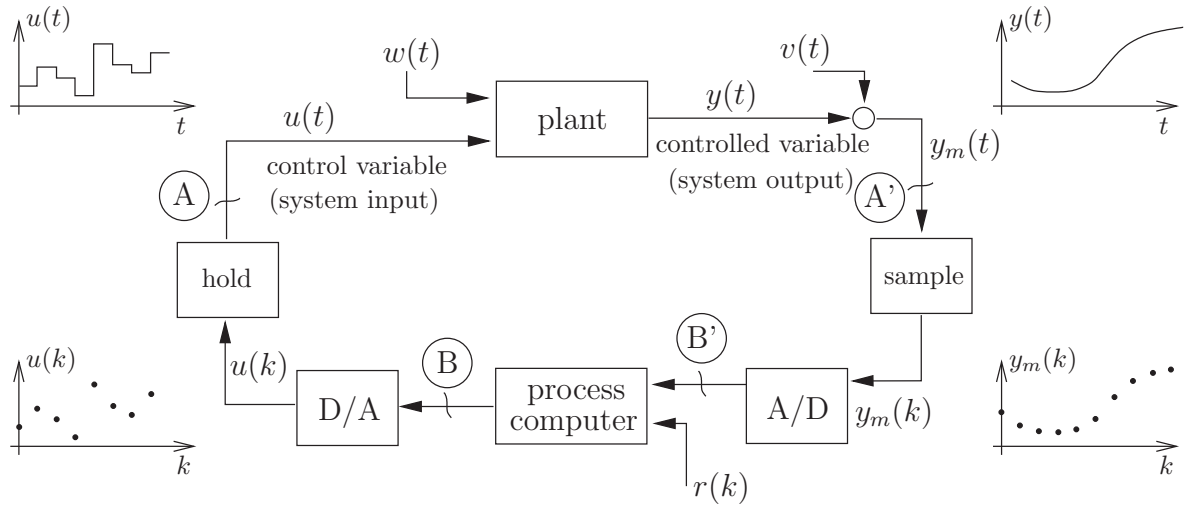


Figure 8: Sampled-data system: Digital control of a continuous-time plant.

of a continuous-time plant is controlled by the system input $u(t)$ which is generated by a process computer. Typically there are also disturbances acting on the system. For instance, the random variables $w(t)$ and $v(t)$ disturb the plant input and the measurement of the system output, respectively. When controlling a continuous-time plant it is necessary to sample the (measurable) system output $y_m(t)$ and to carry out a conversion with an analogue-to-digital converter (A/D). The controller output $u(k)$ is converted by a digital-to-analogue converter (D/A) and is hold for the sampling time. As is illustrated by the time plots in figure 8 in general a zero order hold is utilized, meaning that the system input $u(t)$ is kept constant between two sampling instants.
Most of design methods for sampled-data systems are deduced from two different point of views of

the system architecture.

1.) If the converters are considered as a part of the controller (cut at the points A and A') the controller itself can be considered as a continuous-time system. The (discrete-time) controller then tries to approximate the behavior of a (known) continuous-time controller. Generally, this treatment is sub optimal, since in the best case the controller behaves as its continuous counter part but neglects the interesting properties and potentials of 'real' discrete-time controllers [ÅW97].

2.) The second possibility is to consider the converters as part of the plant (cut at the points B and B'). Then the plant itself is a discrete-time system from the point of view of the controller. The behavior of the plant is described exactly at the sampling instants but typically there is no consideration of the behavior in between [ÅW97]. Nevertheless, also without special design methods which consider the continuous nature of the plant [CF95], in most cases the system will behave good-natured between the sampling instants.

It is mentioned that it is unfortunately a common method to entirely neglect the sampling, which can lead to a large mismatch between assumed and real system behavior. In order to minimize the error one has to choose a high sampling frequency, but then more computational performance is required than in fact is necessary and additional problems may arise due to ill- conditioning and the limited word lengths provided by the controllers.

4.2 Origin of latency and jitter

Typically sampled-data systems work on the principle 'sample-then-output'. I.e., subsequent to the measurement of the controlled variable the calculation of an adequate control variable is carried out which ideally should immediately be realized to the system. Without special precautions the appearing input/output latency τ should be as small as possible in relation to the sample period T. A frequently utilized rule of thumb is $\tau/T \leq 10\%$. But in fact the sensitivity of the system and the control design to a certain amount will 'decide' whether the latency is tolerable or not (see subsection 4.4). The local distribution of the plant and the controller leads to additional latency due to the field bus communication such that the input/output latency τ then increases. Figure 9

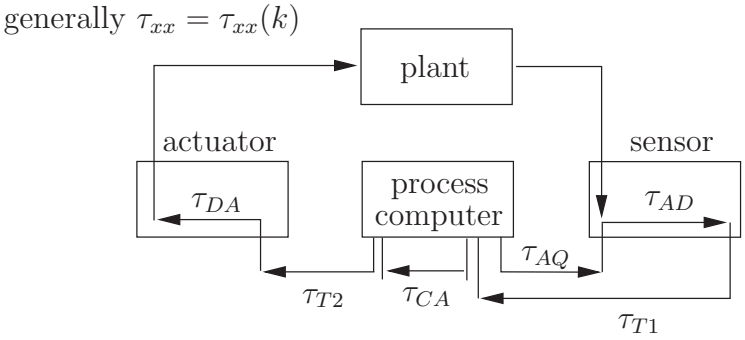


Figure 9: Origin of time delays in computer controlled systems

shows the origin of time delays in control systems. Subsequently, the different latencies τ_{xx} (which in general will be time varying) are further explained[2]. It should be mentioned, that the number and the classification of the different kinds of delays is rather arbitrary. Frequently, it suffices to distinguish between three kinds of delays, namely the computational delay of the controller and the communication delays between the sensor/controller and the controller/actuator, respectively [Nil98]. We here specially distinguish between some more latencies in order to give some more insight into the operation of the sampled-data system.

