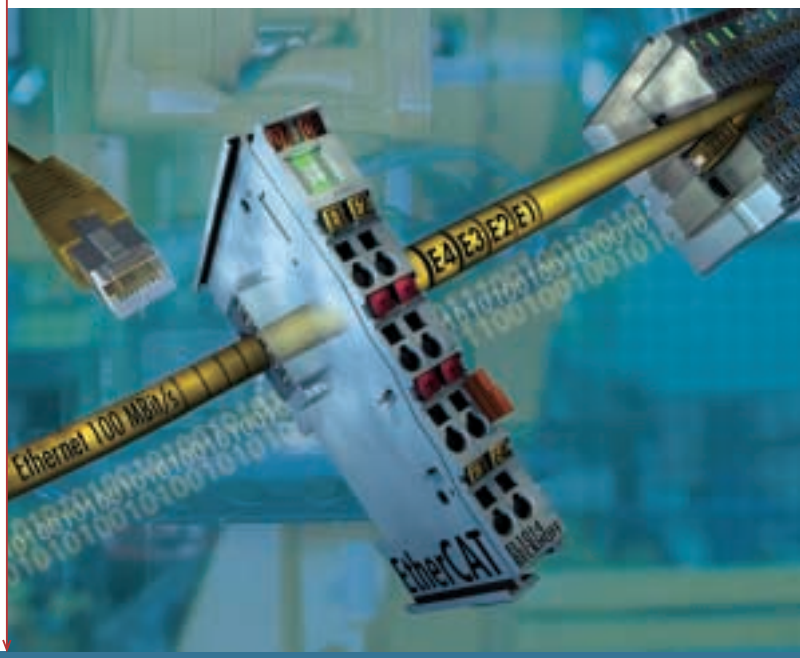
The technical response to the question: "Why EtherCAT?"

EtherCAT – the Ethernet fieldbus



Automation technology worldwide increasingly uses Ethernet as the physical communication layer. Ethernet has already become established at the command level, for factory networking and for inter-control communication. Due to the high component quantities used in office environments, it is comparatively powerful and cost-efficient.

At the field level, the fieldbusses developed in the 90s continue to dominate the markets. Fieldbusses have created their respective market segments focussing on different aspects, and with maximum gross transfer rates of 0.5 – 16 MBaud they continue to be quite adequate for many applications. However, due to the very low quantities of components sold compared with the general IT world, overall costs are rather high – both for the control and for the field devices, and particularly for the wiring. The required flexibility with regard to signal granularity was and is achieved through modular field devices that operate a subordinate sub-bus. Such sub-bus increases the complexity of the overall system and has significant performance disadvantages with regard to dead time and achievable cycle time.

Why EtherCAT?

While Moore's Law (doubling of performance approximately every 2 years) has remained valid over recent years, and will continue to apply in future, for PC-based control systems, no significant further development has occurred for fieldbusses. Instead, the majority of fieldbus organizations and "large" automation companies chose Ethernet as the future extension or replacement for "proprietary" fieldbus technology, and associated standards were developed.

Against this background, a company such as Beckhoff can expect to be asked: "Why yet another Ethernet approach?" The short, but concise answer is: "EtherCAT takes a different route and is currently by far the most powerful Ethernet approach, and the one that is best tailored to automation requirements!" A sound, technically substantiated response is presented in this article.

Ethernet at the I/O level is not without problems

Compared with existing fieldbusses, Ethernet has not only advantages. Ethernet features which make it less suitable for automation technology applications should therefore be considered very carefully and circumvented wherever possible. The main differences between the different approaches for adapting Ethernet to automation technology can be found in these Ethernet features:

- High overhead for communication with devices that have to exchange small quantities of data frequently.
- High connection costs per device compared with classic fieldbus nodes (transformer, PHY, MAC and required processor performance).
- Lack of real-time capability which, on closer inspection, is caused by slow run times in the software stacks, rather than by the fact that Ethernet is used as the transmission medium.
- Unfavorable topology. Ethernet now commonly uses a star topology, although this is rather unfavorable when it comes to system wiring and can lead to excessive cabling effort or highly cascaded communication dependencies.

