

# EtherCAT topology types and their influence on system characteristics

→ EtherCAT is the real-time Ethernet fieldbus for automation applications. In addition to top performance and low system costs, EtherCAT also offers flexible topology options. The Beckhoff system fully utilizes all EtherCAT technology features such as redundancy, Hot Connect, hot swap or synchronization of several EtherCAT networks. To this end, the EtherCAT Terminal system has been continuously expanded with a vast range of I/O terminals.

## EtherCAT operating principle

EtherCAT is an Ethernet-based communication technology that is optimized for the special requirements of advanced automation technology.

It uses standard Ethernet cables, such as those used for office networking. The EtherCAT master controlling the network and the communication (EtherCAT is a master/slave system) is managed in software on an Industrial PC, for example. The only hardware requirement is a standard network card.

The EtherCAT slave devices are based on a special EtherCAT communication chip referred to as the EtherCAT Slave Controller, or ESC, which handles all process data communication. It ensures uniform transfer speed and reliable communication, irrespective of the device-specific implementation.

The process data are exchanged between EtherCAT and the application controller via common process data interfaces with a DPRAM.

The network topology can be determined online via the EtherCAT network configurator by reading the port number and link status for each individual device.

The address space offers the opportunity to address more than 65,000 devices in one segment. The operating state of all devices is checked during each cycle. Device errors are detected synchronously with the cycle, so that the application program can respond in real-time.

### Processing "on the fly"

EtherCAT datagrams are processed dynamically. Read and write accesses are only executed on a small section of the telegram. The telegram is immediately forwarded to the next EtherCAT device (see figure 1) (i.e. it is not received first, then processed and then sent).

### Processing sequence

EtherCAT operates in full duplex mode. Telegrams are sent on a wire pair in the processing direction, from the master to the slave. The EtherCAT device processes the frames only in this direction and forwards them to the next device until the telegram has passed through all devices. The last device returns the telegram via

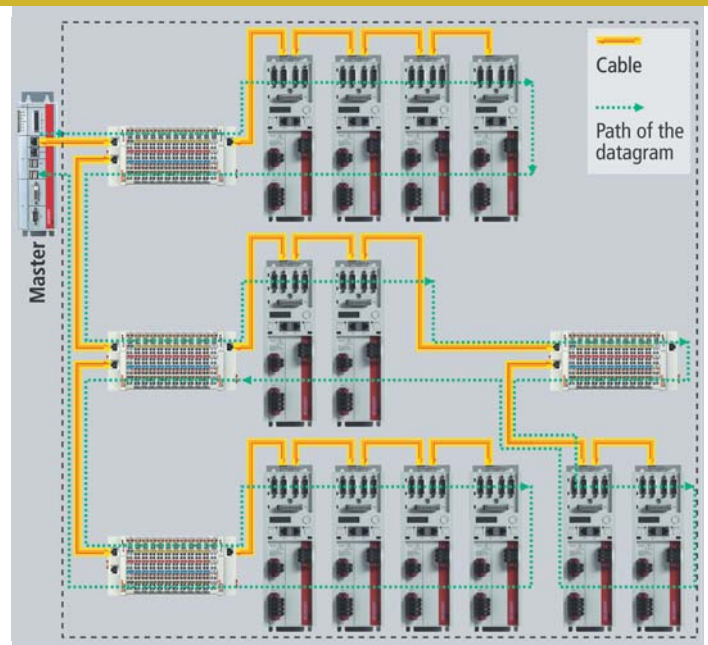


Figure 1: Dynamic datagram processing

the second wire pair in the cable back to the master in the forwarding direction. EtherCAT always forms a logical ring structure, irrespective of the chosen topology (see figure 1).