[2]The focus of this paper is to qualitatively compare the origin of latency and jitter with respect to the different architectures and the impacts on the control design. A quite comprehensive paper on a similar subject is by Lonn & Axelsson [LA99] who investigated eight different OS-bus-combinations by hand of their worst and best case latency and jitter for control tasks. In their paper the authors distinguish between event- and time-triggered operating systems, event- and time-triggered communication, and the presence or absence of a global time.

4.2.1 Acquisition delay τ_{AQ}

Sometimes it is necessary to acquire a new measurement of the system output by the process computer. Since this involves in general the transmission of a message over the bus, an acquisition delay τ_{AQ} occurs.

Event-triggered architectures Basically, the same problems arise which have been discussed already in section 3. Thus, the acquisition delay τ_{AQ} is time varying and depends, e.g. on the bus load and the message priority.

Time-triggered architectures No additional latency is generated if the schedule is designed such that the acquisition is demanded exactly the necessary duration earlier in order to deliver a just-in-time measurement. For that purpose of course, the sensor node must be synchronized to the (global) bus time.

4.2.2 Sensor delay τ_{AD}

The sensor delay τ_{AD} includes the conversion time of the A/D converter and further time delays due to signal processing, like for instance, (anti-aliasing) filter operation, outlier detection, sensor monitoring and built-in self test operation. A time varying delay component is frequently induced if an (intelligent) sensor node possesses its own time base (oscillator) and works periodically without being synchronized to the rest of the distributed system.

4.2.3 Communication delays τ_{T1} and τ_{T2}

The communication delay τ_{T1} describes the time demand for the communication from the sensor to the controller, whereas τ_{T2} stands for the communication delay between the controller and the actuator. Both delays τ_{T1} and τ_{T2} strongly depend on the bus concept. The communication delay is probably the main reason for latencies and jitter in the control loop.

Event-triggered architectures What has been said in section 3 about the origin of delay still holds true, i.e. the latency and the jitter depend on the bus work load, the message priority, the data rate and so on. Hence, in event-triggered architectures the communication delay may be time varying and quite susceptible for jitter. One method in order to overcome the problem of time variation is by Luck & Ray [LR90] who introduced buffers (sizes longer than the worst case communication delay) at the reception sides of the network. However, one disadvantage of this scheme is that the delay is larger than necessary.

Time-triggered architectures In time-triggered architectures it is essential to synchronize the actions of all participating nodes to a global time. Since the (off-line) scheduling predefines the time windows for all actions, the result is a time scheme with constant latencies and no jitter (regardless of the actual bus work load). If no synchronization is implemented, the latency and the jitter will most likely be of higher magnitude than for event-triggered systems as has been shown in section 3.

4.2.4 Computational delay τ_{CA}

The computational delay of the control algorithm is denoted by τ_{CA}. In general, the operating system and its scheduling policy will at its most influence τ_{CA}. Further, the computation time for the algorithm may vary due to state dependent calculation branches of the algorithm.

Event-triggered architectures If the process computer has to service merely only plant, an event-triggered operating system may ensure a nearly ideal behavior with minimum latency and jitter of the sampling period. The more plants have to be serviced, the more likely latency and jitter will occur. For instance, in systems with rate monotonic scheduling (RMS) the control task usually possesses the highest frequency and hence also the highest priority. This ensures a good time behavior since other tasks are preempted. Problems arise if several control loops have

to serviced [Cer03]. Low latency/jitter for one control loop can be ensured only at the cost of the other. A jitter correction is possible through the use of time variant controller, but makes the algorithm more complicated and requires additional system functions like time stamps.

Time-triggered architectures What was mentioned for the communication delays also holds true here. The (off-line) scheduling introduces delays. Assuming that a global synchronization is realized, the delays are constant and there is no jitter. It does not matter how many systems are serviced; every process has its dedicated time slots and time invariant algorithms can be used.

4.2.5 Actuator delay τ_{DA}

Lastly, there will be a delay between the reception of a new command for the control variable and the realization at the plant. In most cases this time demand is negligible in comparison with the other mentioned delays [Hus97].

A further source of delays and jitter are transient errors [San00] which are neglected here. For instance, messages can be lost or corrupted. Time-triggered bus concepts may be more susceptible for transient errors, since the retransmission of messages is often turned off in order to not disrupt the global schedule. It is true that in time-triggered architectures a missing message immediately is detected, but leads for the present to a delay of an entire sampling period!
In the summary, it can be concluded that event-triggered systems generally lead to less delays. Unfortunately, there are a lot of influences on the delays, making them time varying. The compensation of such jitter requires the knowledge of the real instants of time at which the sampling is carried out and the plant input is realized. The control algorithm itself has to be time varying. In contrast, time-triggered systems generally lead to higher but constant delays. Since the delays arise from the scheduling, they are known in advance and a delay compensation can be carried out by a time invariant control algorithm.
The following two subsections give some insight into the scheduling and the control design process for time-triggered architectures.

