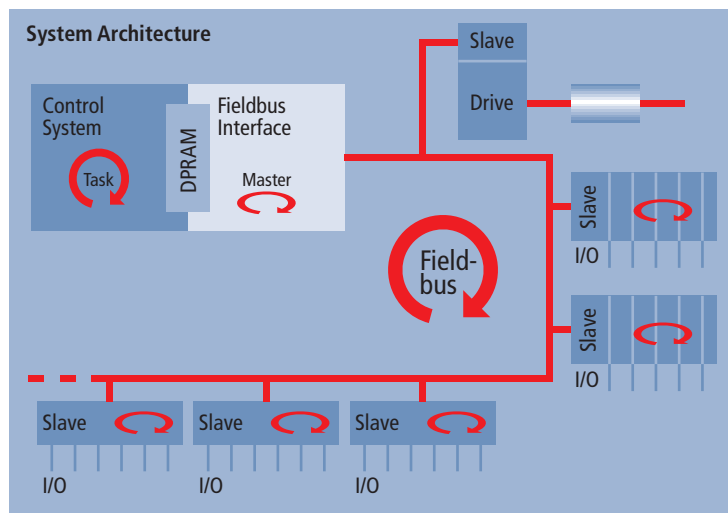


→ The performance of fieldbus systems in production applications is often described in terms of the achievable communication cycle times. The real-time capability of the different systems is then often assessed based on these figures. Other approaches also take into account the repeatability of the communication cycles. However, more critical parameters are usually ignored, such as the synchronization behavior of both the central control applications and the subordinate, decentralized application parts with the fieldbus cycle.

## Performance of fieldbus systems in production applications

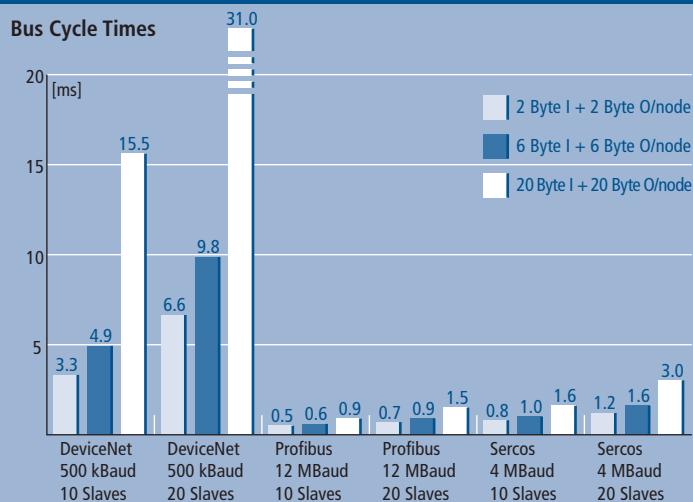
# Cycle time is not everything



Comparing fieldbus systems topology in production applications is secondary for analyzing the performance parameters: The differences in physical propagation times of bus, ring and tree structures are comparatively small. However, the baud rates and particularly the protocol efficiency are significant. In cyclic systems, this is particularly relevant for the overall cycle time.

### System structure

Let us take a closer look at a control system with decentralized fieldbus devices. The control (e.g. an IPC or a PLC) cyclically executes one or several application programs (e.g. Motion or PLC tasks). The fieldbus interface is designed as a plug-in card with a dedicated processor. Process data are exchanged with the control via a shared memory area (DPRAM-Dual Port RAM), which is accessed alternately. The simplest – and most common – communication principle for fieldbus systems is cyclic data exchange between a central fieldbus master and many decentralized fieldbus slaves. In this context it is initially irrelevant whether the data exchange and the bus access occurs via polling (e.g. Profibus), via time slicing (e.g. Sercos) or via the shift register principle (e.g. Interbus). Even the CAN-based DeviceNet,



with bus access according to the multi-master principle, is nearly always a polling system at the protocol level and can therefore usually be treated as a cyclic fieldbus. A CANopen system, typically using event-driven communication in multi-master mode, can be operated in strictly cyclic mode – e.g. for controlling axes. For the general fieldbus systems under consideration, the controllers' fieldbus interface acts as the master that controls the bus cycle. Correspondingly, the decentralized fieldbus devices are slaves. They only respond to requests from the master and can therefore communicate new input data only in line with the bus cycle.

