EtherCAT | System Description (index.html)

EtherCAT Distributed Clocks

An introduction into distributed clock technology as a feature of the EtherCAT protocol is provided below. This section provides an overview of the main aspects from a user perspective, which should be sufficient for standard EtherCAT applications. The subsequent section provides further details and more detailed descriptions for interested users. This information is not essential for standard operation of an EtherCAT slave.

Contents of this documentation

This introduction is limited to a description of the functions that are relevant for the user. Further and more detailed information about EtherCAT in general and distributed clocks in particular can be found under http://www.ethercat.org/ (http://www.ethercat.org/).

The main terminology is shown in italics below.

The following page comprises the sections

- The principle of distributed clocks (2469118347.html#2470179851)
- Optional additional ESC functions (2469118347.html#2470182411)
- Application of distributed clocks and synchronicity with the control system in the PC

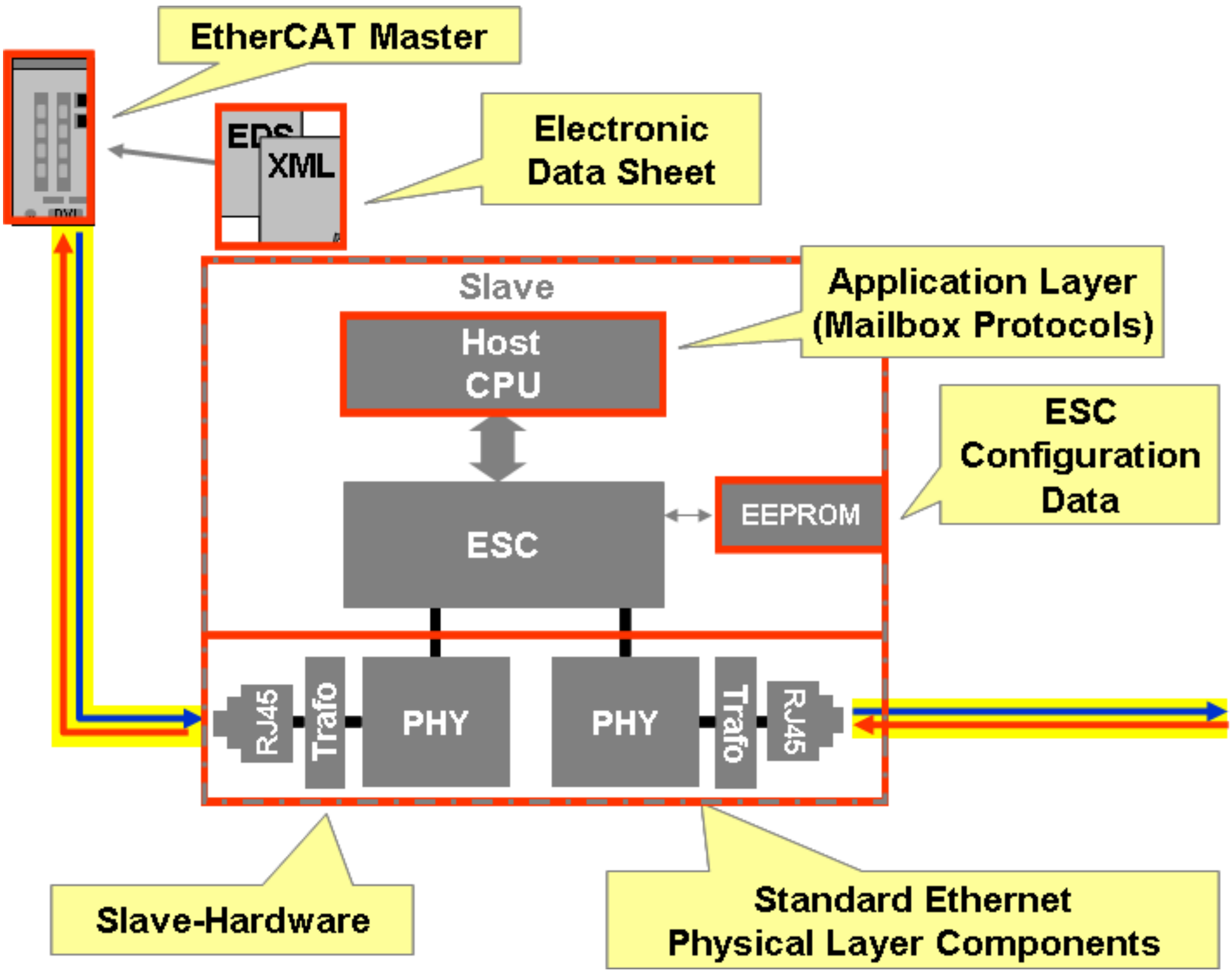
The principle of distributed clocks

In EtherCAT terminology the term "distributed clocks" refers to a logical network of distributed clocks. By using distributed clocks the EtherCAT real-time Ethernet protocol is able to synchronize the time in all local bus devices within a very narrow tolerance range. If an *EtherCAT slave* supports distributed clock functionality it contains its own clock that initially operates locally after switch-on, based on an independent clock generator within the EtherCAT slave (quartz, oscillator, ...). In the EtherCAT strand there is a selected EtherCAT slave that represents the *reference clock* (M, see Fig. *Distributed Clocks in the EtherCAT System*), to which the slave clocks (S) of the other devices and the controller synchronize. The reference clock therefore represents the *system time*. Adjustment and synchronization are handled automatically and consecutively by the *EtherCAT master* provided it supports distributed clock functionality, such as the Beckhoff TwinCAT EtherCAT master. To this end the EtherCAT master sends a special *EtherCAT datagram* at short intervals (with sufficient frequency to ensure that the slave clocks remain synchronized within the specified limits), in which the EtherCAT slave with the reference clock enters its current time. This information is then read from the same datagram by all other EtherCAT slaves featuring a slave clock. Due to the ring structure of EtherCAT this is possible if the reference clock is topologically located *before* all other slave clocks. The EtherCAT master therefore selects the first distributed clock-capable EtherCAT slave as reference clock.