4.3 Scheduling in time-triggered architectures

If the task scheduling and the bus communication of a time-triggered system are synchronized, the so called input/output latency τ is constant. Figure 10 shows the generation of the input/output latency τ. Thereby an almost ideal situation is shown, since there is a very harmonic cooperation

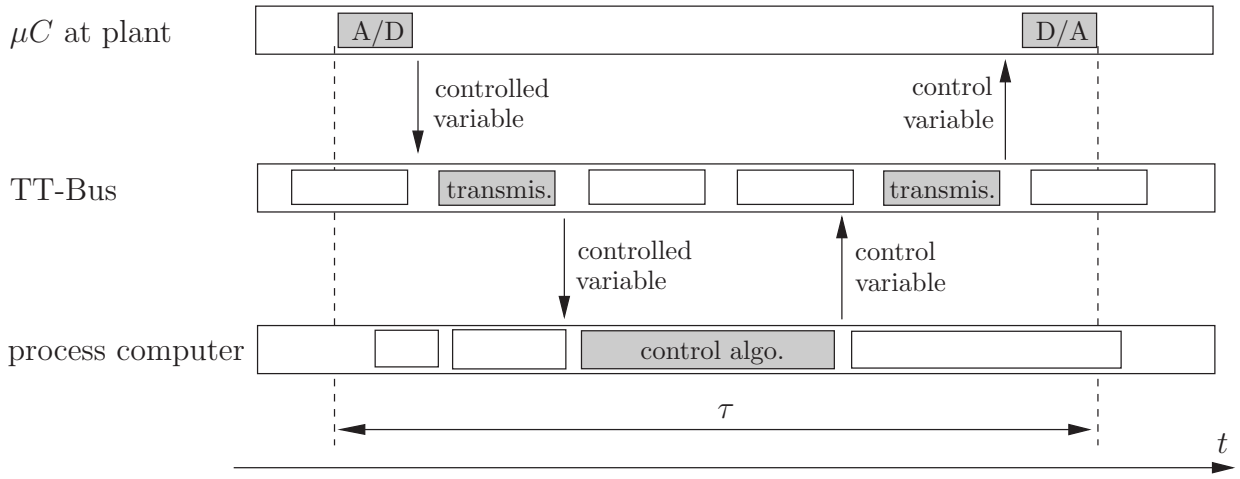


Figure 10: Input/output latency τ for a harmonized task scheduling and bus communication.

between the task scheduling and the bus communication. In the real application such a realization only rarely will succeed. As already mentioned, the execution time of the control algorithm is not

necessarily constant so that buffers are introduced in the succession of the tasks and the corresponding time windows. Besides, a suitable scheduling for the currently envisaged node could lead to very disadvantageous relations for another node. Thus an optimized (entire) system design is required which in general yields a compromise.
Figure 11 exemplarily illustrates a situation with two plants in the same network, both serviced by the same process computer. For process 1 a scheduling succeeds with minimum latency in corre-

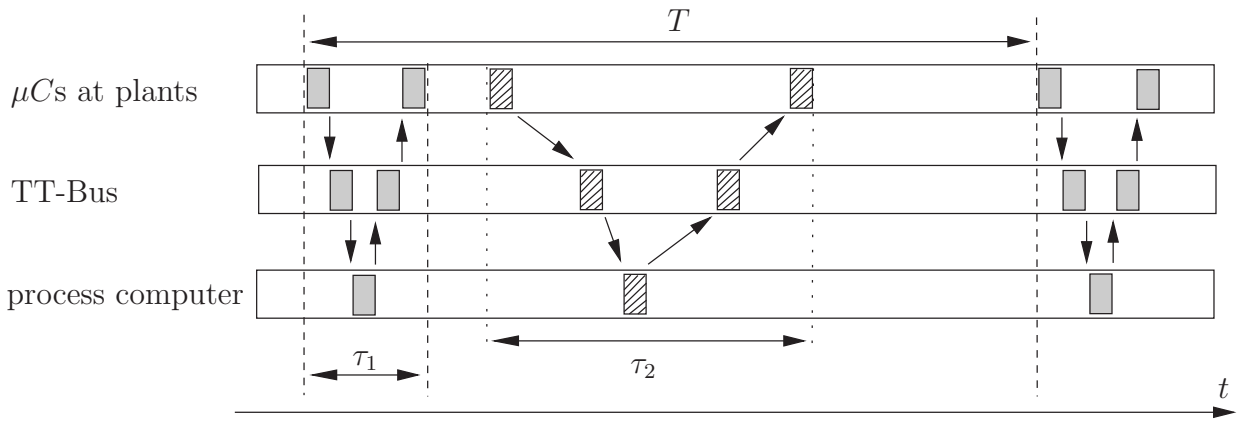


Figure 11: Relation between the input/output latency τ and the cycle time T.

spondence to figure 10; the division τ_1/T takes on the minimum possible value. Due to bus and/or computational activities such a scheduling for process 2 does not succeed. That does not necessarily mean a tragedy. Of course, one will strive after a small as possible latency, since latencies in general will decrease the control performance in comparison to the non delayed system (if the same design goal of the controller is used). But what is more important for the shown architecture is the fact that there is a constant input/output latency τ. Hence, the latency can be compensated by the control design without the need of a time variant control law.
The following example should clarify some of the mentioned statements by hand of real measurements.