### Fieldbus cycle time

The cycle time of the fieldbus systems is the best-known parameter. It predominantly depends on the baud rate and the protocol efficiency. The Figure "Bus cycle" provides an overview of typical cycle times for some fieldbus systems, depending on the baud rate, the number of devices and the number of data bytes exchanged. Naturally, these calculations are based on several assumptions: For DeviceNet, a bus load of 80% was assumed for pure polling operation. For Profibus, a delay of 0.3 ms was assumed between two cycles (e.g. for accessing the DPRAM). The protocol time constants (e.g. the station response time) for typical devices also had to be estimated.

### Master cycle delay/Master firmware cycle

An intelligent fieldbus control interface with dedicated processor has been assumed. This fieldbus master copies the process output image to the slaves via the fieldbus and gathers the input data. The input process image is copied to the DPRAM of the control.

For reasons of data consistency, in most implementations the input process image is only copied to the DPRAM of the control once it is complete. In a cyclic fieldbus system, therefore, this only occurs once the fieldbus cycle is complete. Copying of the input data may also be delayed, because initially enabling of the DPRAM has to be awaited. Delay times in fieldbus masters typically range between 0.1 and 3 ms.

### Control cycle time ("task")

The fieldbus performance should normally reasonably match the performance of the associated control. For typical applications, a sensible parameter is the cycle time of the associated control task.

For PC-based control systems, task cycle times of well below 1 ms are achieved due to the high performance of the PC processors. This is particularly useful for fast tasks, such as axis control. For typical PLC applications, cycle times between 2... 10 ms are usually chosen.

For "classic" PLC solutions, typical cycle times of 10... 20 ms are still common.

### Slave cycle delay/Slave firmware cycle

In most fieldbus systems, the slave devices also have a processor with associated firmware. This is polled cyclically by the fieldbus master and then queries the inputs and copies the output data to the outputs.

Modular slave devices such as the Beckhoff Bus Terminal system have a local bus that connects the I/O modules with the actual fieldbus device, the Bus Coupler. Depending on the configuration, a complete update can take between 0.1... 2 ms, even with the very fast Beckhoff K-bus.

But even for devices with fixed configuration, the local firmware cycle cannot be ignored – the inputs are queried and the outputs updated according to its cycle.

### Interfaces: Synchronization or free running

The typical fieldbus system under consideration is a multi-processor system with a large number of interfaces. The programs on the different processors usually run cyclically, in rare cases event-driven (interrupt-driven).

The simplest operating mode for the processes is free running: Decoupled via a shared memory area, both cyclic and event-driven processes can be processed largely independent from each other. The fieldbus itself can also be interpreted as a distributed shared memory area.

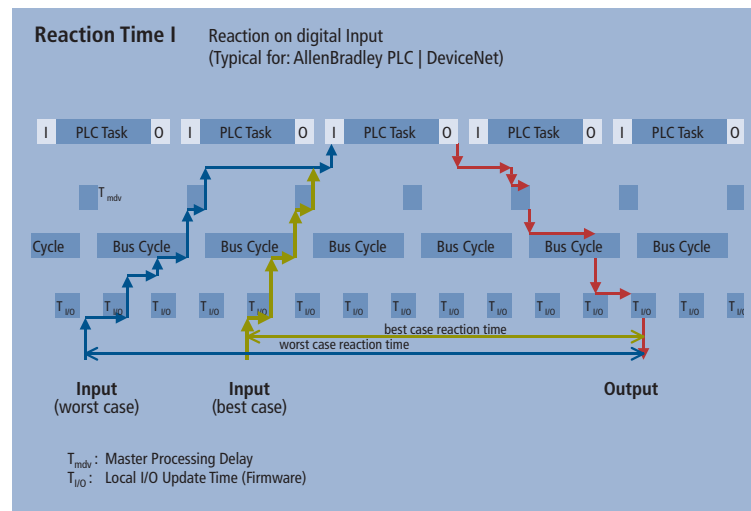
Alternatively, the different processes may be synchronized. For the slave side this means that the firmware cycle or the local bus cycle is synchronized with the fieldbus cycle. The outputs of an I/O module are immediately updated once the new output data is received from the most recent fieldbus cycle. The input data is immediately read by the local slave but not communicated until the next fieldbus cycle. Of particular interest is the synchronization of the fieldbus master with the control. In fast bus systems, fieldbus cycle and control task happen one after the other, so that the control task receives current input data each time. The outputs are written directly after completion of the control task with the next fieldbus cycle. For very fast control tasks (e.g. NC tasks for axis control) or relatively slow bus cycle times, the input and output data can be copied at the start of the control task, and the bus cycle can run at the same time as the control task. In this case the input data is from the previous bus cycle. Therefore, the control task does not have to wait for the current bus cycle to be completed.