### Standardized system time and synchronization

EtherCAT devices implement a high-precision time in hardware, more precisely in the ESC. These distributed clocks (DC) give the EtherCAT synchronization mechanism its name. The first DC device after the master is usually used as a reference clock to which all other devices are synchronized. This includes compensation of different clock start times, including the master clock, and delay due to cables and other hardware. The uniform timebase generated in this way can be used to implement applications with simultaneous and synchronous read or write access to several devices. This mechanism provides a timebase with a deviation of well below 1  $\mu$ s for high-precision drive or measuring applications.

### Physical layer and signal generation

EtherCAT is transferred on the following media: 100BASE-TX, 100BASE-FX and E-bus. E-bus, which is based on the LVDS physical layer, is used for internal com-

munication and can be implemented in a compact and cost-effective manner. The 100BASE encoding options are connected with the ESC via PHYs. The MII (Media Independent Interface) is used as the interface. Each ESC generates a new physical signal so that a uniform signal quality is achieved, irrespective of the topology. It also permits an unlimited number of media changes.

### Topology options with EtherCAT

Topologies can differ significantly in terms of complexity. Bandwidth options include simple topologies such as line, tree, star or mixed configurations, as well as topologies with cross-segment and cross-system data and time stamp exchange.

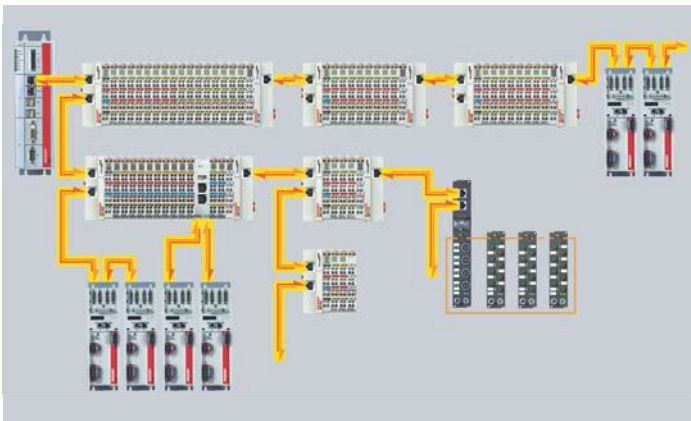


Figure 2: Tree structure

#### Tree structure

The tree structure combines the "daisy chain" topology with a stub/line topology. EtherCAT supports these classic topologies in pure form and in any combina-

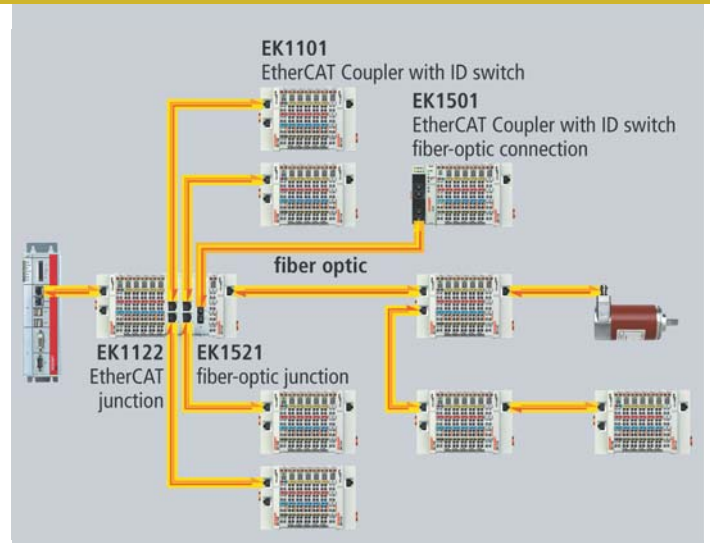


Figure 3: Star topology with real-time

tion. EtherCAT devices differ in terms of the number of ports. Devices with more than two ports (many ESCs support up to four ports) are used for connecting stubs (figure 2).

#### Star topology

In the EtherCAT Terminal system, the star topology can be conveniently implemented with the EK1122 2-port EtherCAT branch terminal. Like the tree topology, the star topology has the advantage that a device failure or line interruption does not lead to uncoupling of other devices (see figure 3). With this topology, the logical ring and the real-time characteristics based on it are maintained.