4.4 Illustrative example

For the experiments a printed circuit board based on the MPC555 is utilized [AST03], which is shown in figure 12. The interrupt driven multi-tasking operating system RTOS-UH [Ger99] and

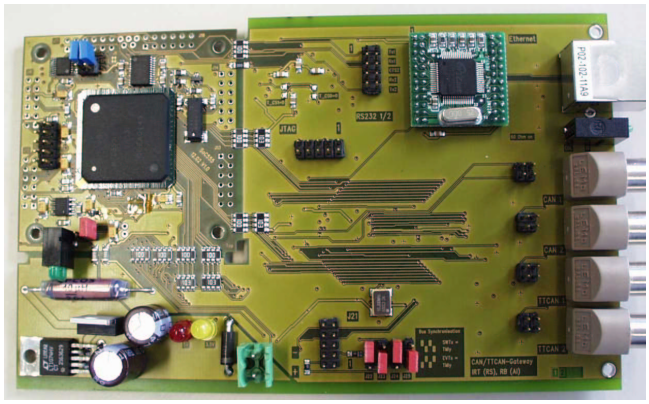


Figure 12: Printed circuit board with PowerPC MPC555, two CAN and two TTCAN interfaces

the application programs reside and run in the internal flash EEPROM of the micro controller [WAG01]. A user program management allows to simply exchange user programs via a terminal interface. In addition to the integrated peripherals of the controller, two TTCAN chips [Har02] and further digital and analogue interfaces have been made available.

In order to investigate time-triggered architectures, the emulation of a time-triggered operating system by the event-triggered OS has been realized [Alb03]. Furthermore, the implemented synchronization layer allows the synchronization of the local OS time to an external global time. In general such a time base is provided by the bus.

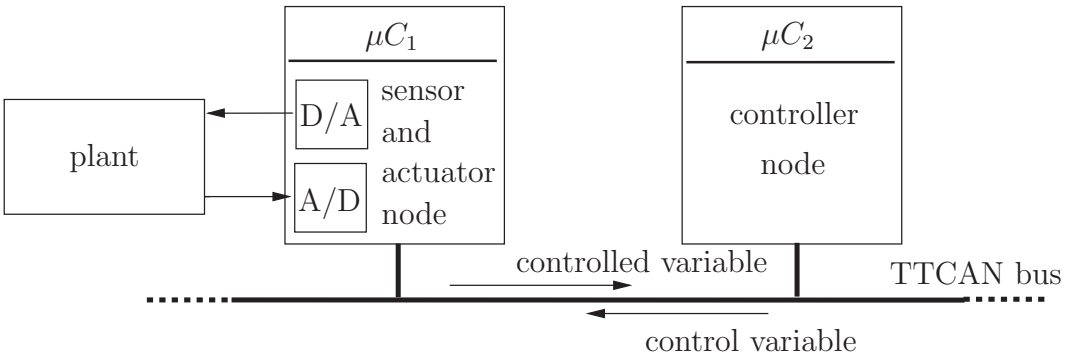


Figure 13: Investigated distributed control system

Figure 13 shows the investigated system structure. An electronic circuit based on operational amplifiers realizes a poorly damped second order plant (P_{T_2} system). The plant (input $u(t)$, output $y(t)$) is uniquely described by the transfer function

$$F(s) = \frac{Y(s)}{U(s)} = \frac{K}{1 + \frac{2D}{\omega_0}s + \frac{1}{\omega_0^2}s^2} \quad .$$

An identification of the system parameter yielded:

cycle frequency of the undamped system	$\omega_0 = 78.4$
static gain	$K = 1.18$
damping constant	$D = 0.159$.

The measured step response of the P_{T_2} system is illustrated in figure 14.

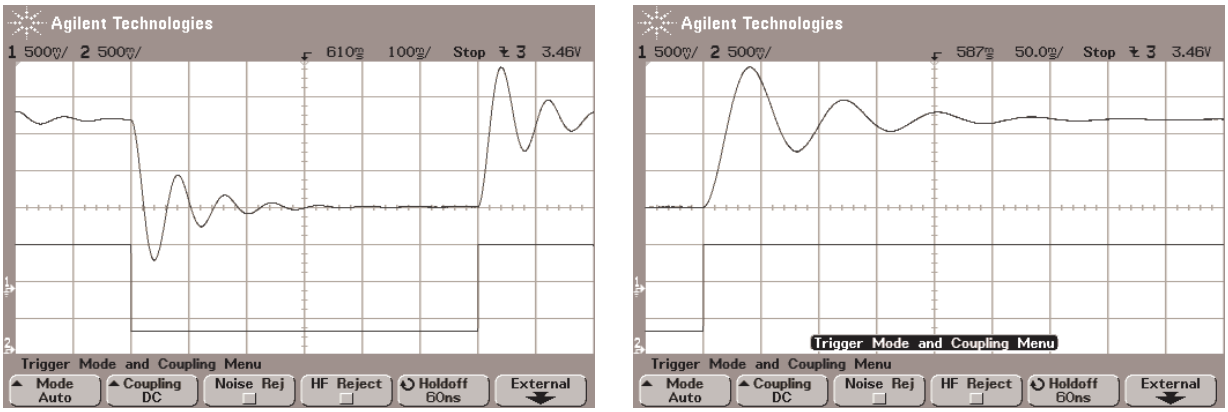


Figure 14: Step responses of the P_{T_2} system.

As is shown in figure 13 the plant is connected to a micro controller μC_1 which serves as the sensor and actuator node. The controlled variable (system output) is measured by an analogue-to-digital converter; the control variable (system input) is switched to the plant by a digital-to-analogue converter. Micro controller μC_2 is responsible for the calculation of the control algorithm. The data exchange with μC_2 is carried out via a TTCAN bus.
In order to get a reference for the best possible control behavior, in a first experiment micro controller μC_1 directly carried out the calculation of the control algorithm. That has been measured

to lead to quite ideal conditions of a vanishing input/output latency τ (μseconds). The system now has been discretized (sampling time $T = 20ms$), using the identified parameters and an observer based state controller has been designed with pole placement for dead-beat (finite settling time, here 2 sampling instants). Figure 15 shows the system's closed loop response when the reference signal is given by a square wave function with a frequency of 5Hz. After the step of the reference value, the controlled variable reaches the reference within two sampling instants.