The effects of the parameters on the response time and the determinism of the system are described below with the aid of various example configurations.

### Scenario I: Continuous free running, "medium-fast" bus system

In this scenario, the bus cycle time is similar to the task cycle time. The fieldbus cycle is free running relative to the control task. The local I/O cycle is not synchronized with the bus cycle either.

We will now look at the response time to an arbitrary input signal. In the worst case, the input signal changes straight after the local I/O cycle has queried the inputs (or is just passing through the hardware input filter). The system must wait for the next local firmware cycle, until the firmware of the decentralized I/O node has detected the event. However, it can communicate this information only during the next polling of the input data through the fieldbus – in the worst

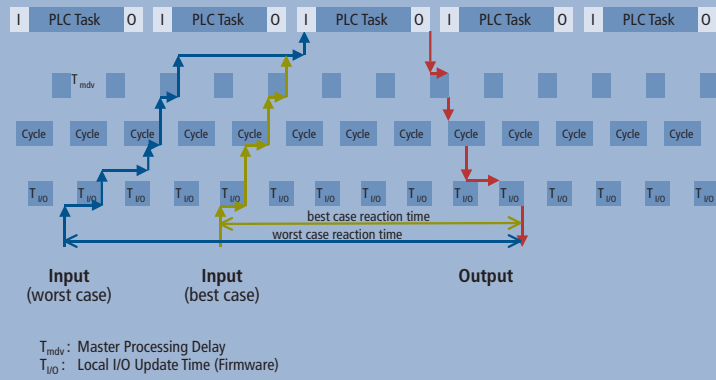


case it will have to wait for a whole fieldbus cycle.

Once the fieldbus cycle is complete, the input information is in the fieldbus master, i.e. in the scanner module of the control. The master copies the data into the DPRAM of the control and then releases the DPRAM – the time required is the master cycle delay. In the worst case, in a free running system the control task will start before the master module releases the DPRAM, i.e. the input data are not processed by this task. It will therefore take another complete control task until the input data in the DPRAM is read and processed by the control task. In the worst case, we also have to assume longest time to write the outputs: The control task does not finish before the start of the next fieldbus cycle. Therefore the master module does not have the most recent output data from the control task. In this scenario an additional fieldbus cycle has to be awaited. The output data is now transferred and just misses the local I/O cycle. As can be seen from the chart, with unfortunate timing the response time in such a system adds up to > 4 bus cycle times or a corresponding number of PLC task times. Even the jitter between ideal case and worst case – a measure of the determinism of the system under consideration – is in the range of 2 PLC tasks.

## Reaction Time II

Reaction on digital Input  
(Typical for: Siemens PLC | Profibus, AB | ControlNet)



This shows that, in this case, the system-inherent determinism of the chosen fieldbus can be ignored.

### Scenario II: Continuous free running, faster bus system

We will now try to speed up the system through a faster fieldbus: The "medium-fast" fieldbus from scenario I is replaced with a fast fieldbus with only half the cycle time compared with the first case. The complete system remains in free running mode.