#### Hot Connect

Hot Connect allows decoupling and coupling of devices or segments during operation. The EK1101 and EK1501 EtherCAT Couplers (fiber-optic) with additional ID switch make this process particularly convenient: they are detected regardless of their position in the network and can be connected to any free port. Thanks to the characteristics of the EtherCAT Slave Controller, the coupling process is detected very quickly. Ports can be switched off from the master before the device or segment is uncoupled.

**Ring structure for cable redundancy**

The ring topology is used for implementing cable redundancy (see figure 4). To this end, the last device in the processing sequence is connected to the master. Any device with at least one free MII port can be used as the last device. The only additional hardware device used for redundancy purposes is a second Ethernet port; otherwise the master remains a software implementation.

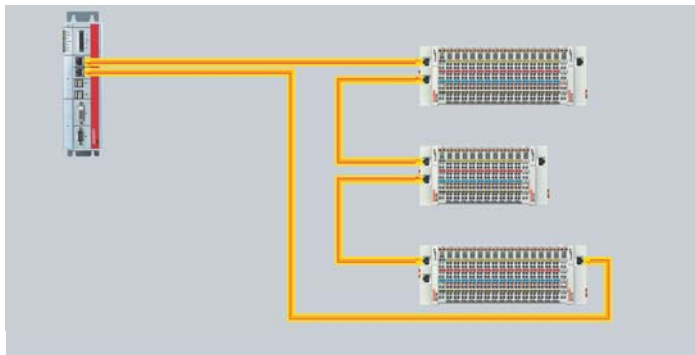


Figure 4: Ring structure for cable redundancy

**Synchronizing several EtherCAT networks**

Data exchange between two or more EtherCAT networks can be easily achieved using several switch ports or a bridge (figure 5). Two switch ports from different segments can be connected for data exchange purposes.

In addition to data exchange, the EL6692 EtherCAT bridge terminal can also be used for synchronizing networks in order to make a homogeneous timebase available across system boundaries. This is particularly beneficial for test rig construction or modular machines with several controllers, for example. In addition, EtherCAT offers further master/master communication options.

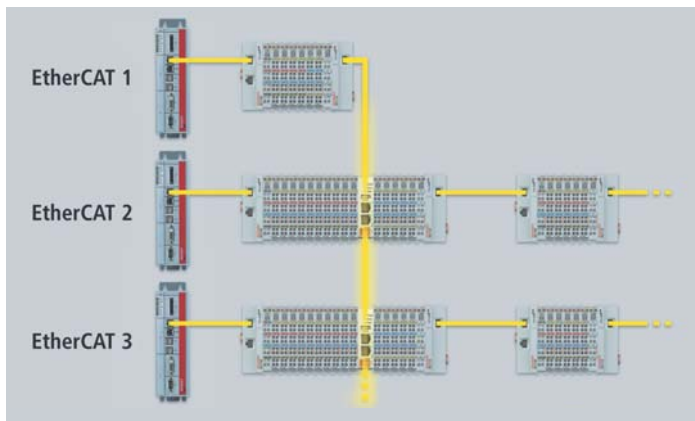


Figure 5: Synchronizing several EtherCAT segments

**Master/master communication**

For acyclic or cyclic data exchange, masters can be connected directly with standard switches and a second network card, or alternatively via an EL6601 (1 port) or EL6604 (4 ports) EtherCAT switch port terminal.