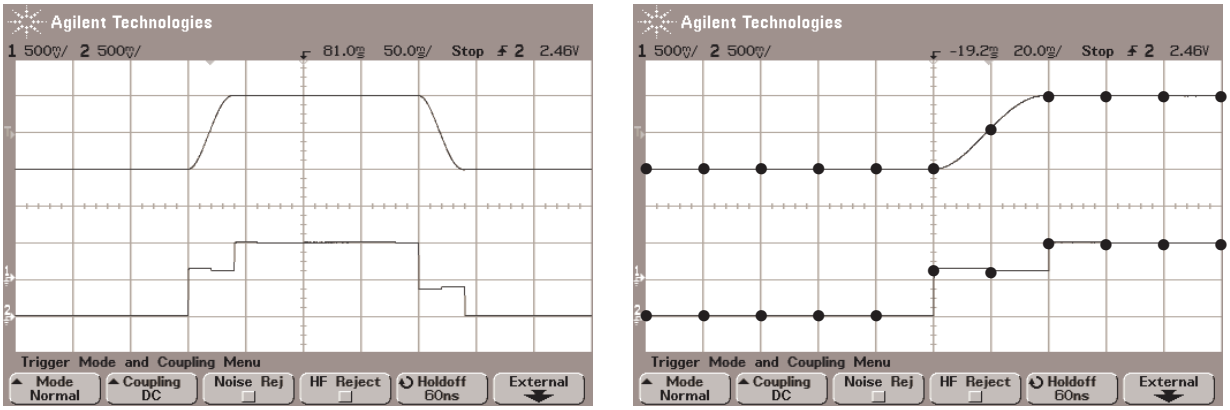


Figure 15: System's closed loop response to a step of the reference (dead-beat design for $T = 20ms$).

Now the architecture of figure 13 is utilized. On both micro controllers a time-triggered operating system is emulated as described in [Alb03]. The schedule corresponds with figure 10, leading to a latency due to the data exchange over the field bus of approximately $\tau = 1ms$. If there is no synchronization between the bus and the micro controllers, the input/output latency is varying (jitter). Then there exist time areas with low differences between the local clocks but also time areas where the difference is in the magnitude of the sampling time. A simple calculation shows the necessity for synchronization:
Consider the utilized oscillators provide an accuracy of 100ppm (parts per million), i.e. the relative error in the frequencies is below 0.01%. When comparing two oscillators in the worst case the one shows an error of +100ppm and the other of -100ppm. For the sampling time $T = 20ms$ the maximum deviation of both clocks within one sampling is

$$\triangle T = 2 \cdot 0.01\% \cdot T = 4\mu s \quad .$$

This corresponds to a maximum error of $200\mu s$ within a second. Thus, the input/output latency within only $100s$ possibly goes through the entire spectrum from 'no delay' to a delay of an entire sampling period! This significantly decreases the control performance and enforces a varying control behavior along the timeline. Figure 16 shows some time plots if the reference value equals a square wave function with a frequency of 5Hz. The original control design has been utilized without any modification.

Figure 17 shows the system's response when the time-triggered operating system is synchronized to the global bus time. In addition, a new control design has been carried out where the constant latency, resulting from the scheduling has been compensated for. Although the latencies differ ($1ms$ to $15ms$) there is always an identical system behavior, which is only shifted by the constant latency. In order to notice that, figure 17 also illustrates the reference signal. However, if the main target is to adjust the reference signal as fast as possible, the original settling time can be achieved only if the sampling time is decreased.

As has been shown by the example, latencies in principle make no problems in time-triggered architectures. It is further emphasized that due to the synchronization of all nodes to the global