An EtherCAT configuration therefore features an EtherCAT master that operates and manages the bus with the connected EtherCAT slaves. *One* of these EtherCAT slaves contains the reference clock, all other EtherCAT devices (i.e. including the EtherCAT master) represent slave clocks.

The *EtherCAT slave controller (ESC*) handles the EtherCAT communication and in particular the distributed clock functionality in an EtherCAT slave. This is an electronic component (chip) such as an ASIC or reprogrammable FPGA or similar. Each EtherCAT slave has such an ESC to ensure that cyclical and acyclical process data can be exchanged between master and slave via the EtherCAT fieldbus. This ESC can handle simple functions such as digital inputs and outputs directly, or it can be connected to a further processor in the EtherCAT slave via serial/parallel interfaces for handling more complex tasks such as drive control. In particular the ESC manages the local distributed clock functionality with the associated tasks, if the EtherCAT slave is required to support this feature.

The schematic of an EtherCAT slave is illustrated in Fig. *Example for ESC functionality and embedding in a typical EtherCAT slave.* In addition the embedding of the ESC in the latter with a selection of its basic functions.



Example for ESC functionality and embedding in a typical EtherCAT slave

From the RJ45 socket the electrical signals are transferred to the PHY (physical interface) via the transformer. It extracts the user data from the coded Ethernet signal and transfers them to the ESC for processing. The EtherCAT telegram is then relayed with minimum delay (due to dynamic processing) to the next EtherCAT slave via the PHY and the socket. The ESC automatically parameterizes itself with configuration data from an EEPROM when the slave starts up. If a further CPU exists in the slave, the slave can communicate with it via interfaces.

The distributed clocks unit of the ESC in the full configuration offers the following features (dependent on the device implementation):

- Clock synchronization between the EtherCAT slaves and the master
- Synchronous generation of output signals (Sync signals)
- Synchronous reading of input signals
- Precise time stamping of input signals (latch signals)
- Generation of synchronous interrupts

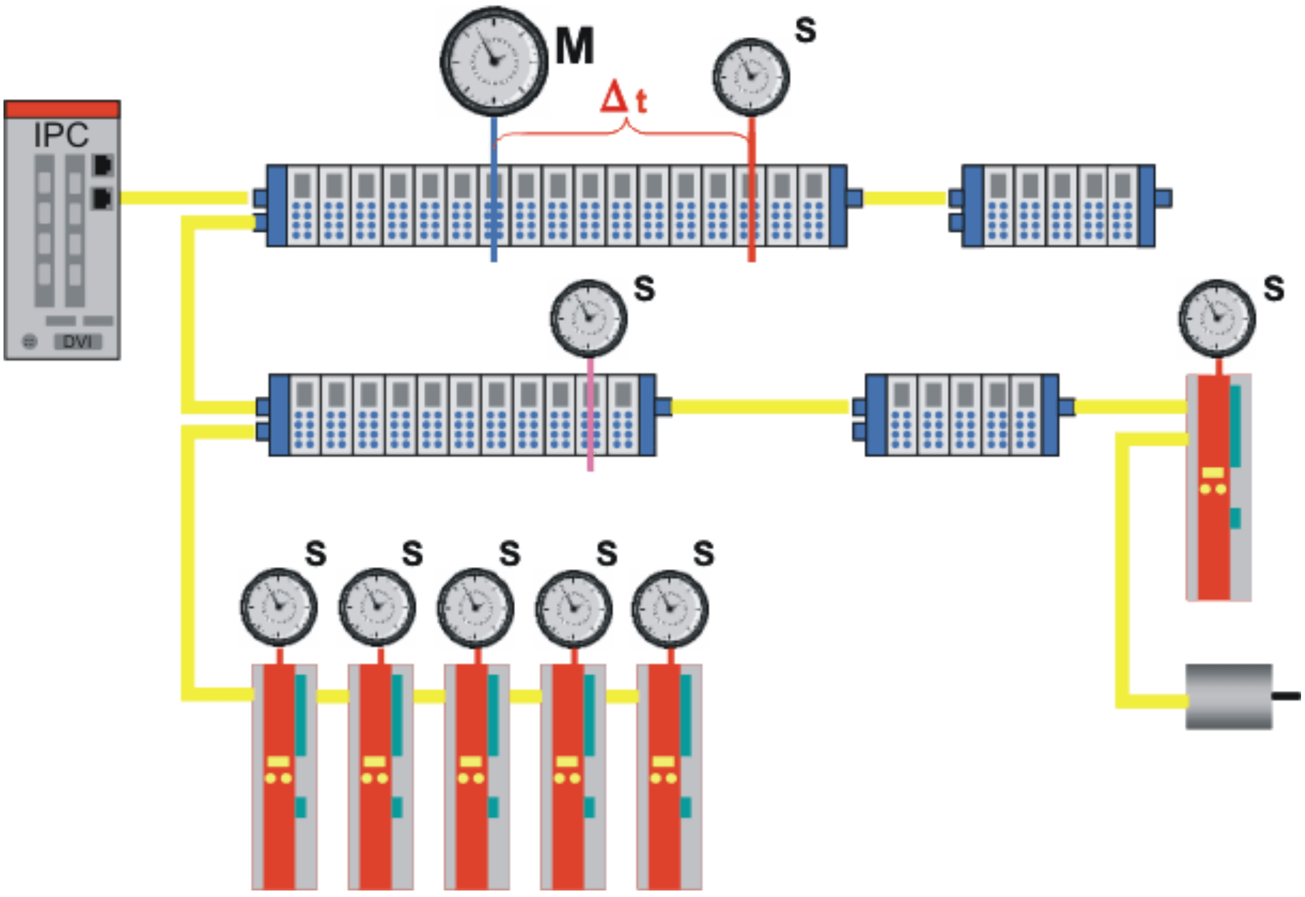
In Fig. *Distributed Clocks in the EtherCAT-System*, the reference clock (M) is the first slave in the EtherCAT strand with EtherCAT functionality after the EtherCAT master IPC. Since each slave causes a small delay – both in the device (S) and in the transmission link – in both directions, the run times (Δt) between the reference clock and the respective slave clock must be taken into account for synchronization of the slave clocks. It is therefore not appropriate to simply copy the local time of the reference clock to all downstream slave clocks. For each slave a separate offset value should be calculated as a function of several parameters.