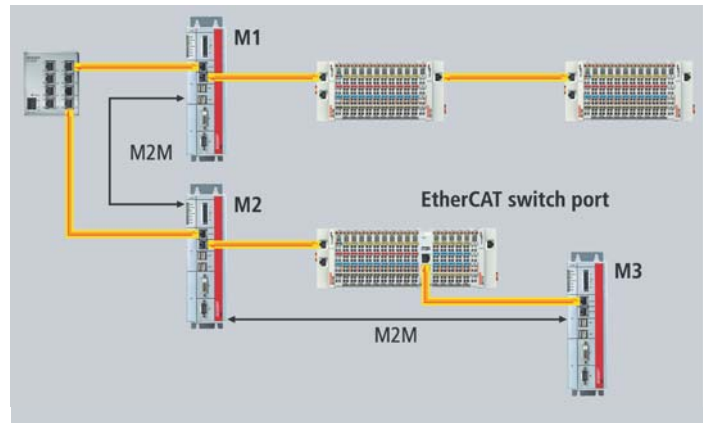


Figure 6: Master/master communication

**Slave/slave communication**

For data exchange between different slaves, the master takes on the function of a router. The master copies the data read by a device from the input process image to the output process image before they are sent. This may even take place within the same control cycle.

For applications with particularly stringent requirements, the topology-dependent version of slave/slave communication can be used: one device adds data to the telegram in transit, which are then analyzed by downstream devices.

The full article is available from [www.pc-control.net](http://www.pc-control.net)

→ EtherCAT [www.beckhoff.com/EtherCAT](http://www.beckhoff.com/EtherCAT)

→ EtherCAT Terminals: [www.beckhoff.com/EtherCAT-IO](http://www.beckhoff.com/EtherCAT-IO)

## Planning aid for users

Some of the main network planning issues are explained below.

### Calculation of the pass-through time

Since EtherCAT uses full duplex, several telegrams can be sent successively without having to wait for the return of the previous frame. The cycle time is, therefore, not the same as the pass-through time. However, in practice the latter is often specified as the minimum cycle time. Disregarding all optimization options and the symmetry of input and output data, it can be determined as follows:

$$t_{\text{Delay}} = m \times t_{\text{E-bus}} + n \times t_{\text{MII}} + t_{\text{PD}} + 2 \times t_{\text{Cable}}$$

Delay	Description
$m \times t_{\text{E-bus}}$	delay due to m devices with 2 E-bus ports
$n \times t_{\text{MII}}$	delay due to n devices with 2 MII ports
$t_{\text{PD}}$	+ delay due to process data length (outputs + inputs) at 100 Mbit/s (neglecting the fact that the process data length in the frame is halved for a symmetric ratio of I/O data per device) + 26 bytes overhead per Ethernet frame + 12 bytes overhead per datagram
$t_{\text{Cable}}$	delay due to 100BASE-TX cable (~ 5 ns/m).
$t_{\text{P}}$	delay in processing direction
$t_{\text{F}}$	delay in forwarding direction

### Delay due to a device with 2 E-bus ports ( $t_{\text{E-bus}}$ )

The delay due to an EtherCAT device with two E-bus ports (e.g. modular I/Os) is determined through the hardware delay of the ESC and is around 0.3  $\mu\text{s}$ .

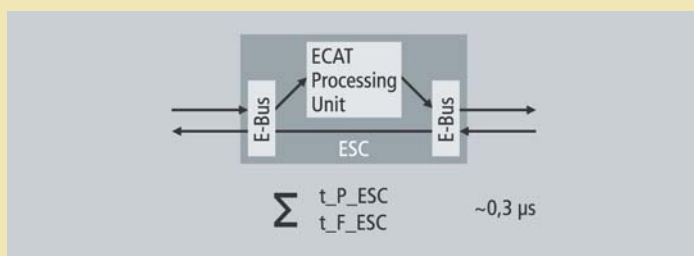


Figure 7: Delay for an E-bus device

### Delay due to a device with 2 MII ports ( $t_{\text{MII}}$ )

The delay due to an EtherCAT device with 2 MII ports (e.g. drive) with two RJ 45 Ethernet connectors is determined through the hardware delay of the ESC and the two PHYs. It is around 1.2  $\mu\text{s}$ , depending on the PHYs.