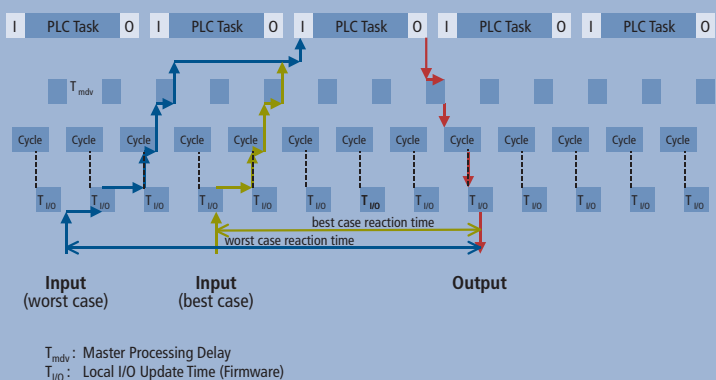
A comparison of the two charts reveals that, while the average response time has improved by approximately 20-25 %, the "best case" is the same in both cases. Overall a rather disappointing result, given that the bus was now twice as fast. We also note that the input data of some bus cycles have not been processed – they are overwritten by the next bus cycle, since no task has been started in the meantime. Furthermore, only every second or third bus cycle transports new output data. Whilst the fieldbus is faster, it is partly inefficient.

### Scenario III: Master free running, slave synchronized

We will now try to improve the result further by introducing synchronization mechanisms. Initially, the slave firmware cycle or the local I/O update is synchronized with the fieldbus cycle. This only leads to an improvement in the response behaviour on the output side: The outputs are always updated directly once new data have been received. The average improvement corresponds to half the duration of a local firmware cycle on the I/O module. The determinism of the system improves correspondingly.

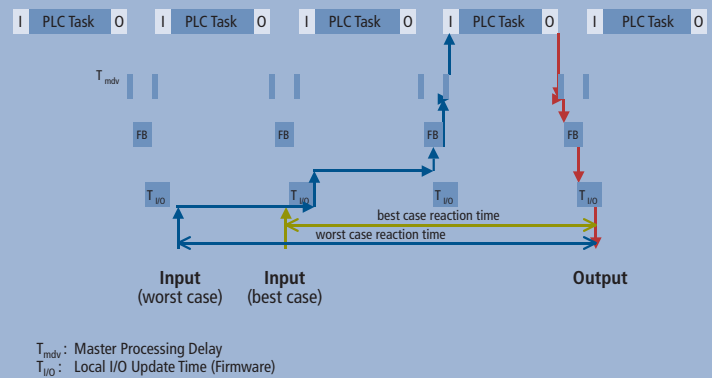
## Reaction Time III

Reaction on digital Input  
(Typical for: Siemens PLC | Profibus I/O)



## Reaction Time IV

Reaction on digital Input



### Scenario IV: Fast bus, fully synchronized

The fieldbus cycle is now also synchronized with the control task. The response behavior on the output side is optimal, the outputs are set deterministically as quickly as possible. Nevertheless, several task cycles (or bus cycles) still pass on the input side, before the input information can be processed. While the determinism of the system is good (it corresponds to the task time), the overall response time remains unsatisfactory. It is also interesting to note that the faster bus (1 ms instead of 5 ms) has done little to improve the response time.

### Scenario V: Synchronized on the master side, free running slave

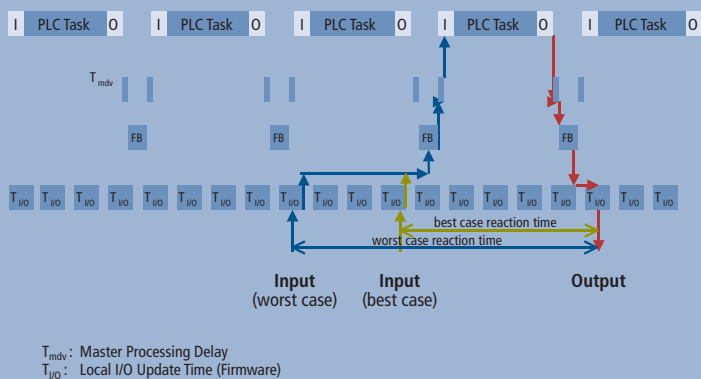
While free running slave firmware (relative to the bus cycle) is slightly disadvantageous on the output side - the input side now also captures information arriving between the "extended" bus cycles. For time conditions as shown in the graphs – task slow relative to local firmware cycle – the response time in particular has improved.