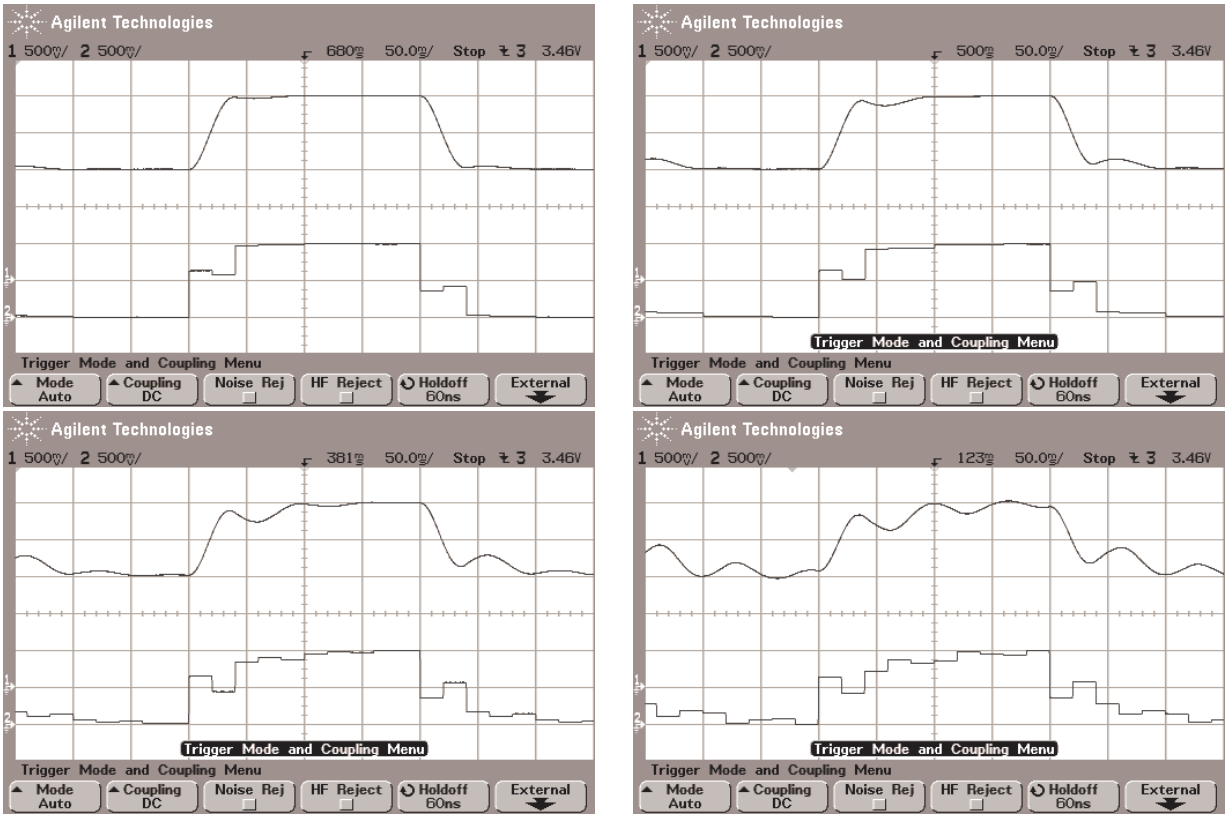


Figure 16: System's closed loop response to a step of the reference (loop closed via the TTCAN bus, dead-Beat design for $T = 20ms$). The time-triggered operating system is not synchronized to the global bus time. The corresponding latencies approximately are $1ms$ (upper left corner), $5ms$ (upper right corner), $11ms$ (lower left corner) and $19ms$ (lower right corner).

time there is qualitative no difference between actuator and sensor latency. Thus, time-triggered architectures does not require a special treatment for both types of delays as it was explained in section 2.

5 Annotation to the time-triggered design process

In time-triggered architectures the entire design process must be carried out off-line in advance. Generally, this design process contains several steps which have to be processed in an iterative fashion. This paper merely dealt with a quite small part of the design process (impacts on the controller design), since it was assumed that the scheduling of the tasks and the bus communication is already be at hand. The design process is all but not a trivial task and a lot of open problems are yet no solved. The following annotation gives a little insight into the arising questions.

One interesting opportunity of time-triggered architectures from the automotive point of view is the already mentioned composability. One of the most challenging problems in this context is concerned with the responsibility for the system integration and liability questions [Stö02]. In any case an iterative procedure will be required which roughly could look as follows:

1. First of all, it is necessary to define the desired functionality of the entire system. Usually, this task will belong to the car manufacturer. Then the necessary subsystems should be determined which have to be connected to the network. This task in general also involves the suppliers.