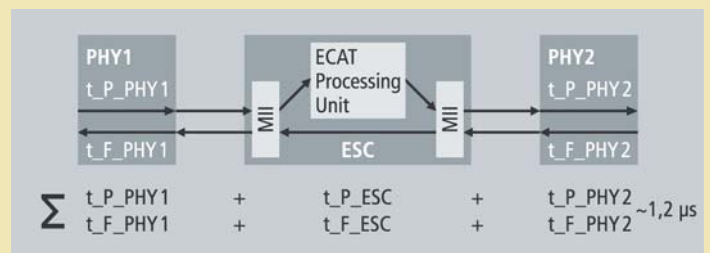


Figure 8: Delay for an MII device

### Choice of topology

With EtherCAT, the choice of topology has no negative influence on the functionality, real-time or other features. This fact allows the topology to be tailored to the physical extent of the system and the application of special functionalities, such as Hot Connect or redundancy.

### Use of Hot Connect

Many applications require a change in I/O configuration during operation. Examples include processing centers with changing, sensor-equipped tool systems or printing machines in which individual printing units are switched off. The protocol structure of the EtherCAT system takes account of these requirements: the Hot Connect function permits parts of the network to be linked and decoupled or reconfigured "on the fly," offering flexible response capability for changing configurations (see also figure 3).

### Use of hot swap

If a complex device is replaced, the replacement device must be parameterized identically. This requires knowledge of special configuration tools and the relevant parameters. In many cases, this expertise is not available on the spot. With EtherCAT, this problem can be circumvented through the hot swap functionality. The parameter data are stored in the master and automatically transferred when the device is switched on.

### Use of distributed clocks

The functionality of the distributed clocks (DC) synchronization mechanism is independent of the network structure. Devices with and without DCs can be positioned as required. The EtherCAT master automatically synchronizes the clocks without the need for special user settings.

## Implementation aspects

With all the options that EtherCAT offers, the consideration for device manufacturers is the associated development effort. ESC supports some features automatically, while others require software and/or hardware extensions.

### Port selection

A primary feature of an EtherCAT device is the number of ports and the port type. In order to maintain the flexibility of the topology, an EtherCAT device should have at least two ports, so that further devices can be connected in a series. For non-modular devices, such as drives, this means at least two MII ports and for modular devices, two E-bus ports can be used.

Devices with additional infrastructure features, such as a distributor terminal for star topology or a converter from 100BASE to E-bus and vice versa, may have several different ports. The number and type of ports play a crucial role in the selection of the ESC, because they can differ significantly.

### Hot Connect

For Hot Connect, support for a position-independent address is recommended so that a machine module can be connected to any free port within the network and be unambiguously identified. This address must still be available after a voltage loss.

EtherCAT uses a second address for Hot Connect, the so-called "station alias." Depending on the implementation, this can be read from an ESC register or transferred as process data. In both cases, a setting option in the slave device is re-

quired, which can be realized via a DIP switch or an operator panel, for example. Another source for the station alias may be the non-volatile configuration memory of the ESC, from which the address can be loaded into a register.

### Hot swap

If a device has to be replaced, complex devices such as drives often require reparameterization. In order to avoid this, EtherCAT describes a procedure for backup parameters. This enables device manufacturers to identify key parameters. All identified parameters are automatically loaded into the new device after a replacement.

The slave firmware requires an extension for backup mechanism support. This includes an extension for identification of the backup objects, storage of these objects in non-volatile memory and data validation.

### Redundancy

For the purpose of cable redundancy, the EtherCAT telegram is sent on two network ports. Normally, only the telegram in the processing direction is processed, while the second telegram passes through the network in the forwarding direction and is not processed. In a situation requiring redundancy, both telegrams pass through part of the network in processing and forwarding directions.

In the slave, the redundancy concept requires no extension. The required functionality is based on available mechanisms, which all ESCs support.

On the master side, a software extension is required that analyzes both telegrams and generates a complete picture of the communication for the master application.