Another factor is that not all Ethernet installations are identical. Apart from different transmission speeds of 10, 100 or 1000 MBaud, a distinction has to be made between half and full duplex communication. Compared with full duplex, half duplex – i.e. data transfer that alternates between two directions – transfers less than half the amount of data, since the system has to pause after each transfer in one direction to compensate for propagation and response times. Half duplex transfers can cause collisions which, while they are intercepted at the lower levels and corrected through repetitions, are not acceptable for deterministic transfers. Collisions therefore have to be avoided at higher levels, yet this in turn is not possible without further deterioration of the usable data rate.

Targets for the development of EtherCAT

Due to the performance capabilities of the PC system, PC-based control technology can support a more central or – more precisely – a hierarchical control philosophy, whereby all the pieces of information that are relevant at a particular

level are known simultaneously in a control system and can be combined with each other. This not only makes system configuration easier, but also enables the use of more intelligent control algorithms. In this configuration, the communication system is "only" responsible for the fast transport of the process data to and from the control. A master/slave communication system is the best choice for this purpose. The development of EtherCAT mainly focused on the following targets:

- Broad applicability. Any control with a commercially available Ethernet controller should be able to be used as an EtherCAT master. Starting with small 16 bit processors to PC systems with 3 GHz processors, any computer should be able to turn into an EtherCAT control system without special connections.
- Full conformity with the Ethernet standard. EtherCAT should be able to coexist with other Ethernet devices and protocols on the same bus. Standard structural components such as Ethernet switches should be usable for EtherCAT.
- Smallest possible device granularity without subordinate sub-bus. More complex nodes or 2 bit I/Os should be able to be used economically as EtherCAT slaves.
- Maximum efficiency. As much of the Ethernet bandwidth as possible should be available for user data transfers.
- Short cycle times. Possible cycle times of significantly less than 100 µs should open up new areas of application, such as closing of lower drive technology control loops.
- Maximum deterministic properties, even without the basis of absolute telegram transfer accuracy.

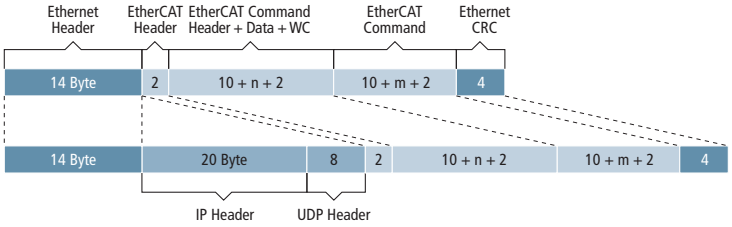
EtherCAT operating principle

From an Ethernet point of view, an EtherCAT bus is simply a single large Ethernet device. This "device" receives and sends Ethernet telegrams.

However, the "device" does not contain an Ethernet controller with downstream microprocessor, but a large number of EtherCAT slaves. These process the incoming telegrams directly and extract the relevant user data, or they insert them and transfer the telegram to the next EtherCAT slave. The last EtherCAT slave sends the fully processed telegram back, so that it is returned by the first slave to the control as a kind of response telegram. This procedure utilizes the fact that Ethernet deals separately with transfers in separate directions (Tx and Rx lines) and operates in full duplex mode: the transmitted telegrams are returned to the control by loop-back through the Rx wire pair. Naturally, like for any other Ethernet device, direct communication without switch may be established using a "crossover" Ethernet cable, thereby creating a pure EtherCAT system.

Telegram processing

Telegrams are processed directly "on the fly". While the telegrams (delayed by only a few bits) are already passed on, the slave recognizes relevant commands and executes them accordingly. Processing is done within the hardware and is therefore independent of the response times of any microprocessors that may be connected. Each device has an addressable memory of 64 kB that can be read or written, either consecutively or simultaneously. Several EtherCAT commands can be embedded within an Ethernet telegram, each addressing individual devices and/or memory areas. The EtherCAT commands are transported in the data area of an Ethernet telegram and can either be coded via a special Ether type or via UDP/IP.



EtherCAT telegram structure (with and without UDP)

While the first variant with special Ether-type is limited to an Ethernet subnet, i.e. associated telegrams are not relayed by the routers, for control tasks this usually does not represent a constraint. The Ethernet MAC address of the first device is used for addressing. This requires a special first EtherCAT device, although this is not necessary for direct communication without switch.

The second variant via UDP/IP generates a slightly larger overhead (IP and UDP header), but for less time-critical applications it enables normal IP routing to be used. On the master side, the frequently already existing TCP/IP stacks can be used. Each EtherCAT command consists of an EtherCAT header, the data area and a subsequent counter area (working counter), which is incremented by all EtherCAT devices that were addressed by the EtherCAT command and have exchanged associated data.