For measuring the offset times the EtherCAT master sends a broadcast read datagram to a special address in all ESCs during the startup phase that causes each slave to record the time when the telegram is received (based on its local clock) in both directions. The master then reads the stored times as a basis for the calculation. These measuring cycles take place several times for all EtherCAT slaves. This enables the EtherCAT master to create a very precise map of the topology in relation to the frame delays between the EtherCAT slaves.

The actions described above ensure synchronous operation of all distributed clocks in an EtherCAT network, e.g. in a production plant, and enable relative times to be specified with high precision within the application. The absolute reference to global reality is provided via the *master clock*, which is usually based on a global world time format (DCF77, GPS, Internet time server, ...) or another time that is designated as valid across the system (network server, PC clock, BIOS clock, IEEE1588, ...). A Beckhoff TwinCAT EtherCAT master starts up using the local PC clock as master clock in order to initialize the reference clock. During system startup and subsequent operation the reference clock (M) can be readjusted based on a master clock, either via direct contact between master clock and control system (e.g. network server) or through feeding into a special EtherCAT slave (e.g. Beckhoff EL6692, EtherCAT bridge terminal).
This higher-level master clock does not violate the primate of the reference clock, since short-term synchronization in the sub-millisecond range, which is critical for real-time control, is solely determined by the reference clock, while synchronization of the complete EtherCAT application, including the control PC with the master clock, is based on longer intervals.

The following effects must be taken into account by the distributed clock control in the EtherCAT master:

- Offset compensation of each slave relative to the reference clock. After system startup the local clocks may start with different start values.
- Offset compensation of the reference clock relative to the master clock. To be taken into account during system start-up.
- Propagation delay measurement/Measurement of the offset times - depending on the number of devices, cable lengths, dynamic changes in the configuration, etc.
- Drift compensation/Drift correction - Each slave clock usually has its own source (quartz, PLL, ...), which means that offset times do not remain constant over a prolonged period (minutes, days). Drift correction deals with this irregularity.



Distributed clocks in the EtherCAT system

If an EtherCAT slave is no longer supplied with synchronization datagrams, because of an interruption of the bus, it can still fulfil all tasks via its local clock. Distributed clock control is jerk-free, both during normal operation and when EtherCAT traffic is reinstated.

Summary of technical data

The distributed clock function in the EtherCAT slave controller (ESC) has the following characteristics:

- Unit *1 ns*
- Universal zero point *1.1.2000 00:00*
- Scope up to *64 bit* (sufficient for 584 years). However, some EtherCAT slaves only support a 32 bit scope, i.e. the register overflows locally after approx. 4.2 seconds and starts again at 0.
- The EtherCAT master automatically synchronizes each local clock with the reference clock in the EtherCAT system with a precision of < 100 ns, irrespective of the distance between individual EtherCAT slaves
- When the EtherCAT system starts up the EtherCAT master usually takes the current time from a master clock, e.g. the hardware-based BIOS clock of its own PC, thus establishing the time reference to the current world time. This time is loaded into the selected reference clock when EtherCAT starts up, which then usually automatically maintains the time based on its local clock generator. Continuous synchronization of the reference clock with the master clock is possible, if required.
- The effective step size of the local clock is generally 10 ns. The remaining digit ("ones") is used for controlling the distributed clocks.

Up to now the distributed clocks function in the ESC was considered without interaction with the environment. Based on this local/global time additional functions can now be realized in the EtherCAT slave.