### Scenario VI: Multi-tasking system, synchronized throughout

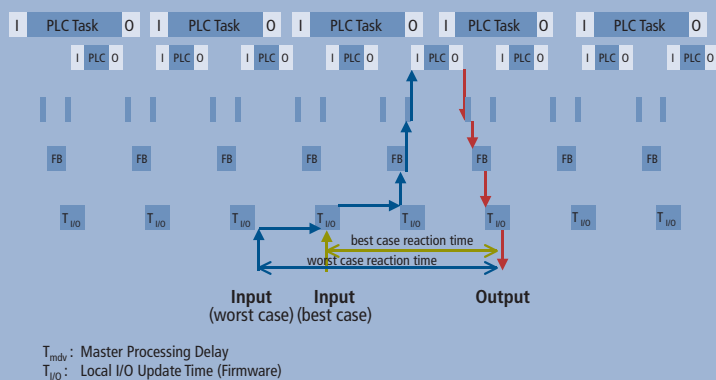
A further improvement of the response time – but mainly of the determinism – can only be achieved by shortening the task cycle time. For a given CPU performance, the simplest way of achieving this is via a multi-tasking control system: The time-critical tasks are programmed in a short task, which is synchronized with the (fast) fieldbus system. All other application parts can then be realized in slower tasks.

This scenario offers optimum response times, and at the same time optimum determinism. This is the usual system architecture of the Beckhoff TwinCAT control system.

## Reaction Time V Reaction on digital Input



## Reaction Time VI Reaction on digital Input Possible with: Beckhoff TwinCAT | Profibus | Sercos



### Scenario VII: Event-driven communication

Up to now, we have considered cyclic bus systems. Some systems – e.g. CANopen or DeviceNet – can also be operated in event-driven mode. In DeviceNet, this mode is called “Change of State”. A change of an input is regarded as an event and is communicated automatically, without request by a master.

The results of measurements for a CANopen system with event-driven communication: Despite a comparatively low baud rate (500 kBaud), the response time approaches that of a comparable Profibus system. However, the Profibus system shows this response time for all inputs, while in the CANopen system delays will have to be expected with high bus loading.

### Results

A comparison of the different scenarios provides the following results:

- | The performance of a fieldbus-networked control system cannot be determined via a single parameter.
- | The pure cycle time of the chosen bus system is usually secondary. Even for the assessment of the real-time behavior of the bus system, the cycle time is only partly useful: In many cases the synchronization behavior is the crucial factor.
- | In most cases, the cycle time of the control is more important than the cycle time of the bus system: Even with a fast fieldbus, a slow control will not achieve real performance.
- | The following generally applies for the fieldbus cycle time: “Fast enough” means faster than the cycle time of the control.
- | Synchronization initially improves the determinism of the system, but not necessarily the response time.

- | Bus systems with event-driven communication have very short response times.
- | An optimum response time with low jitter can only be achieved by tuning the system. This requires knowledge of the different temporal factors – and a system that permits such “tuning”.
- | A short response time is only one of several criteria for determining the system performance. For many applications, the determinism of the system may be the crucial factor.

### Summary

For determining the performance of a fieldbus system, the bus cycle time is only one of many parameters. Other important parameters are the synchronization behavior and the fieldbus firmware cycle times on the master and slave side. Knowledge of the control cycle times is very important for selecting a suitable bus system.

Some typical scenarios serve to describe the interrelationships of the parameters. The scenarios prove that the bus cycle time alone can only provide an estimation of the achievable response times and determinism – and thus about crucial real-time characteristics. The synchronization mechanisms of modern fieldbus systems can also be applied usefully outside drive communication applications. A fully synchronized communication system improves the determinism, but not necessarily the response time. The optimum choice or tuning of the bus system always requires an analysis of the real-time requirements: Different parameters have to be chosen, depending on whether the application to be solved is determined by response time or determinism.

→ Martin Rostan, Product Manager Fieldbus Systems