Fieldbus Memory Management Unit (FMMU)

Using the telegram structure described above, several EtherCAT devices can be addressed via a single Ethernet telegram with several EtherCAT commands, which leads to a significant improvement of the usable data rate. However, for 2 bit input terminals with precisely 2 bit of user data, the overhead of a single EtherCAT command is still excessive.

The Fieldbus Memory Management Unit eliminates this problem and the available data rate to be utilized is almost 100 percent – even for devices with only 2 bits of user data, as described. Similar to the Memory Management Units (MMU) in modern processors, the FMMU converts a logical address into a physical one via an internal table. The FMMU is integrated in the EtherCAT slave ASIC and enables individual address mapping for each device.

In contrast to processor-internal MMUs that map complete memory pages (in the range of 4 kB), the FMMU also supports bit-wise mapping. This enables the two bits of the input terminal to be inserted individually anywhere within a logical address space. If an EtherCAT command is sent that reads or writes to a certain memory area, instead of addressing a particular EtherCAT device, the 2 bit input terminal inserts its data at the right place within the data area. All other terminals that also detect an address match within their own FMMU also insert their data, so that many devices can be addressed simultaneously with a single EtherCAT command.

Transmission layer	Cable	Max. length	Costs (connection/meter)	Assembly characteristics	Delay (per connection in µs)
Ethernet (100 Base TX)	CAT 5 (copper)	100 m	o / ++ (+[1])	+	approx. 1
Ethernet (100 Base FX)	Fiber optic (fiber glass)	100 km	- / --	-	approx. 1
E-bus (copper)	CAT 5 (copper)	10 m	++ / ++ (+[1])	+	0
E-bus (fiber optic, in preparation)	Fiber optic (plastic fiber)	100 m	+ / +	++	0

Characteristics of the different physical layers
([1] industrial cable type i. e. suitable for drag chain applications, oil-resistant)

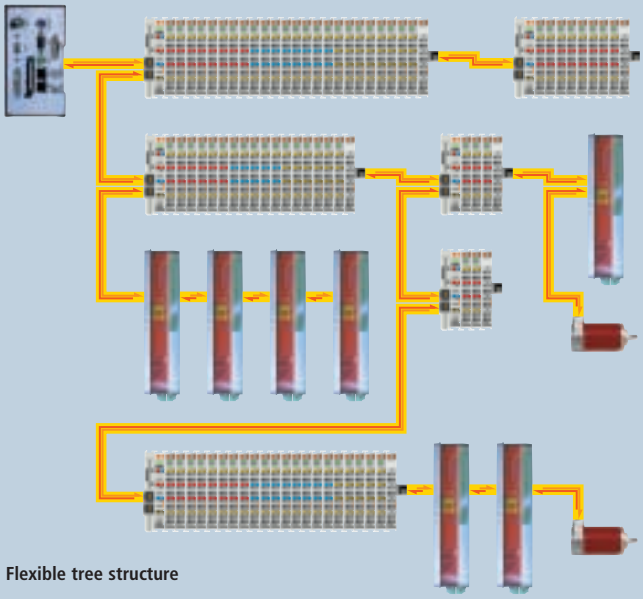
Since the FMMU is present in each device and is configured individually, the master can already assemble complete process images during the initialization phase and subsequently exchange them via a single EtherCAT command. Additional mapping is no longer required in the master, so that the process data can be assigned directly to the different control tasks (PLC, NC, etc.). Each task can create its own process image and exchange it within its own timeframe. The physical order of the EtherCAT devices is completely independent and is only relevant during the first initialization phase.
The logical address space for the FMMUs is 4 GB. From the master perspective, an EtherCAT system can therefore be regarded as a large distributed memory that can be written and read without restriction. EtherCAT is therefore a type of serial backplane for automation systems that enables connection to distributed process data for both large or very small automation devices. Via a standard Ethernet controller and a standard Ethernet cable (CAT 5), practically any number of I/O channels without restrictions on the distribution can be connected to automation devices, which can be accessed with high bandwidth, minimum delay and near-optimum usable data rate.