Optional additional ESC functions

The distributed clock time, which is synchronized with high precision, is used by the ESC for responding to specifiable times or signals from outside the ESC via a capture/compare unit. The ESC characteristics are defined by the configuration of the EtherCAT slave during startup and can usually not be changed by the user.

Action based on specified time: Compare - Sync0/1

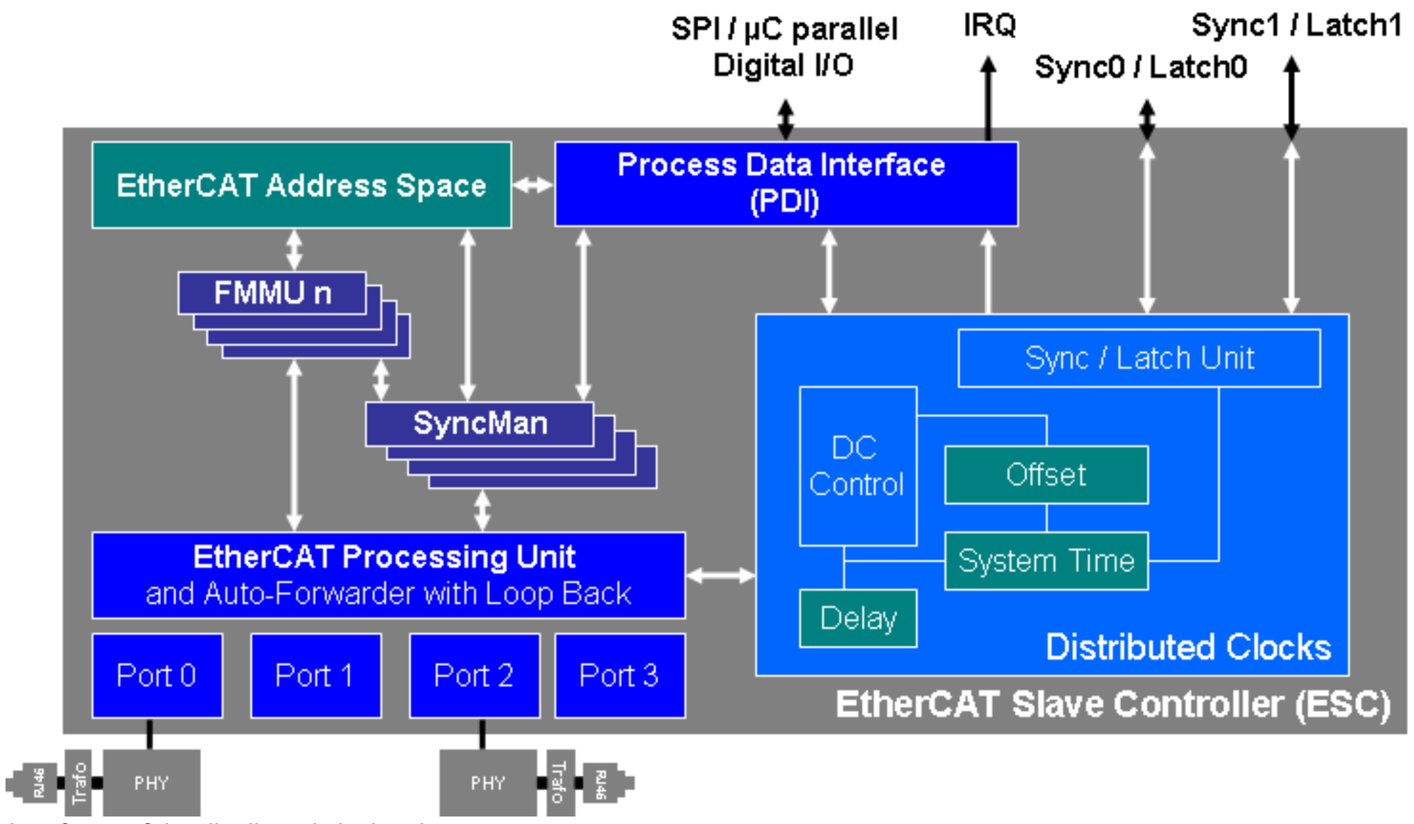
The distributed clock unit in the ESC usually features 2 interrupts that can be triggered time-controlled. These interrupts are referred to as *SYNC0* and *SYNC1*. In this case the compare unit in the ESC would be active: If the local distributed clock time matches a user-defined standard time the ESC triggers an interrupt and the associated processes. The standard time can be set *once*, which leads to a *one-off* action in the ESC. This option is used in Beckhoff timestamp terminals, for example.
The ESC may also automatically load new default values, which in this case leads to a cyclical sequence of ESC actions. This option is used for Beckhoff oversampling terminals, for example. The configuration data loaded on ESC start-up from a slave EEPROM is parameterizable, for example, can be used to determine which action an ESC will carry out if a SYNC0/SYNC1 signal is encountered. For example, it can write output data, read input data, or initiate communication with a connected microcontroller.

Reaction to an external signal: Capture - Latch 0/1

If an ESC is configured accordingly it can store the current local time if an external event occurs, i.e. it can place it into a buffer without delay using a capture unit. Examples for such external events are: arrival of the EtherCAT frame, end of the EtherCAT frame, edge on a dedicated pin of the ESC, communication with a connected microcontroller, and a wide range of other options.