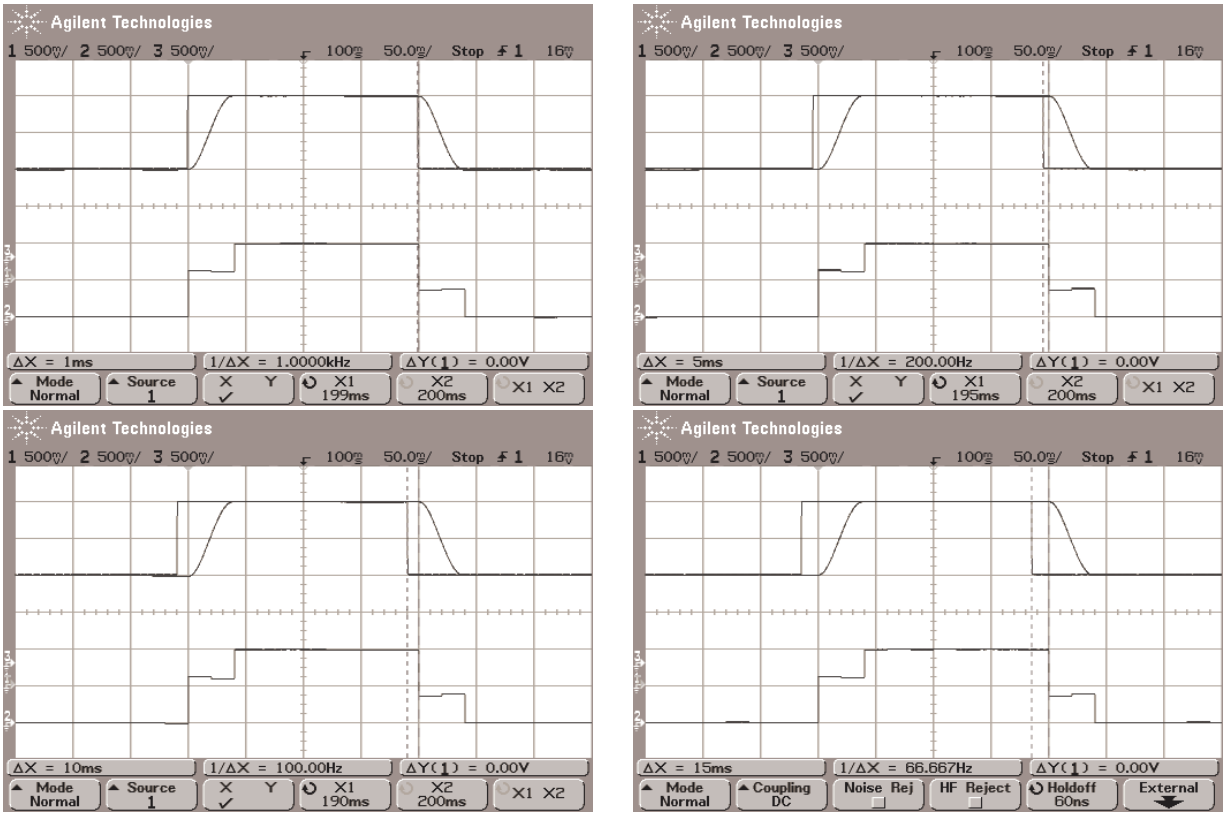


Figure 17: System's closed loop response to a step of the reference (loop closed via the TTCAN bus, dead-Beat design for $T = 20ms$). The time-triggered operating system is synchronized to the global bus time. The latencies, resulting from the scheduling are $1ms$ (upper left corner), $5ms$ (upper right corner), $10ms$ (loewer left corner) and $15ms$ (lower right corner). The controller design compensates for the constant latency of the scheduling.

2. In the next step the system integrator has to be determined. He will supervise the subsystem designers and take over responsibility for the system integration.

3. The subsystem designers now have to specify the requirements of their subsystems. For instance, they have to define interfaces between the subsystems and to specify timing requirements like the release times and deadlines for tasks as well as the sampling frequencies. Further, precedence and exclusion constraints may be formulated. This part of the scheduling already requires the use of powerful tools, e.g. in order to estimate realistic values for the worst case execution times (WCETs) of the task processing. Without powerful tools the assumptions may be too conservative and it is more likely that there is no feasible schedule.

4. The system integrator tries to find an appropriate communication schedule. For this task efficient tools are of essential importance, since the scheduling problem is generally NP-complete, implying that heuristics must be used to solve the problem [MG01]. Recent approaches for heuristics dedicated to the scheduling of heterogenous time/event-triggered systems are, for instance, proposed in [Pop03].
   If there is a feasible solution for the scheduling, the procedure ends. Possibly the result of the scheduling is considered by the subsystem designers and the control algorithms are modified in order to compensate for latencies. If there is no feasible solution, one has to return to step 3 with modified requirements.

6 Summary and conclusions

During the regular (periodical) operation of distributed control systems, time-triggered architectures seem to be ideally suited. On the one hand they often lead to a higher latency compared with event-triggered architectures but on the other hand there is no jitter if all participating nodes at the network are synchronized to a global time. Thus, time-triggered architectures offer very interesting properties from the control design point of view. For instance, a delay compensation can be carried out by time invariant control algorithms. The main advantage of event-triggered systems is their ability to fastly react to asynchronous external events. Thus, they are leading to a better real-time performance in comparison with time-triggered systems.
One drawback of time-triggered architectures is the deficiency on flexibility and scalability [San00]. A small change in one subsystem may in general imply an entire new system design. The system design itself is very complicated and there is still a lack of adequate tools for the design process.
Several times throughout this paper it has be mentioned that it depends on the actual application whether a time-triggered or an event-triggered behavior is more suitable. Very safety critical systems, like X-by-wire require fault-tolerance and redundancy. The implementation of such systems probably will fail without the framework of time-triggered architectures. For large-scale problems in the automotive field, like global vehicle dynamics control time-triggered architectures are a good choice and offers some interesting opportunities. For many other applications time-triggered architectures may be counterproductive due to the complicated design process and their inflexibility.
This paper on purpose showed a black or white view in order to illustrate some typical properties. Due to lack of space hybrid architectures which try to combine the advantages of event- and time-triggered architectures have not been considered. Nevertheless, many future systems will show hybrid architectures as can be deduced from the specification of future bus protocols [BBE+02]. They will combine the strict time-triggered operation (constant latencies, synchronized ECUs, sensor and actuator nodes) with the possibility to react to asynchronous events.

References

[AG03] A. Albert and W. Gerth. Evaluation and Comparison of the Real-Time Performance of CAN and TTCAN. *9th international CAN in Automation Conference, iCC, Munich*, pages 05/01–05/08, 2003.

[Alb03] A. Albert. Emulation verschiedener Schedulingverfahren für die Untersuchung verteilter Regelungssysteme. Technical Report FV/SLF 03/23, Robert Bosch GmbH, 2003. unpublished.

[APF02] L. Almeida, P. Pedreiras, and J. Fonseca. The FTT-CAN Protocol: Why and How. *IEEE Transaction on Industrial Electronics*, 49(6):1189–1201, Dec 2002.

[AST03] A. Albert, R. Strasser, and A. Trächtler. Migration from CAN to TTCAN for a Distributed Control System. *9th international CAN in Automation Conference, iCC, Munich*, pages 05/09–05/16, 2003.

[ÅW97] K.J. Åström and B. Wittenmark. *Computer Controlled Systems*. Prentice-Hall Information and System Sciences Series, Englewoods Cliffs, N.J., 3 edition, 1997.

[AWG03] A. Albert, B. Wolter, and W. Gerth. Distinctness of Reaction – Ein Messverfahren zur Beurteilung von Echtzeitsystemen (Teil 2). *at – Automatisierungstechnik*, 51(10), Oct. 2003.