Physical layer

A wide range of physical Ethernet media are available, starting from the good old "Yellow Cable" to high-speed fiber optic lines. In addition to the classic CAT-5 copper cables (100 base TX), the 100 MBaud Ethernet used by EtherCAT also offers fiber optic physical layers (100 base FX). Since EtherCAT is fully compatible with Ethernet, all associated physical layers can be used. EtherCAT transmissions often involve very short distances, for example between two EtherCAT terminals within the same terminal block, so that an additional physical layer was developed – the E-bus. The E-bus is based on LVDS transmission (Low Voltage Differential Signal, IEEE standard P1596.3-1995). Apart from terminal communication, it is also suitable for cost-effective "long-distance transmissions" up to approximately 10 m.
Since all transferred data consist of fully compatible Ethernet telegrams, the physical layer can be changed anywhere and any number of times. In a system consisting of different control cabinets and machine modules, for example, for each unit the most cost-effective physical layer can be used. Within a control cabinet the E-bus is sufficient; between the cabinet and the machine modules standard Ethernet copper technology is used up to distances of 100 m. For even larger distances or extreme EMC loads, fiber optic technology can be implemented anywhere within the system.
The only prerequisite for the transmission medium is full duplex capability, since EtherCAT responds so quickly that usually the response is already sent back to the master, while the master is still sending the last bytes of its query. Half duplex physical layers as used in radio transmission, for example, would cause collisions and cannot be used for EtherCAT. In the current implementation, EtherCAT utilizes 100 MBaud Ethernet. However, the transmission principle is independent of the speed and can be used in future for 1 GBaud or more without modification.

Topology

The topology of a communication system is one of the crucial factors for the successful application in the automation industry. The topology has significant influence on the cabling effort, diagnostic features, redundancy options and hot-plug-and-play features. While the star-shaped cabling commonly used for standard Ethernet (100 base TX) has advantages with regard to hot-plug-and-play and redundancy, the cabling effort and switches required in distributed applications with many devices are not really acceptable.
In logic terms, in EtherCAT the slaves represent an open ring bus. At the open end, the master sends in telegrams, either directly or via standard Ethernet switches, and receives them at the other end after they have been processed. All telegrams are relayed from the first device to the next ones. The last device returns the telegram back to the master. Since a normal Ethernet cable is bi-directional (separate Tx and Rx cables), and since all EtherCAT slaves can also transfer in the reverse direction, the result is a physical line.
Branches, which in principle are possible anywhere, can be used to build a flexible tree structure from the line structure. A tree structure enables very simple wiring; individual branches, for example, can branch into control cabinets or machine modules, while the main line runs from one module to the next.



Flexible tree structure

Telegram transfer

Each EtherCAT slave has two Rx and Tx interfaces, via which the telegrams are transferred in one or the other direction. The telegrams are always analyzed in the outgoing direction, the return direction is used for amplifying and regenerating the signal and for locating and closing of line interruptions (Figure a).

An EtherCAT slave can detect whether a carrier signal is present on its outgoing or return line. Furthermore, each EtherCAT slave can switch the data flow in such a way that a telegram received via the outgoing line can be transmitted both via the Tx interface of the outgoing line or of the return line. Conversely, a telegram received via the return line can be sent via the Tx interface of the return line or the outgoing line (Figure b).

If the EtherCAT slave no longer detects a carrier signal on its return line, it short circuits the Rx interface of the outgoing line with the Tx interface of the return line, i.e. a telegram received via the outgoing line is processed and then sent back via the Tx interface of the return line. This functionality makes additional termination modules unnecessary, since the EtherCAT slave automatic detects that no further slave follows.

Since this situation can be caused by a short-term interruption, in the event of a short circuit the system will try to send a carrier signal or the passing telegram via the Tx interface of the outgoing line, so that it can detect the interruption being closed again. A carrier signal is then detected again via the Rx interface of the outgoing line, and the short circuit is removed (Figure c).

Obviously, an interruption may also occur on the outgoing line. If the EtherCAT slave no longer detects a carrier signal at the Rx interface of its outgoing line, it short circuits the Tx interface of the return line with the Rx interface of the outgoing line. In this case, no carrier signal or telegram is sent via the Tx interface of the return line (Figure d).