Connection to an external logic - SPI/µC parallel/IO/IRQ

An ESC is to be used not only as a stand-alone unit, but also has interfaces for communicating with other electronic units. These could be, for example, a controller that controls a power drive or the evaluation electronics of a rotary encoder; see Fig. *Example for ESC functionality and embedding in a typical EtherCAT slave.* Communication via these interfaces can also take place under distributed clocks control. The position query to a rotary encoder electronics, for example, is thus chronologically equidistant in the ns range.



Interfaces of the distributed clock unit

Fig. *Interfaces of the Distributed Clocks Unit* shows a schematic of the distributed clocks unit with its interfaces and the interaction with the EtherCAT bus.

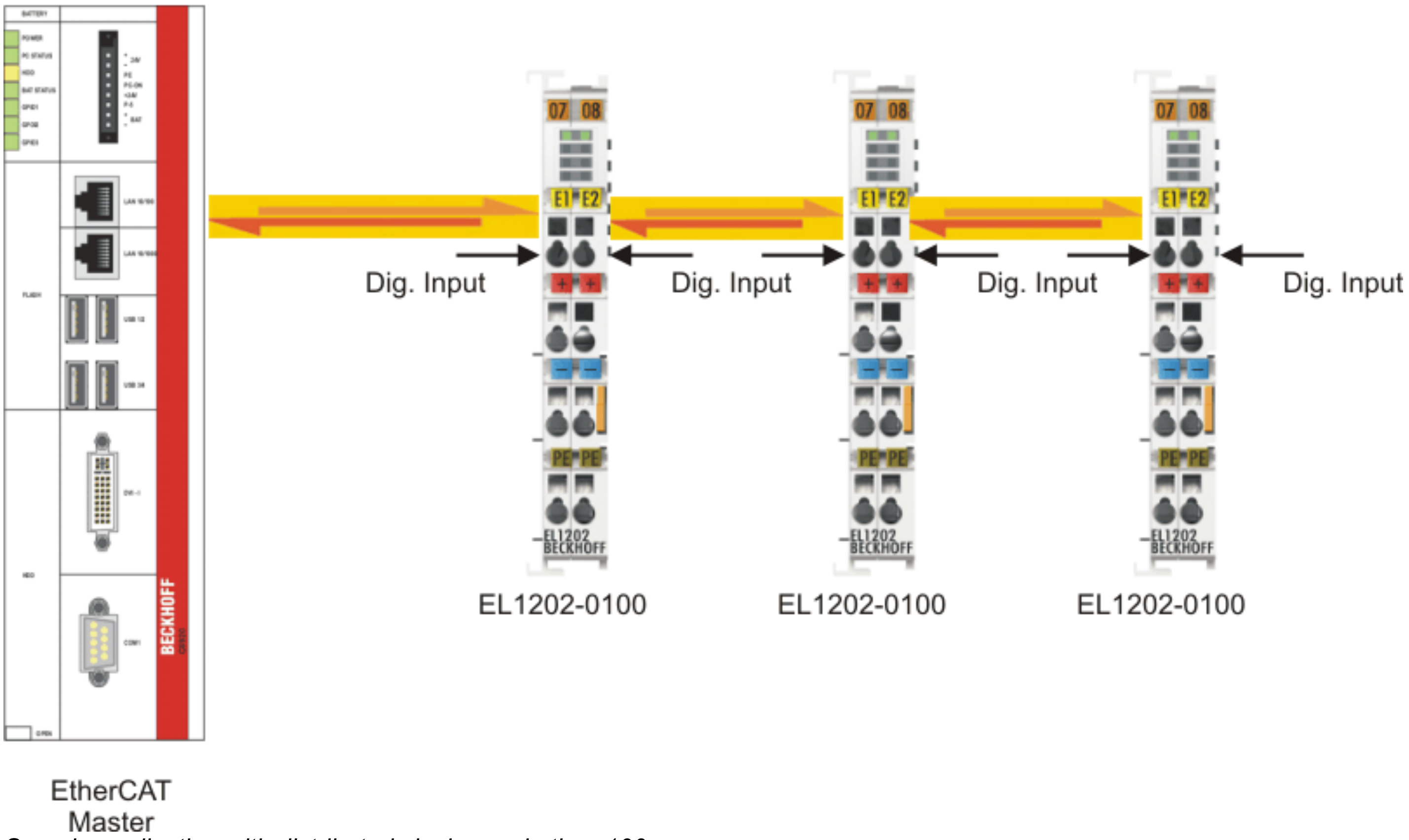
Application of distributed clocks and synchronicity with the control system in the PC

A distributed clock network ensures that all EtherCAT slaves supporting this feature are operated synchronously with a tolerance of < 100 ns. An application with input channels will be used as an example to illustrate this principle, followed by an application involving output channels.

The following example involving three Beckhoff EL1202-0010 devices illustrates synchronous generation of SYNC signals (2469118347.html#2470183947) for simultaneous reading of the inputs. Other ESC functions (latch signals (2469118347.html#2470185483), connection of external logic) are described in detail in the respective slave documentation.