[BBE+02] R. Belschner, J. Berwanger, C. Ebner, H. Eisele, S. Fluhrer, T. Forest, T. Führer, F. Hartwich, B. Hedenetz, R. Hugel, A. Knapp, J. Krammer, A. Millsap, B. Müller, M. Peller, and A. Schedl. *FlexRay – Requirements Specification*. FlexRay Consortium, Internet: http://www.flexray.com, Version 2.0.2, April 2002.

[CAN90] CAN. Controller Area Network CAN, an Invehicle Serial Communication Protocol. *SAE Handbook 1992, SAE Press*, pages 20341–20355, 1990.

[Cer03] A. Cervin. *Integrated Control and Real-Time Scheduling*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, ISRN LUTFD2/TFRT–1065–SE, 2003.

[CF95] T. Chen and B. Francis. *Optimal Sampled-Data Control Systems*. Communications and Control Engineering Series. Springer-Verlag Berlin Heidelberg New York, 1995.

[FMD+00] T. Führer, B. Müller, W. Dieterle, F. Hartwich, R. Hugel, and M. Walther. Time Triggered Communication on CAN (Time Triggered CAN - TTCAN). *7th international CAN in Automation Conference, iCC, Amsterdam*, pages 92–98, 2000.

[FPW97] G.F. Franklin, J.D. Powell, and M. Workman. *Digital Control of Dynamic Systems*. Addison Wesley Longman, Inc., 3 edition, 1997.

[Ger99] W. Gerth. *Handbuch RTOS-UH Version 4.2*. Institut für Regelungstechnik, Universität Hannover, Internet: http://www.rtos.irt.uni-hannover.de/, 1999.

[Har02] F. Hartwich. *TTCAN IP Module User's Manual, Version 1.6*. Robert Bosch GmbH, Automotive Equipment Division 8, Development of Integrated Circuits (MOS), 2002.

[Hus97] H. Husmann. *Ein Beitrag zur Untersuchung des dynamischen Verhaltens feldbusgestützter Regelkreise*. PhD thesis, Institut für Regelungstechnik, Universität Hannover, Fortschrittberichte VDI, Reihe 8, Nr. 655, VDI-Verlag, 1997.

[Kop97] H. Kopetz. *Real-Time Systems – Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers Boston/Dordrecht/London, 1997.

[Kop00] H. Kopetz. A Comparison of CAN and TTP. *Annual Reviews in Control*, 24:177–188, 2000.

[LA99] H. Lonn and J. Axelsson. A Comparison of Fixed-Priority and Static Cyclic Scheduling for Distributed Automotive Control Applications. *Euromicro Conference on Real-Time Systems*, 1999.

[LH02] G. Leen and D. Heffernan. TTCAN: A New Time-Triggered Controller Area Network. *Microprocessors and Microsystems*, 26(2):77–94, 2002.

[LKK03] P. Leteinturier, N.A. Kelling, and U. Kelling. TTCAN from Applications to Products in Automotive Systems. *Proceedings of the SAE International Conference, paper ID 2003-01-0114*, pages 1–10, 2003.

[LR90] R. Luck and A. Ray. An Observer-Based Compensator for Distributed Delays. *Automatica*, 26(5):903–908, 1990.

[MFH+02] B. Müller, T. Führer, F. Hartwich, R. Hugel, and H. Weiler. Fault Tolerant TTCAN Networks. *8th international CAN in Automation Conference, iCC, Las Vegas*, 2002.

[MG01] C. Siva Ram Murthy and G.Manimaran. *Resource Management in Real-Time Systems and Networks*. The MIT Press, 2001.

[Nil98] J. Nilsson. *Real-Time Control Systems with Delays*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, ISRN LUTFD2/TFRT–1049–SE, 1998.

[Org] International Standardization Organization. ISO 11898-1 (Controller Area Network, Data Link Layer), ISO 11898-2 (High-Speed Transceiver), ISO 11898-3 (Fault-Tolerant Low-Speed Transceiver), ISO 11898-4 (Time-Triggered Communication).

[Pop03] T. Pop. *Scheduling and Optimisation of Heterogenous Time/Event-Triggered Distributed Embedded Systems*. PhD thesis, Dept. of Computer and Information Science, Linköping University, ISBN 91-7373-676-7, 2003.

[San00] M. Sanfridson. Timing Problems in Distributed Real-Time Computer Control Systems. Technical Report TRITA-MMK 2000:11, Mechatronics Lab, Department of Machine Design, Royal Institute of Technology KTH, Stockholm, 2000.

[Stö02] G. Stöger. *Management und Verhalten vernetzter Elektroniksysteme im Kfz*. IIR Deutschland GmbH, 2002.

[WAG01] B. Wolter, A. Albert, and W. Gerth. User-Expandable, On-The-Chip Real-Time Operating System for High Performance Embedded Mechatronic Systems. *Proc. of the 1st IEEE Int. Conf. on Information Technology in Mechatronics, ITM'01*, pages 255–261, Okt. 2001.

[WAG03] B. Wolter, A. Albert, and W. Gerth. Distinctness of Reaction – Ein Messverfahren zur Beurteilung von Echtzeitsystemen (Teil 1). *at – Automatisierungstechnik*, 51(9), Sep. 2003.

[Wol02] B. Wolter. *Messung der Dienstgüte von Echtzeitbetriebssystemen durch Walsh-Korrelation*. PhD thesis, Institut für Regelungstechnik, Universität Hannover, Fortschrittberichte VDI, Reihe 8, Nr. 964, VDI-Verlag, Düsseldorf, 2002.