Diagnostic

For the operation of a distributed bus system, diagnostic options are no doubt just as significant as performance data, topology features or cabling effort. In this respect too, EtherCAT meets the expectations of a modern communication system. Unlike party line bus systems (e.g. Profibus or CAN), in which all devices are connected to the same physical cable, and the signals are sent to all devices without being refreshed, Ethernet (at 100 MBaud or above) and naturally also EtherCAT uses pure point-to-point transfers. Faults or even sporadic weak points that are impossible to trace in party line systems, or only with special measurement arrangements, can be located accurately.

Breaks in the logical communication ring are located and closed automatically. Each device monitors the carrier signals both in the outgoing and in the return direction and can detect faults.

Protection measures

EtherCAT checks (via checksum) whether a telegram was transmitted correctly and was processed correctly by all addressed devices (using a working counter). The standard Ethernet checksum found at the end of the Ethernet telegram is used for this purpose. Since one or several slaves modify the telegram during the transfer, the checksum is recalculated for each slave. If a checksum error is detected, a status bit is set in the EtherCAT slave, and an interrupt to the master is triggered if necessary, so that a fault can be located precisely.

During a read operation, the slave inserts the addressed data in the designated data field, from where they are retrieved during writing as required. In both cases, the addressed slave increments a working counter positioned at the end of each EtherCAT command. Since the master knows how many slaves are addressed by the telegram, it can detect from the working counter whether all slaves have exchanged their data correctly.

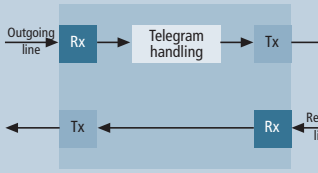


Figure a

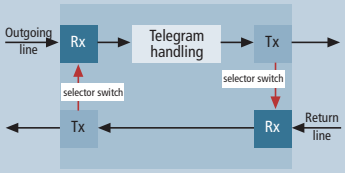


Figure b

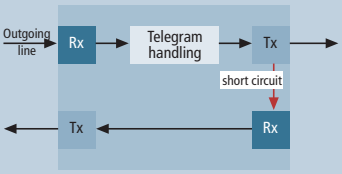


Figure c

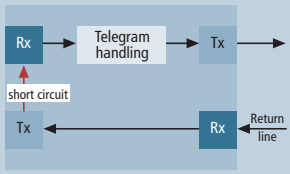


Figure d

Distributed clocks

In EtherCAT, distributed clocks enable all fieldbus devices to have the same time. A particular device contains the master clock, which is used to synchronize the slave clocks of the other devices and of the control. To this end, the control sends a special telegram at certain intervals (as required in order to avoid the slave clocks diverging beyond specified limits), in which the bus device containing the master clock enters its current time. The fieldbus devices with slave clocks then read the time from the same telegram. Due to the ring structure of EtherCAT this is possible if the master clock is located before the slave clocks in the ring.

Protocol

Fieldbusses have to meet different requirements in terms of the data transmission characteristics. Parameter data are transferred acyclically and in large quantities, whereby the timing requirements are relatively non-critical, and the transmission is usually triggered by the control. Diagnostic data are also transferred acyclically and event-driven, but the timing requirements are more demanding, and the transmission is usually triggered by a peripheral device. Process data, on the other hand, are transferred cyclically with different cycle times. It is important that "dropped cycles" are avoided. The timing requirements are most stringent for process data communication.

EtherCAT has different addressing options for different types of communication, optimized for the particular requirements.

Addressing

Physical addressing: With physical addressing, part of the 64 kB address space of a slave is read or written. A telegram invariably addresses precisely one slave. This addressing is used particularly for transferring parameter data. Two options for slave addressing are available, i.e. auto-increment or fixed-address, which are described below.

Logical addressing: With logical addressing, slaves are not addressed individually, but rather a section of a 4 GB logical address space is addressed that may stretch across any number of slaves. The assignment of physical slave addresses to logical addresses in the EtherCAT bus is configured in the master and transferred to the Fieldbus Memory Management Units (FMMU) of the slaves during the startup phase. An FMMU deals with the assignment of a logical address to the physical addresses of a device. For each FMMU, the following items are configured: a logical, bit-oriented start address, a physical start address, a bit length, and a type that specifies the direction of the mapping (inputs or outputs). Any data of an EtherCAT slave can thus be mapped bit-wise to any logical address. When a telegram with logical addressing is received, the slave checks whether one of its FMMUs shows an address match. If appropriate, it inserts data at the associated position of the data field into the telegram (input type) or extracts data from the associated position of the telegram (output type). Telegrams can therefore be assembled flexibly and optimized to the requirements of the control. This addressing mode is particularly suitable for transferring cyclic process data.