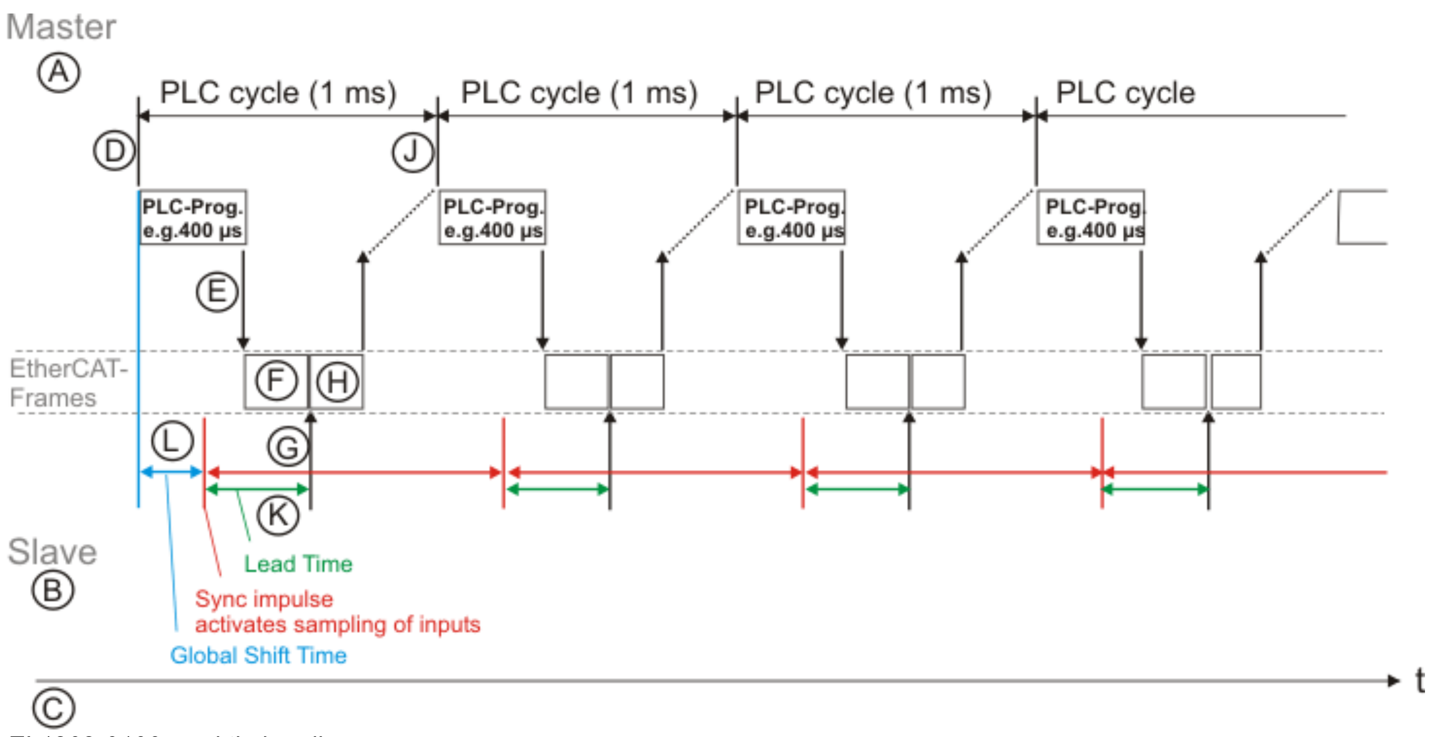
(2469118347.html#2470183947) (2469118347.html#2470185483)

Distributed clocks with input channels



Sample application with distributed clocks, cycle time 100 µs

In Fig. *Sample application with distributed clocks, cycle time 100 µs*, the Master PC, with its PLC on an EtherCAT master, drives an EtherCAT configuration with 3 EL1202-0100 digital two-channel input terminals with distributed clocks support (distributed clocks support is achieved through conversion of the EL1202 to the EL1202-0100 in the System Manager; see the corresponding documentation). It is irrelevant which EtherCAT slaves are used and what the distances are between the slaves. The controlling PLC and therefore the EtherCAT fieldbus are operated with a cycle time of 1 ms, i.e. inputs of the EL1202-0100 input terminals are queried at 1 ms intervals. In the EL1202-0100 the local distributed clock is used for sampling the two input channels: With each SYNC interrupt the ESC directly reads the current input value (0 or 1). When the EtherCAT frame passes the terminals shortly afterwards, each EL1202-0100 stores its two bits at the prescribed position in the frame. The distributed clock ensures that input sampling is highly consistent based on the same interval with a tolerance of < 100 ns. In addition, by default *all* EL1202-0100 in the whole EtherCAT network read their inputs at the *same* global time, irrespective of their configuration. In a timing diagram the situation looks as follows:



EL1202-0100 read timing diagram

In Fig. *EL1202-0100 read timing diagram*, the sequence of 4 complete I/O cycles is illustrated taking an EL1202-0100 as an example (cf. Fig. *Sample application with distributed clocks, cycle time 100 µs)*. Key to the processes within such a cycle:

- **A, B:** Fig. *EL1202-0100 read timing diagram* is subdivided in grey into a master and a slave side – the transport layer of the Ethernet frame moves in between them.
- **C:** The time axis in x direction enables temporal correlation of the actions described in this section.
- **D:** The PC's real-time clock starts a new processing cycle with a cycle time of 1 ms, for example. In the example the calculation takes 400 µs. The specified program code calculates the output process image based on the current input process image.
- **E:** After the calculation new output data are available for the fieldbus. The EtherCAT frame (or EtherCAT frames) is sent with the process data. Depending on the data quantity the length of an Ethernet frame may be between 7 and 128 µs. It can therefore take some time until it has been transferred to the fieldbus, has passed through all slaves, and was received back by the network card.
- **F:** The EtherCAT frame now passes through all EtherCAT slaves upstream of the current EL1202-0100.
- **G:** Depending on the number of EtherCAT slaves the frame passes the EL1202-0100 after a certain delay (several µs) in order to retrieve the input data (more precisely: it is dynamically processed by the ESC). These data must now be available for collection.
- **H:** The frame then passes through all further slaves until the complete frame is received back again at the network port of the PC.
- **J:** When the next PLC calculation cycle is started by the real-time clock, current input data from the fieldbus are therefore available.

The processes taking place in the EL1202-0100 used as an example are described in more detail below:

- **K:** So that the input data transferred to the frame at (G) is available in the ESC in time to enter the frame, the inputs must be read before the expected frame passage by means of an appropriate *lead time (green)*. Otherwise old data would be included in the frame where applicable. Reading of the physical inputs is triggered by the distributed clocks unit in the ESC through the SYNC signal.
  This time can be regarded as a safety margin: The shorter this safety margin, the more current the input data. If the chosen time is too short, the input data may no longer be available because the EtherCAT frame may reach the terminal too early, in which case no new process data are coupled into the frame! This lead time is therefore calculated

automatically by the EtherCAT master as standard.

- **L:** From the perspective of the master with its own real-time clock the SYNC pulse locally for the terminals therefore occurs with a *global shift time* after the real-time tick. This global shift time is taken into account once when EtherCAT starts up, after which the real-time tick in the PC and the SYNC pulses in the EtherCAT slaves operate with a constant cycle time (in this case 1 ms).
- The Beckhoff TwinCAT EtherCAT master automatically calculates the global shift time based on boundary conditions such as configuration, max. program execution time, cycle time etc., to ensure that operation of all EtherCAT slaves is as safe as possible and as current as possible.
- The user can modify the automatically calculated global shift time by applying a *manual shift time* both at the global level for all EtherCAT slaves and at the local slave level for each individual EtherCAT slave.

Correction of the PC real-time

Two time periods work alongside each other in Fig. *EL1202-0100 read timing diagram*: the PC with its real-time-tick *and* the distributed clock system with its distributed clocks in the EtherCAT slaves and the EtherCAT master (which is nevertheless located in the PC), as indicated by the grey dividing lines A//B.
The reason for the introduction of a second time base for the EtherCAT slaves with derived SYNC interrupts is the jitter with which the EtherCAT frames pass through the individual EtherCAT slaves. Due to the passage through the slaves, the potentially variable PLC runtime and the quality of the real-time tick the time when a frame is sent by the PC and when it actually passes a slave may vary in the µs range. If local slave events were started solely based on the passage of the communication frame, the actions in the slave would be variable. For high-precision applications this is not tolerable.
Through the introduction of clocks that are controlled at local slave level each EtherCAT slave has a high-precision local time base with a tolerance of < 100 ns relative to all other clocks, thereby improving the quality of all time-based actions by several orders of magnitude.

However, due to the two different time bases the "field time" (distributed clocks/system time) would deviate from the "PC-time" (operating system/BIOS clock) after a short time, with the deviation increasing continuously.