Multiple addressing: With multiple addressing, physical address areas of several slaves can be read. The first slave is addressed via the station address. The multiple read flag in the telegram is enabled, through which the subsequent slaves in the ring can recognize that they are being addressed. Each slave that has the required physical address area enters its station address and the associated data into the data field, as long as there is space available inside the telegram frame.

Broadcast: A broadcast write can be used to write physical address areas of all slaves; state transitions, for example, may be carried out simultaneously for all slaves. With a broadcast read, physical address areas of all slaves are read. For applications operating properly only if all slaves are in the OK state, this OK state can be queried very simply and quickly for all devices via a broadcast read.

Internode communication

Although EtherCAT uses a clear master/slave communication model, thereby ideally supporting hierarchical control technology, internode communication between EtherCAT devices can be created very easily using the features of the Fieldbus Memory Management Unit (FMMU). To this end, memory areas from the 4 GB logical address space are allocated for internode communication and cyclically exchanged by the master. The master alternately issues a read query and, in the next cycle, a write command for the respective memory area. All devices that are configured accordingly insert their internode communication data or retrieve them during the next cycle. For the master, these data are transparent – it merely deals with the cyclic exchange.

Compared with party line bus systems, in which all devices are connected to the same communication medium, one cycle is 'wasted'. However, this is more than compensated for by the outstanding usable data rate and the associated short cycle times. The strategy described above also has the advantage, that the internode communication data are collected from several sources and then simultaneously arrive at all addressed destinations during the next cycle. At a cycle time of, for example, 100 µs, approximately 1000 bytes can be sent from almost any number of sources to the same number of destinations.

Ethernet via EtherCAT

In addition to the already described EtherCAT addressing mode for the communication with the EtherCAT devices, an Ethernet fieldbus is also expected to feature standard IP-based protocols such as TCP/IP, UDP/IP and all higher protocols based on these (HTTP, FTP, SNMP, etc.). Ideally, individual Ethernet telegrams should be transferred transparently, since this avoids restrictions with regard to the protocols to be transferred.

There are two different basic approaches for transferring acyclic Ethernet telegrams in cyclic fieldbus mode. In the first variant, an appropriate time slice is allocated, in which the acyclic Ethernet telegrams can be embedded. This time slice has to be chosen sufficiently large to be able to accommodate complete Ethernet telegrams. The common MTU (Maximum Transmission Unit) is 1514 bytes, corresponding to approximately 125 µs at 100 MBaud. The minimum resulting cycle time for systems using this variant is approximately 200-250 µs. A reduction of the MTU often leads to problems: While the IP protocol allows fragmentation in principle, this is often not recommended, and it will no longer be available in the next generation (IPv6). Data consistency problems may also occur, particularly in UDP/IP transfers.

EtherCAT utilizes the second variant, in which the Ethernet telegrams are tunneled and re-assembled in the associated device, before being relayed as complete Ethernet telegrams. This procedure does not restrict the achievable cycle time, since the fragments can be optimized according to the available bandwidth (EtherCAT instead of IP fragmentation). In this case, EtherCAT defines a standard

Fieldbus connections

Due to the performance capability of EtherCAT, even complex bus devices, such as fieldbus master modules, can be realized. No additional connections are required in the control. EtherCAT works as a kind of PCI bus substitute, usually even providing a gain in performance. Fieldbus master scanner devices in controls have a process image interface for exchanging cyclic process data and a mailbox interface for transferring acyclic parameter or diagnostic data. With EtherCAT, the process image can usually be transferred with a single telegram, with the master sending the output process image and receiving the input process image.

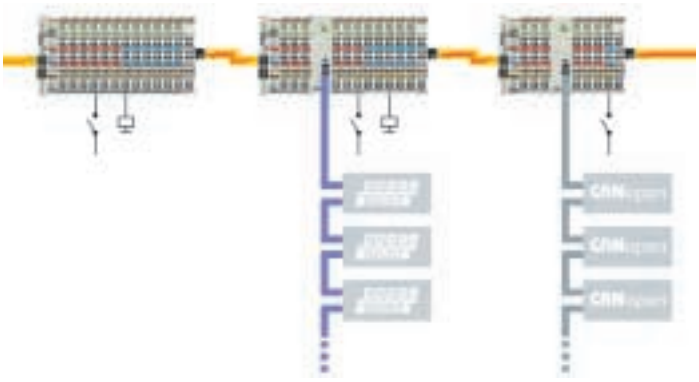
channel, which in principle enables any EtherCAT device to participate in the normal Ethernet traffic. In an intelligent drive controller that exchanges process data with a cycle time of 100 µs, for example, an HTTP server can be integrated that features its own diagnostics interface in the form of web pages.

Another application for transferred Ethernet telegrams will be hub and switch terminals. They offer standard Ethernet ports with associated RJ45 sockets at any location within the EtherCAT system, through which any Ethernet device may be connected. For example, this may be a service computer that communicates directly with the control and queries the web page of an intelligent EtherCAT device, or simply routes it to the intranet or Internet via the control. In addition to the hub terminal, the switch terminal already has a switch block integrated, offering several Ethernet connections in a single device. The EtherCAT master also features software-integrated switch functionality, which is responsible for the routing of the individual Ethernet telegrams from and to the EtherCAT devices and the IP stack of the host operating system. The switch functionality is identical with that of a standard layer 2 switch and responds to the Ethernet addresses used irrespective of the protocol.

Conclusions

In automation technology, there is currently a trend to use Ethernet also at the field level. Various approaches promise high bandwidth, low costs, simplified vertical integration, utilization of standard components from the office sector and low configuration and diagnostic effort, and all that combined with the required real-time capability.

At closer inspection however, many of the arguments become weak or change to the opposite: The comparatively high bandwidth of 100 MBaud Ethernet is ruined if typical I/O nodes with few bytes of process data are used, each addressed by one telegram. A device with four bytes per direction, for example, achieves a usable data rate of 3-4 percent. Costs too tend to argue against use in the field. Apart from the pure connection costs, another factor is the relatively high computing capacity required for processing of the telegrams in the slaves. The use of standard components usually reaches its limit when a certain degree of real-time capability is required. Furthermore, the typical switched star cabling does not lend itself very well for use in the field. Even the configuration does not become easier: The allocation of the required IP addresses requires IT knowledge and tools and frequently leads to conflicts with the IT department.

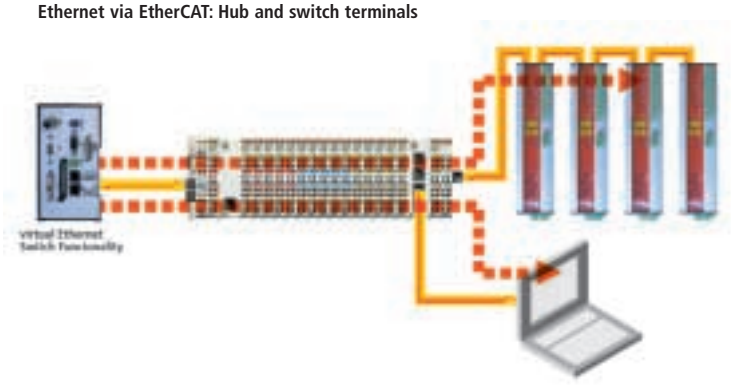


Special EtherCAT devices: Fieldbus master

EtherCAT takes a different route and combines the advantages of fieldbus technology with the otherwise indisputable advantages of the Ethernet world. The available bandwidth is almost fully utilized, and the costs are reduced to a simple ASIC connection in the EtherCAT device. Standard components are used where they are in fact standard - in the control and not in the 2 bit I/O terminal. EtherCAT does not require IP addresses, and configuration is automatic – controlled by the master using simple algorithms. Vertical integration is nevertheless available. Devices requiring an IP address can have one and are then integrated fully transparently in the network.

EtherCAT enables high-performance machine controls to be realized, capable to exchange many distributed signals with cycle times significantly below 100 µs. Moreover, the system is just as suitable for cost-effective control applications where cycle times three orders of magnitude larger are sufficient, e.g. building automation with 100 ms. In this case, any commercially available PC or any controller with integrated Ethernet port can be used as a master. EtherCAT therefore offers a unified, powerful communication basis for the entire automation sector. The same system technology can be used from "small" PLCs for less than 100 € to high-performance CNC.

Ethernet via EtherCAT: Hub and switch terminals



→ You can find the long version of this article under www.pc-contol.net