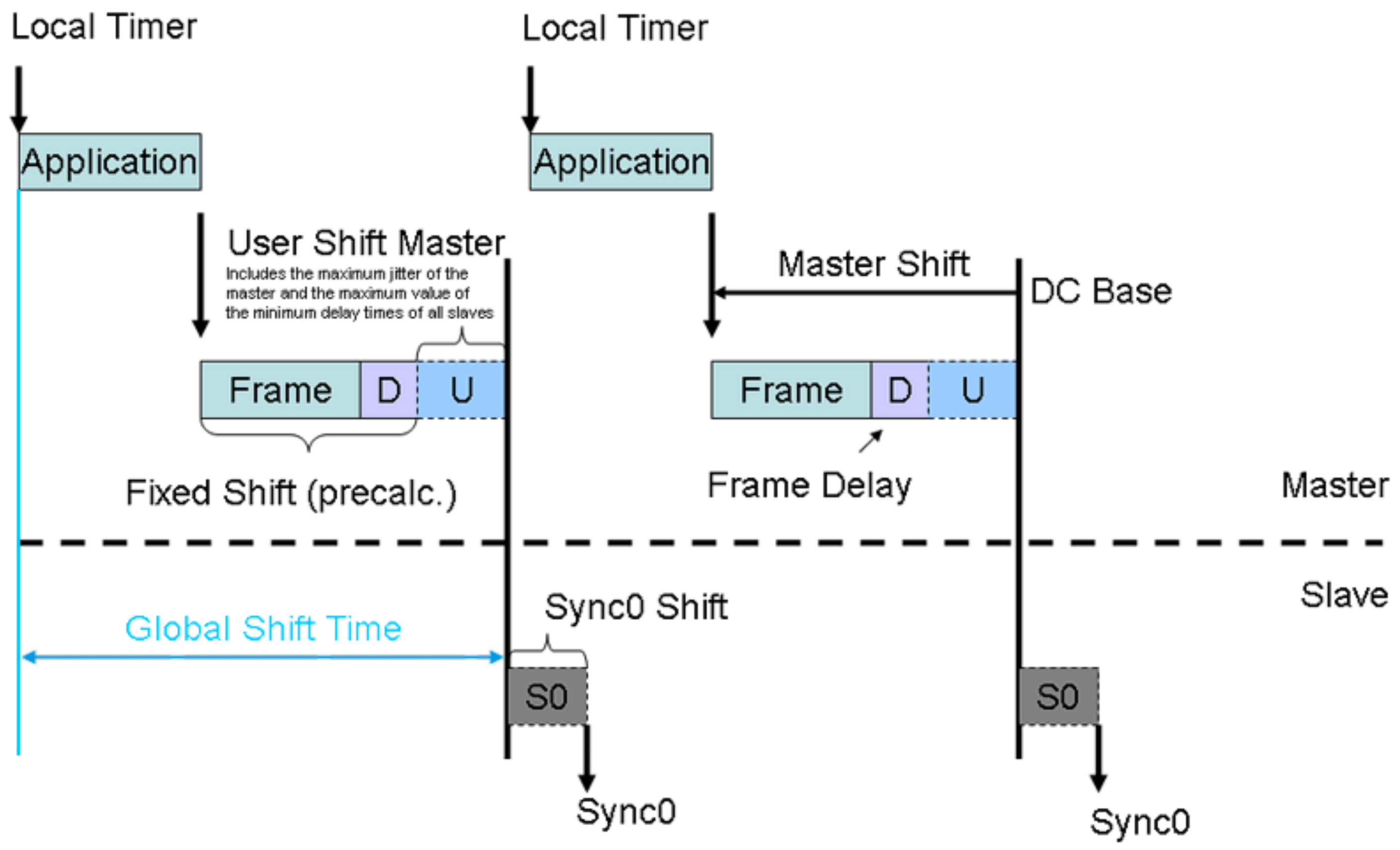
The TwinCAT EtherCAT master therefore intervenes in the PC clock and cyclically adjusts it based on the EtherCAT reference clock in order to ensure that the real-time tick of the control system and the SYNC interrupts locally for the terminals are synchronous with the set/calculated global shift time (Fig. *EL1202-0100 read timing diagram*, blue).

Synchronization with external master clock

In order to ensure long-term synchronicity of an automation system controlled in this way with an external reference clock (e.g. world time or local network time), suitable EtherCAT slaves must be used that relay the master clock time to the EtherCAT master, or the master clock must be connected directly to the control system. The EtherCAT master then undertakes the required steps for maintaining a low control deviation between master and reference clock.

Calculation of global shift time

A blue "global shift time" is specified in Fig. *EL1202-0100 read timing diagram.* Its value specifies when – in relation to the overall system behavior of the ESC – a SYNC0/1 action, for example, starts in the slave. The global shift time is composed of various elements, some of which can be changed by the user; see Fig*. Time specification of the SYNC pulse for the local slave with regard to the real-time tick of the master PC.*



Time specification of the SYNC pulse for the local slave with regard to the real-time tick of the master PC

The elements of the global shift time according to Fig. *Time specification of the SYNC pulse for the local slave with regard to the real-time tick of the master PC* are:

- **Application**
  The maximum expected calculation time for the program code (cannot be modified).
- **Frame**
  The length of the Ethernet frame (7..128 µs, perhaps several frames) (cannot be modified).
- **D**
  Total delay caused by the frames passing through the EtherCAT slaves, including jitter calculation.
- **U**
  User shift master: shift time predefined with default values by the EtherCAT master depending on the component (can be modified by the user).
  Input and output modules have different values
- **S0**
  User shift time in the slave - usually 0, although in some EtherCAT slaves values <>0 are predefined, can be modified by the user (see respective slave documentation)

Since the elements listed above can be summed to form the global shift time, it can be useful to enter *negative* values in the respective dialog – for instance in order to use a negative SlaveSync0 shift time to trigger the reading of input signals earlier than is foreseen as standard by the EtherCAT master (see Fig. *Time specification of the SYNC pulse for the local slave with regard to the real-time tick of the master PC*).

Note

Attention! No plausibility check takes place!

The mentioned notes and information should be used advisedly. The EtherCAT master automatically allocates settings that support reliable and timely process data acquisition.
User intervention at this point may lead to undesired behavior.

If these settings are changed in the Beckhoff TwinCAT System Manager, no plausibility checks are carried out on the software side.
Correct function of the EtherCAT slaves with all conceivable setting options cannot be guaranteed!
Unless specified otherwise in the associated slave documentation, we strongly advise against changing the automatic settings.

Distributed clocks with output channels

The logic used for the output channels is essentially the same as that for the input channels. The only difference is that the local slave SYNC clock usually takes place *after* a data-carrying EtherCAT frame has passed, instead *before* the frame, as is the case with the input channels. Therefore the calculation of the global shift time specified above (Fig. *EL1202-0100 read timing diagram,* blue) takes place separately for input and output slaves. The global shift time determined in the standard way for the group of output modules is thus larger than that calculated for the group of the input modules. However, both lie within the useful range of 0 – 100% of the EtherCAT cycle time.

In the following sections, access to the distributed clocks settings from the TwinCAT System Manager will now be explained.